Predicting Fraud Transaction

Using Machine learning algorithm with PySpark in Python

By~ Aritra Nayak (A21004)

Index

- Project Workflow
- Problem Definition
- Project Goal
- Data set Information
- EDA
- Data Preparation
- Feature Engineering
- Model Building
- Evaluation
- Evaluation Metrics
- Model Evaluation
- Conclusion

Project Workflow

- **Project Definition and project goal** : The first phase of the workflow includes defining the problem and setting up project goal
- **Data collection and exploration :** Collection of data and checking for validity with EDA
- **Data preparation :** Preparing data for model building
- **Model building :** Building different machine learning model using different techniques
- **Model evaluation/Conclusion :** **Eva**luating performance of the model and selecting the best model

Problem Definition

- Now a days fraud transaction has become a great threat to society.
- Fraud transactions fuel the criminal misuse of identity details and pose a serious threat to national security. This issue is heightened by the growing sophistication of business transactions using payment cards, causing its policing to be increasingly difficult.
- In this project we will try to predict the Fraud transaction by identifying certain patterns using machine learning algorithm along with PySpark which may help in early detection of Fraudulent transaction and in turn can save a individual/organisation from an upcoming threat.

Project Goal

We will consider "isFraud" as our target variable.

We have a historic dataset containing a transaction is fraud or not along with different parameters given for a transaction that is already done. We will build predictive models using machine learning technique using the data set.

Find the best model that will give highest accuracy score in classifying a fraud transaction so that an individual or an organisation can take proper measure before making a transaction.

Data Source

- The dataset that we have used is named as "synthetic financial dataset.csv" for fraud detection.
- This dataset has been taken from UCI Machine Learning repository.

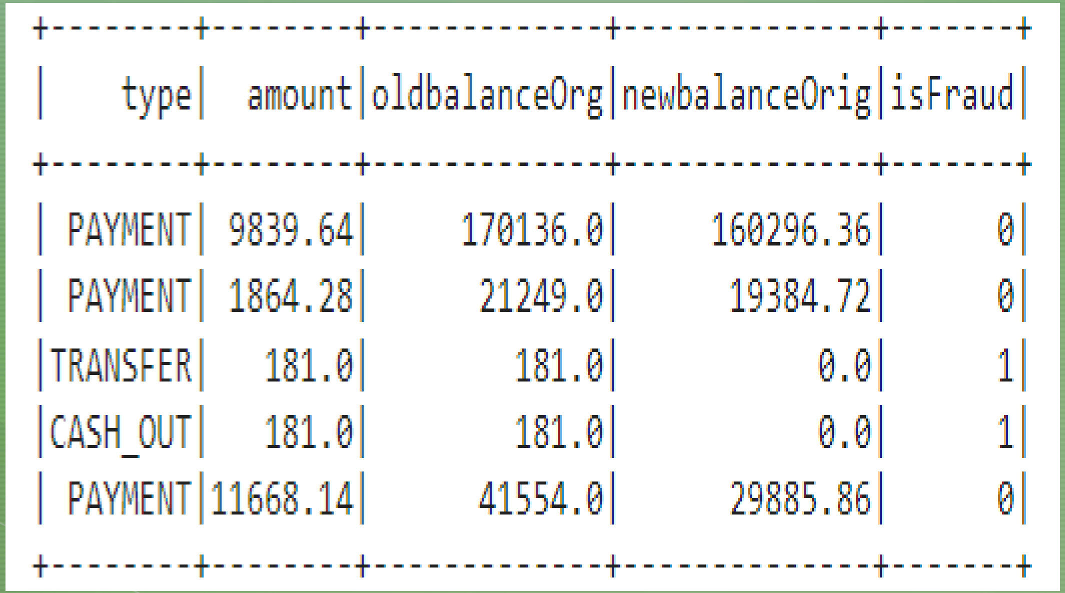
Data set Information

Independent Variables

- Type of transaction done ((categorical variable)
- Amount: Amount of money for present transaction (continuous variable)
- oldbalanceOrg :balance before making the transaction
- newbalanceOrig : new balance after making the transaction

