



Universitetet
i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study program/specialization: Computer Science	Spring semester, 2016 Open / Confidential
Author: Nketah, Gabriel Uchechukwu (signature author)
Supervisor(s): Associate Professor Dr. Tomasz W. Wiktorski	
Title of Master's Thesis: Comparison of Cloud Machine Learning Services	
ECTS: 30	
Subject headings:	Pages: 50 + Attachments/ other: Stavanger – 15 th June, 2016

This thesis is dedicated to God

Acknowledgement

Special thanks go to my supervisor *Associate Professor Dr. Tomasz W. Wiktorski*, for his encouragement and for devoting his time to supervise my Thesis. And to my wife – *Mrs. Nketah S. Oluwatobi*, I say a big thank you to her. I could not have accomplished my study program without her immense love and constant support. For my welfare throughout my stay here in Norway, I thank my brothers *Mr. Nketah Victor, Mr. Nketah Benjamin, Mr. Nketah Daniel* and their families.

Thank you,

Nketah, Gabriel Uchechukwu – *June, 2016*

Abstract

In this paper, three Cloud Machine Learning services are considered using the same dataset to run predictions on each of them. These services are quantitatively and qualitatively analyzed and compared by considering their mode of operation, data processing, prediction creation, model creation, cost, and algorithm. Although Google Prediction API offers fast model training and model creation as compared to Windows Azure Machine Learning Studio and Amazon Machine Learning; it has lesser visualization tools. The focus of this thesis is to provide better understanding on how these services operate, identify important research directions in Machine Learning field, and present a clear picture of the functionalities of the services to aid the decision of developers when choosing which service will best suit their Machine Learning solutions.

Keywords: Machine Learning – *ML*, Algorithms, Predictive Analytics, and Cloud-computing.

When I was a programmer, I was very good at figuring out all the algorithms and writing them all down. Today, I think I would try to figure out how to program a computer to learn something – Eric Schmidt, Google.

List of Instances

Instance 1 – Sample Code for Dataset Formatting	36
Instance 2 – Sample Codes to Test Microsoft Azure Machine Learning Online.....	36

List of Tables

Table 1 – Dataset Description.....	5
Table 2 – Dataset Attribute Information	7
Table 3 – Results from Louis Dorard's Comparison.....	8
Table 4 – Results from Weka Experiment	9
Table 5 – Results from Google Prediction API Experiment.....	10
Table 6 – Results from Prior Knowledge Experiment	11
Table 7 – Results from BigML Experiment.....	11
Table 8 – Pricing	31
Table 9 – Data Processing.....	32
Table 10 – Algorithms and Evaluations.....	33
Table 11 – Model Processing.....	34

Table of Figures

Figure 1 – Probabilistic Modeling Pipeline	2
Figure 2 – Supervised Learning Model Method	4
Figure 3 – Sample Dataset before Formatting	5
Figure 4 – Sample Dataset after Formatting	6
Figure 5 – Google Prediction API Workflow	13
Figure 6 – Google Developers Console Project Dashboard.....	14
Figure 7 – List of Methods Google API Provides.....	15
Figure 8 – Google Trainmodels.Get Method Results	16
Figure 9 – Google Trainedmodels.Analyze Method Result.....	16
Figure 10 – Google Trainmodels.Predict Method Results.....	17
Figure 11 – Azure Machine Learning Work-flow	18
Figure 12 – Blanc Experiment on Azure Workspace.....	19
Figure 13 – Azure Data-upload Window	20
Figure 14 – Azure Data Visualization Tool: Age Field Visualization.....	20
Figure 15 – Azure Data Visualization Tool: Model Scoring Result.....	21
Figure 16 – Azure Model Evaluation Result	22
Figure 17 – Azure Classification Algorithms and Web Service Interactive Testing Window	23
Figure 18 – Azure Web Service Dashboard.....	23
Figure 19 – Connected Models on Azure Studio.....	24
Figure 20 – Automatically Activating Web Service on Azure Studio.....	25
Figure 21 – Amazon Machine Learning Workflow [11]	26
Figure 22 – Amazon Data-source Console	26
Figure 23 – Amazon Data Insight: Target/ Label Values’ Binary Attributes	27
Figure 24 – Amazon Data Insights: Job Field Categorical Attributes	27
Figure 25 – Amazon Model Performance Evaluation.....	28
Figure 26 – Amazon Model Creation Summary	29
Figure 27 – Amazon Interactive Real-Time Prediction Testing Window	30
Figure 28 – AUC Metric of Amazon Machine Learning Model	30

Table of Contents

Acknowledgement	iii
Abstract	iv
List of Instances	v
List of Tables	v
Table of Figures	vi
Table of Contents	vii
Chapter 1: Introduction	1
1.1 Thesis Organization	1
Chapter 2: Literature Review	2
2.1 Machine Learning Overview	2
2.1.1 Machine Learning Algorithms and Models	3
2.1.1.1 Supervised Learning	4
2.1.1.2 Unsupervised Learning	5
2.1.2 Machine Learning Data Preparation	5
2.1.2.1 Machine Learning Data Classifications	8
2.2 Prior Research Comparing Machine Learning Services	8
2.2.1 Comparing Machine Learning APIs Performance	9
2.2.2 Machine Learning Throwdown.....	9
2.2.2.1 Waikato Environment for Knowledge Analysis	10
2.2.2.2 Prior Knowledge	11
2.2.2.3 Google Prediction API.....	12
2.2.2.4 BigML API	12
Chapter 3: Experiments and Results	13
3.1 Google Prediction API.....	13
3.1.1 Configuring Google Prediction API	14
3.1.1.1 Step 1: Upload Dataset.....	15
3.1.1.2 Step 2: Train Model	15
3.1.1.3 Step 3: Send Prediction Queries.....	16

3.2 Windows Azure Machine Learning Studio.....	17
3.2.1 Configuring Azure Machine Learning Studio.....	18
3.2.1.1 Step 1: Upload Dataset.....	19
3.2.1.2 Step 2: Split Dataset.....	20
3.2.1.3 Step 3: Train Model.....	21
3.2.1.4 Step 4: Evaluate Model.....	21
3.2.1.5 Step 5: Deploy and Test Model.....	23
3.3 Amazon Machine Learning.....	25
3.3.1 Configuring Amazon Machine Learning.....	26
3.3.1.1 Step 1: Upload Dataset.....	27
3.3.1.2 Step 2: Split Dataset.....	28
3.3.1.4 Step 5: Create Predictions.....	29
Chapter 4: Comparative Analyses.....	31
4.1 Comparing Google, Amazon, and Microsoft Machine Learning Services.....	31
4.1.1 Configuration Process.....	31
4.1.2 Data Processing.....	32
4.1.3 Model Processing.....	33
Chapter 5: Conclusion.....	35
Appendix.....	36
References.....	40

Chapter 1: Introduction

In the past, developing Machine Learning – *ML* solutions requires deep Information Technology – *IT* know-how and custom software which makes the implementation to be somewhat expensive. However, in a bid to simplify ML solutions, different cloud-based service providers in recent years have created open doors to provide IT professionals with relatively inexpensive and easily accessible ML solutions. Targeting audiences like data analysts, developers, and business intelligence professionals amongst others professionals in numerous fields. These open doors started when cloud-computing emerged as a new paradigm for delivering and hosting services for consumption over the internet. With these open doors, there emerge need to consider and compare the functionalities of these services which will be the focal point of this thesis. Cloud-computing is a computing model that provides resources *like storage* as general services that can be deployed on the web. Users can access or lease these services through the internet in an on-demand method. One of the primary aims of cloud-computing is to enhance the opportunity to replace the expensive up-front capital infrastructure expenses with a relatively inexpensive infrastructure that is scalable for businesses.

The explosive growth of digital data, big data analytics, cloud-computing frameworks, cloud data market, and computer systems interconnections via the internet has increased the way individuals and organizations depend on the information that are stored and transmitted via the cloud-computing media. This rapid growth led to the awareness of the need to develop smart cloud-computing since different cloud and web service providers make use of cloud computing to deliver and host services over the internet. This trend has impacted the IT industry tremendously over the past few years and in recent years; it has prompted the three major cloud-computing service providers *Amazon, Google, and Microsoft* to release cloud ML services. These services are Amazon ML, Google Prediction API, and Windows Azure ML Studio. One of the goals behind releasing these services is to apply the strength of cloud-computing to solve big data processing problem. ML solutions can be adapted into many critical application areas like image recognition, data mining, and natural language processing among many others.

1.1 Thesis Organization

Chapter 2 gives an overview of ML, survey of existing works based on comparing some existing and related ML services. In addition, some common ML algorithms are considered alongside the associated workflow of modern data science principles. Chapter 3 presents the experiments and results from some predictive analytics models that were built using different ML algorithms and modeling strategies on each service. These services vary in multiple ways from the types of model that can be created, to the level of expertise required, and the ease with which each of the services can be integrated into businesses. In chapter 4, comparison of the three services, discussion, and analysis of the results from the comparisons are presented. Finally, conclusions are presented in chapter 5.

Chapter 2: Literature Review

In a traditional programming paradigm, the computer takes data and programs it as input to produce a desired output; but in Machine Learning – *ML* paradigm, the computer takes data and desired output as input to produce a program. The major advantage of the ML paradigm when compared with the traditional program is that, the program that is developed by the ML instance has been trained with several training data and desired output which gives the proficiency of making relatively accurate prediction of a desired output based on the provided data [15].

The modern probabilistic modeling for massive data is based on the concept of discovering useful patterns in massive data and can be very useful in cases where we have people we know from different communities like work, city, and neighborhood. These separate communities can be revealed automatically using probabilistic modeling. Figure 1 show the probabilistic modeling pipeline, where knowledge is used to make assumptions about unseen data. Assumption plus data are used to discover particular patterns and then, finally, the patterns are used to make predictions. Given a particular model, we can use an algorithm to discover the hidden patterns in the data. A classic example of algorithm that can do this is the *stochastic variation inference*. This is a scalable and generic inference that works in a lot of different kinds of model and scales to massive datasets by taking a massive dataset and an estimate of the patterns. Pattern recognition field also known as *statistical classification* is a research field that has a close tie with ML. The goal of pattern recognition is to build a classifier that can be used to derive the class of an input pattern which is the training process in ML [17].

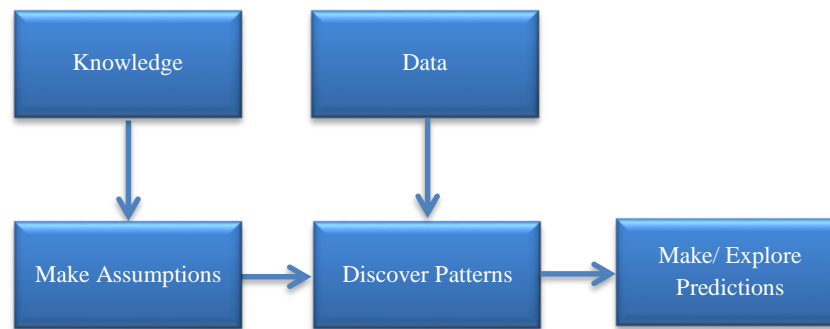


Figure 1 – Probabilistic Modeling Pipeline

2.1 Machine Learning Overview

In the 1960s, researchers in cognitive science were thinking of how possible it would be for an entity *human, animal, or computer* to sit on a table and learn something through means of abstracting what the entity has learnt mathematically into a setting where collections of points describing measurements can be extracted. Then in the 1980s, researchers came up with an abstraction called *Rules-Based AI* which is a combination of computer science and AI. In this abstraction, precise rules are set for the computer to work with. One good example can be seen in a robotic vacuum-cleaner that runs into a wall, backs up, turns sideways, tries to continue, if it runs into a wall again, it backs up, turns sideways and continues with the operations. Several years later, focus gradually shifted from building an AI from the very beginning to simply teaching computers things that humans are good at. The idea of mimicking humans'

intelligent abilities by a machine is rooted in AI and cognitive science, and the fact that computers do not make mistakes as long as they are programmed correctly makes them a suitable mode of learning what humans can learn. A good example is how humans can learn the shape of a car and when any other thing that looks so much like a car is seen, humans easily recognize it as a car [4][17].

Machine Learning can be defined as a field of study that gives computers the ability to learn without being explicitly programmed [1]. A computer program is said to learn from experience E with respect to some class of task T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E [6].

The above definitions can be simplified as the ability of making a program or a tool to predict future performance measure of a given task. This means that a computer can be programmed in such a way that it can possess the ability to gain knowledge from past experience just as a wise saying goes – *experience is the best teacher*. One of the primary objectives of an ML solution is to program computers to use sample data or past experience to solve a given problem. A classic ML solution consists of an ML model that is developed using dataset and ML algorithms. The model is developed by inputting dataset about past events into the algorithm. To achieve a good ML solution that is capable of answering predictive questions, the problem needs to be framed by giving consideration to past events alongside with what is about to be predicted, and after framing the problem, next is to gather historical data *information from a related past events*, create model *aim of ML model is to make accurate prediction from unseen data*, and finally, make and explore predictions.

In ML, data is considered the key component because it handles virtually everything. Therefore, ML can be described in terms of its data-power as a computer algorithm that search for patterns in data and the perfect tool for describing these patterns is naturally embedded in statistics; this makes the relationship between ML and statistics to be fairly significant [4][17]. Predictive analytics also happens to be among the underlying technologies behind ML. It is the scientific way the past is used to predict the future with the aim of deriving a desired outcome, this outcome can then be used to analyze questions about future events and also, provide answers to business related questions based on probability. Predictive analytics is very much part of day-to-day activities which have made it to be deeply integrated into the society at large. From predicting what kind of food we like, to determining if a customer will default when given a loan, it can also be used to predict flights in and out of airports and so many examples we can think of in client-consumer's daily routine. The outcomes of any of these predictions will be determined by the use of predictive analytics. A visit on *ebay.com* or any online stores could give an ideal example of predictive analytics, there, every time we search for an item, we will be presented with some additional catalogs of items, these additional items are generated from predictive analytics and the psychology of human buying patterns, which means, customers that bought the item we searched for also bought the items on the additional catalogs. This concept provides a highly effective marketing strategy [16][17]. With the combination of predictive analytics, cognitive science, AI, statistics, pattern recognition and other relative fields, alongside with optimization, algorithms, and other tools *like visualization tools*, the functionality of ML could be easily abstracted.

2.1.1 Machine Learning Algorithms and Models

In mathematics and computer science, an algorithm is defined as a method that is expressed as a finite list of well-defined instructions for executing a function – *simply defined as the step-by-step procedure for calculations*. The three most important criteria for algorithms are time, resource usage *like memory*, and simplicity *simple to implement or code*. In the case of ML algorithms, the three main resources are memory, training data, and time – *with the rate digital data is becoming more available; the time to process the huge available data seems to be really limited [17][21]*. ML algorithms operate by discovering patterns in data and then use the discovered patterns to develop a model that provides the summary of the important properties in the data. It can also figure out how to perform important tasks by generalizing from examples. ML algorithms and ML models are not the same thing. The algorithm consumes the dataset and detects patterns that map the variables to the target. Once the pattern is learned,

the algorithm outputs a model that represents the relationships in the patterns.

There are many powerful ML algorithms and each one has its own strengths and weaknesses. New understanding in life naturally precedes learning; this also applies to ML algorithms. The act of making an instance produce an answer after the instance has passed through a *learn* phase is called *predict*. The *learn* phase, in turn is any understanding that could be acquired from previous knowledge. Supervised learning and unsupervised learning are the two important approaches by which ML achieves its *learning* phase. Although these two approaches are not exhaustive because there are methods that are kind of in-between supervised and unsupervised learning. The distinct difference between these two learning approaches is that for supervised learning, marked patterns are made available while for unsupervised learning marked patterns are not made available. ML algorithms normally fall into supervised or unsupervised learning categories.

Figure 2 describes the supervised learning model method, where models are trained with known inputs and outputs, the concept is to create a function that can predict future outputs when provided with new inputs *example: classification algorithms and regression algorithms*. On the other hand, unsupervised depends on the system to self-analyze the data and infer patterns to create models *example: clustering algorithms and anomaly detection* [17].

As a rule of thumb, a dumb ML algorithm with lots and lots of data beats a clever one with modest amount of it. After all ML is all about letting data to handle the heavy lifting [17].



