Most Cost-Efficient and Accurate Way to Predict Rain in Australia

Nabeel Bacchus
Dept. of Computer Science
The University of Houston
Houston, USA
nybacchus@uh.edu

Sammy Tawakkol
Dept. of Computer Science
The University of Houston
Houston, USA
stawakkol@uh.edu

***Abstract*—The project experiments with and compares the performance of the Amazon Web Services and Google Cloud when building Classification models with machine learning algorithms such as Logistic Regression, K-Nearest Neighbor, and Decision Tree. By calculating the time it takes to build a different classification model, choosing the service that can build the model faster would suggest which cloud service will be cheaper to use while providing the same results. In addition to testing different machine learning algorithms, the scalability of the models and the performance of different instances the cloud services provide will be tested. The dataset will be split into 1000, 10000, and 100000 rows to test how well the cloud services handle scalability while building the classification models. The instances tested for AWS are t2.medium, t2.large, and t2.xlarge while the instances tested for Google Cloud are e2.medium, e2-standard-2, and e2-standard-4.**

Keywords—Machine Learning, Cloud Services, Classification

I. Introduction

Cloud services offer a lot of processing power that is useful to train and develop machine learning models. It makes it easier for companies to experiment with machine learning properties and scale up projects as demand increases. It can be too high of a cost for companies to purchase the needed hardware to build, train, and deploy the models they want to create. By using cloud services that use the pay-per-use model, companies and individuals are able to use the powerful processing power without actually investing in hardware. Additionally, most of these cloud services such as Amazon Web Services, or AWS, and Google Cloud provide very advanced AI services that are easy to implement to suit one's needs[1].

With so many different cloud services available for companies and individuals to use, it would also be important to compare them to determine which cloud service provides the best tools for the lowest price. We will be comparing two of the most popular cloud services, Amazon Web Services and Google Cloud, to see which one is faster at building and training a Classification model, provides better scalability, and is the most cost-efficient for predicting rain in Australia.

Amazon Web Services provides a service called Amazon SageMaker which provides the tools to build, train, and deploy machine learning models for predictive analytics applications. Deploying machine learning models can be very challenging, but Amazon SageMaker is able to simplify this process by using common algorithms and other tools to speed up the machine learning process. Many companies do not have the budget to hire specialists and maintain resources dedicated to AI development. Amazon SageMaker uses integrated tools to automate labor-intensive manual processes and reduce human error and hardware costs. Machine learning modeling components are included in the Amazon SageMaker tool set. Software capabilities are abstracted in intuitive SageMaker templates. They provide a framework to build, host, train, and deploy machine learning models at scale in AWS. When using SageMaker, machine learning modeling gets simplified into three steps: preparation, training, and deployment. Preparation includes creating a machine learning instance in Amazon Elastic Compute Cloud, or EC2, and running a Jupyter Notebook on said instance. SageMaker provides a variety of built-in training algorithms, such as linear regression and image classification, or the developer can import custom algorithms. Training includes making an Amazon S3 bucket to store the model. SageMaker Model Monitor provides continuous automatic model tuning to find the set of parameters, or hyperparameters, to best optimize the algorithm. During this step, data is transformed to enable feature engineering. Deployment is when the model is ready, the service automatically operates and scales the cloud infrastructure. It uses a set of SageMaker instance types that include several graphics processing unit accelerators optimized for machine learning workloads[2].

Google Cloud provides a service called Google Cloud ML Engine that is a hosted platform to run machine learning training jobs and predictions at scale. It can be used to train a complex model by leveraging the GPU and TPU infrastructure. The outcome from this step, a fully-trained machine learning model, can be hosted in other environments including on-prem infrastructure and public cloud. The service can also be used to deploy a model that is trained in external environments. Cloud ML Engine automates all resource provisioning and monitoring for running the jobs. It can also manage the lifecycle of deployed models and their versions. It can also perform hyperparameter tuning that influences the accuracy of predictions, which removes the need to manually change hyperparameters to obtain the best model. The steps to building a model in Cloud ML Engine includes data preparation, model creation, model training, model deployment, predictions, and monitoring. Data preparation usually takes place outside of the ML Engine but services such as Cloud Dataflow and Cloud Dataprep are provided to analyze and prepare the dataset. Once the prepared dataset is sent to the Cloud Storage, Python-based toolkits such as Scikit-learn and TensorFlow are used to create the model. To train the model, the training job can run across a cluster of machines backed by GPUs and TPUs. The compute resources required for running the job are dynamically

