Tampere University

Roope Mäkelä

END-TO-END MACHINE LEARNING LEVERAGING CLOUD SERVICE PROVIDERS

A Comparison of AWS and GCP

Faculty of Information Technology and Communication Sciences
Bachelor’s Thesis
Supervisor: Tapio Elomaa
April 2022

ABSTRACT

Roope Mäkelä: End-to-End Machine Learning Leveraging Cloud Service Providers - A Comparison of AWS and GCP
Bachelor's Thesis
Tampere University
Science and Engineering
April 2022

Cloud service providers give users resources as service products through the internet. Using cloud services has many benefits compared to physical systems. The high scalability means services automatically adjust to the demand of the user. Within machine learning, where data sets are very large and can change from project to project, the scalability of cloud services is particularly advantageous.

This thesis examines how the user can leverage specific services from cloud service providers in order to implement an end-to-end machine learning project. To achieve this, a theoretical example with Google Cloud Platform is conducted. Mainly the capabilities of the services are examined, without going deep into the technology behind them.

The thesis compares two cloud service providers for implementing a machine learning project: Amazon Web Services and Google Cloud Platform. The comparison focuses on the different services provided for each stage of implementation and how the capabilities of the services differ across the platforms. The comparison is made using existing research and information from the websites and product offerings of the respective platforms.

The comparison shows that neither platform is universally the better choice. The decision will largely depend on the specific project and what factors are most important to the user. While Amazon generally excels in machine learning algorithms and incorporating external models, Google offers a larger amount of relevant services across the implementation process. In the end, the platforms are ranked in terms of individual factors, but a general overall choice cannot be made.

Keywords: Amazon Web Services, Google Cloud Platform, Machine Learning, Cloud Computing, Cloud Service

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

PREFACE

When choosing this thesis topic, I wanted to find something that would combine business and machine learning together. While this thesis does not directly look at the business side of things, I feel the scope reduces the threshold between business and technology, where the reader does not necessarily need to be exposed to programming to be able to understand the contents of the thesis. Through writing the thesis I was also able to examine how the internet can provide universal access to knowledge and resources that previously have been available only for a select few.

I enjoyed the entire research and writing process because I was able to see how much more comfortable I became with the topic. Initially, I would not have been able to say anything about the two platforms, but now I would feel comfortable discussing them and their services with others. I hope that the research conducted for this thesis is something I can use in my future business and computer science studies.

Here I would also like to thank my supervisor, Tapio Elomaa, for their support throughout the entire process; my parents, for discussing the contents and ideas with me and helping come up with the initial topic; my friends, for their help proofreading the thesis; and finally myself, for staying motivated and interested in the topic and writing to the highest of my abilities.

Tampere, 26th April 2022

Roope Mäkelä

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

Amazon ML	Amazon Machine Learning
API	Application Programming Interface
AWS	Amazon Web Services
Cloud MLE	Google Cloud Machine Learning Engine
CPU	Central Processing Unit
GCE	Google Compute Engine
GCP	Google Cloud Platform
GCS	Google Cloud Storage
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
IaaS	Infrastructure as a Service
ML	Machine Learning
NoSQL	Not Only Structured Query Language
PaaS	Platform as a Service
S3	Amazon Simple Storage Service
SaaS	Software as a Service
SQL	Structured Query Language

1. INTRODUCTION

More and more computing is done in the cloud. Individuals and companies leverage resources from remote servers located in data centers around the world to handle data and run applications. Using this network of servers is known as cloud computing [1]. This method is beneficial due to the largely cheaper prices, as opposed to buying or constructing the required physical technological infrastructure. Using the cloud further allows for high scalability, meaning resources can be increased or reduced as required by the amount of data being processed [2]. This growing demand has prompted technology companies to become cloud service providers, offering their existing infrastructure as a service to users.

Another growing domain today is machine learning (ML). Progress is constantly made in ML and the amounts of data being processed are extremely large. As a result, many companies use cloud computing to train and deploy ML models, taking advantage of the pre-existing and relatively cheap cloud infrastructure provided by cloud service providers. With the continuous advances and the goal to make current ML technologies accessible to all, many cloud service providers have implemented a subsection for ML within their services in the past decade. This includes the possibility to share and download third-party models, as well as leverage the latest ML models and technology from the cloud service providers themselves.

From the global cloud service providers, four have risen to be the most dominant in the market. Amazon: Amazon Web Services (AWS); Google: Google Cloud Platform (GCP); Microsoft: Microsoft Azure; and IBM/Watson. Based on the quarterly earnings in 2018, GCP was the fastest growing of the four and AWS held the largest market share [3]. Today GCP and AWS are considered the most popular amongst developers and will therefore be the focus of the comparison conducted in this thesis.

So far both cloud computing and cloud services have been mentioned. Generally these terms are similar and sometimes interchangeable, however for this thesis a differentiation is made. Cloud computing refers to processing and manipulating data in the cloud, consisting mainly of the required infrastructural resources, such as the processing power of the CPUs being leveraged in the virtual machines. Cloud services on the other hand refer to the different computing engines, data storage options and deployment related services that cloud providers offer the user as products. Cloud services will be the main focus of the comparison in this thesis. Across the providers, the available virtual machines and hardware options supporting them are so similar, especially at the scale of the analysis in this thesis, that they will not be discussed. On the other hand, the individual service products the cloud service providers offer have differences in usage and capabilities, prompting insightful comparisons.

This thesis is structured as follows. Chapter 2 gives an overview of the benefits of cloud computing for ML and an introduction into cloud services. Brief overviews of GCP and AWS are then presented. Chapter 3 presents a theoretical end-to-end implementation of an ML project using GCP. This exploration gives an understanding of the different resources and services that are important in the ML implementation process. This follows directly into chapter 4: a comparison of AWS and GCP. The main highlights from chapter 3 are utilized to illustrate differences and similarities between the providers. Chapter 5 finishes the thesis with a summary of the presented information and a discussion into the choice between the two cloud service providers. Overall, this thesis acts as a guide for implementing an end-to-end ML project leveraging cloud service providers, while comparing two of the most dominant providers in the current market.

2. CLOUD SERVICES AND MACHINE LEARNING

The popularity of cloud computing is constantly increasing. Companies and individuals prefer to work in the cloud, due to the many benefits it offers. In the past two decades, market-dominating technology companies have seized this opportunity to provide cloud computing and cloud services to consumers. Using their existing internal technological infrastructure, they are able to provide users access to their own computing power and latest technological knowledge through the internet. This chapter provides an overview of cloud computing, its advantages and introduces two of the market-dominant cloud service providers; Amazon and Google.

2.1 Cloud Computing

Cloud computing in itself refers to the sharing of computing resources through network access. These include storage, servers, services and applications [4]. The main characteristics are that cloud computing is an on-demand self-service, meaning that minimal interaction between the provider and consumer is required. It is highly elastic, allowing resources to be scaled depending on consumer demand. Cloud computing provides broad network access, where capabilities can be accessed through common devices, such as laptops. Finally, it is a measured service, which automatically measures and optimizes resource usage, improving transparency between the consumer and the provider [4].

