Bigdata Parallel Programming Project Report

Abdul Rehman Basith Shajahan

Introduction:

This is a project on predicting credit card client's default payment status. I have used spark for implementing this project. Spark is an open source computing engine used for processing and analyzing huge data. It distributes the data across the clusters and work on them in parallel. Due to parallel data processing, computational time is faster in spark than in Pandas. I have used Pyspark (python API to support Spark) to build the project.

This project has two phases:

i) Importing all the required libraries and creating a spark session in the local system to perform data analysis and building predictive machine learning models in a Jupyter notebook.

ii) Deploying the built file in a cloud cluster and getting results from the cloud.

I have used Google cloud platform (GCP) for deploying my file in the cloud. I have tired implementing three machine learning algorithms namely Logistic regression, Decision tree and Random forest. And used ROC as a metric to evaluate the performance of each model.

Data:

I have used Default credit card client's dataset which has 25 features/attributes. All the features are integer type. Some preprocessing is done earlier to convert categorical variable like Sex Marriage and Education into numerical variables. Below is the list of feature names and description about each feature.

1.ID = ID of each user
2.LIMIT_BAL = Amount of given credit to the customer
3.SEX = Gender of the customer (Male or female)
4.EDUCATION = Level of education (High school, University, Graduate school)
5.MARRIAGE = Marital status (married or unmarried)
6.AGE = Age of the customer
7.PAY_0 = Repayment status in September 2005
8.PAY_2 = Repayment status in August 2005
9.PAY_3 = Repayment status in July 2005
10.PAY_4 = Repayment status in June 2005
11.PAY_5 = Repayment status in May 2005
12.PAY_6 = Repayment status in April 2005
13.BILL_AMT1 = Amount of bill statement in September 2005
14.BILL_AMT2 = Amount of bill statement in August 2005
15.BILL_AMT3 = Amount of bill statement in July 2005
16.BILL_AMT4 = Amount of bill statement in June 2005
17.BILL_AMT5 = Amount of bill statement in May 2005
18.BILL_AMT6 = Amount of bill statement in April 2005
19.PAY_AMT1 = Amount paid in September 2005

20.PAY_AMT2 = Amount paid in August 2005
21.PAY_AMT3 = Amount paid in July 2005
22.PAY_AMT4 = Amount paid in June 2005
23.PAY_AMT5 = Amount paid in May 2005
24.PAY_AMT6 = Amount paid in April 2005
25.DEFAULT = Default payment of the customer.

Project steps: As mentioned earlier this project has two phases namely the project is built in Jupyter notebook using pyspark. Later I deployed the project in Google cloud platform's Dataproc cluster. Below are steps involved in the project implementation.

PHASE 1: Building project in Jupyter notebook using pyspark.

1. **Creating a spark session and loading data:** Initialized a spark session and loaded the given data into variable named credit.
2. **Data exploration:** In this step we will have look at the data schema, the summary of the data and see how variables are distributed and related with each other.

Schema of data: It can be observed that all the features are integer type.

job-5d7fde02
Start time: 22 May 2020, 16:00:22 Elapsed time: 8 min

Output Configuration

Line wrapping

```
20/05/22 14:03:17 INFO org.apache.hadoop.yarn.clie
20/05/22 14:03:21 INFO org.apache.hadoop.yarn.clie
root
 |-- ID: integer (nullable = true)
 |-- LIMIT_BAL: integer (nullable = true)
 |-- SEX: integer (nullable = true)
 |-- EDUCATION: integer (nullable = true)
 |-- MARRIAGE: integer (nullable = true)
 |-- AGE: integer (nullable = true)
 |-- PAY_0: integer (nullable = true)
 |-- PAY_2: integer (nullable = true)
 |-- PAY_3: integer (nullable = true)
 |-- PAY_4: integer (nullable = true)
 |-- PAY_5: integer (nullable = true)
 |-- PAY_6: integer (nullable = true)
 |-- BILL_AMT1: integer (nullable = true)
 |-- BILL_AMT2: integer (nullable = true)
 |-- BILL_AMT3: integer (nullable = true)
 |-- BILL_AMT4: integer (nullable = true)
 |-- BILL_AMT5: integer (nullable = true)
 |-- BILL_AMT6: integer (nullable = true)
```

