**Big Data Machine Learning using Spark MLlib**

**Submitted by:**

VenkataSaiGoutham Gonuguntla, [vgonuguntla@hawk.iit.edu](mailto:vgonuguntla@hawk.iit.edu), A20450688

Harsh Dave, [hdave3@hawk.iit.edu](mailto:hdave3@hawk.iit.edu), A20452142

FNU Deepanshu, [fdeepanshu@hawk.iit.edu](mailto:fdeepanshu@hawk.iit.edu), A20449479

Khanderao VenkataSaiAkshayKishore, [vkhanderao@hawk.iit.edu](mailto:vkhanderao@hawk.iit.edu), A20458999

**Literature Review**

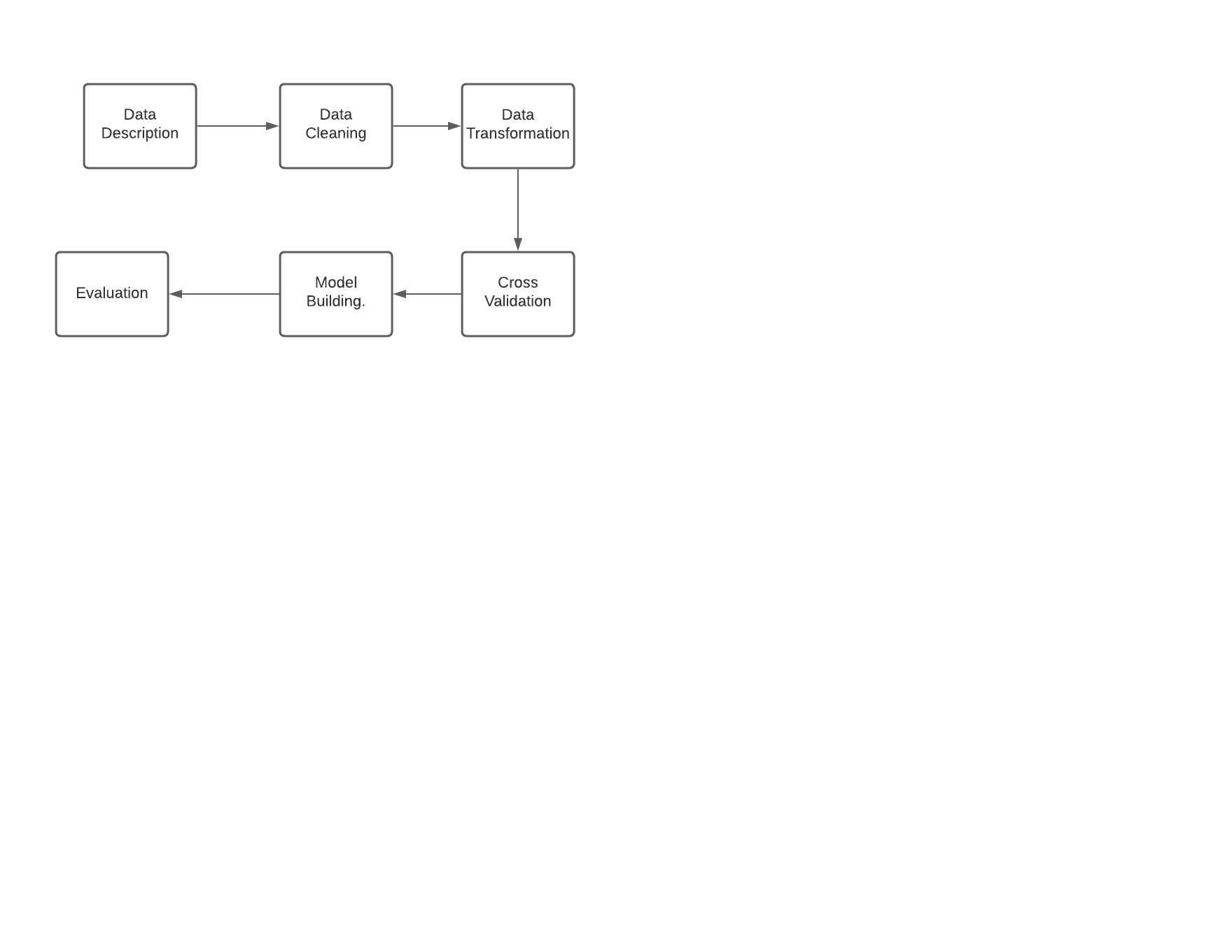
Machine learning is an algorithm that performs predictions without explicitly being programmed to do so. The predictions can be a numerical value, or it can be categorical value these both can be predicted using techniques called regression and classification. Before performing the predictions using a machine learning model, the data needs to be clean and tidy; this process is called data cleaning, exploratory data analysis. Finally, Using the training data, which is clean and tidy, we can predict the unseen data.

Coming to data that is the essential input for the machine learning model, the available data is enormous and varied in nature. We need algorithms such that they can handle massive datasets; a solution to this is Big Data. We can combine Big Data and machine learning to handle massive data and perform predictions.

Spark MLlib is one of the big data technologies mainly focused on machine learning on big data. Spark runs on Hadoop, Apache Mesos, Kubernetes, standalone, or the cloud, against diverse data sources enabling big data analysis. It can be implemented in Java, Python, R, and Scala. The performance is 100x times faster than Map Reduce. Almost every machine learning algorithms are available in Spark MLlib.

**Classification-Credit card fraud detection:**

Fraud is one of the major problems for credit card companies because of the large number of transactions that are processed daily and because many fraudulent transactions look a lot like regular ones. Identifying these fraudulent credit card transactions is a common type of imbalanced binary classification where the main focus is on the positive class, i.e. (is fraud) class.