Figure 2 – Supervised Learning Model Method

2.1.1.1 Supervised Learning

The supervised learning abstraction in ML tries to look out for the mapping between inputs and outputs using already known results. A good instance is when trying to predict some unknown answers using data that contains answers that are readily available. After the data has been derived, the data is split into two random sets, *training dataset* and *testing dataset* often in ratio 70:30 respectively. The 70 percent is used for training a robust model what it needs to learn from; while the 30 percent is used to test the model to be sure the model behaves well with new datasets. After selecting and cleaning the dataset to conform with ML requirements, a point is selected in the dataset that shows *this is the right answer* which is also known as *label or target*, then an algorithm *one or more* is chosen to compute the outcome, and finally, run the program on the dataset to see if the correct answer is gotten from the testing data. The final output becomes a model that can be used against more data to get needed answers based on probability [15][17].

A good example of supervised ML algorithm is the classification algorithms. The algorithm is used to arrange data into different classes that can then be used to predict discrete variables based on some other attributes in the datasets. It is very suitable for predicting responses that can have a few known values based on other columns in the dataset. Figure 3 presents sample dataset with known input variables like job, age, marital status and etc. the last column presents the known outcomes *label* while the other columns presents data input features *vectors*. For each row of training data inputs, there is also a column that indicates the known outcomes based on the combination of data input features. The remaining data input columns would be considered as not used. This illustrates the formula for producing a supervised learning model with known input data elements and known outcomes as inputs *both must have been labeled*. The algorithm analyzes these two inputs from training data and produces a prediction model based on abstracting inferences about the data. Another example of supervised learning algorithm is the

regression algorithm. This algorithm is used for predicting continuous variables like profit and loss, based on other attributes in the dataset [15][17].

```
yes"; "no"; "unknown"; 14; "may"; 57; 2; -1; 0; "unknown"; "no"; 43; "services"; "married"; "primary"; "n
"; "no"; "unknown"; 23; "may"; 250; 1; -1; 0; "unknown"; "no"; 31; "services"; "married"; "secondary"; "n
llular"; 30; "jul"; 149; 2; -1; 0; "unknown"; "no"; 53; "admin."; "married"; "secondary"; "no"; 105; "no"
"cellular"; 6; "may"; 765; 1; 342; 2; "failure"; "yes"; 23; "services"; "single"; "tertiary"; "no"; 363;
lular"; 4; "feb"; 204; 1; -1; 0; "unknown"; "no"; 61; "admin."; "married"; "unknown"; "no"; 4629; "yes";
nknown"; 26; "may"; 314; 1; -1; 0; "unknown"; "no"; 54; "technician"; "divorced"; "secondary"; "no"; 784
"cellular"; 7; "jul"; 630; 3; -1; 0; "unknown"; "no"; 32; "technician"; "single"; "tertiary"; "no"; 360;
lar"; 5; "feb"; 701; 1; -1; 0; "unknown"; "no"; 40; "unemployed"; "married"; "secondary"; "no"; 219; "yes
; 279; 1; -1; 0; "unknown"; "no"; 34; "blue-collar"; "married"; "secondary"; "no"; 1831; "yes"; "no"; "un
73; 1; 182; 1; "success"; "no"; 31; "unemployed"; "single"; "primary"; "no"; 406; "no"; "no"; "cellular"
```

Figure 3 – Sample Dataset before Formatting

2.1.1.2 Unsupervised Learning

In the unsupervised learning abstraction, developing ML solutions becomes much more difficult when compared with supervised learning because the ML algorithm is not provided with known input-data and known outcomes. It has to analyze the data by itself and get common patterns and structures to develop a predictive model. This technique could be likened to *density estimates* in statistics where there are attempts to find patterns in the input data. A good example of unsupervised learning algorithm is *clustering algorithm and anomaly detection*. Clustering algorithm discovers natural hidden patterns and classifications in datasets by grouping the data-points based on similarity. The discovered patterns and classifications can then be used to predict the class for a given variable. Purchasing patterns from stores, *buyer or non-buyer, price range etc.*, social network graphs *classification of related people to work, school, family etc.* and many more are good examples of cluster analysis classifications. To better make clustering algorithm to be more functional, a vast amount of data that can form clusters is needed to objectively describe the data [15][17].

Table 1 – Dataset Description

Data Set Characteristics	Multivariate
Number of Instances	45211
Area	Business
Attribute Characteristics	Real
Number of Attributes	17
Date Donated	2012-02-14
Associated Tasks	Classification
Missing Values	N/ A
Number of Web Hits	206204

2.1.2 Machine Learning Data Preparation

In ML, data does majority of the process. So it is important to understand how to format data to meet the requirements for the ML services in-question. Generally, before loading dataset into any ML service, the dataset could be saved in either a database, excel spreadsheet, or in the *comma-separated value – .csv*

format. The .csv file format is like a table in a database with rows and columns. Each row is a single data observation or an example that we want to learn from *example is sales data of a particular car*, and each column contains an attribute of the observation or the property of the data *example is sales price of the particular car*. Each observation contains attributes that are separated by a comma. Most times the first row is used for headers to describe the contents in the columns. For example, in Amazon ML, each row contains examples while the columns contain the variable except the column that contains the label. Label in this case *binary classification* is a *Yes* or *No* answer. The dataset can then be used to design an ML model that would answer the question like: *Which customers are most likely to accept or subscribe to our new products?* With this model, we can attempt to extend marketing offers towards customers who are most likely to receive our new product marketing campaign offer based on the existing reactions of customers during previous product marketing campaigns [27][28][29]. The dataset for the experiments in chapter three can be freely accessed at [27], which is the official website of *UCI Machine Learning Repository*. UCI is a center for ML and intelligence systems that has a collection of databases, domain theories, and data generators that are used by ML community for the empirical analysis of ML algorithms. Its archive was created by *David Aha* and fellow graduate students at UC Irvine in 1987 and has since then been widely used by educators, students, and researchers over the globe as primary source of ML datasets. Among the numerous dataset at UCI, we are going to work with the historical telemarketing campaign dataset which is a very captivating dataset because it can be used for most business marketing campaigns [27][28][29]. This particular dataset was gathered during the campaign by the Portuguese banking institution with the goal to predict if clients will subscribe for a term deposit. It has 41188 rows and 21 columns that contain basic information about customers, describing their behavior towards the telemarketing campaign. With this dataset, ML model will be developed that will try to identify potential class of customers that are most likely to respond favorably to a marketing campaign of a new product. This will help to know how to focus marketing campaigns of new products towards the predicted class of customers [27][28][29].

```
age,job,marital,education,default,housing,loan,contact,month,day_of_week,duration,campaign,pdays,p
44,blue-collar,married,basic.4y,unknown,yes,no,cellular,aug,thu,210,1,999,0,nonexistent,1.4,93.444
53,technician,married,unknown,no,no,no,cellular,nov,fri,138,1,999,0,nonexistent,-0.1,93.2,-42,4.02
28,management,single,university.degree,no,yes,no,cellular,jun,thu,339,3,6,2,success,-1.7,94.055,-3
39,services,married,high.school,no,no,no,cellular,apr,fri,185,2,999,0,nonexistent,-1.8,93.075,-47.
55,retired,married,basic.4y,no,yes,no,cellular,aug,fri,137,1,3,1,success,-2.9,92.201,-31.4,0.869,5
30,management,divorced,basic.4y,no,yes,no,cellular,jul,tue,68,8,999,0,nonexistent,1.4,93.918,-42.7
37,blue-collar,married,basic.4y,no,yes,no,cellular,may,thu,204,1,999,0,nonexistent,-1.8,92.893,-46
39,blue-collar,divorced,basic.9y,no,yes,no,cellular,may,fri,191,1,999,0,nonexistent,-1.8,92.893,-4
36,admin.,married,university.degree,no,no,no,cellular,jun,mon,174,1,3,1,success,-2.9,92.963,-40.8,
```

Figure 4 – Sample Dataset after Formatting

Sometimes, datasets come in formats that are not supported by ML services *most ML services supports .csv formats*. An instance is the dataset for this thesis. The dataset has the label values as *Yes/ No* instead of *1's/ 0's*, and semi colons instead of commas. The few lines of code in the Appendix – *Instance 1* could be used to re-format the data. What the code does is to import the .csv format module, define the data path as variable, open file handle for the file and call content, and finally, create .csv header and give it the open file handle which indicates delimiter '=' as input file. The result of the cleanup changed the label value from as *Yes/ No* to *1's/ 0's* as required. The .csv file must satisfy the following conditions [11]:

- It must consist of one observation per line.
- Each observation must be terminated with an end-of-line character.
- The attribute values in each observation must be separated by a comma.
- It must be in plain text using character set like EBCDIC, Unicode, and ASCII.
- Each observation must have the same number of attributes and in the same sequence.

- End-of-line characters must not be included in attribute values, even if the attribute value is enclosed in double quotes.
- Each observation must not exceed 10MB. Amazon rejects any observation that exceeds 10MB during processing. If the rejected observations are more than 10,000, it rejects the entire .csv file.

Table 2 – Dataset Attribute Information

Input Variables – Client Data		
1.	age	numeric
2.	job	categorical – type of job ('admin.', 'blue-collar', 'entrepreneur', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown')
3.	marital	categorical – marital status ('divorced', 'married', 'single', 'unknown') – note that divorced means: divorced or widowed
4.	education	categorical ('basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'illiterate', 'professional.course', 'university.degree', 'unknown')
5.	default	categorical – has credit in default? ('yes', 'no', 'unknown')
6.	housing	categorical – has housing loan? ('yes', 'no', 'unknown')
7.	loan	categorical – has personal loan? ('yes', 'no', 'unknown')
8.	contact	categorical – contact communication type ('cellular', 'telephone')
9.	month	categorical – last month of year ('jan.', 'feb.', ... , 'nov.', 'dec.')
10.	day_of_week	categorical – last contact day of the week ('mon', 'tue', 'wed', 'thu', 'fri')
11.	duration	numeric – last contact duration in seconds
12.	campaign	numeric – number of contacts performed during campaign
13.	pdays	numeric – number of days that passed by after the client was last contacted from the previous campaign, numeric 999 means that client was not previously contacted
14.	previous	numeric – number of contacts performed before the campaign
15.	poutcome	categorical – outcome of the previous campaign ('failure', 'success', 'nonexistent')
16.	emp.var.rate	numeric – unemployment variation rate, quarterly indicator
17.	cons.price.idx	numeric – consumer price index, monthly indicator
18.	cons.conf.idx	numeric – consumer confidence index, monthly indicator
19.	euribor3m	numeric – euribor 3 month rate, daily indicator
20.	nr.employed	numeric – number of employees
Output Variable – Desired Target (Label)		
21.	y	binary – has the client subscribed a term deposit? ('yes', 'no')

2.1.2.1 Machine Learning Data Classifications

A *class* is a set of things having some properties in common and differentiated from others by kind, type, or quality which means classification is the act of classifying things [35]. During classification ML, focus is to find an ML object or a mathematical function *known as classifier* that is capable of predicting the class a particular data point belongs to. Often times, datasets come with data that are separated into two or more classes and classifying them is one of the crucial worries in ML paradigm because encountering training error when training classifiers is very much likely. The error classifier is the number of misclassified training points divided by the total number of training points. A practical example of a classification problem can be described using a problem like sexually transmitted diseases *STD* diagnosis in a hospital. The result from this diagnosis can only have two classes – *sick and healthy patients*. Also, it will have each point belonging to a particular patient – *data points are measurements*. During classification of the patients, there is tendency of making error. In a bid to err on the side of caution then it would be better to mistakenly classify a healthy patient as sick rather than classifying a sick patient as healthy. To get this done, the threshold *classification line* will be moved further into the healthy patients' class. Some examples of classifiers are linear classifiers, ensemble classifiers, and nearest neighbor classifiers. In the nearest neighbor classifier, the training-data is used as the classifier. For example, when given a data point y , to classify the point, we simply find the training-data point that is closest to y and assign the class label of that training data point. In the case of ensemble classifiers, many weak classifiers are trained and then, the results from these trainings are combined to form a *majority vote* with an assumption that the classifiers are stochastically independent. The training error provides insights on the behavior of classifiers on training data [17].

A loss function is used to quantify mistakes. It maps the set of class labels times the set of class labels to a non-negative number. And a classifier is a function that carves up the feature space into regions and takes as input a point in the feature space. A classifier could be considered to be weak if almost every data point would be an error – weak classifier = error rate slightly above 50%. [17].

Table 3 – Results from Louis Dorard's Comparison

	Amazon	BigML	PredicSis	Google
Accuracy (AUC)	0.862	0.790	0.858	0.743
Training Time (S)	135	5	17	76
Predictions Time (S)	188	1	5	369

Amazon ML	Most Accurate
BigML	Fastest
PredicSis	Best trade-off
Google	Last

2.2 Prior Research Comparing Machine Learning Services

This section will present two prior researches by *Louis Dorard* [14] and *Nick Wilson* [20]. Both were based on the comparison of existing and related ML services.

Table 4 – Results from Weka Experiment

S/N	Weka Data Preparation Merit
1.	It has lots of features which makes it very powerful with in-built data processing tools.
2.	It has a command line interface, GUI, and a Java API.
3.	Provides some support for data streaming with Massive Online Analysis (MOA).
S/N	Weka Data Preparation Demerit
1.	Maximum dataset size greatly depends on the chosen algorithm, computer's memory, JVM heap size used for Weka, and data properties.
2.	It has limited options for controlling Weka from languages other than Java.
3.	It is not cloud-based which means it must be downloaded and installed as an application.
S/N	Weka Model Creation And Usage Merit
1.	Modes can be shared with anyone that has Weka installed and it provides wide range of models.
2.	It can be created from the GUI without writing code and reports cross-validation scores.
S/N	Weka Model Creation And Usage Demerit
1.	It does not support automatic optimization (model parameters might need to be tweaked).
2.	Decision tree visualization is fine for small models but becomes messy for moderate-sized models.
3.	It needs more memory usage especially during training time.
S/N	Weka Prediction Making Merit
1.	Predictions are very fast because they are done locally.
S/N	Weka Prediction Making Demerit
1.	Predictions are done by writing code.

2.2.1 Comparing Machine Learning APIs Performance

This research was done by *Louis Dorard* [14] who is the author of *Bootstrapping ML*, the first guide to Predictive APIs, and a co-founder of *PAPIS.io*, the world's first conference on Predictive APIs and Predictive Apps. His goal is to help people use new ML technologies to make their apps and businesses smarter. In his research, he did not present the details on how he carried out his experiments on the four ML services; he simply made a summary and presented his results. He compared BigML, Google Prediction API, PredicSis, and Amazon ML services using the same dataset. The dataset he used has 150,000 instances of previous reaction of defaulters and none-defaulters of loans. The problem he tackled was the *Kaggle credit challenge*, which is a binary classification challenge where the input-output values that corresponds to individual applicants who later either default or did not default was used to draw predictions for assigning credit scores based on the predicted probability of default. This is a challenge most financial institutions are likely to face when giving out loan and his goal was to use ML to predict if a new person applying for loan will default after the loan application has been granted. The comparison was based on time it took to train, time it took to predict, and accuracy of predictions. In

Table 3, his result shows that PredicSis performed best, being second fastest and second most accurate. Google prediction API happens to be the slowest and less accurate of all. Amazon ML was most accurate, although it was at the expense of the time to train and predict. BigML was the fastest in training and prediction, but was not very accurate. He performed his experiments using the free tier services.

2.2.2 Machine Learning Throwdown

In this research, *Nick Wilson* [20] compared BigML API, Google Prediction API, Prior Knowledge, and Waikato Environment for Knowledge Analysis – *WEKA*. His research was focused on data preparation *describing how data is being handled by each ML service*, ML model creation and usage *describing the*

ease of use, creation, integration, and robustness of data, and finally, prediction creation *describing the accuracy, easiness, and integration of creating predictions*. His research is more detailed as compared to the first research. To further simplify his research, I decided to put most of his results in tables.

Table 5 – Results from Google Prediction API Experiment

S/N	Google Prediction API Data Preparation Merit
1.	It has good integration with other Google services and libraries for various programming languages.
2.	It automatically detects data type, supports updating a dataset, and or a model.
3.	It supports uploading data either by the website, command line tool, and API.
S/N	Google Prediction API Data Preparation Demerit
1.	It has poor support for data with missing values and limited data training up to 250 MB.
2.	It is a bit strict about data format with no header row allowed, objective column must be the first, etc.
3.	Libraries are very thin wrappers on top of the API.
4.	It is somewhat strict about data format (no header row allowed etc.).
S/N	Google Prediction API Model Creation And Usage Merit
1.	It supports automatic optimization. This means; no need to tweak parameters.
2.	It uses arbitrary string data (E.g. “an email subject”).
3.	It provides report of basic cross-validation scores that indicates how well it can make predictions.
S/N	Google Prediction API Model Creation And Usage Demerit
1.	It only support traditional supervised ML and offers black box model with very slow model creation.
2.	It provides poor support for missing data and does not allow models to be downloaded for offline usage.
3.	Developers must write code to create a model.
S/N	Google Prediction API Prediction Making Merit
1.	Individual predictions via API calls are very fast and predictions are accurate.
S/N	Google Prediction API Prediction Making Demerit
1.	Predictions are mostly done by writing code and it supports only one prediction per API call, no batch predictions.