First 5 rows in the dataset:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_
0	1	20000	2	2	1	24	2	2	-1	-1	...	0	0	0	0	689	
1	2	120000	2	2	2	26	-1	2	0	0	...	3272	3455	3261	0	1000	
2	3	90000	2	2	2	34	0	0	0	0	...	14331	14948	15549	1518	1500	
3	4	50000	2	2	1	37	0	0	0	0	...	28314	28959	29547	2000	2019	
4	5	50000	1	2	1	57	-1	0	-1	0	...	20940	19146	19131	2000	36681	

5 rows × 25 columns

Summary of the data:

	0	1	2	3	4
summary	count	mean	stddev	min	max
ID	30000	15000.5	8660.398374208891	1	30000
LIMIT_BAL	30000	167484.32266666667	129747.66156720246	10000	1000000
SEX	30000	1.6037333333333332	0.4891291960902602	1	2
EDUCATION	30000	1.8531333333333333	0.7903486597207269	0	6
MARRIAGE	30000	1.5518666666666667	0.5219696006132467	0	3
AGE	30000	35.4855	9.217904068090155	21	79
PAY_0	30000	-0.0167	1.1238015279973335	-2	8
PAY_2	30000	-0.13376666666666667	1.1971859730345495	-2	8
PAY_3	30000	-0.1662	1.1968675684465686	-2	8
PAY_4	30000	-0.22066666666666668	1.1691386224023357	-2	8
PAY_5	30000	-0.2662	1.1331874060027525	-2	8
PAY_6	30000	-0.2911	1.149987625607897	-2	8
BILL_AMT1	30000	51223.3309	73635.86057552966	-165580	964511

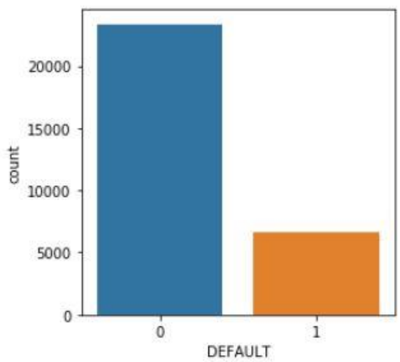
From the summary and looking manually into the dataset I have inferred three points:

1. It can be observed that marriage has a label '0' but in the description of data it is given marriage has values 1,2 and 3.
2. Education column has 0, 5 and 6 as labels which are not specified in the description of data.
3. We can see that all the columns have -2 as minimum. But in the description of data it given that -1 indicates pay duly. There is no information regarding -2.

We can deal with these during data cleaning

Check number of classes in target feature (DEFAULT) is balanced or imbalanced: It can be observed that class ‘0’ is almost 3.5 times more than class ‘1’. Due to this imbalance in classes the classification models will have low predictive accuracy on the minor class. First, I will try to build models on imbalanced target feature dataset and then I will try to handle the imbalance in classes and run the same models on modified data to see the difference.

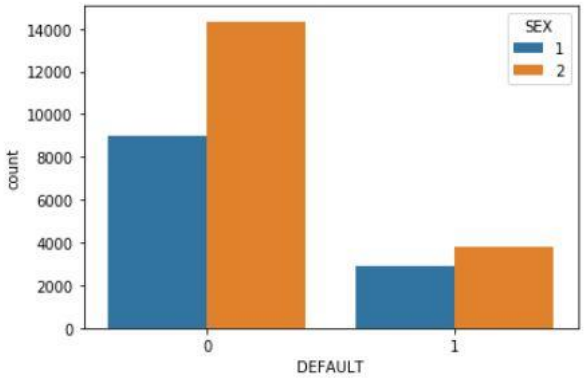
```
20/05/22 14:03:50
+-------+-----+
|DEFAULT|count|
+-------+-----+
|      1| 6636|
|      0|23364|
+-------+-----+
```