managed by ML Engine. In the Model Deployment stage, the model is serialized into a supported format and uploaded to a Google Cloud Storage bucket. A model is then registered in Cloud ML Engine pointing to the location of the Cloud Storage bucket where the serialized object is uploaded. In the Prediction stage, Google Cloud ML Engine hosts the fully-trained models for prediction. Like the training jobs are managed across multiple resources, the infrastructure required for model hosting is also dynamically managed by ML Engine. In the Monitoring stage, predictions made are monitored by Google Cloud Stackdriver. This allows for the prediction to be monitored on an ongoing basis by using APIs to examine running jobs[3].

Scalability is important because it allows for the ability to increase and decrease the resources needed to meet changing demands. It offers convenience by allowing easy customization to fit the needs of what a company or individual needs. It is flexible and is able to respond quickly to work overloads. Finally, It allows for companies to save money by allowing them to avoid purchasing expensive hardware and only paying for what is needed.

Cost-efficiency is an important factor that needs to be considered so companies and individuals would not spend more money than they have to. Analyzing the accuracy and time to train the model on each instance allows users to determine the correct instance that is needed to train their model without splurging their money on a very expensive instance.

II. Approach

In order to test the time it takes to train a model, we first must pick a dataset to train and choose a hypothesis. The dataset chosen to test the performance of the instances is called “Rain in Australia”[4]. It contains about 10 years of daily weather observations across the whole of Australia. It contains 145460 rows and 23 columns. The dataset contains the following features:

- **Date:** The date of observation
- **Location:** The common name of the location of the weather station
- **MinTemp:** The minimum temperature in degrees celsius
- **MaxTemp:** The maximum temperature in degrees celsius
- **Rainfall:** The amount of rainfall recorded for the day in mm
- **Evaporation:** The so-called Class A pan evaporation (mm) in the 24 hours to 9am
- **Sunshine:** The number of hours of bright sunshine in the day.
- **WindGustDir:** The direction of the strongest wind gust in the 24 hours to midnight
- **WindGustSpeed:** The speed (km/h) of the strongest wind gust in the 24 hours to midnight
- **WindDir9am:** Direction of the wind at 9am
- **WindDir3pm:** Direction of the wind at 3pm
- **WindSpeed9am:** Wind speed (km/hr) averaged over 10 minutes prior to 9am
- **WindSpeed3pm:** Wind speed (km/hr) averaged over 10 minutes prior to 3pm
- **Humidity9am:** Humidity (percent) at 9am
- **Humidity3pm:** Humidity (percent) at 3pm
- **Pressure9am:** Atmospheric pressure (hpa) reduced to mean sea level at 9am
- **Pressure3pm:** Atmospheric pressure (hpa) reduced to mean sea level at 3pm
- **Cloud9am:** Fraction of sky obscured by cloud at 9am. This is measured in ‘oktas' , which are a unit of eighths. It records how many eighths of the sky are obscured by clouds. A 0 measure indicates completely clear sky whilst an 8 indicates that it is completely overcast.
- **Cloud3pm:** Fraction of sky obscured by clouds (in "oktas": eighths) at 3pm.
- **Temp9am:** Temperature in degrees Celsius at 9am
- **Temp3pm:** Temperature in degrees Celsius at 3pm
- **RainToday:** Boolean: 1 if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0
- **RainTomorrow:** The amount of rain next day in mm. Used to create a response variable RainTomorrow. A kind of measure of the "risk".