2.2.2.1 Waikato Environment for Knowledge Analysis

The Waikato Environment for Knowledge Analysis simply known as *Weka* is a powerful application that is equipped with suite of algorithms and collection of visualization tools for analyzing data and model predictions. It is a standalone open source GLP Java application that can be installed and ran on a computer. Weka is free software that is developed at the University of Waikato, New Zealand. The software is licensed under the GNU General Public License. The original version of Weka was a non-Java Tcl/ Tk front-end modeling algorithms with data processing utilities that were design in C. The main objective of designing this software was for the purpose of analyzing data from agricultural domains but newer versions *Java-based Weka 3* has been diversified to analyze data in different application areas. [3][7][9]. In addition to the advantages of Weka as described in

Table 4, Weka processes some collections of data pre-processing techniques, it is readily available for free under the GNU General Public License, it is portable *runs on almost any computing platform*, it possess collections of modeling techniques, and it supports wide range of algorithms, exposing wide variety of parameters of each algorithm. Unlike the other three ML services which are cloud-based, Weka requires vast ML expertise because developers will have to select the right algorithms, pre-process the data, and tune the parameters [3][7][9]. The predictions techniques in Weka is based on assumption of availability of data as relation, describing each data point by a fixed number of attributes. It provides access to SQL databases using Java Database Connectivity and the result returned by a database query can be processed by Weka. But to handle multi-relational data mining, we need separate software that can convert a collection of linked database tables into a single database table that is suitable for processing using Weka. One general disadvantage of algorithms provided by Weka is that, it cannot perform sequence modeling [2].

Table 6 – Results from Prior Knowledge Experiment

S/N	Prior Knowledge Data Preparation Merit
1.	API supports uploading rows of data as individual or in bulks as JSON data.
2.	It is fairly easy to get started and supports uploading data incrementally.
S/N	Prior Knowledge Data Preparation Demerit
1.	It is picky about data format and does not detect data type automatically.
2.	Data Training is limited to 10, 000 rows and 300 columns by default – equivalent to 15MB CSV.
3.	Developers must write code to properly format data.
S/N	Prior Knowledge Model Creation And Usage Merit
1.	It supports automatic optimization. This means; no need to tweak parameters.
2.	It supports reasoning about multiple fields, performs well with missing data, and reveals data structure.
S/N	Prior Knowledge Model Creation And Usage Demerit
1.	It does not allow models to be downloaded for offline usage.
2.	It only has support for one type of model and the model creation is somewhat slow.
3.	Developers must write code to create/ explore a model.
4.	It does not support cross-validation scores which indicate how well predictions are made.
S/N	Prior Knowledge Prediction Making Merit
1.	Predictions can be done in batches.
2.	Information indicating how confident it is about the prediction it makes are provided.
S/N	Prior Knowledge Prediction Making Demerit
1.	Predictions are done by writing code.

2.2.2.2 Prior Knowledge

Prior Knowledge provides the support to help build intelligent applications and refers to all information about the problem available in addition to the training data. Simply put, it tries to make sense out of data. From the name prior knowledge, we can easily link it with ML. Basically, in every real world applications, certain amount of information on the problem is often known beforehand the same way we need known outcomes to make predictions for ML solutions. The Veritable API provided by prior knowledge gives access to predictive database and is currently in public beta which makes it easy for developers to get started, but it does not allow developers to work with data on the web interface. It also provides additional operations that allow developers to explore similarities between different columns and rows in datasets. A rich definition, attempting to differentiate knowledge from ordinary information states

that, Knowledge is what gives us the ability to extract new information from existing information. This definition gives an overview of the feature of *knowledge-based systems (KBS)*. KBS has a distinct capability to apply limited information that is physically stored in the knowledge base on an unlimited state space [12].

Table 7 – Results from BigML Experiment

S/N	BigML Data Preparation Merit
1.	It can handle 64GB of data and allows developers to get started easily.
2.	It is good at parsing poorly-formatted data and allows uploading data via API or website.
3.	It supports remote sources like Amazon S3 through URL and possesses libraries for popular languages.
4.	It can auto detect data types even if developers do not specify them.
S/N	BigML Data Preparation Demerit
1.	Adding data to existing dataset/ model is not supported.
S/N	BigML Model Creation And Usage Merit
1.	It provides a decision tree with a property known as white box. Can be downloaded for offline use or explored on the website.
2.	It support creating model on the website without coding and also supports fast model creation.
3.	It supports automatic optimization. This means; no need to tweak parameters.
4.	It does not time out or crash on unique string fields.
S/N	BigML Model Creation And Usage Demerit
1.	It only supports traditional supervised ML and only one model type.
2.	It does not report cross-validation scores that indicate how well the platform makes prediction.
3	It does not make use of its text data type when learning a model.
S/N	BigML Prediction Making Merit
1.	Predictions are faster when done offline and predictions can be made on web without coding.
2.	Predictions can be done offline from models that were stored locally.
S/N	BigML Prediction Making Demerit
1.	Individual predictions via API are slow, and it supports one prediction per API call, no batch prediction.

2.2.2.3 Google Prediction API

Detailed description of the functionalities of Google prediction API will be considered in chapter three but in Table 5, the result from the prior research is presented. Google Prediction API has a special feature of integrating with other services provided by Google and it detects data types automatically.

2.2.2.4 BigML API

BigML concepts focus on developers with less expertise by making ML available to everyone irrespective of their programming skills. It provides one type of model called *decision tree*. This model is powerful in such a way that it allows developers to look inside the model *white box model* to understand what it learned from the data and how predictions about unseen data are done using this information. The web interface or API can be used to create and explore predictive models. This allows any developer to do everything on the web from uploading data, to creating models, exploring models, and making predictions without writing a single line of code [20]. With BigML, developing ML solutions is quite easy. All the detailed required to add data-driven decisions and predictive power into our ML solution is being taken care of by BigML [25].

Chapter 3: Experiments and Results

Based on recent research, there is no formal research directly tackling the topic that is being considered in this thesis. However there exist several researches that are in some extent related to this topic, two of which have been described in chapter 2. Conclusion from the first research was summarized in the author's research description and in Table 3. In the case of the second research, which is more detailed, the author came up with conclusions that BigML API came out first in the data preparation aspect, Google Prediction API is much more convenient for working with text, Prior Knowledge could help discover complex relationship in dataset, Weka has power to tweak model parameters, and Big ML provides the opportunity to have a view into data based on white box model property [20].

Weka has proved itself to be a useful and even essential tool in the analysis of real world data sets. It reduces the level of complexity involved in getting real world data into a variety of ML schemes and evaluating the output of those schemes. It has also provided a flexible aid for ML research and a tool for introducing people to ML in an educational environment – Stephen R. Garner [13].

Now, let us delve into the functionalities of the three ML services in-question *Windows Azure ML Studio*, *Amazon ML*, and *Google Prediction API* using the power of ML to extract patterns from the dataset and then use the extracted values to build intelligent system that can provide a cost-effective approach where manual programming will not suffice. For each service, we are going to present detailed description of the configuration process, how data is uploaded and processed, model creation and training, model scoring and evaluation, and finally, run predictions. Using the free tier subscriptions on all the services, the experiments will try to show how simple and easy it is to configure and build ML solutions on each service using the services' APIs. The data source and data formatting will not be included in the experiments because chapter 2 already discussed these. Also, the data upload sections in the experiments will be based on the assumptions that the dataset is already saved locally on our computer.

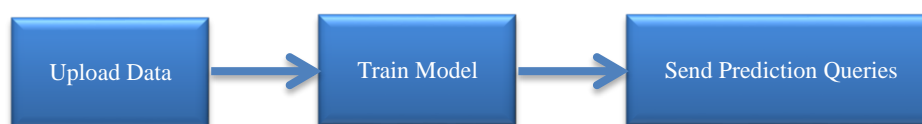


Figure 5 – Google Prediction API Workflow

3.1 Google Prediction API

Google Inc. headquartered in Googleplex, Mountain View, California, USA, is a multinational technology company that specializes in internet-related products and services like *cloud computing*, *software*, and *etc.* With the company's rapid growth, it later came up with more internet-related products and services like *Google Search*, *Google Docs*, *Gmail*, *Google Drive*, and *etc.*[29][30]. The company later diversified into communications hardware, Android mobile operating system, and web browser. In the bid of diversifying into the communication hardware, it partners with major electronics manufacturers to produce high-quality in-expensive Nexus devices. In December 2013, google.com was listed as the most visited website [31]. The company developed a special website known as the *Google Developers Console* for software development, technical resources, and APIs. This website provides developers with useful documentation

on using Google APIs and other tools for developing solutions. Its APIs contains sets of application programming interfaces for communicating with Google Services and easy integration with other services [32].

Among the APIs Google provides is the *Google prediction API*. This service is a cloud-based ML tool created and released by Google in 2011 with the aim to make ML model design easily accessible and deployed for so many applications in a wide variety of fields via a RESTful interface. The API is fairly simple to use, it requires some basic calls and it will provide pattern-matching and ML capabilities. After it learns from the training data, it can predict a numerical value or choose a category that describes a new piece of data [32]. It makes use of cloud infrastructure to provide ML tools that are common among other Google products and APIs, but requires multiple pieces to set up. Tools for uploading data are separate from the tools for creating models and making predictions. One important feature in Google prediction API is that it can learn models from arbitrary text data; this can be very useful especially when detecting spam or sentiment analysis. Google Cloud Platform does not provide user friendly interface but its API allows developers to programmatically access resources that are exposed by services within applications. Google offers free and standard tier subscriptions with a 300 dollars credit which can be used for up to sixty days.

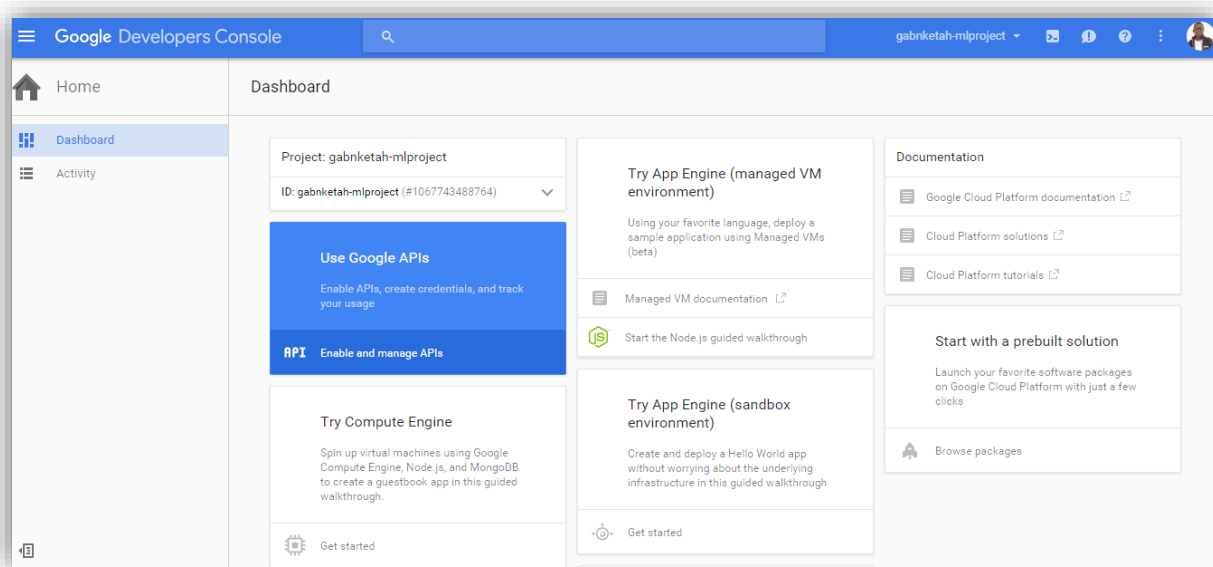
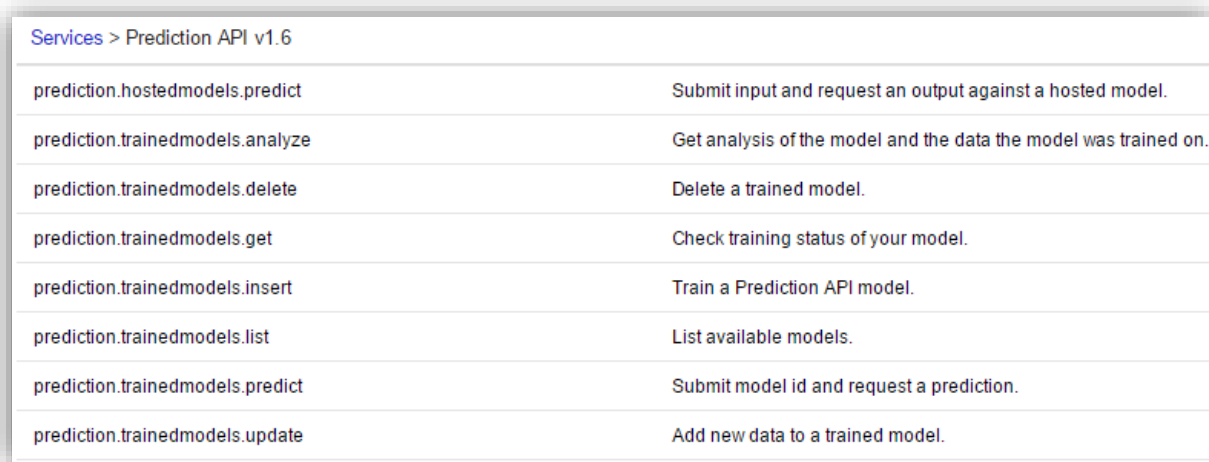


Figure 6 – Google Developers Console Project Dashboard

3.1.1 Configuring Google Prediction API

Google Prediction API provides ML tools on top of Google's Cloud Infrastructure. Configuring this service is fairly easy, all that is required is a valid credit card, valid Google account, and Google Cloud Platform Console project with both the Google Prediction API and Google Cloud Storage API activated. Equipped with a valid Google account and a credit card, visit <https://console.cloud.google.com/>, there, click *sign up for free trial* on the top-right corner on the window and in less than a minute, a new window for activating billing information pops up, and once we click *Enable*, Google automatically creates a new project dashboard for us as shown in Figure 6, and also send a welcome message – *Thanks for signing up for the 60-days free trial. We have given you \$300 in the free trial credit to spend. If you run out of credit, do not worry, you will not be billed until you give your permission.* To begin exploring this powerful prediction tool, we click on my *first project option* and select *create a new project*. Once we input a project name, Google generates a project ID. Enabling the prediction API on the cloud platform is quite straight-forward. Once the API is enabled, Google sends a confirmation message – *this API is enabled,*

but you cannot use it in your project until you create credentials. Click, “Go to Credentials” to do this now – strongly recommended.



Services > Prediction API v1.6	
<code>prediction.hostedmodels.predict</code>	Submit input and request an output against a hosted model.
<code>prediction.trainedmodels.analyze</code>	Get analysis of the model and the data the model was trained on.
<code>prediction.trainedmodels.delete</code>	Delete a trained model.
<code>prediction.trainedmodels.get</code>	Check training status of your model.
<code>prediction.trainedmodels.insert</code>	Train a Prediction API model.
<code>prediction.trainedmodels.list</code>	List available models.
<code>prediction.trainedmodels.predict</code>	Submit model id and request a prediction.
<code>prediction.trainedmodels.update</code>	Add new data to a trained model.

