Big Data Parallel Programming Project

Abhishek Pulicherla

August 2020

1 Introduction

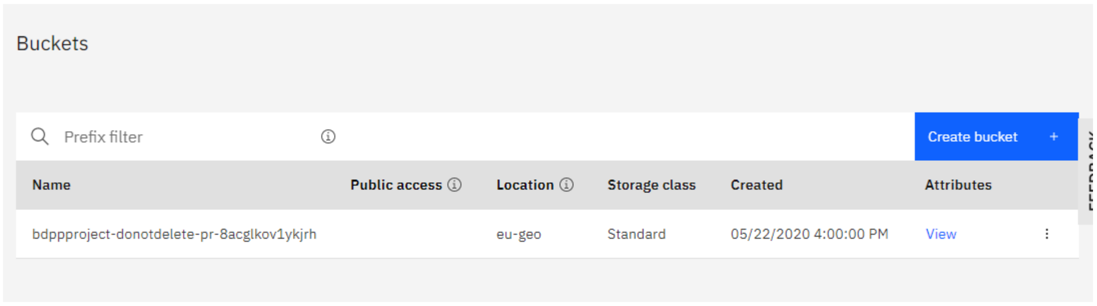
This project aims to classify the credit card payment defaulters, using the Default of Credit Card Clients Data Set in the Apache Spark Framework and then implement the feature engineering and machine learning algorithms on IBM Watson Studio, which is a IBM cloud service and the Google Cloud Platform in Spark Context.

2 Implementing on IBM Cloud

IBM Cloud provides an application called IBM Watson studio, in which we can run Jupyter environment and all the python and spark libraries.

2.1 Create a new bucket

First, I start with creating a new bucket for data storage on the IBM Cloud platform, Figure 1.

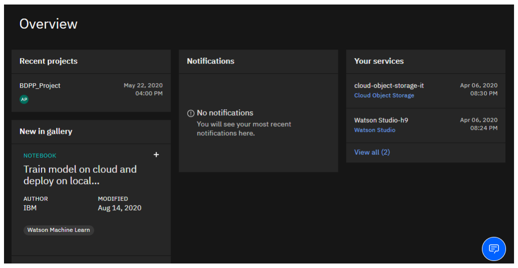


Name	Public access	Location	Storage class	Created	Attributes
bdppproject-donotdelete-pr-8acglkov1ykjrh		eu-geo	Standard	05/22/2020 4:00:00 PM	View

Figure 1: Creating a bucket in IBM Cloud

2.2 Start the IBM Watson Service

Next, I started the IBM watson service, Figure 2



Overview

Recent projects

BDPP_Project — May 22, 2020 04:00 PM

New in gallery

NOTEBOOK

Train model on cloud and deploy on local...

AUTHOR IBM — MODIFIED Aug 14, 2020

Watson Machine Learn

Notifications

No notifications

You will see your most recent notifications here.

Your services

cloud-object-storage-it — Cloud Object Storage — Apr 06, 2020 08:30 PM

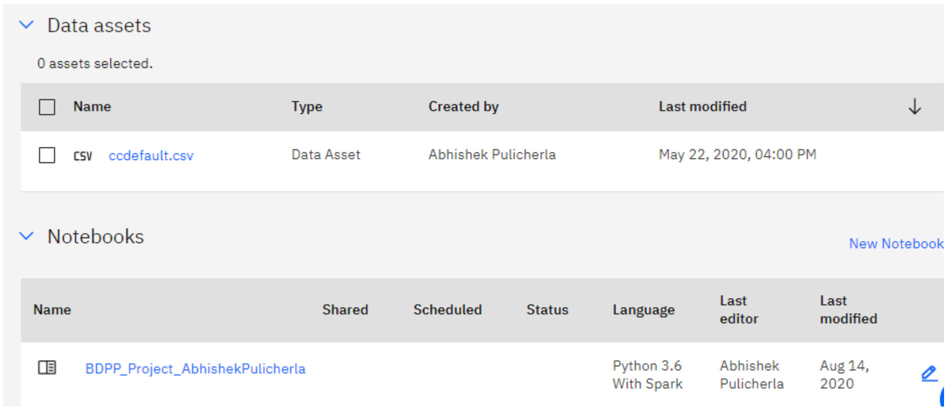
Watson Studio-h9 — Watson Studio — Apr 06, 2020 08:24 PM

View all (2)

Figure 2: IBM Watson studio dashboard

2.3 Run the Job

Next, I upload the dataset and the code to run the Job. The Job runs on one worker node, Figure 4



Data assets

0 assets selected.

