Up and Running Google AutoML and AI Platform

Building Machine Learning and NLP Models Using AutoML and AI Platform for Production Environment

NAVIN SABHARWAL
AMIT AGRAWAL

Up and Running Google AutoML and AI Platform

Building Machine Learning and NLP Models Using AutoML and AI Platform for Production Environment

Navin Sabharwal
Amit Agrawal

www.bpbonline.com

FIRST EDITION 2021

Copyright © BPB Publications, India

ISBN: 978-93-88511-926

Distributors:

BPB PUBLICATIONS
20, Ansari Road, Darya Ganj
New Delhi-110002
Ph: 23254990/23254991

DECCAN AGENCIES
4-3-329, Bank Street,
Hyderabad-500195
Ph: 24756967/24756400

MICRO MEDIA
Shop No. 5, Mahendra Chambers,
150 DN Rd. Next to Capital Cinema,
V.T. (C.S.T.) Station, MUMBAI-400 001
Ph: 22078296/22078297

BPB BOOK CENTRE
376 Old Lajpat Rai Market,
Delhi-110006
Ph: 23861747

To View Complete BPB Publications Catalogue Scan the QR Code:

Published by Manish Jain for BPB Publications, 20 Ansari Road, Darya Ganj, New Delhi-110002 and Printed by him at Repro India Ltd, Mumbai

www.bpbonline.com

Dedicated to

The people I love and the God I trust.

—Navin Sabharwal

My family and friends.

—Amit Agrawal

About the Authors

Navin Sabharwal is an Innovator, Thought Leader, Author and Consultant in areas of AI and Machine Learning, Cloud Computing, Big Data Analytics, Software Product Development, Engineering, and R&D. He has authored books on technologies such as GCP, AWS, Azure, AI and Machine learning systems, IBM Watson, chef, GKE, Containers, and Microservices He is reachable at **Navinsabharwal@gmail.com**.

Amit Agrawal holds a master degree in Computer Science & Engineering from MNNIT (Motilal Nehru National Institute of Technology, Allahabad), one of the premier institutes of Engineering in India. He is working as a principal Data Scientist and researcher delivering solutions in the fields of AI and Machine Learning. He is responsible for designing an end-to-end solutions and an architecture for enterprise products. He is reachable at **agrawal. amit24@gmail.com**.

About the Reviewer

Riya Naval is a Senior Data Scientist practicing the latest AI Technologies. She is responsible for designing, developing, and delivering end-to-end solutions based on AI. She is reachable at **https://www.linkedin.com/in/riya-naval-010ab2b4**.

Acknowledgements

To my family, Shweta and Soumil, for being always there by my side, for letting me sacrifice your time for my intellectual and spiritual pursuits, and for taking care of everything while I was immersed in authoring this book. This and other accomplishments of my life wouldn't have been possible without your love and support. To my mom and my sister for the love and support as always; without your blessings, nothing is possible.

To my coauthor, Amit, thank you for the hard work and quick turnarounds you took to deliver this book. It was an enriching experience, and I look forward to working with you again soon. To Ayush and Pallavi for making this publication happen.

To all my team members who have been a source of inspiration with their hard work, their ever-engaging technical conversations, and their technical depth. Your always flowing ideas are a source of happiness and excitement every single day. To Piyush Pandey, Sarvesh Pandey, Amit Agrawal, Vasand Kumar, Punith Krishnamurthy, Sandeep Sharma, Amit Dwivedi, Gaurav Bhardwaj, Nitin Narotra, Shakuntala, Divjot, and Vivek, thank you, for being there and making technology a fun.

To all my other coauthors, colleagues, managers, mentors, and guides, in this world of 7 billion, it was a coincidence that brought us together, but it has been an enriching experience to be associated with you and learn from you. All the ideas and paths are an assimilation of conversations that I have had and experiences that I have shared. Thank you.

—Navin

To my parents, brothers, and wife, Riya, for being always an inspiration for me.

To my coauthor, Navin, thank you for your guidance and feedback.

To my colleagues, Ayush Agarwal and Pallavi Aggarwal, thank you for your technical support.

— *Amit*

Thank you, goddess Saraswati, for guiding me to the path of knowledge and spirituality.

Asato Ma Sad Gamaya, Tamaso Ma JyotirGamaya,
Mrityor Ma AmritamGamaya

Lead us from ignorance to truth, lead us from
darkness to light, lead us from Illusion to Reality

Preface

Google AutoML and AI Platform are power tools offered by Google Compute Platform for developers to create a compelling AI and Machine Learning applications with ease. Machine learning and deep-learning models are getting mature and accurate with a large amount of data that is being used into these models every day. With easier availability of out of the box functionality on a cloud platform, more and more developers are shifting their focus on developing analytics and machine learning based applications on the cloud platforms.

Google's AI technologies are continuously learning through enormous amounts of data such as Wikipedia Articles for Question–Answering Systems, Text Classification Systems, and so on.

The trickiest parts of developing and testing machine learning models and selecting the best ones that work in a particular use case are simplified by GCP for the developer community.

With the advent of Google's Architecture search algorithms that enable AutoML to automatically search and find the best model for your dataset, it is possible for a developer to design AI systems by providing an abstraction over data scientist activities such as Data Analysis, Model Identification, Training, Evaluation, and so on.

Google AutoML is one of the services from Google Cloud Platform that makes it easy for the developers to design and build machine learning models and use them across various applications. As an example, Text Classification can be used for Intent identification while designing conversation systems.

This book is relevant for programmers who have an interest in designing and developing the AI-based systems. It also acts as a guide for the development of AI systems using Google AutoML and AI Platform. It also enables you to design an end-to-end machine learning pipeline using Google's BigQuery, Dataprep, and Dataproc. This book provides a step-by-step guidance to the readers that helps them to build machine learning and Natural Language Processing models. This book will help the readers through practical examples as well as challenges and best practices.

Downloading the code bundle and coloured images:

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/dcca3

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Table of Contents

Chapter 1
Introduction to Artificial Intelligence

The digital revolution in the 1980s brought with it a wide spectrum of options, where machines or computers could ease out the life of humans in ways that had not been imagined earlier. Using machines for different tasks by programming them, made them problem-solvers in all sorts of domains that were earlier reserved for specific tasks. This ease of programmability also opened the doors for self-programming or rather Intelligent Systems and, hence, the idea of Artificial Intelligence (AI) got started.

With Artificial Intelligence, several human characteristics of "**Thinking, Decision-Making, Trends' Analysis, and Problem-Solving**" are being done by computer systems or integrated machines. Artificial Intelligence has now created a mark in almost every arena where smart or intelligent solutions can be integrated. Machine Learning and Artificial Intelligence algorithms are helping computing machines to adopt to human-like intelligence and decision-making skills in tasks like medical diagnosis, proving mathematical theorems, search engines, and voice recognition.

Structure

In this chapter, we will cover the following topics:

- Machine Learning
    - o Supervised Learning
    - o Unsupervised Learning
    - o Reinforcement Learning
    - o Applications of Machine Learning
- Natural Language Processing
    - o Pre-processing
    - o Parts of Speech (POS) Tagging
    - o Entity Extraction
    - o Sentiment Analysis
    - o Document Summarization
    - o Topic Modeling

Objectives

After studying this chapter, you should be able to:

- Understand the basics of Machine Learning and Natural Language Processing
- Understand the applications of Machine Learning and Natural Language Processing

Machine Learning

Machine Learning can be easily defined as a branch of Artificial Intelligence that deals with self-training or programing and adapting according to the given data that speaks of the problem statement. The problem statements can range from mere identification of entities to a tag or class to decision-making if the given input favors an expected output. Some algorithms can help one identify if a fruit is Tomato or Persimmon, which are pretty similar looking; such algorithms classify an entity in different classes. Another example can be predicting the stock market trends based on historical behaviors and real-time variables.

The major difference from traditional computer software is that the developer does not write the code that instructs the system the way to tell the difference between the two objects. Instead, an algorithm has been taught the way to generate these decisions by being trained on an oversized amount of data. One thing that has to be understood is that if any Machine Learning Algorithm is a solution for a problem statement, a relevant and considerably sized dataset is always a prerequisite to start with.

Machine learning is categorized into three main types, namely, Supervised Learning, Unsupervised Learning, and Semi-Supervised Learning. We are going to look at an overview of these categories in this chapter.

Supervised Learning

Supervised Learning is a term used for algorithms where we have training data with accurate labels against them. The significance of such algorithms is the accuracy of labeled data or prediction as input helps in having crisp decision boundaries with the training of algorithm. The labeled data provides the guidance or knowledge to the system to understand and map the data. The labeling of the data is done by human subject matter experts.

Another advantage with having a labeled dataset is when the training dataset trains the algorithm, the testing dataset can help in better accuracy judgment, as error can be easily identified and propagated. The training data needs to be carefully chosen because an imbalanced training dataset can cause imperfect decision boundaries by straining them due to lack of proper examples. Hence, the training dataset should be similar to the final dataset in its characteristics and should provide the algorithm with the labeled parameters required. An imbalance in this scenario would mean much more records for a particular type and very less records for another type.

The algorithm then finds connections between the parameters provided and essentially establishes a cause and effect relationship between the variables in the dataset. At the end of the training, the algorithm develops a draft of how the data works and the relationship between the input and the output.

Another important factor is to identify important features in the training data because irrelevant features can possibly influence the training of the algorithm. A few examples of supervised learning are regression, classification, and more.

Unsupervised Learning

Having a labeled dataset may not be possible in many cases. There is a cost of getting labeled data as well, as humans need to label the data. In the problem statements, where such constraints cannot be met, Unsupervised Machine Learning is the solution.

Unsupervised machine learning holds many advantages by being able to work with unlabeled data. Firstly, human effort is not required to make the dataset machine-readable that allows much larger datasets to be used by the program. Also, with the unlabeled data the algorithm has a better scope to draw conclusions and relationships in the feature set available. Another huge advantage can be its usage in real-time data, where input is analyzed and labeled with the learners.

The labels generated in the unsupervised learning work differently from the labels in supervised learning. In supervised learning, the decision boundaries have to be intently drafted based on the labels, whereas in unsupervised learning, the labels are an evolving space that can get updated with the drift of decision boundaries at a given time based on the hidden correlation it deduces.

The creation of these hidden structures makes the unsupervised learning algorithms versatile. Instead of a defined and set statement, the unsupervised learning algorithms can adapt to the data by dynamically changing hidden structures. These offer more post-deployment development than the supervised learning algorithms.

Clustering is one of the famous unsupervised learning algorithms, to split the data according to similarity, clustering can also help in identifying the anomalies in our dataset.

Reinforcement Learning

Reinforcement Learning is another Machine Learning domain that solves problem statements that are more of a path traversal rather than definitive output. Games and robotic movements can be a great example of using Reinforcement Learning. Tracing a set of paths such that system would provide reward for every correct path and penalize for every wrong path.

Reinforcement learning is all about learning and inspiring from experience and interaction. It features an algorithm that improves upon itself and learns from new situations like trial and error methods.

The rules of the game are not suggestive of the steps but rather a system learning of what kind of actions are a reward and what is the penalty. Such rules that are called **Policy** are often the most important part of such algorithms, where an agent is interacting with an environment. At each time step, the agent receives the environment's current state, and the agent must choose an appropriate action in response. After the agent executes the action, the agent receives a reward and a new state. Basically, each iteration of the agent is analyzed by the interpreter to judge whether a penalty or reward is called for.

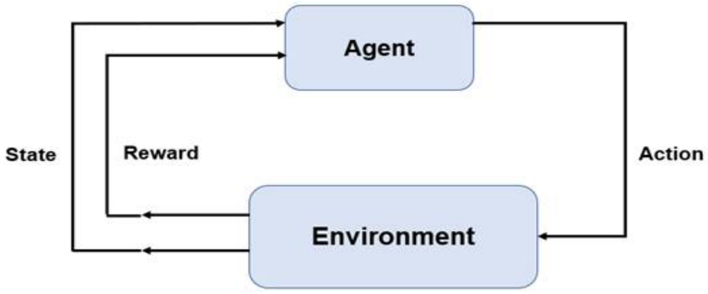


Figure 1.1: *Reinforcement Learning*

Applications of Machine Learning

Adaptability, versatility, and effectivity are the key features to sell any Machine Learning algorithm. The dynamic and high-powered nature of the versatile Machine Learning solutions is a feature utilized in situations where the answer is required to continue improving post-deployment.

Machine Learning algorithms and solutions are versatile and can be used to support and scale business teams. Be it image recognition, weather predictor, spam email filtering, fraud detection, and so on, Machine Learning has a huge scope in solving such problems. Cognitive assistants or virtual chatbots are another such example. Making any customer interaction a personal experience can significantly improve a business, so these chatbots analyze customer queries, preferences, and short context to curate an instant preset answer to assist the customer.

The customer profiling is another important application of Machine Learning. Streaming services like Netflix or Hulu curate a "You may also like this," where they push suggestions based on the profile they have analyzed per customer. Finding items relative to each customer can be based on n number of parameters. The streaming services often consider the likes and interests of customers, whereas ecommerce services often curate "You may also like this" mostly based on the common preferences of similar customer profiles or rather the customers with similar wish lists or shopping patterns. Social Media on the other hand, curate advertisements, and news-feeds based on tentative likeability and revisits to certain domains of information.

Natural Language Processing

Natural Language Processing (NLP) is an important part of Computer Science in **Artificial Intelligence (AI)** that deals with how computers analyze, understand, and derive meaning from human language in a smart and practical way. Thus, enabling computers to understand and process human language.

With the help of NLP, we can build models and processes to analyze text. NLP has its application in numerous real-world scenarios like speech to text, language translation, text generation, summarization, sentiment analysis, topic modeling, text mining, automatic question answering, and so on.

Any language data, if deconstructed, is a mere stored stream of Unicode characters or entities for a system that consumes some physical space in bits and bytes all because machines do not understand any human language at the root level. NLP to a machine is bringing out sense in some stream of characters, purely by analyzing its constraints, rules, and patterns against some training data. The training helps in understanding of language syntax and semantics and generating complying information similar to that of the data that is trained.

Example: The word "ACT" has no isolated meaning by itself to a machine; it is not a keyword and clearly only consuming some memory. By analyzing English language documents and rules for its form and relative meaning to other words, we get that this English word "ACT" can have numerous word forms with several meanings in multiple contexts.

Each and every human language entity like the preceding example is holding a chain of information that makes NLP capable of many useful jobs like correcting grammar, converting speech to text, and automatically translating between languages.

NLP algorithms

Human languages are pretty complicated that make NLP difficult to learn and implement. In spite of that, NLP offers some techniques such as Word/Sentence Embeddings, Syntactic and semantic analysis, and so on that enables organizations to develop software that understands human language. Some of the heavily used NLP algorithms/steps are as follows:

- **Pre-processing:** The text that is received as human language often requires some sanitation, and there are several steps that can be incorporated.
    - o **Stop-words Removal:** Removes redundant characters and words of the text that hold little meaningful information.
    - o **Stemming and Lemmatization:** Converts words into their root form.
    - o **Tokenizer:** Extracts token or phrases out of text (document or sentence).
- **Parts of Speech Tagging:** Parts of Speech (POS) tagging is an NLP technique that identifies or extracts structure of a sentence. For semantic analysis, POS tags or Parts of Speech Tags can determine what entities or tokens hold importance across the text. As an example, for a sentence "There is a seminar on Neurons and Neural Networks in Sydney," the POS tagging for each of extracted tokens are as follows:

  "There(/EX) is(/VBZ) a(/DT) seminar(/NN) on(/IN) Neurons(/NNS) and(/CC) Neural(/NNP) networks(/NNS) in(/IN) Sydney(/NNP)"
- **Entity Extraction:** Named Entity Recognition (NER) is one of the most useful techniques in NLP that helps in extracting the entities from the text. It is generally based on grammar rules and supervised models. It highlights the essential concepts and references within the text. The NER labels classify atomic elements and important phrases within the sentence into categories like people, locations, organizations, dates, and

more. Many cloud vendors such as IBM, Amazon, Google, Microsoft, and more provide NER feature as a service such that same can be leveraged in any application or product without a need for development from scratch. The Entity Recognition feature provided by these vendors have been trained on general English text or document such as Wikipedia articles.. Therefore, these models work best for identification of generic entities such as Person, Location etc. However, Google has released AutoML Natural Language service that enables developers or data scientists to train custom entity model domain specific dataset provided by them. It may be possible that token or phrase can have different meaning in different domain or context, and, hence, refers to different entities. As an example, if you run entity recognition system over a sentence that has been trained on different domains then you would see different results as follows:

Sentence: "There is a seminar on Neurons and Neural Networks in Sydney",

- **Generic NER:** There is a seminar (\ Activity) on Neurons and Neural Networks in Sydney(\ Location)."
- **IT Domain NER:** There is a seminar (\ Activity) on Neurons and Neural Networks (\ Artificial Neural Networks) in Sydney(\ Location)."
- **Medical Domain NER:** There is a seminar (\ Activity) on Neurons and Neural Networks (\ Biological Neural Networks) in Sydney(\ Location)."

As you can see, if you use Generic NER, it won't recognize Neurons and Neural Networks as an entity. However, IT domain and Medical domain NER recognizes it as an entity, but entity type is different. For IT domain, Neurons and Neural Networks have been identified as Artificial Neural Network, but in Medical domain, it was recognized as Biological Neural Networks. Therefore, it is quite important to use entity recognition system that is close to your domain.

- **Sentiment Analysis:** Sentiment analysis/opinion mining uses NLP techniques to identify the context and subtle projection of "feelings" of a text. The sentiment or opinion can be the features that extract the emotion of the given text. Identifying this emotion on a scale of positive, negative, and neutral is the simplest categorization of sentiments;

the relative score is also generated on a scale implying the same. The number of emotions or sentiments can be more in numbers, likely emoting all human emotions as anger, sad, happy, overwhelmed, and more. A single data input can have an n number of sentiments to be deduced. Keeping a mark of all such emotions, connotations, and sub-text is essential to sketch out the dominant sentiment and, hence, generate a score accordingly. The statement "Abandonment of Chernobyl has proved to make the place far more beautiful" has multiple sentiments in reflection; abandonment specifies a negative connotation even when the beauty of a place is being discussed which is a positive sentiment, and, hence, the score generated will be a factor of both sentiments. Strong unilateral sentiments tend to generate better score predictions as shown in the following examples:

For example:

- **(Negative statement):** The staff members in this hotel are extremely rude and totally ignorant.

  Sentiment Score: -1.17232

- **(Positive statement):** The ABC Health insurance is affordable as compared with the other insurance companies.

  Sentiment Score: 0.19836

Sentiments are often a virtual feature that can extensively help in deducing the delivery and meaning of a text. Sentiment analysis can also help guide the text generation engines to be more appropriate and human like. As an example, refer the following two cases of a Cognitive Assistant and user interaction to see how sentiments can direct the Chat Bots' response.

- **Case 1:** Positive Sentiment

  User:- "Excellent Services provided"

  Chat Bot:- "Glad to hear that! Would you Like to give your feedback to boost our employees' morale?"

- **Case 2:** Negative Sentiment

  User:- "Worst Services ever, will never order from you people again"

  Chat Bot:- "Deeply regret about your experience. Please detail your issue in the feedback segment, so that we can help with issue redressal at the earliest."

- **Document Summarization:** Summarizers can greatly help in identifying crux information of a text. Summarization can be a very handy service in cases where large collection of documents is to be analysed. Summarization can be of the following two types: extractive and abstractive summarization. In **Extractive Summarization**, we pull out segments of text to generate the summary. In **Abstractive Summarization**, the summary of document is generated based on the important phrases, named entities and extracted topics and other feature sets that are provided for the given documents. Few algorithms that are used for text summarization are LexRank, TextRank, and Latent Semantic Analysis.
- **Topic Modelling:** Topic modelling is an NLP technique that is used to extract an important token or word or phrases out of a text (document or sentence). It is based on an unsupervised machine learning method where collections of documents can be provided, and it can extract important topics out of it. It has major applications in Question–Answering Systems to find out answers with relevant information and in Document summarization to generate summary that should contain tokens or words or phrases that defines crux of document. Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA),and so on are some of the topic modelling algorithms.

With the advent of technology, implementation of the aforementioned NLP tasks are quite feasible now. Even some of these capabilities are available as a service that can be leveraged by any application or system. In the next few chapters, we will see in detail how Google AutoML and AI Platform can help us quickly put together AI solutions to different types of business needs and create compelling products using the power of Google AutoML.

Conclusion

In this chapter, we have discussed about basics of machine learning and Natural Language Processing (NLP) including Supervised and Unsupervised learning, Natural Language Processing, Sentiment Analysis, Document Summarization, Entity Recognition, and so on. In the next chapter, we will deep dive into Google AutoML services.

Chapter 2 Introducing the Google Cloud Platform

Today, Artificial Intelligence is everywhere from social media sites, shopping websites, to content providers. From being used in state-of-the-art systems built by Silicon Valley companies, AI and Machine Learning are now being leveraged by most products and solutions across the globe. In some cases, it is quite easy to develop and build machine learning capability in these systems, and sometimes, it is very difficult to do it due to lack of skilled resources on Machine Learning. There is a general dearth of resources in the areas of Data Science and Machine Learning in the industry today.

However, Google came up with an innovative solution where developers can also develop and build Artificial Intelligence capabilities in their products and solutions using Tensorflow and AutoML. AutoML enables developers to use AI algorithms as a service that allows them to integrate with any system. Developers can now provide data as per their business requirement and AutoML will automatically train the AI model for you. AutoML has completely revolutionized the way AI is being consumed by developers and architects. AutoML takes the complexity away while integrating and using Machine Learning in products and solutions.

In this chapter, we will go through the different Machine Learning services/APIs provided by Google Cloud Platform such as AutoML, AI Platform, Tensorflow, and so on.

Structure

In this chapter, we will cover the following topics:

- AutoML Natural Language and it's applications in Document Classification, Entity Extraction, and Sentiment Analysis
- AutoML Tables
- AutoML Translation
- AutoML Video Intelligence and Vision
- AI platform
- TensorFlow
- Creation of an account in GCP
- Creation of Google Cloud Storage Bucket

Objectives

By learning this chapter, we will be able to:

- Understand the basics of AutoML Natural Language, Video Intelligence, Vision, and Google AI platform.
- Understand GCP and Google Cloud Storage Bucket.

Introduction

AutoML, also known as **Automated Machine Learning**, is defined as a process of automated generation of Machine Learning or NLP (Natural Language Processing) models, thereby, reducing development efforts and time that is being spent in testing and validation. It provides easy-to-use interface to train, test, and validate Artificial Intelligence based models. This helps developers and Data scientists to build machine learning models with high efficiency, scalability, and productivity while preserving model quality.

Earlier development of machine learning models required people of the following different skillset:

- **Data Engineer:** Responsible for data collection and transformation as per business requirement

- **Data Scientist:** Responsible for data analysis, identification of models to be used as per business requirement, validation, and testing
- **Developer:** Responsible for development of module that will enable machine learning model to be consumed by other applications

As an example, if you are planning to develop a Spam blog classification system where it will flag blog as spam or not spam, an organization would need a Data Engineer, to crawl spam and not-spam blogs and transform them into features as per requirement of Artificial Intelligence algorithms and Data Scientist, to decide on type of AI algorithms to be used; validated, and tested. Apart from this, it also requires a developer to release this as feature to other systems such as Plugin in a user's browser such that whenever a user opens any blog, it should flag it as spam or not-spam.

Now, with AutoML, the developer gets most of the things as out of the box available through API calls and simple applications and use cases can be done without getting into much complexity. As an example, developers can leverage translation services from various cloud vendors such as IBM, Microsoft, and so on to enable language translation capability in their product or solution by using REST APIs without the need to develop language translation system from scratch.

AutoML offers various services such as AutoML Natural Language, AutoML Tables, AutoML Translation, and so on that can be leveraged to enable AI features in your applications as shown in *Figure 2.1:*

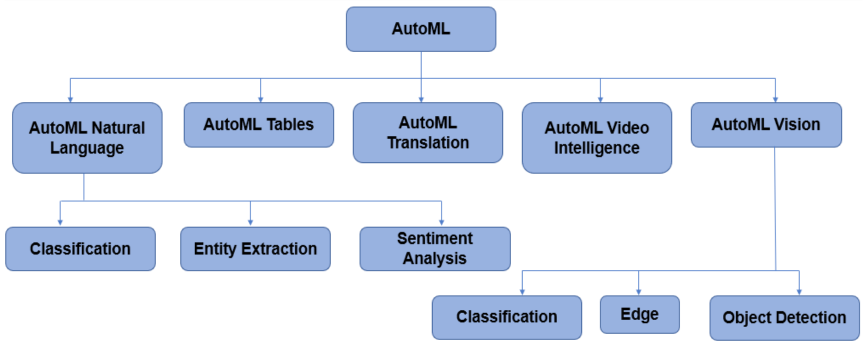


Figure 2.1: *AutoML Services*

Now, we will discuss each of these services in detail.

AutoMLNatural Language

AutoML Natural Language is a natural language service that provides capability to syntactically and semantically parse the content (text, sentence, or document) to generate knowledge out of it. The generated knowledge can be used in NLP-based applications or systems such as Question–Answering systems, Document Summarization systems, Search Engine, Recommendation system, and so on. Some of the NLP tasks such as stop-word removal, POS tagging, parse trees, Contextual grammars, dependency trees, and so on that have been used to perform pre-processing steps as a part of an NLP pipeline, now, can be implemented using this service without any significant coding effort.

There are two modes in which AutoML Natural Language can be used:

- **Out-of-Box Model:** In this mode, an organization can use out-of-box model provided by AutoML without a need to train the system with your own data. As an example, we can leverage Entity Extraction feature of this service to identify entities such as Place, Location, etc. out of the sentence without providing the training data.
- **Custom Model:** In this mode, an organization needs to provide their task specific data. As an example, if you want to implement a document classification system that would classify your documents into particular categories, then you should go with this mode.

AutoML Natural Language API provides the following important features that can be leveraged easily in any application.

AutoML Text & Document Classification

AutoML Text & Document Classification feature is used to classify content (text, sentence, or document) into relevant categories that can be defined during the training phase. It employs Google's state-of-art technology for Neural Architecture search to find out a relevant Neural Network architecture that will fit perfectly for your dataset. This feature can be used in tasks such as Document classification, Text classification, Intent Classification (for conversation systems), and so on.

AutoML Entity Extraction

AutoML Entity Extraction feature is used to extract entities such as location, place, contact number, and more from the text. AutoML enables you to provide your tagged data where each of the sentences should be tagged with relevant entities for the training. This in-turn will train custom entity extraction model where a Neural Network architecture will be identified using Google's Neural Network Architecture search technology. This feature can be used in NLP tasks such as Question Answering, Document Summarization, Conversation Systems, and so on.

AutoML Sentiment Analysis

AutoML Sentiment Analysis feature is used to identify sentiments from text or audio. This feature has been gaining popularity in design of conversation systems where responses to a user's question will be given depending on sentiment presented by the user via text or audio. This feature enables you to train your sentiment model over your own dataset that should have been tagged via Language Expert, or you can also leverage the out-of-box model provided by AutoML for sentiment identification.

AutoML Tables

Building an end-to-end machine learning pipeline is always a challenging and time-consuming task for structured datasets such as Stock Market Dataset. It includes lot of tasks such as normalization, type of feature it contains, column or feature encoding, missing value identification etc. Most of the efforts and time of data scientists and analysts is spent on this, as nature of data plays a major role for the selection of models to be used. As an example, for regression problems, if features are independent, then the problem is linear regression problem otherwise non-linear.

AutoML Tables eases design of end-to-end machine learning pipelines by providing an automated way to perform the following operations.

- Missing Value Computation
- Identification of type of values in feature
- Data Normalization
- Feature Encoding

Apart from this, it can automatically identify the best model for your dataset from Google's model repositories. It varies from traditional models like linear regression, Tree-based models, etc. to Neural Network based models for complex tasks. Due to this, the total time required for building production ready machine learning models has been reduced drastically. It provides an intuitive user interface where all of these steps can be performed and monitored. As an example, a developer or data scientist can see what all actions have been performed over the dataset and their results. This way, the developers or data scientists can ensure that the trained machine learning model is of high efficiency and accuracy. The following are two types of problems that can generally be solved using AutoML Tables:

- **Regression Problems:** A regression problem is a type of problem where final prediction will be numeric. As an example, if you want to predict share price of a particular company, then you would go with regression models, as predicted price will be a numerical value.
- **Classification Problems:** In classification problem, each of the data point should be classified into a particular category. As an example, in Spam Blog Classification System, each blog should be classified as a spam or not-spam. This type of classification system is known as a binary classification. If there are more than two classes, then it is known as multi-class classification. However, in some cases, each data point may belong to multiple classes at a time. This type of classification problem is known as a multi-label classification. As an example, the document may have content related to both sports and politics, in this case, the document will be classified into both sports and politics.

AutoML Translation

Machine or language translation has been gaining popularity day by day, as it enables an organization to build multi-lingual capabilities into their products and applications. The following are some of the examples of such systems:

- **Search Engine:** The search engine can understand a user's query in various languages and returns list of documents relevant to query. In this system, all the documents will always be in their base language, and if a user asks query in some other language, then language translator first detects language in which query was asked and then converts it to target language to fetch relevant documents.

- **Conversation System:** With the advent of the translation technology, now, it is very easy to build multi-lingual capabilities in conversation systems. Here, a user can ask query in various languages, and these systems can leverage the language translation service from Google as a REST API to convert it into base language to fetch relevant response from the conversation system.

AutoML Translation service from Google enables you to implement multi-lingual capability in multiple products or applications. Similarly, this works in two ways, namely, the out-of-box and custom model as follows:

- **Out-of-Box Model:** In this mode, an organization can use the out-of-box model provided by AutoML without the need to train the system with your own data. As, the out-of-Box model has been trained on the benchmark dataset; therefore, it works for most of the scenarios/implementations. But, sometimes, it may forget domain specific knowledge after conversion. As an example, if you are building closed domain multi-lingual conversation system that answers IT related questions such as "Facing an issue with Outlook," and a user enters this query in other language (say in German) then "Outlook" might be understood as Outlook of something and not Microsoft Outlook. In such a case, the conversation system won't be able to recognize it correctly as issues with Microsoft Outlook.
- **Custom Model:** In this mode, an organization needs to provide their domain specific data for training to recognize domain specific words during the conversion process. As an example, in our closed-domain conversation system for IT domain example, if we train custom translation model on IT related dataset then during conversion, the translation service would know that "Outlook" has nothing to do with outlook of something, but Microsoft Outlook is an email service from Microsoft. To generate the training set, this mode requires human language translators for required languages. It introduces extra cost that makes this mode expensive as compared with the out-of-box model.

These features of AutoML translation service enables an organization to either use the out-of-box model or even train the model over a custom dataset. This motivates them to decide upon which mode to use according to their budget and requirements. Google provides feature of AutoML translation service via REST API. It means it is very easy and convenient for an organization to integrate and use with their existing systems.

AutoML Video Intelligence

AutoML Video Intelligence provides a set of services to generate insight or knowledge out of the video. This knowledge can be used to develop a system such as Video search engine. This enables a user to enter the query and search over a corpus of videos that was unimaginable earlier. It provides two services that are as explained follows:

AutoML Video Intelligence Classification

AutoML video Intelligence Classification service classifies video segment or shot into a particular action. As an example, if you want to develop a classification system that can classify one of the shots or segments from the recorded video from any games such as Football, Cricket, Chess etc. into particular actions ("goal" for football or "check"), then you should go with this.

In this service, you would need to provide a video corpus along with associated actions to train it. Then, it will automatically identify which model to be used based on your data using Google's architecture search capability and train a model for you. It is a very important feature that lets you generate an insight out of the video. As an example, output of this system can be provided as training data to any search engine that can be retrieved later on if a user wants to see all videos or shots or segments containing goals from a Football game.

AutoML Video Intelligence Object Tracking

Object tracking is a sub-branch of computer vision that tracks position of an object over a different time frame; for example, tracking suspicious moving vehicle for security issues over a period of time. Nowadays, it has been used in surveillance for security purpose, monitoring of traffic to avoid congestion, and more. Google has released this feature as a service. It means anyone who has access to Google Cloud can leverage this feature by enabling the service from the cloud console.

AutoML Vision

AutoML Vision is a service from Google that provides image classification and object detection as a service. AutoML vision service enables us to train a custom model specific to your dataset. For example, imagine you have millions of images that have been tagged by human experts into various categories of animals such as Cat, Dog, and more, then you can use this tagged data to develop an image classification system for your application or product. In other words, it implements supervised machine learning to train the custom models.

Google has also released Vision API that uses the out-of-box model to detect landmarks, faces, optical character recognition, and more. This can be easily integrated with your application or website via REST API to enable these features.

AutoML provides four services which are Vision Classification, Vision Edge - classification, Object Detection, and Vision Edge -Object Detection.

AutoML Vision Classification

AutoML Vision Classification service classifies an object or image into defined categories that have been provided during the training phase. As an example, if you are developing an image classification system that classifies image into categories then you should go with this.

In this service, you would need to provide a corpus of images along with associated labels or category to train it. Then, it will automatically identify which model to be used based on your data using Google's architecture search capability and train a model for you. It is a very important feature and lets you detect and identify the category of the images. You can develop a search engine on the top of the result of this service. This helps a user if he wants to see a list of images of products or animals or landmarks, and more.

AutoML Vision Edge - Classification

AutoML Vision Edge – Classification service is similar to Vision classification service, but Edge variation is for implementing at IOT or edge devices. As an example, consider a scenario where you have installed a camera in the forest to record the movements of the animals, and you also want to count the number of animals of a particular category such as Lion, Tiger, and so on. So, for identification of animals in real-time, image classification system using Vision Edge Classification service can be developed and implemented at camera end only. This way, the classification system installed at the camera will identify a category of animals such as Lion or Tiger, as soon as an animal passes through the camera and send this information to the server connected to it.

AutoML Vision Object Detection

AutoML Vision Object Detection service detects an object or multiple objects from an image. It enables you to train your own custom model with your own data.

In this service, you would need to provide a corpus of images tagged with objects such as tomatoes and cucumber in a salad to train it. Then it will automatically identify which model to be used based on your data using Google's architecture search capability and train a model for you. It is a very important feature and lets you detect and identify objects from a given image. You can develop a search engine on the top of the result of this service. This helps a user to fetch list of dishes across restaurants that may contain specific toppings.

AutoML Vision Edge – Object Detection

AutoML Vision Edge – Object Detection service is similar to Vision Object Detection service, but Edge variation of this service is only for implementing at IOT or edge devices. As an example, in self-driving cars, it is necessary that all objects such as road signs, traffic lights, incoming vehicles, and more should be detected in real-time to decide upon an optimal path to be followed by a vehicle. For the same purpose, the Edge version of object detection service is of a great importance, as it performs a task with low-latency and high accuracy.

AI platform

Designing an end-to-end AI product has always been challenging and difficult for an organization whether it is hardware requirement or resource with adequate skillset. Not every organization or company can afford high computing devices such as GPU, TPU etc., high storage requirements and more, as these are quite expensive. In order to solve all these problems, Google came up with a service known as the AI Platform; a framework that helps an organization to build any machine learning model with any data size of any type. The AI platform integrates with the following Google services to build an end-to-end AI based system:

- **Google BigQuery:** It is a data warehouse provided by Google over which we can perform analysis. You can store your dataset in Google BigQuery and later on the same data can be accessed by the AI platform for training and testing purposes.
- **Google Cloud Dataflow:** It is a data-processing service from Google. It enables developers to develop any processing tasks such as missing value computation as a part of developing machine learning system. Google's AI platform provides an integration with Cloud dataflow for any pre-processing tasks that need to be applied over dataset.
- **Google Cloud Object Storage:** It is a storage service provided by Google. You can store your dataset in the cloud storage for training and testing purposes. This is a file storage service, although Google BigQuery is mainly used for analysis purpose. A model can be trained locally or over the cloud that can be stored in the cloud storage. This stored model can be accessed during prediction or inference step.
- **TensorFlow:** The AI platform also provides an integration with Google's deep-learning library i.e. TensorFlow, if you want to develop or code your own model from scratch and use it on the AI platform.

TensorFlow

TensorFlow (TF) is a deep-learning library developed by Google. It is one of the most widely used library for developing machine learning, Natural Language Processing, and deep-learning models. It offers out of the box trained models such as BERT, Universal Sentence Encoder for sentence embeddings that will be used for implementation of the state-of-art machine learning and Natural Language Processing tasks such as Machine Translation, Question–Answering System etc.

TensorFlow has been designed in such a way that it will be useful for the developers as well as data scientists. As an example, data scientists can use the out of the box trained models to check the performance of the models on a new dataset, whereas the developers can directly use them in applications or products as a module or package. However, if the data scientist feels that the out of the box trained model is not good for their task in-hand, then they can change the model architecture by writing their own custom code. So, TensorFlow has greatly reduced the time required to develop machine learning or deep-learning models by providing an out-of-box trained model for some of the common pre-processing tasks such as sentence embedding for Question–Answering System. TensorFlow provides you the flexibility to train your model across various computation platforms such as CPUs, GPUs, and TPUs.

There are two modes in which TensorFlow runs, namely, Eager and Graph execution which is explained as follows:

Eager Execution

In **Eager Execution**, all operations are executed immediately as soon as they are called from python. It lets the developers and data scientists to execute their TensorFlow code without getting into hassle of designing or creating graphs. In this mode of execution, actions return actual values instead of generating a computation graph that will run later on. This mode of execution is quite useful and easy if you want to do hands-on TensorFlow, and this also makes debugging a little bit easier.

Eager execution in TensorFlow provides the following:

- **An instinctive interface:** It uses Python data structure and can structure your TensorFlow code automatically as well.
- **Fast debugging:** It can be integrated with Python debugging tools for fast debugging purpose.
- **Out-of-box control flow:** Uses basic Python programming control flow instead of graph computation flow, thereby, simplifying the model training process.

Graph Execution

Graph Execution, as the name suggests, represents all operations in your model in the form of a computational graph. All the non-leaf nodes represent operations, whereas the leaf nodes represent variable and constant .A computation graph is an arrangement of nodes along with operations in your model. Therefore, in this type of execution, TensorFlow first creates a computation graph and then executes them with the values later on. As an example, if you want to implement an expression such as f(x,y)=x^2y+y+2, then TensorFlow will first build a graph similar to as shown in *Figure 2.2:*

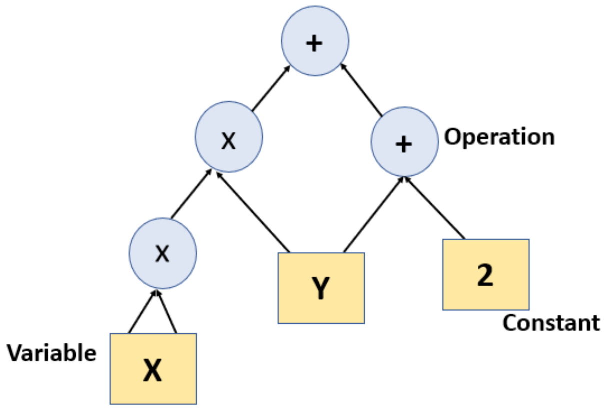


Figure 2.2: *Graph Execution*

We have discussed previously about the usage of the pre-trained models directly in your code. Let's now discuss about Estimators that provide a way to use these models.

Estimators

Estimators provide high-level APIs to access pre-trained machine learning models instead of going into low-level details of TensorFlow. The idea of the difference between high-level and low-level APIs of TensorFlow is quite similar to what we have with programming languages. As an example, developers are more comfortable with high level languages such as Java, Python etc. as compared with the assembly language. Due to this reason, it is always advisable for beginners to use high-level APIs that enables them to use pre-defined estimators directly in their Python code.

There are two following ways of working with Estimators:

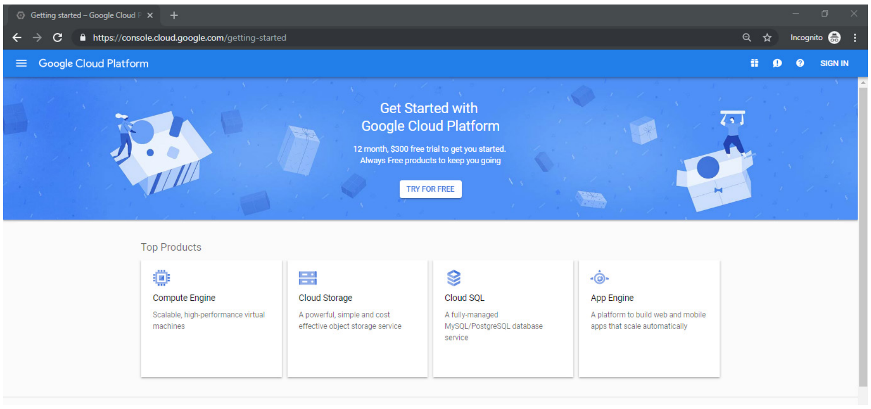
- **Pre-made Estimators:** Pre-made estimators are much easier to use as they provide much higher-level APIs than the base TensorFlow APIs. With the pre-made estimators, there is no need to create graphs or sessions explicitly. The pre-made estimators take care of all the initializations of the graphs and sessions behind the scenes. With the pre-made Estimators, now it is easy to test various model architectures and can be easily tested with minimal code changes. As an example, estimator `tf.estimator.DNNClassifier` can be used to develop a classification model.
- **Custom Estimators:** Customer estimators are the ones that have to be developed explicitly if the required model is not available in the pre-made estimators. It makes use of low-level APIs of TensorFlow to develop estimators.

Next, we will see account creation steps on Google Cloud Platform. We will begin by creating an account and setting up the Google Cloud as a foundation for our hands-on practice on the Google AutoML services.

Creation of GCP account

This section shows how to set up an account on Google Cloud platform. Go to the URL **https://console.cloud.google.com** and follow the following steps:

1. Click on **`TRY FOR FREE`** button as shown in *Figure 2.3*.


Getting started – Google Cloud P
https://console.cloud.google.com/getting-started
Incognito
Google Cloud Platform
SIGN IN
TRY FOR FREE

Get Started with Google Cloud Platform

12 month, $300 free trial to get you started.
Always Free products to keep you going

Top Products

Compute Engine
Scalable, high-performance virtual machines

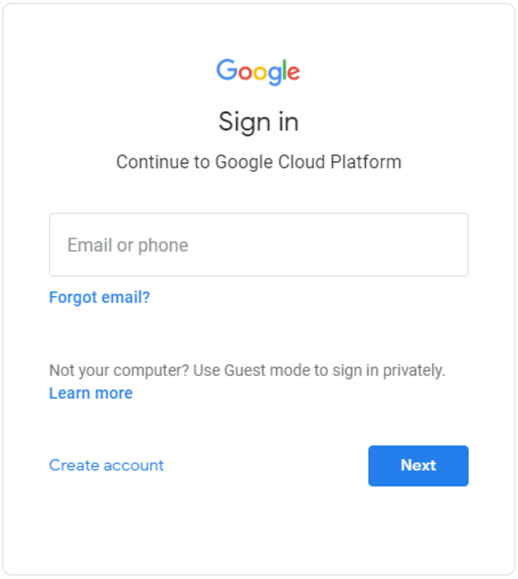
Cloud Storage
A powerful, simple and cost effective object storage service

Cloud SQL
A fully-managed MySQL/PostgreSQL database service

App Engine
A platform to build web and mobile apps that scale automatically

Figure 2.3: *GCP – Account Creation*

2. This will redirect you to Google sign-in page as shown in *Figure 2.4*. Please provide credentials of your Gmail account.