Figure 7 – List of Methods Google API Provides

3.1.1.1 Step 1: Upload Dataset

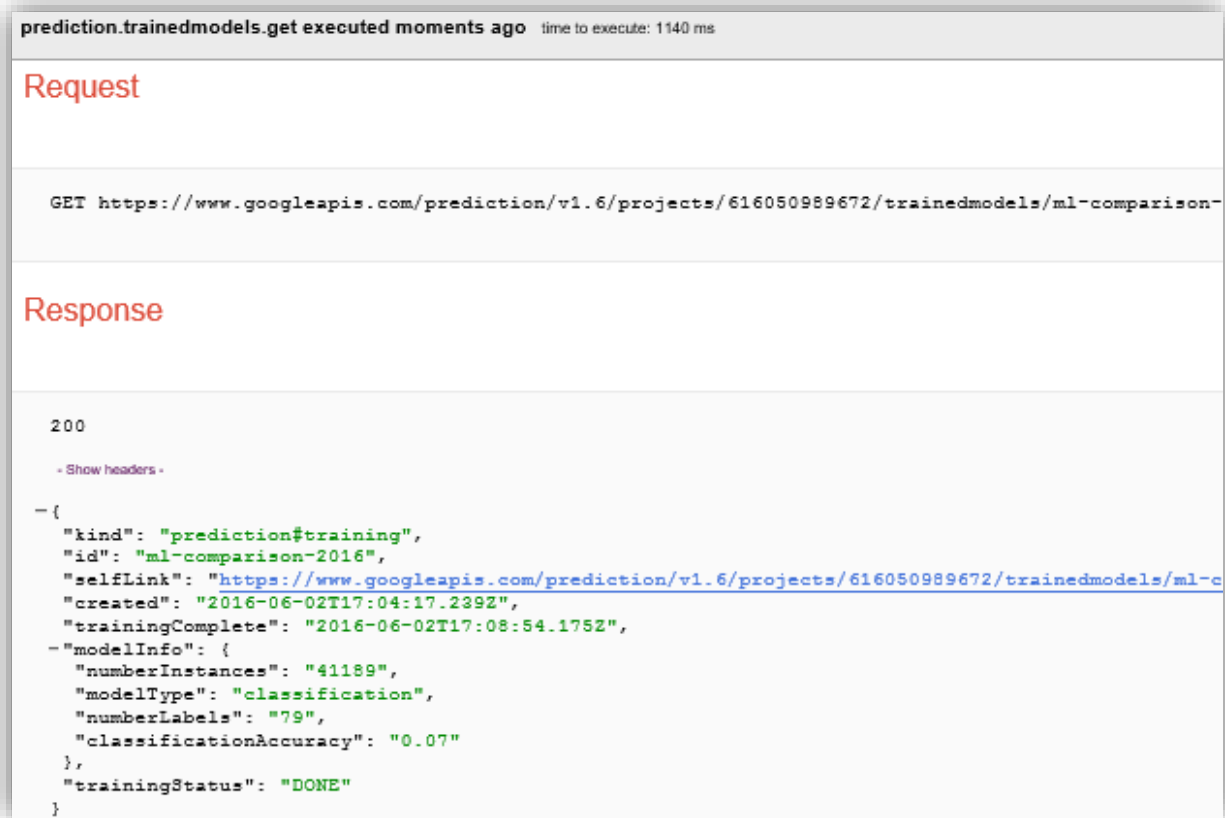
Once we are done setting up the billing information and project dashboard, next is to upload data which is the first step shown in the experiment workflow in Figure 5. One good thing at this point is that we do not need to configure the storage; Google Cloud Platform integrates easily with Google Storage service. To do this, click on the icon directly on top of the *Home* option in Figure 6, scroll down the options and click on *Storage*, there we create a bucket or use an existing bucket. The Google Cloud Storage lets us store unstructured objects in containers called buckets. We can serve static data directly from Cloud Storage, or we can use it to store data for other Google Cloud Platform services [32]. We need to input a bucket name, storage class, and location to complete the bucket creation process, and the dataset upload is done with few clicks. Once the dataset has been successfully uploaded, Google takes charge from there and automatically normalizes the values; detect data types, select features, and etc. This can be described as a *black-box* because there are no visualization tools that allow developers to know what is happening behind the scene.

3.1.1.2 Step 2: Train Model

From the previous section, we can see that Google takes charge once data has been uploaded on its API platform. It further takes charge of splitting the dataset into training and testing sets, one for training the model while the other to evaluate the model. It does this automatically and does not allow developers to determine the ratio of the splitting, but it allows developers to make use of a pre-trained model instead of using a new dataset. The model training process is fast and asynchronous, and the model can only be available for use after the training is completed. To start the training process, navigate to the Google APIs Explorer at developers.google.com/apis-explorer/#p, there Google provides multiple methods that we can call as shown in Figure 7. First, select the `prediction.trainedmodels.insert` method, add the project ID *the project associated with the model – string*, and include the Model ID *unique name for predictive model – string*, and Data Storage Location which are the two configuration elements needed at this stage. Once we have input the configuration elements and click on *Execute*, It takes some milliseconds and Google returns a result. In the result, Google provides a self-link path that contains the unique API key, data storage location, and model ID. This shows that Google has been able to get hold of the dataset in its storage and ready to perform the training process.

Next, we go back to the list of methods on the API explorer in Figure 7 and select the `prediction.trainedmodels.get` method. This method takes the project ID and model ID as input parameters

and returns status report of the trained model as shown in Figure 8. From this report, we can see the status of the model training as *DONE*, and Google also provides some other useful information like model type *classification*, number of instances, classification accuracy, and etc. How Google detects the model type is commendable and it also has a unique feature that allows data to be added on the go without having to start the whole model training phase from the very beginning. This feature is really good for long term spanned systems because the model can be easily adapt to new data and conditions.



```
prediction.trainedmodels.get executed moments ago time to execute: 1140 ms

Request

GET https://www.googleapis.com/prediction/v1.6/projects/616050989672/trainedmodels/ml-comparison-

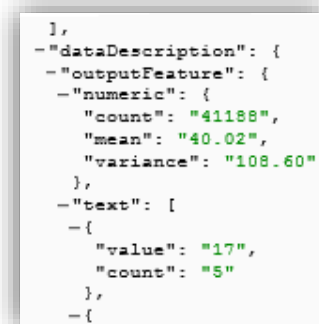
Response

200

- Show headers -

- {
  "kind": "prediction#training",
  "id": "ml-comparison-2016",
  "selfLink": "https://www.googleapis.com/prediction/v1.6/projects/616050989672/trainedmodels/ml-c",
  "created": "2016-06-02T17:04:17.239Z",
  "trainingComplete": "2016-06-02T17:08:54.175Z",
  "modelInfo": {
    "numberInstances": "41189",
    "modelType": "classification",
    "numberLabels": "79",
    "classificationAccuracy": "0.07"
  },
  "trainingStatus": "DONE"
}
```

Figure 8 – Google Trainmodels.Get Method Results



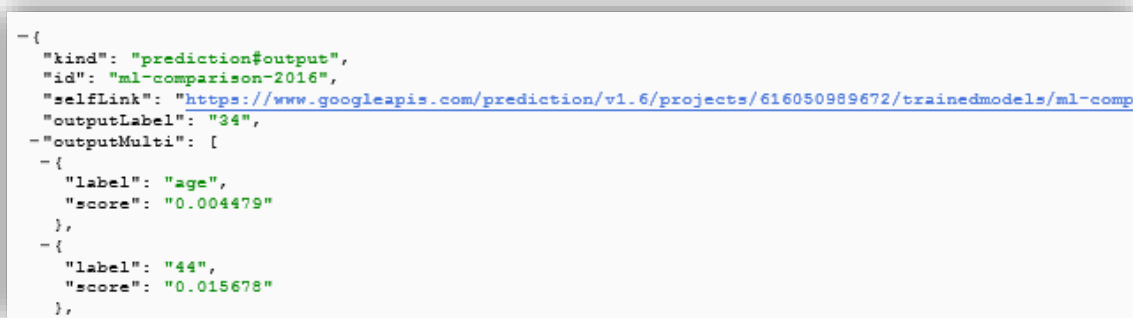
```
],
- "dataDescription": {
  - "outputFeature": {
    - "numeric": {
      "count": "41188",
      "mean": "40.02",
      "variance": "108.60"
    },
    - "text": [
      - {
        "value": "17",
        "count": "5"
      },
      - {
```

Figure 9 – Google Trainedmodels.Analyze Method Result

3.1.1.3 Step 3: Send Prediction Queries

At this stage, we have been able to successfully build an ML model on the Google API. But before we go to the next step which is to make predictions, it is good to utilize the data analysis method which Google provides. Though this method does not provide visualization, it does provide some statistical analysis of

the data like the confusion matrix, mean, variance, and etc. as shown in Figure 9. Google allows predictions to be generated using an API call that returns values – *allows only one prediction per API call, does not support batch prediction* that returns values. To make predictions against the trained model, go back to the list of methods on the explorer and select the *prediction.trainedmodels.predict* method. This method takes the columns in the dataset as input parameters excluding the target value and returns predictions in numeric as shown in Figure 10.



```

- {
  "kind": "prediction#output",
  "id": "ml-comparison-2016",
  "selfLink": "https://www.googleapis.com/prediction/v1.6/projects/616050989672/trainedmodels/ml-compari
  "outputLabel": "34",
  "outputMulti": [
    - {
      "label": "age",
      "score": "0.004479"
    },
    - {
      "label": "44",
      "score": "0.015678"
    },
  ],
}

```

Figure 10 – Google Trainmodels.Predict Method Results

3.2 Windows Azure Machine Learning Studio

Microsoft Corporation – *headquartered in Redmond, Washington, USA*, is a multinational technology company that manufactures, develops, supports, licenses, and sells computer electronics, personal computers, and computer software like *Microsoft Windows, Office Suite, Internet Explorer, and etc.* The company has also gained recognition over the years with huge experience and expertise in developing, building, and managing datacenters for over 25 years and has since then achieved providing its users with core infrastructure and foundational technologies for over 200 online services, including Microsoft Azure ML Studio, OneDrive, Skype, Xbox Live, Office 365, MSN, Bing among others [15][24]. Microsoft has become one of the world's most valuable companies and it has over the years become the world's largest software maker by revenue [23][24]. In recent research, Azure is said to be available in about 141 countries, and supports 10 languages and 19 currencies, all backed by Microsoft \$15 billion investment in global datacenter infrastructure. Its state of the art infrastructures which comprise several thousands of servers, edge computing nodes, content distribution networks, and fiber optics networks has been part of its success story in the fast growing information technology industry [3]. Among all the powerful tools from Microsoft, we are going to focus on Microsoft Azure ML Studio. So far, this tool has been deployed as datacenters in 19 regions across the globe *Singapore, Amsterdam, Melbourne, Sao Paulo, and etc.* [5]. It was officially announced in October 2008 and later released as *Windows Azure* in February 2010 [10] with the goal of providing developers of any expertise level to easily deploy it. This means that developers can use this powerful cloud-based infrastructure to quickly creates, share, test, fix, retrain, and deploy powerful ML experiments through a global network of Microsoft-managed datacenters without purchasing or provisioning our own hardware. Before ML was invented, analytical tasks are performed using Python, R, and Hadoop among other programming languages. However, with Microsoft Azure, the process of analytical tasks has been eased with a more user friendly and easy to use platform. This platform brings all the programming and statistical concepts together, from statistical algorithms, to analytical algorithms among other highly recognized industry standard algorithms. The concept of drag-and-drop algorithms that Azure provides is mind blowing. The drag and drop interface could be used to connect pre-programmed components of data science pipeline together. The primary predictive analytics algorithms currently used in Azure ML are: classification, regression, and clustering [15].

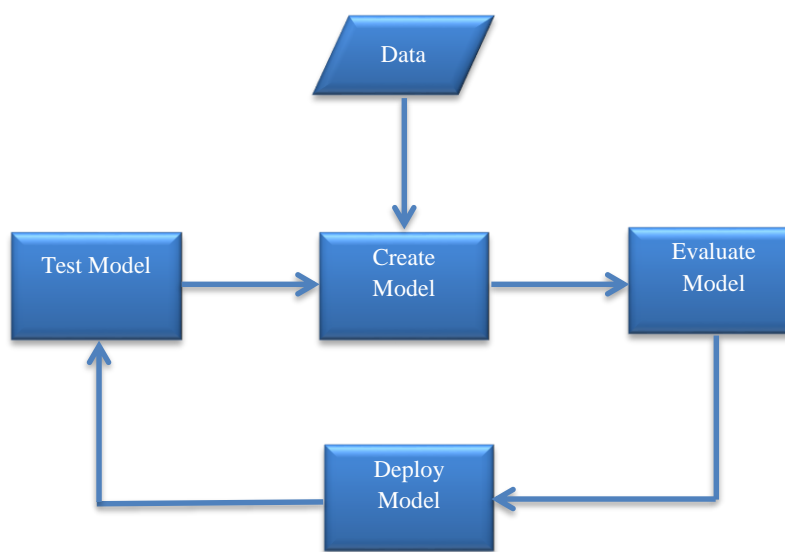


Figure 11 – Azure Machine Learning Work-flow

3.2.1 Configuring Azure Machine Learning Studio

Azure ML Studio is the key interactive analytics workbench that is used for creating ML solutions in the Microsoft Azure cloud. It is a powerful drag-and-drop visual designer interface tool that brings together powerful algorithms, statistical analytic tools, and ML experience into a simple cloud service. Its workspace is equipped with pre-built templates and features that are readily available for developing and testing predictive analytic solutions in the cloud. Creating an experiment in its workspace requires either a free or standard tier subscription with a valid Microsoft account. Each of the tiers provides different level of pricing, service features, and capabilities [15][19]. Equipped with any of the subscription tiers, valid Microsoft account, valid credit card, and a modern web browser, we begin the experiment by visiting the Windows Azure official portal at <https://manage.windowsazure.com>, there we click *Machine Learning* on the left navigation bar, click on the *New* icon *bottom left of the page*, fill in the require fields. With a final click on *Create an ML workspace*, we are set to start developing our ML experiment. During the registration process, Microsoft will require a workspace name, workspace owner's Microsoft account, location, and storage account *we can either select option to create a new storage account or use an existing storage account*. The workspace is designed in a way that we just drag-and-drop searchable models from the left pane into the workspace. These models can be dropped anywhere on the workspace and connecting the models is done by simply drawing lines from the output port of a model to the input port of the connecting model. The left navigation pane provides the list of all Azure ML modules which are searchable and can be used for predictive analytics model creation. While the center pane provides the visual designer where models are assembled in a flow-chart pattern by simply dragging and dropping, and then the right pane provides visualization of selected module's properties. Once we have created our workspace, next is to create an experiment. This is done by clicking on the + *New* option on the bottom left of the dashboard as shown in **Error! Reference source not found.**, and then we can name the experiment.

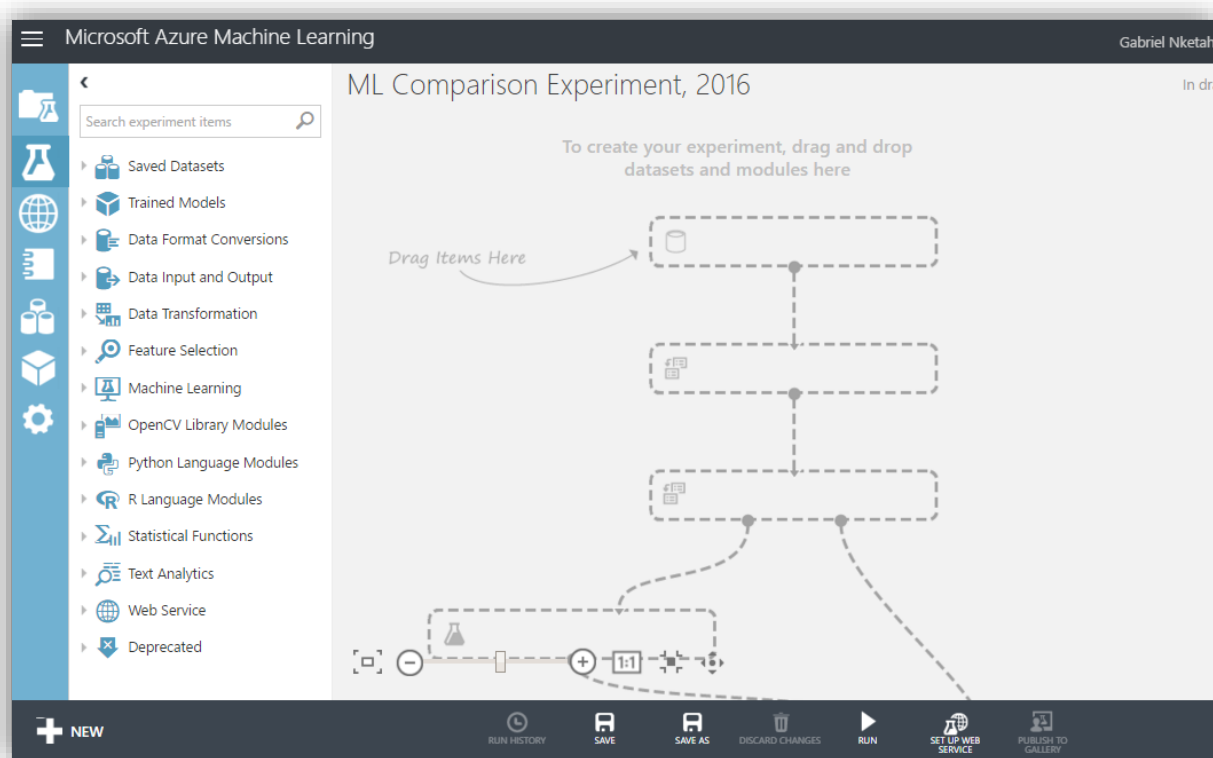


Figure 12 – Blanc Experiment on Azure Workspace

A breakthrough in ML would be worth ten Microsoft, and the new Azure ML services takes on the ambitious challenge with a truly differentiated cloud-based offering that allows easy access to the tool and processing workflow that today's data scientists need to be quickly successful. Armed with only a strong hypothesis, a few large data sets, a valid credit card, and a browser, today's ML entrepreneurs are learning how to mine for gold inside many of today's big data warehouses – Bill Gates [15]. Azure brings so many services to your fingertips in a reliable, secure, and environmentally sustainable way. You can do immense things with Azure, such as create a single VM with 32TB of storage driving more than 50, 000 IOPS or utilize hundreds of thousands of CPU cores to solve your most difficult computational problems [5].

