|  |
| --- |
| **DataCO MARKET ANALYSIS** |
|  |
| Team Members:  Fayiq Ahmed K, Supriyo Dey, Akhil Raj, Bharath B, Sai Charan  Mentor: Animesh Tiwari |

# **Introduction**

In recent years, the rise of the Internet of things (IoT) as an emerging technology has been unbelievable, more companies are moving towards the adoption of these technologies and many IoT sensors are being deployed to share information in real-time which leads to the generation of a huge amount of data. This data when used correctly, will be very helpful to the company to discover hidden patterns for better decision making in the future. For example, with the DataCo company, dataset customer segmentation analysis was performed in this project which helps the company to better understand its customers and target them to increase customer responsiveness and the company's revenue. With a lot of options available to analyze data, it is very difficult to decide which method and machine learning model to use since the performance of the model vary on the parameters available in the data.

This project aims to compare 10 popular machine learning classifiers, and measure their performance to find out which machine learning model performs better. Since the dataset used is related to supply chain important parameters are identified and the machine learning models are trained with the dataset for detection of fraud transactions, late delivery of orders, sales revenue and quantity of products which customer orders.

he machine learning classifiers used in this project are Logistic Regression, K-Nearest Neighbor Classiifer, Random Forest Classifier , Decision Tree Classifier, Naïve Bayes Classifier, Bagging, Boosting.

# **Data Collection**

The dataset consists of roughly 180k transactions from supply chains used by the company DataCo Global for 3 years.

The continuous columns are:

['Days for shipping (real)', 'Days for shipment (scheduled)', 'Benefit per order', 'Sales per customer', 'Late\_delivery\_risk', 'Category Id', 'Customer Id', 'Customer Zipcode', 'Department Id', 'Latitude', 'Longitude', 'Order Customer Id', 'Order Id', 'Order Item Cardprod Id', 'Order Item Discount', 'Order Item Discount Rate', 'Order Item Id', 'Order Item Product Price', 'Order Item Profit Ratio', 'Order Item Quantity', 'Sales', 'Order Item Total', 'Order Profit Per Order', 'Order Zipcode', 'Product Card Id', 'Product Category Id', 'Product Description', 'Product Price', 'Product Status']

The Categorical Columns are:

['Type', 'Delivery Status', 'Category Name', 'Customer City', 'Customer Country', 'Customer Email', 'Customer Fname', 'Customer Lname', 'Customer Password', 'Customer Segment', 'Customer State', 'Customer Street', 'Department Name', 'Market', 'Order City', 'Order Country', 'order date (DateOrders)', 'Order Region', 'Order State', 'Order Status', 'Product Image', 'Product Name', 'shipping date (DateOrders)', 'Shipping Mode']

In Total there are 53 Columns Initially in the dataset

# **Data Cleaning**

The total data set consists of 180519 records and 53 columns

Only the below Columns has null values in the dataset:

Order Zipcode 86.239676

Product Description 100.000000

Order Zipcode and order description both have highest null values, and for the given problem the Zipcode is not required and also the description column is empty, so dropping the two columns.

To make it easier for analysis some unimportant columns are dropped

Customer Zipcode,Product Status,Late\_delivery\_risk,Customer Email,Customer Password,Customer Fname,Customer Lname,Product Image,shipping date (DateOrders), Order Profit Per Order.

The target variable for our problem is Order Status Column.

The unique categories present for the order status are:

['COMPLETE' 'PENDING' 'CLOSED' 'PENDING\_PAYMENT' 'CANCELED' 'PROCESSING'

'SUSPECTED\_FRAUD' 'ON\_HOLD' 'PAYMENT\_REVIEW']

This is the distribution of each category in the Order Status Column:

COMPLETE 32.955534

PENDING\_PAYMENT 22.065267

PROCESSING 12.132795

PENDING 11.204915

CLOSED 10.866446

ON\_HOLD 5.431007

SUSPECTED\_FRAUD 2.250179

CANCELED 2.045214

PAYMENT\_REVIEW 1.048643

Chart, bar chart

Description automatically generated

The Categories Complete, Processing, Closed, Cancelled can be categorised as NON-FRAUD as the order is processed or in processing state. The Categories PENDING\_PAYMENT, PENDING, ON\_HOLD, SUSPECTED-FRAUD, PAYMENT\_REVIEW can be categorised as FRAUDULANT as the order is on hold, or on review.

After the change these are the categories in the Order Status column and there weightage:

NOT\_FRAUD 57.999989

FRAUD 42.000011

**A picture containing logo

Description automatically generated**

Days for shipping(real) and Days for shipping(scheduled) are telling the number of days it took to deliver the order, why not create 1 column telling if the order shipment was delayed, on-time or before-time

After the necessary changes we where able to create a new column Order Shipment, with the below categories and weightages:

DELAY 57.279289

BEFORE\_TIME 24.022956

ON\_TIME 18.697755

# **Data Visualization**

**Univariate and Bivariate Analysis:**

1. Top 10 departments which have the highest Fraud:

TeTChart, histogram

Description automatically generated

The above plot shows the top 10 Departments that are effected by fraud.