**Work-Flow:**

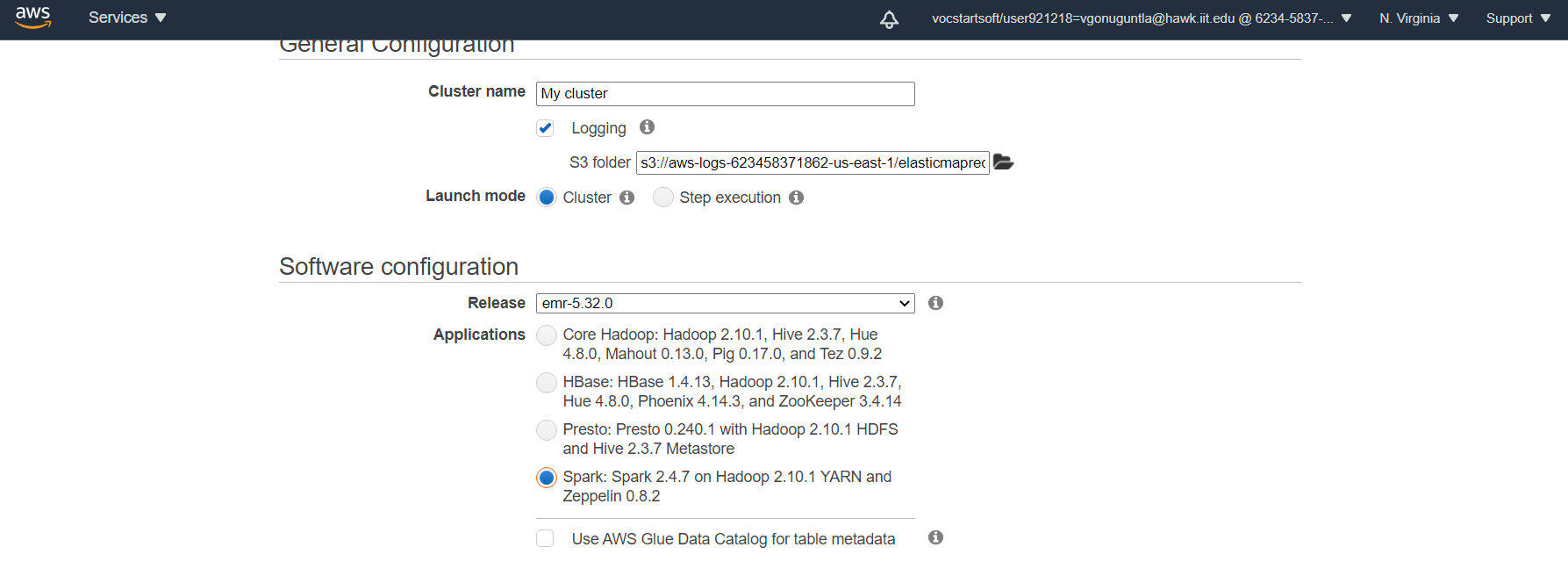
We need to perform data cleaning, data transformation, and model building on these datasets. These steps are implemented using the Spark MLlib and Scikit- learn packages in python language. These are explained in detail below.

We are detecting the binomial variable/class. Whether fraud or not. We will use a supervised learning classification technique called logistic regression using the above packages.

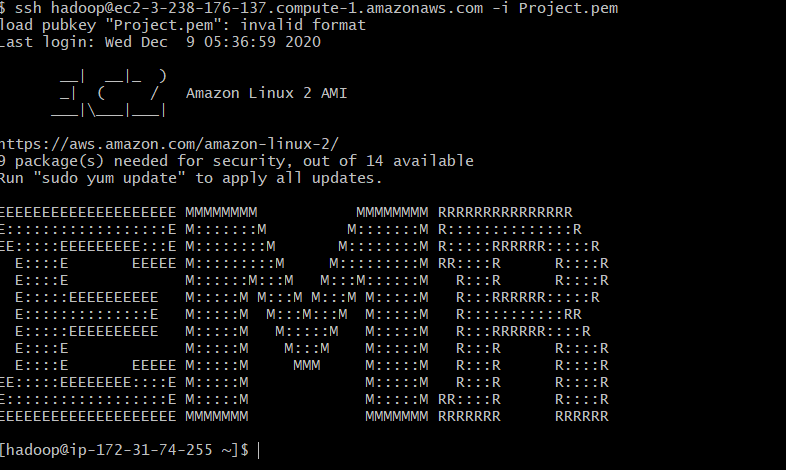
**Using Spark MLlib:**

To use Spark Mllib, we have used AWS EMR instance. Let us see how to create an EMR instance.

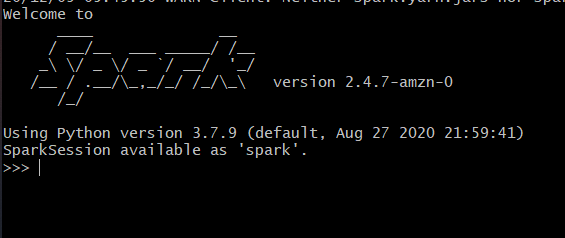
**Step 1:** Create a cluster by selecting EMR service and select Spark as software configuration as below.



**Step 2:** Once the cluster is ready, open terminal and ssh hadoop@<DNS Name> -I EMR-key-pair.pem. You should see the below screen.



**Step 3:** Enter the pyspark command, which will redirect to the pyspark shell, where you can execute the .py files you have written for machine learning models to detect credit card fraud. The pyspark shell looks something like this.



**Step 4:** Before running the code files, we need to send the local files to the Hadoop cluster. This can be done running the below command in your windows/Mac machine terminal.



In this way, you need to send creditcard.py, creditcard.csv, and Requirements.txt files available in the classification folder.