3.2.1.1 Step 1: Upload Dataset

To upload dataset, navigate to the left side-pane and select *Upload Data*, the system will display options to upload as shown in **Error! Reference source not found.**, click *From File* and navigate to the file on the computer. In less than 40 seconds, our data is ready for the experiment, drag-and-drop it on the workstation and use the visualization tool to have an insight about the data. To use the visualization tool, right-click on the connector display option below the dataset module as shown in Figure 15 – *small circle that is at the bottom of each model*. Two models are connected by drawing a line from the output connector of one model to the input connector of the other model. After clicking on the connector, select *visualize data* option, then the system will present some statistical and visualization features for quick analysis of the data. There, the visualization tool presents the number of rows, number columns, and a click on each column will provide statistical information and data-insight of the particular column. Figure 14 shows the histogram visualization of the age field. Based on the statistical analysis of the data using the same visualization tool, there are zero missing values and this depicts that the data is almost accurate. If the missing values are substantial, it will be advisable to update the records in the dataset or just drop the dataset completely in order to get most accurate results from the experiment. Most importantly, the visualization features gives the ability to easily infer key data elements, combinations, associations, and

patterns which helps in creating a powerful predictive analytics solution [15][19]. The goal at this stage is to upload data and have an insight of the data to confirm it is good enough for the experiment.

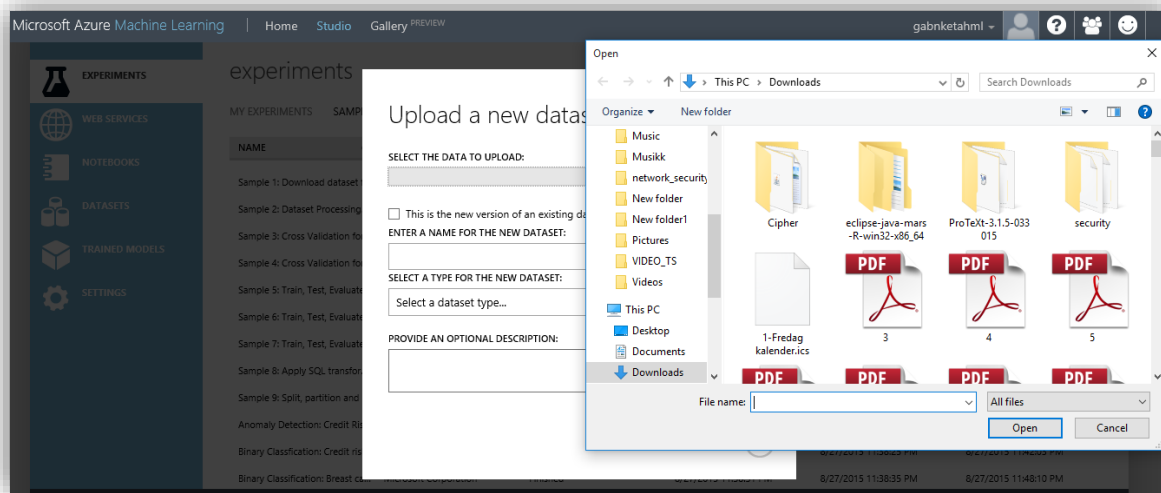


Figure 13 – Azure Data-upload Window

3.2.1.2 Step 2: Split Dataset

Unlike Google Prediction API that does not allow developers to determine how dataset is split into training and testing sets, Azure supports splitting dataset into two logical sets based on ratio of choice. To do this, expand the data transformation modules on the left pane, drag-and-drop the *Split Data Module* and then, connect the module to the *Data Module* by drawing a line from the output of the *Dataset Module* into the input of the *Split Data Module*. On the right pane of the Azure Studio, there we set the fraction of the rows to 0.7 and random speed 1234 which will randomly split the data into two sets, diverting 70 percent of the data to a training area in the experiment and the remaining 30 percent will be reserved to test the new ML model for accuracy.

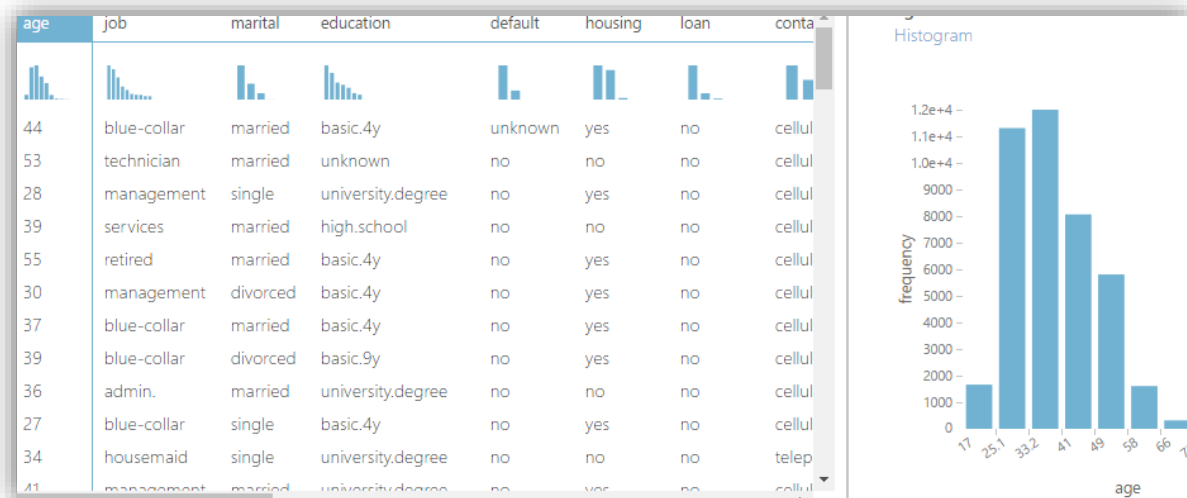


Figure 14 – Azure Data Visualization Tool: Age Field Visualization

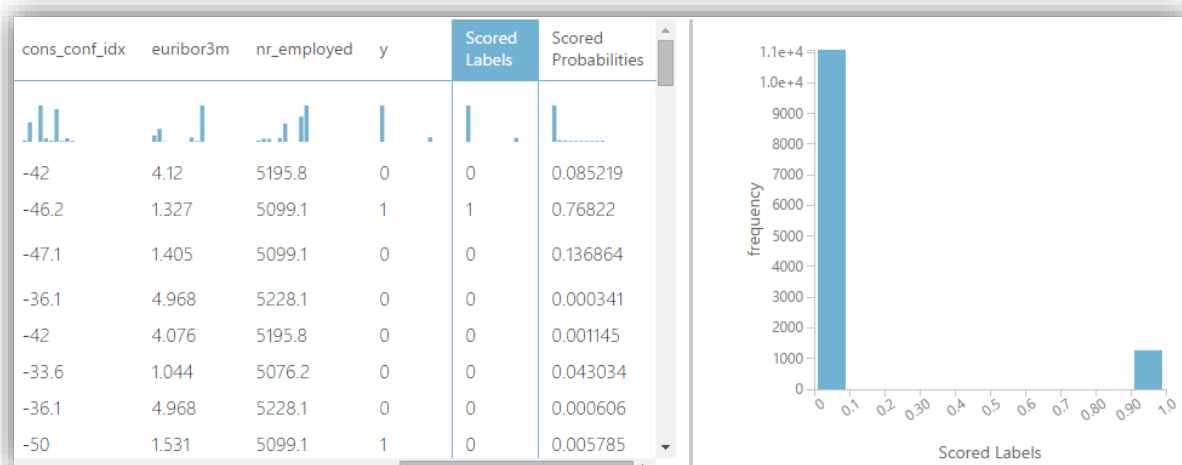


Figure 15 – Azure Data Visualization Tool: Model Scoring Result

3.2.1.3 Step 3: Train Model

Training the model here means teaching the model how to evaluate the dataset and an algorithm will be selected to perform the training task. To do this, expand the modules on the left pane, expand the *Train* submodule, drag-and-drop the *Train Model* onto the workspace, and connect the *Train Model* to the *Split Model* as shown in Figure 15. For this experiment, we are going to select the *Two-Class Boosted Decision Tree* algorithm which is one of the several classification algorithms Azure provides as shown in Figure 17. This algorithm is a method that is mostly used in ML, data mining, and statistics fields. It abstracts a decision tree as a predictive model that maps observations from a data to the target value. The logic behind the tree is based on the assumption that the leaves will represent the class labels while the branches will represent conjunctions of features that will point to those class labels with the goal to develop a model that can predict the value of the label based on the several data inputs [26]. To complete the configuration of the training process, drag the chosen algorithm from the left pane onto the workspace and configure by indicating which column in the dataset is the outcome column. Indicating the column is done by launching the column and click on the *Train Model*. Properties panes will pop-up on the right pane of the Workspace, there, the column can be selected. Once we click the *RUN* option, it means we have given Azure ML algorithm the charge to infer patterns from all the other columns in each row of the dataset.

3.2.1.4 Step 4: Evaluate Model

The Windows Azure ML official website describes scoring a model as the process of applying model to new data with the goal of generating predictions and other values. The scoring generated using the Azure ML model can include [10]:

- Cluster assignments.
- List of recommended items.
- Forecasts for time series models.
- Probability scores associated with these outputs.
- Predicted class or outcome for classification models,
- Estimates of projected demand, volume, or other numeric quantity, for regression models.

To implement the model scoring, expand the *Score Model* submodule from the left pane, drag-and-drop the *Score Model Module* onto the workspace, and connect it to the *Train Model* and *Split Data Model*. After the models have been connected, click the *RUN* option at the bottom of the page and wait while the result is being processed. A green check sign indicates beside each model as shown in Figure 15. Once all

the models have been checked, the result can be visualized. From the result in Figure 19, scored label and scored probability columns are added to the dataset. This is an indication that the chosen algorithm can make calculation for each row and provide numerical probability factor that represents the model's potential for accurately predicting each row in the dataset based on the specific values found in each the row's other columns [15].

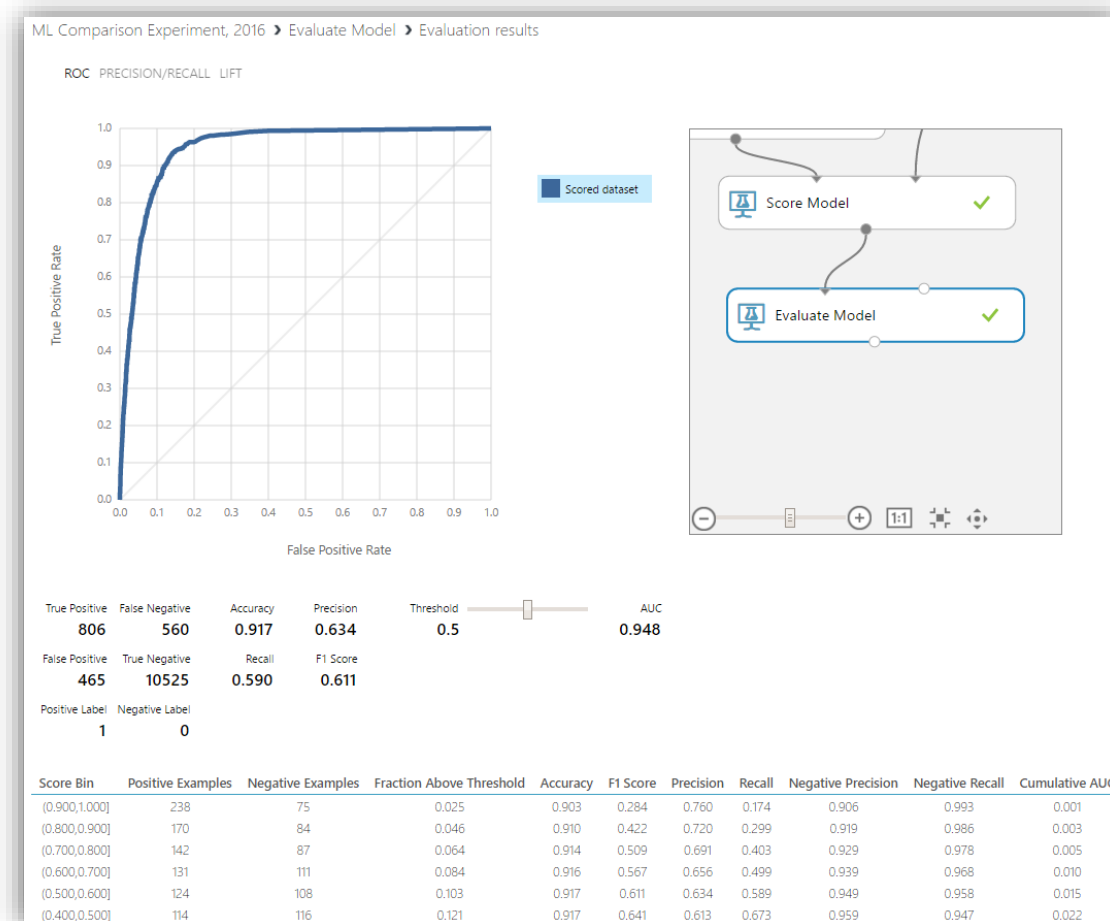


Figure 16 – Azure Model Evaluation Result

Finally, to evaluate the new model, we drag the *Evaluate Model* to the workspace and connect it to the *Score Model* then we click the *RUN* option. The powerful visualization tool allow us to see how accurate our model really is, and the time to run the evaluation was 1minutes 50seconds with a visualization tool showing as each stage is processed and marked complete with a green check mark on the right of each model. Figure 16 presents the result from the evaluation as a set of curves and metrics that shows that the trained and validated datasets are somewhat identical. Also, the blue and red lines almost exactly on-top of each other indicates a reasonably accurate prediction model. We can save the experiment by clicking on the *Save As* option at the bottom of the screen.

Warning: if data used as input when scoring a model has missing values, the missing values are used as inputs, but nulls are propagated, which usually means that the result is also a missing value [10]. Remember, the key to a great Azure ML solution is to develop iteratively, where the keys to success are to fail fast and fail; often [15].