1. Top 10 Products which have the he highest fraud:Chart

   Description automatically generated

The above plot shows the top 10 products that are effected by fraud.

1. Shipping Mode which have the highest fraud:

A picture containing chart

Description automatically generated

The above plot shows which type of shipping mode does a fraud order go through.

1. Region wise analysis for the departments: Chart, bar chart

   Description automatically generated

The above plot shows the fraud order trend region wise.

1. Top 10 product s effected by fraud:Chart

   Description automatically generated

The above plot shows the worst effected products for fraud orders.

1. Year wise analysis for fraud orders:

Line chart

Description automatically generated

Year wise and month wise trend for fraud orders placed

1. Month wise analysis for fraud orders:

Chart, line chart

Description automatically generated

1. Date wise analysis for fraud orders:

Chart, line chart

Description automatically generated

1. Day wise analysis for fraud orders:

Icon

Description automatically generated with medium confidence

1. Weightage of payment methods used by customers:

Chart, pie chart

Description automatically generated

The above pie chart shows the weightage of payments methods in the dataset.

1. Visualizing the customers location present in the dataset via Portly:

Map

Description automatically generated

# **Outler Treatment**

Below shows the percentage of outliers in each column:

Benefit per order 10.493078

Category Id 0.000000

Category Name 0.000000

Customer City 0.000000

Customer Country 0.000000

Customer Id 0.663642

Customer Segment 0.000000

Customer State 0.000000

Customer Street 0.000000

Delivery Status 0.000000

Department Id 0.200533

Department Name 0.000000

Market 0.000000

Order City 0.000000

Order Country 0.000000

Order Customer Id 0.663642

Order Id 0.000000

Order Item Cardprod Id 0.000000

Order Item Discount 4.175184

Order Item Discount Rate 0.000000

Order Item Id 0.000000

Order Item Product Price 1.134507

Order Item Profit Ratio 9.583479

Order Item Quantity 0.000000

Order Item Total 1.076341

Order Region 0.000000

Order Shipment 0.000000

Order State 0.000000

Order Status 0.000000

Product Card Id 0.000000

Product Category Id 0.000000

Product Name 0.000000

Product Price 1.134507

Sales 0.270332

Sales per customer 1.076341

Shipping Mode 0.000000

Type 0.000000

order date 0.000000

order day 0.000000

order month 0.000000

order year 0.000000

% of loss of data from the original data set is: 17.40% Since the percentage of removal is very high we will not be removing the outliers from the original dataset as data loss is very high

# **Checking for Multi-Collinearity**

We can visualize the multi-collinearity of the dataset via a Heatmap

Chart

Description automatically generated

There is high multi-collinearity present in the data, we will be treating the same using VIF(Variance Inflation Factor).

After calculating VIF and via Backward elimination keeping only features with VIF less that 10, we have the below features:

VIF\_Factor Features

0 3.407359 Benefit per order

1 3.840572 Order Customer Id

2 7.183878 Order Item Discount

3 6.882066 Order Item Discount Rate

4 4.272494 Order Item Id

5 3.414860 Order Item Profit Ratio

6 3.809974 Order Item Quantity

7 6.938691 Product Category Id

8 6.216378 Product Price

9 4.259147 order month

10 3.787456 order date

11 3.013430 order day

And, now after removing the multi-collinearity from the dataset the heatmap is as shown below:

Chart, histogram

Description automatically generated

# **Checking for Model Assumptions:**

Assumption 1:

The target variable i.e., Order Status is a categorical variable.

The datatype of Order Status column is: object and it has 2 categories.

Assumption 2:

As shown in the above heatmap we have removed the multicollinearity from all the numeric columns of the dataset

# **Final Data-Set Details:**

The total number of continuous columns now that can be used in the dataset are: 12

The total number of categorical columns that can be used in the dataset are 18

# **Feature Engineering:**

For feature engineering we have decided to normalize the continuous columns using PowerTransformer to make it normally distributed, and for the categorical columns we used LabelEncoder.

**Distribution of numerical columns before applying PowerTransformer:**

Graphical user interface

Description automatically generated with low confidence

**Distribution of numerical columns after applying PowerTransformer:**

Graphical user interface, application

Description automatically generated with medium confidence

# **Statistically checking the significance of each variable:**

We applied the StatsModel Logit() model in our dataset to get features which have p-value < 0.05.

To select the significant variables we will selected them based on the p-values of the same, the hypothesis is such that:

H0: Feature is significant where p-value < 0.05

H1: Feature is not significant where p-Value > 0.05

And after getting the base model, we have applied backward elimination to get the best features from the dataset so that we can proceed with the model making and predict the target. The best features are listed below:

Graphical user interface

Description automatically generated with low confidence

Using the above features we created a dataset so that we proceed with the final model building.