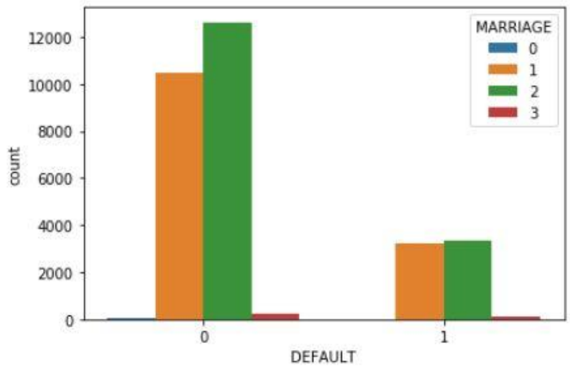
Count of default based on gender, Marriage status and education:

i)

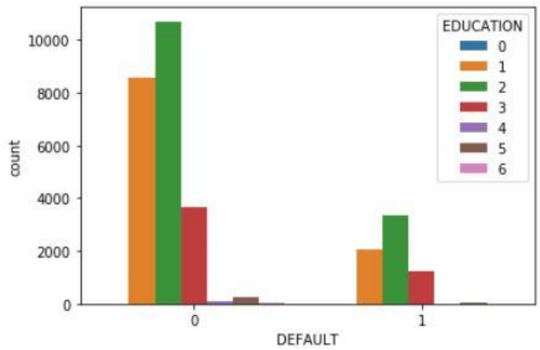
Default	SEX	count
1	2	3763
1	1	2873
0	1	9015
0	2	14349



Default	Marriage	count
1	0	5
1	2	3341
1	1	3206
1	3	84
0	0	49
0	1	10453
0	2	12623
0	3	239



Default	Education	count
1	2	3330
0	5	262
1	1	2036
1	3	1237
0	0	14
1	4	7
0	1	8549
1	5	18
0	6	43
0	2	10700
0	3	3680
1	6	8
0	4	116



Let's look at the correlation btw the default feature and remaining features: Correlation coefficient ranges from -1 to 1. When the value is close to 1 it means that there is a positive correlation between the features. And when the value is close to -1 it means there is a negative correlation between the features. Here we can observe that Default feature (target feature) is positively correlated with PAY_n, education and age features while it is negatively related with all the other features.

```
('Correlation to Default for ', 'ID', -0.013951954838986256)
('Correlation to Default for ', 'LIMIT_BAL', -0.1535198763935072)
('Correlation to Default for ', 'SEX', -0.03996057770544174)
('Correlation to Default for ', 'EDUCATION', 0.02800607765625021)
('Correlation to Default for ', 'MARRIAGE', -0.024339215683404455)
('Correlation to Default for ', 'AGE', 0.013889834301962877)
('Correlation to Default for ', 'PAY_0', 0.32479372847862237)
('Correlation to Default for ', 'PAY_2', 0.26355120167216783)
('Correlation to Default for ', 'PAY_3', 0.2352525137249171)
('Correlation to Default for ', 'PAY_4', 0.2166136368424239)
('Correlation to Default for ', 'PAY_5', 0.2041489138761644)
('Correlation to Default for ', 'PAY_6', 0.18686636165354611)
('Correlation to Default for ', 'BILL_AMT1', -0.019644197143221576)
('Correlation to Default for ', 'BILL_AMT2', -0.014193218088215746)
('Correlation to Default for ', 'BILL_AMT3', -0.014075518043214713)
('Correlation to Default for ', 'BILL_AMT4', -0.010156495880289678)
('Correlation to Default for ', 'BILL_AMT5', -0.006760463841014792)
('Correlation to Default for ', 'BILL_AMT6', -0.005372314914815569)
('Correlation to Default for ', 'PAY_AMT1', -0.07292948777785163)
('Correlation to Default for ', 'PAY_AMT2', -0.058578706582901575)
('Correlation to Default for ', 'PAY_AMT3', -0.056250350990331634)
('Correlation to Default for ', 'PAY_AMT4', -0.05682740089288694)
('Correlation to Default for ', 'PAY_AMT5', -0.055123515621088755)
('Correlation to Default for ', 'PAY_AMT6', -0.053183340326127954)
('Correlation to Default for ', 'DEFAULT', 1.0)
```