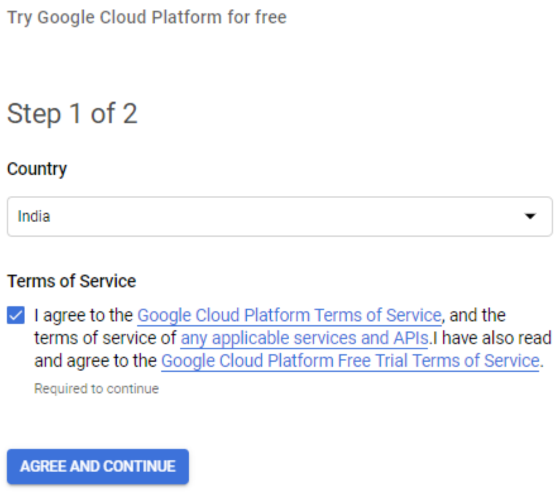
Continue to Google Cloud Platform

Email or phone

Not your computer? Use Guest mode to sign in privately.

Figure 2.4: *Google Sign-in*

3. Select your country, agree to the **`Terms of Service`**, and click on **`AGREE AND CONTINUE`** button as shown in *Figure 2.5.*


Try Google Cloud Platform for free

Step 1 of 2

Country

India

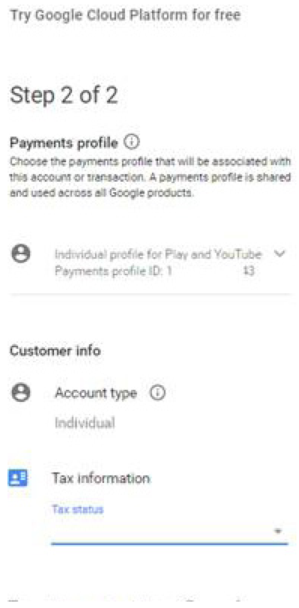
Terms of Service

☑ I agree to the Google Cloud Platform Terms of Service, and the terms of service of any applicable services and APIs.I have also read and agree to the Google Cloud Platform Free Trial Terms of Service.

Required to continue

Figure 2.5: *Google Sign-in – Step 1*

4. This will take you to the second step as shown in *Figure 2.6.*


Try Google Cloud Platform for free

Step 2 of 2

Payments profile

Choose the payments profile that will be associated with this account or transaction. A payments profile is shared and used across all Google products.

Individual profile for Play and YouTube
Payments profile ID: 1 13

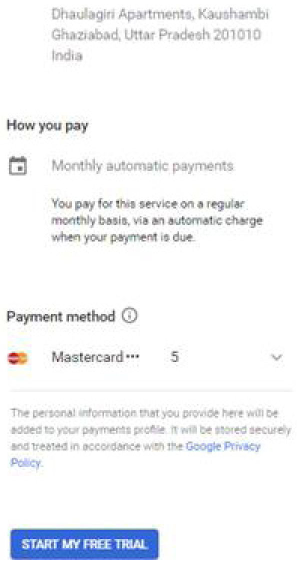
Customer info

Account type
Individual

Tax information
Tax status

Figure 2.6: *Google Sign-in – Step 2 (Part -I)*

5. Scroll down and provide payment method of your choice as shown in *Figure 2.7*. Fill in the required details such as Name, Address, and the Payment information.


Dhaulagiri Apartments, Kaushambi
Ghaziabad, Uttar Pradesh 201010
India

How you pay

Monthly automatic payments

You pay for this service on a regular monthly basis, via an automatic charge when your payment is due.

Payment method

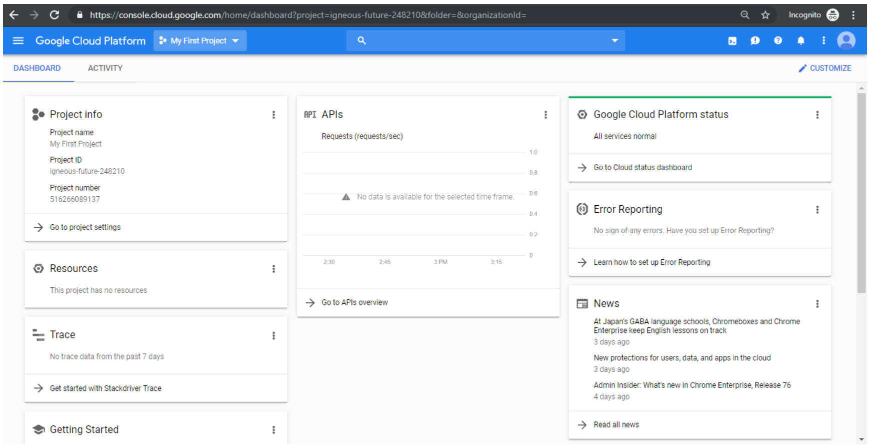
Mastercard ••• 5

The personal information that you provide here will be added to your payments profile. It will be stored securely and treated in accordance with the Google Privacy Policy.

Figure 2.7: *Google Sign-in – Step 2 (Part -II)*

For every new account, Google provides $330 credit with a validity of 12 months, which is enough to explore all the exercises in the book. Please ensure that you stop the services after doing the exercises and keep a tab on the total spend. An excessive usage of the cloud account would incur charges beyond the free limit.

1. Click on **`START MY FREE TRIAL`** button once you have provided all the required details. Please note that this will take some time for the registration to be completed.



Project info
Project name
My First Project
Project ID
igneous-future-248210
Project number
516266089137
Go to project settings

Resources
This project has no resources

Trace
No trace data from the past 7 days
Get started with Stackdriver Trace

Getting Started

APIs
No data is available for the selected time frame.
Go to APIs overview

Google Cloud Platform status
All services normal
Go to Cloud status dashboard

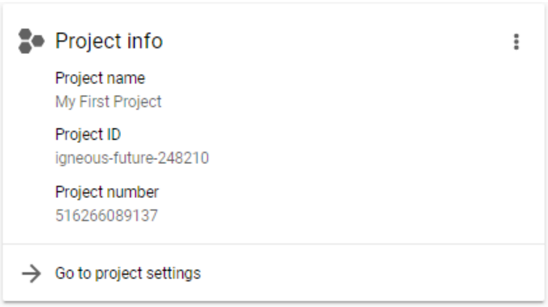
Error Reporting
No sign of any errors. Have you set up Error Reporting?
Learn how to set up Error Reporting

News
At Japan's GABA language schools, Chromeboxes and Chrome Enterprise keep English lessons on track
3 days ago
New protections for users, data, and apps in the cloud
3 days ago
Admin Insider: What's new in Chrome Enterprise, Release 76
4 days ago
Read all news

Figure 2.8: *GCP Home Page*

2. New project with a name `My First Project` will be created automatically as soon as a user signs-in.

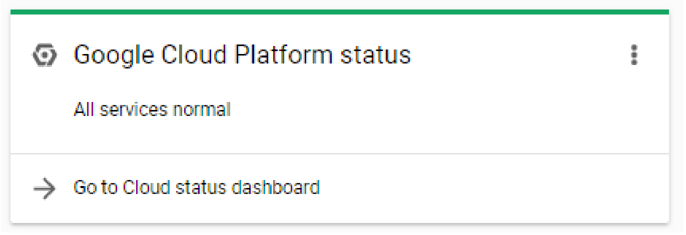
3. **`Dashboard`** tab will give you the details about the selected project. This view is further divided into cards wherein each provides specific information. As an example, **`Project info`** card gives details about the project such as **`Project name,`** **`Project ID`**, and **`Project number`**.



Project info
Project name
My First Project
Project ID
igneous-future-248210
Project number
516266089137
Go to project settings

Figure 2.9: *Project info*

4. **`Google Cloud Platform status`** card as seen in *Figure 2.10* gives a quick service status check. Green indicates everything is working fine.



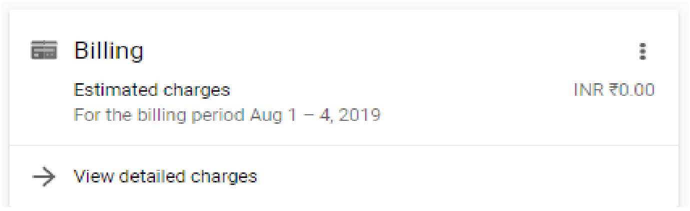
Google Cloud Platform status

All services normal

Go to Cloud status dashboard

Figure 2.10: *Google Cloud Platform status*

5. The **`Billing`** card shows the billing period for the selected project as shown in *Figure 2.11*.



Billing

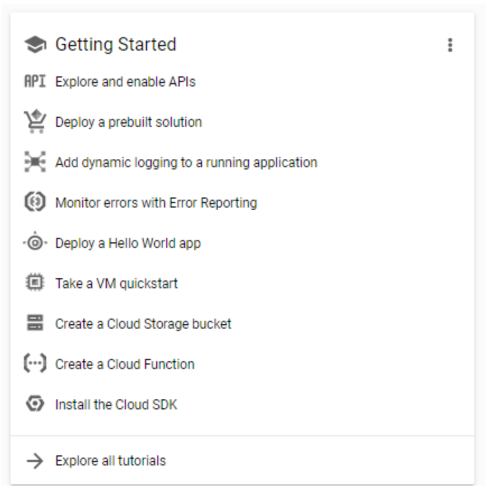
Estimated charges INR ₹0.00

For the billing period Aug 1 – 4, 2019

View detailed charges

Figure 2.11: *Billing Information*

We also have a card for Quick Starts for various services of GCP as shown in *Figure 2.12*. Have a look at these and go through the service details if you are interested in knowing more about a particular service.



Getting Started

Explore and enable APIs

Deploy a prebuilt solution

Add dynamic logging to a running application

Monitor errors with Error Reporting

Deploy a Hello World app

Take a VM quickstart

Create a Cloud Storage bucket

Create a Cloud Function

Install the Cloud SDK

Explore all tutorials

Figure 2.12: *Getting Started*

6. In the left-hand side of the top-most panel is the navigation button as shown in *Figure 2.13*. On clicking this button, it opens the navigation menu where all the GCP services are categorized and listed.

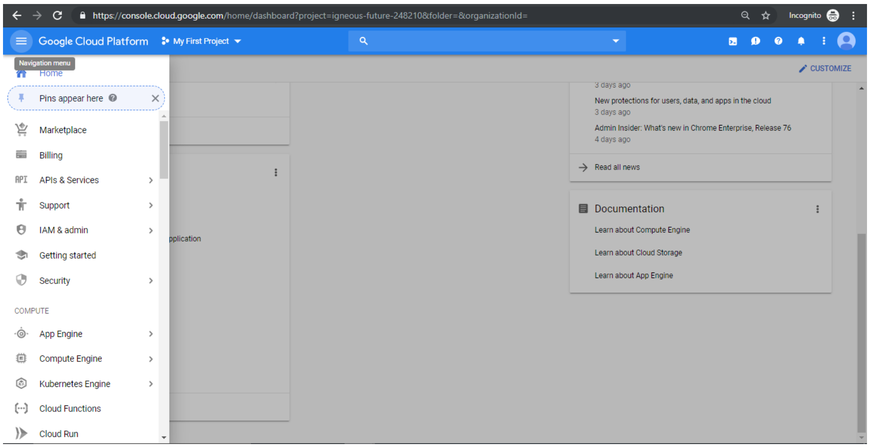


Figure 2.13: *GCP Catalogue*

Creating a Google Cloud Storage bucket

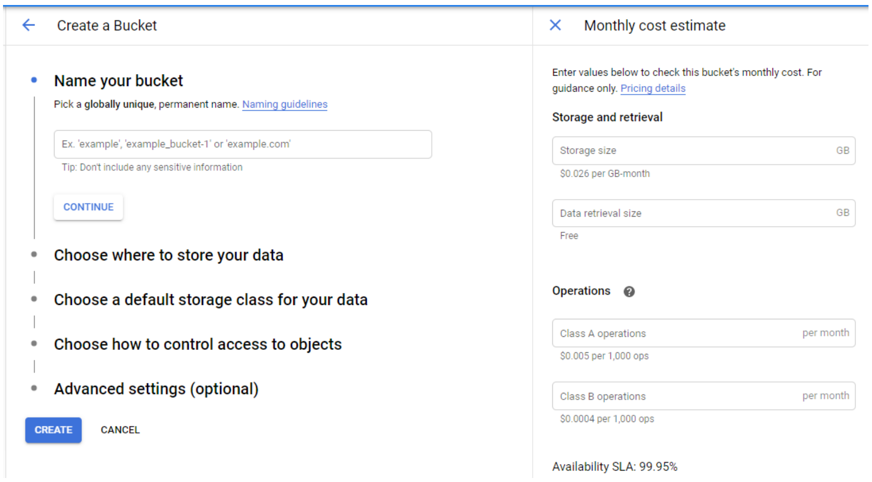
Cloud storage is a storage service offered by Google Cloud. You can upload the data in any file format. It stores the data in a Storage Bucket. All the storage buckets are associated with a project.

After a project has been created, you can create Cloud Storage buckets, upload data to your buckets, and download the data from your buckets. All the GCP services would use the data from this Google Cloud Storage bucket.

Let's see the following steps to create a Google Cloud Storage bucket.

Go to the Google storage URL **https://console.cloud.google.com/storage/browser**

Click on the **`Create Bucket`** button to open the Bucket creation form as shown in *Figure 2.14*.


Create a Bucket

Name your bucket

Pick a **globally unique**, permanent name. Naming guidelines

Ex. 'example', 'example_bucket-1' or 'example.com'

Tip: Don't include any sensitive information

CONTINUE

Choose where to store your data

Choose a default storage class for your data

Choose how to control access to objects

Advanced settings (optional)

CREATE CANCEL

Monthly cost estimate

Enter values below to check this bucket's monthly cost. For guidance only. Pricing details

Storage and retrieval

Storage size GB

$0.026 per GB-month

Data retrieval size GB

Free

Operations

Class A operations per month

$0.005 per 1,000 ops

Class B operations per month

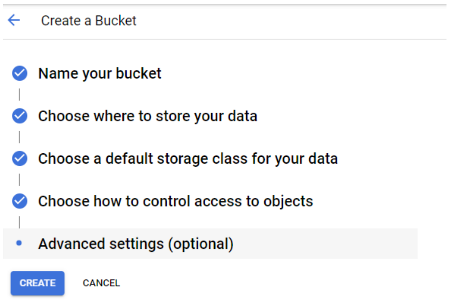
$0.0004 per 1,000 ops

Availability SLA: 99.95%

Figure 2.14: *Create a Bucket - I*

Fill in the Bucket information as follows:

1. **`Name your bucket`**
2. Choose the Location where you want to create a bucket
3. Select a Default storage class for the bucket. The default storage class is best for short term storage and frequently accessed data.
4. Select an Access control model to determine how you control access to the bucket's objects.
5. Select **`Advanced settings (optional)`**. This allows you to add bucket labels, set a retention policy, and choose an encryption method.
6. Click the **`CREATE`** button.


Create a Bucket

Name your bucket

Choose where to store your data

Choose a default storage class for your data

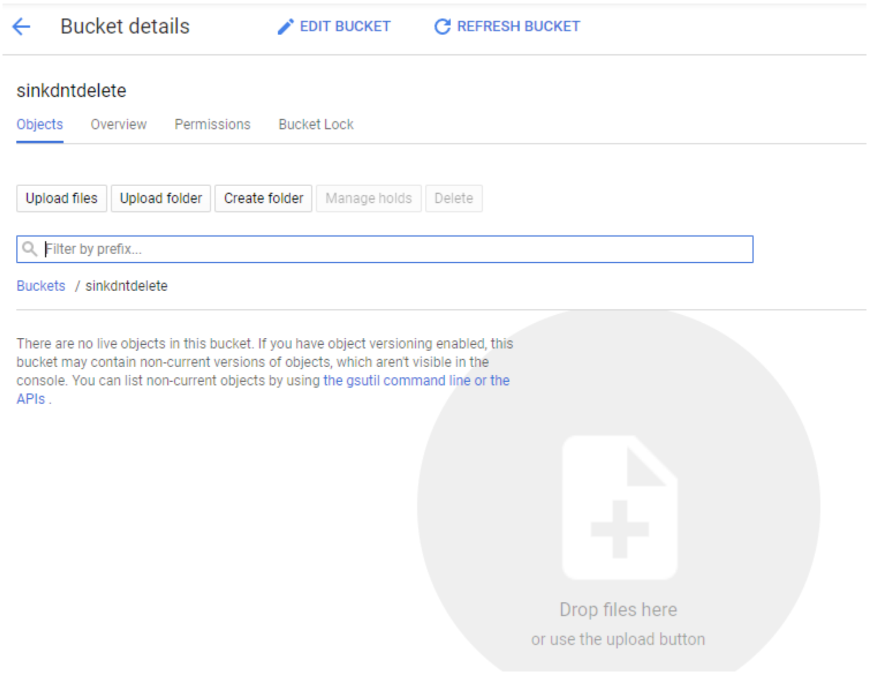
Choose how to control access to objects

Advanced settings (optional)

CREATE CANCEL

Figure 2.15: *Create a Bucket - II*

After clicking the **CREATE** button, a bucket will be successfully created. You will be redirected to a page as shown in *Figure 2.16:*

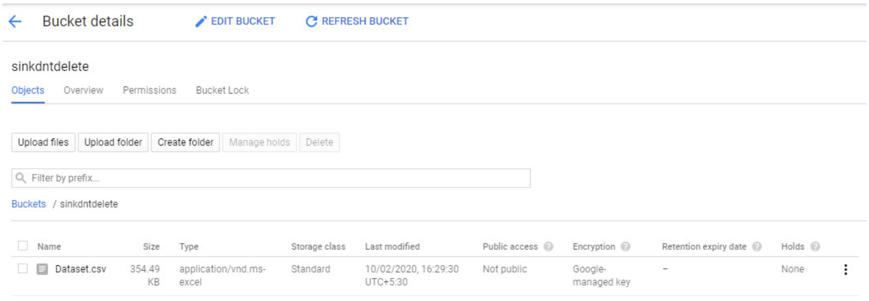


There are no live objects in this bucket. If you have object versioning enabled, this bucket may contain non-current versions of objects, which aren't visible in the console. You can list non-current objects by using the gsutil command line or the APIs .

Figure 2.16: *Bucket details*

Let's see how to upload data inside a bucket.

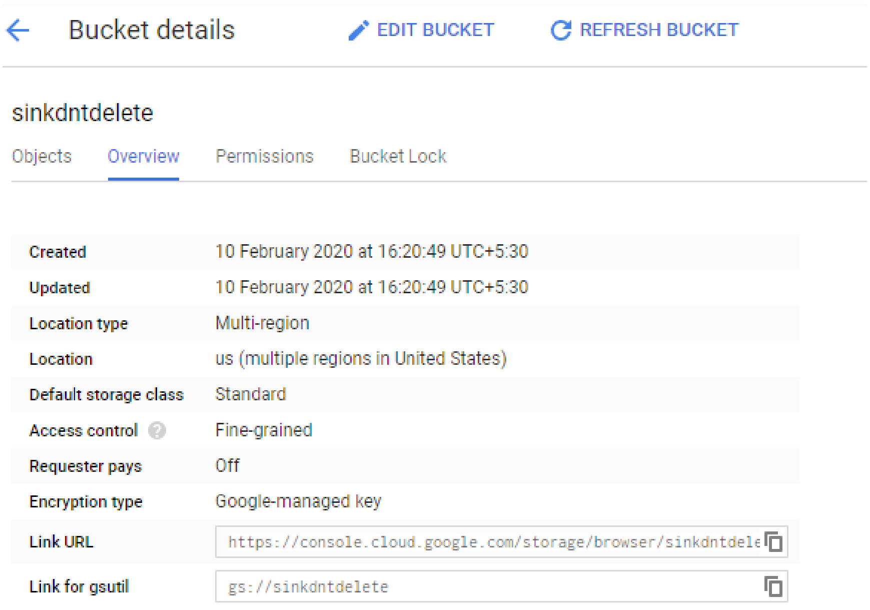
You can upload files/folders or use Drag & Drop files option from your system as shown in *Figure 2.17:*



Name	Size	Type	Storage class	Last modified	Public access	Encryption	Retention expiry date	Holds
Dataset.csv	354.49 KB	application/vnd.ms-excel	Standard	10/02/2020, 16:29:30 UTC+5:30	Not public	Google-managed key	–	None

Figure 2.17: *Bucket - List of Files*

In the overview section, it gives details about bucket and bucket **`Link URL`** and **`Link for gsutil`** that will be used to access the bucket from outside whenever required.



Created	10 February 2020 at 16:20:49 UTC+5:30
Updated	10 February 2020 at 16:20:49 UTC+5:30
Location type	Multi-region
Location	us (multiple regions in United States)
Default storage class	Standard
Access control	Fine-grained
Requester pays	Off
Encryption type	Google-managed key
Link URL	https://console.cloud.google.com/storage/browser/sinkdntdele
Link for gsutil	gs://sinkdntdelete

***Figure 2.18:** Bucket Overview*

Conclusion

In this chapter, we have discussed about the various services offered by Google AutoML. We have also discussed about the creation of an account in Google Cloud platform and creation of Google Cloud Storage bucket. In the next chapter, we will discuss about how an AutoML Natural Language Service can be used to train the custom models as per your dataset.

Chapter 3
AutoML Natural Language

AutoML enables developers or data scientists to build machine learning models over a custom dataset in a simplistic way. It implements all the stages of machine learning pipeline such as data preparation, feature engineering, model selection, hyper parameter tuning, and selection of evaluation metrics.

AutoML Natural Language uses Google's model architecture search algorithm to find best model and its architecture for your dataset. AutoML Natural Language analyzes and performs syntactic analysis (structure) and semantic analysis (meaning of text) to generate knowledge out of it. This knowledge or information can be used as a pre-requisite for the development of the state-of-art NLP technologies such as Question–Answering System, Document Summarization, Recommendation systems, and so on. In this chapter, we will see how an AutoML Natural Language service can be used for implementation of a classification system.

Structure

In this chapter, we will cover the following topics:

- Design and implementation of an Issue Categorization System using AutoML
  - o Description and analysis of the dataset
  - o Machine Learning model training, evaluation, and prediction for Issue Categorization System
  - o REST API configuration to release features of this system for other applications
- Design and implementation of Sentiment Analysis System using AutoML
  - o Description and analysis of the dataset
  - o Machine Learning model training, evaluation, and prediction for Sentiment Analysis System
  - o REST API configuration to release the features of this system for other applications

Objectives

After studying this chapter, you should be able to:

- Understand the problem formulation for machine learning tasks
- Understand the importance of data analysis
- Understand Model training, evaluation, and prediction using Google AutoML
- Understand how a machine learning model that has trained on AutoML can be used for prediction in your application via REST API

Issue Categorization System

In this use case, we will simulate a service desk team of an organization responsible for solving users' issues. If an organization has a large user base, then it increases operational cost as well as load on the service teams supporting the end users. The example of a service desk is an end user computing service desk that provides support to end users on issues faced on desktops, laptops, mail, messaging, and so on. As an example, if a user is facing some issues with his printer or wants to know about the insurance policy, then the user will reach out to the service desk team of an organization. In such case, someone from the service desk team tries to understand a user's query or issue and accordingly reply based on the **Standard Operating Procedures (SOPs)** that have been drafted as per the organization standards. However, some of these queries or issues are quite common and consumes lot of time and efforts of the service desk teams to provide a resolution. What if there is a system that can automatically identify or categorize a user's query or issue to a particular category and provide a required SOP document or can work as an input to a system that automatically assigns an issue to a concerned person in the support team. For an example, if a user is facing an issue while resetting a password in Microsoft Outlook, then either the user will get an SOP, or it may get assigned to a support person who is an expert in solving Outlook related issues to solve it.

Categorization of issues or queries from a user is known as an Issue Categorization system. We have formulated this problem as a classification problem in machine learning where the issue categories are classes. So, the Issue Categorization system will receive an issue or a query description and categorize the same to a category to which it should belong.

Therefore, in this book, we will walk through the implementation of the Issue Categorization system using the state-of-art technologies used by Google AutoML. For this use case, we have used the Natural Language service of AutoML where we will upload our labeled dataset that contains issue description along with their category. Here, AutoML will automatically find the best model architecture using Google's architecture search algorithms to be used based on our dataset. Following are the complete details of the implementation of a system:

- **Prepare Training Data:** In this step, feature engineering such as missing value imputation, data validation, one-hot encoding, and more will be performed over a raw dataset.
- **Create and import Dataset:** In this step, prepared data will be uploaded, and a dataset will be created as required by AutoML. This dataset works as an input to the model training process.

- **Model Training:** In this step, AutoML uses Google's architecture search algorithm to analyze your dataset and find the best model to train.
- **Evaluate Model:** In this step, trained model will be evaluated over the test data. If it is not efficient, then it also offers you to train different version of the same model but with different values of hyperparameter.
- **Model Prediction:** In this step, the trained model will be used to predict the category for any new issue or query raised by the user. Applications such as an Automated Ticket Assignation System can be integrated with it to predict the category for any new issue or query and accordingly assign it to a relevant support person.

The following diagram gives a complete overview of the different stages of an Auto ML Workflow:

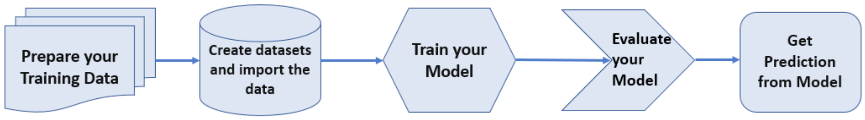

Prepare your Training Data
Create datasets and import the data
Train your Model
Evaluate your Model
Get Prediction from Model

Figure 3.1: *AutoML Workflow*

Data Analysis

For this implementation, the dataset file (Name: `Dataset.csv`) can be downloaded from GitHub URL **https://github.com/automl-book/AutoML**. In this dataset, each and every query would belong to only and only one category. So, we will go with the training of a single-labeled multi-class classification model. A dataset contains 4965 records distributed across three categories, namely, CPU, memory, and password reset related issues; and classes are named as `CPU_Utilization`, `Password_Reset`, and `Memory_Utilization`. Each class is having different number of variations as shown in the following table:

Classes	No of Variations
CPU_Utilization	204
Password_Reset	4223
Memory_Utilization	538
Total	**4965**

However, as per the best practices, training data should contain equal or nearly-equal number of variations for each class. But in our use case, each class contains different number of variations. As the class `Password_Reset` has almost 20 times a greater number of queries as compared with class `CPU_Utilization`, this dataset suffers from class-imbalance problem and would degrade the performance of the class with a smaller number of variations. However, AutoML identifies class imbalance problem in a dataset and solves this automatically. This is an important feature of the AutoML, where it provides a guided way to the engineers to evaluate and solve the data science problems.

In the next section, we will see how the aforementioned discussed dataset will be uploaded to build an Issue Categorization System using an AutoML Natural Language Service on Google Cloud Platform.

Model Training

We will first enable an AutoML Natural Language service from Google Cloud. Follow the following steps to enable an AutoML Natural Language service.

1. Go to URL **https://cloud.google.com/**and login with your credentials.
2. Click on burger Menu and select the **`Natural Language`** option as shown in *Figure 3.2*.

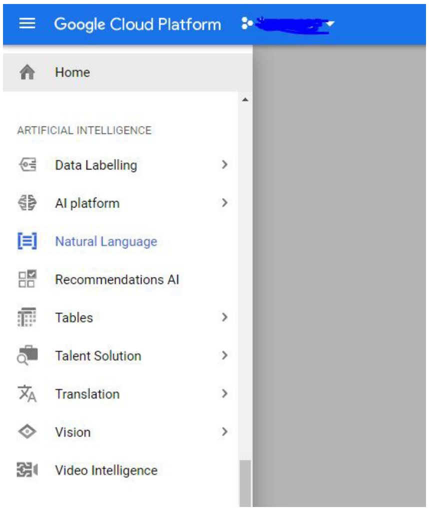
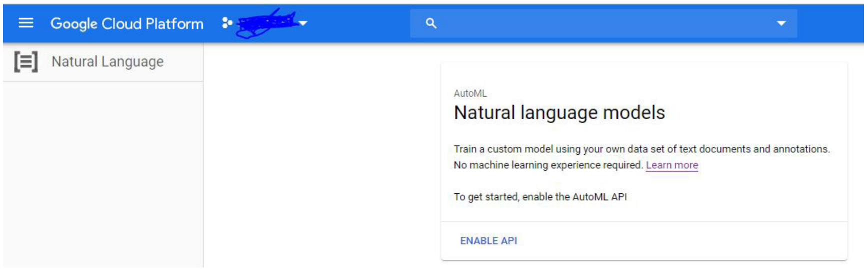


Figure 3.2: *Selection of Natural Language Service*

3. This will populate the screen as shown in *Figure 3.3*. Next, click on the **`ENABLE API`** button to enable the AutoML Natural Language API.



AutoML

Natural language models

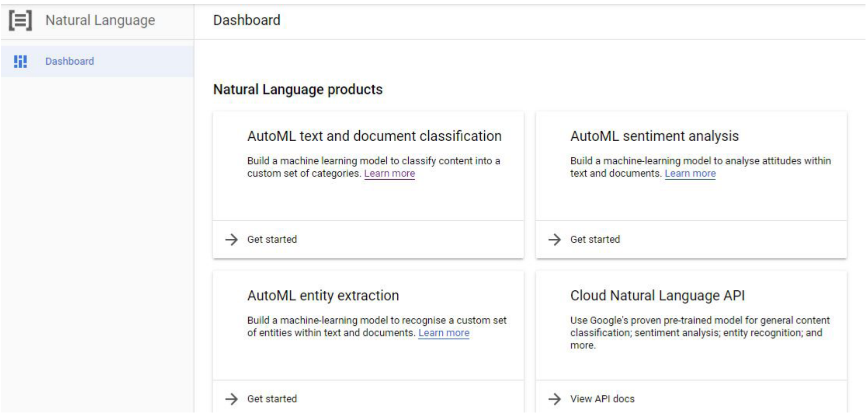
Train a custom model using your own data set of text documents and annotations. No machine learning experience required. Learn more

To get started, enable the AutoML API

ENABLE API

Figure 3.3: *Natural Language Models*

4. Once enabled, it will populate the screen that shows details of the features provided by the Natural Language Service as shown in *Figure 3.4*.



Natural Language products

AutoML text and document classification

Build a machine learning model to classify content into a custom set of categories. Learn more

Get started

AutoML sentiment analysis

Build a machine-learning model to analyse attitudes within text and documents. Learn more

Get started

AutoML entity extraction

Build a machine-learning model to recognise a custom set of entities within text and documents. Learn more

Get started

Cloud Natural Language API

Use Google's proven pre-trained model for general content classification; sentiment analysis; entity recognition; and more.

View API docs

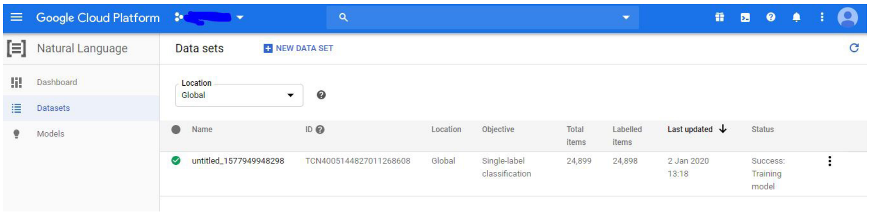
Figure 3.4: *Natural Language Products*

Following are the names and details of the features provided by the Natural Language Service.

- **`AutoML text and document classification:`** Build a machine learning model to classify the content (text or document or sentence) into categories defined during the model training.
- **`AutoML sentiment analysis:`** Build a machine learning model to analyze sentiments out of the text (document or sentence). It also enables you to train the model on your dataset.

- **`AutoML entity extraction:`** Build a machine learning model to extract entities out of the text (document or sentence). It also enables you to train the model on your dataset for custom entities.
- **`Cloud Natural Language API:`** Enables you to use the pre-trained models for classification, sentiment, and entity recognition system over textual content (document or text). It can be integrated directly to your applications or products via REST API.

5. Next, click on the card named as AutoML text and document classification option to build a custom model for the Issue Categorization System. You will be redirected to a Dataset page as shown in *Figure 3.5:*

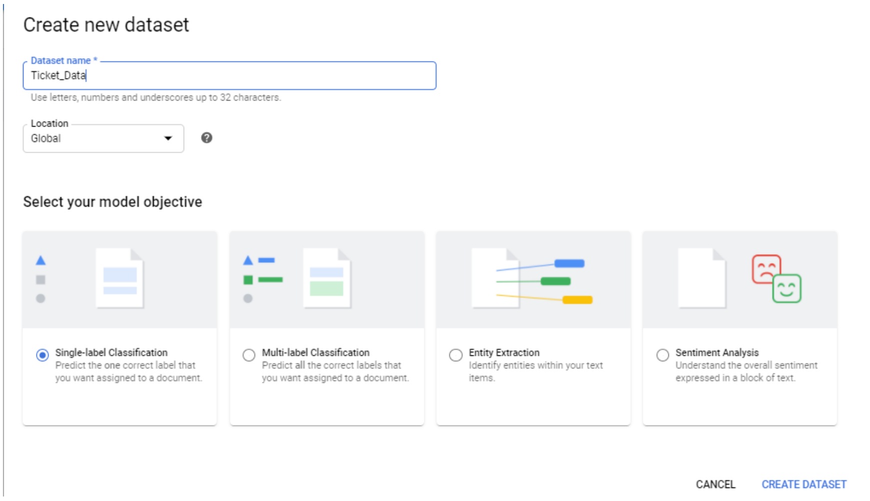


Location
Global

	Name	ID	Location	Objective	Total items	Labelled items	Last updated	Status
✓	untitled_1577949948298	TCN4005144827011268608	Global	Single-label classification	24,899	24,898	2 Jan 2020 13:18	Success: Training model

Figure 3.5: *Natural Language - Dataset*

6. Now, to upload your dataset, click on the **`NEW DATA SET`** link, and you will be redirected to a screen as shown in *Figure 3.6.*


Create new dataset

Dataset name *
Ticket_Data

Use letters, numbers and underscores up to 32 characters.

Location
Global

Select your model objective

[✓ Single-label Classification] Predict the one correct label that you want assigned to a document.

Multi-label Classification Predict all the correct labels that you want assigned to a document.

Entity Extraction Identify entities within your text items.

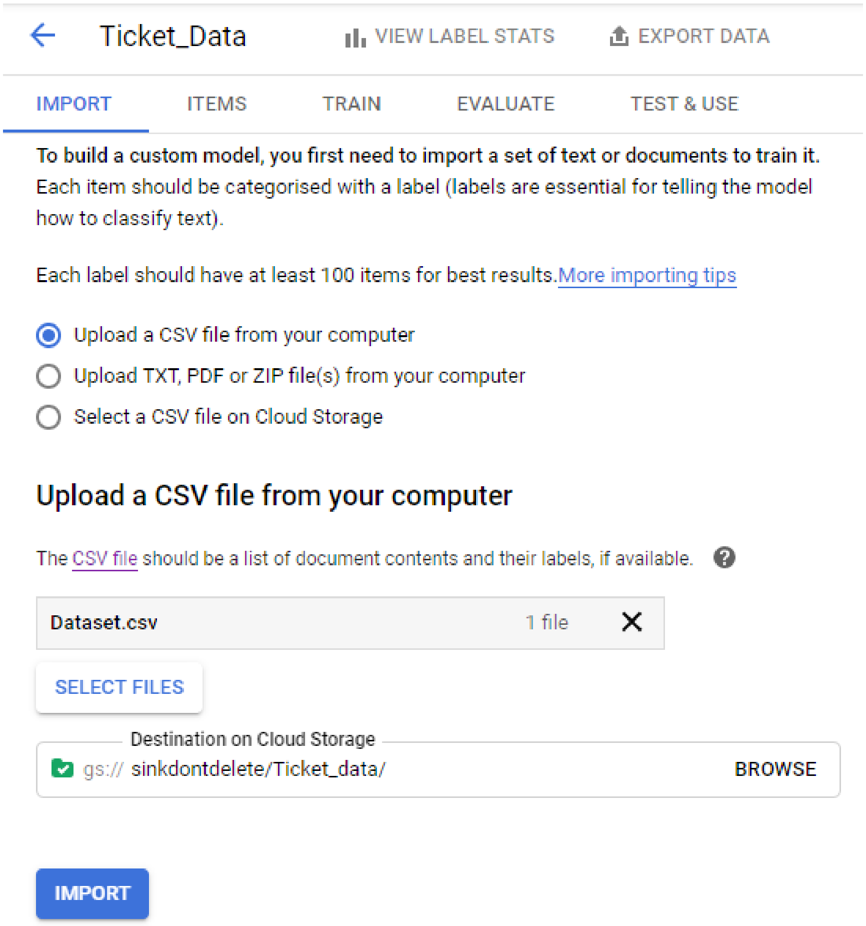
Sentiment Analysis Understand the overall sentiment expressed in a block of text.

Figure 3.6: *Issue Categorization System – Create Dataset*

7. Next, enter the values for various fields as follows:
   - **`Data set name:`** Name of the dataset. For this use case, value will be `Ticket_Data`.
   - **`Location:`** Defines the geographical region in which your data will be stored. You can select Global or European. For this use case, select Global as a value of the location.
   - **`Model Objective:`** Defines what type of model you want to train. It offers the following types of models:
     - o **Single-Label Classification:** Select this if the text (document or sentence) belongs to only one category or group. As an example, issue such as "Not able to send an email via Microsoft Outlook," as you can see that a user is facing an issue while sending an email. Therefore, it belongs to the "Email Issues" category.
     - o **Multi-Label Classification:** Select this if the text (document or sentence) belongs to more than one category. As an example, a document talks about Politics as well as Sports; therefore, it will be categorized to Politics and Sports.
     - o **Entity Extraction:** Select this if you want to extract entities out of text (document or sentence).
     - o **Sentiment Analysis:** Select this if you want to identity sentiments out of text (document or sentence).
8. For our use case, select the **`Single-Label classification`** option as all ticket descriptions corresponding to issues categorized to only single category in our dataset, and then click on the **`CREATE DATA SET`** button.
9. Next, AutoML provides the following three options depending upon your dataset file format and location to upload.
   - **`Upload CSV file from your system:`** If the dataset file is in CSV format and resides in your system, then select this option. For our use case, we will go with this option.

- **`Upload TXT, PDF, or ZIP files from your system:`** If the dataset file is in TXT, PDF, or ZIP format and resides in your system, then select this option.
- **`Select a CSV file from cloud storage:`** If the dataset file is in CSV format and stored in cloud storage service such as Object storage, then select this option.

10. Click on the **`SELECT FILES`** button, it will open up browse files box. Select your dataset file and upload it.
11. Define destination on cloud storage as `gs://sinkdontdelete/Ticket_data` (or bucket name that you have defined during the creation of the bucket)as created in *Chapter 2*, and click on the **`IMPORT`** button as shown in *Figure 3.7*.



To build a custom model, you first need to import a set of text or documents to train it. Each item should be categorised with a label (labels are essential for telling the model how to classify text).

Each label should have at least 100 items for best results.More importing tips

[✓ Upload a CSV file from your computer]
Upload TXT, PDF or ZIP file(s) from your computer
Select a CSV file on Cloud Storage

Upload a CSV file from your computer

The CSV file should be a list of document contents and their labels, if available.

Dataset.csv 1 file

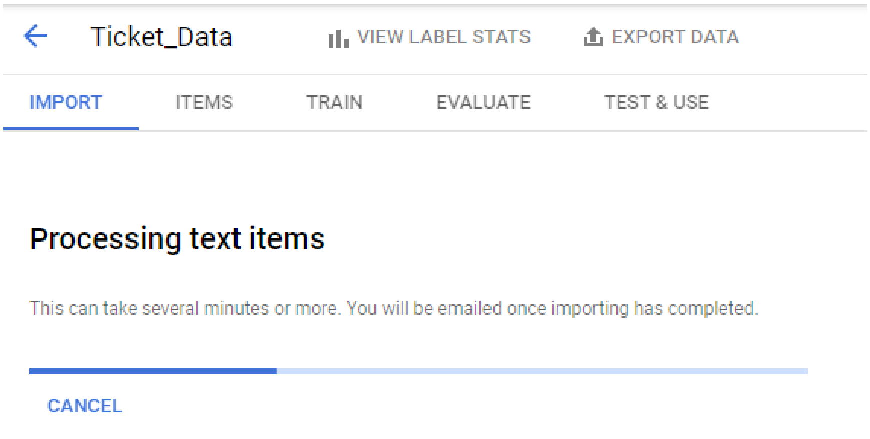
SELECT FILES

Destination on Cloud Storage: gs:// sinkdontdelete/Ticket_data/ BROWSE

IMPORT

Figure 3.7: *Issue Categorization System – Upload Data*

12. Once you click on **`IMPORT`** button then you will be able to see progress of your dataset as shown in *Figure 3.8*.

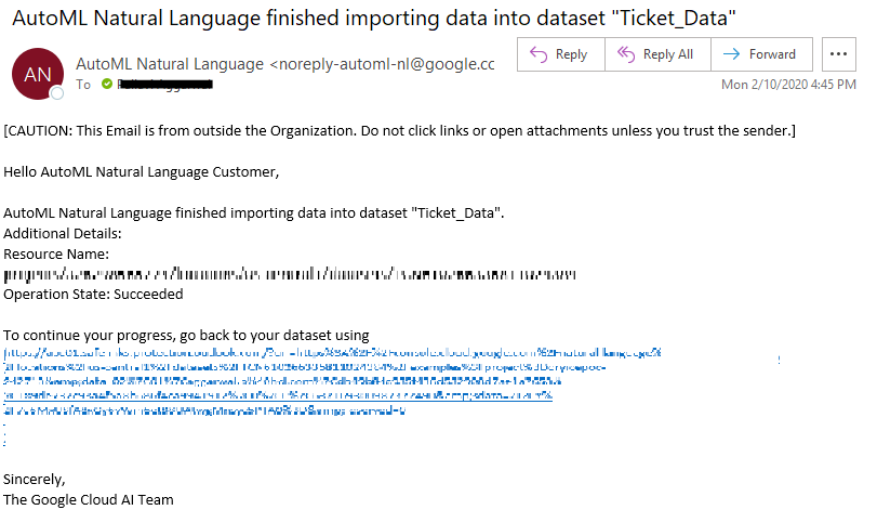


Processing text items

This can take several minutes or more. You will be emailed once importing has completed.

Figure 3.8: *Issue Categorization System – Data Import*

13. This process may take few minutes. You will receive an email on your registered email address associated with your project as soon as the upload process has successfully completed as shown in *Figure 3.9*.



AutoML Natural Language finished importing data into dataset "Ticket_Data"

AutoML Natural Language <noreply-automl-nl@google.cc

To [redacted]

Mon 2/10/2020 4:45 PM

[CAUTION: This Email is from outside the Organization. Do not click links or open attachments unless you trust the sender.]

Hello AutoML Natural Language Customer,

AutoML Natural Language finished importing data into dataset "Ticket_Data".
Additional Details:
Resource Name:
[illegible]
Operation State: Succeeded

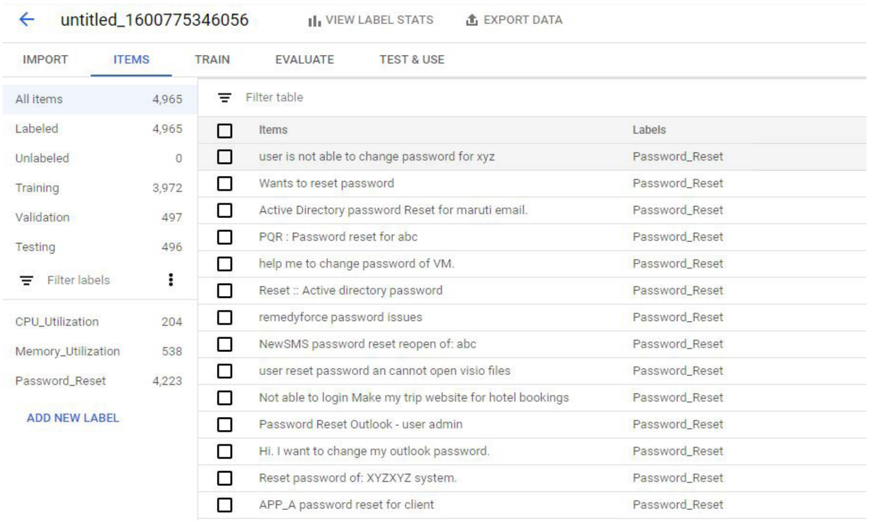
To continue your progress, go back to your dataset using
[illegible]

Sincerely,
The Google Cloud AI Team

Figure 3.9: *Issue Categorization System – Data Upload Email Confirmation*

Once the data has been successfully uploaded, then you can see the following details about your dataset under the **`ITEMS`** tab as shown in *Figure 3.10*.