Four deployment models are involved with cloud computing: public cloud, private cloud, community cloud and hybrid cloud. Community cloud provides the cloud infrastructure for a specific group of consumers, while private cloud exists for a single organization that may consist of multiple individual consumers. In both cases it can be operated by a third party or a consumer organization and can exist on or off premises. Public cloud is for the general public and operated by an organization. The organization owns and manages the cloud infrastructure on its premises and may charge the consumers individually for its use [4]. Dominant public cloud providers today are Amazon, Google and Microsoft. Public cloud is the deployment model in focus for this thesis. The final model, hybrid cloud, is any combination of the aforementioned three models [4].

Cloud computing has three distinct service models. Software as a Service (SaaS) provides the consumer with applications that can be run through a programming interface or a thin client interface, for example a web browser. Infrastructure as a Service (IaaS) allows the consumer to use fundamental computing resources, such as storage, processing and networks, to run software. In this case the consumer does not control cloud infrastructure, but can control deployed applications, storage and possibly network components. Platform as a Service (PaaS) gives consumers access to deploy self or third party created applications on the cloud infrastructure in programming languages recognized by the provider. The consumer has no control on the underlying infrastructure, but can possibly configure the environment where applications are deployed [4]. The model provided for ML by cloud service providers generally falls into IaaS.

2.1.1 Advantages of Cloud Computing

Computing is being transferred into the cloud due to the various benefits it offers in contrast to local implementation. The two most prominent benefits are reduced costs and scalability. Acquiring and managing the hardware and systems required for high demand processing is expensive. By using cloud computing, a company can remove wages needed for the expert staff operating the systems, they can have system updates be included in their cloud computing contract, and they can eliminate time delays relating to updates or unexpected system problems [5]. In addition, most cloud providers charge the consumer for the exact resources that have been used, removing possible costs for unused resources.

With the large amounts of resources providers have available, they are able to immediately scale in response to the consumer's demand. The resources are optimized and allocated in exact amounts, so nothing is wasted and the consumer gets exactly what they pay for [6]. ML deals with very large amounts of data and the more complex a model is, the more processing power it requires. Therefore the available storage and computing resources of cloud service providers are particularly beneficial for consumers in the realm of ML.

There are multiple other advantages with cloud computing, some of which are mentioned next. Pre-built tools and services are available for consumer use. Cloud service providers have high-end security features such as encryption. Cloud services are accessible from almost any device with an internet connection. Data security means that hardware failures do not result in data loss. Teams working on the same project can work from any location globally. The providers constantly update their services and resources, ensuring the latest technological advances to consumers [6]. All of these benefits drive companies and individuals towards cloud service providers and the two providers that have risen to be most dominant in recent years are Google and Amazon, which are introduced next.

2.2 Google Cloud Platform

Google first released its App Engine as a cloud computing service, allowing developers to run web applications on Google infrastructure in 2008. It had the aim of providing developers an easy start with their web app and making scalability easier when the app's demand increased. App Engine was in preview mode with a limited number of developers being able to use it until 2011, when Google made it a fully supported Google service and the first product in the Google Cloud Platform (GCP) [7].

Today GCP includes many more products which can be used in combination with each other. A recent addition in 2017 was a subdivision of GCP, Google AI, which includes all products and services related to artificial intelligence and ML. Google AI was launched in order to democratise the advantages of research into ML; including Google's own advances. While Google's TensorFlow, an open-source platform for ML, has previously made ML more accessible, the aim of Google AI is to make global research more accessible [8]. For the purpose of this thesis, this subdivision will not be directly reference since a mix of ML, storage and data manipulation services will be discussed, which all fall under the larger GCP umbrella.

The central ML service GCP provides is the Google Cloud Machine Learning Engine (Cloud MLE). It is Google's managed infrastructure for training and deploying large-scale ML models. Cloud MLE can train models built with TensorFlow, Skicit-learn, Keros and XGBoost platforms and also provides a range of built-in models that can be fit to data [9]. Cloud MLE can be trained using batches (a set number of data) or is scalable to online data sets, where the data is constantly being updated and training over the entire data set is not feasible. Essential features in Cloud MLE are that it is able to perform distribution training and automatically hyper-optimize parameters while training, minimising the need for iterative experiments [10]. Cloud MLE will be discussed in depth in Chapters 3 and 4.

2.3 Amazon Web Services

Amazon established its first cloud service in 2002, and its first ML service, Amazon Machine Learning (Amazon ML) in 2015 [11]. Due to this first-mover advantage into the cloud service provider market, AWS holds the dominant market share in cloud infrastructure at 32% as of quarter 3 in 2020. For context, the next two cloud providers, Microsoft Azure and GCP, were at 19% and 7% respectively at the same time period according to the Synergy Research Group [12].

In 2016 Amazon stopped updating and accepting new users to Amazon ML, while existing users still had access to the service [11]. In 2017, Amazon launched a new ML service, Amazon SageMaker, with the intent of removing some of the heavy lifting typically associated with ML and making it easier for developers of all skill levels to create and deploy

ML models [13]. SageMaker is a managed, end-to-end ML service, which also provides the option to directly deploy trained models into a hosted environment [14].

Amazon SageMaker continues to get updates today, with the most recent having been in December 2021 with the addition of seven new features [14]. Although it would be interesting to compare Amazon ML with Amazon SageMaker and see how SageMaker makes ML more accessible, that will not be included in the thesis. Instead the focus will be on Amazon SageMaker, which new users can still access today, and how this compares to Google Cloud MLE for an end-to-end ML project. The next chapter provides a simple theoretical example of implementing a full ML project using a a cloud service provider and their available infrastructure.

3. END-TO-END MACHINE LEARNING IMPLEMENTATION ON GCP

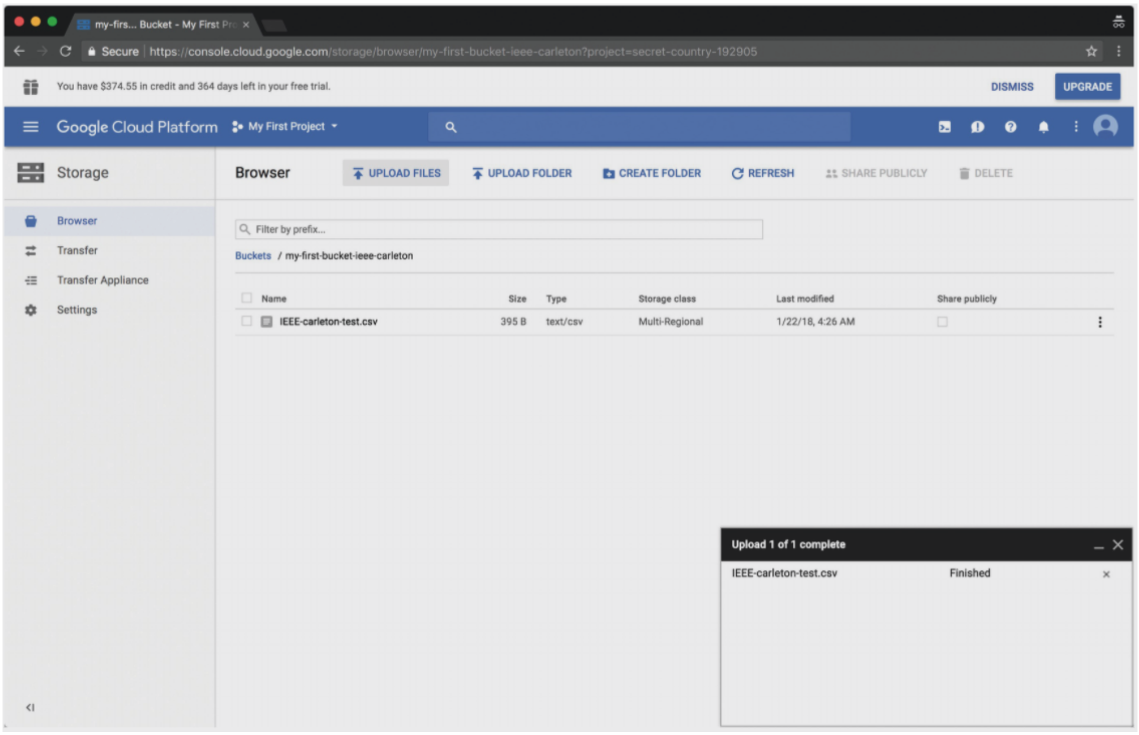
Large cloud service providers, such as GCP and AWS, provide the needed services to complete a full end-to-end ML project. The user starts from a data set and finishes with a trained and deployed ML model that can be used for predictions on new data. This chapter explores these services and how they are combined to achieve the desired final product. While cloud service providers have similar services and procedures, they are not identical. For the purposes of this chapter GCP will be the platform of focus. Then in Chapter 4, AWS will be compared to GCP utilizing the main points arising from this exploration.

