**Project No: 5**

**Project Title: Fraud Transaction Detection in Supply Chain**

**Prepared by: Rose Mary Jose**

**Date: 10-06-23**

Prepared for “Thoughtware Training Private Limited”, under Guidance of its CEO Mr. Pattabhi Raman. The project will be subject to further research, modification and exclusive use of “Thoughtware Training Private Limited”

**Objective of the Project**

Objective of the project is to detect the fraud transaction in supply chain. If the transaction is fraud, then it the predicted output will be 1 else 0.

Fraud transaction detection in the supply chain has become essential to mitigate financial losses, protect brand reputation, and maintain operational integrity. With globalized and complex supply chains, identifying fraudulent activities such as counterfeit products, unauthorized diversions, or deceptive invoicing is crucial. Through advanced analytics and machine learning, businesses can analyze transaction patterns, detect anomalies, and prevent fraudulent incidents. This leads to reduced financial risks, improved regulatory compliance, and enhanced trust among stakeholders. Benefits include cost savings, minimized legal liabilities, and sustained customer loyalty. Industries like pharmaceuticals, electronics, and luxury goods heavily rely on fraud detection to ensure genuine product delivery, maintain ethical practices, and safeguard the market from counterfeit infiltration.

Based on the dataset and its attributes, classification models can be utilized for fraud transaction detection, such as decision trees, random forests, logistic regression, etc.

**Dataset**

Source:<https://www.kaggle.com/datasets/shashwatwork/dataco-smart-supply-chain-for-big-data-analysis?select=tokenized_access_logs.csv>

The dataset considered consist of 53 columns and 180519 rows. Each column and its description are given below.

* Type: Type of transaction made
* Days for shipping (real): Actual shipping days of the purchased product
* Days for shipment (scheduled): Days of scheduled delivery of the purchased product
* Benefit per order: Earnings per order placed
* Sales per customer: Total sales per customer made per customer
* Delivery Status: Delivery status of orders: Advance shipping, Late delivery, Shipping canceled, Shipping on time
* Late\_delivery\_risk: Categorical variable that indicates if sending is late (1), it is not late (0).
* Category ID: Product category code
* Category Name: Description of the product category
* Customer City : City where the customer made the purchase
* Customer Country: Country where the customer made the purchase
* Customer Email: Customer's email
* Customer Fname: Customer name
* Customer Id: Customer ID
* Customer Lname: Customer lastname
* Customer Password: Masked customer key
* Customer Segment: Types of Customers: Consumer, Corporate, Home Office
* Customer State: State to which the store where the purchase is registered belongs
* Customer Street: Street to which the store where the purchase is registered belongs
* Customer Zipcode: Customer Zipcode
* Department Id : Department code of store
* Department Name: Department name of store
* Latitude: Latitude corresponding to location of store
* Longitude: Longitude corresponding to location of store
* Market : Market to where the order is delivered: Africa, Europe, LATAM, Pacific Asia, USCA
* Order City: Destination city of the order
* Order Country: Destination country of the order
* Order Customer Id: Customer order code
* order date (DateOrders): Date on which the order is made
* Order Id: Order code
* Order Item Cardprod Id: Product code generated through the RFID reader
* Order Item Discount: Order item discount value
* Order Item Discount Rate: Order item discount percentage
* Order Item Id: Order item code
* Order Item Product Price: Price of products without discount
* Order Item Profit Ratio: Order Item Profit Ratio
* Order Item Quantity: Number of products per order
* Sales: Value in sales
* Order Item Total: Total amount per order
* Order Profit Per Order: Order Profit Per Order
* Order Region: Region of the world where the order is delivered: Southeast Asia, South Asia, Oceania, Eastern Asia, West Asia, West of USA, US Center, West Africa, Central Africa, North Africa, Western Europe, Northern, Caribbean, South America, East Africa, Southern Europe, East of USA, Canada, Southern Africa, Central Asia, Europe, Central America, Eastern Europe, South of USA.
* Order State: State of the region where the order is delivered
* Order Status: Order Status: COMPLETE, PENDING, CLOSED, PENDING\_PAYMENT, CANCELED, PROCESSING, SUSPECTED\_FRAUD, ON\_HOLD, PAYMENT\_REVIEW
* Product Card Id: Product code
* Product Category Id: Product category code
* Product Description: Product Description
* Product Image: Link of visit and purchase of the product
* Product Name: Product Name
* Product Price: Product Price
* Product Status: Status of the product stock: If it is 1 not available, 0 the product is available
* Shipping date (DateOrders): Exact date and time of shipment
* Shipping Mode: The following shipping modes are presented: Standard Class, First Class, Second Class, Same Day

From the following many columns are unnecessary and some new columns can be created from existing. Thus, new columns such as order\_yr, order\_month, order\_day, order\_hour are extracted from the order date (DateOrders) column. Total\_price is created using order item quantity and order item total. Unnecessary columns are removed and then the column number reduced to 29.