Name	Type	Created by	Last modified
CSV ccdefault.csv	Data Asset	Abhishek Pulicherla	May 22, 2020, 04:00 PM

Notebooks

New Notebook

Name	Shared	Scheduled	Status	Language	Last editor	Last modified
BDPP_Project_AbhishekPulicherla				Python 3.6 With Spark	Abhishek Pulicherla	Aug 14, 2020

Figure 3: The dataset and the code on IBM

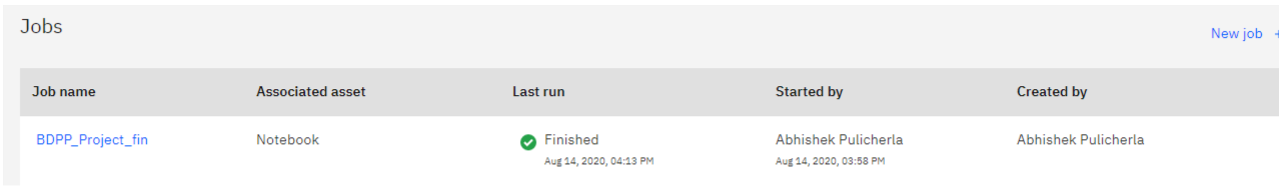
3 Google Cloud Platform

Google cloud platform offers different API's to run the spark environment.

3.1 API's

We need to activate the following API's to run the operation

- Compute Engine
- Google Storage Bucket
- Dataproc


Jobs

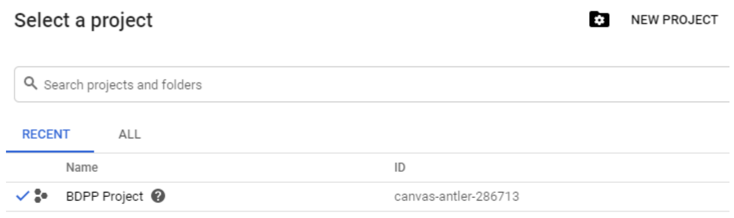
New job

Job name	Associated asset	Last run	Started by	Created by
BDPP_Project_fin	Notebook	Finished Aug 14, 2020, 04:13 PM	Abhishek Pulicherla Aug 14, 2020, 03:58 PM	Abhishek Pulicherla

Figure 4: Running the Job on the IBM Cloud platform

3.1.1 Creating a Project

First, I created a new project, Figure 5.


Select a project

NEW PROJECT

Search projects and folders

RECENT ALL

Name	ID
BDPP Project	canvas-antler-286713

Figure 5: Creating a project(GCP)

3.1.2 Creating a Bucket

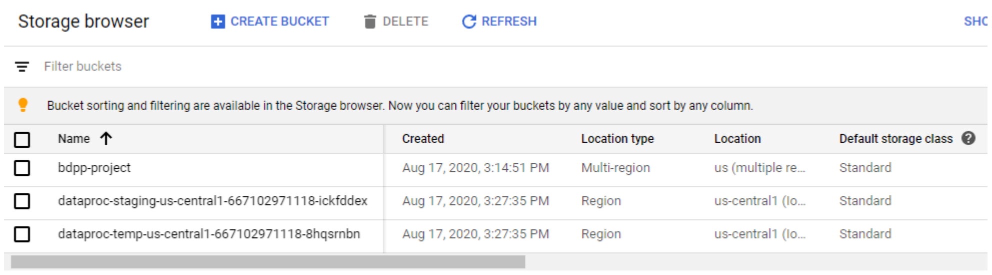
Next, I created a bucket to upload and store the notebook consisting of the code and the dataset, Figure 6.