The purpose of this dataset is to predict if rain will occur the next day in Australia, or the “RainTomorrow” variable, which produces a binary answer. To predict this variable and check how fast each instance is able to build and train the model, we will use three different machine learning algorithms: Logistic Regression, K-Nearest Neighbor, and Decision Trees. This is to find the most accurate model and determine which model is built and trained the fastest.

To determine which cloud service has better scalability, the data set will be split into three different sizes: 1000, 10000, and 100000 entries. Then, the model will be built and trained on those sizes and the accuracy and time to build the model will be compared.

The Amazon EC2 instance types that we will be experimenting with are t2.medium, t2.large, and t2.xlarge. The t2 type instances are Burstable Performance Instances that provide a baseline level of CPU performance with the ability to burst above the baseline. The t2.medium instance uses 2 vCPUs, 4 gigabytes of memory to operate, and costs about $0.0464 per hour. The t2.large instance uses 2 vCPUs, 8 gigabytes of memory to operate, and costs about $0.0928 per hour. The t2.xlarge uses 4 vCPUs, 16 gigabytes of memory to operate, and costs about $0.1856 per hour.

The Google Cloud Compute instance types that we will be experimenting with are e2.medium, e2-standard-2, and e2-standard-4. The e2 type instances provide a variety of compute resources for the lowest price on Compute Engine. The e2.medium instance uses 2 vCPUs, 4 gigabytes of memory to operate, and costs about $0.0464 per hour. The e2-standard-2 instance uses 2 vCPUs, 8 gigabytes of memory to operate, and costs about $0.15901 per hour. The e2-standard-4 instance uses 4 vCPUs, 16 gigabytes of memory to operate, and costs about $0.31802 per hour.

Overall, we will be measuring the performance of the AWS and Google Cloud instances and the classification models by:

- Comparing the accuracy of each model
- Comparing the time to build and train each model
- Comparing the cost to use the instance to the amount of time needed to create the model.

III. Implementation

The implementation strategies fall in line with each respective service and their instances. For the Amazon AWS service, the EC2 service will be used to run each instance, with datasets and results stored in the S3 storage system. SageMaker and QuickSight will be used for the machine learning and classification solutions.

Through Google Cloud and its services, Google Compute will be used for the instance service, and Cloud Storage for dataset and result storage. Google Colab can be used to effectively run code through the Jupyter Notebook both on the cloud service and a local PC, and Dataflow will be used with Vertex AI for the machine learning and classification solutions.

The Python Programming language will be used through the Jupyter Notebook environment. The Numpy library will be used to analyze and clean up data, Pandas will manipulate the dataset, MatPlotLib and Seaborn will be used for the visualization of any results, and Scikit Learn will be used for executing the deployment, training, and testing of the machine learning models.

With these steps, after using the storage systems with the dataset and results, do exploratory analysis with dataflow and quicksight, deploy, build, and run the models with Sagemaker, Google AI, and Jupyter Notebook on each instance. With all of this, analyze and report on time it takes to train models, and the accuracy of the results of such models.

The end goal from this is to identify which services are most cost efficient to predict rainfall in Australia for any end user either accessing a public website, or a corporation using it to disseminate information to a large group of users, such as news networks or businesses which depend on weather to have daily operations.

IV. Tools Used

Table I. Tools Used Table

Tool Used	Category
Python	Programming Language
Jupyter Notebook	Python Environment
Google Cloud	Cloud Service
Amazon AWS	Cloud Service
Google Cobol	Cloud/Local Service
Numpy	Python Library
Pandas	Python Library
MatPlotLib/Seaborn	Python Library
Google Compute	Machine Learning Instance
Google Vertex AI	Machine Learning Instance
Google Dataflow	Machine Learning Instance
Amazon SageMaker	Machine Learning Instance
Amazon Quicksight	Machine Learning Instance
Scikit Learn	Python Machine Learning Tool

Fig. 1. Shows that tools and libraries that are being used and what kind of tool it is.