## reading dataset  
data = pd.read\_csv('fraud\_detection\_dataset.csv',encoding='latin-1')  
## creating new column from existing columns  
data['Total\_price']=data['Order Item Quantity']\*data['Order Item Total']

data['order\_yr']= pd.DatetimeIndex(data['order date (DateOrders)']) .year  
data['order\_month'] = pd.DatetimeIndex(data['order date (DateOrders)']).month  
data['order\_day'] = pd.DatetimeIndex(data['order date (DateOrders)']) .weekday  
data['order\_hour'] = pd.DatetimeIndex(data['order date (DateOrders)']) .hour

## droppping the un necessary columns from the dataset  
df = data.drop(['Sales per customer','Category Id','Customer Email','Customer Fname','Customer Id','Customer Lname','Customer Street','Customer Zipcode','Customer Password','Department Id','Latitude','Longitude','Order Customer Id','order date (DateOrders)','Order Id','Order Item Cardprod Id','Order Item Discount','Order Item Id','Order Item Product Price','Order Item Profit Ratio','Sales','Order Item Total','Order Profit Per Order','Product Status','Product Description','Product Card Id','Product Category Id','Product Image','Order Zipcode','shipping date (DateOrders)'],axis='columns')  
df.head()

Type Days for shipping (real) Days for shipment (scheduled) \  
0 DEBIT 3 4   
1 TRANSFER 5 4   
2 CASH 4 4   
3 DEBIT 3 4   
4 PAYMENT 2 4   
  
Benefit per order Delivery Status Late\_delivery\_risk Category Name   
0 91.250000 Advance shipping 0 Sporting Goods   
1 -249.089996 Late delivery 1 Sporting Goods   
2 -247.779999 Shipping on time 0 Sporting Goods   
3 22.860001 Advance shipping 0 Sporting Goods   
4 134.210007 Advance shipping 0 Sporting Goods   
  
Customer City Customer Country Customer Segment ... Order State   
0 Caguas Puerto Rico Consumer ... Java Occidental   
1 Caguas Puerto Rico Consumer ... Rajastán   
2 San Jose EE. UU. Consumer ... Rajastán   
3 Los Angeles EE. UU. Home Office ... Queensland   
4 Caguas Puerto Rico Corporate ... Queensland

Order Status Product Name Product Price Shipping Mode Total\_price   
0 COMPLETE Smart watch 27.75 Standard Class 314.640015   
1 PENDING Smart watch 327.75 Standard Class 311.359985   
2 CLOSED Smart watch 327.75 Standard Class 309.720001   
3 COMPLETE Smart watch 327.75 Standard Class 304.809998   
4 PENDING\_PAYMENT Smart watch 327.75 Standard Class 298.250000   
  
 order\_yr order\_month order\_day order\_hour   
0 2018.0 1.0 2.0 22.0   
1 2018.0 1.0 5.0 12.0   
2 2018.0 1.0 5.0 12.0   
3 2018.0 1.0 5.0 11.0   
4 2018.0 1.0 5.0 11.0

Here the fraud detection is done using the column ‘Order Status’ as our target variable and the rest columns as our feature variable. Suspected fraud attribute of that column is encoded into 1 and the rest into 0. Thus, fraud detection is done using classification models. Which will predict whether the future transactions are fraud or not.

**EDA**

The dataset considered consist of **53 columns and 180519 rows**, in which unnecessary columns are dropped and left with **29 columns**.

Info of the dataset returns the information about the data frame. It includes the number of columns, column labels, datatypes, non-null count of the columns, memory usage, and range index. The dataset consists of 180519 rows and 29 columns, in which 16 are of dtype objects, 8 of int64, and 4 of float64.

RangeIndex: 31665 entries, 0 to 31664  
Data columns (total 28 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Type 31665 non-null object   
 1 Days for shipping (real) 31665 non-null int64   
 2 Days for shipment (scheduled) 31665 non-null int64   
 3 Benefit per order 31665 non-null float64  
 4 Delivery Status 31665 non-null object   
 5 Late\_delivery\_risk 31665 non-null int64   
 6 Category Name 31665 non-null object   
 7 Customer City 31665 non-null object   
 8 Customer Country 31665 non-null object   
 9 Customer Segment 31665 non-null object   
 10 Customer State 31665 non-null object   
 11 Department Name 31664 non-null object   
 12 Market 31664 non-null object   
 13 Order City 31664 non-null object

14 Order Country 31664 non-null object   
 15 Order Item Discount Rate 31664 non-null float64  
 16 Order Item Quantity 31664 non-null float64  
 17 Order Region 31664 non-null object   
 18 Order State 31664 non-null object   
 19 Order Status 31664 non-null object   
 20 Product Name 31664 non-null object   
 21 Product Price 31664 non-null float64  
 22 Shipping Mode 31664 non-null object   
 23 Total\_price 31664 non-null float64  
 24 order\_yr 31664 non-null float64  
 25 order\_month 31664 non-null float64  
 26 order\_day 31664 non-null float64  
 27 order\_hour 31664 non-null float64  
dtypes: float64(9), int64(3), object(16)  
memory usage: 6.8+ MB

Describe function is used to get the summary statistics of the dataset.