3.Data cleaning: Let us 1st fix the three flaws which we noted down earlier in the summary of the data.

1. Assign '0' to other label in marriage column i.e., change all '0' to '3'. After transforming we will have only 3 distinct values in marriage column.
2. Assign '0','5' & '6' to other label in education column i.e., change all '0','5' & '6' to '4'. After transforming we will have 4 distinct values in education column.
3. And I don't want negative values in Pay_x columns so I will change instances with '-2' to 0. Which means duly paid. After transforming we will have 9 to 8 distinct values in Pay_n column's

```
+------------------------+      +-------------------------+
|count(DISTINCT MARRIAGE)|      |count(DISTINCT EDUCATION)|
+------------------------+      +-------------------------+
|                       4|      |                        7|
+------------------------+      +-------------------------+

+------------------------+      +-------------------------+
|count(DISTINCT MARRIAGE)|      |count(DISTINCT EDUCATION)|
+------------------------+      +-------------------------+
|                       3|      |                        4|
+------------------------+      +-------------------------+
```

```
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
|count(DISTINCT PAY_0)|count(DISTINCT PAY_2)|count(DISTINCT PAY_3)|count(DISTINCT PAY_4)|count(DISTINCT PAY_5)|count(DISTINCT PAY_6)|
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
|                  11|                  11|                  11|                  11|                  10|                  10|
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+

+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
|count(DISTINCT PAY_0)|count(DISTINCT PAY_2)|count(DISTINCT PAY_3)|count(DISTINCT PAY_4)|count(DISTINCT PAY_5)|count(DISTINCT PAY_6)|
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
|                   9|                   9|                   9|                   9|                   8|                   8|
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
```

Missing values: Let's check if there are any missing values in the data.

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0

1 rows × 25 columns

Since we don't have any missing values we can proceed further.

4. **Feature scaling:** We need to scale the data before passing to any machine learning model. First, I have created a big vector called features by combining all the features in the dataset. To do this I have used VectorAssembler () transformer from pyspark. And then I used a StandardScaler () to get a scaled feature column. Below is the output of combined feature vector and scaled feature vector.

```
+--------------------+--------------------+
|            features|      scaledFeatures|
+--------------------+--------------------+
|(25,[0,1,2,3,4,5,...|[-1.7319642067123...|
|[2.0,120000.0,2.0...|[-1.7318487385830...|
|[3.0,90000.0,2.0,...|[-1.7317332704536...|
|[4.0,50000.0,2.0,...|[-1.7316178023242...|
|[5.0,50000.0,1.0,...|[-1.7315023341948...|
+--------------------+--------------------+
only showing top 5 rows
```

Create final dataset: We extract only features (Scaled features) and labels column for giving input to the models.

```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|[-1.7319642067123...|    1|
|[-1.7318487385830...|    1|
|[-1.7317332704536...|    0|
|[-1.7316178023242...|    0|
|[-1.7315023341948...|    0|
+--------------------+-----+
only showing top 5 rows
```

5. **Model training and prediction:**

Splitting data into training and testing sets: I have divided the dataset into 80% and 20% for training and testing respectively. Below is the count of data in train and test sets.

```
Training Dataset Count: 23861
Test Dataset Count: 6139
```