V. Roles

Table II. Roles

Role	Person Responsible
Building and Deploying Models	Both
Analyzing Performance of Models and Instances	Both
Writing Reports	Nabeel
Cleaning Data	Nabeel
Analyzing Initial Data	Sammy
Implementing models on services	Sammy

Fig. 2. Shows what roles each member will perform to complete the project

VI. Results

The following are the tables of results. The first three are the timing results for Amazon, next three for Google.. The next set of six are divided the same way for accuracy in the same order. They will show the result for Logistic Regression (Regression), K-Nearest Neighbors (KNN), and Decision Tree (Decis-Tree). The results are displayed below in Table 3 to Table 14:

TABLE III. T2.XLARGE TRAINING TIME

Model/Training size	1000	100000	100000
Regression	0.0094267	0.09503991	1.69566238
KNN	0.0043462	0.1427754	0.16888775
Decis-Tree	0.0225226	0.25350527	3.53805208

Fig. 3. Shows the time it took to train each model on Amazon's t2.xlarge instance in seconds.

TABLE IV. T2.LARGE TRAINING TIME

Model/Training size	1000	100000	1000000
Regression	0.0099416	0.09445295	1.61380679
KNN	0.0047305	0.01442324	0.16761214
Decis-Tree	0.0234650	0.23333830	3.67861899

Fig. 4. Shows the time it took to train each model on Amazon's t2.large instance in seconds.

TABLE V. T2.MEDIUM TRAINING TIME

Model/Training size	1000	100000	1000000
Regression	0.0093694	0.09433571	1.68100277
KNN	0.0043948	0.01457980	0.17124041
Decis-Tree	0.0248510	0.24671627	3.86633959

Fig. 5. Shows the time it took to train each model on Amazon's t2.medium instance in seconds.

TABLE VI. E2.STANDARD-4 TRAINING TIME

Model/Training size	1000	10000	1000000
Regression	0.0240739	0.1242056	0.4485327
KNN	0.0047155	0.0142881	0.0439606
Decis-Tree	0.0188957	0.2025898	0.7858159

Fig. 6. Shows the time it took to train each model on Google's e2.standard-4 instance in seconds.

TABLE VII. E2.STANDARD-2 TRAINING TIME

Model/Training size	1000	10000	1000000
Regression	0.0116520	0.1069996	0.7016204
KNN	0.0039408	0.0168335	0.0646989
Decis-Tree	0.0269514	0.2308174	3.1139359

Fig. 7. Shows the time it took to train each model on Google's e2.standard-2 instance in seconds.

TABLE VIII. E2.MEDIUM TRAINING TIME

Model/Training size	1000	10000	100000
Regression	0.0201533	0.1325864	2.5758995
KNN	0.0052483	0.0171631	0.2944071
Decis-Tree	0.0269514	0.2308174	3.1139359

Shows the time it took to train each model on Google's e2.medium instance in seconds.

TABLE IX. T2.XLARGE MODEL ACCURACY

Model/Training size	1000	10000	100000
Regression	0.8263300	0.8466542	0.8500650
KNN	0.7800555	0.7923977	0.8105066
Decis-Tree	0.7812862	0.7747594	0.7925032

Fig. 9. Shows the accuracy of each model on Amazon's t2.xlarge instance.

TABLE X. T2.LARGE MODEL ACCURACY

Model/Training size	1000	10000	100000
Regression	0.8293188	0.8486936	0.8500650
KNN	0.7813214	0.7933120	0.8105066
Decis-Tree	0.7645486	0.7832202	0.7925032

Fig. 10. Shows the accuracy of each model on Amazon's t2.large instance.

TABLE XI. T2.MEDIUM MODEL ACCURACY

Model/Training size	1000	10000	100000
Regression	0.815066	0.8282288	0.8492563
KNN	0.7817785	0.7936987	0.8105066
Decis-Tree	0.7460177	0.7830795	0.7925053

Fig. 11. Shows the accuracy of each model on Amazon's t2.medium instance.