Describe function of the numerical column summarizes the central tendency, dispersion and shape of the dataset’s distribution. It returns the count, mean, standard deviation, minimum value, maximum value, 25th, 50th, and 75th percentile values. From this it is observable that columns such as ‘Benefit per order’, ‘Product price’, and ‘Total price’ have a wide range and need to be standardized.

count mean std min   
Days for shipping (real) 31665.0 3.440139 1.593126 0.00   
Days for shipment (scheduled) 31665.0 2.887573 1.388701 0.00   
Benefit per order 31665.0 22.123943 106.783328 -3366.00   
Late\_delivery\_risk 31665.0 0.576788 0.494076 0.00   
Order Item Discount Rate 31664.0 0.102172 0.070703 0.00   
Order Item Quantity 31664.0 2.454680 1.560829 1.00   
Product Price 31664.0 120.050241 134.446438 11.29   
Total\_price 31664.0 486.610242 493.757299 8.47   
order\_yr 31664.0 2015.938542 0.841709 2015.00   
order\_month 31664.0 6.302110 3.459768 1.00   
order\_day 31664.0 3.018949 1.990650 0.00   
order\_hour 31664.0 11.384853 6.938111 0.00   
 25% 50% 75% Days for shipping (real 2.000000 3.000000 5.000000   
Days for shipment (scheduled) 2.000000 4.000000 4.000000   
Benefit per order 6.910000 30.700001 64.169998   
Late\_delivery\_risk 0.000000 1.000000 1.000000   
Order Item Discount Rate 0.040000 0.100000 0.160000   
Order Item Quantity 1.000000 2.000000 4.000000   
Product Price 50.000000 59.990002 129.990005   
Total\_price 124.790001 293.985001 728.000000   
order\_yr 2015.000000 2016.000000 2017.000000   
order\_month 3.000000 6.000000 9.000000   
order\_day 1.000000 3.000000 5.000000   
order\_hour 5.000000 11.000000 17.000000

Describe function of the categorical columns returns count, unique elements, and the frequency of top category. From this top category of each column and their respective frequencies are observable.

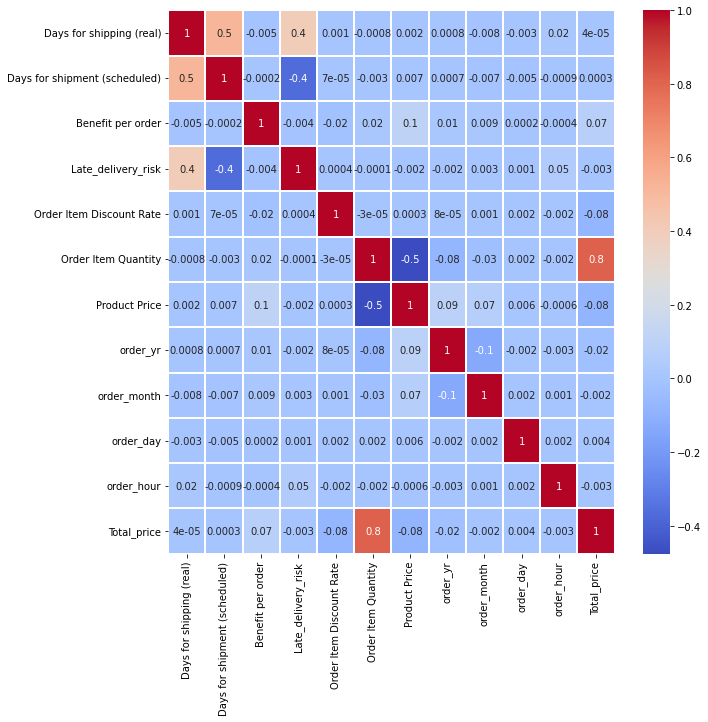
count unique top freq  
Type 31665 4 DEBIT 13281  
Delivery Status 31665 4 Late delivery 18264  
Category Name 31665 41 Cleats 6154  
Customer City 31665 562 Caguas 12771  
Customer Country 31665 2 EE. UU. 18461  
Customer Segment 31665 3 Corporate 14224  
Customer State 31665 44 PR 13204  
Department Name 31664 11 Apparel 11096  
Market 31664 5 Europe 8760  
Order City 31664 2855 Santo Domingo 423  
Order Country 31664 151 Estados Unidos 4358  
Order Region 31664 23 Central America 4781  
Order State 31664 952 Inglaterra 1156  
Order Status 31664 9 COMPLETE 11460  
Product Name 31664 91 Perfect Fitness Perfect Rip Deck 6145  
Shipping Mode 31664 4 Standard Class 18589