Logistic Regression: I have trained a logistic regression model with the following parameters 1. maxIter = 10, 2. regParam = 0.3, elasticNetParam = 0.8. I have used areaUnder ROC as the metric to measure the performance of the model. In addition, I have calculated time to train predict and evaluation of model. Below are the performance details of logistic regression.

```
('Training accuracy for Logistic Regression: ', 0.7811072461338586)
Time to train Logistic Regression model: 4.589 seconds
Train areaUnderROC for Logistic Regression: 0.5
+--------------------+--------------------+----------+-----+
|            features|         probability|prediction|label|
+--------------------+--------------------+----------+-----+
|[-1.7315023341948...|[0.78110724613385...|       0.0|    0|
|[-1.7311559298066...|[0.78110724613385...|       0.0|    0|
|[-1.7310404616773...|[0.78110724613385...|       0.0|    0|
|[-1.7302321847716...|[0.78110724613385...|       0.0|    0|
|[-1.7301167166422...|[0.78110724613385...|       0.0|    1|
+--------------------+--------------------+----------+-----+
only showing top 5 rows

+-----+----------+-----+
|label|prediction|count|
+-----+----------+-----+
|    1|       0.0| 1413|
|    0|       0.0| 4726|
+-----+----------+-----+

Time for prediction Logistic Regression model: 2.179 seconds
('Test areaUnderROC for Logistic Regression: ', 0.5)
('Test accuracy for Logistic Regression: ', 0.769832220231308)
Time to evaluate Logistic Regression model: 2.437 seconds
```

Decision Tree: I have trained a decision tree model with maxDepth =3. I have used areaUnder ROC as the metric to measure the performance of the model. In addition, I have calculated time to train predict and evaluation of model. Below are the performance details of decision tree.

```
Time to train Decission Tree model: 4.698 seconds
+--------------------+--------------------+----------+-----+
|            features|         probability|prediction|label|
+--------------------+--------------------+----------+-----+
|[-1.7315023341948...|[0.88404886561954...|       0.0|    0|
|[-1.7311559298066...|[0.78734984984984...|       0.0|    0|
|[-1.7310404616773...|[0.78734984984984...|       0.0|    0|
|[-1.7302321847716...|[0.63099921321793...|       0.0|    0|
|[-1.7301167166422...|[0.88404886561954...|       0.0|    1|
+--------------------+--------------------+----------+-----+
only showing top 5 rows
```

```
+-----+----------+-----+
|label|prediction|count|
+-----+----------+-----+
|    1|       0.0|  873|
|    0|       0.0| 4466|
|    1|       1.0|  540|
|    0|       1.0|  260|
+-----+----------+-----+

Time for prediction Decission Tree model: 1.823 seconds
('Test_roc for Decission Tree:', 0.2669554876892791)
('Test accuracy for Decisson Tree: ', 0.8154422544388337)
Time to evaluate Decission Tree model: 2.172 seconds
```

Random Forest: I have trained a logistic regression model with numTrees = 8. I have used areaUnder ROC as the metric to measure the performance of the model. In addition, I have calculated time to train predict and evaluation of model. Below are the performance details of logistic regression.

```
Time to train Random Forest model: 4.703 seconds
+--------------------+--------------------+----------+-----+
|            features|         probability|prediction|label|
+--------------------+--------------------+----------+-----+
|[-1.7315023341948...|[0.84985931695129...|       0.0|    0|
|[-1.7311559298066...|[0.82010994974235...|       0.0|    0|
|[-1.7310404616773...|[0.70595115745883...|       0.0|    0|
|[-1.7302321847716...|[0.64124228021187...|       0.0|    0|
|[-1.7301167166422...|[0.47347644227806...|       1.0|    1|
+--------------------+--------------------+----------+-----+
only showing top 5 rows

+-----+----------+-----+
|label|prediction|count|
+-----+----------+-----+
|    1|       0.0|  949|
|    0|       0.0| 4502|
|    1|       1.0|  464|
|    0|       1.0|  224|
+-----+----------+-----+

Time for prediction Random Forest model: 1.953 seconds
('Test_roc for Random Forest:', 0.7767454825948163)
('Test accuracy for Random Forest: ', 0.8089265352663301)
Time to evaluate Random Forest model: 2.015 seconds
```