- **`ALL items:`** shows the total number of records in your dataset
- **`Labeled:`** This shows the total number of labeled records in your dataset. In our dataset, all records or variations have already been labeled. So, it is equal to All items.
- **`Unlabeled:`** This shows the total number of unlabeled records in your dataset. In our dataset, not even a single record is unlabeled. So, its value is 0.
- **`Training:`** Shows the number of records to be used during model training
- **`Validation:`** Shows the number of records to be used during the validation step
- **`Testing:`** Shows the number of records to be used during testing phase



All items	4,965
Labeled	4,965
Unlabeled	0
Training	3,972
Validation	497
Testing	496
CPU_Utilization	204
Memory_Utilization	538
Password_Reset	4,223

Items	Labels
user is not able to change password for xyz	Password_Reset
Wants to reset password	Password_Reset
Active Directory password Reset for maruti email.	Password_Reset
PQR : Password reset for abc	Password_Reset
help me to change password of VM.	Password_Reset
Reset :: Active directory password	Password_Reset
remedyforce password issues	Password_Reset
NewSMS password reset reopen of: abc	Password_Reset
user reset password an cannot open visio files	Password_Reset
Not able to login Make my trip website for hotel bookings	Password_Reset
Password Reset Outlook - user admin	Password_Reset
Hi. I want to change my outlook password.	Password_Reset
Reset password of: XYZXYZ system.	Password_Reset
APP_A password reset for client	Password_Reset

Figure 3.10: *Issue Categorization System – Data Analysis*

14. During the training of any Artificial Intelligence models, the dataset has to be divided into three parts as follows:
    - **`Training Set:`** The number of records to be used for training a model
    - **`Validation Set:`** The number of records to be used for validating a model and hyperparameter tuning

- Hyperparameter tuning can be defined as a process of identifying optimal values of parameters that defines model architecture. For an example, if you are working on any Neural Network based approach such as an LSTM (Long Short Term Memory), then you need to find optimal values of the parameters such as the number of layers, number of nodes, and so on which would work best for your architecture.
- **`Test Set:`** Number of records to be used for testing a trained model.
- This set contains records that have not been seen during the training and validation phase.

15. By default, the AutoML Natural Language service uses 80% of the records for training, 10% for validation set, and the rest 10% for the testing set. It also takes care of the class imbalance problem (if any) in the dataset. As per AutoML, the minority class should have at least 100 records in the training set as shown in the following table, where minority class i.e. `CPU_Utilization` has more than 100 records.

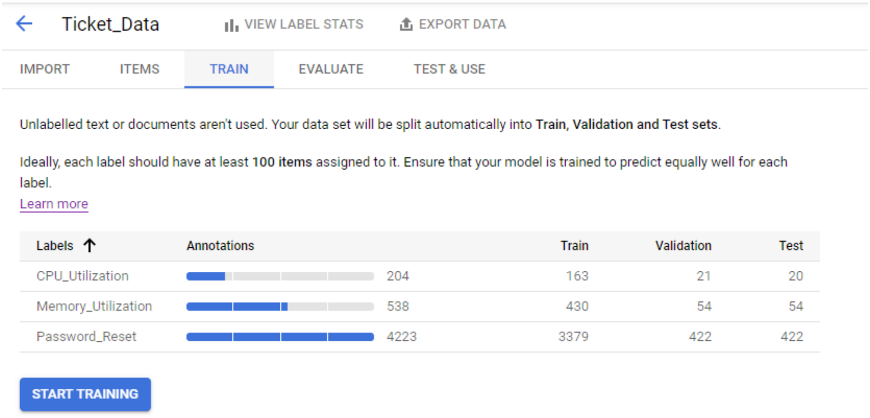
Class / Category	Number of Records	Training	Validation	Testing
CPU_ Utilization	204	163	21	20
Memory_ Utilization	538	430	54	54
Password_ Reset	4223	3379	422	422

However, you can also define your custom splitting criteria where you can tag individual record as `TRAIN, VALIDATE, TEST`, and `UNASSIGNED` by defining them in a new column as follows:

- **`TRAIN:`** This tag defines that record belongs to a training set.
- **`VALIDATE:`** This tag defines that record belongs to a validation set.
- **`TEST:`** This tag defines that record belongs to a testing set.
- **`UNASSIGNED:`** This tag defines that record will be assigned automatically to a training or validation or testing set.

In this case, splitting into a training, validation, and testing set will be performed on the basis of tags only.

For our use case, we will go with the automated splitting offered by AutoML. Next, click on the **`TRAIN`** tab, and then click on the **`START TRAINING`** button as shown in *Figure 3.11*.



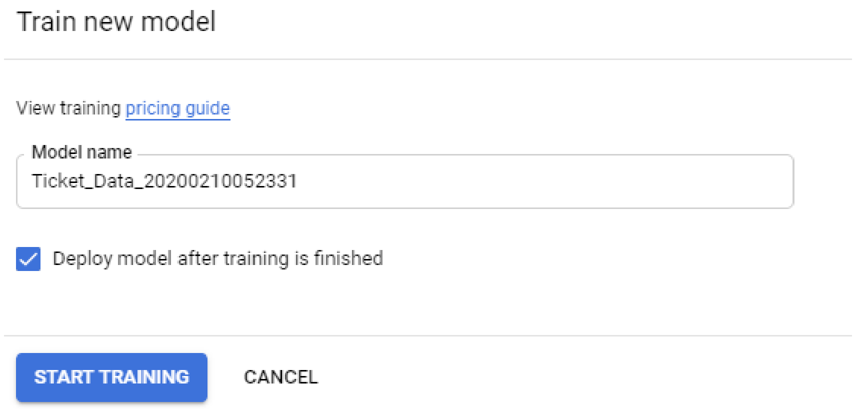
Unlabelled text or documents aren't used. Your data set will be split automatically into **Train, Validation and Test sets**.

Ideally, each label should have at least **100 items** assigned to it. Ensure that your model is trained to predict equally well for each label.

Labels ↑	Annotations	Train	Validation	Test
CPU_Utilization	204	163	21	20
Memory_Utilization	538	430	54	54
Password_Reset	4223	3379	422	422

Figure 3.11: *Issue Categorization System – Model Training*

16. It will be redirected to the page as shown in *Figure 3.12*.



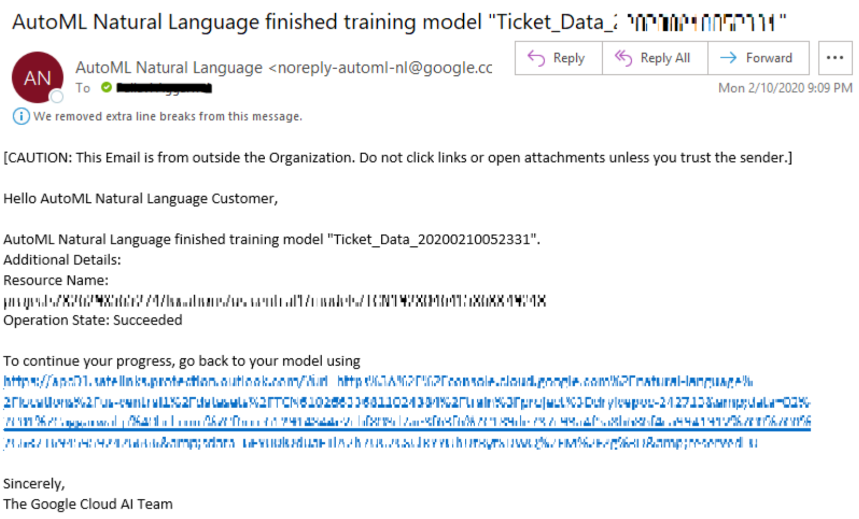
Model name: Ticket_Data_20200210052331

☑ Deploy model after training is finished

Figure 3.12: *Issue Categorization System – Train Model*

17. Define the **`Model name`** and check the box against the **`Deploy model after training is finished`** if you want to deploy the model automatically as soon as training is finished. Then click on the **`START TRAINING`** button to start the model training.

18. The model training process may take several hours depending on the size of the dataset. You will receive an email on your registered email address as soon as the model training is completed as shown in *Figure 3.13*.


AutoML Natural Language finished training model "Ticket_Data_[illegible]"

AutoML Natural Language <noreply-automl-nl@google.cc

To [redacted]

Mon 2/10/2020 9:09 PM

We removed extra line breaks from this message.

[CAUTION: This Email is from outside the Organization. Do not click links or open attachments unless you trust the sender.]

Hello AutoML Natural Language Customer,

AutoML Natural Language finished training model "Ticket_Data_20200210052331".
Additional Details:
Resource Name:
[illegible]
Operation State: Succeeded

To continue your progress, go back to your model using
[illegible]

Sincerely,
The Google Cloud AI Team

Figure 3.13: *Issue Categorization System – Email Confirmation for Model Training*

19. Please note that AutoML automatically uses a validation set to optimize the model hyperparameter values and select the best model for you. You will receive an email only once the AutoML has trained the best model for you.

20. Click on the link mentioned in an email body that you must have received after completion of the model training to check the performance of the model. As aforementioned, AutoML uses a test dataset to evaluate the performance and quality of the trained model.

Model Evaluation

AutoML uses performance metrices such as Precision and Recall to measure the performance of a model as follows:

- **Precision:** It is defined as the ratio of the number of relevant variations or records of a particular class that have been classified to same class by the model to the total number of records in a dataset that have been classified to the same class by the model. As an example, in our use case, precision can be defined as the ratio of the number of records from category `CPU_Utilization` that are classified to `CPU_Utilization` by the model to the total number of records that have been classified to `CPU_Utilization` by the model. Mathematically, it can be defined as follows:

Precision = Number of True Positive/(Number of True Positive + Number of False Positive)

- **Recall:** It is defined as the ratio of the number of relevant variations or records of a particular class that have been classified to the same class to the total number of records that belong to the category if the same class in the dataset. As an example, in our use case, recall can be defined as the ratio of the number of records from the category `CPU_Utilization` that are classified to `CPU_Utilization` by the model to the total number of records that belong to the category `CPU_Utilization` in the training dataset. Mathematically, it can be defined as follows:

  Recall = Number of True Positive/(Number of True Positive + Number of False Negative)

  where, True Positive, False Positive, and False Negative can be defined as follows:

  - **True Positive:** The number of records for the category `CPU_Utilization` from the dataset that have been classified to the category `CPU_Utilization` by the model.
  - **False Positive:** The number of records that have been classified incorrectly to the category `CPU_utilization` by the model.
  - **False Negative:** The number of records that should have been classified to the category `CPU_Utilization` by the model, but these have been classified incorrectly to another category such as `Memory_Utilization` and `Password_Reset` by model.

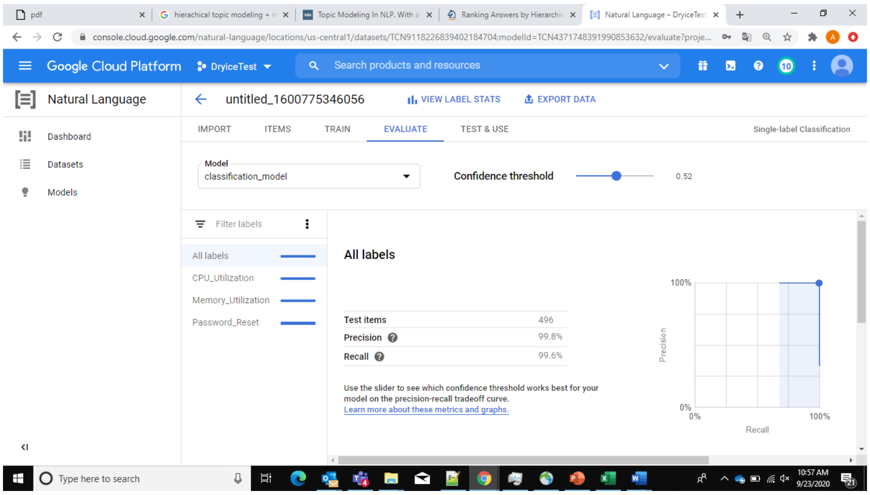
The confidence score plays a major role in evaluating the performance of a classification model. It is a probabilistic score that tells us how much confident the model is in categorizing an issue or query from the user to a particular category. As an example, for an issue description such as **CPU utilization is high** the model categorizes it into category `CPU_Utilization` with a confidence score of 0.98. It means that the model is 98% sure that the given issue description belongs to the category `CPU_Utilization`. It can take any value between 0 and 1.

However, an analyst or a data scientist has to define after what value of the confidence score for a query or an issue description should be considered as relevant. This value of confidence score is known as the Threshold value. It can take any value between 0 and 1; and all the predictions from the model with confidence score greater than threshold value are considered as relevant. As an example, if the threshold value is 0.70, then all the predictions from the model with the confidence score greater than 0.70 will be considered as relevant else considered as irrelevant.

The selection of the threshold value must be done with utmost care to avoid any negative impact on the performance of a model. As an example, if you choose large value of the threshold value near to 1, then most of the issues or queries that belongs to the category `CPU_Utilization` will be classified to other categories in our dataset. This increases False Negative that will have a negative impact on the recall. However, if you choose a low value of the threshold value near to 0, then most of the issues or queries that don't belong to the category `CPU_Utilization` will be classified to the category `CPU_Utilization` in our dataset. This increases False Positive that will have a negative impact on the precision.

As you can see, precision and recall has a trade-off with each other. As soon as the recall increases, precision decreases and vice-versa. It is completely dependent on the business requirement which one, i.e. precision or recall, to prioritize. However, our use case is an Issue Categorization system, where we want the model to classify an issue or a query to correct the category in most of the cases. Therefore, we would prefer precision over recall, but we will not compromise more on recall as well. Thus, every use case is a balancing act between precision and recall using the confidence score.

However, an AutoML Natural Language service provides a feature to change the threshold value known as the confidence threshold score bar. It helps an analyst or a data scientist to see how recall and precision will affect with changes in the threshold value as shown in the following *Figure 3.14*. For our use case, we have selected the threshold value as 0.52, as this gives a decent value of recall and precision, which is acceptable for our use case.



Model: classification_model

Confidence threshold 0.52

All labels

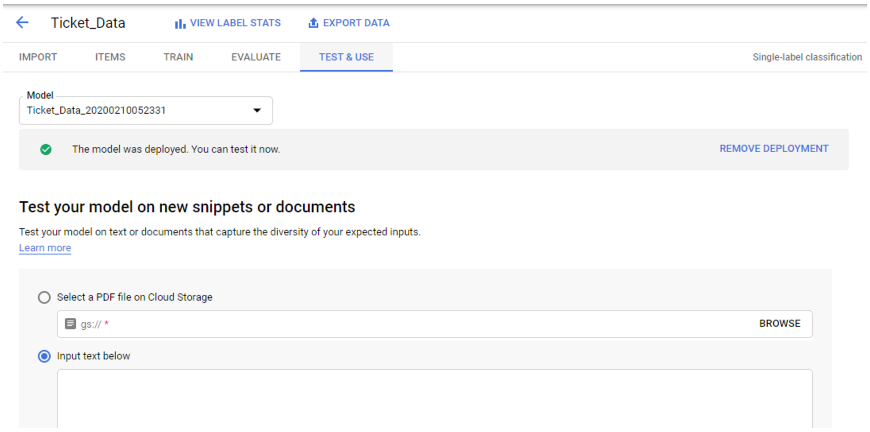
Test items	496
Precision	99.8%
Recall	99.6%

Use the slider to see which confidence threshold works best for your model on the precision-recall tradeoff curve.
Learn more about these metrics and graphs.

Figure 3.14: Issue Categorization System – Model Evaluation

Model Prediction

Once the model has been trained and evaluated, you can check the prediction from the model on a new query or issue by clicking on the **TEST & USE** tab as shown in *Figure 3.15*:



Model: Ticket_Data_20200210052331

The model was deployed. You can test it now.

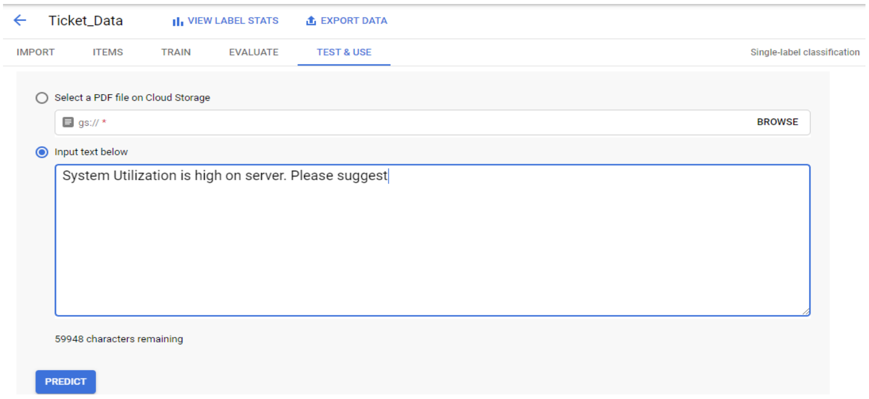
Test your model on new snippets or documents

Test your model on text or documents that capture the diversity of your expected inputs.
Learn more

Select a PDF file on Cloud Storage
gs:// *
Input text below

Figure 3.15: Issue Categorization System – Model Prediction - I

Define a query or an issue description as `CPU Utilization is high on server`. Please suggest in the input text in the following field as shown in *Figure 3.16*. Next, click on the **`PREDICT`** button to see the model results.



Select a PDF file on Cloud Storage

gs:// *

Input text below

System Utilization is high on server. Please suggest

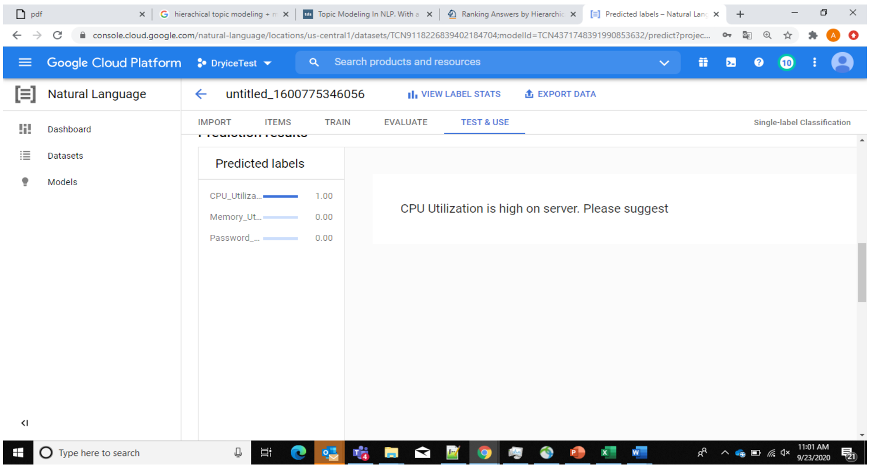
59948 characters remaining

Figure 3.16: Issue Categorization System – Model Prediction - II

The prediction from the model for an issue description shows the probability distribution amongst the three classes or categories with the probability score as follows:

CPU_Utilization	**1.0**
Password_Reset	**0.00**
Memory_Utilization	**0.00**

Amongst these classes or categories, the `CPU_Utilization` is the only one whose confidence score is greater than the threshold value i.e. 0.52. Hence, a query or an issue description is correctly classified to the category `CPU_Utilization` by the model as shown in *Figure 3.17*.



console.cloud.google.com/natural-language/locations/us-central1/datasets/TCN9118226839402184704;modelId=TCN4371748391990853632/predict?projec...

Predicted labels

CPU_Utiliza...	1.00
Memory_Ut...	0.00
Password_...	0.00

CPU Utilization is high on server. Please suggest

Figure 3.17: Issue Categorization System – Model Prediction Result

REST API configuration

Now, consider a scenario where you want to expose this model to an external application. As an example, an organization is working on the development of an Automated Issue Assignation System, where one step is to categorize a user's issue or query into a particular category. Once the category has been identified then the issue or query will be assigned to the concerned person in the support team of an organization.

Now, in order to use this feature in an Issue Assignation system, the Issue Categorization model has to be exposed as a REST API such that whenever a user submits his query or an issue, it will make an API call to fetch a relevant category for that query or issue description. It involves the following steps:

- Creation of service account for accessing an AutoML Natural Language Service
- Installation and configuration of Google Cloud SDK
- Code development, to get the model prediction for a query or an issue description
- Create REST API to get the model prediction

Follow the following steps to create a service account for accessing an AutoML Natural Language Service:

1. Go to URL **https://cloud.google.com/** and login with your credentials.
2. Select the option **`IAM & admin`** from the left side navigation menu

3. Then, click on the service accounts link as shown in *Figure 3.18:*

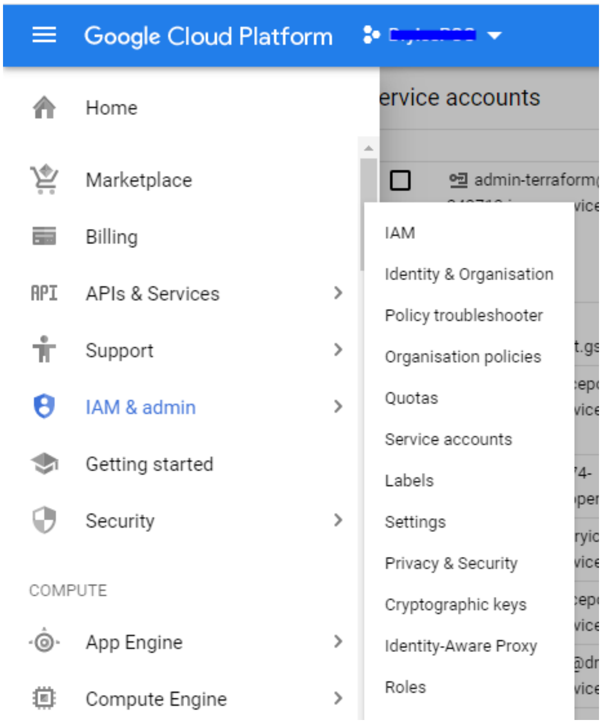
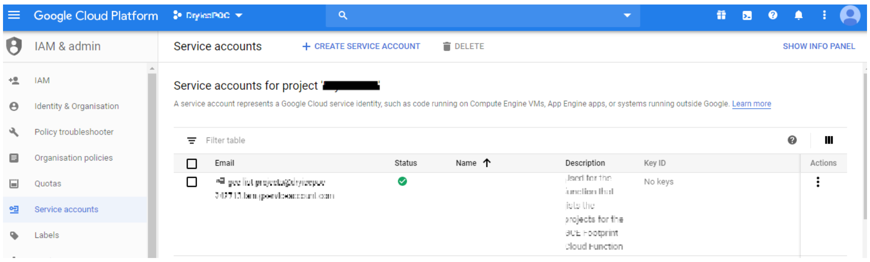


Figure 3.18: *IAM Service Account Creation*

You will be redirected to a page as shown in *Figure 3.19:*



Service accounts for project '[redacted]

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. Learn more

	Email	Status	Name ↑	Description	Key ID	Actions
☐	[illegible]	✓		Used for the function that lists the projects for the GCE Footprint Cloud Function	No keys	⋮

Figure 3.19: *List of Service Accounts*

4. click on the **`CREATE SERVICE ACCOUNT`** link and provide the following details:
   - **`Service account name:`** Defines name of the service account
   - **`Service account ID:`** Defines ID of the service account
   - **`Service account Description:`** Defines the description for this service account.
5. Then, click on the **`CREATE`** button as shown in *Figure 3.20*:

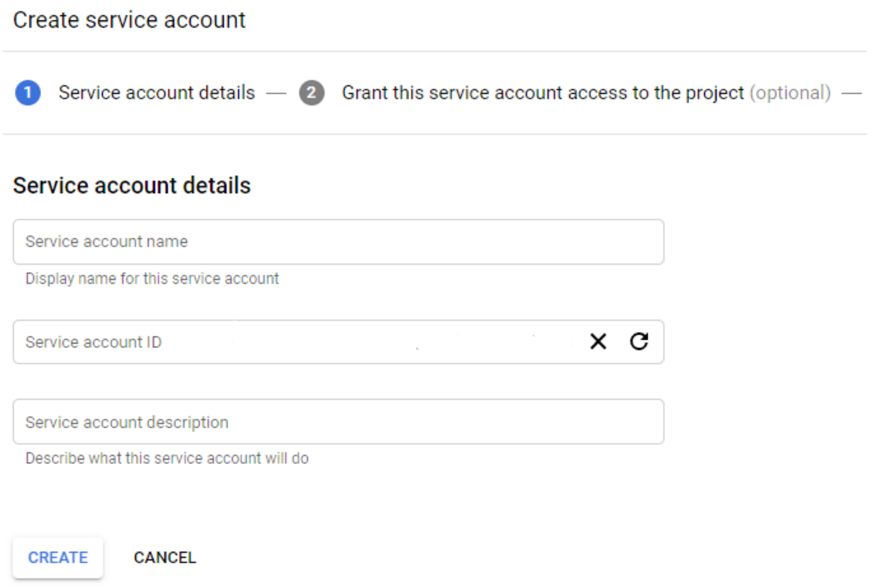
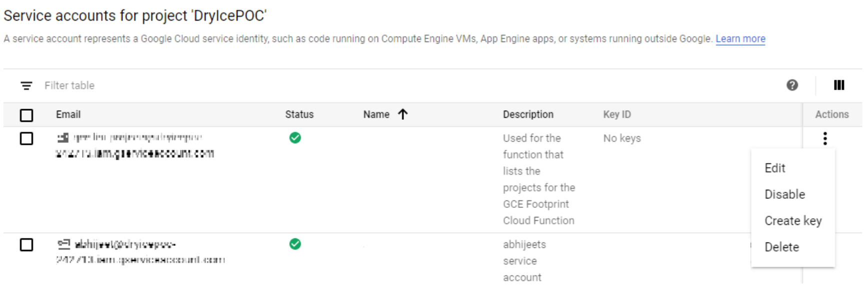

Create service account
1 Service account details — 2 Grant this service account access to the project (optional) —
Service account details
Service account name
Display name for this service account
Service account ID
Service account description
Describe what this service account will do
CREATE
CANCEL

Figure 3.20: *Define details for new Account*

6. After the creation of the service account, it will be listed in the service account page as shown in *Figure 3.21*.
7. Now, go to the **`Action`** column and select an option **`Create Key`** after clicking on the burger menu against recently created service account as shown in the following figure.

8. Then, select the **`Key`** type as **`JSON`** and click on the **`Create`** button. It will automatically download the key file in your system.



Service accounts for project 'DryIcePOC'

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. Learn more

Filter table

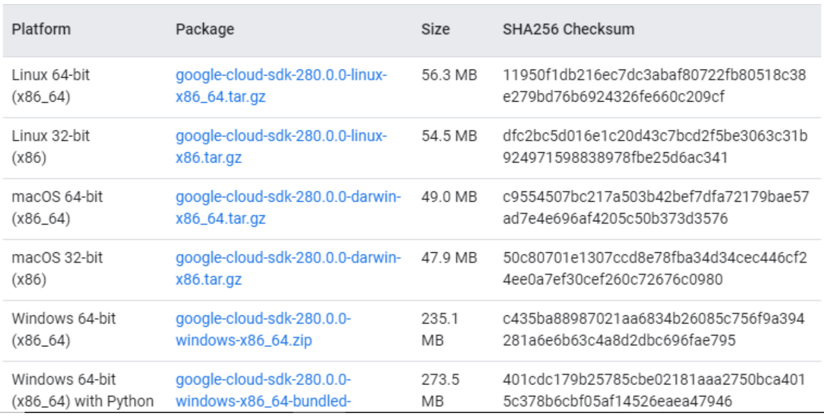
	Email	Status	Name ↑	Description	Key ID	Actions
☐	[illegible]	✓		Used for the function that lists the projects for the GCE Footprint Cloud Function	No keys	⋮
☐	[illegible]	✓		abhijeets service account		

Figure 3.21: *IAM Service Accounts List*

With this, we are done with the creation of the service account for using an AutoML Natural Language service via Google Cloud SDK. Next, please follow the following steps to install and configure python SDK of Google Cloud:

1. **Pre-requisite:** Python 3.6.x and Pip are required to be installed in your system.
2. Go to URL **https://cloud.google.com/sdk/docs/downloads-versioned-archives** and download the archive relevant to your platform as shown in *Figure* 3.22. For this book, we will go with Windows platform.

Note: To check version of windows, go to **Control Panel | System | System Type** and download corresponding versions as shown in *Figure* 3.22.



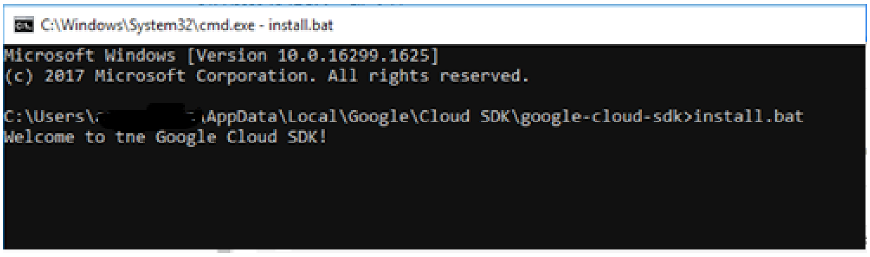
Platform	Package	Size	SHA256 Checksum
Linux 64-bit (x86_64)	google-cloud-sdk-280.0.0-linux-x86_64.tar.gz	56.3 MB	11950f1db216ec7dc3abaf80722fb80518c38e279bd76b6924326fe660c209cf
Linux 32-bit (x86)	google-cloud-sdk-280.0.0-linux-x86.tar.gz	54.5 MB	dfc2bc5d016e1c20d43c7bcd2f5be3063c31b924971598838978fbe25d6ac341
macOS 64-bit (x86_64)	google-cloud-sdk-280.0.0-darwin-x86_64.tar.gz	49.0 MB	c9554507bc217a503b42bef7dfa72179bae57ad7e4e696af4205c50b373d3576
macOS 32-bit (x86)	google-cloud-sdk-280.0.0-darwin-x86.tar.gz	47.9 MB	50c80701e1307ccd8e78fba34d34cec446cf24ee0a7ef30cef260c72676c0980
Windows 64-bit (x86_64)	google-cloud-sdk-280.0.0-windows-x86_64.zip	235.1 MB	c435ba88987021aa6834b26085c756f9a394281a6e6b63c4a8d2dbc696fae795
Windows 64-bit (x86_64) with Python	google-cloud-sdk-280.0.0-windows-x86_64-bundled-	273.5 MB	401cdc179b25785cbe02181aaa2750bca4015c378b6cbf05af14526eaea47946

Figure 3.22: *Download SDK*

3. Then, extract the downloaded archive file to any location on your file system
4. Launch the `google-cloud-sdk\install.bat` script and follow the installation instructions
5. Open the command prompt and go to the folder or path where the extracted folder with a name `google-cloud-sdk` is present
6. Run the following command:

   ```
   Install.bat
   ```

   This will install all the required command line tools necessary to work with any resource of Google Cloud platform.



```
C:\Windows\System32\cmd.exe - install.bat
Microsoft Windows [Version 10.0.16299.1625]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\[redacted]\AppData\Local\Google\Cloud SDK\google-cloud-sdk>install.bat
Welcome to tne Google Cloud SDK!
```

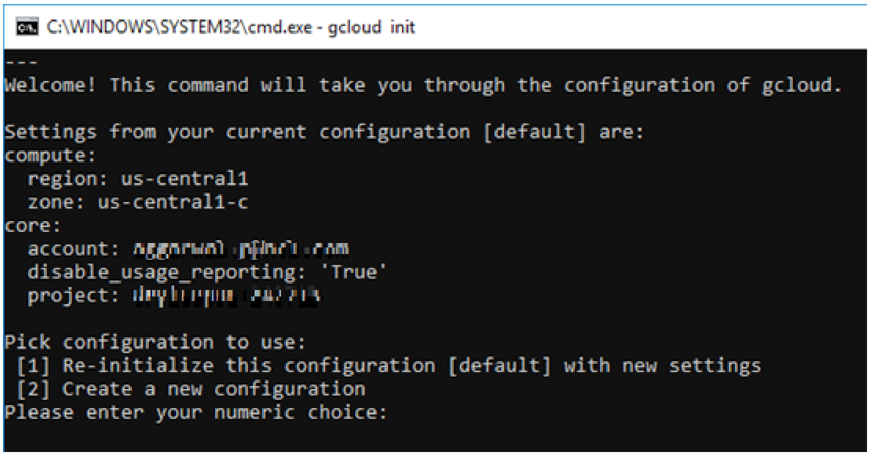
Figure 3.23: *Install SDK*

7. After the installation of Google cloud SDK, we need to initialize SDK using the `gcloud init` command. It performs the following:
   - Authorizes cloud SDK to use your account to access Google Cloud Platform
   - Setup cloud SDK configuration and setup the base properties such as the current project, Google compute engine region, and zone
8. Open the command prompt and run the following command:

   ```
   gcloud init
   ```

   This will initialize a new configuration with the default values. To change the configuration, select either one of the following options as shown in *Figure 3.24:*
   - Re-initialize this configuration[default] with new settings.

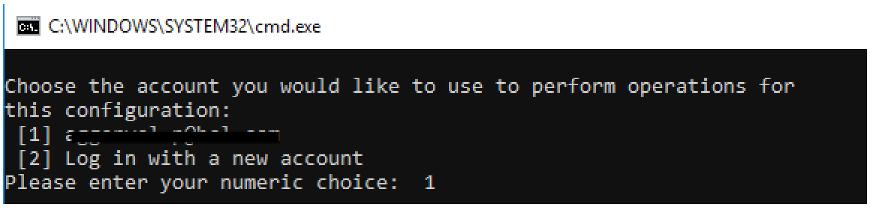
- Create a new configuration



```
C:\WINDOWS\SYSTEM32\cmd.exe - gcloud init
---
Welcome! This command will take you through the configuration of gcloud.

Settings from your current configuration [default] are:
compute:
  region: us-central1
  zone: us-central1-c
core:
  account: [redacted]
  disable_usage_reporting: 'True'
  project: [redacted]

Pick configuration to use:
 [1] Re-initialize this configuration [default] with new settings
 [2] Create a new configuration
Please enter your numeric choice:
```

Figure 3.24: *SDK Configuration –Select Configuration*

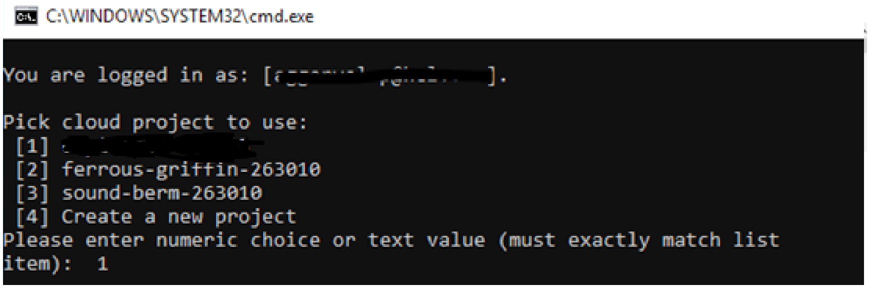
9. Next, it prompts you with two following options for the account login as shown in *Figure 3.25:*
   - Log in with a new account
   - <Name of logged-in account>
10. If you want to use the previously logged-in settings, then select the same otherwise login with a new account.
11. Then, it sets an account related properties in configuration to the specified account.



```
C:\WINDOWS\SYSTEM32\cmd.exe

Choose the account you would like to use to perform operations for
this configuration:
 [1] [redacted]
 [2] Log in with a new account
Please enter your numeric choice:  1
```

Figure 3.25: *SDK Configuration - Login*

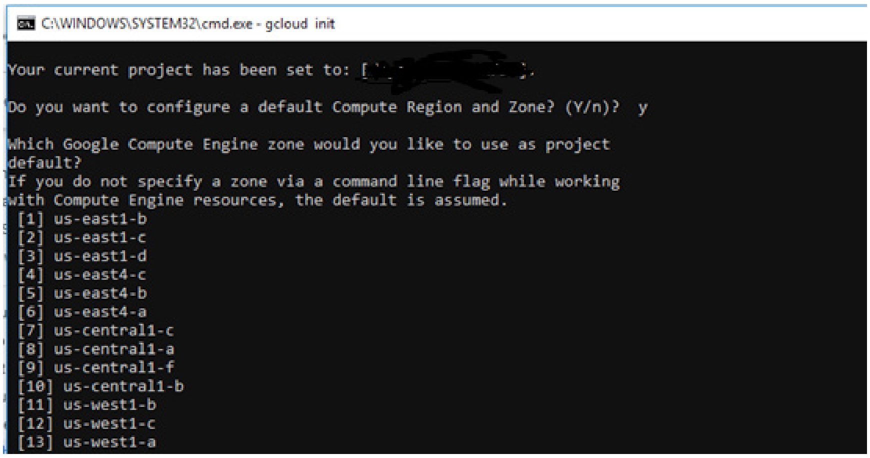
Next, it shows list of all projects associated with selected account as shown in *Figure 3.26*.



```
C:\WINDOWS\SYSTEM32\cmd.exe

You are logged in as: [[redacted]].

Pick cloud project to use:
 [1] [redacted]
 [2] ferrous-griffin-263010
 [3] sound-berm-263010
 [4] Create a new project
Please enter numeric choice or text value (must exactly match list
item):  1
```

Figure 3.26: *SDK Configuration –Select Project*

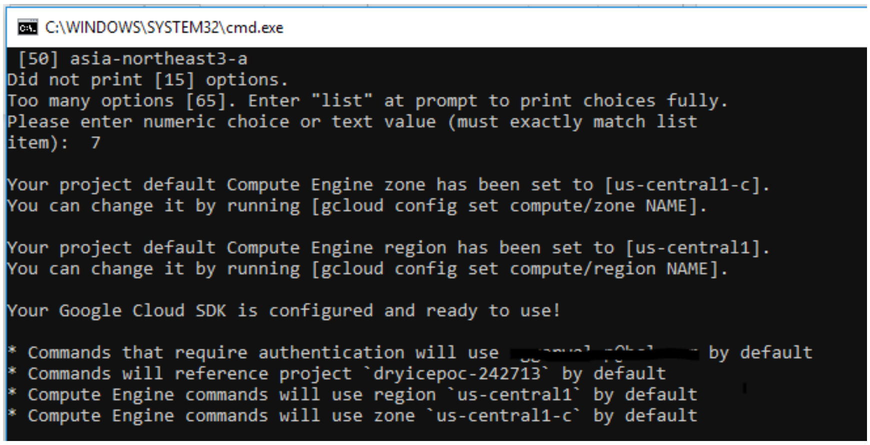
1. However, you have an option to create a new project or select an existing one. For this book, we will go with the already created project. Please enter the numeric 1 to proceed.
2. Next, it will prompt you to select the values for regions and zones as shown in *Figure 3.27*.



```
C:\WINDOWS\SYSTEM32\cmd.exe - gcloud init

Your current project has been set to: [[redacted]].

Do you want to configure a default Compute Region and Zone? (Y/n)?  y

Which Google Compute Engine zone would you like to use as project
default?
If you do not specify a zone via a command line flag while working
with Compute Engine resources, the default is assumed.
 [1] us-east1-b
 [2] us-east1-c
 [3] us-east1-d
 [4] us-east4-c
 [5] us-east4-b
 [6] us-east4-a
 [7] us-central1-c
 [8] us-central1-a
 [9] us-central1-f
 [10] us-central1-b
 [11] us-west1-b
 [12] us-west1-c
 [13] us-west1-a
```

Figure 3.27: *SDK Configuration –Select Region*

Once completed, it shows a message that your Google Cloud SDK is configured and ready to use. It also shows the current configuration of the selected project.



```
C:\WINDOWS\SYSTEM32\cmd.exe
 [50] asia-northeast3-a
Did not print [15] options.
Too many options [65]. Enter "list" at prompt to print choices fully.
Please enter numeric choice or text value (must exactly match list
item):  7

Your project default Compute Engine zone has been set to [us-central1-c].
You can change it by running [gcloud config set compute/zone NAME].

Your project default Compute Engine region has been set to [us-central1].
You can change it by running [gcloud config set compute/region NAME].

Your Google Cloud SDK is configured and ready to use!

* Commands that require authentication will use [redacted] by default
* Commands will reference project `dryicepoc-242713` by default
* Compute Engine commands will use region `us-central1` by default
* Compute Engine commands will use zone `us-central1-c` by default
```

Figure 3.28: *SDK Configuration - Confirmation*

With this, we are done with the setup of SDK of Google cloud. Next, we will see how model prediction can be accessed or fetched programmatically in Python. Please follow the following steps to configure the code to test the deployed model.

Note: The complete codebase for this exercise can be downloaded from GitHub URL **https://github.com/automl-book/AutoML/blob/master/chapter%203/automl_book.py**.

1. Import `automl` package from `google.cloud` library
2. Define the values for following parameters:
    - `project_id`: Define the value as automlgcp
    - `model_id`: Define value as 'TCN14567890876540944'
    - `content`: Define the text that needs to be classified i.e. "`System Utilization is high on server. Please suggest`"

```
from google.cloud import automl


project_id = automlgcp
model_id = 'TCN14567890876540944'
```

```
issue_description= "System Utilization is high
on server. Please Suggest "
```

Note: You need to define a value of your project-id as per your configured project, and `model_id` will be the id of your trained model.

3. Create an instance of `automl.PredictionServiceClient` class This class enables the Python code to make remote calls to AutoML services that maps to Google Cloud API.
4. Next, call a function `model_path` with `model_id, project_id` and Google Compute engine zone value as '`us-central1`'
5. Set the content type of issue description as text/plain to generate snippets to be sent via APIs as follows:

```
prediction_obj = automl.PredictionServiceClient()


# Retrieves path of the model
model_details = automl.AutoMlClient.model_path(
    project_id, 'us-central1', model_id
)


issue_snippet = automl.TextSnippet(
    content=issue_description,
    mime_type='text/plain')
payload_data = automl.ExamplePayload(text_
snippet=issue_snippet)
```

6. Now, call the predict method of AutoML with parameters as `model_details` and `payload_data`. Here, `model_details` contains the information about model, whereas `payload_data` contains details about the issue to be classified.
7. As the model provides multiple categories with their probability distribution, you need to iterate through all responses and accordingly show them to the user. In other words, you can either show the category with the highest confidence score or all the categories with their respective confidence score as follows:

```
predicted_response = prediction_obj.
predict(name=model_details, payload=payload_data)


for result_payload in predicted_response.payload:
    print(u'Predicted class name: {}'.format(
result_payload.display_name))
    print(u'Predicted class score: {}'.format(
result_payload.classification.score))
```

8. The complete code is as follows:

```
from google.cloud import automl

project_id = ‘automlgcp’
model_id = ' 'TCN14567890876540944'
issue_description = "System Utilization is high on
server. Please Suggest "

def classification():

      prediction_obj = automl.
PredictionServiceClient()

# Retrieves path of the model
model_details = automl.AutoMlClient.model_
path(project_id, 'us-central1', model_id)

issue_snippet = automl.TextSnippet(content=issue_
description,
mime_type='text/plain')
payload_data = automl.ExamplePayload(text_
snippet=issue_snippet)


predicted_response = prediction_obj.
predict(name=model_details, payload= payload_data)


for result_payload in predicted_response.payload:
```

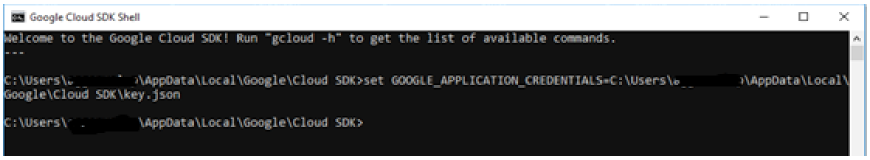
```
print(u'Predicted class name: {}'.format(result_
payload.display_name))
print(u'Predicted class score: {}'.format(result_
payload.classification.score))


if __name__==’__main__’:
      classification()
```

9. Set an environment variable `GOOGLE_APPLICATION_CREDENTIALS` value to the location of the downloaded service account key file.