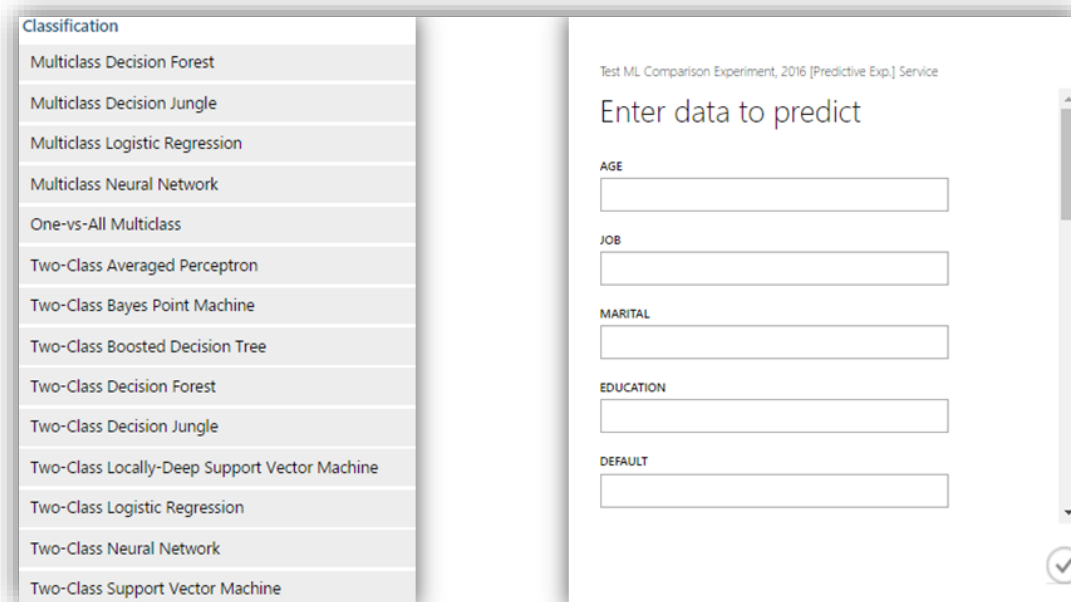


Figure 17 – Azure Classification Algorithms and Web Service Interactive Testing Window

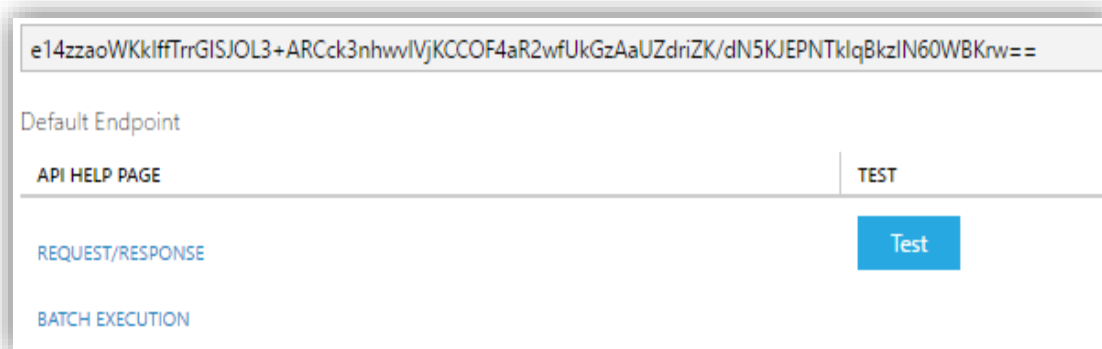


Figure 18 – Azure Web Service Dashboard

3.2.1.5 Step 5: Test Model

Before we can make predictions, we must first deploy the model as a web service; note that the option *Deploy Web Service* was not available as a valid option before this stage, meaning that, before this option can be valid, a predictive model that can produce reasonable results must have been developed. Next, select *Prepare Web Service* option and the system automatically add a web service input and output modules, and also, it saves the trained model in experiment repository of trained models which can be found under the *Trained Models* navigation bar. The web service input and output modules indicate the data flow of web service API. However, there is need to help the system infer web service input schema by excluding the label column which is done by pulling the *project column* from the left pane, drop it on the workspace, connect it to the *dataset* module and *score model* module, click on the *project module* and launch the column selector, there we select all rules, and exclude the label column. Also, there is need to help the system infer web service input schema by including only the *scored labels* column and the *scored probabilities* column. Once we click on the *Run* option and click on the *Publish Web Service* icon, a dashboard pops up after some seconds providing API help links for online predictions. At this point, the model has been deployed as a web service and ready for prediction creation. To make predictions, click

the *Test* icon on the experiment in Figure 18 and in few seconds the web service interactive testing window pops up as shown in Figure 17. There, we input the parameters in the column of the dataset excluding the label column, and once these values have been entered, a click on the check mark icon will submit the prediction request. Azure does the processing and returns result as a probability factor at the bottom-left of the dashboard. The web service dashboard also provides all necessary information needed for invoking new predictions over the web. The *Excel Download* option allows us to download the excel spreadsheet that contains the API key – *access key*, web service URL – *http endpoint address for calling the web service*, *Schema* – *http URL with information of all input and output parameters*. The *API key* is the unique identifier needed to authenticate caller when invoking the web service. The *API help page* links displays series of API usage information that describes how to call Azure web service based on individual, one or more input records. The most stunning feature of the webpage is the free sample codes provided in C#, Python, and R programming languages which provides developers with the tool to have a complete working client application that can call Azure web service as shown in Instance 2 in *Appendix*. With this feature, all we need to do when creating client testing application is to insert the API key value and populate the input data with valid values from the input database. Sample Data is a feature for web service users to get started with using your web service by downloading the Excel workbook, Sample Data will be enabled, which will make a small sample from your training dataset available to only those with the correct API key. The API key is the unique security identifier that must be passed on every web service request to authenticate callers. To disable Sample Data, go to the Configuration section of the web service dashboard and click *NO* for *Enable Sample Data* [15][19].

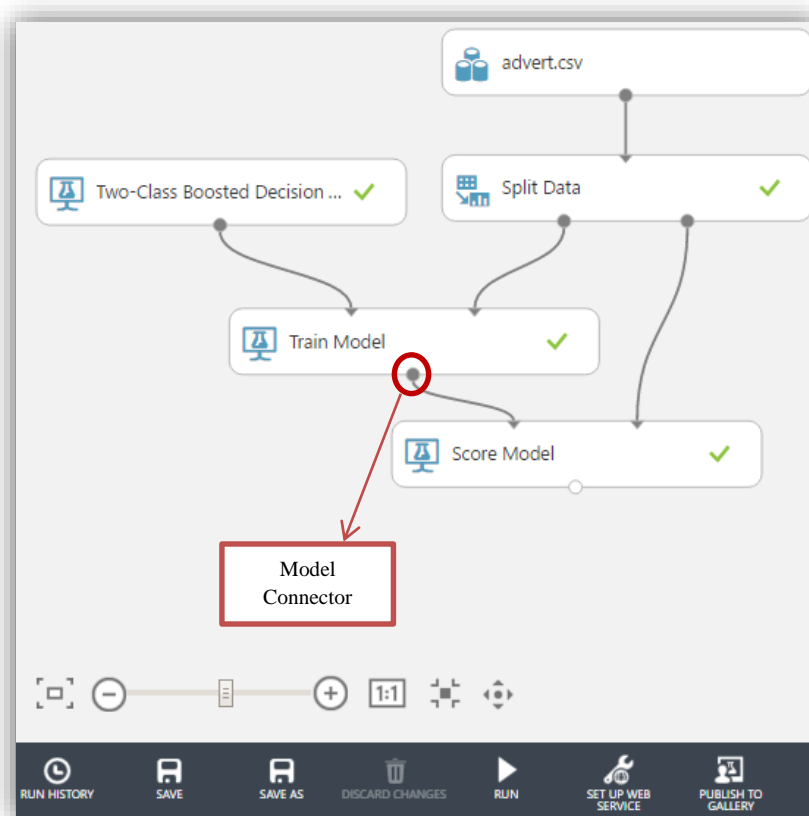


Figure 19 – Connected Models on Azure Studio

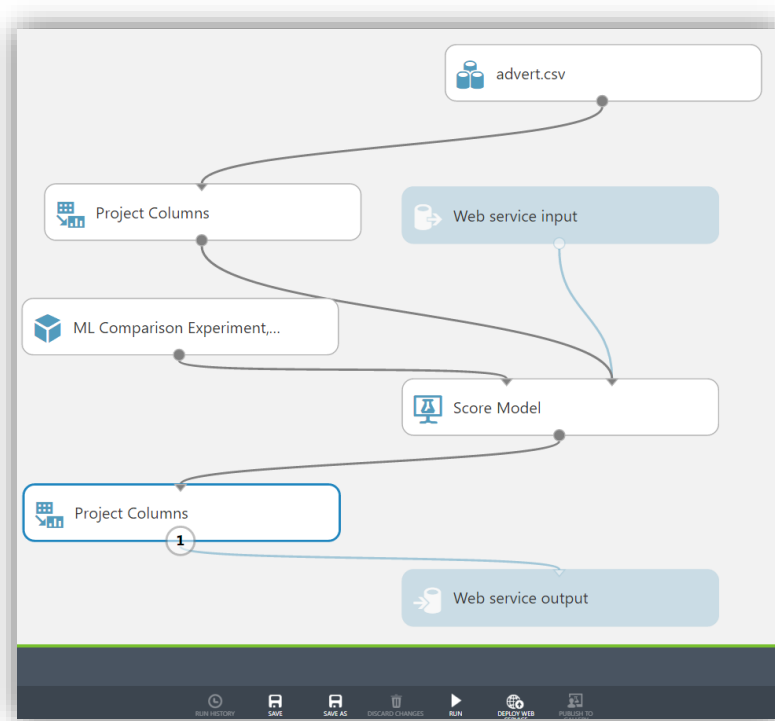


Figure 20 – Automatically Activating Web Service on Azure Studio

3.3 Amazon Machine Learning

Amazon.com, Inc. headquartered in Seattle, Washington, USA is the largest internet-based retailer in the USA with focuses on electronic commerce and cloud computing [19]. It started its operation by providing bookstores online and later, it diversified to selling software, toys, food, streaming music, downloading videos and etc. It produces popular consumer electronics like Amazon Kindle e-book readers, fire television, fire smart phones, and etc. Later, the company further diversified into providing services like Amazon Web Services – AWS among others. In 2006, AWS started offering cloud infrastructure services *IaaS* to businesses in the form of web services and recent research shows that AWS provides a highly reliable, robust, scalable, infrastructure platform in the cloud for hundreds of thousands of businesses in 190 countries around the globe [22]. Amazon ML is a robust cloud-based service developed by Amazon and equipped with powerful visualization tools and wizards that guides developers of all skill levels through the creation of ML models. It focuses mainly on a fully automated tool for supervised learning and provides access to build three types of ML solutions which includes binary classifications *predict one or two outcomes*, multi-class classifications *predict among three or more outcomes*, and regression *predict numeric value*. The binary classifications could be used to answer a simple *YES* or *NO* questions like: Is a particular product an automobile or not? Is the current user a robot or not? In multi-class classifications, it is quite similar to binary classifications except now instead of just two categories, we have three or more. Multi-class could be used to answer questions like: Is the pastor preaching standing, sitting or walking? Is this movie a romance, thriller or comedy? In regression, numeric values are computed. We can use regression to answer questions like: What will the temperature be in Oslo tomorrow? What is the likely amount to sell this car? All the section that covers Amazon ML is largely drawn from the Amazon experiment console and [11].

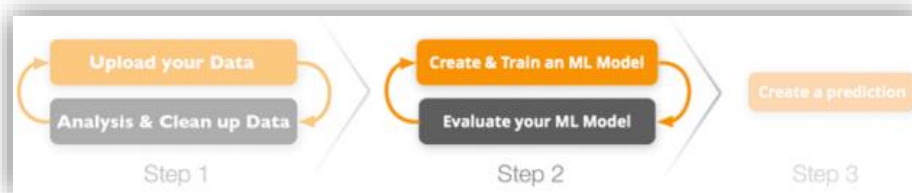


Figure 21 – Amazon Machine Learning Workflow [11]

3.3.1 Configuring Amazon Machine Learning

In the case of AWS, once we sign-up for any of the services AWS provides, we automatically get signed-up for all other services provided by AWS including Amazon ML and Amazon Simple Storage Service *simply known as S3 Bucket*. These two are the services needed to develop ML solution. To sign up, go to <http://aws.amazon.com> and select *Sign Up*. The on-screen instruction provides short and detailed interactive guide on how to sign up. After signing up, AWS provides 12 months of free tier access including use of Amazon EC2 for 750 hours per month, 5GB of storage on the S3 bucket, and more. The configuration process seems to be quite short, simple and straight-forward.

A data-source is an Amazon ML object that holds the location of your input data, the attribute names and types, the name of the target attribute, and descriptive statistics for each attribute – data-sources are limited to 1TB [11].

The screenshot displays the Amazon Data-source Console interface for a specific data source. The data source is named 'ds-cSu4CCgflYn' and is titled 'ML Comparison 2016'. The console provides the following information:

- Datasource information:** ID: ds-cSu4CCgflYn, Name: ML Comparison 2016, Creation time: 04/20/16 04:00:36, Status: Completed, Message: Not available, Input schema: View input schema.
- Target information:** Target name, Target type, Target visualization (represented by a small bar chart).
- Input data:** S3 location: s3://gabnketah/customers.csv, Data format: CSV, Data rearrangement: None.
- Processing information:** Number of records seen: 41188, Records that failed to process: 0, Log: Download log.
- Attribute information:** Number of attributes: 21, Binary: 1, Categorical: 10, Numeric: 10, Text: 0.

A dropdown menu is visible over the 'Use this datasource to' button, listing the following actions: Copy settings to create a new datasource, Create (train) an ML model, Evaluate an ML model, and Generate batch predictions.

Figure 22 – Amazon Data-source Console

3.3.1.1 Step 1: Upload Dataset

With the assumption that the .csv dataset file has been locally saved on the computer, to upload it onto the AWS platform, sign in to AWS Management Console and open the Amazon S3 bucket console at <https://console.aws.amazon.com/s3>, there Amazon presents options to either create a new bucket or choose an existing bucket where the dataset can be uploaded. Once the *Add Files* is selected, a dialog box appears where we can navigate to where the dataset is located on the computer. The entire process is fast and straight-forward *less than two minutes*. After the dataset has been successfully uploaded to the S3 bucket, the service requires the creation of a data-source that will reference the location of the dataset in the S3 bucket.

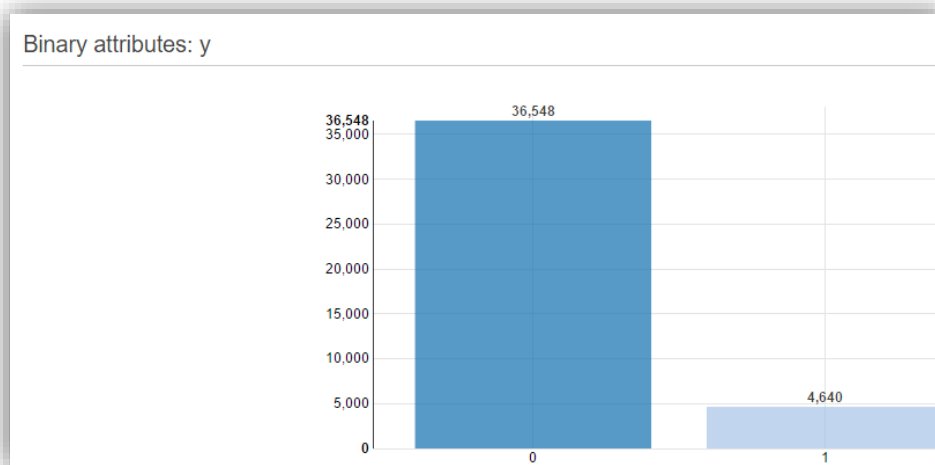


Figure 23 – Amazon Data Insight: Target/ Label Values' Binary Attributes

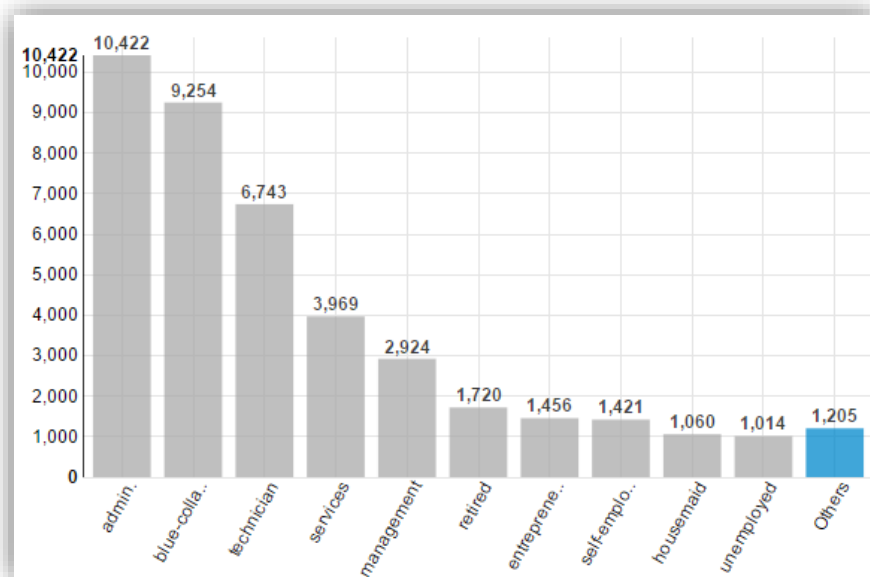


Figure 24 – Amazon Data Insights: Job Field Categorical Attributes

To create the data-source, visit at <https://console.aws.amazon.com/machinelearning/>, click on the *Create New* option, the system will prompt us to indicate where the data is located, click on S3 and then input the location of the data in the S3 bucket. The data-source gets validated in few seconds and requires a *schema*. For this experiment, we will allow Amazon to automatically create the schema *which is the*

recommended option on the Amazon ML console but there is option to provide a separate file when The goal of creating the data-source is because Amazon ML needs to know exact location of the data in the S3 bucket so that it can easily infer the names of the attributes in the data, types of each attributes *binary, numeric, text, and categorical types*, and the name of *label* attribute. Also, the ML model training and evaluation process uses the data-source ID to reference the data. Amazon ML supports using a single data input file, or multiple files as data input. The collection of multiple files as data inputs must satisfy the conditions that all files must have the same data schema and all files must reside in the Amazon S3 prefix with path for the collection of the inputs ending with '/' character. The screenshots in Figure 23 and **Error! Reference source not found.** the powerful visualization tool on the data-source console that provides insight to better understand our data. Amazon ML infers the data type of each attribute on a sample of each attribute's values. The schema consists of attributes and their assigned data types. At this stage, it is important we assign the appropriate data type because this will assist Amazon ML to ingest the data correctly and correctly process the feature of the attributes which will eventually influence the predictive performance of the ML model [11]. One important thing to note about our data-source is that it does not store data; it only references the data that is stored in the S3 buckets. When the data that is being referenced is moved away from the S3 bucket, Amazon will not be able to access the data or use it for any operation.