3.1.3 Creating a cluster

Next, I created a cluster in the Dataproc API on GCP, with 1 master and 2 worker nodes. I installed the Anaconda and Jupyter notebook environment along with the creation of the cluster, Figure 7.

3.1.4 Running the Job

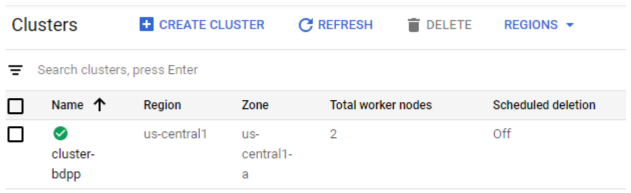
We can check the created clusters in VM Instance on the compute engine API, Figure 8. After the clusters are created, then we need to run the job by importing the python code and the dataset from the storage bucket.



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

Name	Created	Location type	Location	Default storage class
bdpp-project	Aug 17, 2020, 3:14:51 PM	Multi-region	us (multiple re...	Standard
dataproc-staging-us-central1-667102971118-ickfddex	Aug 17, 2020, 3:27:35 PM	Region	us-central1 (lo...	Standard
dataproc-temp-us-central1-667102971118-8hqsrnbn	Aug 17, 2020, 3:27:35 PM	Region	us-central1 (lo...	Standard

Figure 6: GCP Storage Bucket



Name	Region	Zone	Total worker nodes	Scheduled deletion
cluster-bdpp	us-central1	us-central1-a	2	Off

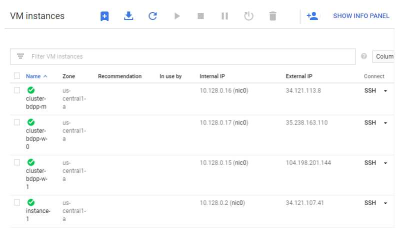
Figure 7: Clusters in GCP

4 Dataset

This research aimed at the case of customers' default payments in Taiwan. This dataset contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005. It has 30,000 observations for 23 features plus two columns with ID and the label for the default status.

5 Features of the Dataset

- ID: ID of each client
- LIMIT_BAL: Amount of given credit in NT dollars (includes individual and family/supplementary credit
- SEX: Gender (1=male, 2=female)
- EDUCATION: (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
- MARRIAGE: Marital status (1=married, 2=single, 3=others)
- AGE: Age in years



Name	Zone	Recommendation	In use by	Internal IP	External IP	Connect
cluster-bdpp-m	us-central1-a			10.128.0.16 (nic0)	34.121.113.8	SSH
cluster-bdpp-w-0	us-central1-a			10.128.0.17 (nic0)	35.238.163.110	SSH
cluster-bdpp-w-1	us-central1-a			10.128.0.15 (nic0)	104.198.201.144	SSH
instance-1	us-central1-a			10.128.0.2 (nic0)	34.121.107.41	SSH

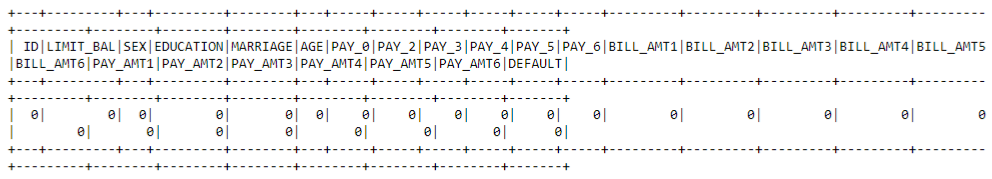
Figure 8: Clusters in the VM Instance(GCP)