**Data visualization**

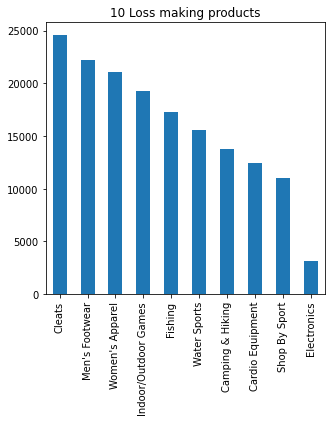
Heatmap is plotted so correlation between each column can be observed. From the heatmap following observations are made:

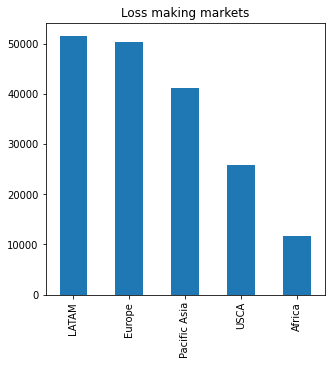
* The total price and order item quantity, days for shipping and days for shipment are positively correlated.
* Late delivery risk shows a weak positive correlation with days for shipping and it shows a weak negative correlation with the days for shipment.
* Product price and order item quantity also shows a negative correlation.

Overall, the feature columns are not showing correlation so multicollinearity is not present.



Bar chart is plotted to observe which all products are making a loss. Similarly, loss making markets are also observed using bar charts. After finding the fraud transactions from the order status the products with fraud activity are also visually observed using bar chart.





**Data Preparation**

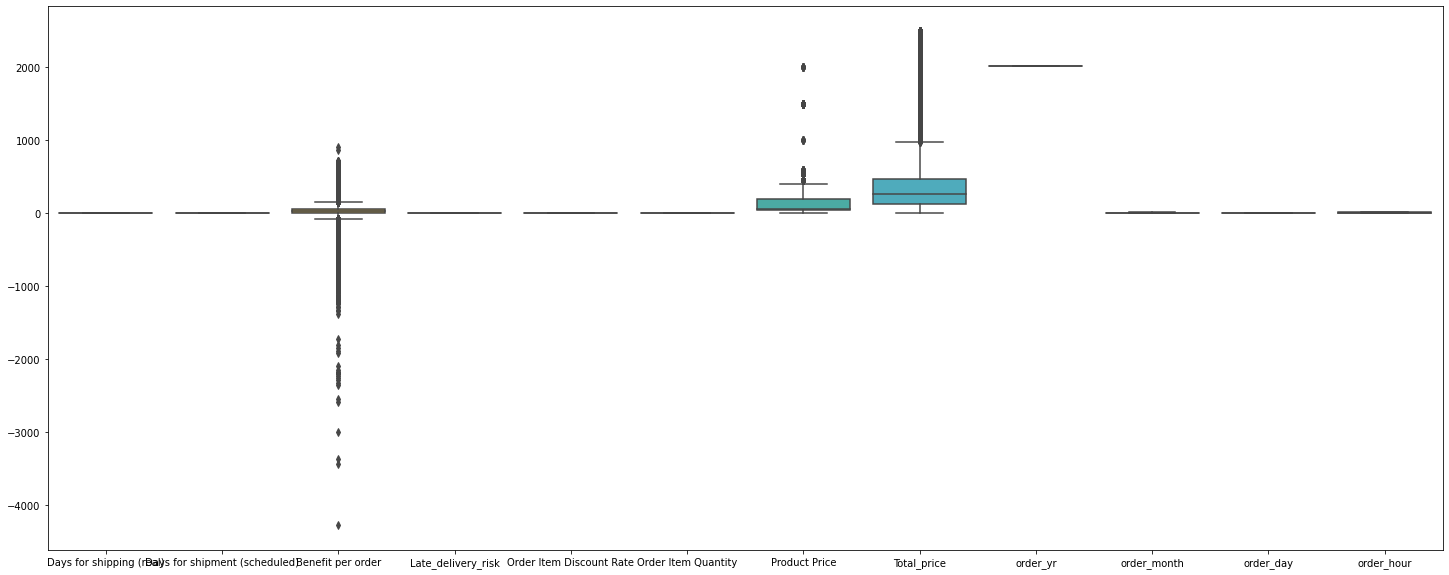
**Detecting and handling missing values and outliers**: It is important steps in data analysis. Missing values affect the accuracy of the result and also it may lead to biased result. Whereas outliers are the abnormal values that happen to have deviated from the normal distribution pattern of data distribution. Boxplot is used to identify outliers, while the isnull function is used to identify missing values.

There are missing values present and we treat null values by dropping them using dropna function. The presence of outliers is shown by the, the dataset. The output of the isnull().sum() function indicates that there are missing values present in the data frame df.

df1.isnull().sum()

Type 0  
Days for shipping (real) 0  
Days for shipment (scheduled) 0  
Category Name 0  
Customer City 0  
Customer Country 0  
Customer Segment 0  
Customer State 0  
Department Name 1  
Market 1  
Order City 1  
Order Country 1  
Order Item Discount Rate 1  
Order Item Quantity 1  
Order Region 1  
Order State 1  
Product Name 1  
Product Price 0  
Shipping Mode 1  
Total\_price 0  
order\_yr 1  
order\_month 1  
order\_day 1  
order\_hour 1  
late\_delivery 0  
fraud 0  
dtype: int64

df1 = df1.dropna(how='any',axis = 0)



From the boxplot it is clear that the outliers are present in the columns ‘Benefit per order’, ‘Product price’, ‘Total price’. Outliers are treated using the winsorization method with 0.05 as lower and 0.95 as upper bound.