As is the case with most cloud service providers, there are several methods to implement ML on GCP. The most simple method is through using Google Cloud Storage (GCS) for storing the training data set on the cloud, using Google Dataprep to transform the data into the correct format and finally, Cloud MLE to create, train and deploy the model [10]. Another option is using Google Compute Engine (GCE) to create a virtual machine to train the model. The downside to GCE is that it is an engine used for any kind of computing, while Cloud MLE is designed specifically for ML [15]. The final option is to use Google Kubeflow to complete the whole process [10]. Due to the scope of the thesis and in the interest of simplicity, the first proposed method is explored, but the other methods will be touched on in Chapter 4.

The end-to-end implementation is divided into two main sections: data pre-processing, and model training and deployment. Data pre-processing refers to the storage of data on the platform and transforming it to be compatible with the ML service. Data training and deployment contains the model design, model training, model hyper-parameter optimization and using the trained model to make predictions. Model training and deployment are combined into one section, since on GCP it is all done through the Cloud MLE [10].

3.1 Data Pre-Processing

Storing data on multiple servers across machines on a network is defined as cloud storage [15]. GCP provides Google Cloud Storage (GCS) for this purpose, where users can save, access and manipulate data on Google infrastructure. GCS can be interacted with through a programming interface using HTTP requests or through the main GCP graphical user interface (GUI) [15]. GCS uses buckets as its organizational structure, where a bucket is similar to a file directory on a computer. Buckets can be arranged hierarchically and data files can be uploaded to buckets as objects. Figure 3.1 illustrates the GUI of GCS, where a bucket has been created with a file uploaded into it.



https://console.cloud.google.com/storage/browser/my-first-bucket-ieee-carleton?project=secret-country-192905

You have $374.55 in credit and 364 days left in your free trial.

Name	Size	Type	Storage class	Last modified	Share publicly
IEEE-carleton-test.csv	395 B	text/csv	Multi-Regional	1/22/18, 4:26 AM	☐

Upload 1 of 1 complete

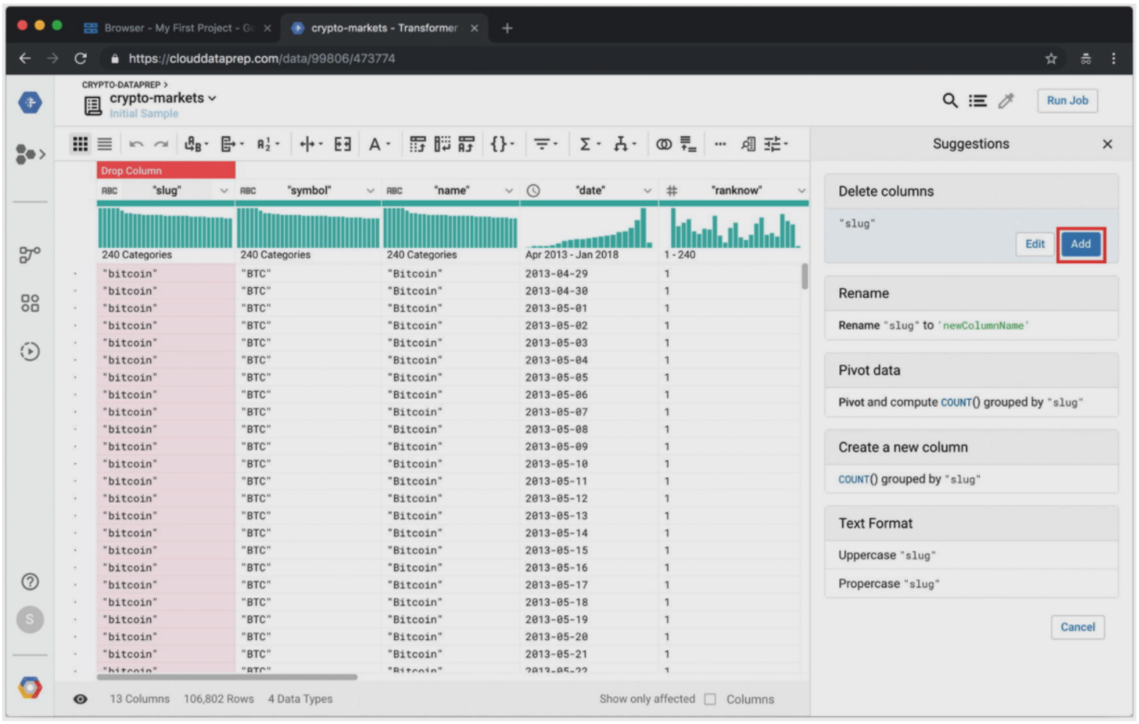
IEEE-carleton-test.csv Finished

Figure 3.1. *An example of a GCS bucket containing one uploaded data file [10].*

There are parameters that need to be chosen when creating a bucket: the type of data, where the buckets will be accessed from geographically, and how quickly the data can be manipulated, which also impacts pricing [10]. Once the data that will be used in training the ML model is successfully saved into a bucket, Google Dataprep is used to prepare the data.

In real world cases, data sets are often unorganized and contain unnecessary information. For example, data may be collected from a ten question survey, but only the relationship between the answers to two of the questions is required for the model. Therefore the original data set cannot directly be used for training the ML model. Google Dataprep serves as a method to organize the data and remove unnecessary variables or information. Dataprep is particularly appealing since it provides a GUI for organizing the data, whereas generally the same organization needs to be completed through programming. Dataprep works for unstructured and structured data sets and leverages Google DataFlow's capabilities of distributed processing [10].

Dataprep uses objects called flows to organize the data sets and the operations that are conducted on them. Once a flow is created, a bucket from GCS can be imported into it. A flow also contains Dataprep recipes that can be created. These recipes contain the transformations that will be done to the data sets. A recipe is first created and then by selecting 'edit recipe' the transformation grid is opened, where the specific data transformations are defined [10]. Figure 3.2 shows the transformation grid of a new recipe on an example data set.