6. Model selection a.k.a. hyperparameter tuning:

Best model for Logistic regression: I tried to get the best model for logistic regression with the help of hyperparameter tuning. Build a paraGrid with 0.1 and 0.01 values for regParam. Used a 10-fold cross validator to get the best model. Below is the performance of the best logistic model for the given parameters.

```
Time to train Logistic Regression best model: 34.720 seconds
+--------------------+--------------------+----------+-----+
|            features|         probability|prediction|label|
+--------------------+--------------------+----------+-----+
|[-1.7315023341948...|[0.83971676529429...|       0.0|    0|
|[-1.7311559298066...|[0.85090943860679...|       0.0|    0|
|[-1.7310404616773...|[0.81287756661542...|       0.0|    0|
|[-1.7302321847716...|[0.69323792065982...|       0.0|    0|
|[-1.7301167166422...|[0.69426190085287...|       0.0|    1|
+--------------------+--------------------+----------+-----+
only showing top 5 rows

+-----+----------+-----+
|label|prediction|count|
+-----+----------+-----+
|    1|       0.0|  957|
|    0|       0.0| 4522|
|    1|       1.0|  456|
|    0|       1.0|  204|
+-----+----------+-----+

Time for predicting Logistic Regression best model: 1.597 seconds
('Test_roc for Logistic Regression best model:', 0.761505969446995)
('Test accuracy for Logistic Regression best model: ', 0.8108812510180812)
Time for evaluating Logistic Regression best model: 1.794 seconds
```

Best model for Random forest: I even tried to get the best random forest model by building a paramgrid with the following parameters: maxDepth = [1,2,3,4 5], minInstancesPerNode = [1,2,3,4,5]. The best model was selected using a 10-fold cross validator. Below are the details of the performance of best random forest model for given parameters.

```
Time to train Random Forest best model: 186.151 seconds
+--------------------+--------------------+----------+-----+
|            features|         probability|prediction|label|
+--------------------+--------------------+----------+-----+
|[-1.7315023341948...|[0.85035623632889...|       0.0|    0|
|[-1.7311559298066...|[0.81902010875014...|       0.0|    0|
|[-1.7310404616773...|[0.67261826908830...|       0.0|    0|
|[-1.7302321847716...|[0.61059012834984...|       0.0|    0|
|[-1.7301167166422...|[0.52382315709409...|       0.0|    1|
+--------------------+--------------------+----------+-----+
only showing top 5 rows

+-----+----------+-----+
|label|prediction|count|
+-----+----------+-----+
|    1|       0.0|  974|
|    0|       0.0| 4516|
|    1|       1.0|  439|
|    0|       1.0|  210|
+-----+----------+-----+

Time for predicting Random Forest best model: 1.437 seconds
('Test_roc for Random Forest best model:', 0.7762950374058195)
('Test accuracy for Random Forest best model: ', 0.8071347124938915)
Time for evaluating Random Forest best model: 1.507 seconds
```

7. **Feature engineering:** The target feature i.e., DEFAULT column has imbalanced number of classes. Model trained on this data will be biased to the majority class. This can be dealt by three methods.
    1. Up sampling: Increasing the minority class by replicating them number of times in the dataset.
    2. Down sampling: Remove few rows with majority class so that there are not much difference btw majority and minority classes.
    3. Assigning weights: Penalize majority class by assigning less weight to them and increase the value of minority class by assigning high weight to them.

