DISTRIBUTED DATA ANALYSIS AND MINING

CREDIT CARD FRAUD PREDICTION

AKHIL VARMA DANTULURI

CHIRANJIV MADAN

MUHAMMAD UMAIR NAEEM

SUPERVISED BY
PROF. ROBERTO TRASARTI

INTRODUCTION

The purpose of this paper is to predict credit card fraud, the dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days. The main purpose of this paper is to predict if a given credit card transaction is fraudulent or genuine. If a fraud is detected it has been tagged as 1, and 0 if otherwise.

The first task was to perform basic data understanding, after that we proceeded to perform a cluster analysis(K-Means), to group transactions into different categories for further analysis.

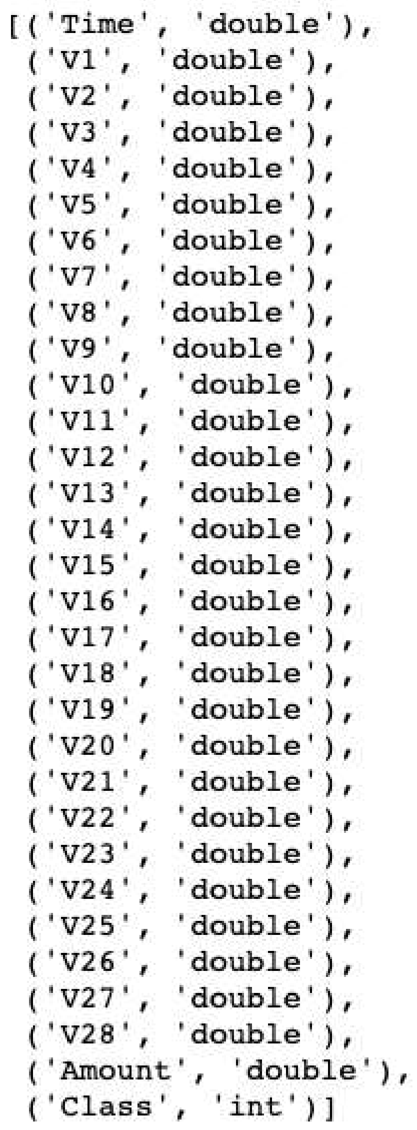
Additionally, different binary classifiers were built to predict whether the transactions were ***Genuine*** or ***Fraudulent*** (LogisticRegression, Decision Tree, Random Forest, Gradient-Boosted Tree). Also as the evaluation measure for different classifiers, we chose the ROC curve, since the data set was unbalanced. We also balanced the data by oversampling the minor class and the difference in results can be seen.

DATA UNDERSTANDING

The dataset we explored can be found on Kaggle. This dataset consists of the transactions made by credit cards in September 2013 by European cardholders. This dataset shows the transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions (Table 1). The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions. It contained only numeric input variables which were the result of a PCA transformation. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which had not been transformed with PCA were 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

The credit card transactions are mostly valid.

This dataset contains Number of rows: 284807, number of features: 31: all the features are numeric data in our case. Our data type is as follows (Figure 1)



```
[('Time', 'double'),
 ('V1', 'double'),
 ('V2', 'double'),
 ('V3', 'double'),
 ('V4', 'double'),
 ('V5', 'double'),
 ('V6', 'double'),
 ('V7', 'double'),
 ('V8', 'double'),
 ('V9', 'double'),
 ('V10', 'double'),
 ('V11', 'double'),
 ('V12', 'double'),
 ('V13', 'double'),
 ('V14', 'double'),
 ('V15', 'double'),
 ('V16', 'double'),
 ('V17', 'double'),
 ('V18', 'double'),
 ('V19', 'double'),
 ('V20', 'double'),
 ('V21', 'double'),
 ('V22', 'double'),
 ('V23', 'double'),
 ('V24', 'double'),
 ('V25', 'double'),
 ('V26', 'double'),
 ('V27', 'double'),
 ('V28', 'double'),
 ('Amount', 'double'),
 ('Class', 'int')]
```

Figure 1

The time range of transactions was 2 days.