10. Open the command prompt and run the following command:

    ```
    set GOOGLE_APPLICATION_CREDENTIALS=key-file-path
    ```

    where the `key-file-path` is the path of the file containing the details of the service account for an AutoML Natural Language service created earlier as shown in *Figure 3.29*.



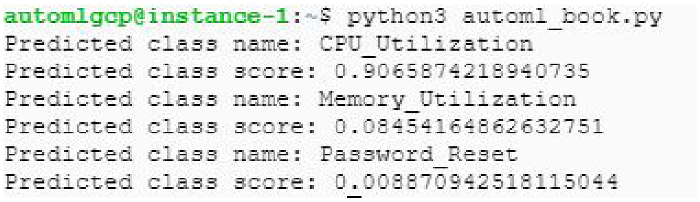
```
Welcome to the Google Cloud SDK! Run "gcloud -h" to get the list of available commands.
---

C:\Users\[redacted]\AppData\Local\Google\Cloud SDK>set GOOGLE_APPLICATION_CREDENTIALS=C:\Users\[redacted]\AppData\Local\
Google\Cloud SDK\key.json

C:\Users\[redacted]\AppData\Local\Google\Cloud SDK>
```

Figure 3.29: *Environment Configuration*

11. Now, open the command prompt and run the following command to execute the previously discussed Python code.

    ```
    python automl_book.py
    ```

12. It shows a list of categories along with the confidence score for an issue description as shown in *Figure 3.30.*



```
automlgcp@instance-1:~$ python3 automl_book.py
Predicted class name: CPU_Utilization
Predicted class score: 0.9065874218940735
Predicted class name: Memory_Utilization
Predicted class score: 0.0845416486263275l
Predicted class name: Password_Reset
Predicted class score: 0.008870942518115044
```

Figure 3.30: *Issue Categorization System – Model Prediction using Python*

We have seen how prediction from model that have been trained on an AutoML Natural Language service can be accessed by Python code. Next, we will see how this feature can be released as a REST API such that any application that can consume REST API can easily integrate with this and use the feature of categorization of a user's issue or query.

In order to create REST API for an Issue Categorization System, flask-restful package needs to be installed. Let us follow the following steps to configure flask-restful in Windows as well Linux operating system.

For Windows server

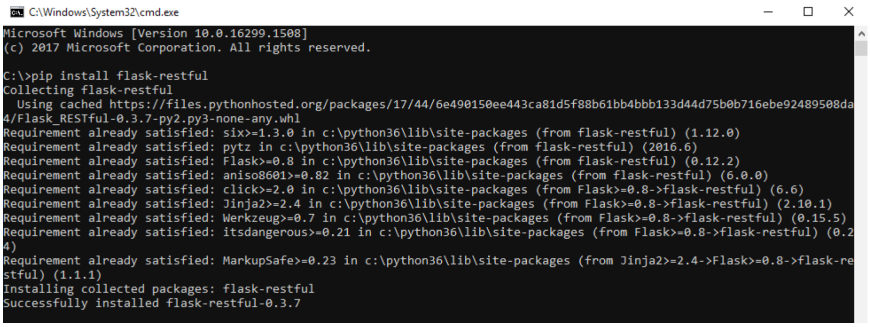
Pre-requisite: Python 3.6.x and Pip needs to be installed in your system.

Creation of REST API:

Step 1: Install Flask-RESTful

Open the command prompt and run the following command as shown in *Figure 3.31:*

```
pip install flask-restful
```



```
Microsoft Windows [Version 10.0.16299.1508]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\>pip install flask-restful
Collecting flask-restful
  Using cached https://files.pythonhosted.org/packages/17/44/6e490150ee443ca81d5f88b61bb4bbb133d44d75b0b716ebe92489508da
4/Flask_RESTful-0.3.7-py2.py3-none-any.whl
Requirement already satisfied: six>=1.3.0 in c:\python36\lib\site-packages (from flask-restful) (1.12.0)
Requirement already satisfied: pytz in c:\python36\lib\site-packages (from flask-restful) (2016.6)
Requirement already satisfied: Flask>=0.8 in c:\python36\lib\site-packages (from flask-restful) (0.12.2)
Requirement already satisfied: aniso8601>=0.82 in c:\python36\lib\site-packages (from flask-restful) (6.0.0)
Requirement already satisfied: click>=2.0 in c:\python36\lib\site-packages (from Flask>=0.8->flask-restful) (6.6)
Requirement already satisfied: Jinja2>=2.4 in c:\python36\lib\site-packages (from Flask>=0.8->flask-restful) (2.10.1)
Requirement already satisfied: Werkzeug>=0.7 in c:\python36\lib\site-packages (from Flask>=0.8->flask-restful) (0.15.5)
Requirement already satisfied: itsdangerous>=0.21 in c:\python36\lib\site-packages (from Flask>=0.8->flask-restful) (0.2
4)
Requirement already satisfied: MarkupSafe>=0.23 in c:\python36\lib\site-packages (from Jinja2>=2.4->Flask>=0.8->flask-re
stful) (1.1.1)
Installing collected packages: flask-restful
Successfully installed flask-restful-0.3.7
```

Figure 3.31: *Installation of flask-restful*

This will install the package and all their dependencies.

Step 2: build the REST API

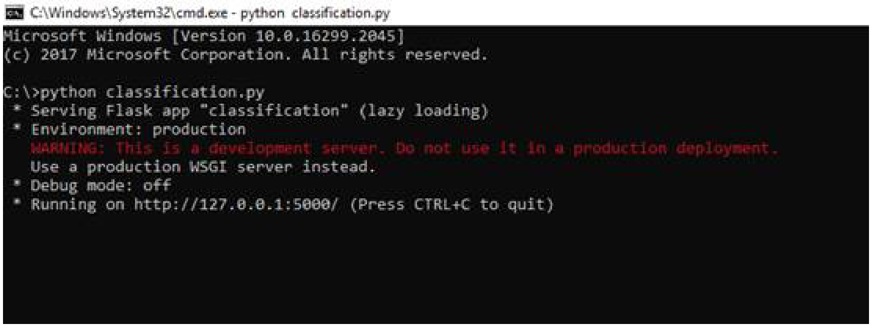
Go to GitHub URL **https://github.com/automl-book/AutoML/blob/master/chapter%203/**and download file named as `classification.py`

Step 3: Deploy Flask REST API

Using flask, deploy the REST API service.

Open the command prompt and run the following command as shown in *Figure 3.32:*

```
python classification.py
```



```
C:\Windows\System32\cmd.exe - python classification.py
Microsoft Windows [Version 10.0.16299.2045]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\>python classification.py
 * Serving Flask app "classification" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Figure 3.32: *Service Deployment*

Step 4: Response from REST API

Now, service has been hosted in the following URL

http://127.0.0.1:5000/classification

Now, you can use URL as generated in previous steps i.e. **http://127.0.0.1:5000/**classification. Flask is good for development of an environment but not for production. For Production environment, REST API should be hosted on Apache Server. Refer the following URL to deploy a service on Apache Server in Windows.

https://medium.com/@madumalt/flask-app-deployment-in-windows-apache-server-mod-wsgi-82e1cfeeb2ed

For Linux server

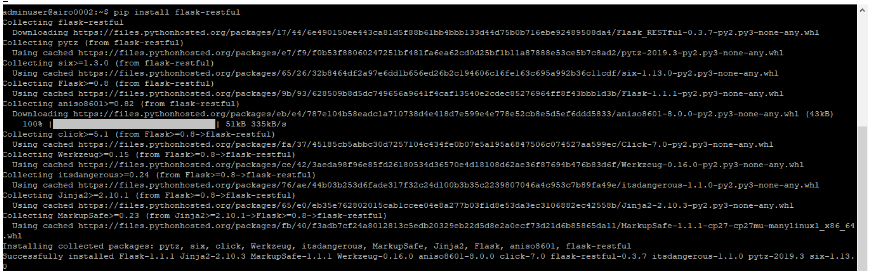
Pre-requisite: Python 3.6.x and Pip needs to be installed in your system.

Creation of REST API:

Step 1: install Flask-RESTful

To install, run the following command on Linux Shell as shown in *Figure 3.33:*

```
$ pip install flask-restful
```



```
adminuser@airo0002:~$ pip install flask-restful
Collecting flask-restful
  Downloading https://files.pythonhosted.org/packages/17/44/6e490150ee443ca81d5f88b61bb4bbb133d44d75b0b716ebe92489508da4/Flask_RESTful-0.3.7-py2.py3-none-any.whl
Collecting pytz (from flask-restful)
  Using cached https://files.pythonhosted.org/packages/e7/f9/f0b53f88060247251bf481fa6ea62cd0d25bf1b11a87888e53ce5b7c8ad2/pytz-2019.3-py2.py3-none-any.whl
Collecting six>=1.3.0 (from flask-restful)
  Using cached https://files.pythonhosted.org/packages/65/26/32b8464df2a97e6dd1b656ed26b2c194606c16fe163c695a992b36c11cdf/six-1.13.0-py2.py3-none-any.whl
Collecting Flask>=0.8 (from flask-restful)
  Using cached https://files.pythonhosted.org/packages/9b/93/628509b8d5dc749656a9641f4caf13540e2cdec85276964ff8f43bbb1d3b/Flask-1.1.1-py2.py3-none-any.whl
Collecting aniso8601>=0.82 (from flask-restful)
  Downloading https://files.pythonhosted.org/packages/eb/e4/787e104b58eadc1a710738d4e418d7e599e4e778e52cb8e5d5ef6ddd5833/aniso8601-8.0.0-py2.py3-none-any.whl (43kB)
    100% |████████████████████████████████| 51kB 335kB/s
Collecting click>=5.1 (from Flask>=0.8->flask-restful)
  Using cached https://files.pythonhosted.org/packages/fa/37/45185cb5abbc30d7257104c434fe0b07e5a195a6847506c074527aa599ec/Click-7.0-py2.py3-none-any.whl
Collecting Werkzeug>=0.15 (from Flask>=0.8->flask-restful)
  Using cached https://files.pythonhosted.org/packages/ce/42/3aeda98f96e85fd26180534d36570e4d18108d62ae36f87694b476b83d6f/Werkzeug-0.16.0-py2.py3-none-any.whl
Collecting itsdangerous>=0.24 (from Flask>=0.8->flask-restful)
  Using cached https://files.pythonhosted.org/packages/76/ae/44b03b253d6fade317f32c24d100b3b35c2239807046a4c953c7b89fa49e/itsdangerous-1.1.0-py2.py3-none-any.whl
Collecting Jinja2>=2.10.1 (from Flask>=0.8->flask-restful)
  Using cached https://files.pythonhosted.org/packages/65/e0/eb35e762802015cab1ccee04e8a277b03f1d8e53da3ec3106882ec42558b/Jinja2-2.10.3-py2.py3-none-any.whl
Collecting MarkupSafe>=0.23 (from Jinja2>=2.10.1->Flask>=0.8->flask-restful)
  Using cached https://files.pythonhosted.org/packages/fb/40/f3adb7cf24a8012813c5edb20329eb22d5d8e2a0ecf73d21d6b85865da11/MarkupSafe-1.1.1-cp27-cp27mu-manylinux1_x86_64
.whl
Installing collected packages: pytz, six, click, Werkzeug, itsdangerous, MarkupSafe, Jinja2, Flask, aniso8601, flask-restful
Successfully installed Flask-1.1.1 Jinja2-2.10.3 MarkupSafe-1.1.1 Werkzeug-0.16.0 aniso8601-8.0.0 click-7.0 flask-restful-0.3.7 itsdangerous-1.1.0 pytz-2019.3 six-1.13.
0
```

Figure 3.33: *Installation of flask-restful*

This will install the package and their dependencies.

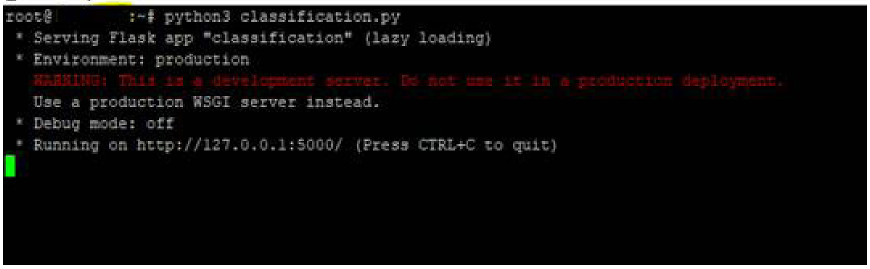
Step 2: Build the REST API

Go to GitHub URL **https://github.com/automl-book/AutoML/blob/master/chapter%203/** and download file named as `classification.py`.

Step 3: deploy Flask REST API

To deploy the REST API service using flask, run the following command on Linux Shell as shown in *Figure 3.34*:

```
$ python3classification.py
```



```
root@        :~# python3 classification.py
 * Serving Flask app "classification" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Figure 3.34: *Service Deployment – An Example*

Step 4: Using REST API

Now, the service has been hosted in the following URL:

http://127.0.0.1:5000/classification

Now, you can use the URL as generated in the previous step i.e. **http://127.0.0.1:5000/classification**. Flask is good for the development of an environment but not for production. For Production environment, API should be hosted on Apache Server. Refer the following URL to deploy a service on Apache Server in Linux.

https://www.codementor.io/abhishake/minimal-apache-configuration-for-deploying-a-flask-app-ubuntu-18-04-phu50a7ft

Please follow the following steps to release the feature of an Issue Categorization system as REST API.

Note: The complete codebase for this exercise can be downloaded from Github URL **https://github.com/automl-book/AutoML/blob/master/chapter%203/classification.py**.

1. Create a file with the name `classification.py`.
2. Copy the following code and paste in the file created in the first step and save it.

```
from google.cloud import automl
from flask import Flask, request
import json

app=Flask(__name__)

@app.route ("/classification", methods=['POST'])
def Classification():
    try:
        json_data = request.get_json(force=True)
        project_id = json_data["project_id"]
        location = json_data["location"]
        model_id = json_data["model_id"]
issue_description = json_data["content"]
        result = []
        prediction_obj = automl.
PredictionServiceClient()
        model_details = automl.AutoMlClient.model_
```

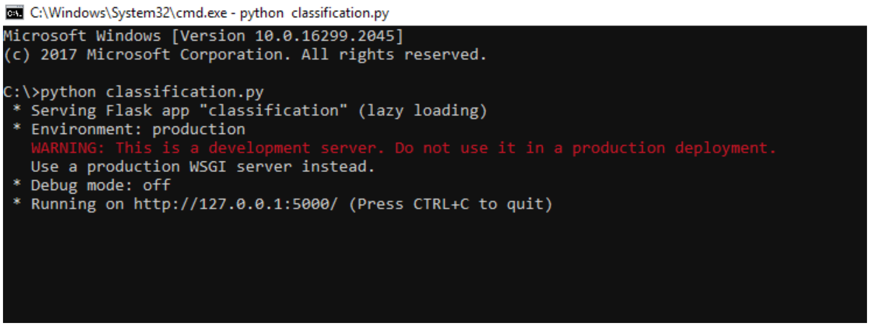
```
path(
            project_id, location, model_id)
issue_snippet = automl.TextSnippet(content=issue_
description,mime_type='text/plain')
payload_data = automl.ExamplePayload(text_
snippet=issue_snippet)
predicted_response = prediction_obj.
predict(name=model_details, payload=payload_data)
        classification = {}
        for result_payload in predicted_response.
payload:
            classification["Class_Name"] = result_
payload.display_name
            classification["Class_Score"] = result_
payload.classification.score
            result.append(classification)
        result = {"results" : result}
        result = json.dumps(result)
        return result
    except Exception as e:
        return {"Error": str(e)}


if __name__ == "__main__" :
    app.run(port="5000")
```

3. The previous code processes an input passed to an API, calls a function to fetch prediction from the model that has been trained in an AutoML Natural Language Service, and sends the response as an API response.

4. Open the command prompt and run the following command:

```
python classification.py
```

This will start a service on **http://127.0.0.1:5000/**as follows:



```
C:\Windows\System32\cmd.exe - python  classification.py
Microsoft Windows [Version 10.0.16299.2045]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\>python classification.py
 * Serving Flask app "classification" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Figure 3.35: *Service Deployment - Issue Categorization System*

1. Now, to test the REST API, we are going to use Postman. This is a REST API client that is used to test the API URL.
2. First, go to the following URL to download the Postman Tool and install on your system:

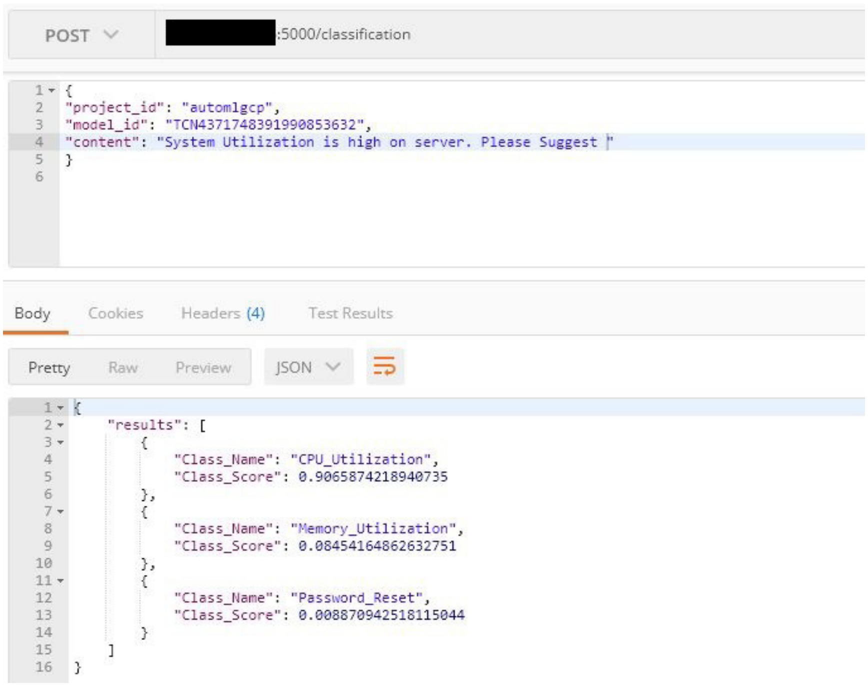
   https://www.postman.com/downloads/.

3. After the installation, Test the foll owing URL and sample request JSON that is being sent to Classification API and response JSON that will be received as a response from the API as shown in the following *Figure 3.36:*

   URL: http://127.0.0.1:5000/classification

   Classification System – Sample Input Request json:

```
{
"project_id": “automlgcp",
 "location": "us-central1",
"model_id": "TCN1928046415863349248",
"content": "System Utilization is high on server.
Please Suggest "
}
Classification System – Sample Output Response json:
{
    "results": [
        {
            "class": "CPU_Utilization",
            "score": 0.906587421890735
        },
        {
            "class": "Password_Reset",
            "score": 0.08454164862632751
        },
        {
            "class": "Memory_Utilization",
            "score": 0.008870942518115044
        }
    ]
}
```



POST [redacted]:5000/classification

```
{
"project_id": "automlgcp",
"model_id": "TCN4371748391990853632",
"content": "System Utilization is high on server. Please Suggest "
}
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview JSON

```
{
    "results": [
        {
            "Class_Name": "CPU_Utilization",
            "Class_Score": 0.9065874218940735
        },
        {
            "Class_Name": "Memory_Utilization",
            "Class_Score": 0.08454164862632751
        },
        {
            "Class_Name": "Password_Reset",
            "Class_Score": 0.008870942518115044
        }
    ]
}
```

Figure 3.36: *Issue Categorization System –API Results*

This brings us to a conclusion of the Categorization system use case. Lets now move forward with other use cases starting with Sentiment Analysis System.

Sentiment Analysis System

Sentiment Analysis is a sub-field of NLP that identifies opinion or sentiments of the given text across blogs, reviews, news, and so on.

It actually helps organization to know about the acceptance of their products and their sentiments toward the same. It is also useful to identify the hate text on social media sites such as Twitter to know the population's mood toward the topic of discussion.

Sentiment Analysis can even help companies to plan for their product releases based on the mood and sentiment of people from particular demographics. Sentiment analysis is being used majorly in conversation systems to generate a response to be given to a user based on the way a question or query has been raised by the user.

Therefore, in this book, we will walk through the implementation of Sentiment Analysis system using the state-of-art technologies used by Google AutoML. For this use case, we have used Natural Language service of AutoML, where we will upload our labeled dataset that contains sentences and corresponding sentiment value. Here, AutoML will automatically find the best model architecture using Google's architecture search algorithms to be used based on our dataset. The following are the complete details of the implementation of such a system.

This implementation would also go through the stages of AutoML workflow as follows:

- **Prepare Training Data:** In this step, feature engineering such as missing value imputation, data validation, one-hot encoding, and more will be performed over a raw dataset.
- **Create and import Dataset:** In this step, prepared data will be uploaded, and a dataset will be created as required by AutoML. This dataset works as an input to model the training process.
- **Model Training:** In this step, AutoML uses Google's architecture search algorithm to analyze your dataset and find the best model to train.
- **Evaluate Model:** In this step, trained model will be evaluated over the test data. If it is not efficient, then it also offers you to train the different version of the same model but with different values of hyperparameter.
- **Model Prediction:** In this step, the trained model will be used to predict the category for any new issue or query raised by the user. Applications such as an Automated Ticket Assignation System can be integrated with it to predict the

category for any new issue or query and accordingly assign it to a relevant support person.

Data Analysis

For this implementation, the sentiment analysis dataset (Name: `train.csv`) can be downloaded from GitHub URL **https://github.com/automl-book/AutoML/tree/master/chapter%203**.

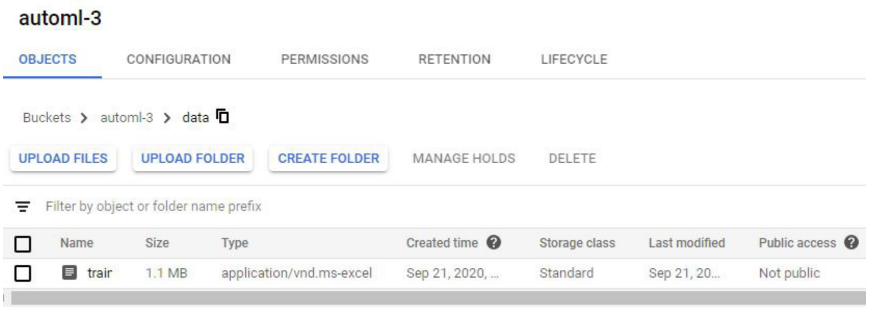
This dataset has phrases that have been extracted from Rotten Tomatoes dataset. In this dataset, each individual sentence has already been converted into multiple phrases. These phrases have been labeled with various sentiment values such as negative, positive, neutral, and so on. However, each and every phrase belongs to only single sentiment value. A dataset contains 29999 records which are distributed across five categories, namely, negative, somewhat negative, neutral, somewhat positive and positive. Each of these categories has been converted into their numeric form as follows:

- 0: Negative
- 1: Somewhat negative
- 2: Neutral
- 3: Somewhat positive
- 4: Positive

Each class is having different number of variations as shown in the following table:

Classes	No of Variations
negative	1102
somewhat negative	4815
neutral	16536
somewhat positive	6016
positive	1530
Total	**29999**

However, as per the best practices, the training data should contain equal or nearly equal number of variations for each class. But in our use case, each class contains different number of variations. As the class neutral has almost 15 times a greater number of phrases as compared with the class negative. Therefore, this dataset suffers from class-imbalance problem and would degrade the performance of class with a smaller number of variations. However, AutoML identifies class imbalance problem in a dataset and solves it automatically. This is an important feature of AutoML, where it provides a guided way to engineers to evaluate and solve the data science problems. For this use case, we will upload the dataset file into Google Cloud Storage bucket that has been created in *Chapter 2* as shown in *Figure 3.37:*



Name	Size	Type	Created time	Storage class	Last modified	Public access
trair	1.1 MB	application/vnd.ms-excel	Sep 21, 2020, ...	Standard	Sep 21, 20...	Not public

Figure 3.37: *Sentiment Analysis –Data Upload*

In the next section, we will see how the previously discussed dataset will be used to build the sentiment analysis System using an AutoML Natural Language Service on Google Cloud Platform.

Model Training

Initially, we will first enable an AutoML Natural Language service from Google cloud. Follow the following steps to enable an AutoML Natural Language service:

1. Go to URL **https://cloud.google.com/** and login with your credentials.

2. Click on the burger Menu and select the Natural Language option as shown in *Figure 3.38:*

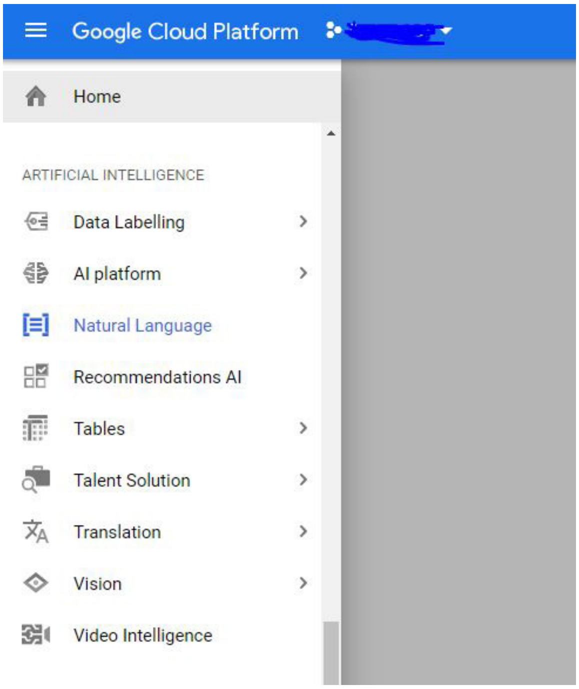
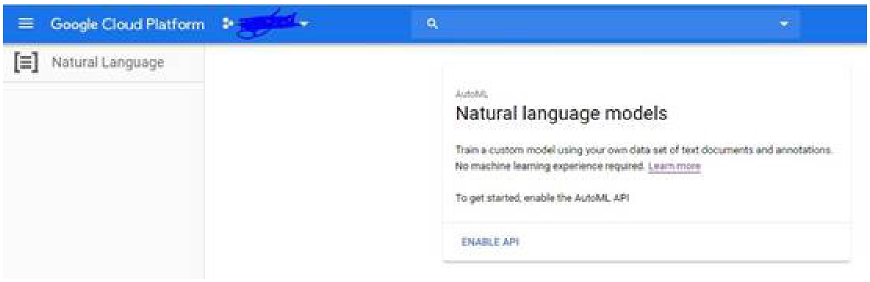


Figure 3.38: *Sentiment Analysis – Natural Language Service*

3. This will populate the screen as shown in *Figure 3.39*. Next, click on the **`ENABLE API`** button to enable the AutoML Natural Language API.



Natural language models

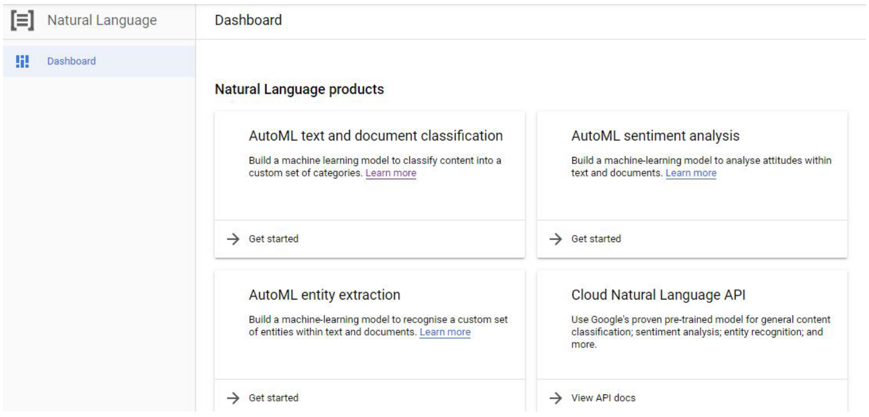
Train a custom model using your own data set of text documents and annotations. No machine learning experience required. Learn more

To get started, enable the AutoML API

ENABLE API

Figure 3.39: *Sentiment Analysis – Natural Language Models*

4. Once enabled, it will populate the screen that shows details of features provided by Natural Language Service as shown in *Figure 3.40*.


Natural Language

Dashboard

Dashboard

Natural Language products

AutoML text and document classification

Build a machine learning model to classify content into a custom set of categories. Learn more

Get started

AutoML sentiment analysis

Build a machine-learning model to analyse attitudes within text and documents. Learn more

Get started

AutoML entity extraction

Build a machine-learning model to recognise a custom set of entities within text and documents. Learn more

Get started

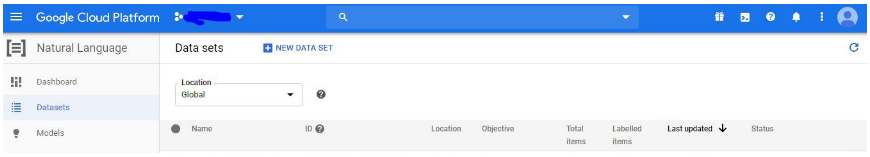
Cloud Natural Language API

Use Google's proven pre-trained model for general content classification; sentiment analysis; entity recognition; and more.

View API docs

Figure 3.40: *Sentiment Analysis – Natural Language Products*

5. Next, click on the card named as AutoML sentiment analysis option to build a custom model for **`Sentiment Analysis System`**. You will be redirected to a **`Dataset`** page as shown in *Figure 3.41*:


Google Cloud Platform [redacted]

Natural Language

Data sets

NEW DATA SET

Dashboard

Datasets

Models

Location: Global

Name	ID	Location	Objective	Total items	Labelled items	Last updated	Status

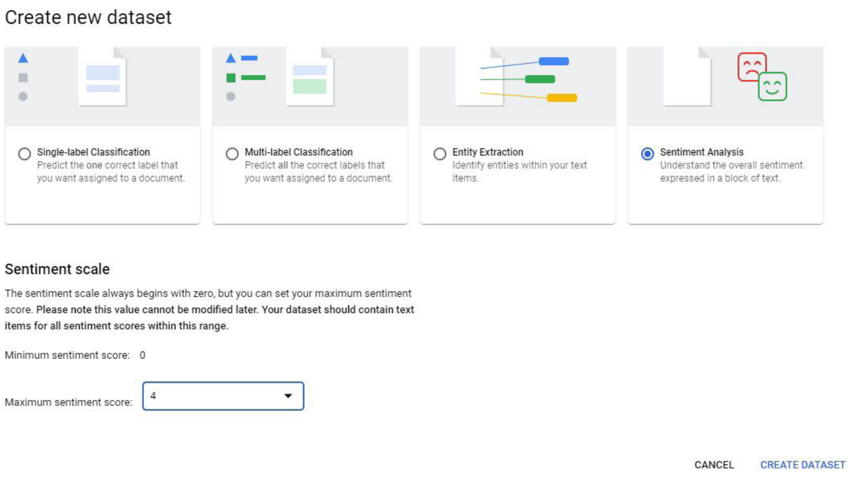
Figure 3.41: *Sentiment Analysis – Dataset*

6. Now, to upload your dataset click on the **`NEW DATA SET`** link and you will be redirected to a screen as shown in *Figure 3.42*.

7. Next, enter the values for various fields as follows:

   - **`Data set name:`** name of the dataset.
   - For this use case, value will be `Sentiment`.
   - **`Location:`** define the geographical region in which your data will be stored
   - You can select Global or European. For this use case, select Global as a value of the location.

- **`Model Objective:`** select model objective as Sentiment Analysis


Create new dataset

○ Single-label Classification
Predict the one correct label that you want assigned to a document.

○ Multi-label Classification
Predict all the correct labels that you want assigned to a document.

○ Entity Extraction
Identify entities within your text items.

◉ Sentiment Analysis
Understand the overall sentiment expressed in a block of text.

Sentiment scale

The sentiment scale always begins with zero, but you can set your maximum sentiment score. **Please note this value cannot be modified later. Your dataset should contain text items for all sentiment scores within this range.**

Minimum sentiment score: 0

Maximum sentiment score: 4

Figure 3.42: *Sentiment Analysis – Create Dataset*

8. Sentiment analysis of an AutoML Natural Language service automatically selects the value of parameter minimum sentiment score as 0. However, the value of parameter maximum sentiment score should be equal to the largest number assigned to any category in your dataset. In our case, the category positive has been mapped to Value 4. Therefore, the value of this field will be 4. Please note that these values cannot be modified later on. If you want to modify them, then you need to go with the new configuration of sentiment analysis feature with the new values.

9. Click on the **`CREATE DATA SET`** button.

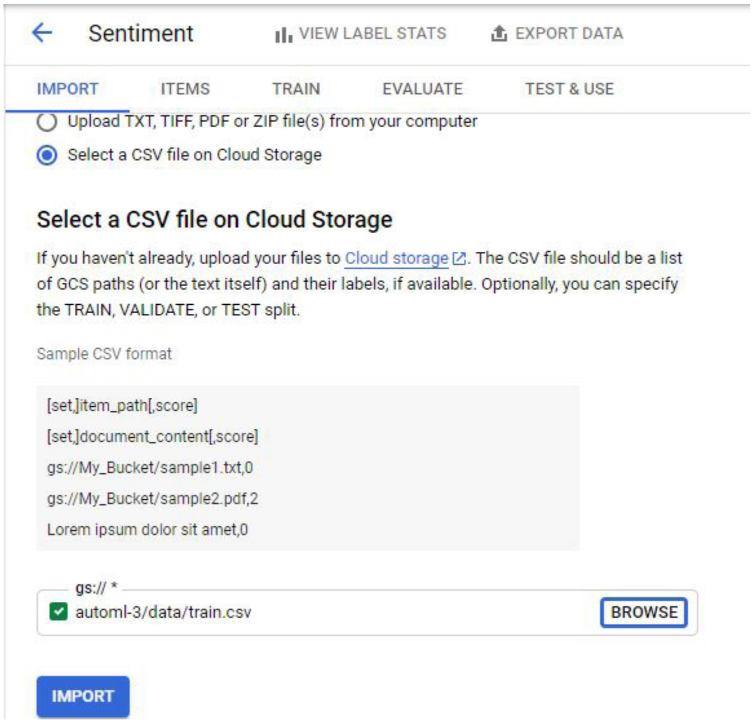
10. Next, AutoML provides the following three options depending upon your dataset file format and location to upload.

    - **`Upload CSV file from your system:`** If the dataset file is in CSV format and resides in your system, then select this option.

    - **`Upload TXT, PDF, or ZIP files from your system:`** If the dataset file is in TXT, PDF, or ZIP format and resides in your system, then select this option.

- **`Select a CSV file from cloud storage:`** If the dataset file is in CSV format and stored in cloud storage service such as object storage. For this use case, we will go with this option.

Then, choose **`Select a CSV file on Cloud Storage`** and click on the **`IMPORT`** button as shown in *Figure 3.43*:



Upload TXT, TIFF, PDF or ZIP file(s) from your computer [✓ Select a CSV file on Cloud Storage]

Select a CSV file on Cloud Storage

If you haven't already, upload your files to Cloud storage. The CSV file should be a list of GCS paths (or the text itself) and their labels, if available. Optionally, you can specify the TRAIN, VALIDATE, or TEST split.

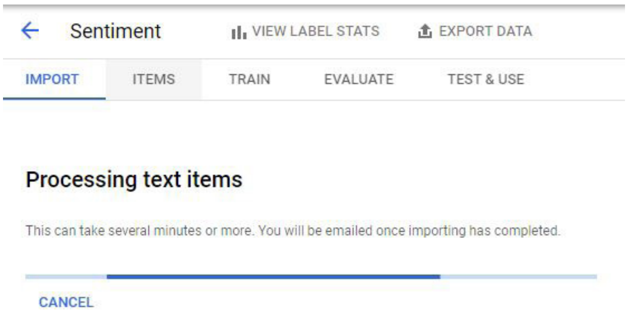
Sample CSV format

```
[set,]item_path[,score]
[set,]document_content[,score]
gs://My_Bucket/sample1.txt,0
gs://My_Bucket/sample2.pdf,2
Lorem ipsum dolor sit amet,0
```

gs:// * automl-3/data/train.csv

Figure 3.43: *Sentiment Analysis – File Upload*

11. Once you click on the **`IMPORT`** button, then you will be able to see the progress of your dataset as shown in *Figure 3.44*:



Processing text items

This can take several minutes or more. You will be emailed once importing has completed.

Figure 3.44: *Sentiment Analysis – File Upload Process*

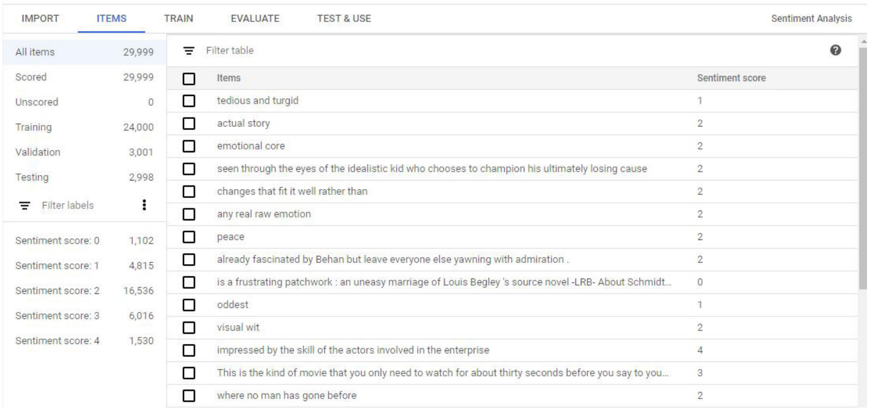
12. This process may take a few minutes. You will receive an email on your registered email address associated with your project as soon as the upload process has successfully completed.

13. Once the data has been successfully uploaded, then you can see the following details about your dataset under **`ITEMS`** tab as shown in *Figure 3.45*.

    - **`ALL items:`** shows the total number of records in your dataset
    - **`Labeled:`** shows the total number of labeled records in your dataset
    - In our dataset, all records or variations has already been labeled. So, it is equal to All items.
    - **`Unlabeled:`** shows the total number of unlabeled records in your dataset

    In our dataset, not even a single record is unlabeled. So, its value is 0.

    - **`Training:`** Shows the number of records to be used during the model training
    - **`Validation:`** Shows the number of records to be used during the validation step
    - **`Testing:`** Shows the number of records to be used during the testing phase



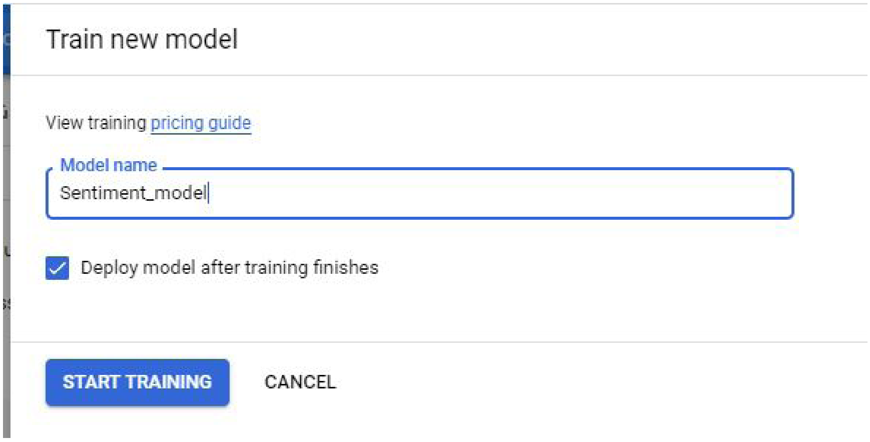
All items	29,999
Scored	29,999
Unscored	0
Training	24,000
Validation	3,001
Testing	2,998
Sentiment score: 0	1,102
Sentiment score: 1	4,815
Sentiment score: 2	16,536
Sentiment score: 3	6,016
Sentiment score: 4	1,530

Items	Sentiment score
tedious and turgid	1
actual story	2
emotional core	2
seen through the eyes of the idealistic kid who chooses to champion his ultimately losing cause	2
changes that fit it well rather than	2
any real raw emotion	2
peace	2
already fascinated by Behan but leave everyone else yawning with admiration .	2
is a frustrating patchwork : an uneasy marriage of Louis Begley 's source novel -LRB- About Schmidt...	0
oddest	1
visual wit	2
impressed by the skill of the actors involved in the enterprise	4
This is the kind of movie that you only need to watch for about thirty seconds before you say to you...	3
where no man has gone before	2

Figure 3.45: *Sentiment Analysis – Data Analysis*

14. During the training of Artificial Intelligence models, the dataset has to be divided into three parts, namely, training, validation, and test set. By default, an AutoML Natural Language service uses 80% of the records for training, 10% for validation set, and the rest 10% for testing the set. It also takes care of the class imbalance problem (if any) in the dataset.

15. Next, click on the **`TRAIN`** tab to start the model training process as shown in *Figure 3.46:*



Train new model

View training pricing guide

Model name: Sentiment_model

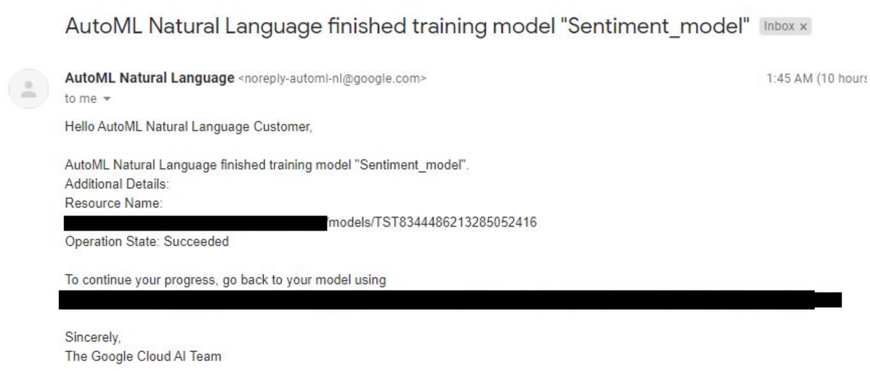
✓ Deploy model after training finishes

Figure 3.46: *Sentiment Analysis – Model Training*

16. Define the Model name as `Sentiment_model` and check the box against the Deploy model after the training gets finished if you want to deploy the model automatically as soon as the training is finished.