    In my project I have tried weight assigning technique. Created a column named weights with two values 0.85 and 0.15 for minority and majority classes respectively. Below is the weights column.

	weights
0	0.85
1	0.85
2	0.15
3	0.15
4	0.15
...	...
29995	0.15
29996	0.15
29997	0.85
29998	0.85
29999	0.85

    30000 rows × 1 columns

8. **Training models with balanced target feature data:** After assigning weights I built three models namely logistic regression, decision tree and random forest. I used the same parameters which I used in the earlier models.

    Logistic regression:

```
('Training accuracy for Logistic Regression on weighted target variable: ', 0.7811072461338586)
Train areaUnderROC for Logistic Regression on weighted target variable: 1.0
+--------------------+--------------------+----------+-----+
|            features|         probability|prediction|label|
+--------------------+--------------------+----------+-----+
|[-1.7315023341948...|[0.85156346063794...|       0.0|    0|
|[-1.7311559298066...|[0.85156346063794...|       0.0|    0|
|[-1.7310404616773...|[0.85156346063794...|       0.0|    0|
|[-1.7302321847716...|[0.85156346063794...|       0.0|    0|
|[-1.7301167166422...|[0.52969590339367...|       0.0|    1|
+--------------------+--------------------+----------+-----+
only showing top 5 rows

('Test areaUnderROC for Logistic Regression on weighted target variable: ', 1.0)
('Test accuracy for Logistic Regression on weighted target variable: ', 0.769832220231308)
```

Decision Tree:

```
+--------------------+-----------+----------+-----+
|            features|probability|prediction|label|
+--------------------+-----------+----------+-----+
|[-1.7315023341948...|  [1.0,0.0]|       0.0|    0|
|[-1.7311559298066...|  [1.0,0.0]|       0.0|    0|
|[-1.7310404616773...|  [1.0,0.0]|       0.0|    0|
|[-1.7302321847716...|  [1.0,0.0]|       0.0|    0|
|[-1.7301167166422...|  [0.0,1.0]|       1.0|    1|
+--------------------+-----------+----------+-----+
only showing top 5 rows

('Test_roc for Decision tree on weighted target variable:', 1.0)
('Test accuracy for Decision tree on weighted target variable: ', 1.0)
```

Random forest:

```
+--------------------+--------------------+----------+-----+
|            features|         probability|prediction|label|
+--------------------+--------------------+----------+-----+
|[-1.7315023341948...|[0.89736533425237...|       0.0|    0|
|[-1.7311559298066...|[0.92257277921544...|       0.0|    0|
|[-1.7310404616773...|[0.91793873793916...|       0.0|    0|
|[-1.7302321847716...|[0.82748536665930...|       0.0|    0|
|[-1.7301167166422...|[0.44556603796344...|       1.0|    1|
+--------------------+--------------------+----------+-----+
only showing top 5 rows

('Test_roc for Random forest on weighted target variable:', 0.999585494586721)
('Test accuracy for Random forest on weighted target variable: ', 0.9555302166476625)
20/05/22 14:09:02 INFO org.spark_project.jetty.server.AbstractConnector: Stopped Spark@
```

Results:

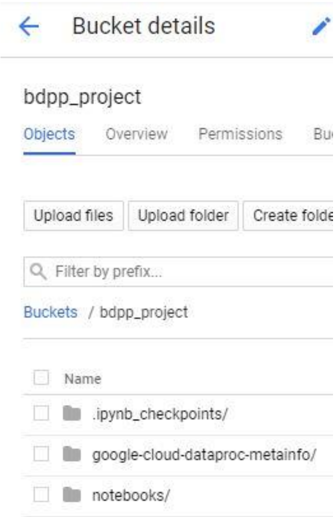
	Before feature engineering	Best Model	After feature engineering
Model	ROC	ROC	ROC
Logistic Regression	0.5	0.761	1
Decision Tree	0.266	---	1
Random Forest	0.776	0.776	0.999