- PAY_0: Repayment status in September, 2005 (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, … 8=payment delay for eight months, 9=payment delay for nine months and above)
- PAY_2: Repayment status in August, 2005 (scale same as above)
- PAY_3: Repayment status in July, 2005 (scale same as above)
- PAY_4: Repayment status in June, 2005 (scale same as above)
- PAY_5: Repayment status in May, 2005 (scale same as above)
- PAY_6: Repayment status in April, 2005 (scale same as above)
- BILL_AMT1: Amount of bill statement in September, 2005 (NT dollar)
- BILL_AMT2: Amount of bill statement in August, 2005 (NT dollar)
- BILL_AMT3: Amount of bill statement in July, 2005 (NT dollar)
- BILL_AMT4: Amount of bill statement in June, 2005 (NT dollar)
- BILL_AMT5: Amount of bill statement in May, 2005 (NT dollar)
- BILL_AMT6: Amount of bill statement in April, 2005 (NT dollar)
- PAY_AMT1: Amount of previous payment in September, 2005 (NT dollar)
- PAY_AMT2: Amount of previous payment in August, 2005 (NT dollar)
- PAY_AMT3: Amount of previous payment in July, 2005 (NT dollar)
- PAY_AMT4: Amount of previous payment in June, 2005 (NT dollar)
- PAY_AMT5: Amount of previous payment in May, 2005 (NT dollar)
- PAY_AMT6: Amount of previous payment in April, 2005 (NT dollar)
- DEFAULT: Default payment (1=yes, 0=no)

6 Data Preprocessing

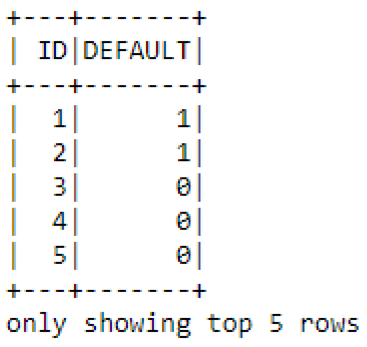
6.1 Checking for Missing Values

First, I checked for any missing values in the dataset using pyspark sql functions and there were none, Figure 9.



```
+---+---------+---+---------+--------+---+-----+-----+-----+-----+-----+-----+---------+---------+---------+---------+---------
+---------+--------+--------+--------+--------+--------+--------+-------+
| ID|LIMIT_BAL|SEX|EDUCATION|MARRIAGE|AGE|PAY_0|PAY_2|PAY_3|PAY_4|PAY_5|PAY_6|BILL_AMT1|BILL_AMT2|BILL_AMT3|BILL_AMT4|BILL_AMT5
|BILL_AMT6|PAY_AMT1|PAY_AMT2|PAY_AMT3|PAY_AMT4|PAY_AMT5|PAY_AMT6|DEFAULT|
+---+---------+---+---------+--------+---+-----+-----+-----+-----+-----+-----+---------+---------+---------+---------+---------
+---------+--------+--------+--------+--------+--------+--------+-------+
|  0|        0|  0|        0|       0|  0|    0|    0|    0|    0|    0|    0|        0|        0|        0|        0|        0
|        0|       0|       0|       0|       0|       0|       0|      0|
+---+---------+---+---------+--------+---+-----+-----+-----+-----+-----+-----+---------+---------+---------+---------+---------
+---------+--------+--------+--------+--------+--------+--------+-------+
```

Figure 9: Checking for missing values

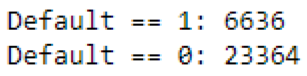


```
+---+-------+
| ID|DEFAULT|
+---+-------+
|  1|      1|
|  2|      1|
|  3|      0|
|  4|      0|
|  5|      0|
+---+-------+
only showing top 5 rows
```

Figure 10: The label default column

6.2 Checking for class imbalance

The Default column has labels 0 and 1. I checked for class imbalance in the labels and there was differential imbalance(about 77% of the DEFAULT label is 0), Figure 11.



```
Default == 1: 6636
Default == 0: 23364
```

Figure 11: The Default column(labels)

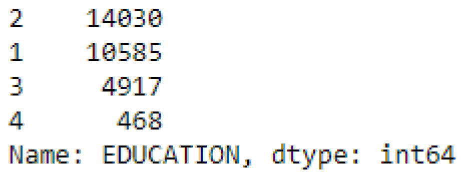
6.3 Data Cleaning

6.3.1 Removing the ID Column

The ID column is removed from the dataset since it is a redundant feature.