Features **V1, V2, … V28** are the principal components obtained with PCA, the only features which had not been transformed with PCA were '**Time**' and **'Amount', PCA** was not done by us because the data set we have explored was already processed with PCA.

the data set did not contains any missing values and duplicate rows

Data transformation

In order to do data transformation, we already had predictive features 0 and 1. Feature selection for clustering as per the correlation map (Figure 2),

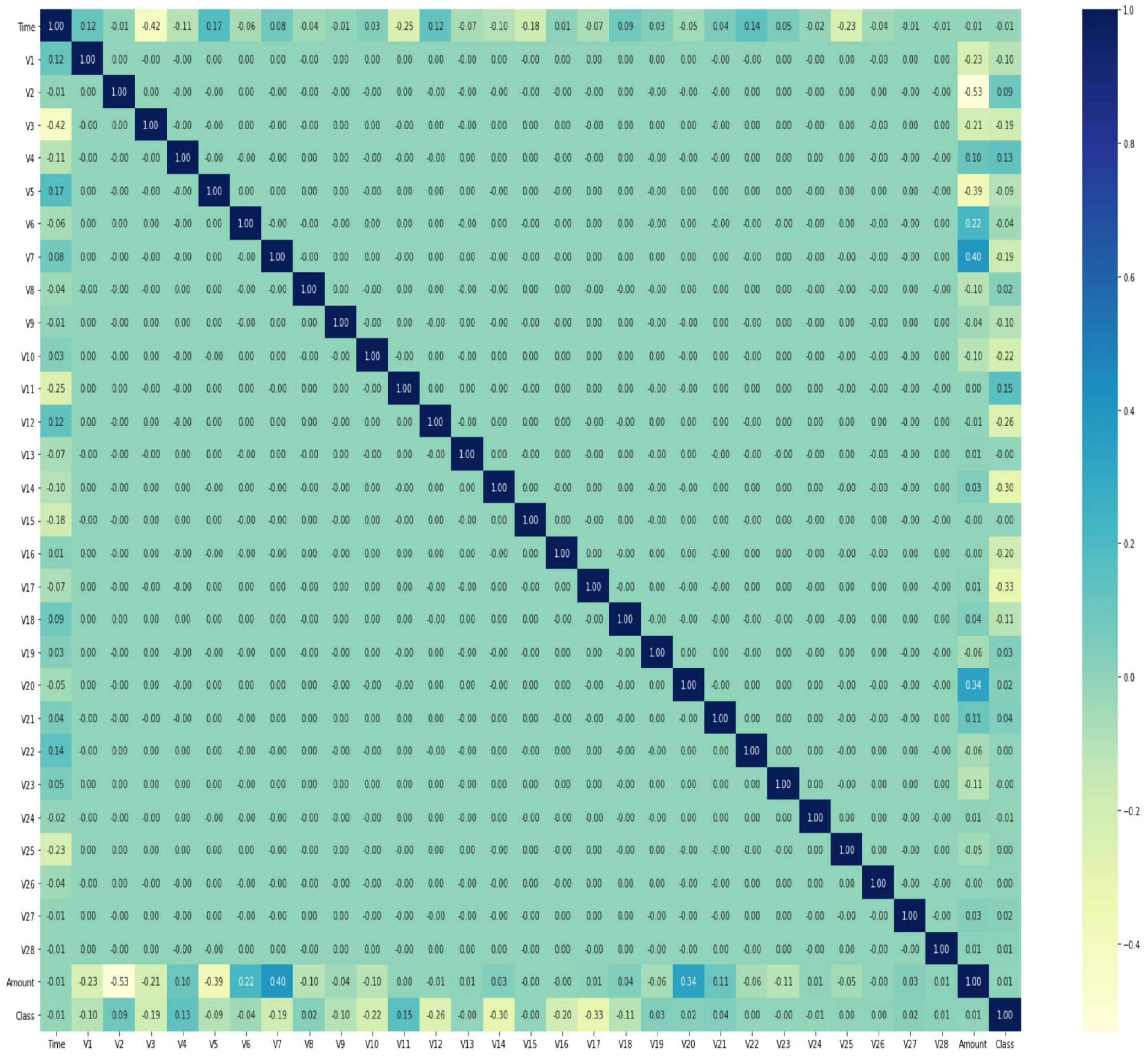
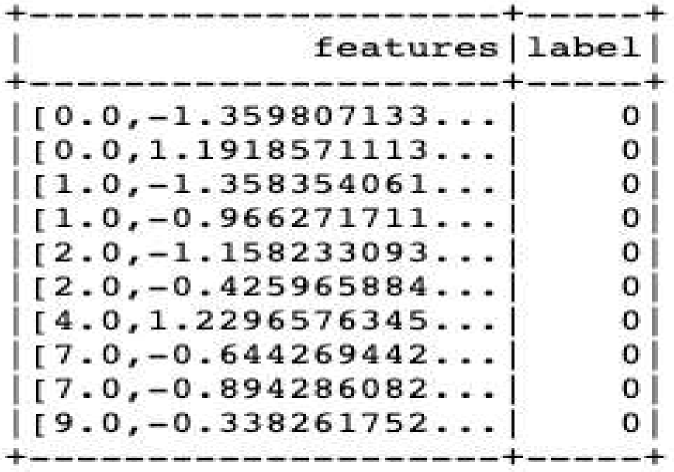