**Duplicate data**: no duplicate data is found in the dataset considered.

**Label encoding and standardizing:** It is evident from the data information that some of the dataset's columns are of the object data type. The nunique function is used to determine the number of distinct values in each data column, while the LabelEncoder is used to transform non-numerical labels to numerical labels. Since the majority of machine learning models can only be used with numerical data label encoding is necessary. standardization is a method of feature scaling in which data values are rescaled to fit the distribution within a range where the mean is zero and the standard deviation is 1, thus the same scale. Some columns of the dataset have large differences between their ranges, these columns are taken into consideration and standardized using the StandardScalar function. Order status, the column name is modified to fraud and the target column is encoded as 0 for non-fraud and 1 for fraud transfer.

# Creating Binary enocode for Suspected Fraud and Late delivery  
df1['late\_delivery']=np.where(df1['Delivery Status'] == 'Late delivery', 1, 0).astype('int64')  
df1['fraud'] = np.where(df1['Order Status'] == 'SUSPECTED\_FRAUD', 1, 0).astype('int64')  
df1['fraud'].value\_counts()

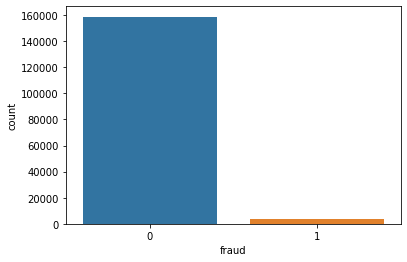
0 31143  
1 522  
Name: fraud, dtype: int64

## label encoding the data so that we can do our data mdoelling  
from sklearn.preprocessing import LabelEncoder,StandardScaler  
le = LabelEncoder()  
df1['Type'] = le.fit\_transform(df1['Type']).astype('int64')  
df1['Category Name'] = le.fit\_transform(df1['Category Name']).astype('int64')  
df1['Customer City'] = le.fit\_transform(df1['Customer City']).astype('int64')  
df1['Customer Country'] = le.fit\_transform(df1['Customer Country']).astype('int64')  
df1['Customer Segment'] = le.fit\_transform(df1['Customer Segment']).astype('int64')  
df1['Customer State'] = le.fit\_transform(df1['Customer State']).astype('int64')  
df1['Department Name'] = le.fit\_transform(df1['Department Name']).astype('int64')  
df1['Market'] = le.fit\_transform(df1['Market']).astype('int64')  
df1['Order City'] = le.fit\_transform(df1['Order City']).astype('int64')  
df1['Order Country'] = le.fit\_transform(df1['Order Country']).astype('int64')  
df1['Order State'] = le.fit\_transform(df1['Order State']).astype('int64)  
df1['Order Region']=le.fit\_transform(df1['Order Region']).astype(‘int64)  
df1['Product Name']=le.fit\_transform(df1['Product Name']).astype('int64)  
df1['Shipping Mode']=le.fit\_transform(df1['Shipping Mode']).astype(int6)

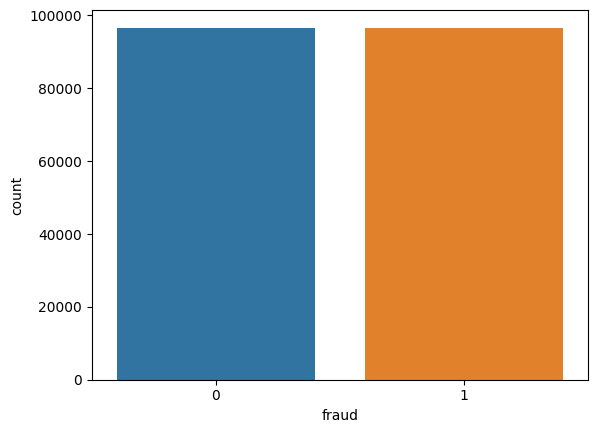
ss = StandardScaler()  
df1[['Total\_price','Product Price']] = pd.DataFrame(ss.fit\_transform(df1[['Total\_price','Product Price']]))

A sample dataset of about 10% of the data is separated from the dataset using sample function and that dataset is taken for final validation of the model. Later the feature and target variables of the dataset for training and testing is defined.