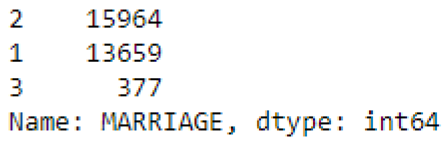
6.3.2 Removing the unnecessary categories in Education and Marriage

In Education, the categories 4, 5 and 6 seems to be the same category. Therefore, I merged it all into category 4, Figure 12. Similarly, with the categories 0 and 3 in Marriage, Figure 13.



```
2    14030
1    10585
3     4917
4      468
Name: EDUCATION, dtype: int64
```

Figure 12: Education



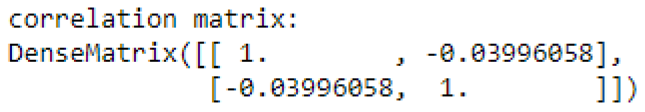
```
2    15964
1    13659
3      377
Name: MARRIAGE, dtype: int64
```

Figure 13: Marriage

- Finally, I renamed the 'PAY_0' column to 'PAY_1' to avoid redundancy.

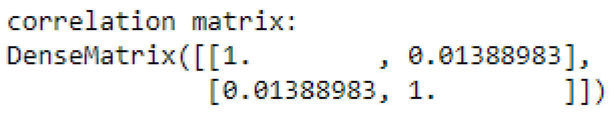
6.4 Feature Correlation

I have checked the correlation of features such as Sex, marriage and age with the default column.



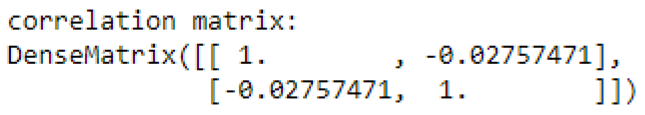
```
correlation matrix:
DenseMatrix([[ 1.        , -0.03996058],
             [-0.03996058,  1.        ]])
```

Figure 14: Correlation matrix between Sex and Default



```
correlation matrix:
DenseMatrix([[1.        , 0.01388983],
             [0.01388983, 1.        ]])
```

Figure 15: Correlation matrix between Age and Default



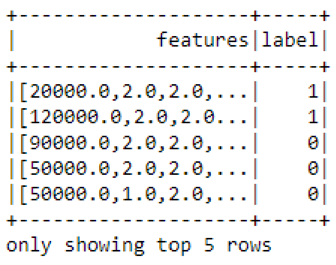
```
correlation matrix:
DenseMatrix([[ 1.        , -0.02757471],
             [-0.02757471,  1.        ]])
```

Figure 16: Correlation matrix between Marriage and Default

7 Training and Evaluation

7.1 Data Preparation

I used Vector assembler to transform the given list of columns into a single vector column to train the ML Models, Figure 17. Then, I split the dataset into trainSet and testSet usig the randomsplit function in Pyspark. I split the dataset into 80% training and 20% testing.



```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|[20000.0,2.0,2.0,...|    1|
|[120000.0,2.0,2.0...|    1|
|[90000.0,2.0,2.0,...|    0|
|[50000.0,2.0,2.0,...|    0|
|[50000.0,1.0,2.0,...|    0|
+--------------------+-----+
only showing top 5 rows
```

Figure 17: Dataset after using vector assembler

7.2 Machine Learning

For training the features, I use three classifiers.

- Logistic Regression: It is a classification algorithm used to assign observations to a discrete set of classes.
- Random Forest: It is an ensemble learning algorithm and that operates by constructing a multitude of decision trees and outputting the set of classes.

- Support Vector Machines: SVM is a supervised machine learning algorithm which uses a technique called the kernel trick to transform the data and then based on these transformations it finds an optimal boundary between the possible outputs.