17. Then, click on the **`START TRAINING`** button to start model training.

18. The model training process may take several hours depending on the size of the dataset. You will receive an email on your registered email address as soon as the model training is completed as shown in the *Figure 3.47*:



AutoML Natural Language finished training model "Sentiment_model" Inbox ×

AutoML Natural Language <noreply-automl-nl@google.com>
to me

1:45 AM (10 hour

Hello AutoML Natural Language Customer,

AutoML Natural Language finished training model "Sentiment_model".
Additional Details:
Resource Name:
[redacted]models/TST8344486213285052416
Operation State: Succeeded

To continue your progress, go back to your model using
[redacted]

Sincerely,
The Google Cloud AI Team

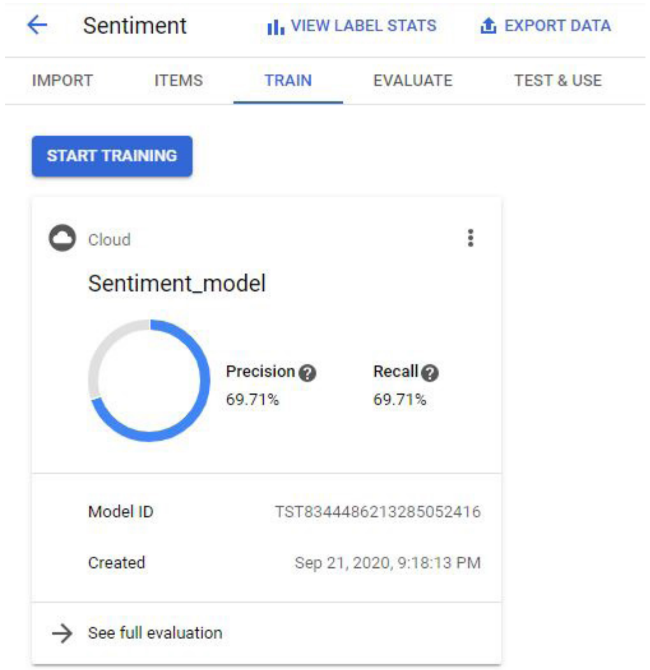
Figure 3.47: *Sentiment Analysis – Model Training Confirmation*

Please note that AutoML automatically uses the validation set to optimize the model hyperparameter values and selects the best model for you. You will receive an email only once AutoML has trained the best model for you.

Click on the link mentioned in the email body that you must have received after the completion of model training to check the performance of the model. As mentioned earlier, AutoML uses the test dataset to evaluate the performance and quality of the trained model.

Model Evaluation

1. Go to the **TRAIN** tab to check your model's performance once the training has been completed as shown in *Figure 3.48*.



Sentiment_model

Precision 69.71%

Recall 69.71%

Model ID TST8344486213285052416

Created Sep 21, 2020, 9:18:13 PM

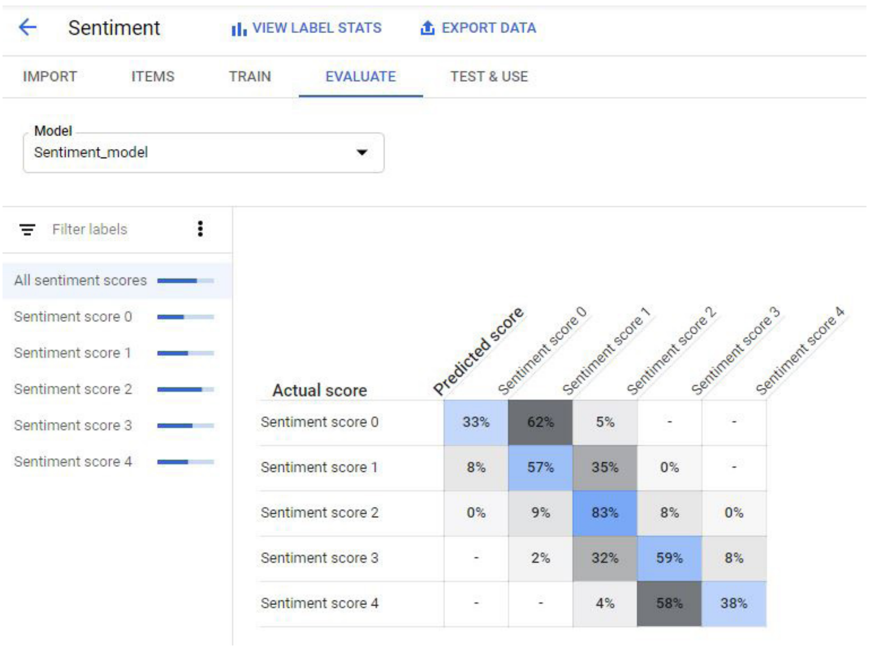
***Figure 3.48:** Sentiment Analysis – Model Training Results*

2. Click on the **See full evaluation** link or go to the **EVALUATE** tab to see the model's performance for each of the sentiment category as shown in *Figure 3.48.*

3. Sentiment analysis model is also being evaluated using the performance metrics such as precision and recall. We have already discussed these two things in detail in our previous use case.

4. For sentiment analysis, AutoML automatically selects the confidence threshold value to calculate recall and precision based on their best model and nature of the data. In this use case, precision and recall both are equal to 69.7%. It means that our trained model is almost 70% precise while doing

prediction and able to classify 70% of data points i.e. phrases correctly.



Actual score	Predicted score: Sentiment score 0	Sentiment score 1	Sentiment score 2	Sentiment score 3	Sentiment score 4
Sentiment score 0	33%	62%	5%	-	-
Sentiment score 1	8%	57%	35%	0%	-
Sentiment score 2	0%	9%	83%	8%	0%
Sentiment score 3	-	2%	32%	59%	8%
Sentiment score 4	-	-	4%	58%	38%

Figure 3.49: *Sentiment Analysis – Model Evaluation*

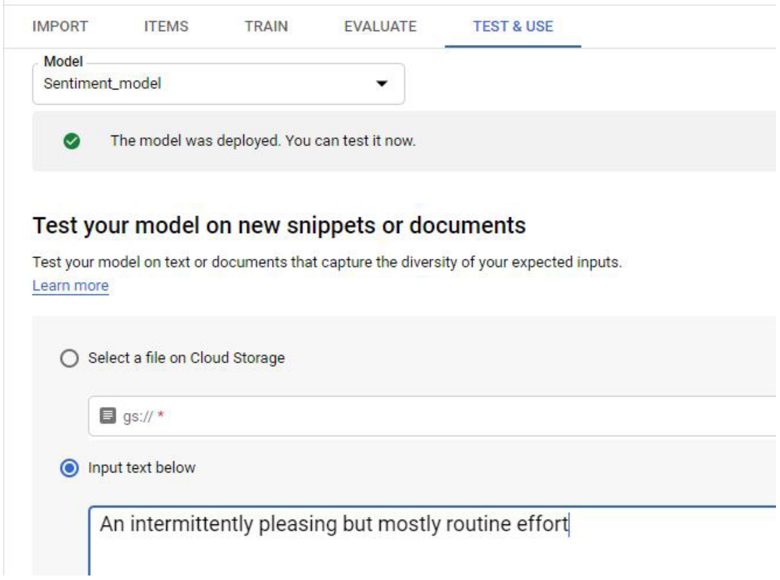
5. The preceding matrix tells us how frequently each of the sentiment label has been predicted correctly and incorrectly by a model. This is being highlighted with a different color coding as follows:

   - Blue, represents sentiment label that has been predicted correctly by a model. This is being represented in intensity of color. As an example, for label Sentiment score 2, intensity of color blue is higher than for the label Sentiment score 0, as former has been predicted correctly for most of the cases than latter one.

   - Grey, represents sentiment label that has been predicted incorrectly by a model. As an example, for the label Sentiment score 0, intensity of color grey is higher than for the label Sentiment score 2, as former has been predicted incorrectly for most of the cases than latter one.

Model Prediction

Once the model has been trained and evaluated, you can check the prediction from the model on a new query or issue by clicking on the **`TEST & USE`** tab as shown in *Figure 3.50*.



The model was deployed. You can test it now.

Test your model on new snippets or documents

Test your model on text or documents that capture the diversity of your expected inputs.
Learn more

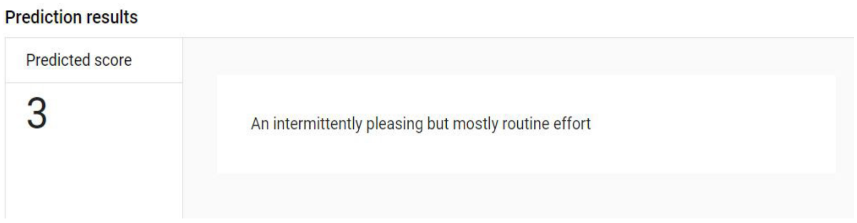
An intermittently pleasing but mostly routine effort

Figure 3.50: *Sentiment Analysis – Model Prediction*

Define a query or phrase as `An intermittently pleasing but mostly routine effort` in input text in the following field as shown in *Figure 3.50*.

Next, click on the **`PREDICT`** button to see the model results.

As a query or feedback from a user is not showing that highly positive sentiments from a user, that's why for this statement, model has returned score as three i.e. somewhat positive.



Predicted score	
3	An intermittently pleasing but mostly routine effort

Figure 3.51: *Sentiment Analysis – Model Prediction Results*

REST API configuration

Now, consider a scenario where you want to expose this model to an external application. Next, consider a scenario where you need to enable the sentiment analysis for the conversation system such that our model can identify sentiments of a user based on a query or response given by the user. This enables the conversation system to respond back to the user based on the sentiments of a user.

Now, in order to use this feature in the conversation system, a sentiment analysis model has to be exposed as a REST API such that whenever a user submits his query or issue, it will make an API call to identify the sentiments of a user. It involves following steps:

- Creation of service account for accessing an AutoML Natural Language Service.
- Installation and configuration of Google Cloud SDK
- Code development to get a model prediction for a query or an issue description.
- Create REST API to get a model prediction

However, we have already discussed about the service account creation and Google Cloud SDK installation and configuration in our previous use case. Therefore, we will go with the code development and REST API creation for the sentiment analysis model. Please follow the following steps to configure the code to test the deployed model.

Note: The complete codebase for this exercise can be downloaded from Github URL **https://github.com/automl-book/AutoML/blob/master/chapter%203/sentiment.py**.

1. Import automl package from `google.cloud` library
2. Define the values for the following parameters:
    - `project_id`: Define the value as automlgcp
    - `model_id`: Define the value as `TST8344486213285052416`
    - `content`: Define the text for the sentiment analysis i.e. `An intermittently pleasing but mostly routine effort`

```
from google.cloud import automl
```

```
project_id = ‘automlgcp’
model_id = 'TST8344486213285052416’
sentence= "Anintermittently pleasing but mostly
routine effort"
```

Note: You need to define the value of project-id as per your configured project, and `model_id` will be the ID of your trained model.

3. Create an instance of `automl.PredictionServiceClient` class. This class enables Python code to make remote calls to AutoML services that maps to Google Cloud API.
4. Next, call a function `model_path` with `model_id, project_id` and Google Compute engine zone value as `us-central1`.
5. Set the content type of the issue description as ‘text/plain’ to generate snippets to be sent via APIs as follows:

```
sentiment_prediction_model = automl.
PredictionServiceClient()


# Retrieves path of the model
model_details = automl.AutoMlClient.model_path(
    project_id, 'us-central1', model_id
)


sentence_snippet = automl.TextSnippet(
    content=sentence,
    mime_type='text/plain')
payload_data = automl.ExamplePayload(text_
snippet=sentence_snippet)
```

6. Now, call a predict method of AutoML with parameters as `model_details` and `payload_data`

   Here, `model_details` contains the information about the model while the `payload_data` contains details about the sentence for which the sentiment needs to be identified.

```
predicted_response = sentiment_prediction_model.
predict(name=model_details, payload=payload_data)
for payload_result in predicted_response.payload:
result = "Predicted sentiment score:{}".
format(payload_result.text_sentiment.sentiment)
return (result)
```

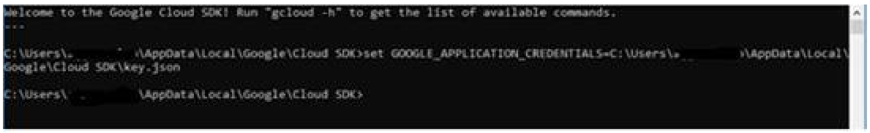
7. The complete code is as follows:

```
from google.cloud import automl

project_id = ‘automlgcp’
model_id = ' TST8344486213285052416’
sentence = "An intermittently pleasing but mostly
routine effort"

def predictSentiment():

sentiment_prediction_model = automl.
PredictionServiceClient()

# Retrieves path of the model
model_details = automl.AutoMlClient.model_path(
project_id, 'us-central1', model_id)

sentence_snippet = automl.
TextSnippet(content=sentence,
    mime_type='text/plain')
payload_data = automl.ExamplePayload(text_
snippet=sentence_snippet)
predicted_response=sentiment_prediction_model.
predict (name=model_details, payload=payload_data)

for payload_result in predicted_response.payload:
result= "Predicted sentimentscore:{}".
format(payload_result.text_sentiment.sentiment)
return (result)


if __name__==’__main__’:
      predictSentiment()
```

8. Set an environment variable **`GOOGLE_APPLICATION_CREDENTIALS`** value to the location of the downloaded service account key file.
9. Open the command prompt and run the following command:

   set **`GOOGLE_APPLICATION_CREDENTIALS=key-file-path`**

   where `key-file-path` is the path of the file containing the details of the service account for an AutoML Natural Language service created earlier.



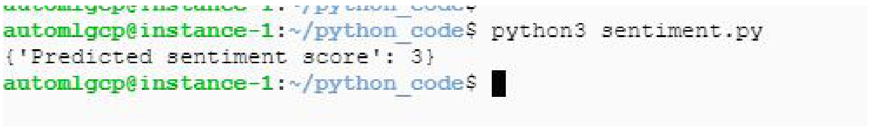
```
Welcome to the Google Cloud SDK! Run "gcloud -h" to get the list of available commands.
...

C:\Users\[redacted]\AppData\Local\Google\Cloud SDK>set GOOGLE_APPLICATION_CREDENTIALS=C:\Users\[redacted]\AppData\Local\
Google\Cloud SDK\key.json

C:\Users\[redacted]\AppData\Local\Google\Cloud SDK>
```

Figure 3.52: *Sentiment Analysis – Environment Readiness*

10. Now, open the command prompt and run the following command to execute the previously discussed Python code.

    `python3sentiment.py`

11. It shows an identified sentiment score for a sentence as shown in *Figure 3.53.*



```
automlgcp@instance-1:~/python_code$ python3 sentiment.py
{'Predicted sentiment score': 3}
automlgcp@instance-1:~/python_code$
```

Figure 3.53: *Sentiment Analysis – Model Prediction using Python*

We have seen how the prediction from the model that have been trained on an AutoML Natural Language service can be accessed by Python code. Next, we will see how this feature can be released as a REST API such that any application that can consume REST API can easily integrate with this and use the feature of the sentiment analysis for sentences.

In order to create REST API for Sentiment Analysis System, flask-restful package needs to be installed that we have already discussed in our previous use case. Now, please follow the following steps to release the features of the sentiment analysis system as REST API.

Note: The complete codebase for this exercise can be downloaded from Github URL **https://github.com/automl-book/AutoML/blob/master/chapter%203/predict.py**.

1. Create a file with the name `predict.py`.
2. Copy the following code and paste in the file created the first step and save it:

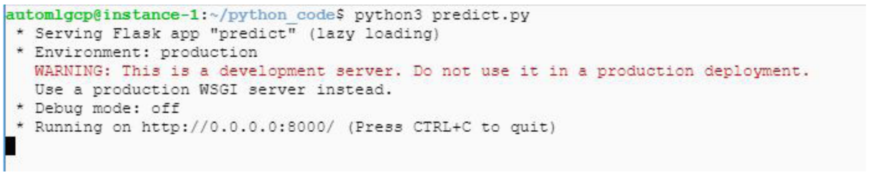
```
from google.cloud import automl
from flask import Flask, request
import json

app=Flask(__name__)

@app.route ("/sentiment", methods=['POST'])
def predictSentiment():
    try:
        json_data = request.get_json(force=True)
        project_id = json_data["project_id"]
model_id = json_data["model_id"]
sentence = json_data["content"]
result = []
sentiment_prediction_model = automl.
PredictionServiceClient()

     # Retrieves path of the model
             model_details = automl.AutoMlClient.
model_path(
           project_id, 'us-central1', model_id)
             sentence_snippet = automl.TextSnippet(
content=sentence,
    mime_type='text/plain')
       payload_data = automl.ExamplePayload(text_
snippet=sentence_snippet)
       predicted_response=sentiment_prediction_model.
predict (name=model_details,    payload=payload_
data)

       for payload_result in predicted_response.
payload:
             result= "Predicted sentiment score: {}"
.format(payload_result.text_sentiment.sentiment)
```

```
                    return (result)

        except Exception as e:
            return {"Error": str(e)}


    if __name__ == "__main__" :
        app.run(port="5000")
```

3. The preceding code processes an input passed to an API, calls a function to fetch prediction from the model that has been trained in an AutoML Natural Language Service, and sends the response as an API response.
4. Open the command prompt and run the following command:

   ```
   Python3 predict.py
   ```

   This will start a service on **http://127.0.0.1:5000/** as shown in *Figure 3.54*.



```
automlgcp@instance-1:~/python_code$ python3 predict.py
 * Serving Flask app "predict" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:8000/ (Press CTRL+C to quit)
```

Figure 3.54: *Service Deployment - Sentiment Analysis*

5. Now, to test the REST API, we are going to use Postman. This is a REST API client that is used to test the API URL.
6. First, go to the following URL to download the Postman Tool and install on your system https://www.postman.com/downloads/
7. After the installation, test the following URL and sample request JSON that is being sent to the Classification API and response JSON that will be received as a response from the API as shown in the following *Figure 3.55.*

   URL: **http://127.0.0.1:5000/sentiment**

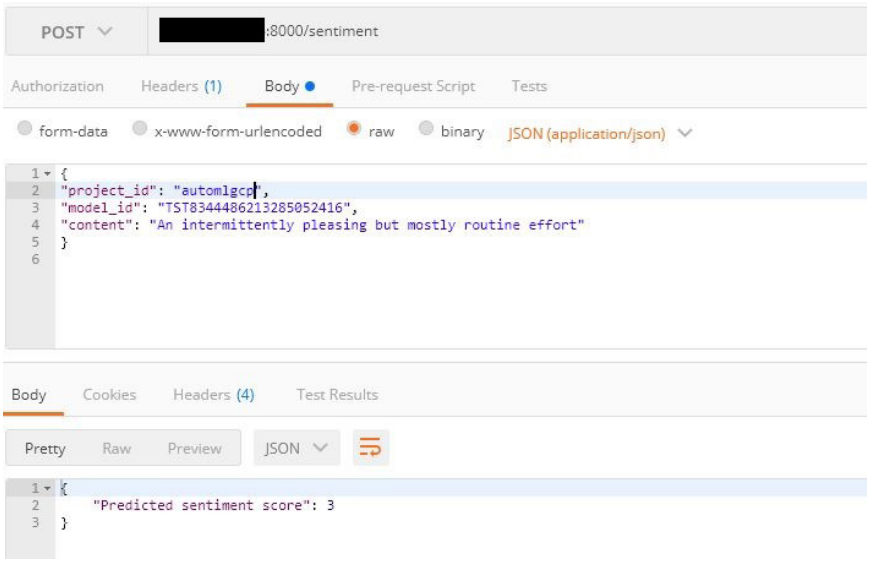
   Sentiment Analysis System – Sample Input Request json:

   ```
   {
   "project_id": “automlgcp",
   "model_id": "TCN1928046415863349248",
   ```

```
"content": "An intermittently pleasing but mostly
routine effort"
}
```

Sentiment Analysis System – Sample Output Response json:

```
{
“Predicted sentiment score”: 3
}
```



POST [redacted]:8000/sentiment

Authorization | Headers (1) | Body ● | Pre-request Script | Tests

form-data | x-www-form-urlencoded | ● raw | binary | JSON (application/json)

```
{
"project_id": "automlgcp",
"model_id": "TST8344486213285052416",
"content": "An intermittently pleasing but mostly routine effort"
}
```

Body | Cookies | Headers (4) | Test Results

Pretty | Raw | Preview | JSON

```
{
    "Predicted sentiment score": 3
}
```

***Figure 3.55:** Sentiment Analysis – Model Prediction using API*

Conclusion

In this chapter, we have discussed about how an AutoML Natural Language service can be used to train the custom model for an Issue Categorization System and Sentiment Analysis System with our dataset. We have also seen how features of this system can be released as REST API to be used by other applications or systems. In the next chapter, we will discuss on how Google’s AI Platform can be used to train custom model developed by you.

Chapter 4
Google AI Platform

Google's AI platform provides a framework for the developers and data scientist to build an end-to-end machine learning pipeline. It enables them to define their own algorithms or model architecture or even code it for their own tasks and datasets. It uses TensorFlow, a Google's deep-learning library for custom model development. In this chapter, we will explain how Google's AI platform can be used for training a custom model for tasks that can't be implemented using AutoML.

Structure

In this chapter, we will cover the following topics:

- Design and implementation of an Issue Categorization system using Google's AI platform
    - o Data analysis for an Issue Categorization system
    - o Model training using Local Machine and AI platform
    - o REST API configuration to release the features of this system for other applications

Objectives

After studying this chapter, you should be able to:

- Understand how Google's AI platform can be used for training of machine learning models
- Understand how TensorFlow can be used to develop custom machine learning model
- Understand how a machine learning model that has been trained on Google's AI platform can be used for prediction in your application via REST API

Introduction

In the previous chapter, we discussed about how an AutoML Natural Language service can be used for implementing various tasks such as Single-Label Classification, Multi-Label Classification, Entity Recognition, and Sentiment Analysis. AutoML also enables you to train a custom model on a dataset provided by you. Here, AutoML uses Google's model architecture search algorithm to find the best model according to your dataset. As an example, if you are developing a text classification system, then you just need to provide your dataset containing text along with their associated labels. Rest is being done by an AutoML Natural Language service starting from identification of model architecture to actual model training to version of model that can be used in production.

However, model architecture search algorithm only searches through repositories of models available with Google. What if you want to include some custom features as well in your dataset? An AutoML Natural Language service doesn't allow you to modify your model architecture. As an example, in our Issue Categorization System, we want to generate embedding of tokens extracted from an issue or a query description using GloVE embedding method, then we can't do that with AutoML. In this case, we can leverage Google's AI platform, where we can add this feature for our Issue Categorization System that will be developed using TensorFlow.

Design of an Issue Categorization system using Google's AI platform

In this section, we will cover an implementation of an Issue Categorization system using Google's AI platform. This is the same use case that we have implemented in *Chapter 3* using AutoML.

Google's AI platform is quite flexible in terms of model training, accessing training, validation and test dataset from various sources such as BigQuery, and Cloud Storage service. Also, you can develop and build models in your system, provided you have sufficient computing resources available with you, alternatively you can train it into AI platform directly.

In *Chapter 3*, we have seen how AutoML selects a model automatically based on your dataset. However, with Google AI platform, we will see how we can train a custom model for an Issue Categorization system and use it to classify new issues or queries.

An end-to-end machine learning pipeline involves the following steps:

- **Data Gathering:** It involves gathering required data from various sources such as database, File repositories such as Dropbox, Google Drive, and so on. As an example, for our Issue Categorization system, data can be gathered or sourced from any database that stores an issue or a query description asked by the users.
- **Data Processing and Splitting:** Gathered data generally contains some unwanted issues such as missing values, noisy information, and more. In this step, feature engineering will be done over the dataset and splitting of dataset into train, validation, and test set.
- **Model Training:** In this step, a model will be trained over training dataset. Size and nature of your dataset affects the time required to train a model. It varies from a few minutes to hours to days or weeks as well.
- **Test Model:** In this step, a trained model will be tested over a validation set. If a system identifies that there is an overfitting or underfitting in the model, then it again goes to model training step with new values of hyperparameters. The

objective of this step is to ensure that trained model should generalize well over dataset.

- **Evaluate Model:** In this step, a trained model will be evaluated over a test set that has not been seen during the model training and validation step.
- **Deploy Model:** In this step, a trained model will be deployed for the production use.

Each of these steps can be aligned with each other as shown in *Figure 4.1.*

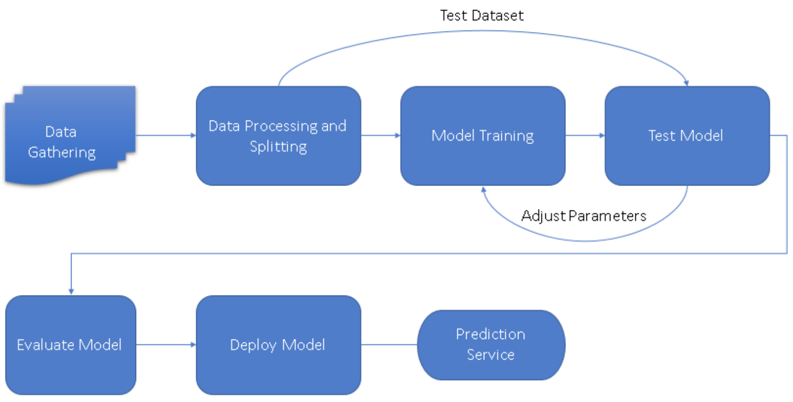


Figure 4.1: *Workflow for Model Deployment*

Data Analysis

Next, we will see an implementation of our Issue Categorization System using GloVE for vector generation of token extracted from an issue or a query description. **GloVE** is an unsupervised technique for representing words into low-dimensional vector space. It is one of the most widely used method of word embedding (representing word or token in lower dimensional space). It takes into consideration global and local statistics of a corpus as compared with the other word embedding methods like word2vec. As an example, words i.e. man and woman are similar in context, as they both are talking about human being. However, in the same context, they are opposite to each other and require knowledge of an entire corpus such as brother–sister etc. to deduce the difference between them.

For training a GloVE model, all the variations of query or issue descriptions will be used to generate a vector representation of all the words or token or phrases.

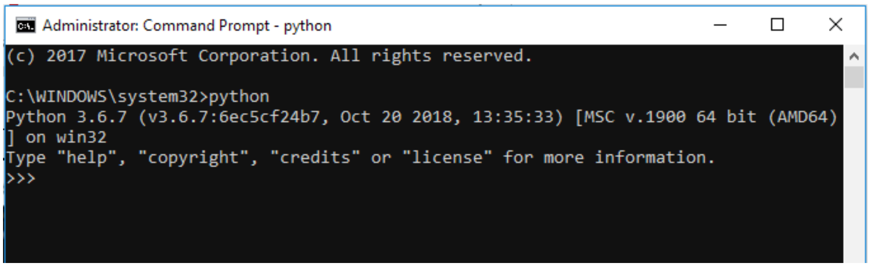
Model Training

Please follow the following steps to install the required pre-requisites for an Issue Categorization system:

1. Make sure python3 is installed on your system

   Open the command prompt and run the following command to check if python is installed or not

   `Python`



Administrator: Command Prompt - python

```
(c) 2017 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>python
Python 3.6.7 (v3.6.7:6ec5cf24b7, Oct 20 2018, 13:35:33) [MSC v.1900 64 bit (AMD64)
] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 4.2: *Python Console*

   This will start Python console in the command prompt. If Python is not installed in your system, then download and install Python as per your Operating System using the following link: **https://www.python.org/downloads/**

2. Next, install Jupyter Notebook that we will use to code.

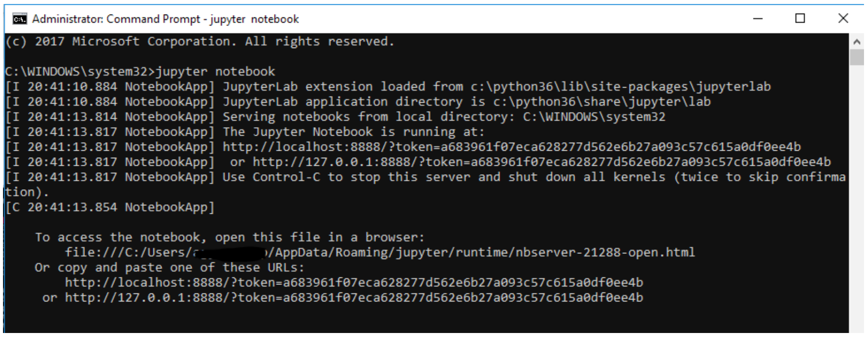
   Open the command prompt and run the following command:

   `pip install notebook`

3. To start the `jupyter` notebook, open the command prompt and run the following command:

   `jupyter notebook`

   The notebook will start in your default browser with the host address as localhost and port number as 8888 along with a unique token id. Now, you can start writing the code as mentioned in the subsequent steps.



```
(c) 2017 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>jupyter notebook
[I 20:41:10.884 NotebookApp] JupyterLab extension loaded from c:\python36\lib\site-packages\jupyterlab
[I 20:41:10.884 NotebookApp] JupyterLab application directory is c:\python36\share\jupyter\lab
[I 20:41:13.814 NotebookApp] Serving notebooks from local directory: C:\WINDOWS\system32
[I 20:41:13.817 NotebookApp] The Jupyter Notebook is running at:
[I 20:41:13.817 NotebookApp] http://localhost:8888/?token=a683961f07eca628277d562e6b27a093c57c615a0df0ee4b
[I 20:41:13.817 NotebookApp]  or http://127.0.0.1:8888/?token=a683961f07eca628277d562e6b27a093c57c615a0df0ee4b
[I 20:41:13.817 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirma
tion).
[C 20:41:13.854 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/[redacted]/AppData/Roaming/jupyter/runtime/nbserver-21288-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=a683961f07eca628277d562e6b27a093c57c615a0df0ee4b
     or http://127.0.0.1:8888/?token=a683961f07eca628277d562e6b27a093c57c615a0df0ee4b
```

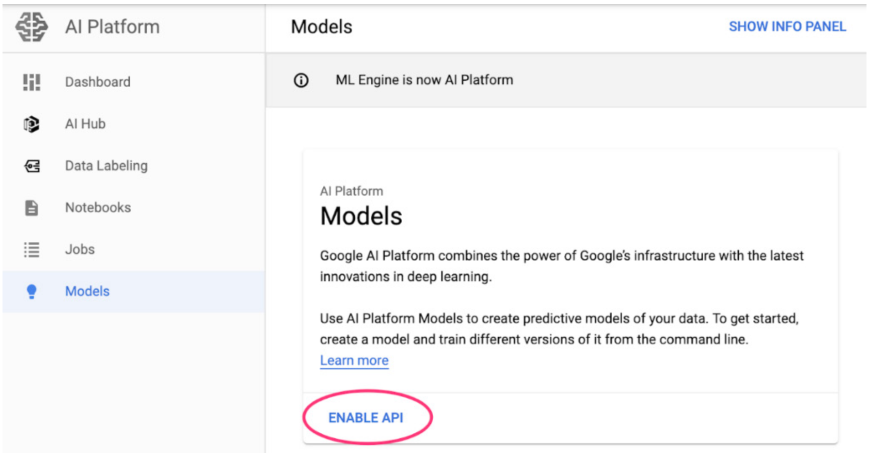
Figure 4.3: *Jupyter Notebook Console*

4. You can also use Google Colab Notebook for the development of a model. It provides a fast and free environment to run your python code in case your system doesn't have sufficient resources available. You can also use the GPU's and TPU's with no cost but for a limited time (12hrs) in Google Colab. You just need a Google account to login to Google Colab Notebook. For this book, we will be using Google Colab Notebook to demonstrate an Issue Categorization system.

5. Go to URL **https://cloud.google.com/** and login with your credentials

6. Select an AI platform option from the left side menu bar

   Then, you will be redirected to a page as shown in *Figure 4.4.*



AI Platform

Models

Google AI Platform combines the power of Google's infrastructure with the latest innovations in deep learning.

Use AI Platform Models to create predictive models of your data. To get started, create a model and train different versions of it from the command line.

Figure 4.4: *Enable API for AI platform*

7. Next, click on the **`ENABLE API`** button to enable AI platform
8. Login to your Google account and click on **https://colab.research.google.com**, you will see a page as shown in *Figure 4.5:*
9. Next, click on the **`NEW NOTEBOOK`** button to start a new notebook

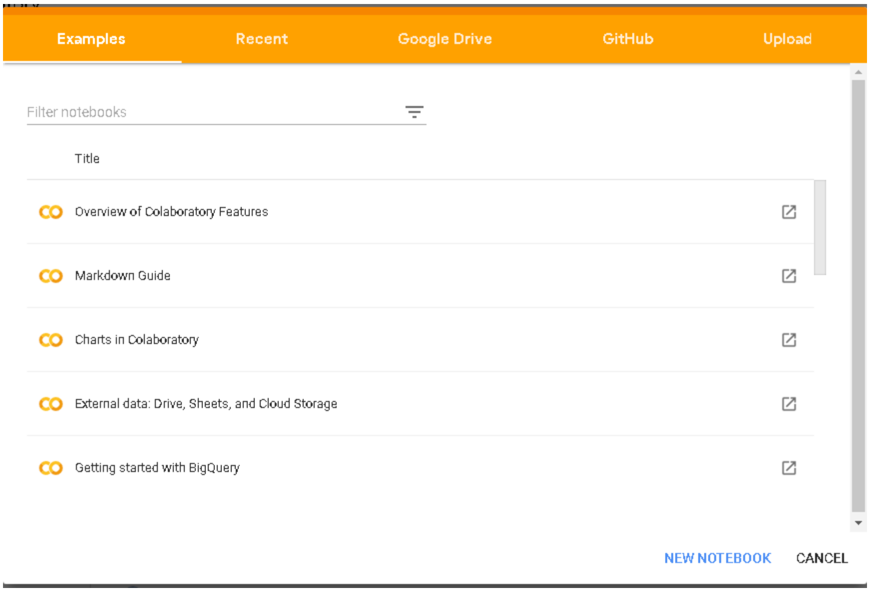


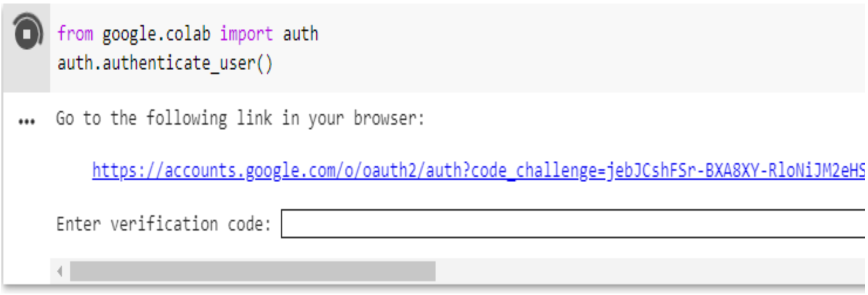
Figure 4.5: *Google Colab*

As mentioned in the previous chapter as well, Google cloud authentication must be done first before using any services of Google Cloud Platform.

10. Use Google Colab's in-built module named as "`auth`" for authentication and authorization to Google Cloud Platform.
11. Next, run the following commands in Google's Colab notebook to install `gcloud` and google cloud `sdk`.

```
pip install google-cloud-automl
pip install gcloud
```

12. Import the `auth` package from `google.colab` and call the function `authenticate_user ()` as shown in *Figure 4.6:*



```
from google.colab import auth
auth.authenticate_user()
```

```
Go to the following link in your browser:

    https://accounts.google.com/o/oauth2/auth?code_challenge=jebJCshFSr-BXA8XY-RloNiJM2eHS

Enter verification code:
```

Figure 4.6: *Google Colab – Authenticate User*

13. Next, open the link in a new tab as shown in figure. This redirects a user to the registered Google account for the verification code.
14. Copy the verification code that you can see in the new tab and paste in the text box against the label Enter Verification Code and press the *Enter* key.
15. Run the following command in Google Colab to define your project as follows:

    ```
    !gcloud config set project <PROJECT_NAME>
    ```

    For this use case, the project name will be `Categorization System`.

Note: Please provide a name of your project that you have created in Google Cloud Platform.

16. Upload your own dataset for this use case. However, Colab provides few datasets by default. These are present in the folder named as sample_data under the **`Files`** tab as shown in *Figure 4.7.*

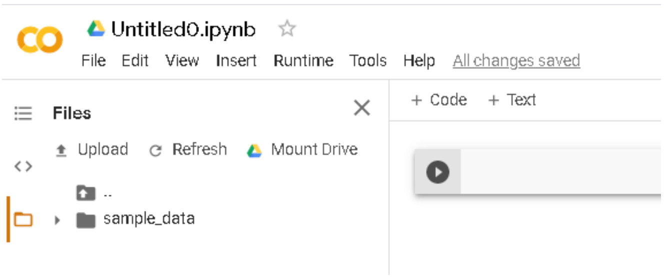


Figure 4.7: *Upload Data Google Colab - I*

17. Now, we will first upload the data to Google colab.
18. Go to the `Upload` button under the `Files` tab as shown in *Figure 4.8*.

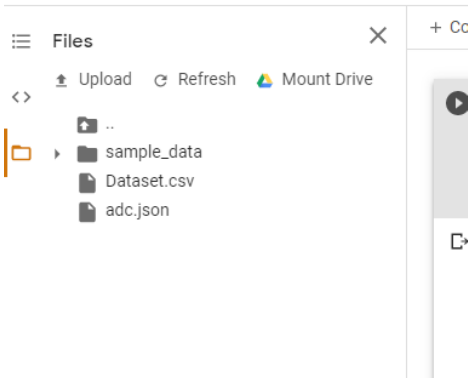


Figure 4.8: *Upload Data Google Colab - II*

19. You can also mount Google drive attached to your Gmail account. This enables storage of all of your dataset files in Google Drive and avoids loss of data due to environment reset. Please ensure that each column in your dataset file should have a header.
20. Download the dataset file named as `Dataset.csv` for this use case from GitHub URL **https://github.com/automl-book/AutoML**
21. Our dataset file has two columns namely, `Text` and `Category`, where the `Text` represents an issue or a query description raised by a user, and the `Category` shows a category to which an issue or a query has been categorized.
22. Next, the dataset will be divided into training, validation, and test set. As a best practice, 80% of the records will go to the training set, 10% will go to the validation, and the rest 10% will go to the test set as follows:

Note: The codebase for this exercise can be downloaded from GitHub URL **https://github.com/automl-book/AutoML/tree/master/chapter%204**

```
import pandas aspd

 data_set = pd.read_csv('Dataset.csv', header = None)
```

```
data_set.columns = ['Text', 'Category']
train_set = data_set.sample(frac=0.8)
data_set.drop(train_set.index, axis=0, inplace=True)
valid_set = data_set.sample(frac=0.5)
data_set.drop(valid_set.index, axis=0, inplace=True)
test_set = data_set
```

Then, classes or categories in your dataset should be converted into the numeric form. As an example, the category `CPU_Utilization` will be mapped to `0`, `Password_Reset` will be mapped 1 and `Memory_Utilization` will be mapped to 2. You can also limit the number of tokens out of the vocabulary to be considered during training. This is generally being done to reduce the computation requirement during the model training with very less effect on the accuracy of model. However, you can also use the entire vocabulary set for the training. For our use case, we have used `20000` words from the token vocabulary. Next, each query or issue description should not be more than a specific length. This is required as a pre-requisite for the model training. For our use case, we have restricted it to `50` as follows:

```
CLASSES = {'CPU_Utilization': 0, 'Password_
Reset': 1, 'Memory_Utilization': 2}
top_tokens = 20000
max_len = 50
```

So, if the length of an issue or a query description is greater than `50`, then it will be truncated and the system will use the truncated description as an input. However, if the length of the description is less than 50, then it will add do padding to the description with the token `<PAD>` to make it a length of `50`. Please note that you should select pad token such that it shouldn't interfere with words or tokens from your vocabulary. Otherwise, this will have a negative impact on the model performance. For our use case, we will append the token `<PAD>` at the end of each of the descriptions if the length is less than `50`.

23. Now, `import numpy` package. This will be used to convert categories into numpy array as follows:

```
    import numpy as np
    def data_map(df):
        return list(df['Text']), np.
    array(df['Category'].map(CLASSES))


    train_text, train_labels = data_map(train_set)
    valid_text, valid_labels = data_map(valid_set)
    test_text, test_labels = data_map(test_set)
```

Next, the embedded representation of each of these description text will be generated using the GloVE method. As GloVE generates a vector representation of word or token. Therefore, each description must be tokenized first into tokens or words before generating the embeddings. Once a word or token embeddings are available, then we can generate sentence embeddings using any methods as follows:

- **Average of word vector:** One of the common approach is to perform average on a vector representation of words in a description. For this book, we will go with this approach.
- **Sentence Embeddings:** You can also use sentence embedding methods such as Universal Sentence Encoder, BERT (Bidirectional Encoder Representations from Transformers), and so on to generate a vector representation of the description.

Import necessary packages as follows:

```
import tensorflow as tf
from tensorflow.keras.preprocessing import sequence, text
from tensorflow.keras ensorflow.kera
from tensorflow.keras. layers import Dense, Dropout,
Embedding, Conv1D, MaxPooling1D, GlobalAveragePooling1D
```

1. For this book, we will use a pre-trained GloVE vector from Stanford. Go to URL **https://nlp.stanford.edu/projects/glove/**and download the pre-trained model of your choice. However, for our use case, we have downloaded Wikipedia 2014 + Gigaword five datasets where each token or word would be represented in `200` dimensions. This will be downloaded to your local system. However, you can also upload the same to Google Colab as explained earlier. GloVe vector file will be available with a name as `glove.6B.200d.txt` in our Google Colab environment.

2. Next, we need to generate an embedding matrix where a vector representation of each of the tokens or words from the description will be stored. Please note that the vector representation is being generated using the GloVE method as follows:

```
      def embedding_matrix_conv(word_index,
embedding_file_path, embedding_dimension):
    embedding_matrix_comb = {}
    with open(embedding_file_path, ‘r’) as embed_file:
        for token_entry in embed_file:
      values = token_entry.split()
            word = values[0]
      coefs = np.asarray(values[1:], dtype='float32')
            embedding_matrix_comb[word] = coefs
  num_words = min(len(word_index) + 1, top_tokens)
    embedding_matrix = np.zeros((num_words,
embedding_dimension))
    for word, word_position in word_index.items():
        if word_position >= top_tokens:
            continue
        embedding_vector = embedding_matrix_comb.
get(word)
        if embedding_vector is not None:
            embedding_matrix[word_position] =
embedding_vector


    return embedding_matrix
```

Now, we will proceed with the model training. For our use case, we will train the Convolution Neural Network (CNN) model that will categorize an issue or a query description to a particular category. As an example, for an issue description CPU Utilization is very high, please suggest, this model will classify it to the `CPU_Utilization`.

The CNN is used in an image processing and computer vision domain, as it captures the spatial and temporal dependencies in an image clearly by applying relevant filters. As an example, you can leverage the CNN to classify the images of animals to their particular category i.e. image of a Cat or Dog will be classified to Cat and Dog classes, respectively.

The CNN reduces the number of parameters to be trained as compared with the other neural network architecture by using the concept of filters. *Figure 4.9* shows one of the applications of the CNN in images where the input image is masked with a filter to generate features out of it.

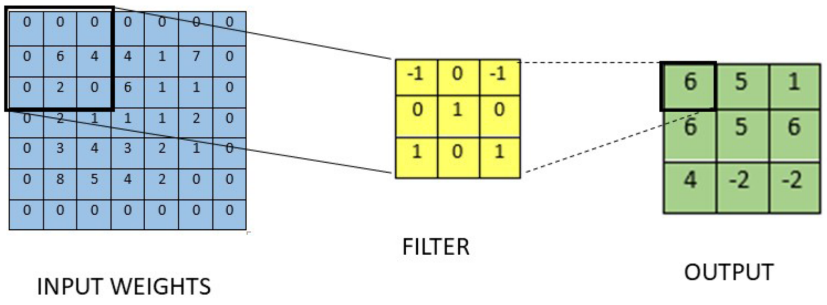


Figure 4.9: *Convolution Neural Network – An Example*

The CNN plays a crucial role by reducing it into a form that will be used for prediction while preserving all the features of an image. This enables the CNN to be scalable with large datasets.

In the CNN, we have a convolutional layer that extracts the high-level features such as edges from the input image. This layer is the building block of the CNN. It consists of a set of independent filters that are convolved with the image giving us the feature maps. These filters are randomly initialized, and they become parameters on subsequent learning by the network.

Each of the neuron is connected to an input image's small chunk for a particular feature map. There is also a parameter sharing in a particular feature map. All the neurons have the same connection weights in a particular feature map. The parameter sharing and local connectivity helps in the reduction of the number of parameters in the whole system and ensures better computational efficiency.

There is also a concept of pooling that makes the CNN differ from other counterparts.

It is a function to reduce the spatial size of the representation progressively in order to reduce the number of parameters and computation. The pooling layer operates independently on each of the feature map.

After the pooling layer, the flattened output is fed to the next layer in the network and then back propagation is applied to every iteration of the training. Over a series of epochs, the model is able to classify the images using the Softmax Classification technique.

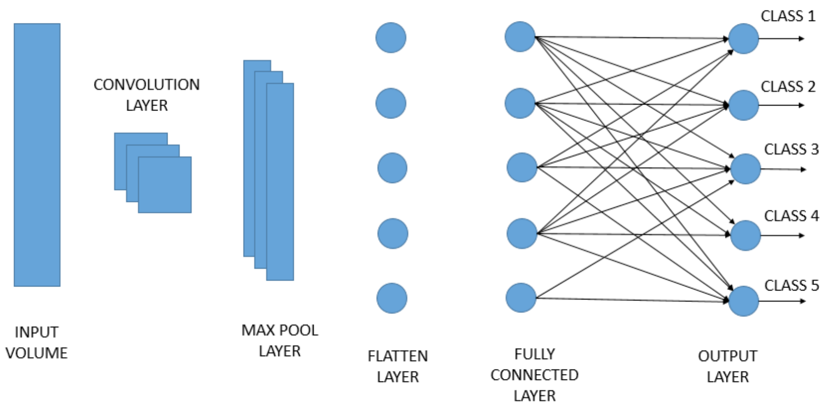


Figure 4.10: *Convolution Neural Network*

From last few years, the Convolution Neural Network has been used in implementation of NLP related tasks such as Text (document or sentence) classification, Entity Recognition, and so on. This is due to the similarity in processing of a vector representation of a description or text with the images. As an example, in case of images, the CNN slides over various parts of images such as objects, contours etc., and for textual data, it will slide over the rows of embedding matrix to go deeper to find hidden meaning out of the text (document or sentence).

The CNN has some following hyperparameters that have to be defined before training a model:

- `filters:` Defines the number of filters (64) to be used
- `dropout_rate:` To avoid overfitting in a model. Value will be `0.2`.
- `embedding_dimension:` Defines the number of dimensions for each of issue or query description. The number of input nodes in the CNN will be equal to the `embedding_dimension` value.

- `kernel_size:` It is similar to n-gram where group of words are being selected as one of the features. In this case, kernel value will be 3.