Target variable

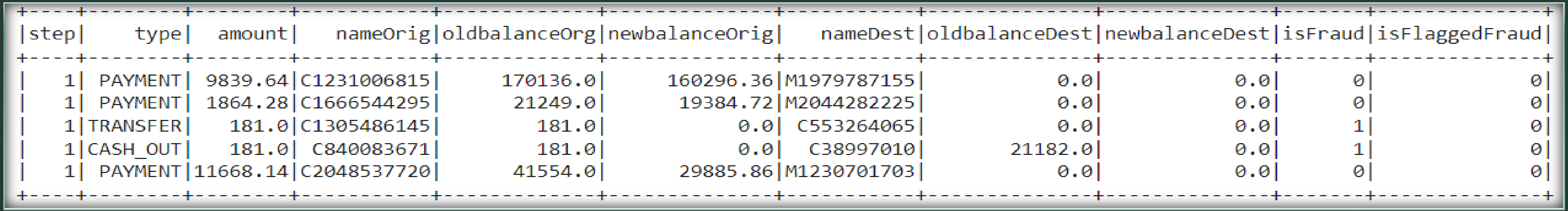
- IsFraud: '1' for yes '0' for no



```
+--------+--------+------------+--------------+-------+
|    type|  amount|oldbalanceOrg|newbalanceOrig|isFraud|
+--------+--------+------------+--------------+-------+
| PAYMENT| 9839.64|    170136.0|     160296.36|      0|
| PAYMENT| 1864.28|     21249.0|      19384.72|      0|
|TRANSFER|   181.0|       181.0|           0.0|      1|
|CASH_OUT|   181.0|       181.0|           0.0|      1|
| PAYMENT|11668.14|     41554.0|      29885.86|      0|
+--------+--------+------------+--------------+-------+
```

EDA~ Exploratory Data Analysis

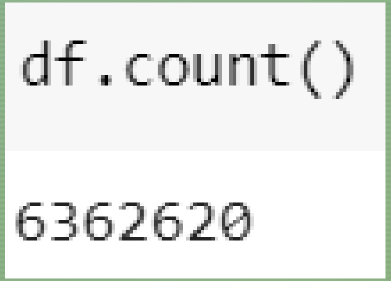
- In the original dataset there were 11 columns all total (as shown below) among which we have drop the unwanted columns and taken only four independent and one target column
- We have found that the ranges of the continuous variables are almost same therefore we haven't done any scaling in EDA.
- Column 'type' is categorial which has been taken care of accordingly in the upcoming steps



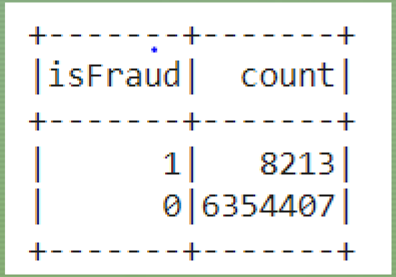
```
+----+--------+--------+-----------+-------------+--------------+-----------+--------------+--------------+-------+--------------+
|step|    type|  amount|   nameOrig|oldbalanceOrg|newbalanceOrig|   nameDest|oldbalanceDest|newbalanceDest|isFraud|isFlaggedFraud|
+----+--------+--------+-----------+-------------+--------------+-----------+--------------+--------------+-------+--------------+
|   1| PAYMENT| 9839.64|C1231006815|     170136.0|     160296.36|M1979787155|           0.0|           0.0|      0|             0|
|   1| PAYMENT| 1864.28|C1666544295|      21249.0|      19384.72|M2044282225|           0.0|           0.0|      0|             0|
|   1|TRANSFER|   181.0|C1305486145|        181.0|           0.0| C553264065|           0.0|           0.0|      1|             0|
|   1|CASH_OUT|   181.0| C840083671|        181.0|           0.0|  C38997010|       21182.0|           0.0|      1|             0|
|   1| PAYMENT|11668.14|C2048537720|      41554.0|      29885.86|M1230701703|           0.0|           0.0|      0|             0|
+----+--------+--------+-----------+-------------+--------------+-----------+--------------+--------------+-------+--------------+
```

Data Preparation

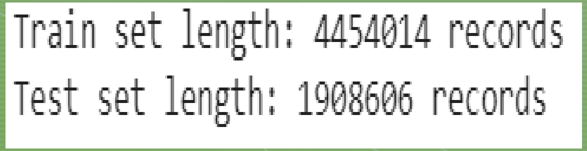
- The data set on which we have done our analysis after selecting some specific columns has 6362620 row and 5 columns
- Among which 8213 is Fraud transaction and 6354407 is non fraud transaction.
- We have splitted the data set at earlier stages to avoid data leakage and the split is done at 70:30 ration as shown in the attached picture



```
df.count()

6362620
```