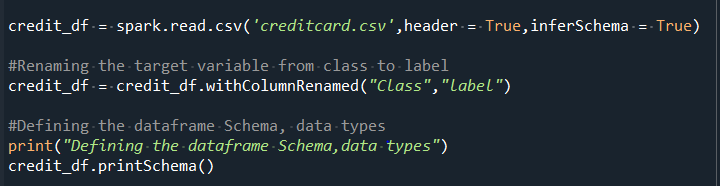
After this, send the csv file to /user/Hadoop using the below command:hadoop fs -copyFromLocal /home/hadoop/creditcard.csv /user/hadoop/creditcard.csv

**Step 5:** Once the above command is successful, we can access the files in the Hadoop cluster. You need to install the required libraries to run the creditcard.py program. This can be done using Requirements.txt file.

sudo python3 -m pip install -r Requirements.txt

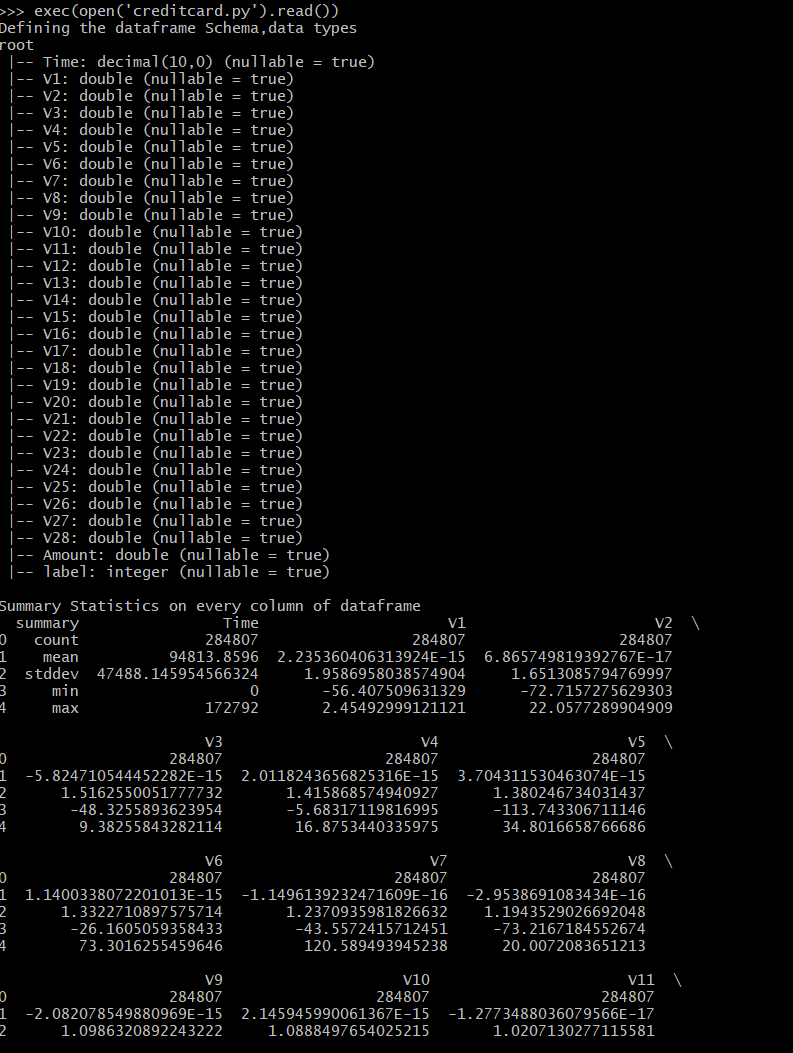
This command installs the required libraries. Now, we are all ready to run the program file to train the machine learning model.

**Data Description:**

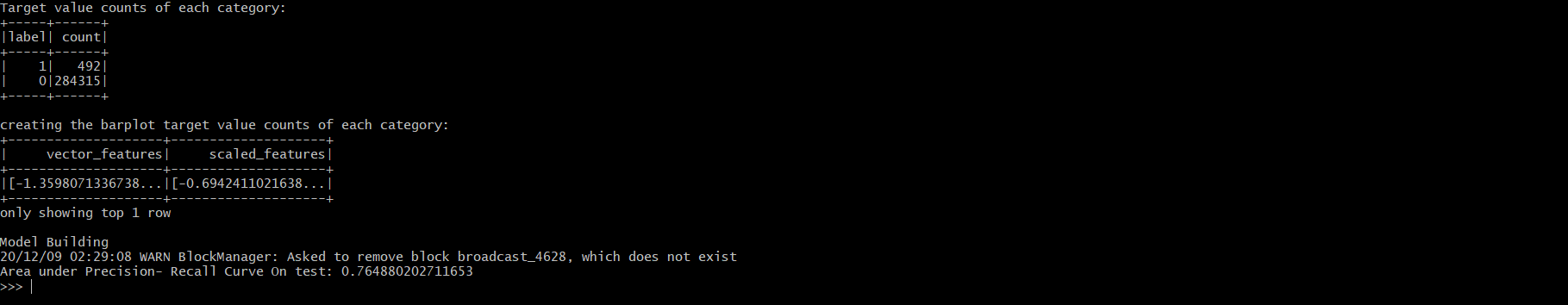
Create a spark data frame reading Creditcard.csv file, and after this step, the target variable has been renamed from class to label.

Let's the schema of Spark data frame and what are the columns available. There are 30 input features and one target variable called a label. Time variable tells about the second at which transaction appeared.