Figure 2

Correlation between features.

As we can see the most of the features have some kind of correlation with the predictive class

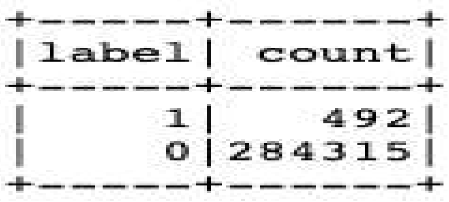
We have to prepare the data, in order to apply our models, in such a way that it is in the format that is supported by the library i.e pyspark in our case. We have all of our features in the numerical format, so now we are supposed to combine them so we will have two columns that we will then feed into the model which are in our case ‘features’ and ‘label’. For this purpose we again used a built-in tool from the pyspark library called VectorAssembler.



```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|[0.0,-1.359807133...|    0|
|[0.0,1.1918571113...|    0|
|[1.0,-1.358354061...|    0|
|[1.0,-0.966271711...|    0|
|[2.0,-1.158233093...|    0|
|[2.0,-0.425965884...|    0|
|[4.0,1.2296576345...|    0|
|[7.0,-0.644269442...|    0|
|[7.0,-0.894286082...|    0|
|[9.0,-0.338261752...|    0|
+--------------------+-----+
```

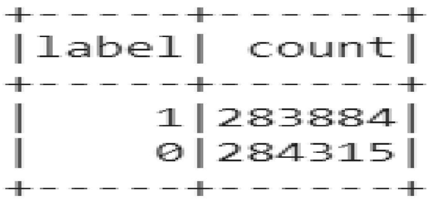
Table 3

Oversampling for classification - Oversampling was done to balance the data, since the dataset provided was unbalanced.The ratio between major and minor class of data, before oversampling (table 1) and later on (table 2) was `ratio: 577` and `ratio: 1` respectively, the following table shows balanced data.



```
+-----+------+
|label| count|
+-----+------+
|    1|   492|
|    0|284315|
+-----+------+
```

Table 1



```
+-----+------+
|label| count|
+-----+------+
|    1|283884|
|    0|284315|
+-----+------+
```

Table 2

Training Split - The dataset was divided into a training set and a test set in the ratio of 80:20. Oversampling of the training set was done to increase the ratio of fraud transactions in the data, to achieve a ratio of 1:1 between the two types of classes in the training set, to improve the evaluation measure of the binary classifiers. We put seed for random split at 23 in both the cases but in case of balanced data we oversampled only the training data,we had 56821 of entries in our test set which is same for both balanced and unbalanced

MODEL EVALUATION

Logistic Regression

We used Logistic Regression from the *pyspark.ml.classification* in our case.
because our dataset is unbalanced (as shown in table 1 and 2). We will first train our model on this unbalanced dataset and then we will create a new dataset using oversampling methods on minority class in our case “1” and then see if balancing the dataset had any impact on the model.