TABLE XII. E2.STANDARD-4 MODEL ACCURACY

Model/Training size	1000	10000	100000
Regression	0.8230550	0.8375237	0.8413187
KNN	0.7758538	0.7935246	0.7988614
Decis-Tree	0.7561669	0.767220	0.7858159

Fig. 12. Shows the accuracy of each model on Google's e2.standard-4 Model instance.

TABLE XIII. E2.STANDARD-2 MODEL ACCURACY

Model/Training size	1000	10000	100000
Regression	0.8200492	0.8378455	0.8386028
KNN	0.7821847	0.7908936	0.8015903
Decis-Tree	0.7739492	0.774065	0.7775463

Fig 13. Shows the accuracy of each model on Google's e2.standard-2 instance.

TABLE XVI. E2.MEDIUM MODEL ACCURACY

Model/Training size	1000	10000	100000
Regression	0.8185802	0.8401768	0.8422299
KNN	0.7784665	0.7924826	0.8084335
Decis-Tree	0.7754658	0.7759396	0.7856917

Fig. 14. Shows the accuracy of each model on Google's e2.medium instance.

From the results, we can see that K-Nearest Neighbors has the fastest training time than any other model across all instances. Decision Trees tend to have the slowest training time across all instances and training size. The e2.standard-4 instance has the fastest training time for K-Nearest Neighbors when the training size is 100000 with a time of approximately 0.043 seconds. The t2.medium instance has the slowest training time for Decision Trees when the training size is 100000 with a time of approximately 3.87 seconds. Overall, we see that Google's instances has faster training times than Amazon's instances

We can see that the model with the highest accuracy is the Logistic Regression model trained with 100000 entries on t2.large and t2.xlarge instance. They both have an accuracy of about 85%. The model with the lowest accuracy is Decision Tree model trained with 1000 entries on t2.medium instance. This results in about 74.6% accuracy. We generally see that the regression models have higher accuracy across all the instances. Decision Trees tend to have the lowest accuracy across all the instances. We found that models trained on Amazon's instances tend to have a higher accuracy for all models than Google's instances.

The most cost-efficient way of predicting rain in Australia would be to use a t2.medium instance and train a K-Nearest Neighbor Classifier model with a random sample size of 1000. While this is the most cost-efficient way to predict rain in Australia, it would produce a model with approximately 78.2% accuracy which is 6.8% less accuracy than the model with the highest accuracy. This was calculated by first calculating how much it cost to train each model. Then, we divided the accuracy of the model by the cost to train the model.

Some additional information that was found was that the least cost-efficient way to predict was using a t2.xlarge instance and train a Decision Tree Classifier model with a random sample size of 100000. This shows that while Amazon is significantly cheaper than Google, as the dataset gets larger, Amazon takes longer to train the model than Google. Thus, for very large datasets, it would be more cost-effective to train a model with Google; and for relatively small datasets, it would be more cost-effective to train a model with Amazon.

VII. CONCLUSION

Amazon is the better choice for predicting rain in Australia in a cost efficient manner. The most accurate way to predict is to use a t2.large model to train a Logistic Regression model with a training set of 100000. The most cost-effective way to predict is to use a t2.medium instance to train a Logistic Regression model with a training set of 1000.

REFERENCES

[1] Guy Hummel, & Cook, J. (2019, May 21). What are the benefits of machine learning in the cloud? Cloud Academy. Retrieved October 12, 2021, from https://cloudacademy.com/blog/what-are-the-benefits-of-machine-learning-in-the-cloud/

[2] Kranz, G., &; Carty, D. (2021, August 31). What is Amazon Sagemaker? SearchAWS. Retrieved October 21, 2021, from https://searchaws.techtarget.com/definition/Amazon-SageMaker.

[3] MSV, J. (2018, October 25). Google Cloud ML Engine: Train, and deploy machine learning models. The New Stack. Retrieved October 21, 2021, from https://thenewstack.io/google-cloud-ml-engine-train-and-deploy-machine-learning-models/.

[4] Young, J., & Young, A. (2017, December 4). Rain in Australia. Retrieved October 18, 2021, from https://www.kaggle.com/jsphyg/weather-dataset-rattle-package/metadata.