**Treating imbalanced data**: Countplot of the fraud column is plotted and from visualization it was observed that the dataset considered is imbalanced, since the number of fraud transaction is only 4062 and the non-fraud transaction is about 176457. If not treated this will degrade the performance of the classifier model. Most of the predictions will correspond to the majority class and treat the minority class as noise in the data and ignore them. This will result in a high bias in the model. So, SMOTE technique is used which will generate a synthetic sample of the minority class. This algorithm helps to overcome the overfitting problem posed by random oversampling. As a result, both non-fraud and fraud have an equal value count. The dataset is now prepared for data modelling.



from imblearn.over\_sampling import SMOTE  
sm = SMOTE(sampling\_strategy = 'minority',random\_state=42)  
X, y = sm.fit\_resample(X, y)  
# Create a new dataframe with resampled data  
df1 = pd.concat([pd.DataFrame(X), pd.DataFrame(y)], axis=1)  
# Check the value counts of the target variable after oversampling  
print(df1['fraud'].value\_counts())



0 30830  
1 30830  
Name: fraud, dtype: int64

**Model Selection**

Next step is to find the model suitable for the project. Machine learning can be understood as a program that has been trained to find patterns within new data and make predictions. These models are represented as a mathematical function that takes requests in the form of input data, makes predictions on input data, and then provides an output in response. There are various types of machine learning models available based on different objective and data sets. Each machine learning algorithm settles into one of the following models: supervised learning, unsupervised learning, reinforcement learning, forecasting and optimization. Supervised learning includes classification and regression. Unsupervised learning includes clustering, association rule and dimensionality reduction.

From observing the objective and the data set considered machine learning technique is identified. Here the objective is to predict the future transactions as fraud or non-fraud. Since the dataset provided consist of target and feature variables supervised learning is to be considered. Since, the problem has only two possible classes and the target variable is binary, the supervised learning technique to be considered is binary classification. Thus, the machine learning models selected for the analysis are decision tree classifier, random forest classifier, MLP classifier, and XGB classifier.

**MLP classifier**: Multilayer perceptron is a feedforward artificial neural network model that maps input data sets to a set of appropriate outputs. An MLP consists of multiple layers and each layer is fully connected to the following one. MLP uses backpropagation for training the network.

**Decision tree classifier**: it generates output as a binary tree-like structure which contains rules to predict the target variable. The tree classification algorithm provides an easy-to-understand description of the underlying distribution of the data. It has a hierarchical tree structure, which consist of root node, branches, internal nodes and leaf nodes.

**Random forest classifier**: it is an ensemble approach that utilizes a number of classifiers to work together in order to identify the class label for unlabelled instances. It builds decision tree on different samples and takes their majority vote for finding the output for classification problems. It adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

**XGB classifier**: it is an optimized distributed gradient boosting library designed for efficient and scalable training of machine learning models. It is an ensemble learning method that combines the predictions of multiple weak models to produce a stronger prediction. XGBoost stands for “Extreme Gradient Boosting” and it has become one of the most popular and widely used machine learning algorithms due to its ability to handle large datasets and its ability to achieve state-of-the-art performance in many machine learning tasks such as classification and regression.

**Model evaluation methods**

The most important task of machine learning is to evaluate the performance. The evaluation metrices for classification and regression are different. Some evaluation metrices such as, precision-recall, are useful for multiple tasks. Since here classification model is considered the evaluation metric used for measuring the performance of the such models are accuracy, confusion matrix, log-loss, and AUC-ROC. Precision-recall is a widely used metrics for classification problem.

**Confusion matrix:** it is a performance measurement for the machine learning classification problems where the output can be two or more classes. It is a table with combination of predicted and actual values. It is extremely useful for measuring the Recall, Precision, Accuracy, and AUC-ROC curves. It has four components: true positive (predicted as positive when it is correct), true negative (predicted as negative when it is correct), false positive (predicted as positive and that’s not correct), false negative (predicted as negative and its wrong).

**Accuracy:** Accuracy simply measures how often the classifier correctly predicts. It is defined as the ratio of the number of correct predictions to the total number of predictions. Accuracy is only useful if the target class is well balanced but not a good choice for the unbalanced classes.

Formula: Accuracy = (TP + TN) / (TP + TN + FP + FN)

**Precision:** Precision is the ratio of true positive predictions to the total number of positive predictions made by the model.It measures the accuracy of positive predictions and the ability of the model not to label negative samples as positive.

Formula: Precision = TP / (TP + FP)

A high precision indicates that the model is good at avoiding false positives.

**Recall (Sensitivity or True Positive Rate):** Recall is the ratio of true positive predictions to the total number of actual positive samples in the dataset.It measures the ability of the model to correctly identify positive samples.

Formula: Recall = TP / (TP + FN)

A high recall indicates that the model is good at avoiding false negatives.

**F1 Score:** The F1 score is the harmonic mean of precision and recall.It is a balanced metric that combines both precision and recall, giving equal importance to false positives and false negatives.