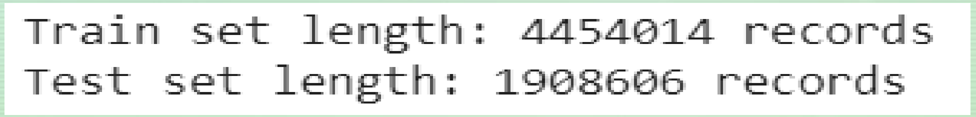
```
+-------+-------+
|isFraud|  count|
+-------+-------+
|      1|   8213|
|      0|6354407|
+-------+-------+
```



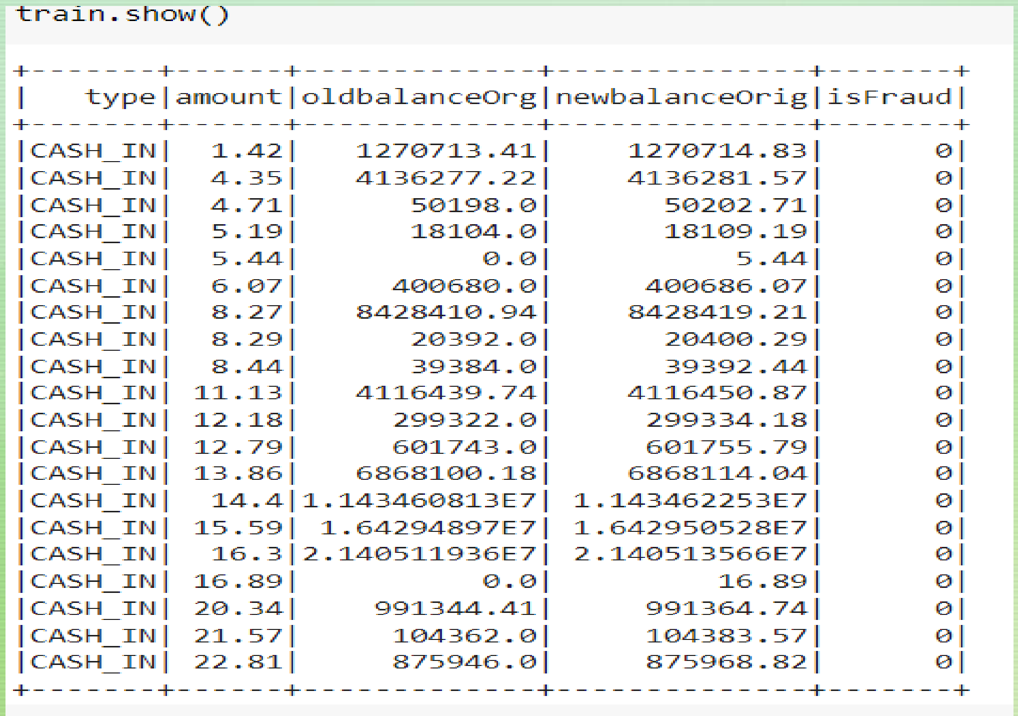
```
Train set length: 4454014 records
Test set length: 1908606 records
```

Train-Test Split

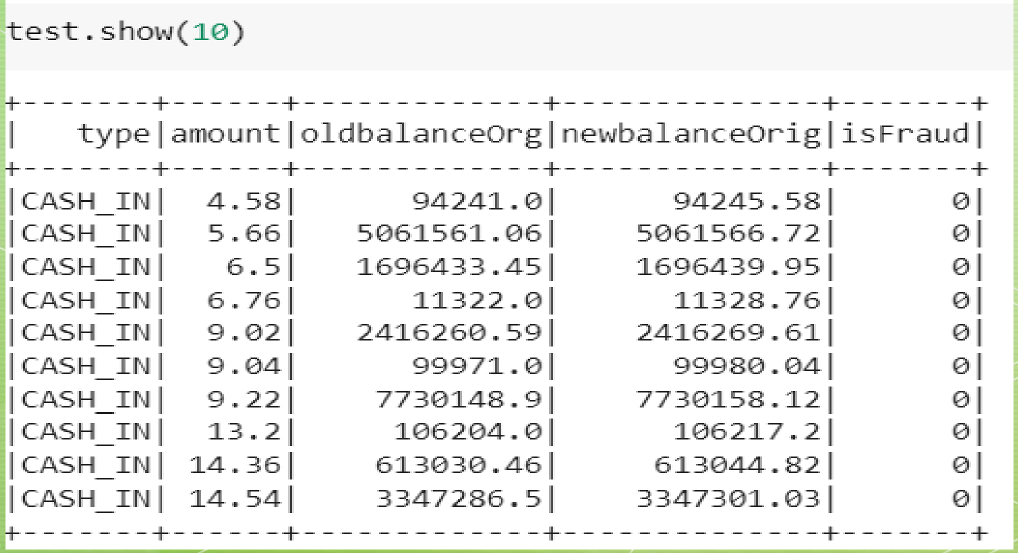
- **Train/Test split - We did the train test split at first so that there is no chance of data leakage in our analysis.**
- **the train-test split is a technique for evaluating the performance of a machine learning algorithm. The procedure involves taking a dataset and dividing it into two subsets.**
- **Train Dataset: Used to fit the machine learning model.**
- **Test Dataset: Used to evaluate the fit machine learning model.**
- We have splitted the dataset into 70:30 ratio for training and validating the model respectively. Also we have splitted by taking seed equals to 42 so that everytime we run the file we will be working with the same splitted file, and there is no time series data we have used randomsplit to split the data into train and test.



```
Train set length: 4454014 records
Test set length: 1908606 records
```