Browser - My First Project - Gc
crypto-markets - Transformer
https://clouddataprep.com/data/99806/473774
CRYPTO-DATAPREP >
crypto-markets
Initial Sample
Run Job
Drop Column
Suggestions
Edit
Add
Cancel
13 Columns 106,802 Rows 4 Data Types
Show only affected Columns

"slug"	"symbol"	"name"	"date"	"ranknow"
240 Categories	240 Categories	240 Categories	Apr 2013 - Jan 2018	1 - 240
"bitcoin"	"BTC"	"Bitcoin"	2013-04-29	1
"bitcoin"	"BTC"	"Bitcoin"	2013-04-30	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-01	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-02	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-03	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-04	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-05	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-06	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-07	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-08	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-09	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-10	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-11	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-12	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-13	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-14	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-15	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-16	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-17	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-18	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-19	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-20	1
"bitcoin"	"BTC"	"Bitcoin"	2013-05-21	1

Delete columns
"slug"

Rename
Rename "slug" to 'newColumnName'

Pivot data
Pivot and compute COUNT() grouped by "slug"

Create a new column
COUNT() grouped by "slug"

Text Format
Uppercase "slug"
Propercase "slug"

Figure 3.2. *An example of the transformation grid in Dataprep [10].*

In Figure 3.2 the leftmost column titled "slug" is about to be deleted from the data set. Other possible transformations include deleting rows and filtering rows. When rows are filtered, the user can either keep or remove rows with a certain value in a specific column. Once all the transformations have been added to the recipe, clicking 'run job' in the top right corner will execute them. Once the job has run, the resulting data set can be exported back to a GCS bucket [10]. After the transformations are complete, the data set is ready to be used to train the ML model.

3.2 Model Training and Deployment

Cloud MLE is mainly interacted with using GCP's command line terminal to train and deploy ML models. It can train models created with TensorFlow, Keras, Scikit-learn and XGBoost [10]. Following the scope of this exploration, the details of the model building with one of these platforms will not be discussed. Rather, the model is considered as a black box that interacts with Cloud MLE. The focus will be the interaction between the model and Cloud MLE through the terminal.

The ML process on Cloud MLE contains three main working components: GCS, Cloud MLE and the program for the ML model. Previously, the data set was stored on GCS and it will automatically be accompanied by the trained model, saved by Cloud MLE into GCS. Cloud MLE uses the model program and data set to train the model, as well as creates a prediction service for new data [10]. The relationship between these components and a high level view of the ML process is illustrated in Figure 3.3, starting from the data set.

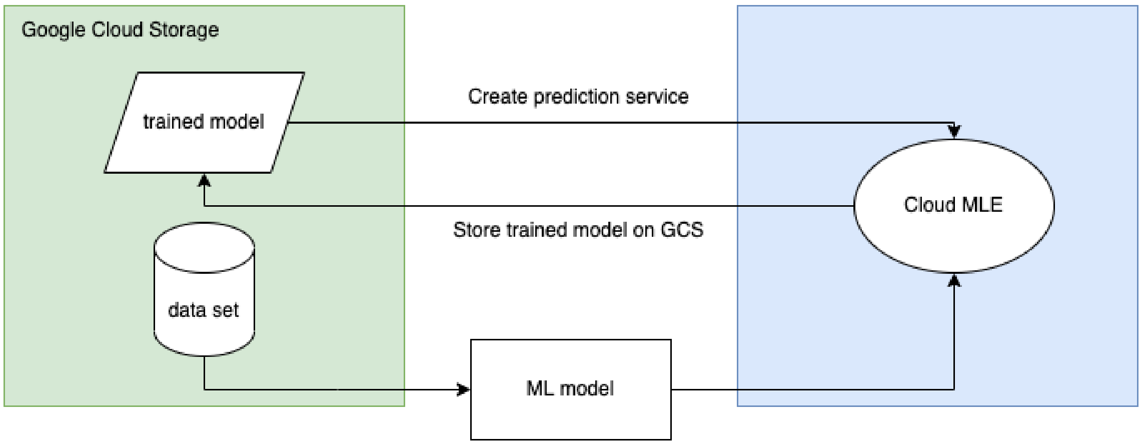


Figure 3.3. *Simplified ML process on Cloud MLE.*

All of the training is completed through GCP's terminal. The specific terminal commands and necessary file contents for training are given in Appendix A for reference. Here a simple guide through the steps is presented. The ML model must be stored in a python package with suffix '.py' to be compatible with Cloud MLE [10]. Most open-source ML platforms, such as TensorFlow, are available to be worked with in the python programming language.

Training the model is conducted using shell files with suffix '.sh'. Within these files, lines of bash codes inform Cloud MLE of the data set, model and other parameters to be used in training. Depending on the bash codes the training can be run as a single instance, distributed training or distributed training with hyper-parameter tuning. For hyper-parameter tuning, a separate file containing bash codes is needed for the parameters. Once the training job is run, its results can be viewed in the Cloud MLE model instance and is stored in GCS [10].

The last step is to deploy the trained model by creating a prediction instance. This is done through the terminal using a shell file. The file contains the folder location of the trained model in GCS. Once the prediction instance is created, the details can be seen in the model version on the dashboard. A separate shell file is finally used to make batch predictions on the test data or new data and will be shown visually in the results section. Other metrics such as error and data loss are automatically provided [10].

This chapter introduced a simple process of creating an end-to-end ML model using GCP, mainly using Cloud MLE and some data pre-processing. This example was a very high level look into the process, without delving into the code behind the model or commands for interacting with the terminal to train the model. Training bash codes and shell files are included in Appendix A, but are not directly relevant to the focus of the following comparison. In the next chapter, the services and processes given by GCP and AWS will be discussed and compared in the scope of an ML project.

4. COMPARISON OF AWS AND GCP

The previous chapter introduced some of GCP's services that can be utilized in implementing an ML project. AWS provides similar services, but both providers' services are unique with different capabilities. This chapter aims to compare these service to reach a consensus on which cloud provider is the optimal choice. Of course, the choice will depend on many factors and there likely will not be one that is a better choice in every case.

Both Amazon and Google allow users to run cloud computing on the same infrastructure that the companies conduct their internal computing on. Since both are market giants and have the best technology available, the actual computing power, efficiency or resource usage will not be compared. At most developer levels these will have no perceptible difference when implementing an ML project. The focus of this comparison will instead be on the individual services offered for data storage, data processing and ML, including built-in models. The comparison follows the order of the implementation discussed in the previous chapter, beginning with data storage, and ends with a summary of the key findings.

4.1 Data Storage

Google offers three storage services pertinent to ML: GCS, Google Bigtable and Google BigQuery [16]. Although there are other storage options available, these support the data format needed for ML and provide other related benefits. GCS was previously discussed in Chapter 3. Its main advantages are that it offers high scalability, consistency, durability and availability. GCS is the central cloud storage for GCP and can be directly accessed through the ML services: Cloud MLE, Cloud AI Training and Google Kubeflow. Trained models are automatically saved into GCS [10]. GCS is the standard storage option, which will be feasible for the majority of users.