```
filters=64
dropout_rate=0.2
embedding_dimension=200
kernel_size=3
pool_size=3
tokenizer=text.Tokenizer (num_words=top_tokens)
tokenizer.fit_on_texts(train_text)
word_index=tokenizer.word_index
embedding_file_path = 'glove.6B.200d.txt'
```

Next, we will start with training of the Convolution Neural Network (CNN) over our dataset, where each query or issue description has been converted to a vector representation using GloVE as a word representation method. For our use case, the architecture of the CNN contains the following sequence of layers:

- **Input Layer:** In this layer, an embedded representation of description would come as an input. Here, the number of nodes at the input layer will be equal to the size of the embedding dimension.
- **One-Dimensional Convolution Layer:** In this layer, a filter runs through a vector representation of the description to extract a feature such as N-grams etc.
- **Max-Pooling Layer:** In this layer, the most relevant feature will be selected. For this use case, we have used Max-Pooling. However, you can also use other pooling methods such as Average Pooling, Min-Pooling, or even a neural network can be used at the pooling layer.
- **One-Dimensional Convolution Layer:** In this layer, a convolution will be performed over the result from Max-Pooling layer to find the combination of n-grams that may come together.
- **Fully Connected Layer:** In this layer, flatting of weight matrix is being done and sent to the SoftMax layer.

- **SoftMax Layer:** The input to this layer will be a vector representation of the features that have been extracted in the previous layer. Then this layer uses an activation function i.e. SoftMax to find the probability distribution across categories or classes.

```
def create_model():
  model = models.Sequential()
  features = min(len(word_index) + 1, top_tokens)
  model.add(Embedding(input_dim=features,
              output_dim=embedding_dimension,
              input_length=max_len,
              weights=[embedding_matrix_
conv(index_word,
                              embedding_file_path,
embedding_dimension)],trainable=True))
  model.add(Dropout(rate=dropout_rate))
  model.add(Conv1D(filters=filters,
              kernel_size=kernel_size,
              activation='relu',
              bias_initializer='he_normal',
              padding='same'))
  model.add(MaxPooling1D(pool_size=pool_size))
  model.add(Conv1D(filters=filters * 2,
              kernel_size=kernel_size,
              activation='relu',
              bias_initializer='he_normal',
              padding='same'))
  model.add(GlobalAveragePooling1D())
  model.add(Dropout(rate=dropout_rate))
  model.add(Dense(len(CLASSES),
activation='softmax'))
  optimizer = tf.keras.optimizers.Adam(lr=0.001)
  model.compile(optimizer=optimizer, loss='sparse_
categorical_crossentropy', metrics=['acc'])
  return model
```

Now, we have defined an architecture of our CNN model. Next, we need to convert the issue or query descriptions' into a sequence of integers, where each number or integer will be an index of token extracted from the description.

```
train_process = tokenizer.texts_to_sequences(train_text)

train_process = sequence.pad_sequences(train_process,
maxlen=max_len)

valid_process = tokenizer.texts_to_sequences(valid_text)

valid_process = sequence.pad_sequences(valid_process,
maxlen=max_len)

test_process = tokenizer.texts_to_sequences(test_text)

test_process = sequence.pad_sequences(test_process,
maxlen=max_len)
```

The next step is to start the actual training process. However, there are two following ways by which a model can be trained:

- **Using your local machine:** If you have a sufficient storage and compute resources such as GPU, TPU available with you, then you should go with this option. In this case, you will train a model in your local system and use it on the AI platform for prediction.
- **Using AI platform:** If you don't have a sufficient storage and compute resources such as GPU, TPU available with you, then you should go with this option only. In this case, the model training and prediction both will happen on the AI platform.

Using your local machine

In this book, we will discuss both the methods of model training. Please follow the following steps to train a model using your local machine:

1. Create an instance of a model architecture that has been created earlier
2. You can also display the summary of your model if you want as follows:

   ```
   model =create_model()

   model.summary()
   ```

3. Next, we will define the model checkpoint directory where the model checkpoints will be stored as follows. The model checkpoints are the state of a model at a particular time. This helps to restart the model training from where it left off, otherwise this would lead to a model training from scratch every time.

```
import os

checkpoint_path = "training_path/cp.ckpt"

checkpoint_directory = os.path.dirname(checkpoint_
path)


callback_path = tf.keras.callbacks.
ModelCheckpoint(filepath=checkpoint_path,save_
weights_only=True,verbose=1)
```

4. Fit the model architecture over the training set with the following arguments:
   - `train_process`: Processed form of training data
   - `train_labels`: Set of labels for each of the descriptions in the training set
   - `epochs`: Number of times or instances a model went through the entire training set
   - `validation_data`: Set of records from the validation dataset
   - `callbacks`: Defines TensorFlow callback path

   This is required to perform the model checkpoints after a successful completion of each epoch.

```
model.fit(train_process,

          train_labels,

          epochs=10,

          validation_data=(test_process,test_
labels),

          callbacks=[callback_path])
```

5. Model training would take some time depending on the size of the dataset. You can perform the prediction as soon as a model is available to use.

6. Next, create a folder named as `model_saved` and store all the model files in that folder as follows:

   ```
   !mkdir -p model_saved
   model.save('model_saved/model')
   ```

   You can also save the model in Google drive mounted on your Google Colab environment.

   Now, we have trained the CNN model for our Issue Categorization System in Google Colab, and the trained model is available in local or Google drive mounted on the Colab account. However, in order to use this model for prediction or inference, this has to be stored in some storage service such as an Object Storage, where all the model file will be stored as an object and can be accessed easily by Google AI Platform whenever required. Please follow the steps to upload the model files in Google's Object Storage service.

7. Go to URL **https://cloud.google.com/** and login with your credentials

8. In the left side menu, select **`Storage`** and click on the **`Browser`** option as shown in *Figure 4.11*.

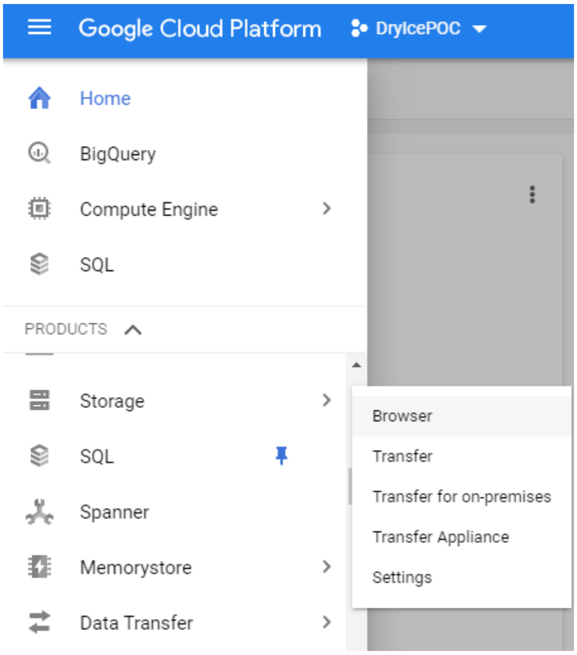
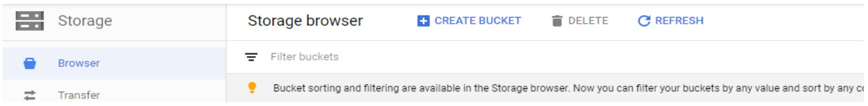


Figure 4.11: *Google Storage Service*

9. Click on the CREATE BUCKET link to create a bucket as shown in *Figure 4.12*.



Bucket sorting and filtering are available in the Storage browser. Now you can filter your buckets by any value and sort by any co

Figure 4.12: *Bucket Creation*

10. Run the following command in Colab to copy the model to the bucket created in the previous step, where `<BUCKET_NAME>` is the name of bucket that will be created.

```
!gsutil cp -r saved_model/my_model/ gs://<BUCKET_
NAME>/text_classification/export_tensorflow
```

We are done with the training of the CNN model in Colab environment and also the trained model has been stored in Google's Storage service.

Using AI platform

Now, we will see the training of the same model on the AI Platform. Please follow the following steps:

1. Upload your dataset file i.e. `Dataset.csv` from Google's Colab environment to Google's Storage service.

   Run the following command to copy the file from Colab to Storage service, where `<BUCKET_NAME>` is the name of the bucket created in this service, and `<FOLDER_NAME>` is the name of the folder that will be created inside the bucket for your dataset.

   ```
   !gsutil cp Dataset.csv gs://<BUCKET_NAME>/<FOLDER_
   NAME>
   ```

2. Upload the pre-trained GloVe vector embedding the file to the same bucket.

   Run the following command:

   ```
   !gsutil cp glove.6B.200d.txt gs:// <BUCKET_
   NAME>/<FOLDER_NAME>
   ```

3. Install `gcsfs` library to enable the pandas library that will read data from the bucket that has been created in Google's storage service. But to install the external dependencies on the AI platform, you would need to create a `setup.py` file

in the root folder of your project in Colab environment as follows:

```
from setuptools import find_packages

from setuptools import setup

REQUIRED_PACKAGES = ['gcsfs==0.5.1']

setup(

    name='trainer',

    version='0.1',

    install_requires=REQUIRED_PACKAGES,

    packages=find_packages(),

    include_package_data=True,

    description='Pandas library to read data from
files stored in GCS.'

)
```

Note: If you want to install other external libraries, then you can define the same in the `REQUIRED_PACKAGES` list in the `setup.py` file.

4. Go to the **`Upload`** button under the **`Files`** tab to upload or create the `setup.py` file as shown in *Figure 4.13*.

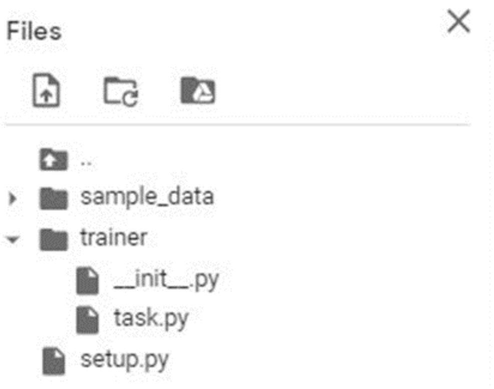


Figure 4.13: *File Upload – setup.py*

5. Create a python package named `trainer` in Colab environment and create a file named `task.py`.

Copy the entire code that we have discussed earlier, paste it in this file, and save it. *Figure 4.13* shows the complete project structure for this use case.

6. In this mode i.e. using an AI platform, the model training process has to be submitted as a job to AI platform, and all the model files will be stored automatically to the bucket created in Google Storage Service.
7. All the codebase would remain the same except reading the pre-trained GloVe vector file. In this case, it will be done by using the following package `tf-io.gfile.GFile`:

```
embedding_matrix_comb = {}

with tf.io.gfile.GFile('gs://ai-test-tensorflow/
Dataset/glove.6B.200d.txt', mode='r') as embed_file:

    for token_entryin embed_file:

             values = token_entry.split()

             word = values[0]

                    coefs = np.asarray(values[1:],
dtype='float32')

             embedding_matrix_comb[word] = coefs
```

8. Define the following parameters:
   - `MODEL_DIR:` directory, where the trained model will be stored.
   - `JOB_NAME:` name of the training model job

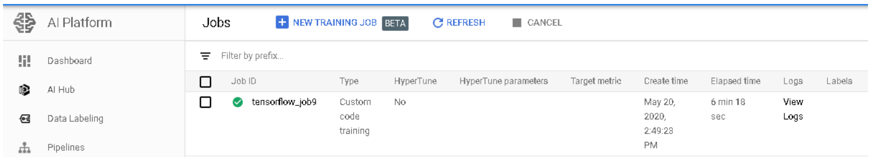
   ```
   %env MODEL_DIR=<PATH TO GCS BUCKET>

   %env JOB_NAME=<JOB_NAME>
   ```

9. Run the following command to create a job on AI platform and submit the model code:

```
!gcloud ai-platform jobs submit training $JOB_NAME
--module-name=trainer.task --package-path=trainer
--region=us-central1 --job-dir=$MODEL_DIR --runtime-
version=2.1 --python-version=3.7
```

The parameters `package-path` and `module-name` will determine the location of the python project and file, respectively.

10. You can monitor the status of your submitted jobs from AI platform console as shown in *Figure 4.14*. You may see a green mark against the job ID if it was successfully completed.



Job ID	Type	HyperTune	HyperTune parameters	Target metric	Create time	Elapsed time	Logs	Labels
tensorflow_job9	Custom code training	No			May 20, 2020, 2:49:23 PM	6 min 18 sec	View Logs	

Figure 4.14: *Job List*

We have seen the training of the Convolution Neural Network (CNN) model for our Issue Categorization System. We have also discussed about the two ways of training the model as follows:

- **Using your local machine:** In this mode, the model has been trained and stored locally and copied to the bucket created in the storage service.
- **Using AI Platform:** In this mode, the model has been trained on AI platform and stored automatically to the bucket created in the storage service.

Model Prediction

Now, our trained model is available in the bucket of Google Storage Service. As a next step, we need to deploy the trained model on AI platform for the prediction or inference task. Please follow the following steps:

1. Set the following environment variables:
   - `MODEL_NAME`: define the user-defined name of a model. For our use case, the `MODEL_NAME` will be `text_classification`.
   - `REGION`: define the region where the trained model will be deployed. You need to select a model region of deployment with the utmost care to avoid the conflict with data privacy policies across regions. As an example, as per the GDPR policy, data for a European user shouldn't go outside the European region. For our use case, the region will be us-`central1`.

```
%env MODEL_NAME text_classification
%env REGION us-central1
```

Run the following command in Google's Colab to create an empty model:

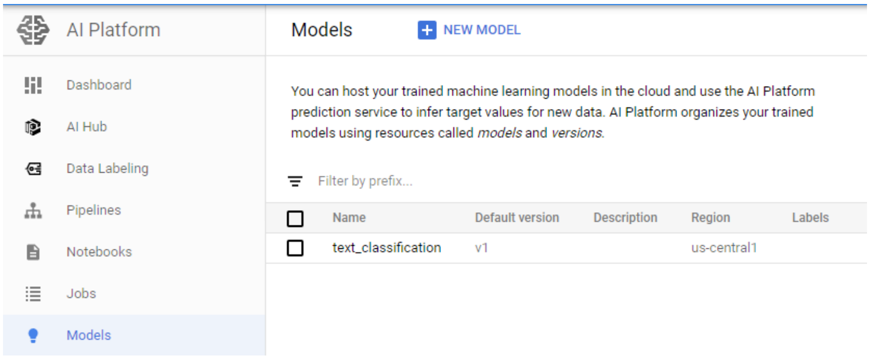
```
!gcloud ai-platform models create $MODEL_
NAME --regions $REGION
```

2. You can see one row has been created with the model name `text_classification`. Please note that the model version will not be displayed till you deploy a model.

   Next, set a value of an environment variable i.e. `MODEL_PATH` to the location where the model files have been stored. For our use case, the model files are stored in one of the bucket of Google's Storage Service. Hence, run the following command to set `MODEL_PATH` variable:

```
%env MODEL_PATH gs://<BUCKET_NAME>/text_
classification/export_tensorflow

!gcloud ai-platform versions create "v1" \
  --model $MODEL_NAME \
  --runtime-version 2.1 \
  --python-version 3.7 \
  --framework tensorflow \
  --origin $MODEL_PATH
```

This will deploy the model on AI platform to be ready for prediction. Now, you can see the details of the model deployed along with its version and region information as shown in *Figure 4.15*.



You can host your trained machine learning models in the cloud and use the AI Platform prediction service to infer target values for new data. AI Platform organizes your trained models using resources called *models* and *versions*.

Name	Default version	Description	Region	Labels
text_classification	v1		us-central1	

Figure 4.15: *List of Models*

With this, we are done with the model training and deployment to Google's AI platform. You can also deploy multiple versions of same model. As an example, in the real-life scenarios, machine learning models should continue to evolve to accommodate new data points as well. Hence, a new version of the same model would come up. In that case, you can deploy the latest trained version of a model, and during the prediction, you can define which version of model you want to use.

In the next step, we will see how a deployed model can be used during the prediction process. To demonstrate this, we have used the following record from the validation set:

```
import json
input_dict={'embedding_9_input': valid_process[1:2]}
with open('sample_instance.json', 'w') as prediction_file:
    json.dump(input_dict, prediction_file)
```

The previous code will take a record from the validation set and dump it as JSON file. The content of JSON file will look like as follows:

```
{"embedding_9_input": [  0,   0,   0,   0,   0,   0,   0,
0,   0,   0,   0,   0,   0, 0,   0,   0,   0,   0,   0,
0,   0,   0,   0,   0,   0,   0, 0,   0,   0,   0,   0,
0,   0,   0,   0,   0,   0,   0,   0,  0,   0,   0,   0,
0,   0, 920, 453,  84,   1,   2]}
```

The downloaded JSON file serves as an input for the deployed model, and it will return the prediction results i.e. a set of categories along with their confidence score.

Run the following command in Google's Colab to submit a JSON file as an input to get prediction:

```
!gcloud ai-platform predict \
  --model $MODEL_NAME \
  --version "v1" \
  --json-instances sample_instance.json
```

This will return an array of confidence scores corresponding to each class or category as follows:

```
[9.673918611952104e-06, 0.9999891519546509,
1.2268571936147055e-06]
```

The previous results were returned when you sent a query or an issue description directly using Gcloud via Google Colab. However, if you want to get the prediction results from this model using python code, then it is also possible. Please look at the following sample python code to fetch prediction results from the deployed model.

```
def predictClass():
    try:
        content= [‘User requested to Change Password as
expired’]
result = {}
        pred_process = tokenizer.texts_to_
sequences(content)
        pred_process = sequence.pad_sequences(pred_process,
maxlen=max_len)
        new_model = tf.keras.models.load_model('model_
saved/model')
        prediction = int(new_model.predict_classes(pred_
process))

        for key, value in CLASSES.items():
            if value==prediction:
                category=key
                result["class"] = category
        result = {"results": result}
        result = json.dumps(result)
        return result


if __name__=='__main__':
       predictClass()
```

In the previous code, for an issue description i.e. User requested to Change Password as expired, first the system will convert it into a sequence of integers followed by the padding if necessary. Next, the trained model will be loaded in the memory and will perform the prediction over the processed form of an issue description.

REST API configuration

Now, consider a scenario where you may need to release a feature of this system to another system or an application. As an example, the conversation system provides the **Standard Operating Procedure (SOPs)** document to the users for an issue or query raised by them. However, in this case, the conversation system should understand the query or issue raised by the user. What if the conversation system sends a query or an issue submitted by the user to our Issue Categorization system via REST API, and this system returns a relevant category? In this case, the conversation system can use the results from our Issue Categorization system and populate the SOPs accordingly.

In the previous chapter, we have already seen how to setup a pre-requisite for REST API either on Windows or Linux server. Now, we will see the steps to release the features of our Issue Categorization System as REST API.

1. Create a file with name `ticketclass.py`
2. Copy the following code and paste in the file created in the first step and save it

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing import sequence,
text
from tensorflow.keras import models
from tensorflow.keras.layers import Dense,
Dropout, Embedding, Conv1D, MaxPooling1D,
GlobalAveragePooling1D
from flask import Flask, request
import json

app=Flask(__name__)


CLASSES = {'CPU_Utilization': 0, 'Password_Reset':
1, 'Memory_Utilization': 2}
top_tokens = 20000
max_len = 50
```

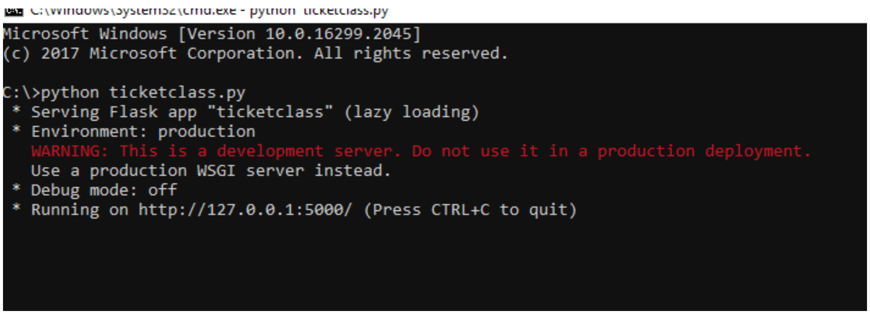
```
def return_data(df):
    return list(df['Text']),
np.array(df['Category'].map(CLASSES))

data = pd.read_csv('Dataset.csv', header = None)
data.columns = ['Text', 'Category']
train = data.sample(frac=0.8)

train_text, train_labels = return_data(train)

tokenizer = text.Tokenizer(num_words=top_tokens)
tokenizer.fit_on_texts(train_text)

app=Flask(__name__)

@app.route ("/predictclass", methods=['POST'])
def predictClass():
    try:
        json_data = request.get_json(force=True)
content= json_data["content"]
        text = list(content)
result = {}
        pred_process = tokenizer.texts_to_
sequences(text)
        pred_process = sequence.pad_sequences(pred_
process, maxlen=max_len)
        new_model = tf.keras.models.load_
model('model_saved/model')
        prediction = int(new_model.predict_
classes(pred_process))

        for key, value in CLASSES.items():
            if value==prediction:
                category=key
                result["class"] = category
        result = {"results": result}
        result = json.dumps(result)
```

```
            return result
        except Exception as e:
            return {"Error": str(e)}
    if __name__ == "__main__" :
        app.run(port="5000")
```

3. The previous code processes an input passed to an API, calls a function that predicts the category of the issue or query description, and sends the response as an API response.
4. Open the command prompt and run the following command;

   `python ticketclass.py`

   This will start a service on **http://127.0.0.1:5000/** as shown in *Figure 4.16.*



```
C:\Windows\System32\cmd.exe - python ticketclass.py
Microsoft Windows [Version 10.0.16299.2045]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\>python ticketclass.py
 * Serving Flask app "ticketclass" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Figure 4.16: *Service Deployment – Prediction System*

5. Now to test the Rest API, we are going to use Postman. This is a REST API client that is used to test the API URL.
6. First, go to the following URL to download the Postman client and install in your system **https://www.postman.com/downloads/**
7. After the installation, test the following URL and sample request JSON that is being sent to the Issue Categorization System API and response JSON that will be received as a response from the API as shown in *Figure 4.17.*

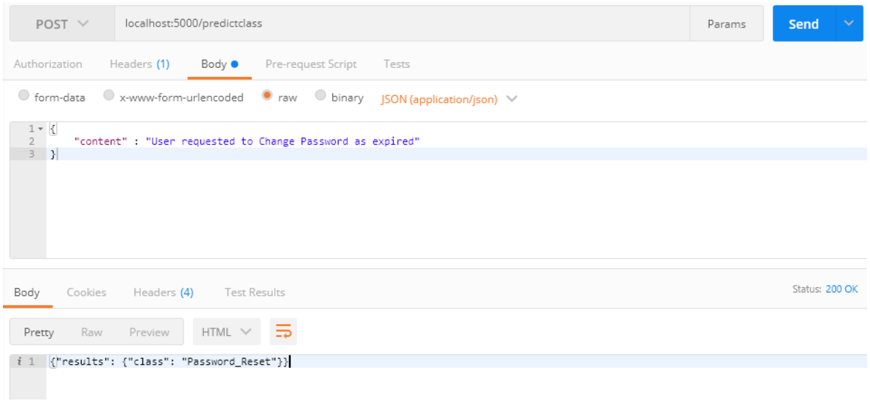
   URL: **http://127.0.0.1:5000/predictclass**

   Classification System – Sample Input Request json:

```
{
"content": "User requested to Change Password as
expired"
}
```

Classification System – Sample Output Response json:

```
{"results": {"class": "Password_Reset"}}
```



localhost:5000/predictclass

JSON (application/json)

```
{
    "content" : "User requested to Change Password as expired"
}
```

Status: 200 OK

```
{"results": {"class": "Password_Reset"}}
```

Figure 4.17: *Prediction System – REST API Result*

Conclusion

In this chapter, we have discussed about how Google's AI platform can be used for the custom model training and deployment. We have created a custom CNN model for an Issue categorization problem and used it in Google AI platform. We have also seen how predictions from trained model can be accessed by other system using REST API.

Chapter 5
Google Data Analysis, Preparation, and Processing Services

In the previous chapter, we have implemented an Issue Categorization System where the Convolution Neural Network (CNN) based text classification system has been developed using Google AI Platform. In this implementation, we have written our own custom code for data analysis, preparation, and processing activities. However, Google provides their own services for data analysis and preparation activities. In this chapter, we are going to discuss on Google Data analysis, preparation, and processing services and their impact on the development of machine learning based systems.

Structure

In this chapter, we will cover the following topics:

- Design and Implementation of Loan Defaulter Prediction system
    - o Introduction and usage of Google BigQuery
        - ▪ Upload data using Google BigQuery
    - o Introduction and usage of Google Dataprep
        - ▪ Data Analysis using Dataprep

- o Introduction and usage of Google Dataproc
    - ▪ Model Training and Evaluation
    - ▪ Model Prediction

Objectives

After studying this chapter, you should be able to:

- Understand the basics and usage of Google BigQuery, Dataprep, and Dataproc
- Understand how Google Dataprep can be used for data analysis
- Understand how Dataproc can be used for training of machine learning models
- Understand how Dataproc can be used for the prediction from a trained model

Introduction

From the last few years, Google has been actively working on solving problems of organizations that are focused towards the development of AI-based solutions with less effort. It has released several libraries, products, and services to reduce the development time for such systems. As an example, TensorFlow, a Google's Deep learning library has completely revolutionized the way we see the development of deep learning or machine learning models, and AutoML, a framework that selects the best model from their set of model repositories.

In this chapter, we will see how other Google services such as Google BigQuery, Google Dataprep, and Google Dataproc can be used to support the building of machine learning models. In this chapter, we will work with the structured dataset as compared with the previous chapters, where we have worked with the unstructured ones. We will start our discussion with an explanation of a use case that will be implemented using these services.

Problem statement

Consider a scenario where a financial institution is struggling to find a list of customers who may default. A prediction system that can identify defaulters pro-actively can save the financial institutions from losses and can help them to decide on whether to onboard a customer or not. In this book, we have taken a dataset from a UCI machine learning repository (https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients). Details of the dataset are as follows:

- Dataset has 24 attributes out of which the payment variable (Yes = 1, No = 0) is a response variable.
- Details of other attributes are as follows:
  - o **`ID`:** ID of each client
  - o **`LIMIT_BAL`:** Amount of the given credit in NT dollars (includes individual and family/supplementary credit)
  - o **`SEX`:** Gender (1=male, 2=female)
  - o **`EDUCATION:`** (1 = graduate school, 2 = university, 3 = high school, 4 = others, 5 = unknown, and 6 = unknown)
  - o **`MARRIAGE`:** Marital status (1 = married, 2 = single, and 3 = others)
  - o **`AGE:`** Age in years
  - o **`PAY_0:`** Repayment status in September, 2005 (-2 = no consumption, -1 = pay duly, 0 = the use of revolving credit, 1 = payment delay for one month, 2 = payment delay for two months, … 8 = payment delay for eight months, and 9 = payment delay for nine months and above)
  - o **`PAY_2:`** Repayment status in August, 2005 (scale, same as the previous one)
  - o **`PAY_3:`** Repayment status in July, 2005 (scale, same as the previous one)
  - o **`PAY_4:`** Repayment status in June, 2005 (scale, same as the previous one)
  - o **`PAY_5:`** Repayment status in May, 2005 (scale, same as the previous one)
  - o **`PAY_6:`** Repayment status in April, 2005 (scale, same as the previous one)
  - o **`BILL_AMT1:`** Amount of bill statement in September, 2005 (NT dollar)

- o **`BILL_AMT2:`** Amount of bill statement in August, 2005 (NT dollar)
- o **`BILL_AMT3:`** Amount of bill statement in July, 2005 (NT dollar)
- o **`BILL_AMT4:`** Amount of bill statement in June, 2005 (NT dollar)
- o **`BILL_AMT5:`** Amount of bill statement in May, 2005 (NT dollar)
- o **`BILL_AMT6:`** Amount of bill statement in April, 2005 (NT dollar)
- o **`PAY_AMT1:`** Amount of previous payment in September, 2005 (NT dollar)
- o **`PAY_AMT2:`** Amount of previous payment in August, 2005 (NT dollar)
- o **`PAY_AMT3:`** Amount of previous payment in July, 2005 (NT dollar)
- o **`PAY_AMT4:`** Amount of previous payment in June, 2005 (NT dollar)
- o **`PAY_AMT5:`** Amount of previous payment in May, 2005 (NT dollar)
- o **`PAY_AMT6:`** Amount of previous payment in April, 2005 (NT dollar)

As previously discussed, the problem is a binary classification problem where each customer or user can be classified into 0 or 1. Therefore, we will build Logistic Classifier over this dataset. As per machine learning pipeline, initially, data has to be analyzed followed by the preparation and then finally model training. Google BigQuery will be used for data analysis as discussed in the next section.

Google BigQuery

Google BigQuery is a cloud service that enables users to store and analyze a large amount of data. This has been designed specifically for analytics purpose. Due to this reason, tables in BigQuery can't be updated (append or update). You will need to delete the previous data before uploading a new one. It is similar to the SQL databases in terms of schema i.e. all the data will be stored in the form of tables and rows. Also, this data can be accessed by an SQL-like query. Hence, it is much faster and can be used for analytics systems. Google BigQuery only works with a structured data. So, if you want to perform analytics on an unstructured data then first convert the unstructured data into a structured one; and then store in BigQuery. As an example, consider you have a corpus of documents and you want to implement Document Search engine on this corpus. In this case, extract a set of entities or phrases out of each document. You can then store the extracted information as follows:

Doc ID	Entity
`Doc_1`	`Entity_1`
`Doc_1`	`Entity_2`
`Doc_2`	`Entity_1`
`Doc_2`	`Entity_3`

As you can see in the table, each of the documents will have a set of extracted entities. Therefore, we would have the same type of information for each document. Now, if a user asks a query, then first it will go to entity recognition system to extract entity from question and then to Google BigQuery to retrieve all documents that have extracted entity mapped to them. Later on, you can implement any Natural Language Processing (NLP) technique on top of this result. With this introduction to BigQuery let us start with our selected use case and see how we will use BigQuery in our scenario.

Please follow the following steps to upload and analyze credit card dataset:

1. Go to URL **https://cloud.google.com/** and login with your credentials

2. Select an option **BigQuery** from the left side menu bar as shown in *Figure 5.1*.

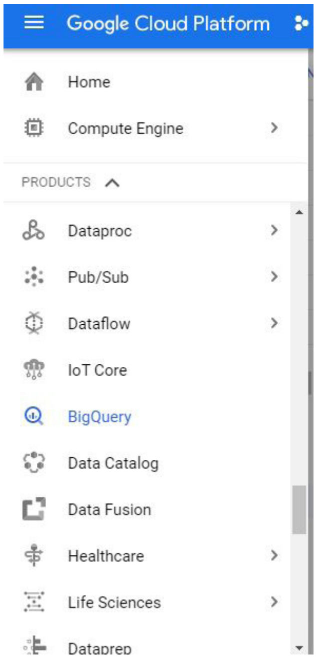


Figure 5.1: *Google BigQuery*

3. This will redirect you to BigQuery console. You should be able see the project associated to your account. If the project is not visible to you then only follow the following steps:

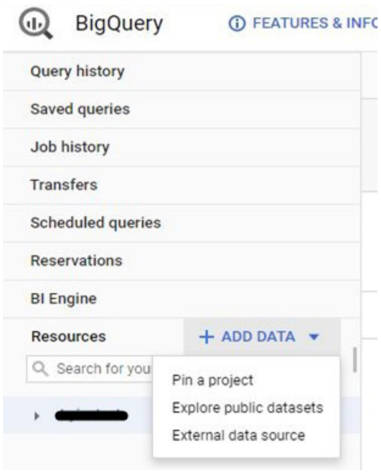
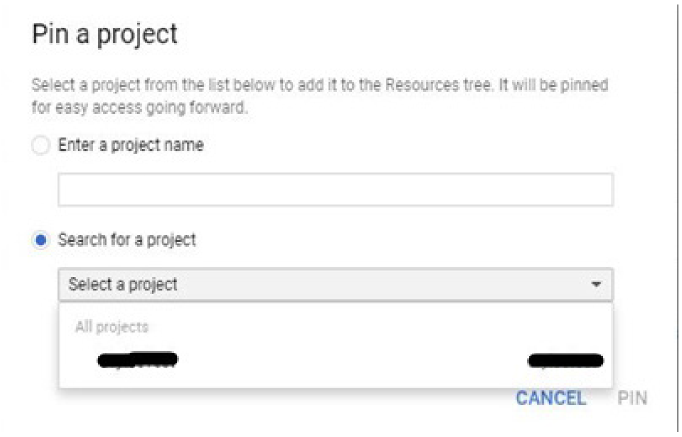


Figure 5.2: *ADD Data*

- Select an option **Pin a project** from the drop–down menu

  This will open up a new window as shown in *Figure 5.3.*

- Select the project associated with your account



Pin a project

Select a project from the list below to add it to the Resources tree. It will be pinned for easy access going forward.

Enter a project name

Search for a project

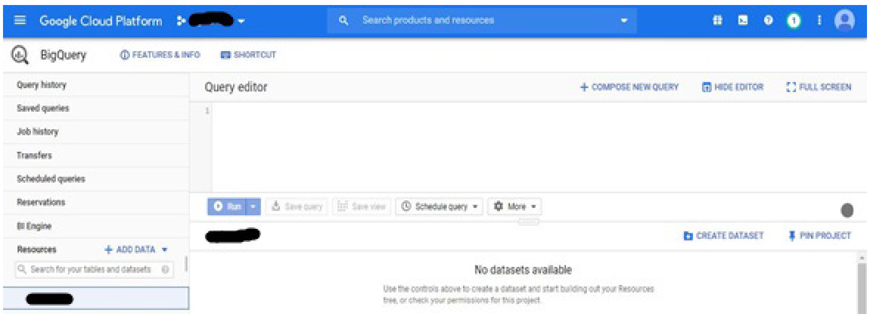
Select a project

All projects

[redacted] [redacted]

Figure 5.3: *Pin a project*

- Select your project and click on the **PIN** button. Once done, you will be able to see your project in BigQuery console.



No datasets available

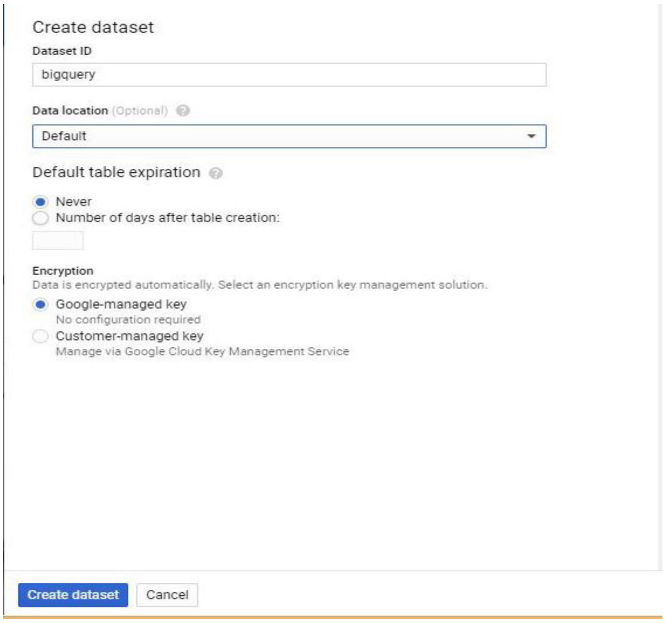
Use the controls above to create a dataset and start building out your Resources tree, or check your permissions for this project.

Figure 5.4: *Google BigQuery Home Page*

1. Click on **CREATE DATASET** link to create the dataset in your project. This will open up a window to define the values of fields as shown in *Figure 5.4.*
2. Fill up the values of following fields:
   - **DatasetID:** Defines the name of the dataset. For our use case, value is bigquery.

- **`Data Location:`** Defines the location of your dataset. Please ensure that it should be in the same zone in which you want to leverage other Google services. For our use case, value is default.

3. Next, click on the **`Create dataset`** button to create a dataset as shown in *Figure 5.5.*



Create dataset

Dataset ID

bigquery

Data location (Optional)

Default

Default table expiration

● Never
○ Number of days after table creation:

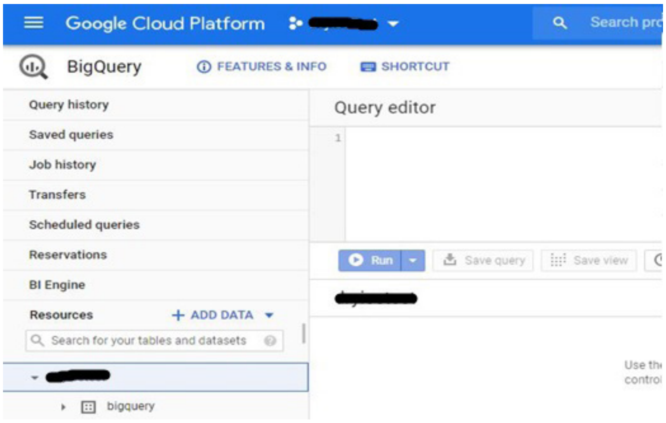
Encryption

Data is encrypted automatically. Select an encryption key management solution.

● Google-managed key
No configuration required
○ Customer-managed key
Manage via Google Cloud Key Management Service

Figure 5.5: *Google BigQuery – Create Dataset*

4. In BigQuery console, you should be able to see the dataset that has been created in the previous step as shown in *Figure 5.6.*



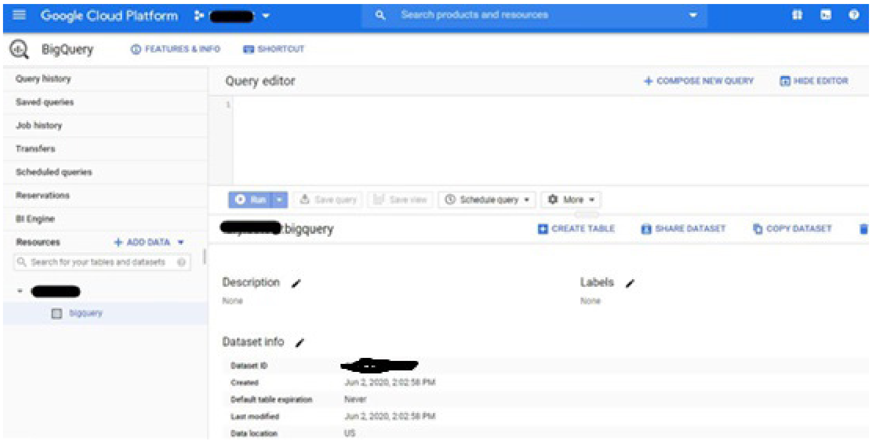
[redacted]

[redacted]

bigquery

Figure 5.6: *Google BigQuery – List of Datasets*

5. Next, click on the link **`CREATE TABLE`** to create a table in your dataset as shown in *Figure 5.7*.



Description
None

Labels
None

Dataset info

Dataset ID	[redacted]
Created	Jun 2, 2020, 2:02:58 PM
Default table expiration	Never
Last modified	Jun 2, 2020, 2:02:58 PM
Data location	US

Figure 5.7: *Google BigQuery – Create Table*

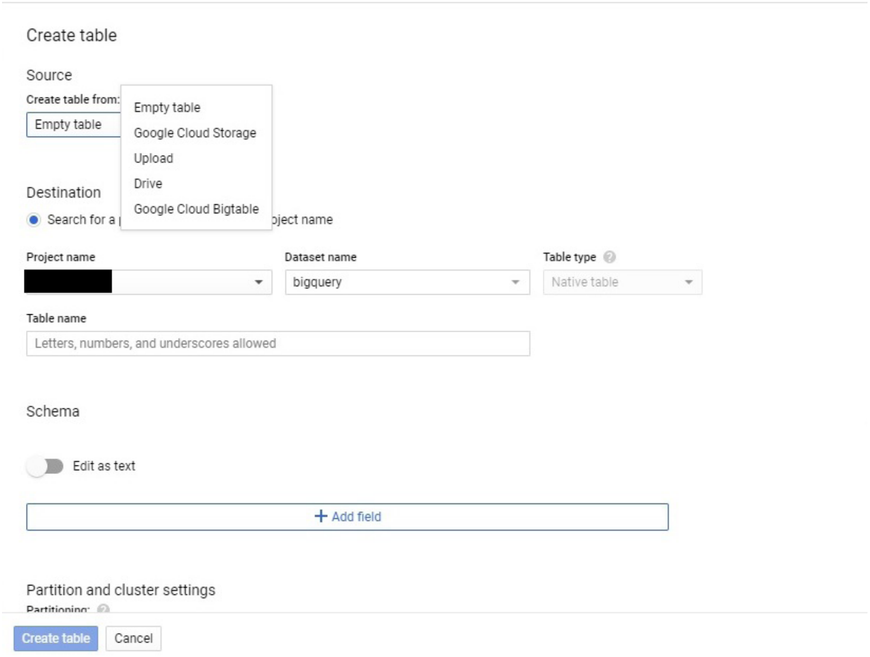
6. This will open up a new window as shown in *Figure 5.8*.

   Select any one of the following options as source:

   - **`Google Cloud Storage:`** Select this option if you want to use Google Cloud Storage Services such as an object storage, block storage, file storage, and archival storage.
   - **`Upload:`** Select this option if you want to upload a file from your local system. For our use case, we will go with this option.

Note: You can download the dataset from the following URL: **https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clientsas** mentioned earlier in the chapter.

- **`Drive:`** Select this option if you want to upload a file from your Google Drive
- **`Google Cloud Bigtable:`** Select this option if you want to use Google's NoSQL database service


Create table

Source

Create table from: Empty table

Empty table
Google Cloud Storage
Upload
Drive
Google Cloud Bigtable

Destination

Search for a project / Enter a project name

Project name: [redacted]

Dataset name: bigquery

Table type: Native table

Table name: Letters, numbers, and underscores allowed

Schema

Edit as text

+ Add field

Partition and cluster settings

Partitioning:

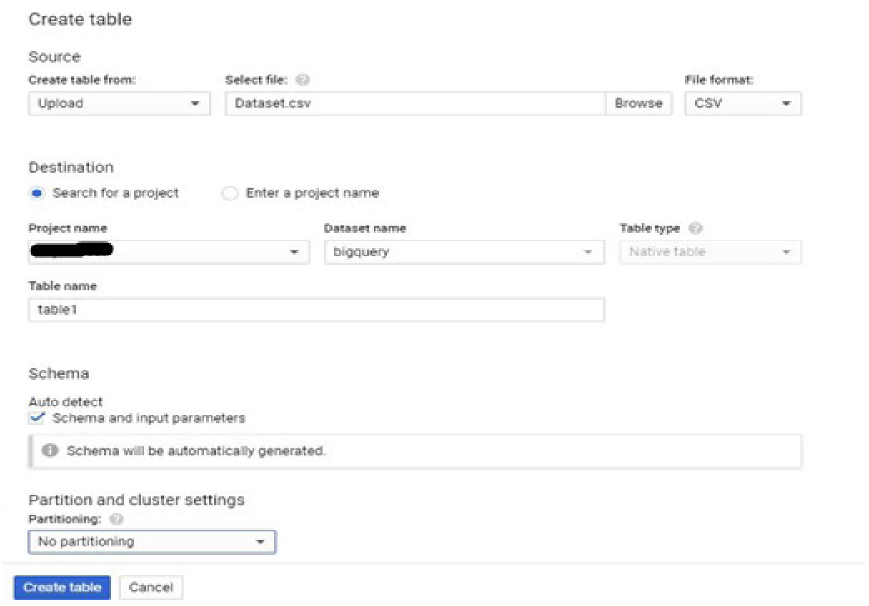
Create table | Cancel

***Figure 5.8:** Google BigQuery – Define Table details - I*

7. Define the value of the following field under **`Destination`** as shown in *Figure 5.9.*

   - **`Project Name:`** Select your current project

- **`Dataset Name:`** Select a dataset basis which the table should be created
- **`Table Name:`** Define the name of the table



Create table

Source

Create table from: Upload
Select file: Dataset.csv Browse
File format: CSV

Destination

● Search for a project ○ Enter a project name

Project name: [redacted]
Dataset name: bigquery
Table type: Native table

Table name: table1

Schema

Auto detect
✓ Schema and input parameters

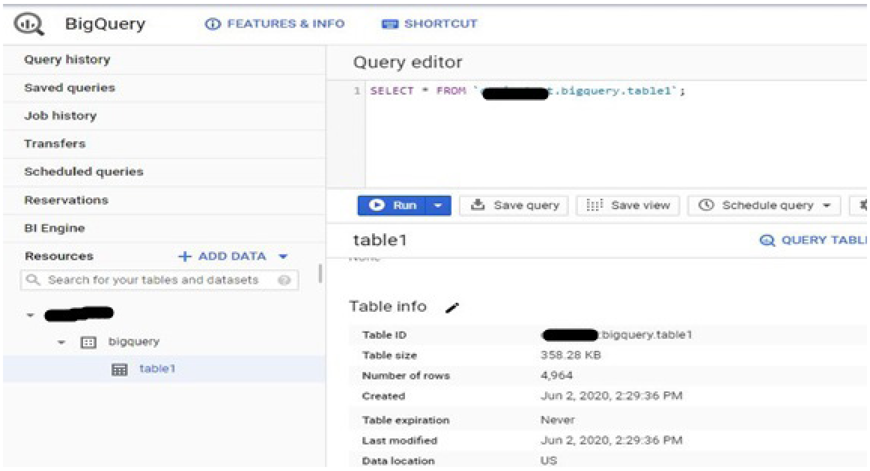
Schema will be automatically generated.

Partition and cluster settings
Partitioning: No partitioning

Figure 5.9: *Google BigQuery – Define Table details - II*

1. Select a checkbox against the label **`Schema and input parameters`** under **`Schema`**. This will auto-detect schema for the table based on your dataset. However, you can also define your custom schema.
2. Click on the **`Create table`** button to create a table

3. Now, you can see the table with a name Table 1 has been created and visible in BigQuery console as shown in *Figure 5.10.*