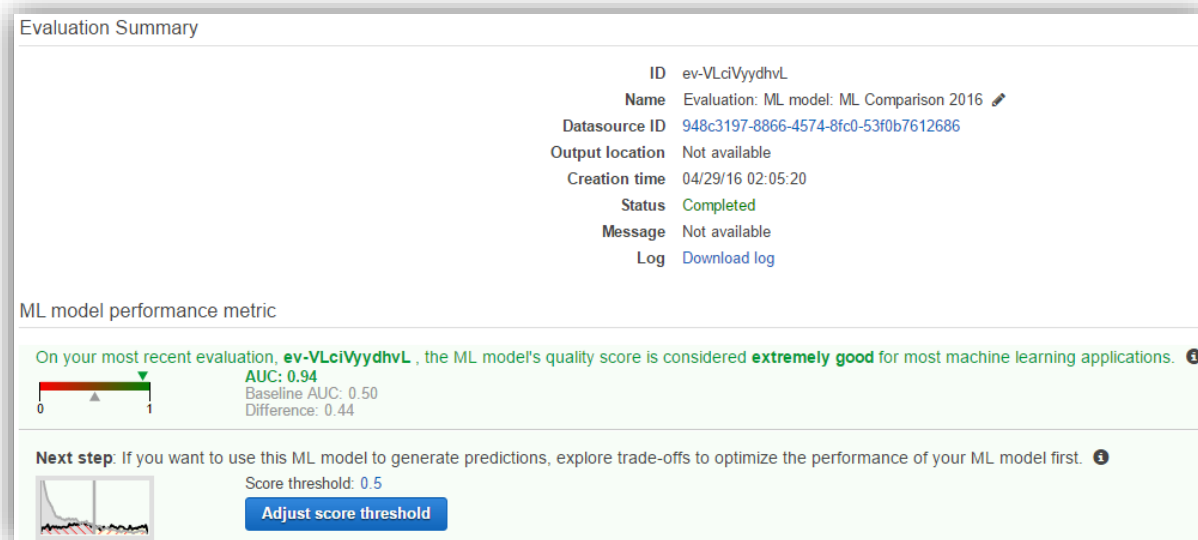


Figure 25 – Amazon Model Performance Evaluation

3.3.1.2 Step 2: Split Dataset

At this point, we have successfully created our data-source and we want to use it to create an ML model. Figure 22 show that we can the data-source to create or train an ML model, evaluate an ML model, and generate batch predictions. To create an ML model using the data-source, select *Amazon Machine Learning, ML Models*, on the *ML Models* summary page, select *create new ML Model*. Since we have a data-source, we will choose *I already created a data-source pointing to my S3 data*. For the *training and evaluation settings*, we select default. Amazon provides a default setting of splitting the input data-source into 70 percent for training and 30 percent for evaluation, though there is option to change this percentage. The goal here is to allow our ML model to generate predictions simply by identifying patterns in our data using our training data-source. When we click on the *finish* icon, we automatically submit request for Amazon ML to split our data-source into ration 70:30, create an ML model to train on the 70 percent, and create an evaluation to evaluate the ML model on the 30 percent of our data. This three request moves into *pending*, then moves to *in progress* after two minutes, and *completed* after seven minutes *ten minutes in total*. We need to wait for the process to reach the *completed* status before

proceeding to the next step. Once the model is ready, Amazon provides simple APIs for making predictions. With these tools, wizards, and APIs; developers do not have to learn complex prediction generation codes or manage infrastructure or learn complex ML algorithms and technology.

The screenshot displays the Amazon ML Model Creation Summary page, which is organized into four main sections:

- ML model summary:**
 - ID: ml-HGGsDudOMQk
 - Name: ML model: ML Comparison 2016
 - Type: Binary classification
 - Creation time: 04/29/16 02:05:20
 - Status: Completed
 - Message: Not available
 - Log: [Download log](#)
- Datasource (training):**
 - Datasource ID: ffa8098c-2d3b-4772-b95e-3bfa2ca8d43f
 - Target: y
 - Input schema: [View input schema](#)
- Evaluations:**
 - Evaluations created: 1
 - Latest evaluation result: Not available
 - Button: [Perform another Evaluation](#)
- Predictions:**
 - Score threshold: 0.5
 - A single dataset:** Generate one-time predictions for a single dataset. Button: [Generate batch predictions](#)
 - Try real-time predictions:** Generate real-time predictions in your browser. Button: [Try real-time predictions](#)
 - Enable real-time predictions:** To enable real-time predictions now, create a real-time endpoint. Button: [Create endpoint](#)

Figure 26 – Amazon Model Creation Summary

3.3.1.4 Step 5: Create Predictions

After training the ML model, Amazon ML generates predictions by identifying patterns in the historical data using the training data-source. Predictions can be adjusted by simply adjusting the score threshold. For this experiment, we will make our threshold 0.5 since that is the same we used for Windows Azure ML. This will allow us make fair comparison. Amazon supports batch and real-time predictions. To create real-time prediction, we need to create end-point. The real-time endpoint allows us to request predictions in real time over the web. The billing is based on the size of the model. For this thesis the size of our model is 400.1 KB. We will incur the reserved the reserved capacity of \$0.001 for every hour our endpoint is active. The prediction charge for real time prediction is \$0.0001 per prediction, rounded up to the nearest penny. Real-time prediction is a single prediction that Amazon makes immediately at no extra cost. We simply input data manually into the Amazon ML interactive web service testing window and the system will return the result as shown in Figure 27. Unlike the real-time prediction, batch prediction takes more time because it is used to generate predictions of set of observations and when low-latency is not required. In the case of batch prediction, first we need to specify which of the models we want for the batch prediction, then we select the dataset we want from the S3 bucket, then click on create batch predictions.

Try real-time predictions

You submitted 20 out of 20 data values for this prediction.

Try generating real-time predictions for free using the web browser on this page. To request a real-time prediction, complete the following form or provide a single data record in CSV format. To provide a data record, choose the **Paste a record** button.

Q: Items per page: << < 1 - 10 of 21 > >>

#	Name	Type	Value
1	age	Numeric	<input type="text" value="28"/>
2	job	Categorical	<input type="text" value="management"/>
3	marital	Categorical	<input type="text" value="single"/>
4	education	Categorical	<input type="text" value="university degree"/>
5	default	Categorical	<input type="text" value="no"/>
6	housing	Categorical	<input type="text" value="yes"/>
7	loan	Categorical	<input type="text" value="no"/>
8	contact	Categorical	<input type="text" value="cellular"/>
9	month	Categorical	<input type="text" value="jun"/>
10	day_of_week	Categorical	<input type="text" value="thur"/>

« < 1 - 10 of 21 > »

Prediction results

Target name y
ML model type BINARY
Predicted label 1

```
{
  "Prediction": {
    "details": {
      "Algorithm": "SGD",
      "PredictiveModelType": "BINARY"
    },
    "predictedLabel": "1",
    "predictedScores": {
      "1": 0.8381996154785156
    }
  }
}
```

Next steps

To enable real-time predictions for your applic (Capacity and use charges apply. [Learn more](#))

Figure 27 – Amazon Interactive Real-Time Prediction Testing Window

Explain this chart

Trade-off based on score threshold

- **91% are correct**
585 true positive
10,675 true negative
- **9% are errors**
317 false positive
778 false negative
- 7% of the records are predicted as "1"
- 93% of the records are predicted as "0"

▼ **Advanced metrics**

False positive rate	0.0288	0
Precision	0.6486	0
Recall	0.4292	0
Accuracy	0.9114	0

True negative (86.4%)
True positive (4.7%)
False negative (6.3%)
False positive (2.6%)

Figure 28 – AUC Metric of Amazon Machine Learning Model

Chapter 4: Comparative Analyses

4.1 Comparing Google, Amazon, and Microsoft Machine Learning Services

Based on the experiments in the previous chapter and also benchmarking the predictive performances of each model on the same dataset, this chapter will present the comparative analysis of the results. The results are not exhaustive because some important functionality that might have happened behind the hood while carrying out the experiments might have been omitted in the comparative analysis results. The goal of these analyses is to provide relatively suitable answers to questions like: *what do we look for in cloud-based ML service platforms, which ML service is most suitable for us, and etc.* To answer these questions, a lot of factors need to be considered. For instance, the duration for free tier subscription or visualization tools could be a factor for an individual.

Table 8 – Pricing

Google Prediction API Subscriptions		
Google Subscription	Free Tier	Standard Tier
Service Level Agreement	None	99.9% availability
Monthly Access Fee	None	\$10 per project
Free Predictions	Up to 100 per day	Up to 10, 000 per day
Additional Predictions	20, 000 lifetime cap	\$0.50 per 1, 000 predictions
Training Data	Up to 5MB trained per day	\$0.002 per MB
Free Streaming Training	None	Up to 10, 000 per day
Additional Streaming Updates	None	\$0.05 per 1, 000 updates
Windows Azure Studio Subscriptions		
Azure Subscription	Free Tier	Standard Tier
Max No. of Modules per Experiment	100	Unlimited
Max Experiment Duration	1 hour per experiment	Up to 7 days per experiment with maximum of 24 hours per module
Max Storage For Training Sets	10G	Unlimited – BYO
Execution/ Performance	Single node	Multiple nodes
ML Studio Usage	None	\$1/ Studio Experiment Hour
ML API Usage	None	\$2/ Production API Compute Hour
Amazon Machine Learning Subscriptions		
Amazon Subscription	Free Tier	Standard Tier
Data analysis and Model Building	\$0.42/ hour	\$0.42/ hour
Batch Predictions	\$0.10/ 1, 000 predictions	\$0.10/ 1, 000 predictions
Real-Time Predictions	\$0.0001/ predictions	\$0.0001/ predictions

4.1.1 Configuration Process

Very generally, cost and time are crucial when developing solutions. The good news here is that all the three services require no initial cost, no minimum fees, and termination fees, rather they support pay-as-you-use plan. Nevertheless, their charges vary based on the different services they provide and the

charging format – *hourly, monthly, or per use*. For instance, looking at the comparison result in Table 8, Amazon ML charges per hour for the compute time used to build predictive models but in the case of making predictions, it charges per prediction. Also, Amazon reserves the capacity of \$0.001 for every hour the real time prediction endpoint is active and the actual prediction charge for real time prediction is \$0.0001 per prediction, which is rounded up to the nearest penny. The three services give reasonable charges but looking at the paid tiers subscriptions, Azure seems to get really expensive over time with \$1 per studio experiment hour and \$2 production API compute hour. Google seems to be fairly expensive but way cheaper when compared with Azure. Amazon seems to be the cheapest, most of its charges are way less than a dollar and the twelve months free tier subscription is quite impressive and it supports generating billions of predictions daily. Note that the pricing does not include storage. We cannot compare storage because an individual might have a paid storage and just integrate it ML solution with his pre-owned storage account. Also, the timing for configuring each service depends on some factors. For instance, having a Google or Microsoft account will automatically reduce the total configuration time. But with the assumptions that a user has no account, Amazon seems to be the fastest – *less than two minutes*.

Table 9 – Data Processing

S/N	Google Prediction API
1.	It detects data types automatically but supports only string and numeric data types.
2.	It has poor support for data visualization but supports updating dataset during model training.
3.	It supports 2.5 GB text file, HTTP request of 2 MB dataset size, spreadsheet, and JSON data formats.
4.	It supports data training of maximum of 250MB and can train models on uploaded text files.
5.	It supports uploading data through different media – the API, web, or command line.
S/N	Windows Azure Studio
1.	It provides powerful data visualization tools – <i>tables, histogram, and statistical summary</i> .
2.	It supports text file directly uploaded on its experiment workspace, Azure Storage, SQL database, Hadoop HiveQL, and URL data sources.
3.	It supports maximum 10 GB dataset size and can train models on uploaded text files.
4.	It supports Boolean, numeric, string, categorical, and timespan data types.
5.	It supports RData, zip, OData values, Hive, SQL tables, text files, and .csv data formats.
S/N	Amazon Machine Learning
1.	Data in any of its storage centers must be referenced.
2.	It supports text files in S3 buckets, AWS RDS, AWS Redshift, and ASW S3 table data sources.
3.	It supports Maximum 100 GB dataset size and can train models on uploaded text files.
4.	It supports Boolean, categorical, numeric, and string data types.
5.	It supports S3, Redshift database, and .csv file data formats.
6.	It provides powerful data visualization tools – <i>tables and histogram</i> .

4.1.2 Data Processing

Although the dataset was pre-processed before uploading to the services, it is good to know that Azure provides custom R programming and Python scripts as data pre-processing tools which will definitely pose threat to the other services. For the three services, uploading data was quite easy and fast *approximately 40 seconds with just few clicks and easy to navigate console*. Since data does most of the lifting in ML, it is important for the services to have tools that can give detailed data insights. I will rank Google the least here. Amazon and Azure provide powerful data insights and visualization tools, but

Amazon seems to be the best because it does not only analyze data, it further gives detailed explanations about the data and support options of playing around with the data in terms of shifting the threshold to suite optimization, changing the ratio of the training and testing dataset, and while playing around with the data, the visualization tool do help in letting us know if what we are doing is good enough. From the result in Table 9, Amazon supports the largest dataset size, followed by Azure, and Google is last. Updating dataset during model training and uploading data through different media seems to be functionality Google possess that can pose threat to the other services. The services have different approach when processing missing values in dataset. Windows Azure can replace missing values with either mean, median, or mode. Google replaces missing strings with "" and missing numbers with 0. And Amazon ML handles allows missing value imputation. Unlike the other two services that allow developers to change the ratio of splitting training and testing datasets, Google does this automatically and does not allow developers to change.

Table 10 – Algorithms and Evaluations

	Amazon	Azure	Google
Accuracy (AUC)	0.94	0.95	0.75
Training Time (s)	420	110	1.14
Predictions Time (s)	180	120	0.772
Algorithms	Classification Regression	Classification Regression Clustering	–

4.1.3 Model Processing

Building an ML model is a core task when developing ML solution. This task is basically done by using algorithms. Table 10 gives the comparisons of the algorithms each service provides. In the case of Google, it provides no clue about the algorithm it uses to build its model which is a major flaw when compared to the other two services but its power to train model incrementally is commendable. Amazon also does not allow developers to choose algorithms just like Google but it allows developers to know which algorithm it chooses for a particular solution via its powerful visualization tool. Once data is uploaded on Google or Amazon platforms, it automatically chooses which algorithm best suites the solution. It will be very fair enough to rank Windows Azure as first here considering the rich and wide range of algorithms it provides alongside with features for easy configuration of its algorithms. It also allows developers to tune parameters of its algorithms for better optimization. *Note that linear models scales better with dataset size, faster to train, and very easy to tune.* Amazon ML and Google supports easy integration with other services they provide especially the storage services.