For projects requiring petabytes of data, the user might consider Google Cloud Bigtable which is Google's NoSQL database as a service [16]. NoSQL is in reference to Not Only Standard Query Language, meaning that the database is not only organized in rows and columns and accessed through SQL queries, but can also be organized by key-value pairs, documents and other models. NoSQL databases are distributed, non-relational and scalable [17]. Cloud Bigtable was built to solve the problem of having a database capable of storing petabytes of data, yet with high availability and low latency. Being NoSQL, there is no need for queries to search through tables which allows for information to be received within milliseconds, regardless of how much data is stored in the database [16].

The third storage option GCP provides is Google Cloud BigQuery. BigQuery is a data warehouse service, optimal for data analytics. BigQuery is used for structured data, accessed by SQL queries. Although it is not NoSQL, BigQuery is still very fast and scalable and removes the need for the user to administrate the operation and scaling of a distributed database [10]. BigQuery in itself is not as beneficial for ML as GCS or Cloud Bigtable when considering training through Cloud MLE, due to the required SQL queries. However, the BigQuery product offering includes built-in ML capabilities through BigQuery ML. By using simple SQL queries, the user is able to train and test built-in ML models on subsets of the data [10]. Currently BigQuery includes a large range of models, including linear regression, classification, clustering, dimensionality reduction and time series forecasting [18]. The models can be trained regardless of the amount of data [10].

A model created within BigQuery will continuously be updated based on the data within the database. This significantly reduces the alienation of ML from business personnel, as they can update the data and immediately see the results in the predictions made by the model. Data storage, model training and model deployment are all conducted within a singular, easy to use service [10].

The three different storage services GCP provides have equivalents on AWS: Amazon Simple Storage Service (S3), Amazon DynamoDB and Amazon Redshift. S3 is Amazon's central cloud storage service, which allows object storage with high availability, security, scalability and performance. A wide range of different types of data can be stored, including applications, websites, archives and data lakes. Depending on the storage tier selected, pricing is adjusted based on how frequently and quickly the data needs to be accessed [19]. Data stored in S3 can be accessed through Amazon SageMaker to train ML models [14].

Amazon's big data service, Amazon DynamoDB, provides the user with the same benefits as Google Cloud Bigtable at a basic operational level. DynamoDB is NoSQL and fully managed, where the user does not need to operate setup and configuration or hardware provisioning. DynamoDB also contains encryption at rest, meaning the user does not need to separately consider protecting data [20].

Amazon's data warehouse service, Amazon Redshift, is rather similar to Google Cloud BigQuery. Redshift is scalable into petabytes of data and is operated on through SQL queries. Amazon Redshift offers built-in ML, which is accessed through SQL queries to create and train the model, similarly to BigQuery. When an ML model is trained through

Amazon Redshift, the training data is first saved into Amazon S3. Amazon SageMaker Autopilot preprocesses the data and finds the ML model which will be most accurate for the training data. The model is then saved into the same Amazon Redshift cluster and can be deployed using SQL queries on new data [21].

The key difference between Google Cloud BigQuery and Amazon Redshift is how they provide built-in ML. BigQuery includes BigQuery ML, which conducts the necessary ML operations on the data within the same service [18]. However, in Redshift, although the ML is initiated within the service, it uses Amazon S3 and SageMaker Autopilot to store and train the data. This additional use of resources has an associated cost which will be added to the user's bill, including the storage and training of the model [21]. In addition, in BigQuery the user is able to select the preferred ML model to be used for training from the provided models listed earlier [10]. Through Redshift, SageMaker Autopilot makes the choice of model automatically based on the data set [21]. While the choice is made based on the most accurate model, the user may prefer another model and not being able to make this choice can be seen as a disadvantage.

In the end, both cloud service providers have similar storage services when it comes to ML. Most users will be satisfied with the core storage services, GCS and S3, which provide very similar features and capabilities. If very large amounts of unstructured data need to be stored and accessed quickly, then Google Cloud Bigtable and Amazon DynamoDB are equally useful. The difference between storage on the two platforms comes from the data warehouse service ML capabilities, where GCP clearly has the advantage.

4.2 ML Model Options

GCP and AWS provide similar services for ML model creation and training with Google Cloud MLE and Amazon SageMaker respectively. With both platforms, the user is able to use their own model created with an external platform to train the data set as seen in Chapter 3 for GCP [10][3]. SageMaker is, however, compatible with more third-party created models than Cloud MLE. Both are compatible with TensorFlow and Keras, while Cloud MLE is also compatible with Scikit-learn and XGBoost and SageMaker adds MXNet, Gluon, Pytorch, Caffe2, Chainer and Torch [10] [9]. If the user wishes to create their own model, the best choice will depend on what third-party platform they are most comfortable creating the model on.

The two providers give a large range of built-in ML models which can automatically be used with a data set, requiring no actual model code from the user. A summary of the different existing models in SageMaker and Cloud MLE is illustrated in Table 4.1. Models are categorized by the domain of ML and the table is compiled using information in AWS SageMaker documentation [14] and Google AI Platform documentation [22]. GCP models provided uniquely by Cloud BigQuery ML, as discussed in Chapter 4.1, are distinguished with an asterisk (*), since they are only available within the specific service [18]. An important note is that algorithms named similarly across the platforms, or on the same row in the table, are not necessarily equivalent to each other.

Table 4.1. *List of built-in ML models available on AWS and GCP.*

Model Domain	*AWS Models*	*GCP Models*
Supervised Learning	Linear Learner XGBoost Factorization Machines K-Nearest Neighbours (k-NN) DeepAR Forecasting Object2Vec	Linear Learner XGBoost Wide and Deep TabNet Matrix Factorization* Time Series Forecasting*
Image Processing	Image Classification Object Detection Semantic Segmentation	Image Classification Object Detection
Unsupervised Learning	Principal Component Analysis Random Cut Forest IP Insights K-means Latent Dirichlet Allocation Neural Topic Model	Principle Compenents Analysis* K-means*
Textual Analysis	BlazingText Sequence-Sequence Algorithm	

The type of problem the user requires ML for will impact their choice of platform. Both platforms provide extensive models for supervised learning and image processing within their primary ML engines, but GCP completely lacks in unsupervised learning and textual analysis. Granted, a single model can be applied to a wide range of real-world use-cases and Google has additional models within BigQuery ML. For the majority of tabular ML problems, where the outcome is predicted from a set of values, both platforms are equally feasible. However, Amazon claims that their built-in models are 10 times more effective than any other platforms' equivalents, due to their highly effective automatic optimization [3]. For any more specific or more complicated problems, Amazon is more likely to have a fitting model which does not need to be adjusted to fit the data.

In summary, both platforms are equally easy to use when using an externally created model and have equally sufficient processing power to train and deploy these. Amazon SageMaker is compatible with more third-party ML platforms for model creation, even though the most popular ones are also supported on Google Cloud MLE. A deciding factor is if the user prefers a built-in model. In this case, AWS provides a larger range of optimized built-in models. If GCP is chosen, this then prompts the further choice of whether Cloud MLE or BigQuery ML is used, depending on where the preferred model is available.