For evaluating the classification algorithms, I used four metrics:

- Accuracy: It determines the classification accuracy by working out the ratio of number of correct predictions to the total number of input samples.
- Precision: In the classification task, it represents the number of true positives divided by the number of elements labeled as positive.
- Recall: It represents the number of true positives divided by the number of actual true positives.
- Area under ROC: ROC is a probability curve plotted against True positives and False positives. Area under ROC represents degree or measure of separability among the classes.

8 Blind Machine Learning

Initially, I have done machine learning without performing any preprocessing on the dataset. The results are as follows:

- Logistic Regression - Test Area Under ROC: 0.7174, Accuracy: 0.8145, Weighted precision: 0.8002, Weighted recall: 0.8145
- Random Forest Classifier - Test Area Under ROC: 0.7607, Accuracy: 0.8202, Weighted precision: 0.8024, Weighted recall: 0.8202
- SVM Classifier - Test Area Under ROC: 0.6840, Accuracy: 0.7816, Weighted precision: 0.8293, Weighted recall: 0.7816

9 Feature Engineering and Pre-processing

I have performed one-hot encoding on the categorical attributes SEX, EDUCATION and MARRIAGE. One hot encoding prevents bias in categorical attributes. For example, the attribute SEX has values 0 and 1 representing male and female, the ML algorithm might understand that one attribute is more important than the other. Therefore, One hot encoding prevents this problem, Figure 18.

All the attributes are numerical, therefore I have done normalization to all the attributes except the categorical ones. I have normalized them using Standard Scaler. I have used a Data pipeline and a vector assembler to combine all the columns into a single column vector. It is useful for combining raw features and features generated by different feature transformers into a single feature vector, in order to train ML models.

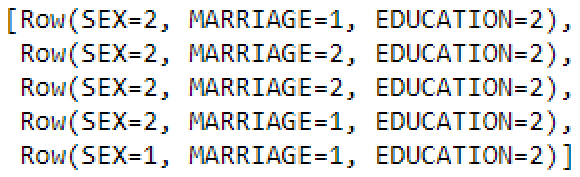

```
[Row(SEX=2, MARRIAGE=1, EDUCATION=2),
 Row(SEX=2, MARRIAGE=2, EDUCATION=2),
 Row(SEX=2, MARRIAGE=2, EDUCATION=2),
 Row(SEX=2, MARRIAGE=1, EDUCATION=2),
 Row(SEX=1, MARRIAGE=1, EDUCATION=2)]
```

Figure 18: Data after One hot encoding

10 ML after Feature Engineering and Pre-processing

- Logistic Regression - Test Area Under ROC: 0.7184, Accuracy: 0.8151, Weighted precision: 0.7977, Weighted recall: 0.8151
- Random Forest Classifier - Test Area Under ROC: 0.7662, Accuracy: 0.8154, Weighted precision: 0.7956, Weighted recall: 0.8154
- Support Vector Machine - Test Area Under ROC: 0.7193, Accuracy: 0.7877, Weighted precision: 0.7825, Weighted recall: 0.7877

11 ML after Upsampling and Downsampling

11.1 Resampling with Pyspark

11.1.1 Upsampling

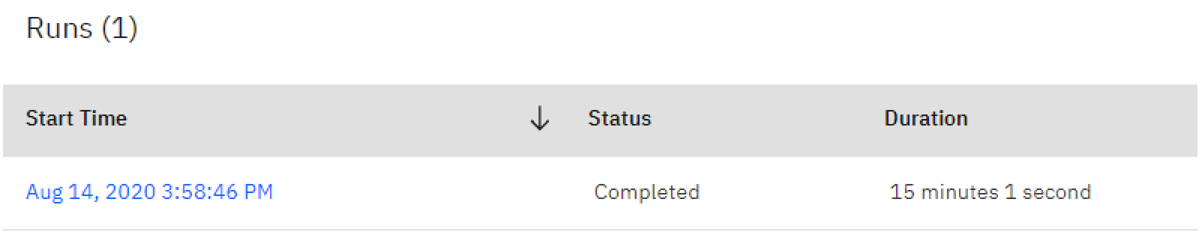
- Logistic Regression - Test Area Under ROC: 0.7201, Accuracy: 0.6691, Weighted precision: 0.6692, Weighted recall: 0.6691
- Random Forest Classifier - Test Area Under ROC: 0.7738, Accuracy: 0.7171, Weighted precision: 0.7279, Weighted recall: 0.7171
- Support Vector Machine - Test Area Under ROC: 0.7193, Accuracy: 0.6744, Weighted precision: 0.6828, Weighted recall: 0.6744