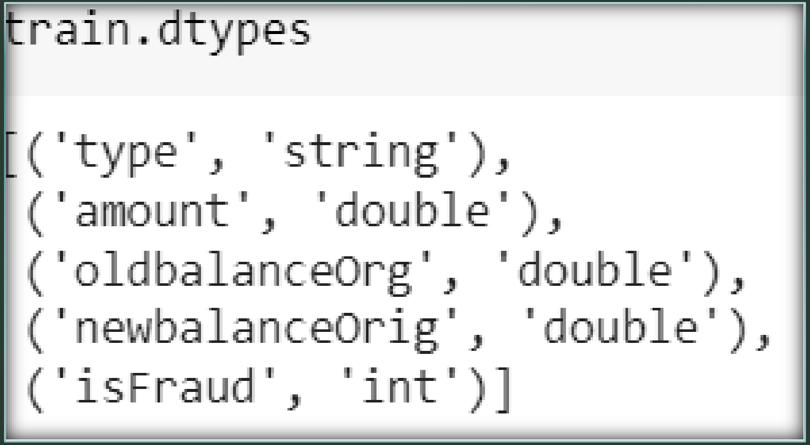


```
train.show()

+-------+------+-------------+--------------+-------+
|   type|amount|oldbalanceOrg|newbalanceOrig|isFraud|
+-------+------+-------------+--------------+-------+
|CASH_IN|  1.42|   1270713.41|    1270714.83|      0|
|CASH_IN|  4.35|   4136277.22|    4136281.57|      0|
|CASH_IN|  4.71|      50198.0|      50202.71|      0|
|CASH_IN|  5.19|      18104.0|      18109.19|      0|
|CASH_IN|  5.44|          0.0|          5.44|      0|
|CASH_IN|  6.07|     400680.0|     400686.07|      0|
|CASH_IN|  8.27|   8428410.94|    8428419.21|      0|
|CASH_IN|  8.29|      20392.0|      20400.29|      0|
|CASH_IN|  8.44|      39384.0|      39392.44|      0|
|CASH_IN| 11.13|   4116439.74|    4116450.87|      0|
|CASH_IN| 12.18|     299322.0|     299334.18|      0|
|CASH_IN| 12.79|     601743.0|     601755.79|      0|
|CASH_IN| 13.86|   6868100.18|    6868114.04|      0|
|CASH_IN|  14.4|1.143460813E7| 1.143462253E7|      0|
|CASH_IN| 15.59| 1.64294897E7| 1.642950528E7|      0|
|CASH_IN|  16.3|2.140511936E7| 2.140513566E7|      0|
|CASH_IN| 16.89|          0.0|         16.89|      0|
|CASH_IN| 20.34|    991344.41|     991364.74|      0|
|CASH_IN| 21.57|     104362.0|     104383.57|      0|
|CASH_IN| 22.81|     875946.0|     875968.82|      0|
+-------+------+-------------+--------------+-------+
```



```
test.show(10)

+-------+------+-------------+--------------+-------+
|   type|amount|oldbalanceOrg|newbalanceOrig|isFraud|
+-------+------+-------------+--------------+-------+
|CASH_IN|  4.58|      94241.0|      94245.58|      0|
|CASH_IN|  5.66|   5061561.06|    5061566.72|      0|
|CASH_IN|   6.5|   1696433.45|    1696439.95|      0|
|CASH_IN|  6.76|      11322.0|      11328.76|      0|
|CASH_IN|  9.02|   2416260.59|    2416269.61|      0|
|CASH_IN|  9.04|      99971.0|      99980.04|      0|
|CASH_IN|  9.22|    7730148.9|    7730158.12|      0|
|CASH_IN|  13.2|     106204.0|      106217.2|      0|
|CASH_IN| 14.36|    613030.46|     613044.82|      0|
|CASH_IN| 14.54|    3347286.5|    3347301.03|      0|
+-------+------+-------------+--------------+-------+
```