```
SELECT * FROM `[redacted].bigquery.table1`;
```

Table ID	[redacted]bigquery.table1
Table size	358.28 KB
Number of rows	4,964
Created	Jun 2, 2020, 2:29:36 PM
Table expiration	Never
Last modified	Jun 2, 2020, 2:29:36 PM
Data location	US

***Figure 5.10:** Google BigQuery – List of Tables*

Now, we have successfully uploaded our dataset to Google BigQuery. In the next section, we will see how Google Dataprep service can be used for the data preparation tasks such as Noise Removal, Missing Value imputation, and so on.

Google Dataprep

Data pre-processing is an important step in machine learning pipeline where data is cleaned and converted to a form which is consumable by machine learning algorithms. There are a few following examples to illustrate the same:

- For the text classification system, all the textual content should be converted into their respective vector representations and categories or classes should be encoded in one-hot representation.
- The AI based prediction system that will predict the sales of a particular company for future years. If the revenue or sales data for a particular period are not available, then either this point will be rejected to be considered for training, or its value will be calculated using the traditional machine learning algorithms.

As you can see from the previous two examples that it may even use a machine learning algorithm as a part of the pre-processing step. Due to this reason, it consumes almost 70-80% of the development time of any application. Therefore, it is always advisable to reduce the development time at this step. Earlier Google had released Dataflow service to create a machine learning pipeline, but it requires you to manually code each and every stage of the pipeline. In order to solve this problem, Google has come up with another service known as Google Dataprep, a fully managed service of Trifacta software. It enables a user to perform data cleaning, analysis, or even visual exploration of data is also possible with it.

Dataprep offers many benefits to the developers or data scientists who actually spend a considerable amount of time at this stage of machine learning pipeline. Some of the benefits can be listed as follows:

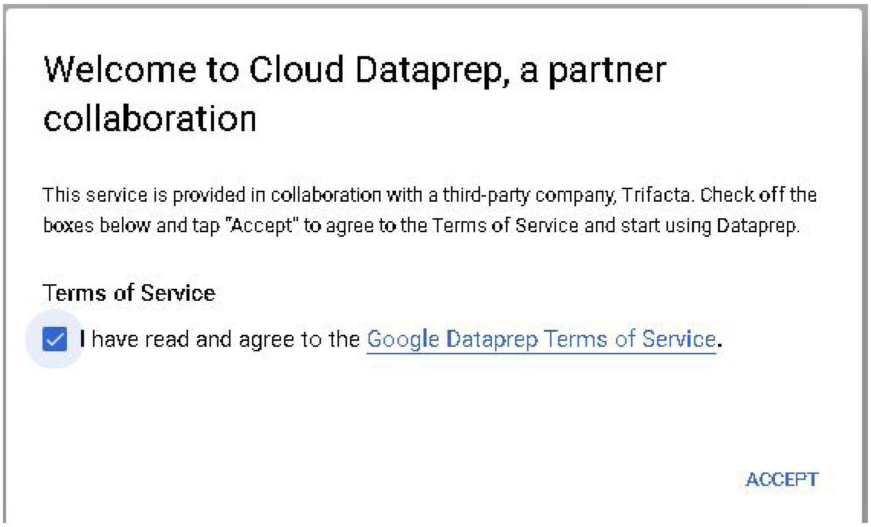
- It can analyze structured as well as unstructured data.
- It offers an integration with Google Cloud Storage service. Now, you just need to upload your dataset to the storage services and rest will be taken up by Dataprep.
- It offers an integration with other sources such as Local File System, BigQuer, and so on.
- It can leverage clean and convert data in AutoML and on AI platform for model training.
- It can analyse multiple data files.

Data Analysis

Now, we will see how Google Dataprep service can be used for performing a pre-processing over our dataset. Please follow the following steps to configure Dataprep for our use case.

1. Go to URL **https://cloud.google.com/** and login with your credentials
2. Select an option Dataprep from the left side menu bar

3. Next, agree to the terms of service of using Google Dataprep service and click on the **ACCEPT** button as shown in *Figure 5.11*.


Welcome to Cloud Dataprep, a partner collaboration

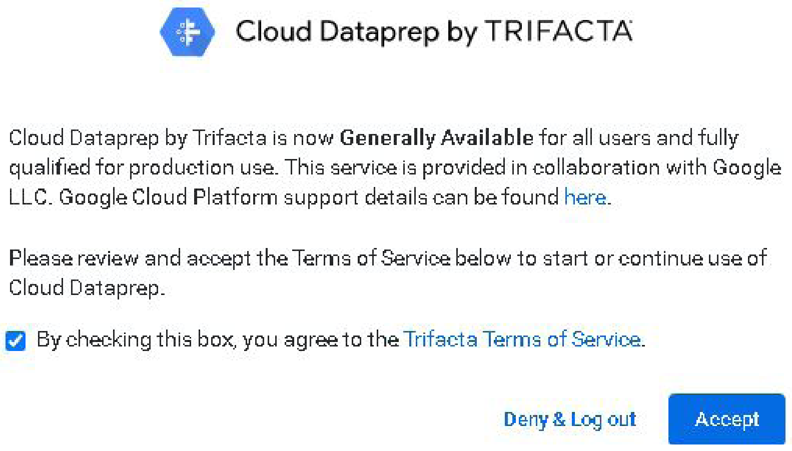
This service is provided in collaboration with a third-party company, Trifacta. Check off the boxes below and tap "Accept" to agree to the Terms of Service and start using Dataprep.

Terms of Service

☑ I have read and agree to the Google Dataprep Terms of Service.

Figure 5.11: *Google Dataprep – Service Creation*

4. Next, subscribe to a plan as per your requirement from the list of available plans for this service. Click on the checkbox to agree to various terms and conditions and click on the **SUBSCRIBE** button.
5. You will receive a notification from TRIFACTA. Click on the checkbox to agree to the **Trifacta Terms of Service** and click on the **Accept** button as shown in *Figure 5.12*.


Cloud Dataprep by TRIFACTA

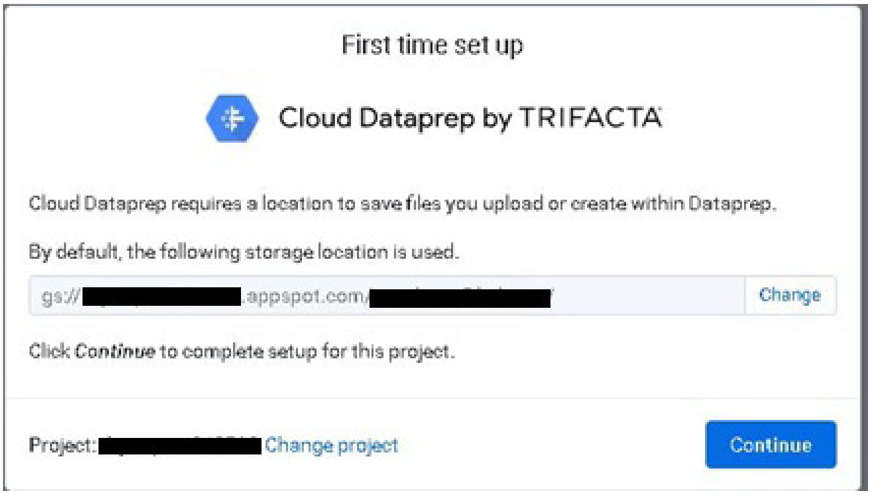
Cloud Dataprep by Trifacta is now **Generally Available** for all users and fully qualified for production use. This service is provided in collaboration with Google LLC. Google Cloud Platform support details can be found here.

Please review and accept the Terms of Service below to start or continue use of Cloud Dataprep.

☑ By checking this box, you agree to the Trifacta Terms of Service.

Figure 5.12: *Service Creation – Terms and Condition*

6. Next, you need to set a storage location for Dataprep jobs, otherwise it will create random buckets in your project which are difficult to manage. Therefore, it is advisable to perform this activity at an initial stage only as shown in *Figure 5.13*.


First time set up

Cloud Dataprep by TRIFACTA

Cloud Dataprep requires a location to save files you upload or create within Dataprep.

By default, the following storage location is used.

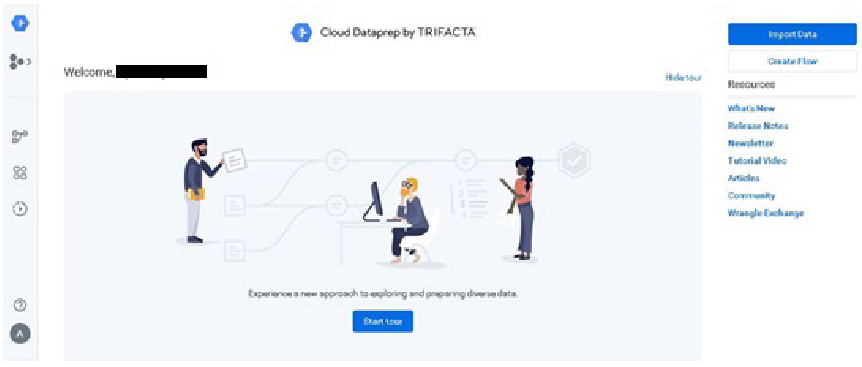
gs://[redacted].appspot.com/[redacted] Change

Click *Continue* to complete setup for this project.

Project: [redacted] Change project Continue

Figure 5.13: *Service Creation – Define Storage Bucket*

7. Click on the **`Change`** button and select your storage service as shown in *Figure 5.13*.
8. Then click on the **`Continue`** button to complete the setup. You will be redirected to a screen as shown in *Figure 5.14*.


Cloud Dataprep by TRIFACTA

Welcome, [redacted]

Experience a new approach to exploring and preparing diverse data.

Figure 5.14: *Google Dataprep Dashboard*

9. On the top-right corner, click on the **`Import Data`** button to upload your dataset. Dataprep offers the following three ways to upload the data:

- **`Upload:`** Select this if you want to upload a file from your local system. For our use case, we will go with this option.
- **`GCS:`** Select this if you want to upload a file from Google Cloud Storage services
- **`BigQuery:`** Select this if you want to upload data from Google BigQuery table

10. Click on the upload link from the left menu bar. click on the Choose a file button to select and upload a file from your local system as shown in *Figure 5.15*. You can also manage the upload location by clicking on the **`Change`** button.

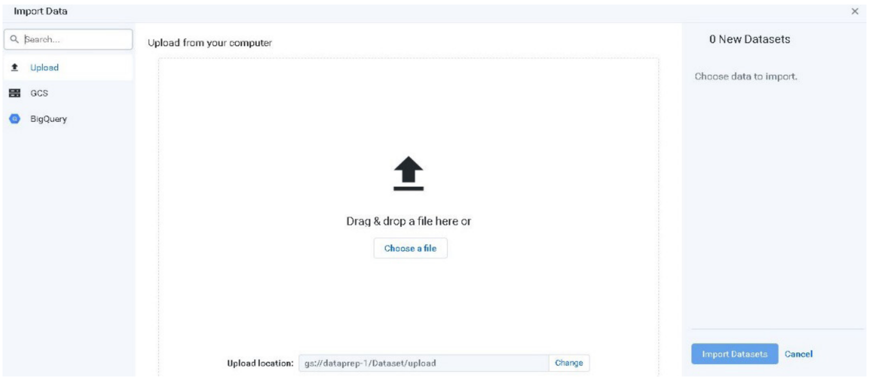
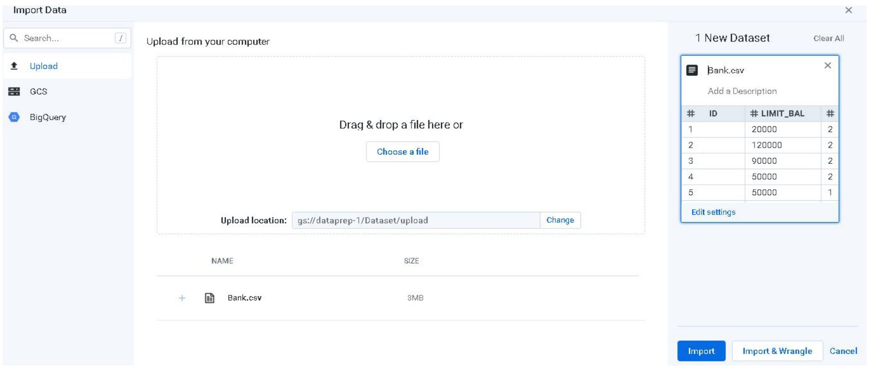


Figure 5.15: *Google Dataprep – Import Dataset - I*

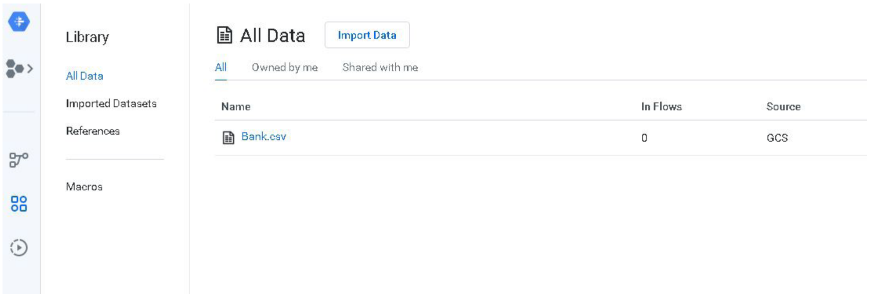
11. Click on the **`Import Datasets`** button on the right-side panel as shown in *Figure 5.15*.
12. After the successful import, you can see a sample data in the preview section on the right-side panel as shown in *Figure 5.16*.



# ID	# LIMIT_BAL	#
1	20000	2
2	120000	2
3	90000	2
4	50000	2
5	50000	1

Figure 5.16: *Google Dataprep – Import Dataset - II*

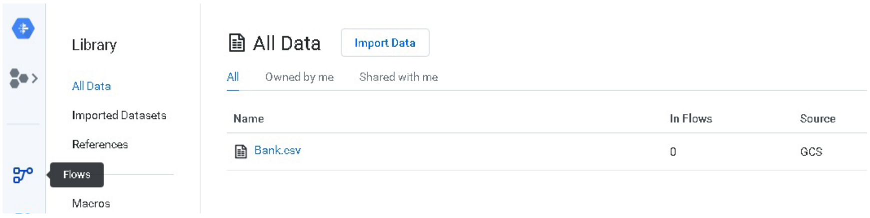
Click on the **`Import`** button in the right-side panel as shown in *Figure 5.16*. This will redirect you to a screen as shown in *Figure 5.17* where you can see your imported data.



Name	In Flows	Source
Bank.csv	0	GCS

Figure 5.17: *Google Dataprep – List of Datasets*

13. Click on the **`Flows`** option from the left-side panel to start the data processing as shown in *Figure 5.18*.



Name	In Flows	Source
Bank.csv	0	GCS

Figure 5.18: *Google Dataprep – Flows*

After selecting **`Flows`**, you will be redirected to a page as shown in *Figure 5.19*. Click on the **`Create Flow`** button as shown in *Figure 5.19*.

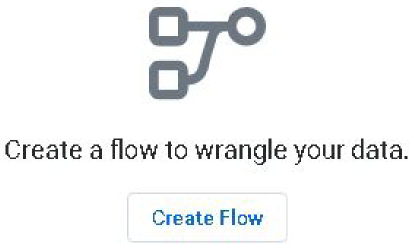


Figure 5.19: *Create Flows*

14. Enter the **`Flow Name`** as `Bank_flow` and **`Flow Description`** as `Pre-process` bank data as shown in *Figure 5.20*. Click on the Create button to **`create`** a flow as shown in *Figure 5.20*.

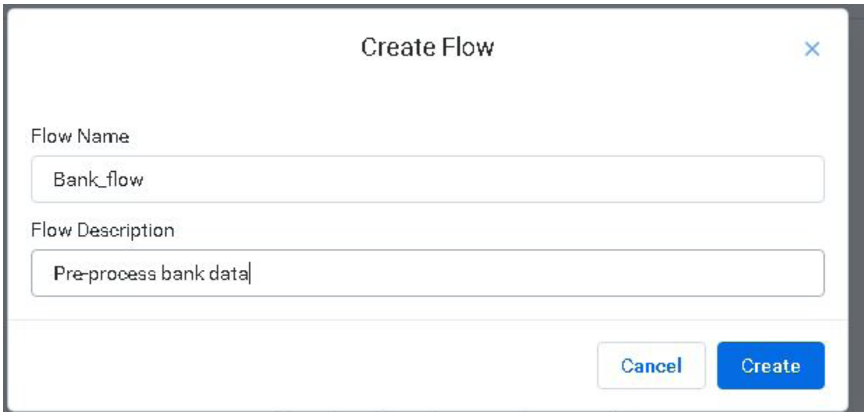


Figure 5.20: *Define Flow Details*

15. Next, add the dataset that you want to pre-process as shown in *Figure 5.21*.

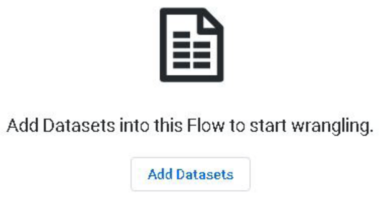
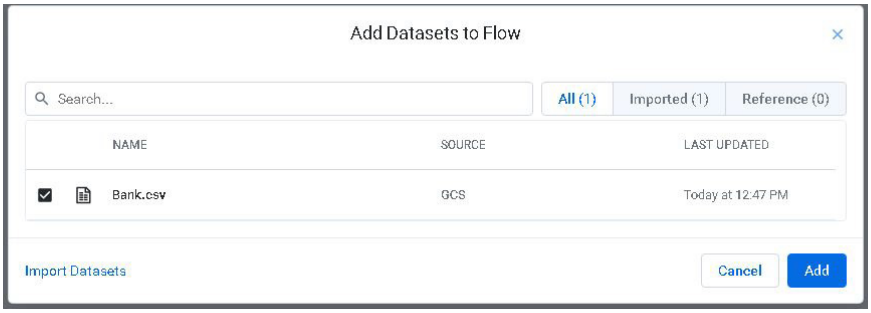


Figure 5.21: *Google Dataprep – Add Datasets*

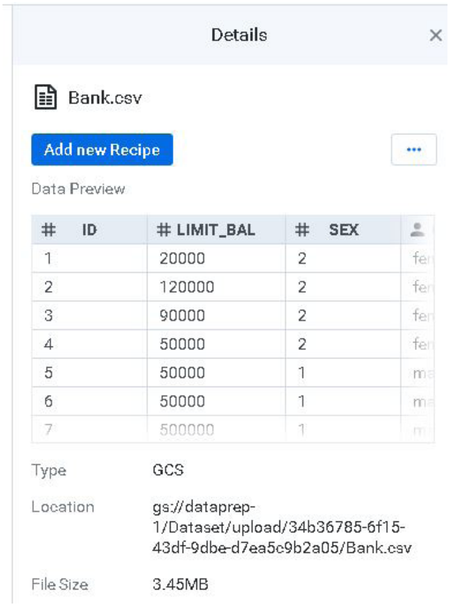
16. Click on the **`Add Datasets`** button to add a dataset as shown in *Figure 5.21*.
17. Next, select a dataset from the list of available datasets and click on the **`Add`** button as shown in *Figure 5.22*.



	NAME	SOURCE	LAST UPDATED
✓	Bank.csv	GCS	Today at 12:47 PM

Figure 5.22: *Google Dataprep – Import Datasets*

You can see a preview of imported datasets on the right-side panel. It shows a data preview and location of the file in Google Cloud Storage.



Details

Bank.csv

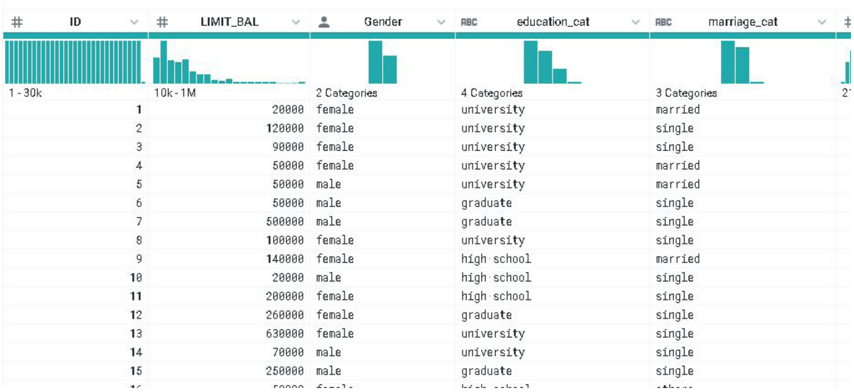
Data Preview

# ID	# LIMIT_BAL	# SEX	
1	20000	2	fe
2	120000	2	fe
3	90000	2	fe
4	50000	2	fe
5	50000	1	ma
6	50000	1	ma
7	500000	1	m

Type	GCS
Location	gs://dataprep-1/Dataset/upload/34b36785-6f15-43df-9dbe-d7ea5c9b2a05/Bank.csv
File Size	3.45MB

Figure 5.23: *Google Dataprep – Data Preview*

18. Click on the **`Add new Recipe`** button to start data processing as shown in *Figure 5.23*.
19. You can see a complete data along with different options to process them as soon as data import is completed as shown in *Figure 5.24*.



# ID	# LIMIT_BAL	Gender	ABC education_cat	ABC marriage_cat	#
1 - 30k	10k - 1M	2 Categories	4 Categories	3 Categories	2
1	20000	female	university	married	
2	120000	female	university	single	
3	90000	female	university	single	
4	50000	female	university	married	
5	50000	male	university	married	
6	50000	male	graduate	single	
7	500000	male	graduate	single	
8	100000	female	university	single	
9	140000	female	high school	married	
10	20000	male	high school	single	
11	200000	female	high school	single	
12	260000	female	graduate	single	
13	630000	female	university	single	
14	70000	male	university	single	
15	250000	male	graduate	single	

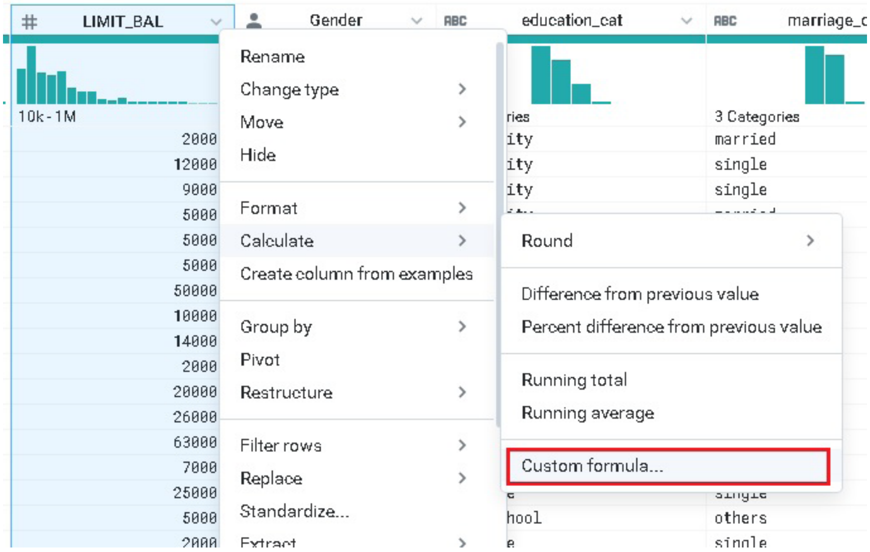
Figure 5.24: *Google Dataprep – Data View*

Now, we have successfully imported our dataset into Dataprep. Next, we will perform some of data pre-processing steps over our dataset as follows:

1. **Min-Max Normalization:** Normalization is a process that converts each value of a particular column to be in fixed range. It is necessary that all columns or attributes of your dataset should be in similar range otherwise column or attribute with higher value introduces bias in the model. As an example, column that shows a revenue of the company will introduce higher biasing than the column that shows an age of an employee. Min-max normalization ensures that all values of columns should be in the range of (a, b) where b > a.

   We will implement min-max normalization over one of the columns i.e. `LIMIT_BAL` in our dataset with the value of a and b as 0 and 1, respectively. Please follow the following steps to implement the same.

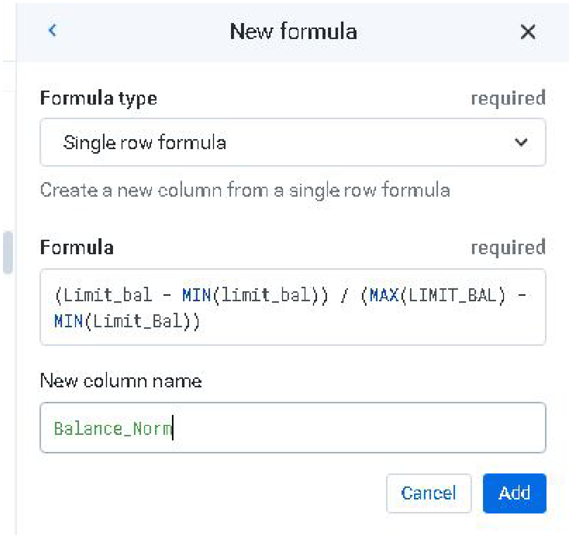
   - Select the column with a name `LIMIT_BAL` and then select **`Custom formula`** under **`Calculate`** option as shown in *Figure 5.25.*



```
2000   married
12000  single
9000   single
5000
5000
5000
50000
10000
14000
2000
20000
26000
63000
7000
25000
5000   others
2000   single
```

Figure 5.25: *Min Max Normalization – Custom Formula*

- Next, define the following values for following the fields:
  - o **`Formula type:`** Select a value of this field as single row formula
  - o **`Formula`**: Define min-max formula to convert values of `LIMIT_BAL` column as follows:
    `Min-Max_Normalization (LIMIT_BAL) = (LIMIT_BAL – MIN(LIMIT_BAL) ) / (MAX(LIMIT_BAL) – MIN(LIMIT_BAL))`
  - o **`New column name:`** Define a new column name as `Balance_Norm`
- Next, click on the **Add** button to add the formula to column as shown in *Figure 5.26*.


New formula

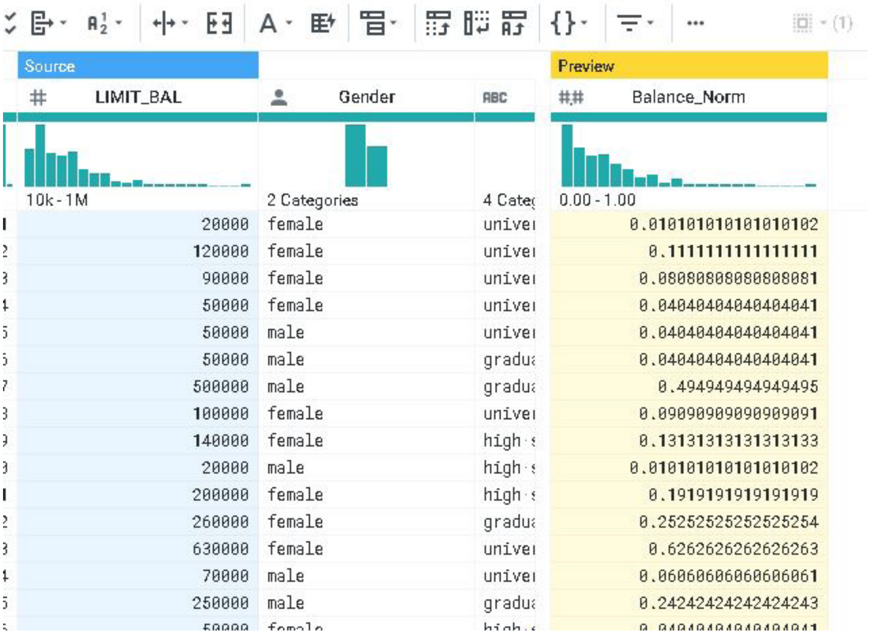
Formula type: Single row formula (required)
Create a new column from a single row formula

Formula (required):
```
(Limit_bal - MIN(limit_bal)) / (MAX(LIMIT_BAL) -
MIN(Limit_Bal))
```

New column name: Balance_Norm

Figure 5.26: *Min Max Normalization – Add Formula*

Now, you can see a new column with a name `Balance_Norm` has been added to view and all the values of this column lie between `0` and `1`.



LIMIT_BAL	Gender		Balance_Norm
10k - 1M	2 Categories	4 Categ	0.00 - 1.00
20000	female	unive	0.010101010101010102
120000	female	unive	0.1111111111111111
90000	female	unive	0.08080808080808081
50000	female	unive	0.04040404040404041
50000	male	unive	0.04040404040404041
50000	male	gradu	0.04040404040404041
500000	male	gradu	0.494949494949495
100000	female	unive	0.09090909090909091
140000	female	high s	0.13131313131313133
20000	male	high s	0.010101010101010102
200000	female	high s	0.1919191919191919
260000	female	gradu	0.2525252525252525
630000	female	unive	0.6262626262626263
70000	male	unive	0.06060606060606061
250000	male	gradu	0.24242424242424243
50000	female	high s	0.04040404040404041

Figure 5.27: Min Max Normalization – Converted Data

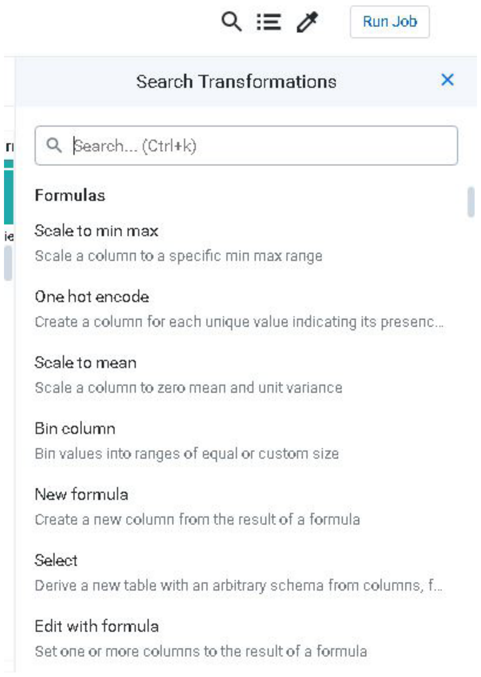
Next, we will convert categorical columns into one-hot encoded columns.

1. **`One-hot Representation:`** In one-hot representation, categorical columns such as Gender will be converted into one-hot encoded columns. As an example, Gender may have two values (Male and Female), so in order to represent or encode this information correctly, two new columns will be added to (one for Male and other for Female) our dataset. And value for columns Male will be one if the row represents an entry for a Male or Female otherwise.

   We will implement one-hot representation using Google Dataprep. Please follow the following steps:

   - Click on **`Search Transformations`** on the **`Flow`** page.

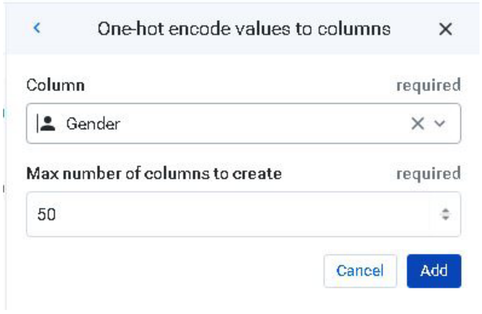
- Search for one-hot encode, and select it as shown in *Figure 5.28*



Scale to min max
Scale a column to a specific min max range
One hot encode
Create a column for each unique value indicating its presenc...
Scale to mean
Scale a column to zero mean and unit variance
Bin column
Bin values into ranges of equal or custom size
New formula
Create a new column from the result of a formula
Select
Derive a new table with an arbitrary schema from columns, f...
Edit with formula
Set one or more columns to the result of a formula

Figure 5.28: *Select One hot Encode*

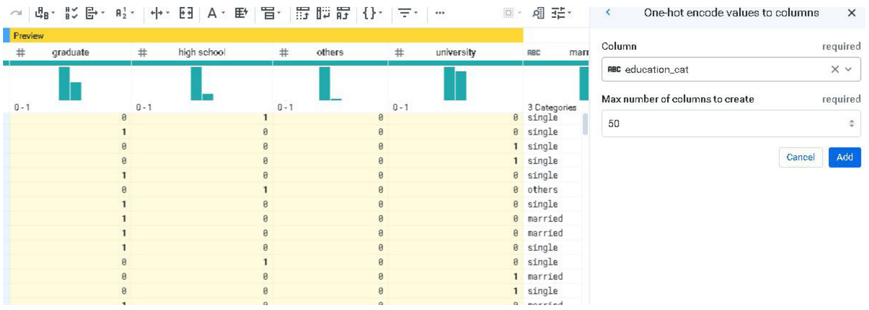
- Next, select the column onto which you want to apply the transformation
- Select the column as **`Gender`** and Maximum number of columns as **`50`**. This puts a capping on maximum number of columns we can have after transformations as shown in *Figure 5.29*. This will add a new step in your current recipe.



Column required
Gender
Max number of columns to create required
50

Figure 5.29: *One hot Encode – Configuration*

- Similarly, we can also apply the same transformations to the columns `education_cat` and `marriage_cat` as shown in *Figure 5.30.*



graduate	high school	others	university	marr
0 - 1	0 - 1	0 - 1	0 - 1	3 Categories
0	1	0	0	single
1	0	0	0	single
0	0	0	1	single
0	0	0	1	single
1	0	0	0	single
0	1	0	0	others
1	0	0	0	single
1	0	0	0	married
1	0	0	0	married
1	0	0	0	single
0	1	0	0	single
0	0	0	1	married
0	0	0	1	single

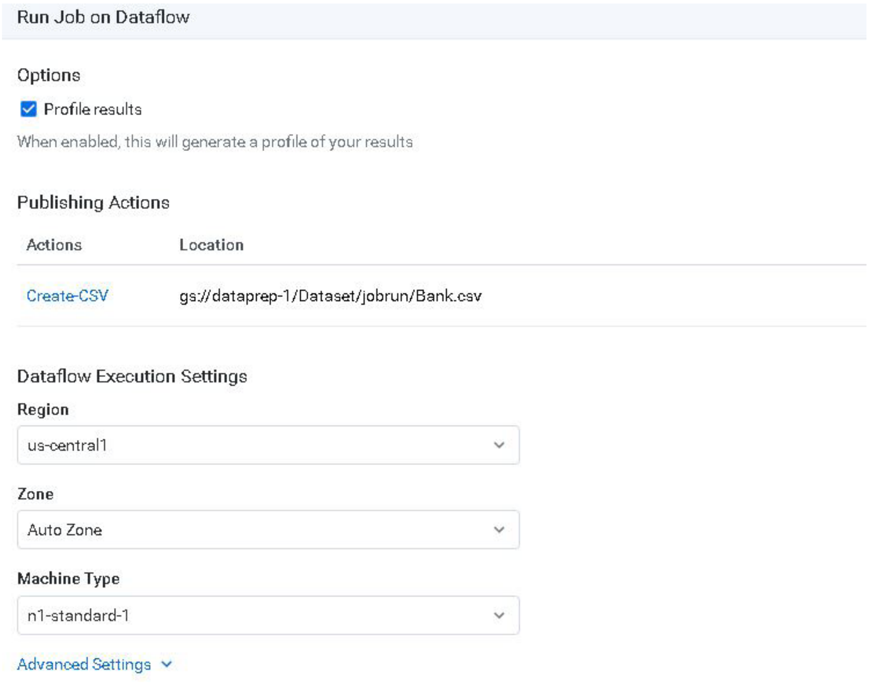
Column: education_cat (required)

Max number of columns to create: 50 (required)

Figure 5.30: *One-hot Encode – Converted Data*

All these transformations will be added to the recipe that will run over Google Dataflow. Please follow the following steps to configure job in Dataflow:

2. Click on the **`Run Job`** button on the top-right section in the recipe page to configure the job for Dataflow as shown in *Figure 5.31.*



Run Job on Dataflow

Options

☑ Profile results

When enabled, this will generate a profile of your results

Publishing Actions

Actions	Location
Create-CSV	gs://dataprep-1/Dataset/jobrun/Bank.csv

Dataflow Execution Settings

Region: us-central1

Zone: Auto Zone

Machine Type: n1-standard-1

Figure 5.31: *Dataflow Job Creation*

3. Define the values for following Dataflow execution settings:
    - **`Region:`** Region in which Dataflow job will run
    - **`Zone:`** Zone in which the Dataflow job will run
    - **`Machine-Type:`** Type of machine to be used for execution of the tasks submitted to Dataflow

    You can also change the publishing actions of the job.

    Click on the **`Create-Csv`** option to perform this. However, output format can be csv, json, or Avro.

    We can also define if we want to create a new file after every run, or it should be appended to the previous file itself.

    Click on the **`Update`** button to save the changes as shown in *Figure 5.32:*

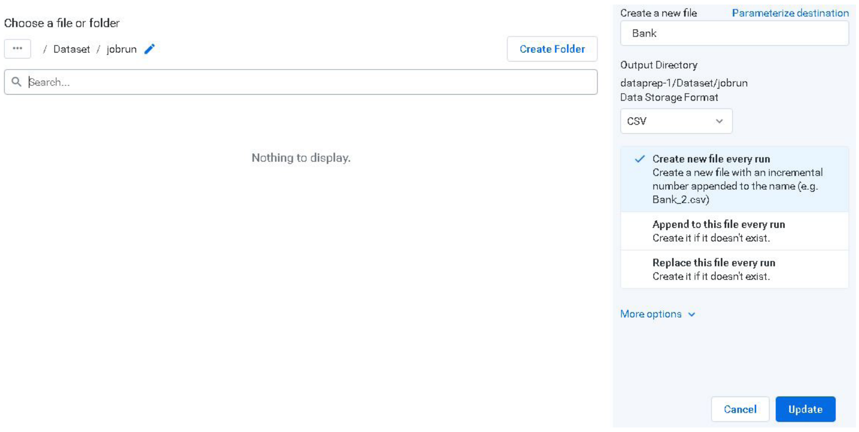
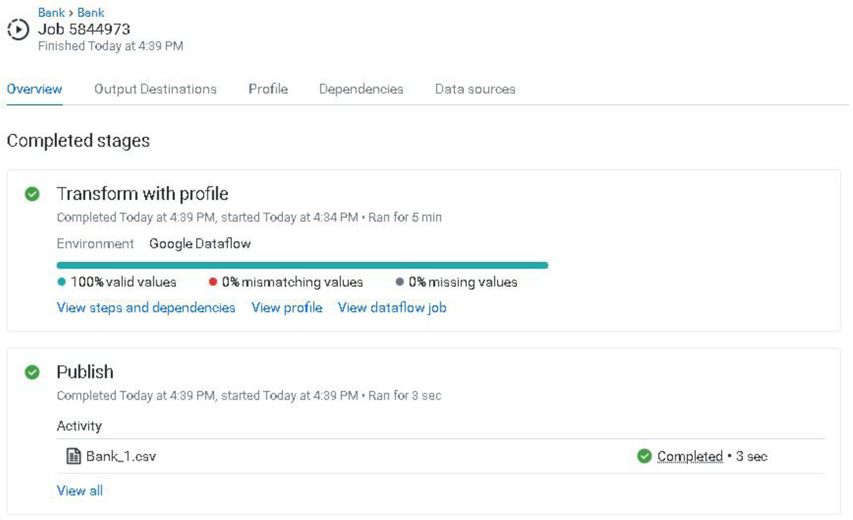

Choose a file or folder
... / Dataset / jobrun
Create Folder
Search...
Nothing to display.
Create a new file
Parameterize destination
Bank
Output Directory
dataprep-1/Dataset/jobrun
Data Storage Format
CSV
✓ Create new file every run
Create a new file with an incremental number appended to the name (e.g. Bank_2.csv)
Append to this file every run
Create it if it doesn't exist.
Replace this file every run
Create it if it doesn't exist.
More options
Cancel
Update

Figure 5.32: *Dataflow Job Configuration*

4. After defining the configuration of a job, click on **`Run Job`** to start the job in Dataflow

5. We can also see the status of jobs on the overview page as shown in *Figure 5.33.*


Bank › Bank
Job 5844973
Finished Today at 4:39 PM

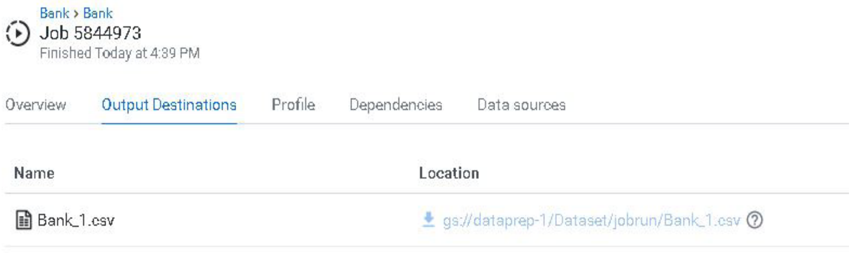
Completed stages

Transform with profile
Completed Today at 4:39 PM, started Today at 4:34 PM • Ran for 5 min
Environment Google Dataflow
● 100% valid values ● 0% mismatching values ● 0% missing values

Publish
Completed Today at 4:39 PM, started Today at 4:39 PM • Ran for 3 sec
Activity
Bank_1.csv ✓ Completed • 3 sec

Figure 5.33: *Dataflow Job Overview*

6. After the successful completion of the job, you can switch to the **`Output Destinations`** tab to check the location of the processed file as shown in *Figure 5.34.*


Bank › Bank
Job 5844973
Finished Today at 4:39 PM

Name	Location
Bank_1.csv	gs://dataprep-1/Dataset/jobrun/Bank_1.csv

Figure 5.34: *Google Dataprep – Converted file in Bucket*

With this, we are done with the pre-processing of our dataset that includes min-max normalization and one-hot encoding for categorical attributes or features. In the next section, we will see how this processed data will be used by Google Dataproc to train a prediction system that will identify if the customer is going to default or not.

Google Dataproc

Availability of an efficient and scalable computational engine is always a major concern for training the machine learning models. Google Dataproc tries to solve this problem by providing a fully managed Hadoop environment that enables the users to perform batch processing, stream processing, and building of machine learning models. It automates cluster management that provides features to the users to create and manage them easily. The important thing is you can switch on the cluster based on the requirement that in-turn reduces the cost. Google Dataproc scales up automatically whenever it is required. It has completely automated the task of managing Hadoop clusters such that the user can now focus more on the task of generating the jobs without worrying about the administrative overheads of managing clusters.

Model Training and Evaluation

We will see how Google Dataproc can be used for training a prediction model on our dataset. Please follow the following steps:

1. Go to URL **https://cloud.google.com/** and login with your credentials
2. Select an option **`Clusters`** under **`Dataproc`** from the left-side menu bar as shown in *Figure 5.35*

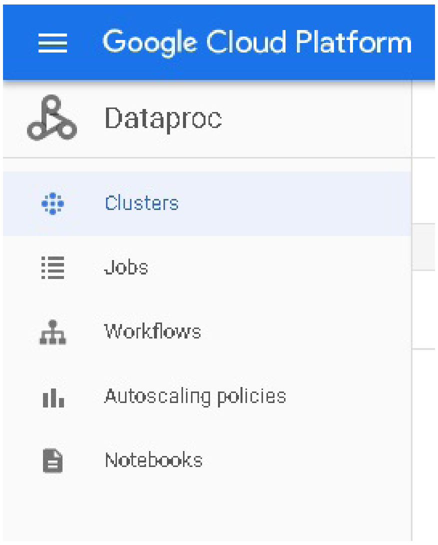


Figure 5.35: *Google Dataprep*

3. Then, click on the link **`CREATE CLUSTER`** as shown in *Figure 5.36*

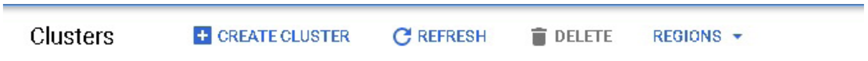
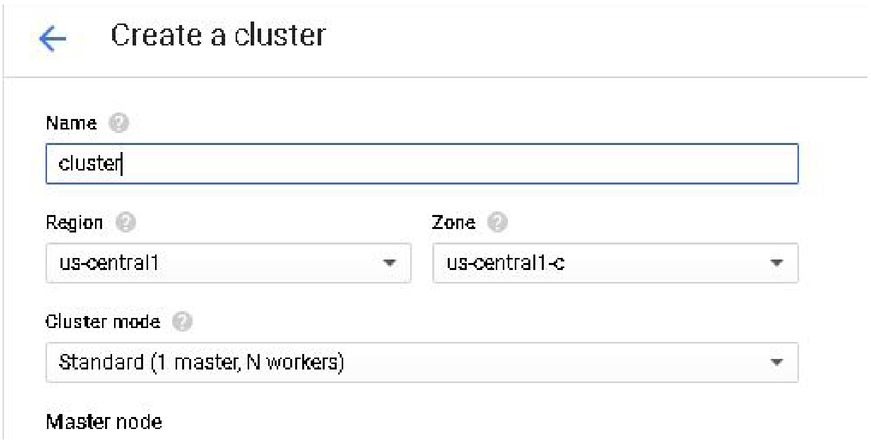


Figure 5.36: *Google Dataprep – Create Cluster*

4. This opens up the cluster configuration windows as shown in *Figure 5.37:*



Create a cluster

Name: cluster

Region: us-central1

Zone: us-central1-c

Cluster mode: Standard (1 master, N workers)

Master node

Figure 5.37: *Define Cluster Details*

5. Define the configuration for the cluster as per the following values:

   - **`Name:`** Define a name for your cluster
   - **`Region:`** Define a region in which the cluster will be deployed
   - **`Cluster Mode:`** Define the cluster model as Standard i.e. 1 master and multiple workers as shown in *Figure 5.37*.

6. Set the Master node configuration as shown in *Figure 5.38.*

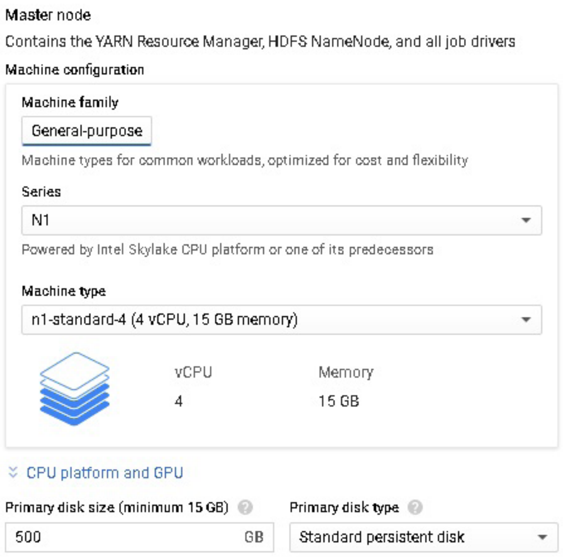

Master node
Contains the YARN Resource Manager, HDFS NameNode, and all job drivers
Machine configuration
Machine family
General-purpose
Machine types for common workloads, optimized for cost and flexibility
Series
N1
Powered by Intel Skylake CPU platform or one of its predecessors
Machine type
n1-standard-4 (4 vCPU, 15 GB memory)
vCPU 4
Memory 15 GB
CPU platform and GPU
Primary disk size (minimum 15 GB): 500 GB
Primary disk type: Standard persistent disk

Figure 5.38: *Define Machine Family*

Set the worker node configuration as shown in *Figure 5.39*. define the number of workers required (minimum 2). For this use case, we will go with 2.

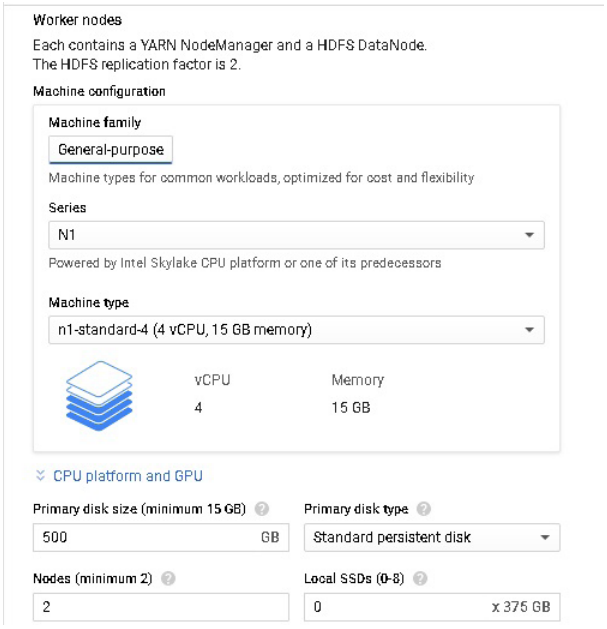
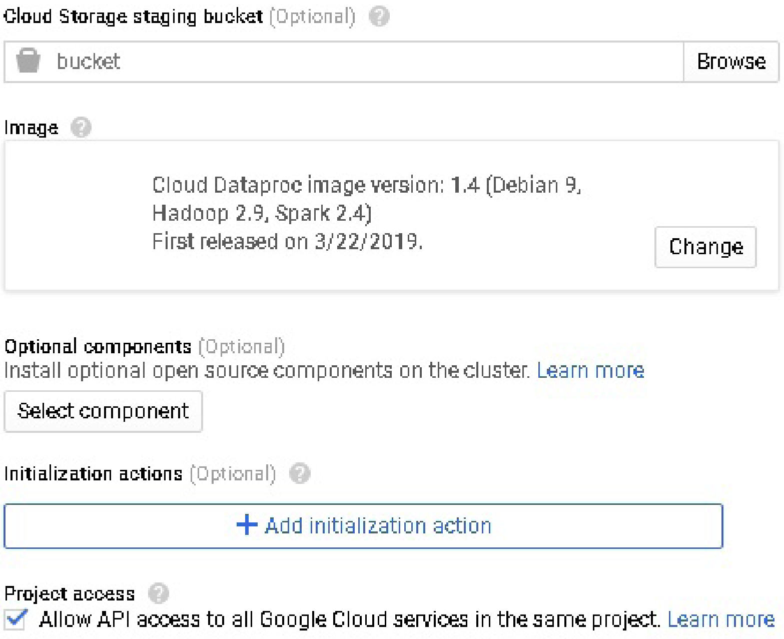

Worker nodes
Each contains a YARN NodeManager and a HDFS DataNode.
The HDFS replication factor is 2.
Machine configuration
Machine family
General-purpose
Machine types for common workloads, optimized for cost and flexibility
Series
N1
Powered by Intel Skylake CPU platform or one of its predecessors
Machine type
n1-standard-4 (4 vCPU, 15 GB memory)
vCPU 4
Memory 15 GB
CPU platform and GPU
Primary disk size (minimum 15 GB): 500 GB
Primary disk type: Standard persistent disk
Nodes (minimum 2): 2
Local SSDs (0-8): 0 x 375 GB

Figure 5.39: *Worker Node Configuration*

7. Next, select the version of Dataproc and Spark for your project. For our use case, Dataproc image version 1.4 and Spark version 2.4 will be used as shown in *Figure 5.40*.


Cloud Storage staging bucket (Optional)

bucket

Image

Cloud Dataproc image version: 1.4 (Debian 9, Hadoop 2.9, Spark 2.4)
First released on 3/22/2019.

Optional components (Optional)
Install optional open source components on the cluster. Learn more

Initialization actions (Optional)

Project access
✓ Allow API access to all Google Cloud services in the same project. Learn more

Figure 5.40: *Google Dataproc – Select Storage Bucket*

With this, we can now run a job in Hadoop cluster that has been created just now. Please follow the following steps to configure the code and job to run on the Hadoop cluster.

Note: The codebase for this exercise can be downloaded from GitHub URL **https://github.com/automl-book/AutoML/tree/master/chapter%205**

8. Create a file named as `modelTrain.py`.

9. Copy the following code and paste it into the file created in the previous step