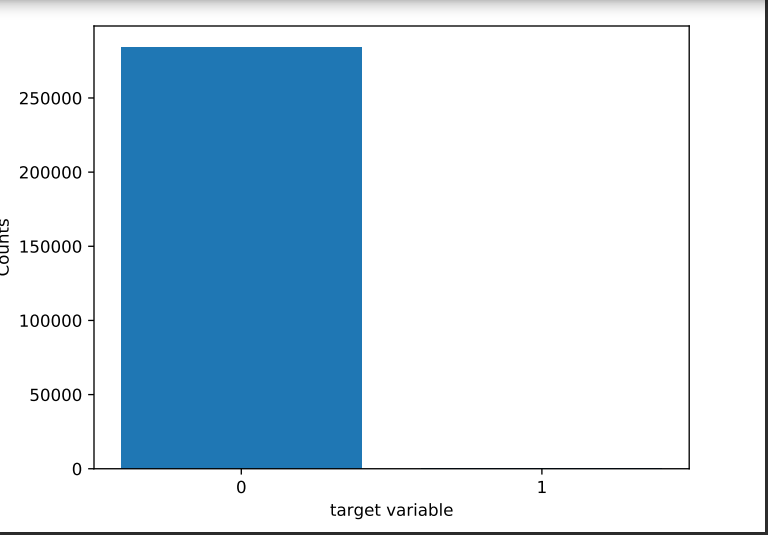
V1 to v28 are the PCA determined features, but those features are not explained to us due to privacy concerns[2]. The amount column tells the amount of transaction done at a time. The label contains 2 types of values 0(not fraud) and 1(fraud).