After building the model, it is essential to evaluate the model to determine its accuracy. This is done by measuring the model accuracy in terms of confidence factor approaching number 1. For the comparative analyses in Table 10, the *area under the curve* – AUC was used as the evaluation metric. This metric is a common evaluation metric that is used for binary classification problems *Binary means something having two parts* whose result will either be true – 1 or false – 0. Let us imagine a plot of true vs false positive rates as a threshold of our binary classification, if the classifier is good, the true positive rate will increase quickly and the AUC will be close to 1. If it is random guessing, it will increase linearly with the false positive rate and the AUC will increase to about 0.5. With the performance value ranging between 0 to 1, it means the higher the AUC the better the model [33]. The AUC results were automatically calculated by the services' but the prediction and training time were calculated based on the normal wall clock time. From the result in Table 10, Google seems to be the fastest in terms of model training and prediction creation, while Amazon will be ranked as the slowest. A general disadvantage of

the three services is that they do not allow models to be downloaded or transferred for inspection or for usage outside their platforms; all models must be used through the service provider. One disadvantage of Amazon ML is it provides relatively limited entry in terms of capabilities and algorithms it offers. In the case of Google, the absence of visualization tools and wizards to guide through the entire process of developing ML solutions is a major turn-off. Unlike Azure ML and Google Prediction API that automatically detects data once uploaded, Amazon ML has a de-merit in this case, we need to create a data-source that will point to Amazon where the data is located and once the data is moved from the location, Amazon cannot infer anything the data.

It is difficult to rank these services in terms of how accurate the models are because the performance of the models depends on algorithm of choice and the dataset. Although the three services' aim is to make building ML solutions easy for developers of all expertise level, Amazon ML and Azure better achieved this by providing visualization tools and wizards that guides through the entire process of developing ML solutions and also provides simple APIs for obtaining predictions without having to implement custom codes to generate predictions or manage infrastructures. Azure also provides excellent pre-built template, drag-and-drop algorithms.

Table 11 – Model Processing

S/N	Google Prediction API
1.	It detects data types automatically and allows models to be incrementally trained.
2.	It does not support batch predictions – <i>one prediction per API call.</i>
3.	It supports classification and regression application types.
4.	It provides single model evaluation but does not support downloading or transferring model.
S/N	Windows Azure Studio
1.	It supports linear and non-linear algorithms.
2.	It supports batch training and cross validation.
3.	It supports classification, regression, clustering, and anomaly detection application types.
4.	It provides powerful data and model visualization tools – <i>tables, histogram, and statistical summary.</i>
5.	It trains model stochastically when data is provided in batches.
6.	It does not support real-time model training but supports incremental model training.
7.	It does not support downloading or transferring model and does not support incremental model training.
8.	It provides single model evaluation and adjustable threshold.
S/N	Amazon Machine Learning
1.	It supports linear algorithms, cross validation and batch training.
2.	It provides single model evaluation and adjustable threshold.
3.	It supports classification and regression application types.
4.	It provides powerful data and model visualization tools – <i>tables and histogram.</i>
5.	It does not support real-time model training and incremental model training.
6.	It does not support downloading or transferring model.

Chapter 5: Conclusion

Machine Learning – *ML* has earned great recognition in the IT industry over the years due to its ability to handle problems that are related to predictive analytics but it does not suffice for some particular problems. In some cases, different approach to a particular problem could be better than ML approach in terms of cost and time. A simple logic for knowing when ML approach will not suffice for a particular problem is to check if all cases can be quickly and robustly covered with simple rules. Modeling a traffic light system is a good example of a problem where ML approach will not suffice. Simple lines of soft coding could be used to easily model rules that will control the functionalities of the system. In cases where coding with rules is difficult to figure out, expensive, and time consuming, then ML approach will be a better approach. For instance, sorting emails to spam or legitimate cannot be easily done with soft coding because the sorting process could be influenced by factors like exceptions, edge cases, uncertainties, and etc. With ML approach, we can make a system learn from dataset that contains spam and legitimate emails, and then we train an ML model against the dataset which will produce a good algorithm that can perform the predictions.

It is good to note that overfitting problem can occur when gathering dataset. This problem occurs when a model has too many parameters relative to the number of observations, this will make the model over-reacts to minor fluctuations in the training dataset. An example is when we want to predict the class of customers that are most likely to subscribe to a television channel and then, we include their gender fields in the model. If the dataset contains mostly information about female users, then the result is most likely to have negative predictions for male users. This problem can be avoided by using techniques like cross-validation, early stopping, Bayesian priors, pruning, and etc. on parameters. Cross-validation was used during the process of the experiments in chapter 3 [9][34].

The fact that the experiments and comparisons in previous chapters were bench-marked using real-world dataset means that it is very likely to have different results with another dataset, and it seems no service is better than the other; the choice will basically depend on many factors which have been highlighted in chapter 4. Nevertheless, I will recommend Windows Azure for developers with little or no knowledge in ML due to its powerful visualization tools and availability of tutorials that can guide through the entire process of developing ML solutions. It is clear that Google is not favorable for developers with low coding skills and it scored lowest in terms of visualization and algorithm selection, it performs best in terms of speed of training and predictions; it does all these in milliseconds. Azure ML is a bit different from the other services in the sense that it does not offer API but it allows developers to create their own data processing pipeline through its studio and then turn that into an API.

Further research in this line could focus on developing ML service that can extract patterns from audio or video datasets. Combining the technical know-how of *ML, Deep Learning, and Data Mining* could provide a perfect tool that can achieve state of the art result in this research. I believe this new research direction could come into play in the nearest future because the Deep Learning field is equipped with methods that are based on neural networks, which gives it power to extract patterns from visual dataset. In the Big Data field, data gets automatically recorded via digital processing media with an inhomogeneous property *generated from different sources or in different contexts*. And finally, ML field has the power to make algorithms extract patterns in dataset.

Appendix

Instance 1 – Sample Code for Dataset Formatting

```
import csv
data_path = 'data/advert.csv'
with open (data_path,'r') as content:
data=csv.reader (content, delimiter=';', quotechar='')
rows=[r for r in data]
output_path='data/cleanup.csv'
with open(output_path,'wb')as content:
data = csv.writer(content,quoting=csv.QUOTE_MINIMAL)
data.writerow(row[0])
for r in rows[1:]:
y=r[-1]
r[-1]=1 if y=='yes'else 0
data.writerow(r)
```

Instance 2 – Sample Codes to Test Microsoft Azure Machine Learning Online

R Programming Language:

```
library("RCurl")
library("rjson")
# Accept SSL certificates issued by public Certificate Authorities
options(RCurlOptions = list(cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl")))
h = basicTextGatherer()
hdr = basicHeaderGatherer()
req = list(
  Inputs = list(
    "input1" = list(
      "ColumnNames" = list("age", "job", "marital", "education", "default", "housing", "loan",
"contact", "month", "day_of_week", "duration", "campaign", "pdays", "previous", "poutcome",
"emp_var_rate", "cons_price_idx", "cons_conf_idx", "euribor3m", "nr_employed"),
      "Values" = list( list( "0", "value", "value", "value", "value", "value", "value", "value", "value",
"value", "0", "0", "0", "0", "0", "value", "0", "0", "0", "0", "0" ), list( "0", "value", "value", "value", "value",
"value", "value", "value", "value", "value", "0", "0", "0", "0", "value", "0", "0", "0", "0", "0" ) )
    ),
    GlobalParameters = setNames(fromJSON('{}'), character(0))
  )
body = enc2utf8(toJSON(req))
api_key = "abc123" # Replace this with the API key for the web service
authz_hdr = paste('Bearer', api_key, sep=' ')
h$reset()
```

```

curlPerform(url =
  "https://ussouthcentral.services.azureml.net/workspaces/927170b501444a45b968d594b19b9d84/services/
  20aa27a8990f4c8998287c28a11c0198/execute?api-version=2.0&details=true",
  httpheader=c('Content-Type' = "application/json", 'Authorization' = authz_hdr),
  postfields=body,
  writefunction = h$update,
  headerfunction = hdr$update,
  verbose = TRUE
)
headers = hdr$value()
httpStatus = headers["status"]
if (httpStatus >= 400)
{
  print(paste("The request failed with status code:", httpStatus, sep=" "))
  # Print the headers - they include the request ID and the timestamp, which are useful for debugging the
  failure
  print(headers)
}
print("Result:")
result = h$value()
print(fromJSON(result))

```

Python Programming Language:

```

import urllib2
# If you are using Python 3+, import urllib instead of urllib2
import json
data = {
  "Inputs": {
    "input1": {
      "ColumnNames": ["age", "job", "marital", "education", "default", "housing",
"loan", "contact", "month", "day_of_week", "duration", "campaign", "pdays", "previous",
"poutcome", "emp_var_rate", "cons_price_idx", "cons_conf_idx", "euribor3m", "nr_employed",
"y"],
      "Values": [[ "0", "value", "value", "value", "value", "value", "value", "value",
"value", "value", "0", "0", "0", "0", "value", "0", "0", "0", "0", "0", "0" ], [ "0", "value",
"value", "value", "value", "value", "value", "value", "0", "0", "0", "0",
"value", "0", "0", "0", "0", "0", "0" ], ]
    },
    "GlobalParameters": {
  }
}
body = str.encode(json.dumps(data))
url =
'https://ussouthcentral.services.azureml.net/workspaces/927170b501444a45b968d594b19b9d84/serv
ices/9bd664636c5e47438ca189f428008772/execute?api-version=2.0&details=true'
api_key = 'abc123' # Replace this with the API key for the web service
headers = {'Content-Type': 'application/json', 'Authorization': ('Bearer '+ api_key)}
req = urllib2.Request(url, body, headers)
try:
  response = urllib2.urlopen(req)

```

```

# If you are using Python 3+, replace urllib2 with urllib.request in the above code:
# req = urllib.request.Request(url, body, headers)
# response = urllib.request.urlopen(req)
result = response.read()
print(result)
except urllib2.HTTPError, error:
    print("The request failed with status code: " + str(error.code))
    # Print the headers - they include the request ID and the timestamp, which are useful for debugging the
    failure
    print(error.info())
    print(json.loads(error.read()))

```

C# Programming Language:

```

// This code requires the Nuget package Microsoft.AspNet.WebApi.Client to be installed.
// Instructions for doing this in Visual Studio:
// Tools -> Nuget Package Manager -> Package Manager Console
// Install-Package Microsoft.AspNet.WebApi.Client
using System;
using System.Collections.Generic;
using System.IO;
using System.Net.Http;
using System.Net.Http.Formatting;
using System.Net.Http.Headers;
using System.Text;
using System.Threading.Tasks;
namespace CallRequestResponseService
{
    public class StringTable
    {
        public string[] ColumnNames { get; set; }
        public string[,] Values { get; set; }
    }
    class Program
    {
        static void Main(string[] args)
        {
            InvokeRequestResponseService().Wait();
        }
        static async Task InvokeRequestResponseService()
        {
            using (var client = new HttpClient())
            {
                var scoreRequest = new
                {
                    Inputs = new Dictionary<string, StringTable> () {
                        {
                            "input1",
                            new StringTable()
                            {
                                ColumnNames = new string[] { "age", "job", "marital", "education", "default",

```


References

- [1] Simon, Phil. *Too Big to Ignore: The Business Case for Big Data*. Vol. 72. John Wiley & Sons, 2013.
- [2] Reutemann, Peter, Bernhard Pfahringer, and Eibe Frank. "A toolbox for learning from relational data with propositional and multi-instance learners." In *AI 2004: Advances in Artificial Intelligence*, pp. 1017-1023. Springer Berlin Heidelberg, 2004.
- [3] Holmes, Geoffrey, Andrew Donkin, and Ian H. Witten. "Weka: A machine learning workbench." In *Intelligent Information Systems, 1994. Proceedings of the 1994 Second Australian and New Zealand Conference on*, pp. 357-361. IEEE, 1994.
- [4] Alpaydm, Ethem. *Introduction to machine learning*. MIT press, 2014.
- [5] Tulloch, Mitch. *Introducing Windows Azure for IT Professionals*. Microsoft Press, 2013.
- [6] Zhang, Qi, Lu Cheng, and Raouf Boutaba. "Cloud computing: state-of-the-art and research challenges." *Journal of internet services and applications* 1, no. 1 (2010): 7-18.
- [7] Garner, Stephen R., Sally Jo Cunningham, Geoffrey Holmes, Craig G. Nevill-Manning, and Ian H. Witten. "Applying a machine learning workbench: Experience with agricultural databases." In *Proc Machine Learning in Practice Workshop, Machine Learning Conference, Tahoe City, CA, USA*, pp. 14-21. 1995.
- [8] Svátek, Vojtech, and Vysoka škola ekonomická v Praze, Praha (Czech Republic) "Learning from observational data with prior knowledge." (1997).
- [9] Witten, Ian H., and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.
- [10] Microsoft. *Windows Azure General Availability. Data*. The Official Microsoft Blog, 2010.
- [11] Amazon Web Services. <http://aws.amazon.com/>. The Official Amazon Website, Accessed, 2015.
- [12] Schölkopf, Bernhard, and Alexander J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.
- [13] Garner, Stephen R. "Weka: The Waikato environment for knowledge analysis." In *Proceedings of the New Zealand computer science research students conference*, pp. 57-64. 1995.
- [14] Louis Dorard. *Kaggle credit challenge: http://www.programmableweb.com/*. The Programmable-Web Official Website, Accessed, 2016.
- [15] Jeff Barnes. *Azure machine learning: Microsoft azure essentials*. ISBN: 978-0-7356-9817-8 (2015).
- [16] Mitchell, Tom M. "Machine learning. WCB." (1997).
- [17] EDX Online Learning Destination. <http://courses.edx.org/>. The EDX Official Website, Accessed, 2016.
- [18] Microsoft. <http://azure.microsoft.com/>. The Official Microsoft Azure Website, Accessed, 2016.
- [19] Jopson, Barney. "Amazon urges California referendum on online tax." *The Financial Times* (2011).
- [20] Nick Wilson. *Machine Learning Throwdown: https://blog.bigml.com/2012/08/02/machine-learning-throwdown*. Accessed, 2016.
- [21] Thomas H. Cormen, Charles Eric Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. Vol. 6. Cambridge: MIT press, 2001.
- [22] Synergy Research Group. *Microsoft cloud revenue leap: Amazon is still way out in front – https://www.srgresearch.com*. The Synergy Research Group Official Website, Accessed, 2016.
- [23] Gartner. *Gartner says emerging markets drove worldwide smartphone sales to 19 percent growth in first quarter of 2015*. Press Release.

- [24]Tech Research International. *Software top 100: the world's largest software companies* – <https://www.softwaretop100.org/software-industry-trends-2011>. The Tech Research Official Website, Accessed, 2016.
- [25]Amir Tabakovic. *BigML: machine learning made beautifully simple*. VP of Business Development, BigML, Inc. Zurich.
- [26]Rokach, Lior, and Oded Maimon. *Data mining with decision trees: theory and applications*. World scientific, 2014.
- [27]Bache, K., and M. Lichman. "UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml>]. Irvine, CA: University of California, School of Information and Computer Science. Begleiter, H. Neurodynamics Laboratory. State University of New York Health Center at Brooklyn. Ingber, L. (1997). Statistical mechanics of neocortical interactions: Canonical momenta indicators of electroencephalography." *Physical Review E* 55 (2013): 4578-4593. Moro, Sérgio, Paulo Cortez, and Paulo Rita. "A data-driven approach to predict the success of bank telemarketing." *Decision Support Systems* 62 (2014): 22-31.
- [28]Moro, Sergio, Raul Laureano, and Paulo Cortez. "Using data mining for bank direct marketing: An application of the crisp-dm methodology." (2011).
- [29]Steve, Richardson. "Google Marketing Analysis." (2011).
- [30]Claburn, Thomas, Cameron Winklevoss, and Tyler Winklevoss. "Google Founded By Sergey Brin, Larry Page... And Hubert Chang." *InformationWeek* (2011).
- [31]Page, Founders Larry, Sergey Brin, area served Worldwide, Larry Page CEO, and Eric Schmidt. "Sunny Spontaneous Group."
- [32]Google. <http://cloud.google.com/prediction/docs/>. The Official Google Developers Console, Accessed, 2016.
- [33]Kaggle. <https://www.kaggle.com/wiki/AreaUnderCurve/>. Accessed, 2016.
- [34]Everitt, B. *The Cambridge dictionary of statistics/BS Everitt*. New York:: Cambridge University Press, Cambridge, UK, 2002.
- [35]Google. <https://www.google.no/>. The Official Google Search Engine, Accessed, 2016.