PHASE 2: Deploying built project in cloud (GCP)

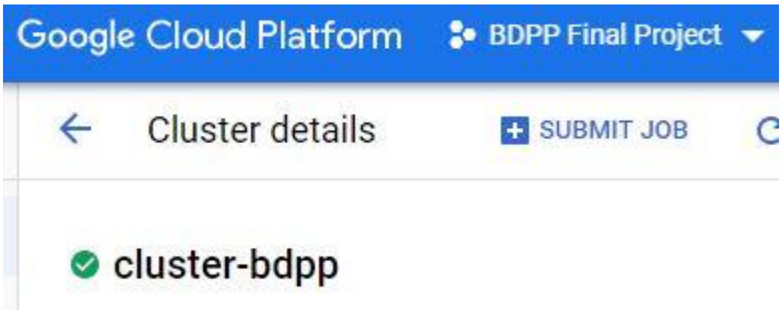
- Create and account in GCP and build a new project. I have named my project as BDPP Final Project.

- We need to enable two API's namely Compute engine and Storage to deploy our project in cloud.

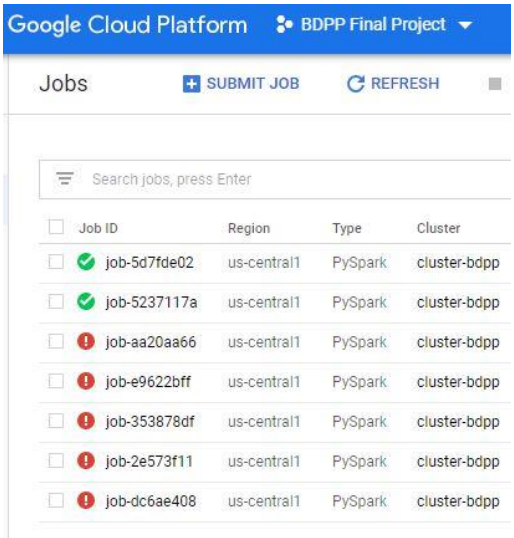
- Then create a bucket in storage API to upload and store the local files in cloud. I created a bucket named bdpp_project to store upload and store my project file and dataset.



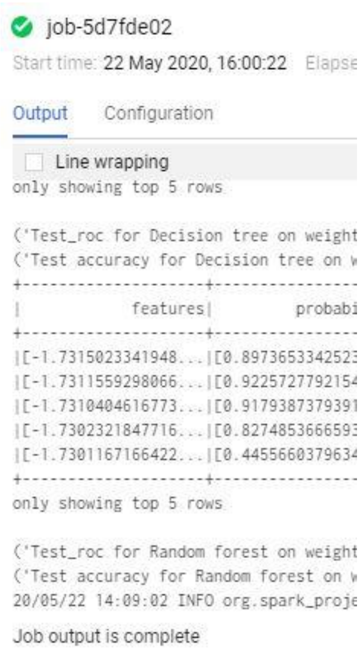
- Then we need to create a cluster to run our project file. Dataproc is an API provided by GCP to create clusters. I have created a cluster named cluster-bdpp which has 1 master node and 2 worker nodes.



- Now we need to submit our project file as a job in Dataproc cluster. We need to submit the file with .py extension in order to run the job successfully.



- As the job runs, we can see the outputs/results of our project in the output tab of that job.



Conclusion: Based on the ROC results of the models with hyperparameter tuning logistic regression has a considerable increase in ROC value. But random forest (without and with parameter tuning) has achieved better results compared to logistic regression.

I am quite puzzled to see 1 and 0.99 values for all the models after assigning weights to the classes. May be assigning weights helped models in predicting the classes correctly. Since the models have almost 100% ROC, I did not do parameter tuning to select the best model. Since assigning weights gave better results, I would suggest dealing with imbalanced classes in the dataset would increase the performance of the models.