Datatypes

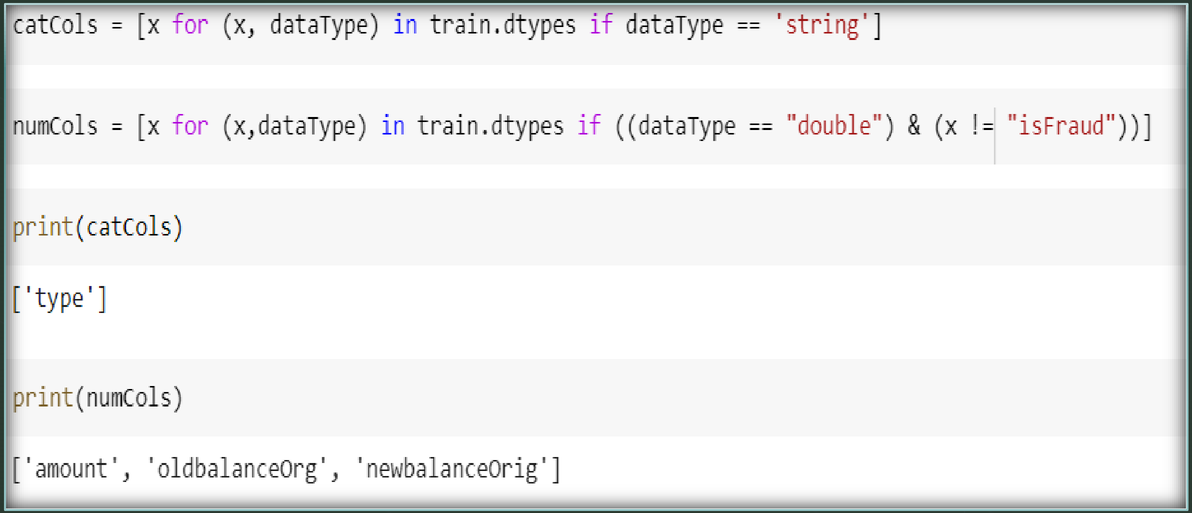


```
train.dtypes

[('type', 'string'),
 ('amount', 'double'),
 ('oldbalanceOrg', 'double'),
 ('newbalanceOrig', 'double'),
 ('isFraud', 'int')]
```

- **Dtypes** - In this dataset the column of type string has been treated as a categorical attribute. But sometimes it may happen that we treat a categorical attribute as a numerical one and vice-versa, based on our requirements. This need to be carefuly indentified and carry on our preprocessing and feature extraction task accordingly.

Separating Numerical & categorical columns

- We have separated the num and cat columns in our dataset to carry out the required processes accordingly like one hot encoding for the unique attribute of a categorical column



```
catCols = [x for (x, dataType) in train.dtypes if dataType == 'string']

numCols = [x for (x,dataType) in train.dtypes if ((dataType == "double") & (x != "isFraud"))]

print(catCols)
['type']

print(numCols)
['amount', 'oldbalanceOrg', 'newbalanceOrig']
```

Checking the distinct value of the categorical column "type"

- We have found that there are five distinct value in the "type" column. They are as follows:

❑ Transfer

❑ Cash_In

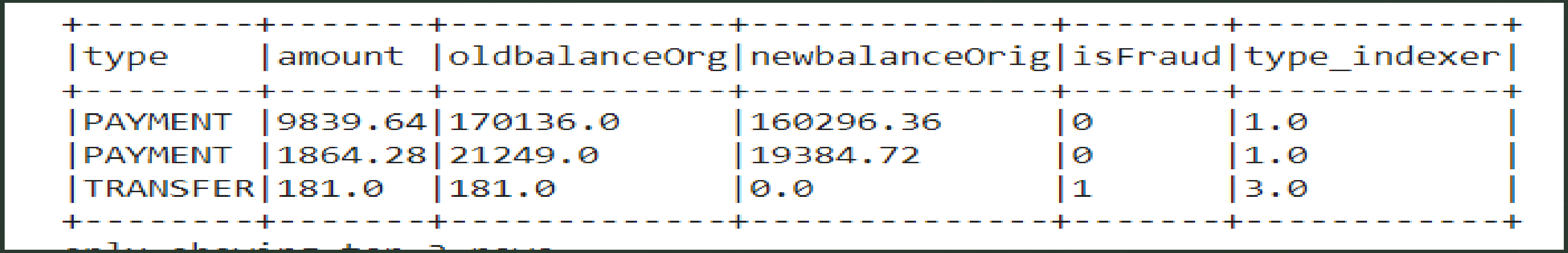
❑ Cash_Out

❑ Payment

❑ Debit

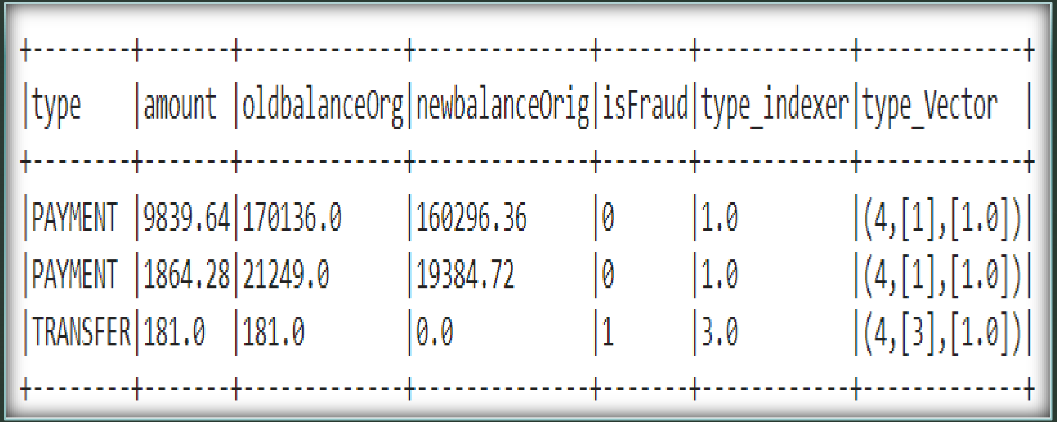
String Indexer

- StringIndexer encodes a string column of labels to a column of label indices. If the input column is numeric, we cast it to string and index the string values. The indices are in [0, num Labels).
- **StringIndexer :** Use to convert a single feature into an index feature as it is mainly used to perform the task of ordering.
- StringIndexer is imported from the module pyspark.ml,feature.
- In the below picture "type_indexer" is the result of string Indexer



```
+--------+-------+-------------+--------------+-------+------------+
|type    |amount |oldbalanceOrg|newbalanceOrig|isFraud|type_indexer|
+--------+-------+-------------+--------------+-------+------------+
|PAYMENT |9839.64|170136.0     |160296.36     |0      |1.0         |
|PAYMENT |1864.28|21249.0      |19384.72      |0      |1.0         |
|TRANSFER|181.0  |181.0        |0.0           |1      |3.0         |
+--------+-------+-------------+--------------+-------+------------+
```

Feature Engineering~ One Hot Encoding



```
+--------+-------+-------------+--------------+-------+-----------+-------------+
|type    |amount |oldbalanceOrg|newbalanceOrig|isFraud|type_indexer|type_Vector  |
+--------+-------+-------------+--------------+-------+-----------+-------------+
|PAYMENT |9839.64|170136.0     |160296.36     |0      |1.0        |(4,[1],[1.0])|
|PAYMENT |1864.28|21249.0      |19384.72      |0      |1.0        |(4,[1],[1.0])|
|TRANSFER|181.0  |181.0        |0.0           |1      |3.0        |(4,[3],[1.0])|
+--------+-------+-------------+--------------+-------+-----------+-------------+
```