4.3 Case-Specific ML Services

While both AWS and GCP provide built-in ML models in their ML engines, they also provide case-specific ML services as individual products. For the purpose of this thesis, a case-specific ML service refers to a service with a pre-trained ML model, where the user can upload their data in the proper format and the service will automatically tune the model to create the desired output. For example, AWS offers Amazon Polly, which is an ML service that converts text to speech. The user uploads the data in text form and the service automatically converts this to speech [23].

By using these ML services, the user does not need to go into model creation or training, but can use these pre-trained models as a service to implement ML on their own use-cases. Using these services essentially completely removes the threshold between the user and ML, where the user does not need prior experience with coding or ML. The ML services offered by AWS and GCP are listed in Table 4.2 and are limited to solely the services that require no ML experience or separate programming on the part of the user. The services listed conduct the training and deployment automatically, using hyper-parameter optimization. The table is compiled using AWS documentation [23] and GCP documentation [24] along with studies by Bisong [10] and Wickham [3]. The table lists the purpose of the services and the corresponding services offered by each platform.

Table 4.2. *Automatic ML services offered by AWS and GCP.*

Purpose	*AWS Service*	*GCP Service*
Natural Language Processing	Comprehend	Natural Language
Conversational Interfaces	Lex	Dialogflow CX
Text-to-Speech	Polly	Text-to-Speech
Speech-to-Text	Transcribe	Speech-to-Text
Language Translation	Translate	Translate
Image/video analysis	Rekognition	Video Intelligence
Image Classification		Vision
Extract Values from Documents		Form Parser

While both providers offer a multitude of other ML services, the ones listed in the above table account for the services where the user is not required to create or train the models themselves. While both providers have a large range of equal services, GCP includes two that AWS does not; image classification and document extraction. On AWS these are services requiring some ML skills from the user in training and tuning the models.

To conclude, both AWS and GCP provide extensive ML services that do a variety of things automatically, with the user not needing to have experience with ML. For users that require ML based data manipulation, but have no ML experience, both services are highly functional in providing the specific service needed. However, since GCP has slightly more use-case specific services, it is at an advantage compared to AWS.

4.4 Model Implementation

This section discusses the different methods of implementing an end-to-end ML project on AWS and GCP. Both cloud providers offer multiple services and combinations of services to implement a complete project. With the scope of this thesis, the different methods are not examined deeply, but are mentioned and described at a high level view in this section. The different methods and their individual capabilities are used for the comparison. Some of the methods have already been discussed in Chapters 3 and 4, but additional methods will be included in this section.

GCP has four methods for the user to choose from. The most common option is using GCS and Cloud MLE to create and train the model as was explored in Chapter 3. This option is the most versatile, since Cloud MLE allows the user to use an externally created ML model [10] or choose from the selection of built-in ML models [22]. The Amazon equivalent of this method, using Amazon SageMaker, provides the same versatility. Either an externally created model or built-in model can be used. SageMaker divides the built-in model usage into a sub-service, SageMaker Autopilot. Using SageMaker Autopilot means the user lets the service automatically train and deploy the model after initialization. As was discussed in Chapter 4.1, SageMaker Autopilot does not allow the user to choose the specific ML model. Rather, it chooses the model which will be most accurate for the training data [14].

The next method is conducting ML through the data warehouse services, Google Cloud BigQuery and Amazon Redshift. Since these were discussed previously in Chapter 4.1, they are not analyzed again here. BigQuery has the advantage to Redshift since the user can select the specific built-in ML model to be used, while in Redshift this selection is automatic through SageMaker Autopilot. Also, this method with Redshift charges the user for the extra required storage and processing resources [14].

For AWS, the methods for implementing an ML project are limited to the three already discussed: SageMaker, SageMaker Autopilot and Redshift. However, GCP boasts one more additional method of implementing ML through Google Kubeflow. This method has not yet been discussed and a short overview will now be presented.

Kubeflow leverages Kubernetes to build and deploy containerized ML workflows called pipelines [10]. Kubernetes is an open-source platform with the purpose of managing containerized workloads and services [25]. However, when using Kubeflow the user does not need to directly work with or be familiar with Kubernetes. Training a large amount of data, evaluating the model and retraining the data to update the model is a very iterative process, using lots of resources on behalf of the user. Kubeflow reduces this iterative process by allowing the user to place ML components into the Kubeflow pipeline. Components such as data transfer, training, deployment and monitoring can be placed into containers in a Kubeflow cluster [10]. An example ML pipeline on Kubeflow is illustrated in Figure 4.1. Each box in the pipeline represents a component, which contains a program file.

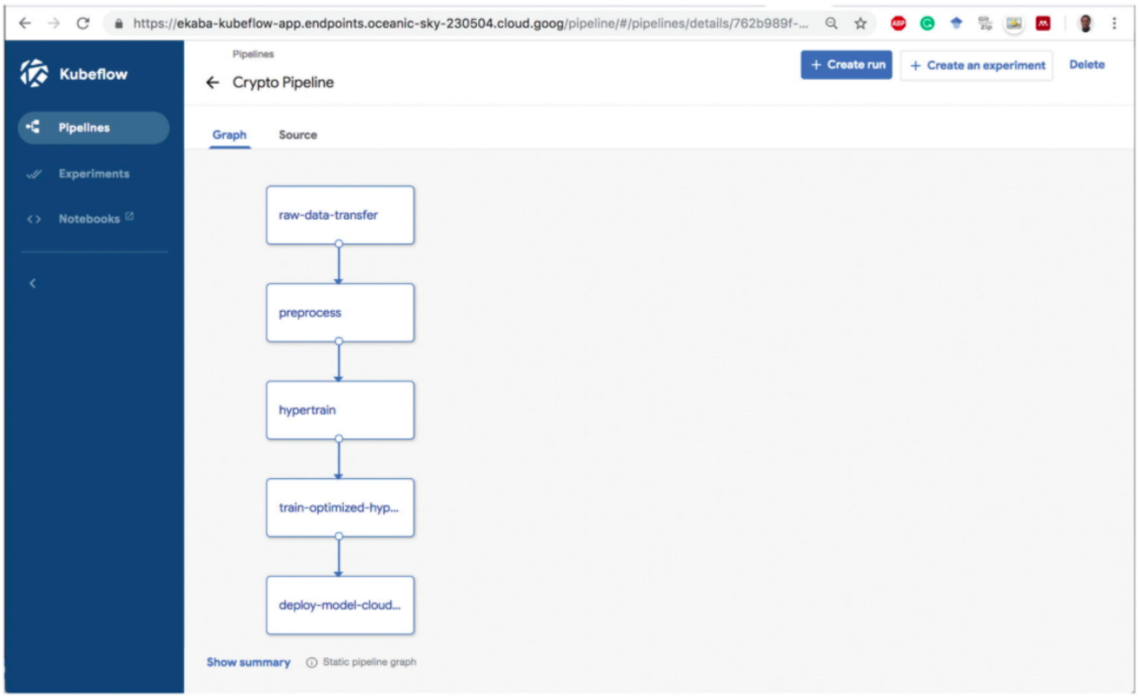


Figure 4.1. *An example of an ML pipeline in KubeFlow [10].*

Each component in the pipeline processes or manipulates the data in a way set by the user. A certain input and output is expected from each component. Once the pipeline is complete, a Kubeflow Experiment is created to execute the pipeline [10]. In essence, the components can manipulate the data however the user requires and are not limited to ML processes. The main advantage of Kubeflow is its distributed training capabilities and the fact that the whole end-to-end implementation process can first be designed before executing it. This method requires more ML knowledge from the user, since each component requires a program code created by the user to operate in the desired way [10].