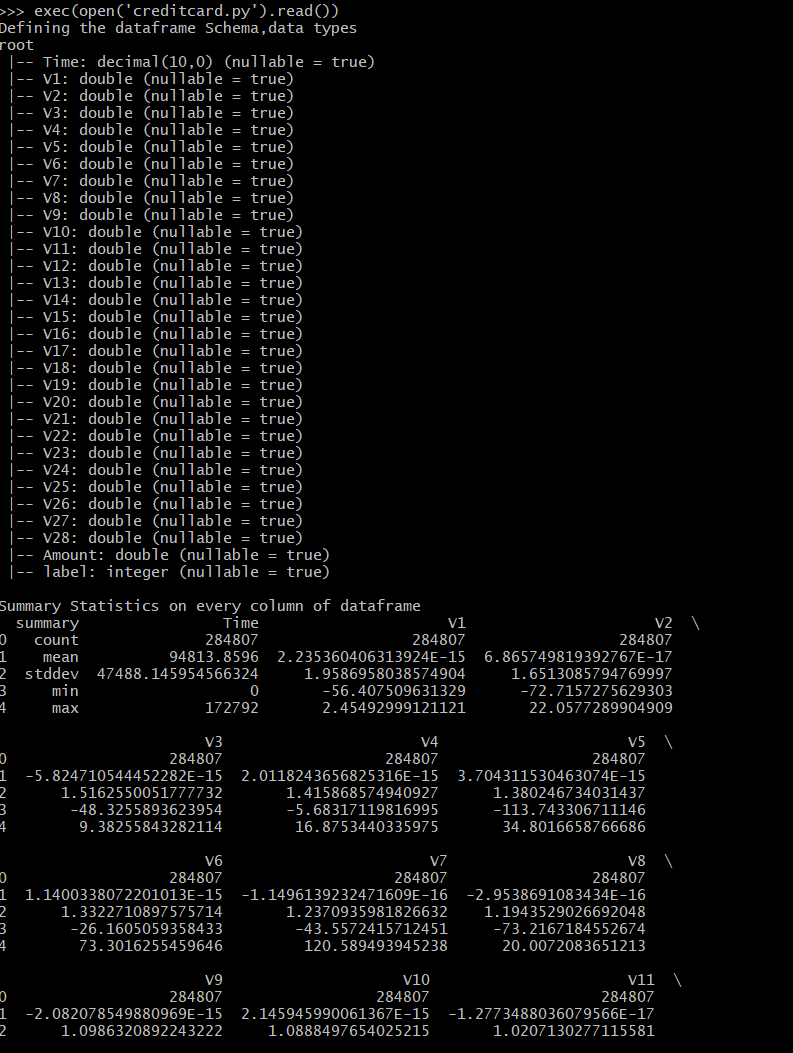


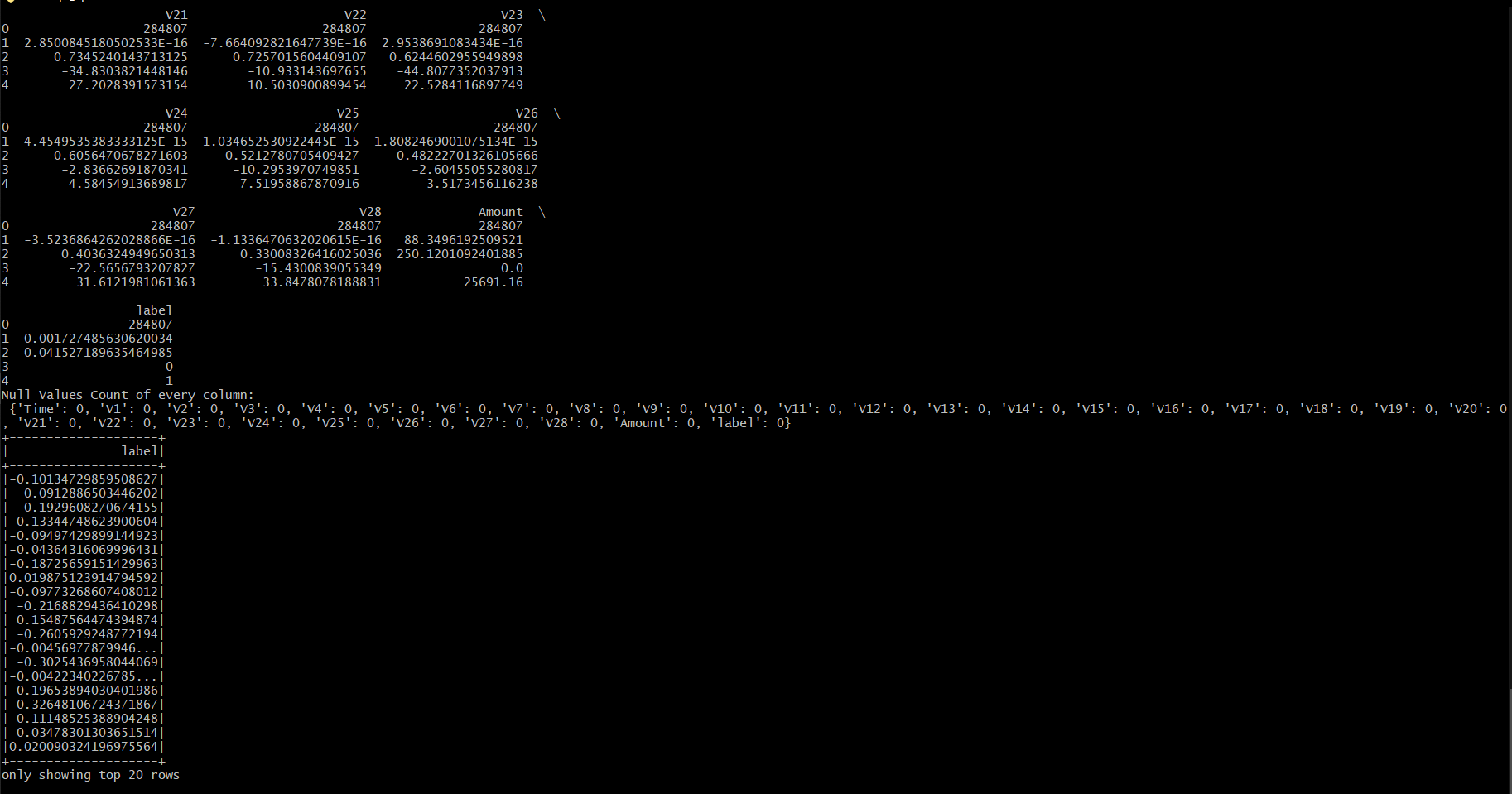
**Target Value Insights:**

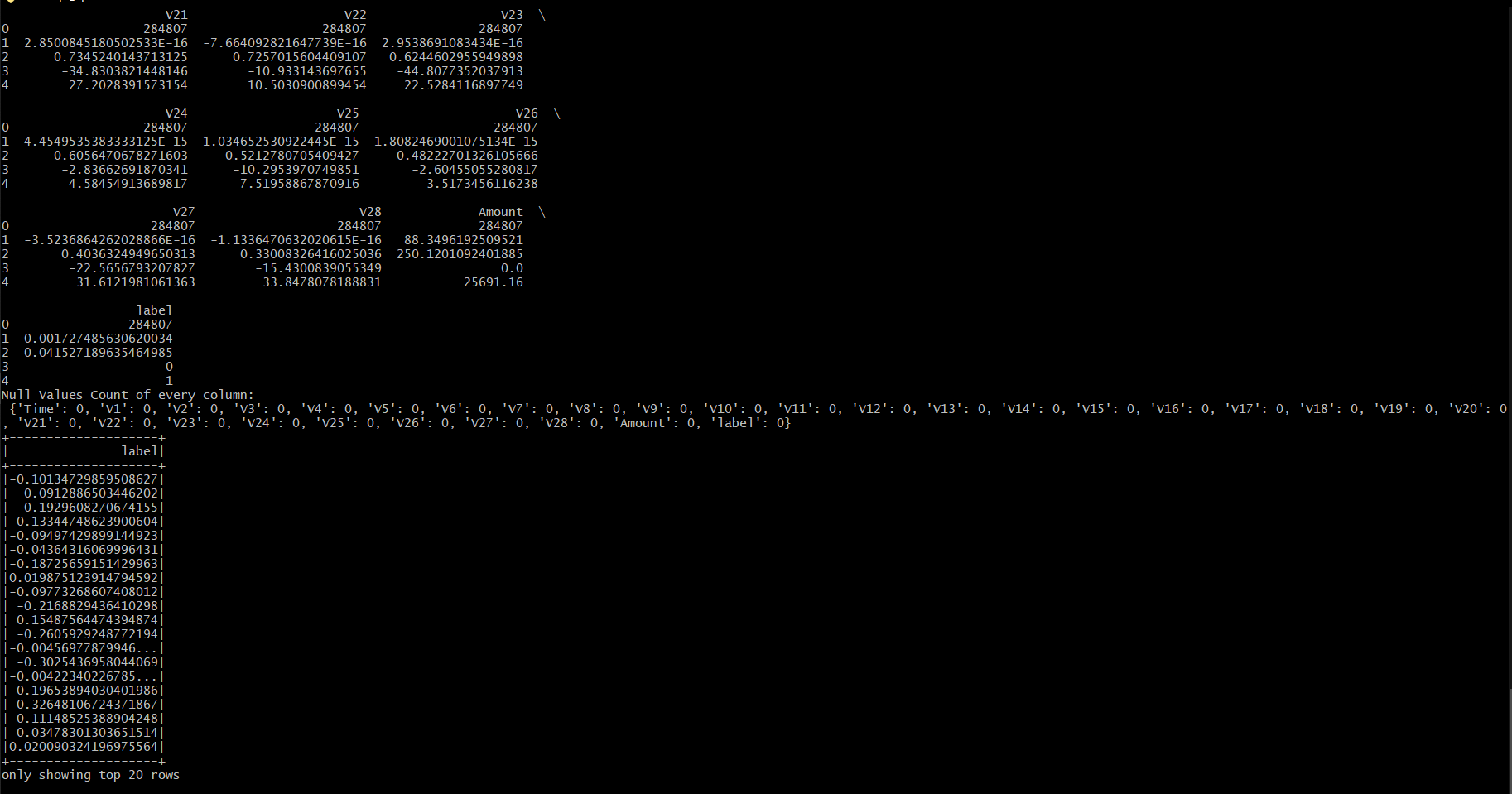
* Target value counts for each category. You can see this true class fraud is available very less in amount, which states that the dataset is highly imbalanced.