Table 4 showing difference between Balanced and Unbalanced Data

Label	**Prediction**	**Count**	
		Balanced	**Unbalance**
1	0	7	32
0	0	55345	56715
1	1	88	63
0	1	1381	11

As we can see that modelgets better at predicting the fraudulent transactions but at the cost of creating more false positive which increased from 11 in unbalanced to 1381 in case of balanced data but it gives us a better performance in case of false negative as it has reduced more than 4.5 times.

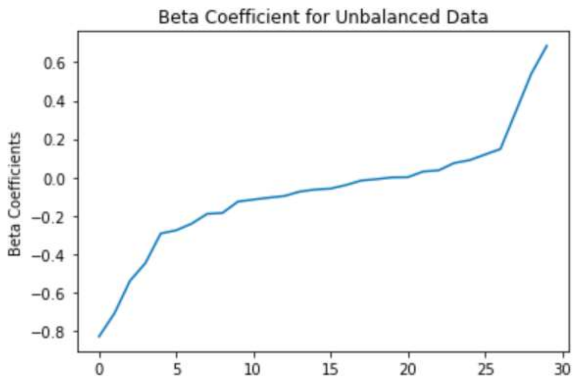


Figure 3

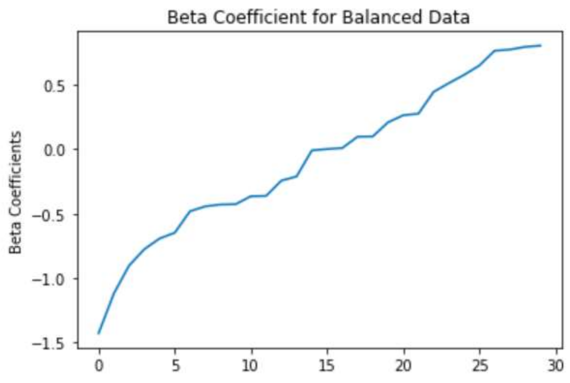
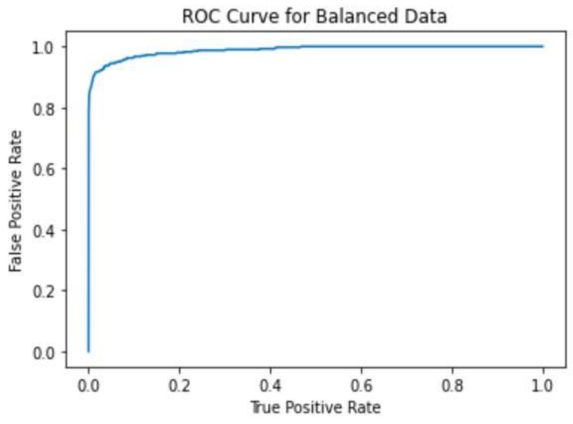


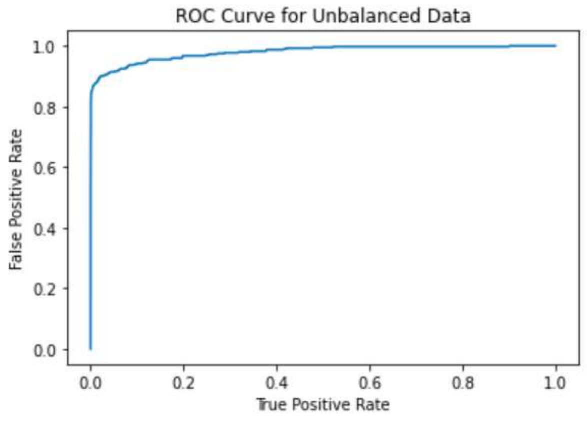
Figure 4

As we can see in figures 3 and 4 that Beta Coefficient seems to stay at 0 for longer in unbalanced than for the balanced set. Precision and recall has a more significant difference in figures 5 and 6. As for the ROC curve we can see that the balanced



```
Training set areaUnderROC: 0.9879661161260987
```

Figure 5



```
Training set areaUnderROC: 0.9796617888008002
```

Figure 6

dataset proves more area under the curve figures 7 and 8.