To conclude, both GCP and AWS provide useful options for implementing ML. Using the core ML engines, Cloud MLE and SageMaker, is similar from the point of view of the user on both platforms and offer similar capabilities. So is using the data warehouse services, minus the discussed disadvantage of SageMaker Autopilot. The main difference between implementation methods comes from Google KubeFlow , where the user can leverage Kubernetes and implement the whole ML process as a pipeline using distributed training. While this is likely not a deciding factor for the common user since it requires vast knowledge of programming and ML, it is an important option to keep in mind.

4.5 Summary

A variety of factors have been considered in the choice between AWS and GCP in implementing an ML project. For the most part, both providers are equally capable of being used for ML. This comparison has focused on the provided services and their capabilities, without diving into the detailed performance of the discussed services. Both providers have the most up-to-date infrastructure and performance capabilities, which will likely not make a difference to the user.

Table 4.3 contains a summary of the comparison points discussed in this chapter. Comparison points are divided into categories in the leftmost column, with the specific point in the column to its right. AWS and GCP are given a score on a two point scale: '+' or '++'. A '+' represents standard functionality and '++' means the functionality is very good and has an advantage compared to the other platform. If a point were sub-average, it would be left blank.

Table 4.3. *Summary of the comparison between AWS and GCP.*

Category	*Comparison Point*	*AWS*	*GCP*
Storage	Central Storage (S3/GCS)	+	+
	Big Data Storage (DynamoDB/Bigtable)	+	+
	Data Warehouse (Redshift/BigQuery)	+	++
Model Options	User's Own External Model	++	+
	Built-in Model	++	++
Case-specific Services	Automatic ML Services	+	++
Model Implementation	Implementation with ML Engine	+	+
	Other Methods	+	++

From the table, no comparison point is awarded a blank result. This shows neither platform has factors that are sub-average and for most user ML cases, either platform is sufficient to conduct the entire ML implementation process. Still, small differences arise that, depending on the user's project, may be a deciding factor. Within the storage category, the ML capabilities of GCP's data warehouse service hold an advantage compared to AWS's equivalent. In the model options category, while the implementation of an external model is equally easy on both platforms, AWS is compatible with more third-party platforms, being an advantage to the user. When choosing a built-in ML model, AWS provides a significantly larger amount of models with more specific purposes, however, the user cannot choose the specific model with Amazon SageMaker Autopilot. On Google Cloud MLE the user can freely choose the model from the provided options. Therefore both AWS and GCP have been awarded '++' for built-in models, since they both have an advantage in the same category. The automatic ML services provided by GCP slightly exceed those of AWS in number and can be better optimized. Considering the entire implementation process, GCP offers a larger variety of methods and the methods are more user-friendly than the AWS equivalents.

In the end, both providers have small nuances that can sway the user's decision into either direction. It is not possible to make a concrete universal decision of which cloud service provider is better for implementing ML, since both have advantages to the other in different areas. The decision will come down to what the specific ML project the user is implementing requires and which services on either platform will provide the best outcome for the specific project.

5. CONCLUSION

This thesis explored how cloud service providers can be leveraged when implementing an ML project. An exploration using GCP's Cloud MLE service to create and train an ML model was conducted. Finally, AWS and GCP were compared with respect to ML projects to explore their offered services and capabilities. The choice of implementing an end-to-end ML project on AWS or GCP is a matter of the specific requirements of the project. Both cloud service providers offer good capabilities in the categories of data storage, external and built-in ML models, and different methods of implementing the entire project.

While the differences found in the comparison have been previously summarized, the main factors are briefly mentioned here. The user should consider whether they wish to use an external or built-in ML model. While both platforms are equally capable of training an externally created model, AWS's built-in ML model will be automatically chosen by Amazon SageMaker Autopilot, while on Google Cloud MLE the user is able to choose the preferred model. Even though AWS has a larger range of built-in models and will choose the one providing highest accuracy, sometimes the user may wish to use a specific model for their data. GCP provides better ML capabilities within their data warehouse service Cloud BigQuery, compared to Amazon Redshift. GCP also offers more automatic ML services and Google Kubeflow, an extra method for an end-to-end ML implementation.

All factors considered, both platforms are very capable of implementing any scale of ML project. Unless one of the mentioned factors is specific to the user's ML project and warrants a clear choice, both platforms are equally capable in conducting the entire process. In the end, the decision may simply be the user's preference of Amazon or Google in general or if they are already an existing user of either platform.

Some factors were not present in the comparison, such as pricing of the services and how effective the infrastructure behind either cloud service provider is. The scope of this thesis focused on the provided services and their capabilities concerning ML. Either way, the high competition within the cloud provider market has driven down prices and pricing across all cloud service providers are largely the same [3]. Since pricing is based on the actual resource usage, for an ML project the total monetary expenditure will not have noticeable differences across platforms. The same is true for infrastructure and processing capabilities. With the largest providers already being established in the technology industry, they are all able to provide equally high standards of resources to the user [3].

This thesis was able to answer its research questions and provide an overview of ML implementation using cloud service providers. Deeper investigation into the hyper-parameter optimization and code behind the ML models did not fit the scope of this thesis, but can be used in further comparison research. Future research could additionally extend to other cloud service providers, such as Microsoft and IBM [3]. Conducting an experiment by implementing an identical ML project on AWS and GCP, using data from a real-world use-case to find differences in prediction accuracy, can be beneficial as well. It would also be interesting to explore the threshold between management and ML personnel, to see which provider can be interacted with the easiest with no prior ML experience.

It should be noted that cloud computing and ML are continuously being developed, with all cloud service providers creating new services and updating existing services frequently. The exploration and comparisons made in this thesis will likely become outdated over time, so future studies should revisit these same ideas to keep them up-to-date.

REFERENCES

[1] *Cloud Computing*. Lexico. URL: https://www.lexico.com/definition/cloud_computing (visited on 02/20/2022).

[2] Bryce, J. *Cloud Computing Scalability: A Quick Guide, and Types*. DataFloq. June 21, 2021. URL: https://datafloq.com/read/cloud-computing-scalability-a-quick-guide-types/ (visited on 02/20/2022).

[3] Wickham, M. *Practical Java Machine Learning: Projects with Google Cloud Platform and Amazon Web Services*. Apress, 2018.

[4] Mell, P. and Grance, T. The NIST Definition of Cloud Computing. *National Institute of Standards and Technology* (2011).

[5] *Benefits of Cloud Computing*. Business Queensland. July 21, 2017. URL: https://www.business.qld.gov.au/running-business/it/cloud-computing/benefits (visited on 03/20/2022).

[6] IBMCloudEducation. *Benefits of Cloud Computing*. IBM. Oct. 10, 2018. URL: https://www.ibm.com/cloud/learn/benefits-of-cloud-computing (visited on 03/20/2022).

[7] *The History of Google Cloud Platform*. A Cloud Guru. Dec. 12, 2018. URL: https://acloudguru.com/blog/engineering/history-google-cloud-platform (visited on 03/20/2022).