Let us also see the bar plot for this. Class 1 is almost invisible in the graph as the count is significantly less.



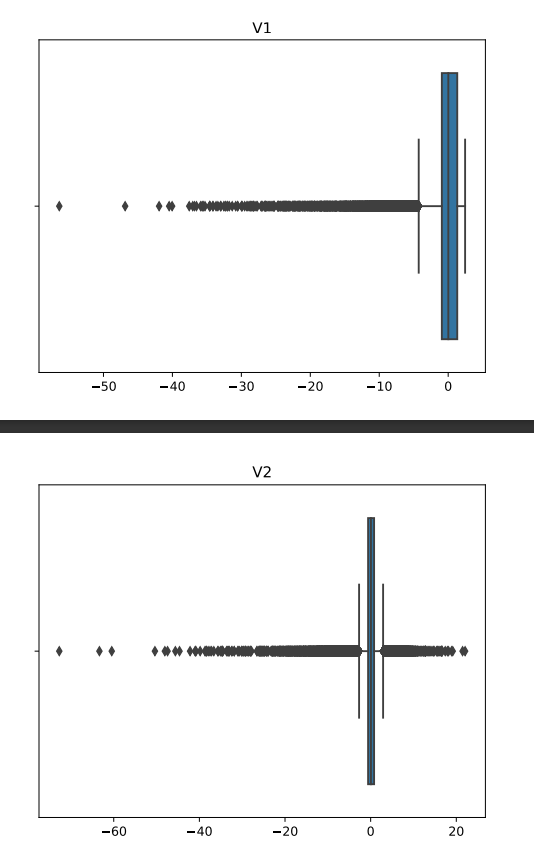
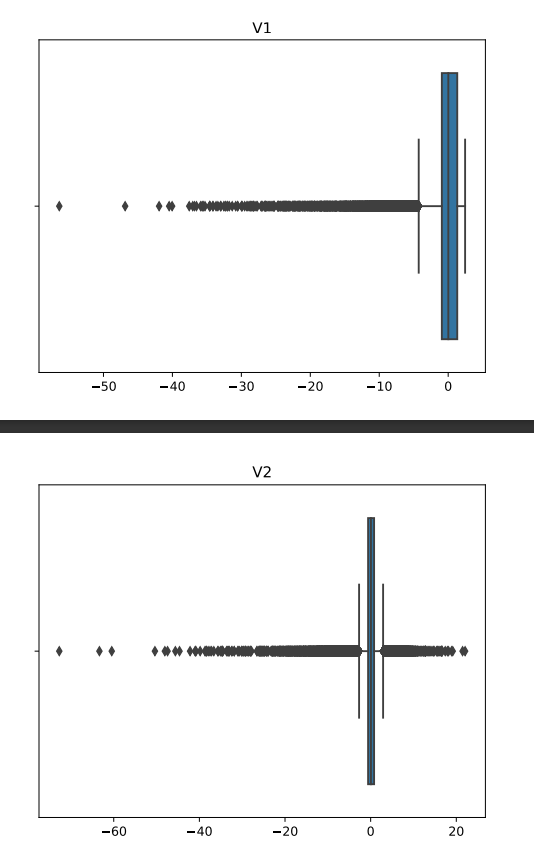
**Input features Insights:**We have calculated summary statistics for each column, which tells min, max, mean, standard deviation for the respective column. The below shows the some of the columns info.

* There are no null values in the dataset.

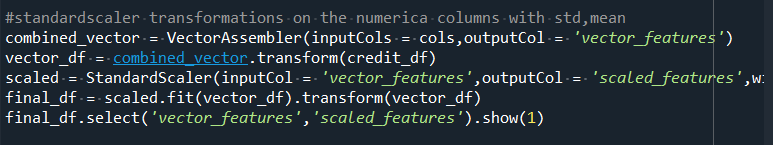
**Correlation between the numeric features to label:**

* The right side Image is the output of correlation values with the label column. It seems almost every numeric feature correlation is less. No value is greater than 0.4.
* There are no strongly correlated variables with the target.
* Hence, we are not considering ignoring the low value features. For our model training we have considered all features v1 to v28, Amount.

**Data Transformations:**

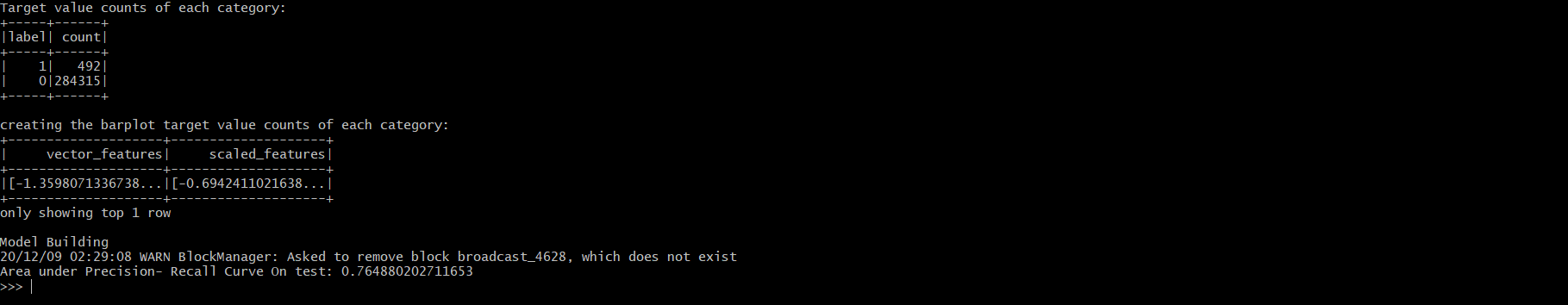
Let us see some of the box plots for the numeric features:

This was the case for almost every numeric feature v1 to v28, which means we have high outliers/range of the features. To overcome these outliers, we can do is transformations. It can be any type of transformation that makes the dataset less prone to outliers.