11.1.2 Downsampling

- Logistic Regression - Test Area Under ROC: 0.7003, Accuracy: 0.6498, Weighted precision: 0.6502, Weighted recall: 0.6498
- Random Forest Classifier - Test Area Under ROC: 0.7613, Accuracy: 0.6993, Weighted precision: 0.7142, Weighted recall: 0.6993
- Support Vector Machine - Test Area Under ROC: 0.6958, Accuracy: 0.6629, Weighted precision: 0.6729, Weighted recall: 0.6629

12 Comparison of performance of ML Algorithms on IBM Cloud, Google Cloud Platform and local system

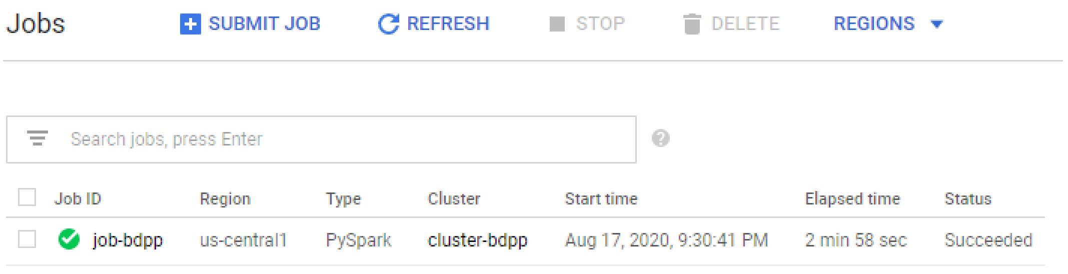
The algorithms ran faster on the cloud platforms compared to my local system. The Job took 15.1 seconds to run on the IBM Cloud, Figure 19. The Job run on GCP took only 2 minutes 58 seconds, Figure 20.



Runs (1)

Start Time	Status	Duration
Aug 14, 2020 3:58:46 PM	Completed	15 minutes 1 second

Figure 19: Job run on IBM Cloud



Job ID	Region	Type	Cluster	Start time	Elapsed time	Status
job-bdpp	us-central1	PySpark	cluster-bdpp	Aug 17, 2020, 9:30:41 PM	2 min 58 sec	Succeeded

Figure 20: Job run on the GCP

ML Algorithms	Local System(sec)	IBM Cloud(sec)	Google Cloud(sec)
Logistic Regression	4.17sec	4.56sec	3.94sec
Random Forest Classifier	19.5sec	9.31sec	5.94sec
Linear Support Vector Classifier	1min 37.6sec	31.5sec	6.57sec

13 Conclusion

13.1 Learnings

- About the Pyspark library and its various functions.
- Working on the IBM Cloud Platform and the Google Cloud Platform, and its various applications.
- Running the Data Analysis and the Machine Algorithms using the Pyspark library on both the cloud platforms

13.2 Comparison of the IBM Cloud and the Google Cloud Platforms

Both the cloud platforms have extensive API's to run the Apache spark framework. On comparison, it was convenient to run the job on IBM cloud since the IBM watson studio consisted of all the functions necessary to run the job unlike on the google cloud platform where each process needed to be run on an individual API. But due to this distributed functionality, algorithms ran faster on GCP than on the IBM cloud.

13.3 Performance of Machine Learning Algorithms

There was a slight improvement of accuracy after feature engineering such as one-hot encoding was performed. Although the class imbalances have been addressed using resampling methods such as col from the Pyspark library, the accuracy has decreased when compared with previous methods. Hence, either the class imbalance has not affected the training accuracy when trained on ML algorithms, or further tuning is needed on the algorithms.