Formula: F1 Score = 2 \* (Precision \* Recall) / (Precision + Recall)

**Splitting Train, Test, Validation data**

Train-Valid-Test spilt is a technique to evaluate the performance of the machine learning model. This will prevent the model from overfitting and is used to evaluate the model effectively. The dataset is split into three train dataset, test dataset and valid dataset. The set of data used for learning, to fit the parameters of the machine learning model is called train dataset. The set of data used to provide an unbiased evaluation of a model fitted on the training dataset while tuning model hyperparameters is called valid dataset. Test dataset is the set of data used to provide an unbiased evaluation of a final model fitted on the training dataset.

## a sample dataset for validation is created.  
sample\_df = df1.sample(frac=0.01,random\_state=1)  
sample\_df.shape

(317, 26)

## defining the feature and traget variables  
X = df1.loc[:,df1.columns != 'fraud']  
y = df1['fraud']

from sklearn.model\_selection import train\_test\_split  
Xtrain,Xtest,ytrain,ytest = train\_test\_split(X,y,test\_size = 0.3,random\_state = 42)

For validating the selected model 10% of the data is selected and removed from the original -dataset. Train test split is done using the train\_test\_split technique in scikit-learn library. The test size is a parameter in this function. For splitting the dataset in the ratio of 70:30 test\_size is given as 0.3. Thus, the data for training, testing and validating the model is split.

**Building Models**

Model building is the essential part of data analytics and is used to extract insights and knowledge from the data to make business decisions and strategies. Model building in data analytics is aimed at achieving not only high accuracy on training data but also the ability to generalize and perform well on new, unseen data. For fitting the data to the selected classification models and predicting the future transactions the respective machine learning models are imported from **sklearn** library. Model is trained using the training data and the predictions are done on the test dataset of each model. The accuracy, confusion matrix and classification report of each model are also obtained for evaluating the model built.

**MLP classifier**: MLP classifier is defined and the training model is fitted. The accuracy of the train and test dataset are checked. The MLP classifier model shows an accuracy of 50 % for training and 49 % for testing data. Precision value indicates that the model is not good at avoiding false positives and recall indicates this model is not good at avoiding false negatives.

print(confusion\_matrix(ytest,mlppred))

print('train data accuracy', accuracy\_score(ytrain,mlptrain\_pred))  
print('test data accuracy', accuracy\_score(ytest,mlppred))  
print('precision,recall and fscore', precision\_recall\_fscore\_support(ytest, mlppred, average='macro'))

[[9126 78]  
 [8859 435]]

train data accuracy 0.5175617441267781  
test data accuracy 0.516866688290626  
precision,recall and fscore (0.6776880343756249, 0.5191649068289, 0.3800065190652281, None)

**Decision Tree classifier**: decision tree classifier model is defined and fitted. This model shows an accuracy of about 95% for both train and test data. Precision and recall are high which indicates it is good at avoiding false positives and false negatives respectively

print(confusion\_matrix(ytest,dtcpred))

print('train data accuracy',accuracy\_score(ytrain,dtctrain\_pred))  
print('test data accuracy', accuracy\_score(ytest,dtcpred))  
print('precision,recall and fscore',precision\_recall\_fscore\_support(ytest, dtcpred, average='macro'))

[[8697 507]  
 [ 72 9222]]

train data accuracy 0.9711783513275567  
test data accuracy 0.9686993188452806  
precision,recall and fscore (0.9698385079302889, 0.9685841603659004, 0.9686740856172025, None)

**Random Forest classifier**: Random Forest classifier model is defined and fitted and this model shows about 95 % accuracy with the train data and test data. Precision and recall are high which indicates it is good at avoiding false positives and false negatives respectively.

print(confusion\_matrix(ytest,rfcpred))

print('train data accuracy',accuracy\_score(ytrain,rfctrain\_pred))  
print('test data accuracy',accuracy\_score(ytest,rfcpred))  
print('precision,recall and fscore', precision\_recall\_fscore\_support(ytest, rfcpred, average='macro'))

[[8679 525]  
 [ 24 9270]]

train data accuracy 0.9724062833047589  
test data accuracy 0.9703211157963023  
precision,recall and fscore (0.9718217776729047, 0.9701886358107977, 0.9702907896993179, None)

**XGB classifier**: XGB classifier model is defined and fitted. This model shows an accuracy of about 99% with the train and test data. Precision and recall are high which indicates it is good at avoiding false positives and false negatives respectively.

confusion\_matrix(ytest,xgbpred)

print('train data accuracy', accuracy\_score(ytrain,xgbtrain\_pred))  
print('test data accuracy', accuracy\_score(ytest,xgbpred))  
print('precision,recall and fscore', precision\_recall\_fscore\_support(ytest, xgbpred, average='macro'))

array([[9159, 45],  
 [ 61, 9233]])

train data accuracy 0.9991891015244891  
test data accuracy 0.9942696507730565  
precision,recall and fscore (0.9942668823550591, 0.9942737235810405, 0.9942695590661742, None)

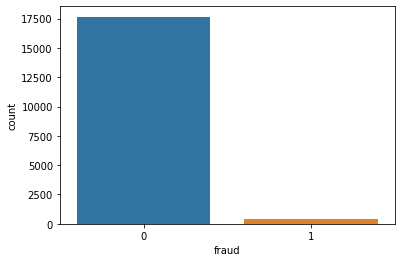
**Models’ Summary**

After model building the next part is to select the model which best for our objective. Here the objective is to find the fraud transactions. From fitting the train test data to the different models and evaluating the model, best model with high accuracy is selected as the final model. In this case of fraud detection, **Random Forest classifier model** and **XGB classifier model** are the optimal model since it shows high accuracy score with both train and test data. The rest model shows low accuracy level with respect to the other models.