```
from __future__ import print_function
from pyspark.context import SparkContext
from pyspark.ml.linalg import Vectors
from pyspark.ml.classification import
```

```
LogisticRegression
from pyspark.sql.session import SparkSession
from pyspark.ml.feature import StringIndexer,
VectorAssembler
from pyspark.ml import Pipeline
from pyspark.ml.evaluation import
BinaryClassificationEvaluator
from pyspark.mllib.evaluation import
MulticlassMetrics

sc = SparkContext()
spark = SparkSession(sc)

# Read the data from BigQuery as a Spark Dataframe.
bank_data = spark.read.format("bigquery").
option("table", "Bank.Defaulters").load()

df = bank_data.select("LIMIT_
BAL","SEX","EDUCATION","MARRIAGE","AGE","PAY_0","
PAY_2","PAY_3","PAY_4","PAY_5","PAY_6","BILL_
AMT1"," BILL_AMT2","BILL_AMT3","BILL_AMT4","BILL_
AMT5","BILL_AMT6","PAY_AMT1","PAY_AMT2","PAY_
AMT3","PAY_AMT4"," PAY_AMT5","PAY_AMT6","payment")
cols = df.columns

stages = []

label_stringIdx = StringIndexer(inputCol =
'payment', outputCol = 'label')
stages += [label_stringIdx]
numericCols = ["LIMIT_
BAL","SEX","EDUCATION","MARRIAGE"," AGE","PAY_0"
,"PAY_2","PAY_3","PAY_4","PAY_5","PAY_6"," BILL_
AMT1","BILL_AMT2","BILL_AMT3","BILL_AMT4","BILL_
AMT5", "BILL_AMT6","PAY_AMT1","PAY_AMT2","PAY_
AMT3","PAY_AMT4","PAY_AMT5","PAY_AMT6"]
assembler_Inputs = numericCols
assembler = VectorAssembler(inputCols=assembler_
Inputs, outputCol="features")
stages += [assembler]
```

```
pipeline = Pipeline(stages = stages)
pipelineModel = pipeline.fit(df)
df = pipelineModel.transform(df)
selectedCols = ['label', 'features'] + cols
df = df.select(selectedCols)

train, test = df.randomSplit([0.8, 0.3], seed = 2018)

logistic = LogisticRegression(featuresCol =
'features', labelCol = 'label', maxIter=10)
model_instance = logistic.fit(train)
model_instance.write().save("gs://automl-1//model")

# To generate predictions from model

predictions = model_instance.transform(test)
predictions.
select("payment","prediction","probability").
show(truncate=False)

predictions.show()


evaluator = BinaryClassificationEvaluator()
print('Area Under ROC', evaluator.
evaluate(predictions))

predictions_data = predictions.
select("label","prediction").rdd.map(tuple)
confusion_matrix = MulticlassMetrics(predictions_
data)
print("recall: ",confusion_matrix.recall())
print("precision: ",confusion_matrix.precision())
```

The previous code uses the dataset from Google BigQuery table. The columns have been divided into two parts, namely, feature vector that serves as an input and class column that serves as a target column or variable. The training set of Logistic Regression model contains 80% of the records whereas the rest 20% will be used to evaluate a trained model.

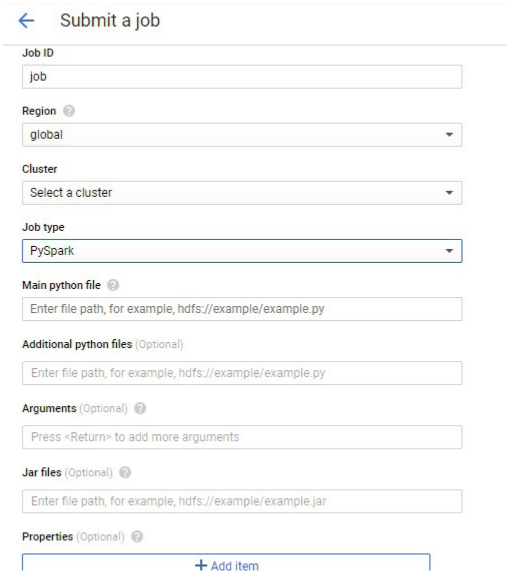
We have also used Multiclass Metrics class to calculate the precision and recall of a trained model.

Run the following command to upload python file to Google Storage service where the name of `python_file` is `modelTrain.py` and `<bucket-name>` is the name of the bucket.

```
gsutil cp <python_file> gs://<bucket-name>
```

Now, you can run your job in Hadoop cluster managed by Google Dataproc. Please follow the following steps to run your job in Hadoop cluster.

1. Go to Dataproc Submit job page as shown in *Figure 5.41* and define the following:
   - **`Cluster:`** Select the cluster from a list of available clusters
   - **`Region:`** Define region in which the job will run. For our use case, select global.
   - **`Job type:`** Pyspark
   - **`Main python file:`** Define the path of a python file(i.e. `modelTrain.py`) stored in Google Storage service.
   - **`Jar files:`** Add gs://spark-lib/bigquery/spark-bigquery-latest.jar. This enables your code to integrate BigQuery a pisto spark environment.


Submit a job

Job ID: job

Region: global

Cluster: Select a cluster

Job type: PySpark

Main python file: Enter file path, for example, hdfs://example/example.py

Additional python files (Optional): Enter file path, for example, hdfs://example/example.py

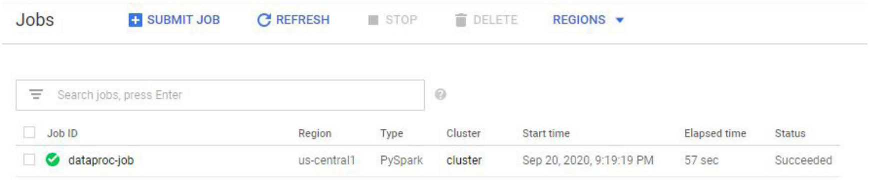
Arguments (Optional): Press <Return> to add more arguments

Jar files (Optional): Enter file path, for example, hdfs://example/example.jar

Properties (Optional)

Figure 5.41: *Google Dataproc – Submit Model Training Job*

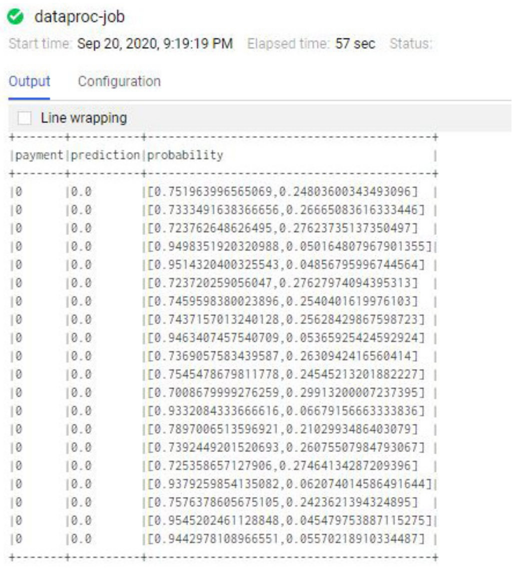
2. Then, click on the **`Submit`** button to submit the job to Hadoop cluster managed by Google Dataproc.
3. Go to Dataproc jobs console to view the progress of job as shown in *Figure 5.42*. You will see a green tick before the job name if it was successful; red otherwise.



Job ID	Region	Type	Cluster	Start time	Elapsed time	Status
✓ dataproc-job	us-central1	PySpark	cluster	Sep 20, 2020, 9:19:19 PM	57 sec	Succeeded

Figure 5.42: *Google Dataproc – Job Overview*

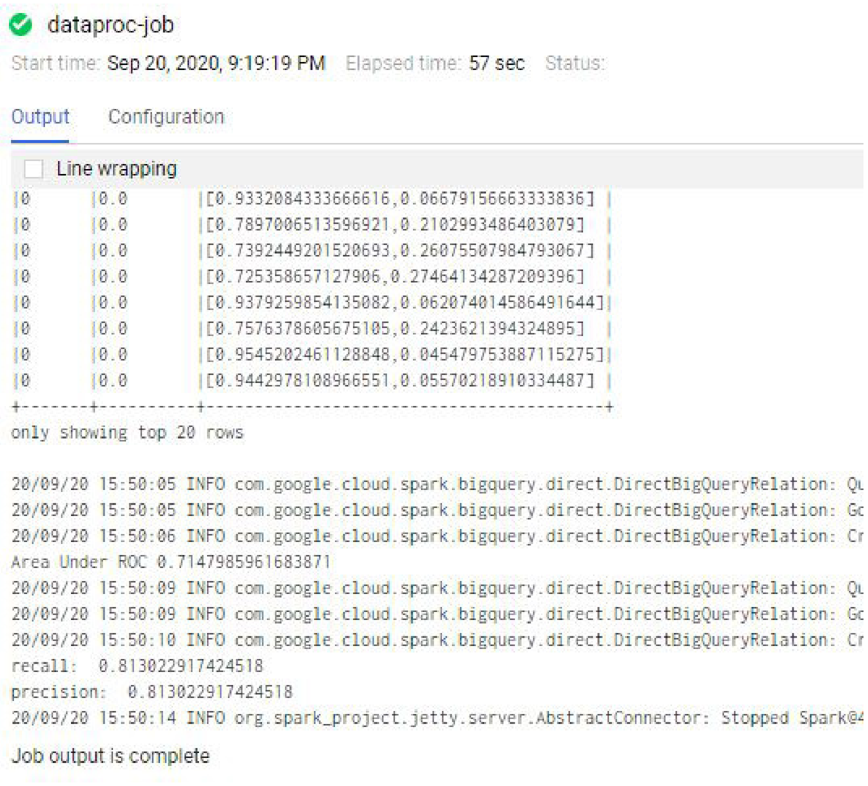
4. Click on the individual job ID to see the model prediction over the test data as shown in *Figure 5.43*. The definition of each of the columns are as follows:
   - **`payment:`** Shows the actual value for each data points in the test set
   - **`prediction:`** Shows the prediction from model for each of data points in the test set
   - **`Probability:`** Shows the probability distribution for each data point across both classes i.e. Yes and No. Here, the Class label with higher probability value will be assigned to that data point.



```
+-------+----------+----------------------------------------+
|payment|prediction|probability                             |
+-------+----------+----------------------------------------+
|0      |0.0       |[0.751963996565069,0.24803600343493096]  |
|0      |0.0       |[0.7333491638366656,0.26665083616333446] |
|0      |0.0       |[0.723762648626495,0.27623735137350497]  |
|0      |0.0       |[0.9498351920320988,0.050164807967901355]|
|0      |0.0       |[0.9514320400325543,0.04856795996744564] |
|0      |0.0       |[0.723720259056047,0.27627974094395313]  |
|0      |0.0       |[0.7459598380023896,0.2540401619976103]  |
|0      |0.0       |[0.7437157013240128,0.25628429867598723] |
|0      |0.0       |[0.9463407457540709,0.05365925424592924] |
|0      |0.0       |[0.7369057583439587,0.2630942416560414]  |
|0      |0.0       |[0.7545478679811778,0.24545213201882227] |
|0      |0.0       |[0.7008679999276259,0.29913200007237395] |
|0      |0.0       |[0.9332084333666616,0.06679156663333836] |
|0      |0.0       |[0.7897006513596921,0.2102993486403079]  |
|0      |0.0       |[0.7392449201520693,0.26075507984793067] |
|0      |0.0       |[0.725358657127906,0.27464134287209396]  |
|0      |0.0       |[0.9379259854135082,0.062074014586491644]|
|0      |0.0       |[0.7576378605675105,0.2423621394324895]  |
|0      |0.0       |[0.9545202461128848,0.045479753887115275]|
|0      |0.0       |[0.9442978108966551,0.05570218910334487] |
+-------+----------+----------------------------------------+
```

Figure 5.43: *Google Dataproc – Sample Model Result*

5. The job details also include the performance metrices (recall, precision, and ROC) of the model over the test set as shown in *Figure 5.44*.



dataproc-job

Start time: Sep 20, 2020, 9:19:19 PM Elapsed time: 57 sec Status:

```
|0       |0.0       |[0.9332084333666616,0.0667915666333836] |
|0       |0.0       |[0.7897006513596921,0.2102993486403079]  |
|0       |0.0       |[0.7392449201520693,0.2607550798479307] |
|0       |0.0       |[0.725358657127906,0.274641342872093961  |
|0       |0.0       |[0.9379259854135082,0.06207401458649164]|
|0       |0.0       |[0.7576378605675105,0.2423621394324895]  |
|0       |0.0       |[0.9545202461128848,0.04547975388711527]|
|0       |0.0       |[0.9442978108966551,0.0557021891033448 7] |
+-------+----------+---------------------------------------+
only showing top 20 rows

20/09/20 15:50:05 INFO com.google.cloud.spark.bigquery.direct.DirectBigQueryRelation: Qu
20/09/20 15:50:05 INFO com.google.cloud.spark.bigquery.direct.DirectBigQueryRelation: Gc
20/09/20 15:50:06 INFO com.google.cloud.spark.bigquery.direct.DirectBigQueryRelation: Cr
Area Under ROC 0.7147985961683871
20/09/20 15:50:09 INFO com.google.cloud.spark.bigquery.direct.DirectBigQueryRelation: Qu
20/09/20 15:50:09 INFO com.google.cloud.spark.bigquery.direct.DirectBigQueryRelation: Gc
20/09/20 15:50:10 INFO com.google.cloud.spark.bigquery.direct.DirectBigQueryRelation: Cr
recall:  0.813022917424518
precision:  0.813022917424518
20/09/20 15:50:14 INFO org.spark_project.jetty.server.AbstractConnector: Stopped Spark@4
```

Job output is complete

Figure 5.44: *Google Dataproc – Model Evaluation*

Recall of the trained model is greater than 0.80; it means for 80% of the data points that belong to the label "Yes" in the test set; the model has correctly classified them as "Yes." In other words, our trained model can identify non-legitimate customer for 80% of the cases.

Precision of trained model is greater than 0.80; it means for 80% of the data points that have been predicted by the model as "Yes" actually belong to the label "Yes." In other words, our trained model can precisely identify non-legitimate customer with a precision of 80%.

We have also calculated the ROC value for this model, as this is a binary classification problem. ROC stands for Receiver Operating Characteristics; it defines the model's ability to clearly discriminate between the two classes, namely, Yes and No. Mathematically, it can be defined as the ratio of True Positive Rate to False Positive Rate as follows:

ROC = True Positive Rate / False Positive Rate,

where True Positive Rate is also known as Recall and False Positive Rate can be defined as

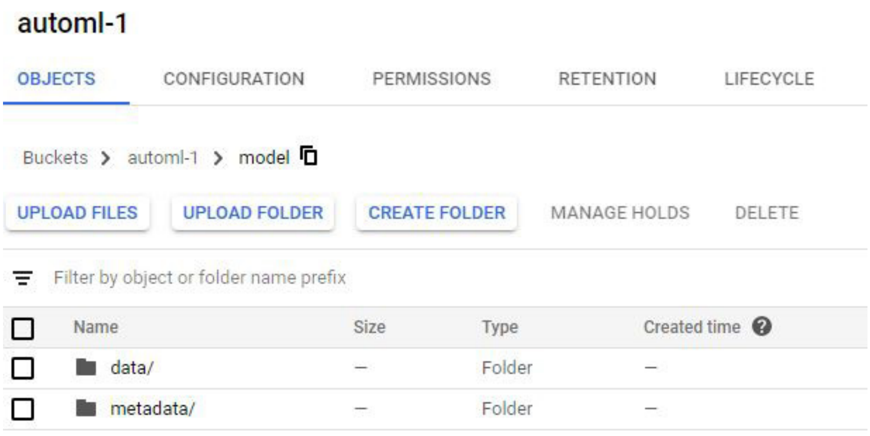
False Positive Rate = False Positive / (False Positive + True Negative)

We have already discussed about False Positive and True Negative in *Chapter 3*. ROC can take any value between 0 and 1. Different values of ROC can be defined as follows:

- **ROC=0:** The ROC value of 0 means the model is performing prediction in the reverse direction i.e. classifying the data point that should belong to 0 to 1 and vice-versa.
- **ROC=1:** The ROC value of 1 means the model has an excellent capability to clearly discriminate between the two classes.
- **ROC>0.5:** The ROC value greater than 0.5 means that the model has good capability to discriminate between the two classes.
- **ROC<0.5:** The ROC value less than 0.5 means that the model has bad capability to discriminate between the two classes.
- **ROC=0.5:** The ROC value of 0.5 means that the model is not able to discriminate between the two classes.

As the ROC value for our trained model is 0.714 which is greater than 0.5 and less than 1.0, it means the model has good capability to discriminate between the two classes, namely, Yes and No. However, you can increase the number of epochs to improve the ROC value.

The trained model will be stored in the bucket (i.e. `automl-1`) created in Google Storage service as soon as the job execution has finished as shown in *Figure 5.45*.



Name	Size	Type	Created time
data/	–	Folder	–
metadata/	–	Folder	–

Figure 5.45: *Google Dataproc – Model in Storage Bucket*

Model Prediction

Now, consider a scenario where you want to use this trained model to get the prediction for a new customer or user who is being on boarded into the bank. We also need to a create job to get the prediction from the model. Please follow the following steps to configure the job for the prediction process.

1. Download a CSV file named as prediction.csv from GitHub **https://github.com/automl-book/AutoML/tree/master/chapter%205**.
2. This file contains details of a new customer or user who is going to be on boarded to the bank.
3. Run the following command to upload this CSV file to the bucket i.e. `automl-1`:

   ```
   ! gsutil cp prediction.csv gs://automl-1
   ```

   Next, we need to read the model file and prediction file that has been stored in the bucket and generate a prediction out of it. Please follow the following steps to configure the code and job to run on the Hadoop cluster.

   - Create a file named as `modelPredict.py`.

- Copy the code as shown follows, and paste it into file created in previous step

```
from __future__ import print_function
from pyspark.context import SparkContext
from pyspark.ml.linalg import Vectors
from pyspark.ml.classification import
LogisticRegression
from pyspark.sql.session import SparkSession
from pyspark.ml.feature import StringIndexer,
VectorAssembler
from pyspark.ml import Pipeline
from pyspark.ml.evaluation import
BinaryClassificationEvaluator
from  pyspark.ml.classification import
LogisticRegressionModel
from pyspark.mllib.evaluation import
MulticlassMetrics

sc = SparkContext()
spark = SparkSession(sc)

df = spark.read.options(header='True',
inferSchema= 'True', delimiter=',').csv("gs://
automl-1/prediction.csv")
cols = df.columns

stages = []

label_stringIdx = StringIndexer(inputCol =
'payment' , outputCol = 'label')
stages += [label_stringIdx]
numericCols = ["LIMIT_
BAL","SEX","EDUCATION","MARRIAGE","AGE"," PA
Y_0","PAY_2","PAY_3","PAY_4","PAY_5","PAY_6"
,"BILL_AMT1", "BILL_AMT2","BILL_AMT3","BILL_
AMT4","BILL_AMT5","BILL_AMT6", "PAY_AMT1","PAY_
AMT2","PAY_AMT3","PAY_AMT4","PAY_AMT5"," PAY_
AMT6"]
assembler_Inputs = numericCols
assembler =
VectorAssembler(inputCols=assembler_Inputs,
outputCol="features")
stages += [assembler]

pipeline = Pipeline(stages = stages)
pipelineModel = pipeline.fit(df)
df = pipelineModel.transform(df)
selectedCols = ['label', 'features'] + cols
df = df.select(selectedCols)
model = LogisticRegressionModel.load("gs://
automl-1/model")

predictions = model.transform(df)
predictions.select("prediction","probability").
show(truncate=False)
predictions.show()

# To dump prediction result as csv into Google
Cloud Storage bucket

predictions.withColumn("probability",
col("probability").cast("string")).
select("ID","prediction","probability").write.
csv('gs://automl-1/prediction')
```

4. Next, run the following command to upload python file to Google Storage service where the name of `python_file` is `modelPredict.py` and `<bucket-name>` is the name of the bucket.

   ```
   ! gsutil cp modelPredict.py gs://<bucket-name>
   ```

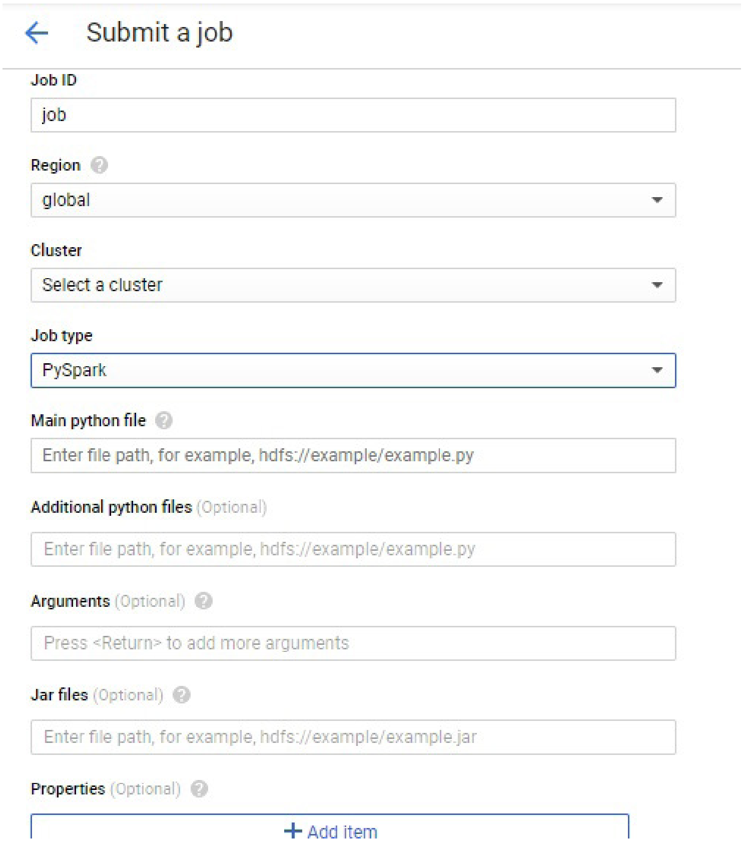
5. Go to Dataproc Submit job page to configure the job to run on Hadoop cluster as shown in *Figure 5.46* and define the following:

   - **`Cluster:`** Select the cluster from a list of available clusters
   - **`Region:`** Define the region in which a job will run

   For our use case, select global.

   - **`Job type:`** Pyspark

- **`Main python file:`** Define the path of a python file stored in Google Storage service. For prediction, python file name is `modelPredict.py`



Submit a job

Job ID: job

Region: global

Cluster: Select a cluster

Job type: PySpark

Main python file: Enter file path, for example, hdfs://example/example.py

Additional python files (Optional): Enter file path, for example, hdfs://example/example.py

Arguments (Optional): Press <Return> to add more arguments

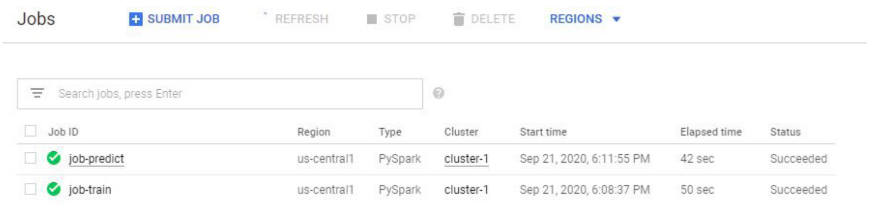
Jar files (Optional): Enter file path, for example, hdfs://example/example.jar

Properties (Optional)

Figure 5.46: *Google Dataproc – Prediction Job Configuration*

6. Then, click on the **`Submit`** button to submit the job to Hadoop cluster managed by Google Dataproc

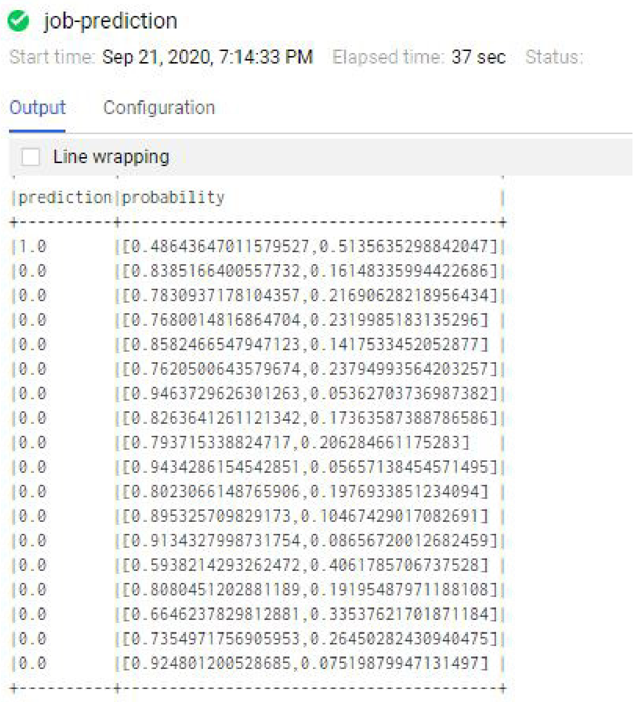
7. Go to Dataproc jobs console to view the progress of the job as shown in *Figure 5.47:*



Job ID	Region	Type	Cluster	Start time	Elapsed time	Status
job-predict	us-central1	PySpark	cluster-1	Sep 21, 2020, 6:11:55 PM	42 sec	Succeeded
job-train	us-central1	PySpark	cluster-1	Sep 21, 2020, 6:08:37 PM	50 sec	Succeeded

Figure 5.47: *Google Dataproc – Prediction Job Overview*

8. Click on job ID to see the prediction from the trained model as shown in *Figure 5.48.*
   - **`prediction:`** Shows the prediction from the model for each of the data point in the test set
   - **`probability:`** Shows probability distribution for each data point across both the classes, namely, Yes and No. Here, the class label with a higher probability value will be assigned to that data point.

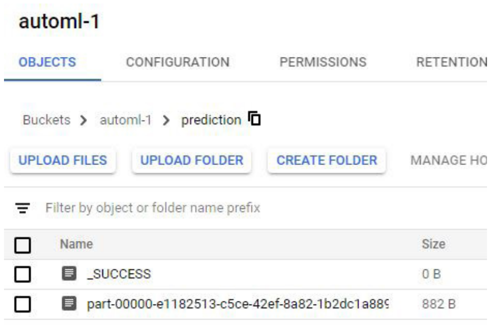


```
|prediction|probability                            |
+----------+---------------------------------------+
|1.0       |[0.4864364701157952 7,0.513563529884204 7]|
```

```
|prediction|probability                              |
+----------+-----------------------------------------+
|1.0       |[0.48643647011579527,0.5135635298842047]|
|0.0       |[0.8385166400557732,0.16148335994422686]|
|0.0       |[0.7830937178104357,0.21690628218956434]|
|0.0       |[0.7680014816864704,0.2319985183135296] |
|0.0       |[0.8582466547947123,0.1417533452052877] |
|0.0       |[0.7620500643579674,0.23794993564203257]|
|0.0       |[0.946372962630126 3,0.05362703736987382]|
|0.0       |[0.8263641261121342,0.17363587388786586]|
|0.0       |[0.793715338824717,0.206284661175283]   |
|0.0       |[0.9434286154542851,0.05657138454571495]|
|0.0       |[0.8023066148765906,0.1976933851234094] |
|0.0       |[0.895325709829173,0.10467429017082691] |
|0.0       |[0.9134327998731754,0.08656720012682459]|
|0.0       |[0.5938214293262472,0.4061785706737528] |
|0.0       |[0.8080451202881189,0.19195487971188108]|
|0.0       |[0.6646237829812881,0.33537621701871184]|
|0.0       |[0.7354971756905953,0.26450282430940475]|
|0.0       |[0.924801200528685,0.07519879947131497] |
+----------+-----------------------------------------+
```

Figure 5.48: *Google Dataproc – Prediction Job Result*

Figure 5.48 shows the predictions for each of the data points corresponding to a new customer or user and provides us with a probability whether a new customer or user is going to be a legitimate customer or fraud one. There are many raw features for each customer that may not be possible to send via REST API for prediction. In such cases, new customers' features should be included in CSV file and a separate job will be deployed over Hadoop cluster. This job will read the trained model and new CSV file from the cloud storage to generate the predictions. The current job also dumps the prediction results as CSV in Google Cloud Storage bucket as shown in *Figure 5.49*.

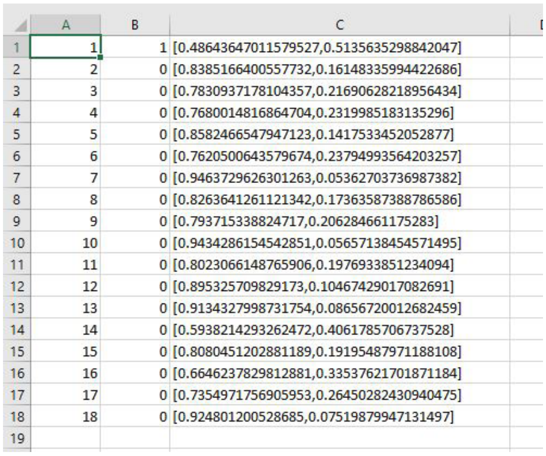


automl-1

Name	Size
_SUCCESS	0 B
part-00000-e1182513-c5ce-42ef-8a82-1b2dc1a88	882 B

Figure 5.49: *Google Dataproc – Prediction Result in Storage Bucket*

Click on the file with prefix as "part" to download predictions from model. Content of downloaded file would be as shown in *Figure 5.50*.



	A	B	C
1	1	1	[0.48643647011579527,0.513563529884204 7]
2	2	0	[0.8385166400557732,0.16148335994422686]
3	3	0	[0.7830937178104357,0.21690628218956434]
4	4	0	[0.7680014816864704,0.2319985183135296]
5	5	0	[0.8582466547947123,0.1417533452052877]
6	6	0	[0.7620500643579674,0.23794993564203257]
7	7	0	[0.9463729626301263,0.05362703736987382]
8	8	0	[0.8263641261121342,0.17363587388786586]
9	9	0	[0.793715338824717,0.206284661175283]
10	10	0	[0.9434286154542851,0.05657138454571495]
11	11	0	[0.8023066148765906,0.1976933851234094]
12	12	0	[0.895325709829173,0.10467429017082691]
13	13	0	[0.9134327998731754,0.08656720012682459]
14	14	0	[0.5938214293262472,0.4061785706737528]
15	15	0	[0.8080451202881189,0.19195487971188108]
16	16	0	[0.6646237829812881,0.33537621701871184]
17	17	0	[0.7354971756905953,0.26450282430940475]
18	18	0	[0.92480120052868 5,0.07519879947131497]
19			

Figure 5.50

Conclusion

In this chapter, we have seen how Google's data analysis, preparation, and processing services can be used for training of machine learning models. With this, we are concluding our exciting journey of discussions on AutoML, Google's AI Platform and Google's data preparation and processing engines. With this, we conclude this book on Google AutoML and AI Platform. We sincerely hope that this would enable you to go further in your knowledge on GCP AutoML and AI Platform, and you will be able to use these features in your applications and projects.

Index