For our model prediction, we used standard scaler transformation with corresponding mean, standard deviation.

The above image shows how the standard scaling transformation is done. In pyspark, the StandardScaler function expects a vector when it is doing transformations on multiple columns so vectorAssembler does the required job by combining multiple columns into a single vector. Now on these vectors, standardscaler[4] fits and transforms it.

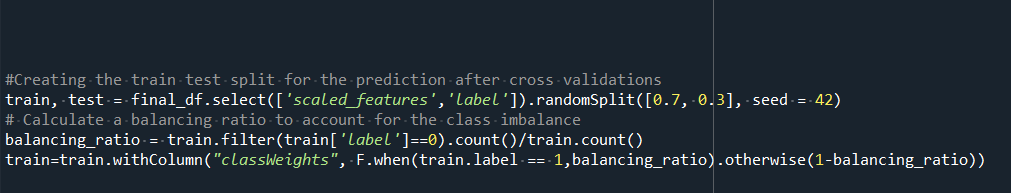
Below is the sample output after the transformations:



These scaled\_features will be used as input to train the model and predict the model once it is fitted.

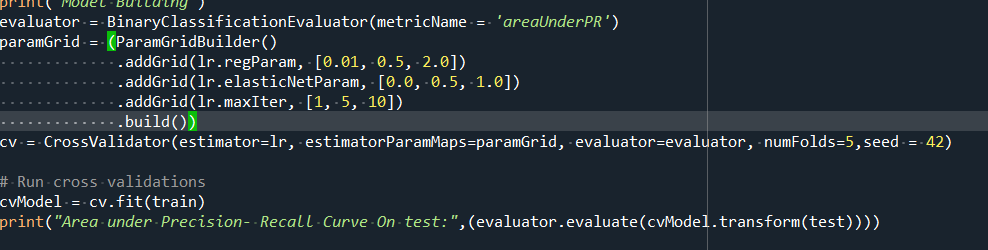
**Model Building:**

For this step, we need to split the data very carefully to train, test as the dataset is very imbalanced where true (fraud ) class values are very less compared to not a fraud.



For this, we have written the above snippet, which takes the percentage of class 0 and when creating a train split it assign values based on the probability of balancing ratio. If the class is 1 the value gets assigned to train with the percentage of class 0 and 0 is 1 – percentage. In this way, we have balanced the classes for training so that there is no bias. Taken reference from[3].

As we have now train, test datasets let us now perform the logistic regression on these. The below snippet is for 5 fold cross-validation training. The random seed value helps to replicate the results. Cross validator has taken the logistic regression as estimator and parameters as elasticnet, max iterations and to evaluate the score for each fold, and we have the area under precision-recall, which is a very good scorer for binary classification.



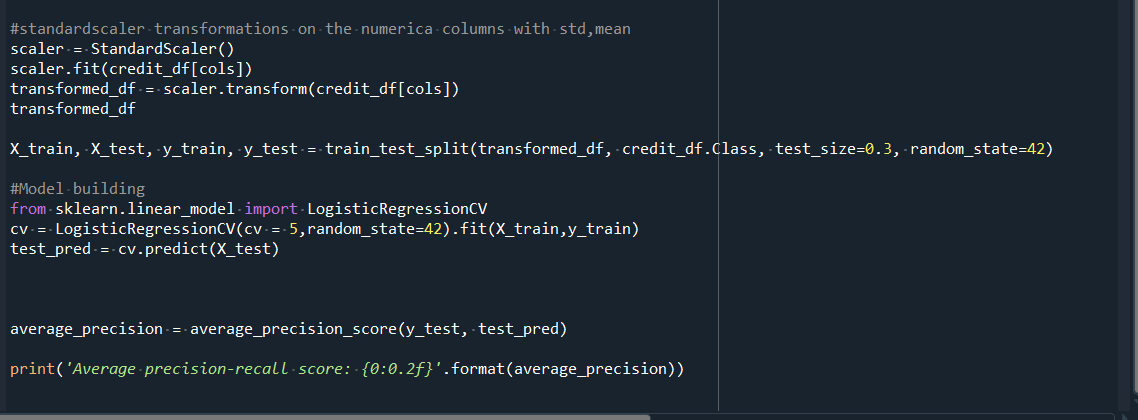
After training the model, we have achieved the area under precision-recall of 0.76 which is pretty good for the given imbalanced dataset.

**Comparing results with scikit – learn:**

The same has been implemented in scikit learn also; each and every step explained above can be replicable through scikit – learn. The implementation can be found in creditcard\_scikit.py. The below image is the small snippet doing the transformation and performing the cross-validation logistic regression.

The area under precision–recall was 0.55.

We can say that pyspark, when implemented with same steps, It was able to learn more features and improve the accuracy of model when compared with scikit learn[6].



**Regression**

Google Colab is the platform used by us, where we perform all the regression part. We install the packages which are necessary for the machine learning algorithms and correlation Matrix. We upload the data file in Google Colab using files.upload() function. Then necessary libraries are imported.