[8] Mannes, J. *Google.ai aims to make state of the art AI advances accessible to everyone*. TechCrunch. May 17, 2017. URL: https://techcrunch.com/2017/05/17/google-ai-aims-to-make-state-of-the-art-ai-advances-accessible-to-everyone/ (visited on 01/15/2022).

[9] *Comparing Machine Learning as a Service: Amazon, Microsoft Azure, Google Cloud AI, IBM Watson*. Altexsoft. Apr. 25, 2021. URL: https://www.altexsoft.com/blog/datascience/comparing-machine-learning-as-a-service-amazon-microsoft-azure-google-cloud-ai-ibm-watson/ (visited on 03/28/2022).

[10] Bisong, E. *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*. Apress, 2019.

[11] TheAWSAuthors. *Amazon Machine Learning Developer Guide*. AWS. Aug. 2, 2016. URL: https://docs.aws.amazon.com/machine-learning/latest/dg/what-is-amazon-machine-learning.html (visited on 02/02/2022).

[12] Panettieri, J. *Cloud Market Share 2022: Amazon AWS, Microsoft Azure, Google, IBM*. Channele2e. Nov. 3, 2020. URL: https://www.channele2e.com/channel-

partners/csps/cloud-market-share-2020-amazon-aws-microsoft-azure-google-ibm/ (visited on 03/27/2022).

[13] Miller, R. *Amazon releases SageMaker to make it easier to build and deploy machine learning models*. TechCrunch. Nov. 29, 2017. URL: https://techcrunch.com/2017/11/29/aws-releases-sagemaker-to-make-it-easier-to-build-and-deploy-machine-learning-models/ (visited on 03/27/2022).

[14] TheAWSAuthors. *Amazon SageMaker Developer Guide*. AWS. Dec. 1, 2021. URL: https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html (visited on 03/27/2022).

[15] Ciaburro, G., Ayyadevara, V. and Perrier, A. *Hands-On Machine Learning on Google Cloud Platform*. Packt Publishing, 2018.

[16] Geewax, J. *Google Cloud Platform in Action*. Manning Publications, 2018.

[17] Mullins, C., Vaughan, J. and Beal, B. *NoSQL (Not Only SQL Database)*. TechTarget. Apr. 2021. URL: https://www.techtarget.com/searchdatamanagement/definition/NoSQL-Not-Only-SQL (visited on 03/31/2022).

[18] *Google Cloud BigQuery Documentation*. GCP. Mar. 31, 2021. URL: https://cloud.google.com/bigquery/docs (visited on 04/01/2022).

[19] TheAWSAuthors. *Amazon Simple Storage Service User Guide*. AWS. Feb. 22, 2022. URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html (visited on 04/02/2022).

[20] TheAWSAuthors. *Amazon DynamoDB Developer Guide*. AWS. Feb. 15, 2021. URL: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html (visited on 04/01/2022).

[21] TheAWSAuthors. *Amazon Redshift Developer Guide*. AWS. Aug. 27, 2021. URL: https://docs.aws.amazon.com/redshift/latest/dg/welcome.html (visited on 04/02/2022).

[22] *Google AI Platform Training Documentation*. Google Cloud. Mar. 23, 2022. URL: https://cloud.google.com/ai-platform/training/docs (visited on 03/29/2022).

[23] TheAWSAuthors. *Overview of Amazon Web Services: Machine Learning*. AWS. Feb. 17, 2022. URL: https://docs.aws.amazon.com/whitepapers/latest/aws-overview/machine-learning.html (visited on 04/02/2022).

[24] *Google Cloud AI and Machine Learning Products*. GCP. URL: https://cloud.google.com/products/ai (visited on 04/02/2022).

[25] TheKubernetesAuthors. *What is Kubernetes*. Kubernetes. July 23, 2021. URL: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/ (visited on 04/03/2022).

APPENDIX A: MODEL TRAINING ON GCP

Training the ML model in Google Cloud MLE requires interacting with the service using the command terminal. Shell files containing bash commands are used to define the locations for the training data, the actual model and the hyper-parameter file. This appendix describes the relevant terminal commands and presents example shell files.

The first required shell file is presented in Program A.1. The file is an example and the different parameters can be changed depending on the specific project. For this example, the file is named 'hyper-tune.sh'.

Program A.1. *hyper-tune.sh file adapted from Bisong [10].*

```
export SCALE_TIER=STANDARD_1 # BASIC | BASIC_GPU |STANDARD_1 |
PREMIUM_1 | BASIC_TPU
DATE=‘date '+%Y%m%d%H%M%S'‘
export JOB_NAME=iris_$DATE
export HPTUNING_CONFIG=hptuning_config.yaml
export GCS_JOB_DIR=gs://iris-dataset/jobs/$JOB_NAME
export TRAIN_FILE=gs://iris-dataset/train_data.csv
export EVAL_FILE=gs://iris-dataset/test_data.csv
echo $GCS_JOB_DIR
gcloud ai-platform jobs submit training $JOB_NAME \
                                    --stream-logs \
                                    --scale-tier $SCALE_TIER \
                                    --runtime-version 1.8 \
                                    --config $HPTUNING_CONFIG \
                                    --job-dir $GCS_JOB_DIR \
                                    --module-name trainer.task \
                                    --package-path trainer/ \
                                    --region us-central1 \
                                    -- \
                                    --train-files $TRAIN_FILE \
                                    --eval-files $EVAL_FILE \
                                    --train-steps 5000 \
                                    --eval-steps 100
```

In the file, the paths to the storage buckets are changed depending on the location and names assigned by the user. The other parameters will not be discussed here, but can be changed depending on the type of training to be performed. The example in Program A.1 performs distributed training using hyper-parameters, but if hyper-parameters are not needed, the command HPTUNING_CONFIG can be removed. In order to use hyper-parameters in the training and this command, another file is needed to contain the hyper-parameters. An example of a hyper-tuning file is shown in Program A.2. The file type for these parameters is '.yaml'.

Program A.2. *hptuning_config.yaml file adapted from Bisong [10].*

```
trainingInput:
  hyperparameters:
    goal: MAXIMIZE
    hyperparameterMetricTag: accuracy
    maxTrials: 4
    maxParallelTrials: 2
    params:
      - parameterName: learning-rate
        type: DOUBLE
        minValue: 0.00001
        maxValue: 0.005
        scaleType: UNIT_LOG_SCALE
      - parameterName: first-layer-size
        type: INTEGER
        minValue: 50
        maxValue: 500
        scaleType: UNIT_LINEAR_SCALE
      - parameterName: num-layers
        type: INTEGER
        minValue: 1
        maxValue: 15
        scaleType: UNIT_LINEAR_SCALE
      - parameterName: scale-factor
        type: DOUBLE
        minValue: 0.1
        maxValue: 1.0
        scaleType: UNIT_REVERSE_LOG_SCALE
```

In Program A.2, there are four parameters being included. The file can contain as many parameters as the user wishes to include, and their individual value parameters can be adjusted to the preference of the user. Finally, the 'hyper-tune.sh' file is run from the terminal. By running this file, the 'hptuning_config.yaml' file will automatically be called from within the first file. To execute the run job in the terminal, only one command is needed:

```
source ./scripts/hyper-tune.sh
```

Once the training is complete, the job details will be visible in the 'Jobs' tab of the Cloud MLE service.