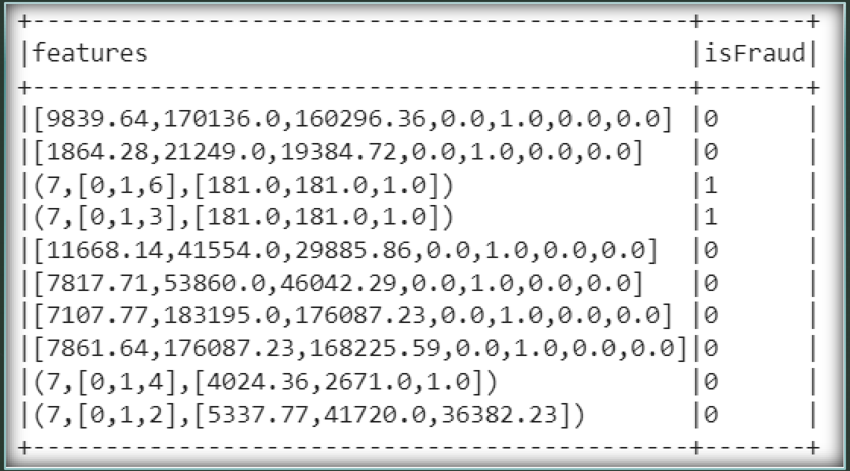
- ❑ One Hot Encoding is used for converting categorical attributes into a numeric vector that machine learning models can understand.
- ❑ OneHotEncoder is imported from the module pyspark.ml,feature.
- ❑ In the below picture "type_vector" is the result of one hot encoding.

Feature Engineering ~ VECTOR ASSEMBLER

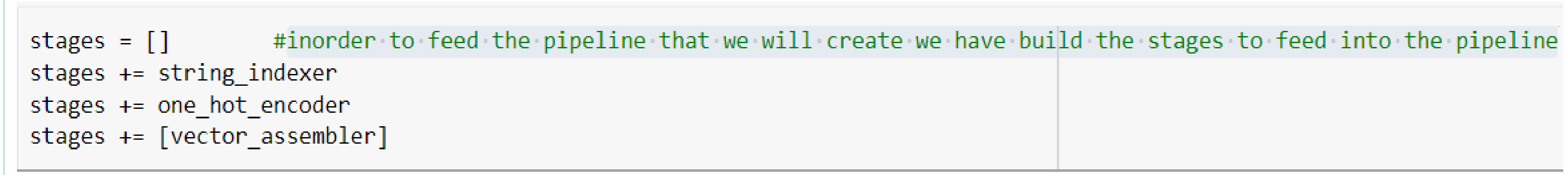
- Using Vector assembler to merge multiple column into a vector column and taken output as a single feature column. Inputs are given for all the necessary columns.

- Showing the transformed vector as 'features' column with the "isFraud" target column.

- VectoreAssembler is imported from the module pyspark.ml,feature.



```
+---------------------------------------------+-------+
|features                                     |isFraud|
+---------------------------------------------+-------+
|[9839.64,170136.0,160296.36,0.0,1.0,0.0,0.0] |0      |
|[1864.28,21249.0,19384.72,0.0,1.0,0.0,0.0]   |0      |
|(7,[0,1,6],[181.0,181.0,1.0])                |1      |
|(7,[0,1,3],[181.0,181.0,1.0])                |1      |
|[11668.14,41554.0,29885.86,0.0,1.0,0.0,0.0]  |0      |
|[7817.71,53860.0,46042.29,0.0,1.0,0.0,0.0]   |0      |
|[7107.77,183195.0,176087.23,0.0,1.0,0.0,0.0] |0      |
|[7861.64,176087.23,168225.59,0.0,1.0,0.0,0.0]|0      |
|(7,[0,1,4],[4024.36,2671.0,1.0])             |0      |
|(7,[0,1,2],[5337.77,41720.0,36382.23])       |0      |
+---------------------------------------------+-------+
```



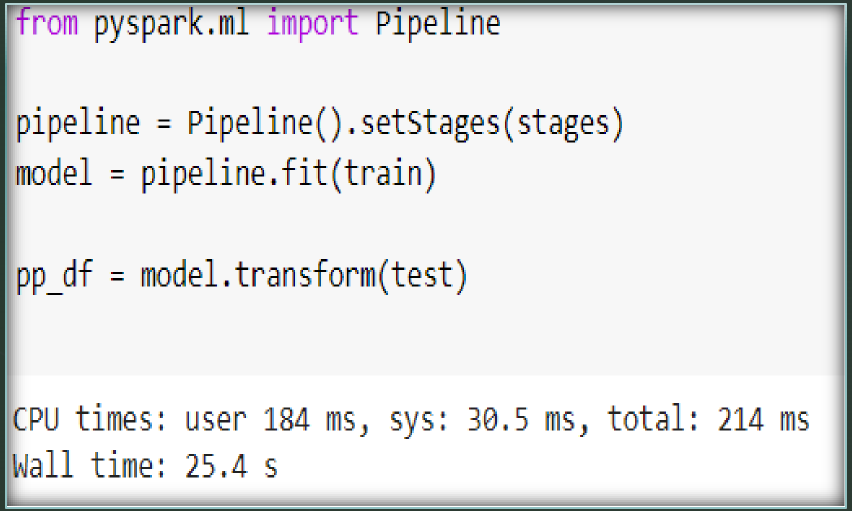
```
stages = []        #inorder to feed the pipeline that we will create we have build the stages to feed into the pipeline
stages += string_indexer
stages += one_hot_encoder
stages += [vector_assembler]
```

Building stages for pipeline which include StringIndexer, OneHotEncoder & VectorAssembler.