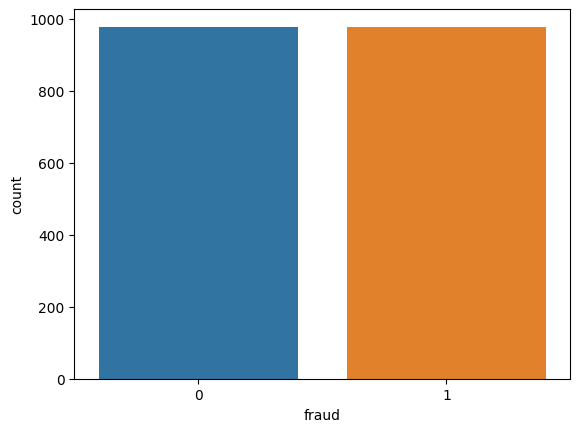
**Model Validation**

Validating the selected model is an important step in data analytics. The selected model is used to predict the target variable of the valid dataset created earlier.

The sample dataset created earlier, which is already checked for missing values and outliers and data preprocessing steps such as label encoding, standardizing is done.



from imblearn.over\_sampling import SMOTE  
sm = SMOTE(sampling\_strategy='minority', random\_state=42,k\_neighbors = 3)  
Xsample, ysample = sm.fit\_resample(Xsample, ysample)  
sample\_df = pd.concat([pd.DataFrame(Xsample),pd.DataFrame(ysample)],axis = 1)  
sample\_df ['fraud'].value\_counts()



0 312  
1 312  
Name: fraud, dtype: int64

From visualizing the count plot of the sample data, it is clear that the data is imbalanced. SMOTE technique used to balance the dataset.

Since the final model chosen are the random forest classifier and XGB classifier model, new values are predicted for the sample data separated earlier and that predicted values are added as a new column to the sample dataset.

Comparing the actual values of the target in valid dataset and the new predicted values using the model selected will provide a clear idea about the accuracy of the model chosen. In this problem the actual and predicted transactions are provided in the data frame which can be cross validated. The value count of the predicted column is calculated to check how many are predicted as fraud (1) and non-fraud (0).

print(confusion\_matrix(ysample,xgb\_pred))  
print('xgb\_accuracy=',accuracy\_score(ysample,xgb\_pred))

precision\_recall\_fscore\_support(ysample, xgb\_pred, average='macro')

[[311 1]  
 [ 3 309]]  
xgb\_accuracy= 0.9935897435897436

(0.9936100267104993, 0.9935897435897436, 0.993589677737485, None)

print(confusion\_matrix(ysample,rfc\_pred))  
print('rfr\_accuracy=',accuracy\_score(ysample,rfc\_pred))

precision\_recall\_fscore\_support(ysample, rfc\_pred, average='macro')

[[287 25]  
 [ 1 311]]  
rfr\_accuracy= 0.9583333333333334

(0.9610615079365079, 0.9583333333333333, 0.9582716049382716, None)

From the confusion matrix it is clear that the out of 3552 data for XGB classifier model 1755 are predicted correctly as 1 and 1766 are predicted correctly as 0. Whereas 21 data is predicted as FN and 10 data predicted as FP. From Random Forest classifier model’s 1609 data predicted as 0 correctly while 2 are predicted as 0 wrongly. 1774 data predicted as 1 correctly and 167 as 1 wrongly.

**Conclusion**

Fraud transaction detection is a binary classification problem where the considered dataset has feature and target variable in which the target variable has two attributes 1 for fraud transaction and 0 for non-fraud transaction. From building models such as MLP, decision tree, random forest and XGB classifiers, Random Forest classifier model and XGB model shows higher accuracy which is chosen for further validation using the sample dataset. The predicted values show an accuracy of 95 and 99 %. But from the confusion matrix we can conclude random forest is the best model since it only predicts fraud as non-fraud only twice and it predict non-fraud as fraud for 167 times but it can be neglected.

Predicting the future transactions, as fraud and non-fraud is important in supply chain management. By preventing the fraud transactions, businesses can safeguard their operations, minimize financial losses, and maintain the integrity of their supply chain processes.

**References**

<https://github.com/subhanjandas/Artificial-Neural-Networks-for-Fraud-Detection-in-Supply-Chain-Analytics-MLPClassifier-and-Keras>

<https://www.qeios.com/read/1VZC8W#:~:text=By%20analyzing%20large%20data%20sets,not%20be%20able%20to%20identify>.