Graphical user interface, text, application, Teams

Description automatically generated

Then we create a data frame to read the data file named games.csv, and show the schema and first ten rows of the data file.

**EDA & Data Cleaning**

There are some feature (“minplayers”,” maxplayers”,” yearpublished”) in the data that needed to be converted into integer type this is done using cast function In pyspark.

A picture containing text

Description automatically generated

As we are predicting the user rating we are the column “users\_rated” should be non-zero and positive to implement this we filter data where all the user rating is greater than 0.

A picture containing Word

Description automatically generated

We drop all the un-related features from the data to effectively increase the model accuracy.

**Histogram:**

As a part of the exploratory Data Analytics we plot the histogram of the dependent variable

Chart, histogram

Description automatically generated

**Correlation Matrix and HeatMap:**

A picture containing icon

Description automatically generated

**Pairplot:**

A picture containing shoji, indoor, tiled

Description automatically generated

**Box Plot:**

Input all the features into single vectors

Standard Scaling all the feature vector to handle the outlier observed in the Boxplots

To check how the model works, we split the data into 3 different sets: first is the training set to train the model, second is the validation set to validate the accuracy of our model, and finally, the last is the test set to test the performance of our model to see how the model functions on the new data.

Chart, histogram

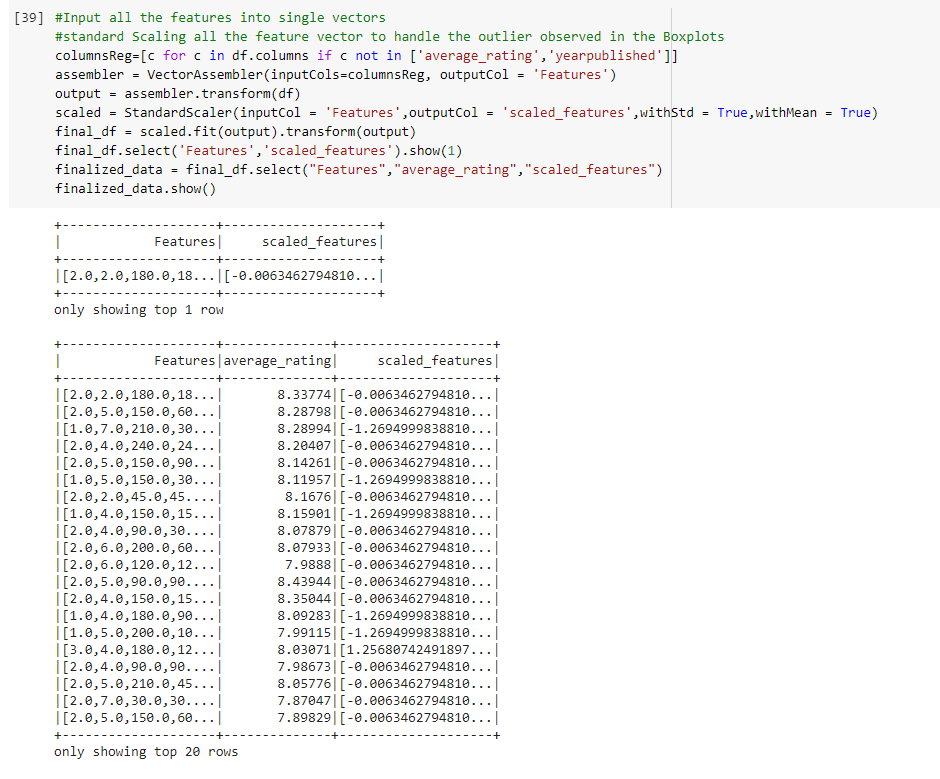
Description automatically generated

Shape, rectangle

Description automatically generated

**Test Train Split:**

Before building the model we have to split the data into train and test data so that we can validate the model. To perform this we use the below code.



**Model Building:**

From the box plot we can see that there are large number of outliers this is due to features lying on different scale to handle this we apply SatandardScalar to get all the features in the same scale. After this we apply the scaled data as the input to linear regression model.

The model Is then applied to do the prediction Below are the results for the prediction.

Table

Description automatically generated

The accuracy of the model is calculated using Mean Square Error which is as below

Graphical user interface, text, application, chat or text message

Description automatically generated

**Future Work:**

As the MSE value is 1.4 to reduce this we are planning to use Random Forest to predict the ratings.

**Comparing the results with Scikt-learn:**

When the same model is run using the sckit learn library it resulted in higher MSE. From this we can say that the that pyspark when implemented with same steps, It was able to learn more features and improve the accuracy of model when compared with scikit learn

Graphical user interface, text, application

Description automatically generated

**References:**

[1] [MLlib | Apache Spark](https://spark.apache.org/mllib/) – Mllib apache spark official page containing all resources.

[2] [**https://www.kaggle.com/mlg-ulb/creditcardfraud**](https://www.kaggle.com/mlg-ulb/creditcardfraud) **-** dataset link used for classification.

[3] [Predict Customer Churn using PySpark Machine Learning | by Marvin Lüthe | Towards Data Science](https://towardsdatascience.com/predict-customer-churn-using-pyspark-machine-learning-519e866449b5) - reference for the balancing ratio used while training.

[4] [Extracting, transforming and selecting features - Spark 3.0.1 Documentation (apache.org)](https://spark.apache.org/docs/latest/ml-features#normalizer) – transformations in pyspark.

[5] [pyspark.ml package — PySpark 3.0.1 documentation (apache.org)](https://spark.apache.org/docs/latest/api/python/pyspark.ml.html#pyspark.ml.tuning.CrossValidator) – Model building and prediction.

[6] [3.2.4.1.5. sklearn.linear\_model.LogisticRegressionCV — scikit-learn 0.23.2 documentation (scikit-learn.org)](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegressionCV.html) – logistic regression cv scikit learn