Pre-steps for pipeline

Building Machine Learning Model

- We have created **pipelines** for our models : A machine learning **pipeline** is used to help automate machine learning workflows. They operate by enabling a sequence of data to be transformed and correlated together in a **model** that can be tested and evaluated to achieve an outcome, whether positive or negative.
- We have fitted the pipeline with train data and then transformed the test data accordingly.



```
from pyspark.ml import Pipeline

pipeline = Pipeline().setStages(stages)
model = pipeline.fit(train)

pp_df = model.transform(test)
```

```
CPU times: user 184 ms, sys: 30.5 ms, total: 214 ms
Wall time: 25.4 s
```

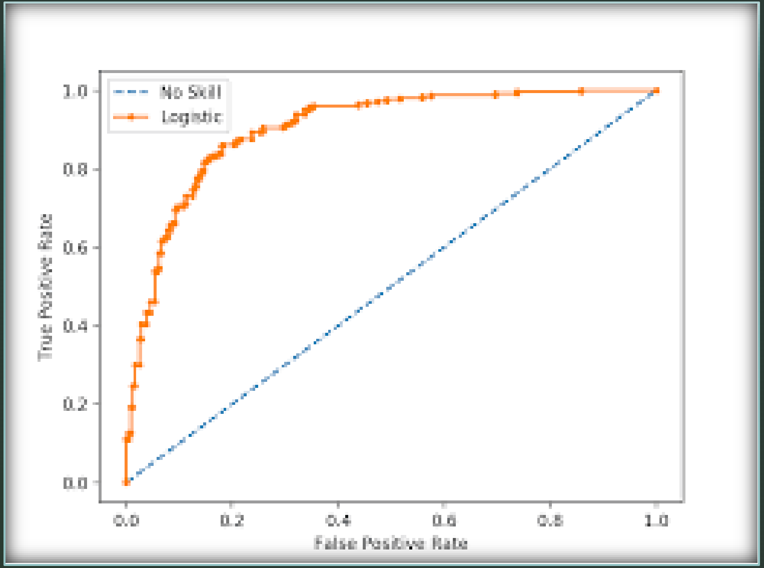
Evaluation ~ Machine Learning Model for classification

- We We have used below machine learning technique :

  - ❑ Logistic regression (Linear Model) - imported from pyspark.ml.classification.
  - ❑ Random Forest (Ensemble Model) - imported from pyspark.ml.classification.

Evaluation Metrics

- **Accuracy** : **Model accuracy** is defined as the number of classifications a model correctly predicts divided by the total number of predictions made.
- **AUC** : AUC stands for "Area under the ROC Curve." it measures the entire two-dimensional area underneath the entire ROC curve. AUC provides an aggregate measure of performance across all possible classification thresholds. In the picture area under the orange curve is AUC score.
- As our data set is imbalance and this is a classification problem so **we will more focus on AUC score.**



Model Evaluation

- **Logistic Regression**
  - Accuracy : 99.91%
  - AUC Score : 99.18%

- **Random Forest Classifier**
  - Accuracy : 99.90 %
  - AUC Score : 97.15%

If we look at the Accuracy score then there is no significant changes happening when we have used ensemble models but if we look at the AUC Score , a slight change has been observed in case of ensemble techniques and highest for Logistic Regression .

Conclusion

- The accuracy of Logistic Regression Model is 99.91%, so we can say that for 100 cases our model can correctly predict 99.91 instances also the AUC score is highest for this, so this model will be our final model.
- Our project goal was to build a predictive model to predict chances of Fraud transaction depending upon each transaction factor and it looks like we have successfully created one with a satisfactory performance score.
- After deploying this model if a person provide his/her transaction details the model will suggest whether he/she is prone to fraud transaction or not ,this way one can take necessary measures to avoid making fraudulent transaction which will eventually reduce the risk of losing money.
- Overall this project will help to increase 'online transaction awareness' among individuals or organizations and help them to adopt safety and sound practices.