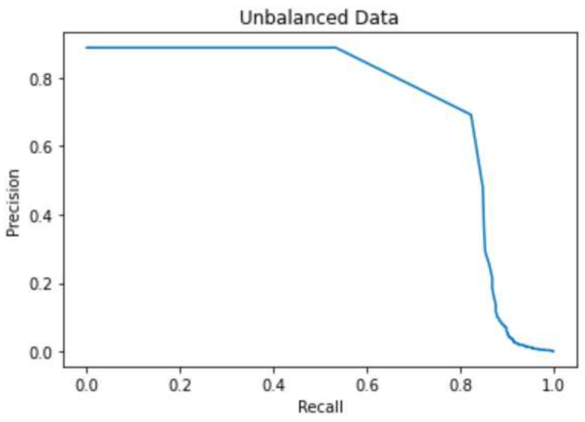


Figure 7

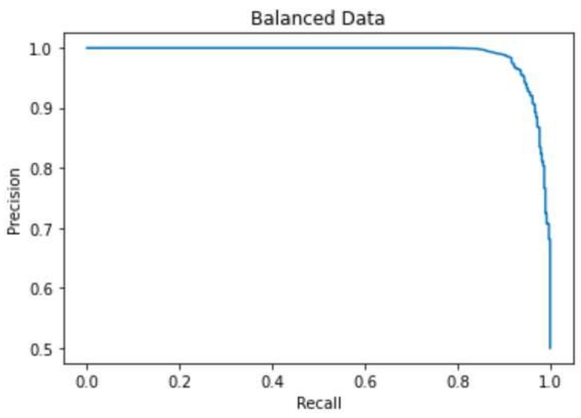


Figure 8

Decision Tree

We took the above prepared dataset and then we applied the decision tree classifier from pyspark.ml.classification and we faced the same issue as previously with using the unbalanced dataset for our tree model.explainParam values of depth 0 with 1 nodes in case of
We took above prepared dataset and applied decision tree classifier from pyspark.ml.classification and we observed the same problem as before with using the unbalanced dataset our tree model.explainParam values of depth 0 with 1 nodes in case of unbalanced dataset and depth 5 with 29 nodes clear indication that model was not learning from data. For Test Area Under ROC we have values of 0.79 and 0.89 unbalanced and balanced respectively.

Table 5 showing difference between Balanced and Unbalanced Data

Label	**Prediction**	**Count**	
		Balanced	**Unbalance**
1	0.0	11	18
0	0.0	54760	56716
1	1.0	84	77
0	1.0	1966	10

Random Forest

We took above prepared dataset and applied decision tree classifier from pys-park.ml.classification and we observed the same problem as before with using the unbalanced dataset our tree model.explainParam values of depth 0 with 1 nodes in case of unbalanced dataset and depth 5 with 35 nodes clear indication that model was not learning from data. For Test Area Under ROC we have values of 0.95 and 0.99 unbalanced and balanced respectively.

Gradient-Boosted Tree

We applied the same process on the Gradient-Boosted Tree with 5 Folds Cross ,which improved the ROC score of 0.96 for balanced and 0.98 for unbalanced to ROC score of 0.98 for balanced and 0.99 for unbalanced with 5 Folds Cross.

Table 6 showing difference in Area under ROC between Balanced and Unbalanced Data

Model Name	**Area Under ROC**	
	Balanced	**Unbalance**
Logistic Regression	0.98	0.98
Decision Tree	0.90	0.79
Random Forest	0.99	0.95
Gradient-Boosted Tree (Cross validation)	0.96 (0.98)	0.99 (0.99)

Conclusions and Future Work

In the classification task, the balanced data would do well to assess markers on a similar classification algorithm, with the Random Forest having a superior score.

A potential future turn of events, for instance, fraud or valid transactions can be arranged independently utilizing a classification algorithm, to realize which sort of transactions are probably going to be the bad ones.