# **Data Modelling**

To measure the performance of different models the machine learning models are trained to detect fraud. And the different measures used to measure the performance of a Regression model are:

**Accuracy:** Accuracy refers to how close a measurement is to the true or accepted value.

**Precision:** Precision talks about how precise/accurate your model is out of those predicted positive, how many of them are actual positive.

**Re-call:** Recall actually calculates how many of the Actual Positives our model capture through labeling it as Positive (True Positive).

**F1-Score** F1 which is a function of Precision and Recall.

1. **Base Model (Logistic regression):**

**Chart, treemap chart

Description automatically generated**

True negative: 23765

True Positive: 13397

False Negative: 9406

False Positive: 7588

The accuracy of the base model is: 68.62028214786912

The precision of the base model is: 63.84083869430546

The re-call value of the base mode is: 58.75104152962329

The f1-score of the base model is: 61.19028044213026

This is the Benchmark that we set for future Machine Learning models, we would be using advanced regression models, tune the same using hyperparameter tuning and get the best model from the same.

Before we proceeded, we checked if there is any im-balance in the target variable:

Logo

Description automatically generated with medium confidence

From the above plot we see that there is no in-balance present in the target variable.

So going ahead with model building:

First, we will do base model and then tune the same using hyperparameter tuning.

We also split the data to train and test dataset using train\_test\_split, and kept the test size as 0.25

1. **Prediction using KNN:**

Chart, treemap chart

Description automatically generated

True negative: 22675

True Positive: 14115

False Negative: 5017

False Positive: 3323

The accuracy of the base model is: 81.52005317970308

The precision of the base model is: 80.94391558664985

The re-call calue of the base mode is: 73.776918252143

The f1-score of the base model is: 77.19442165709597

1. **Prediction using Decision Tree:**

Chart, treemap chart

Description automatically generated

True negative: 22897

True Positive: 16180

False Negative: 2952

False Positive: 3101

The accuracy of the base model is: 86.58763571903391

The precision of the base model is: 83.91680929412375

The re-call calue of the base mode is: 84.57035333472716

The f1-score of the base model is: 84.24231380001562

1. **Prediction using Random Forest:**

Chart, treemap chart

Description automatically generated

True negative: 24704

True Positive: 15724

False Negative: 3408

False Positive: 1294

The accuracy of the base model is: 89.58120983824507

The precision of the base model is: 92.39628628510988

The re-call calue of the base mode is: 82.18691197992891

The f1-score of the base model is: 86.99308437067772

1. **Prediction using Naïve Bayes:**

Chart, treemap chart

Description automatically generated

True negative: 19574

True Positive: 16606

False Negative: 2526

False Positive: 6424

The accuracy of the base model is: 80.16840239308664

The precision of the base model is: 72.10594876248372

The re-call calue of the base mode is: 86.79698933723604

The f1-score of the base model is: 78.77235425264456

We now will apply Hyperparameter tuning and using the best parameters for the same.

1. **Hyperparameter tuning for KNN**

Best Parameters for KNN {'n\_neighbors': 25, 'p': 2}

Chart, treemap chart

Description automatically generated

True negative: 22295

True Positive: 14518

False Negative: 4614

False Positive: 3703

The accuracy of the base model is: 81.57101706182141

The precision of the base model is: 79.6772954283519

The re-call calue of the base mode is: 75.88333681789672

The f1-score of the base model is: 77.73405081251842

1. **Hyperparameter Tuning for Random Forest:**

Best parameters for Random Forest:

{'criterion': 'gini',

'max\_depth': 13,

'max\_features': 3,

'min\_samples\_leaf': 1,

'min\_samples\_split': 2,

'n\_estimators': 165}

Chart, treemap chart

Description automatically generated

True negative: 23046

True Positive: 15380

False Negative: 3752

False Positive: 2952

The accuracy of the base model is: 85.14513627298913

The precision of the base model is: 83.89701069168667

The re-call calue of the base mode is: 80.38887727367761

The f1-score of the base model is: 82.10548793508434

1. **Hyper Parameter tuning for Decision tree:**

Best parameters for Decision tree are:

{'criterion': 'gini',

'max\_depth': 19,

'max\_features': 6,

'min\_samples\_leaf': 26,

'min\_samples\_split': 15}

Chart, treemap chart

Description automatically generated

True negative: 24334

True Positive: 14008

False Negative: 5124

False Positive: 1664

The accuracy of the base model is: 84.95900731220918

The precision of the base model is: 89.38233792751403

The re-call value of the base mode is: 73.21764582897762

The f1-score of the base model is: 80.49649465578669

1. **Prediction using Bagging:**

First we applied bagging without any base estimators:

Chart, treemap chart

Description automatically generated

True negative: 24748

True Positive: 15250

False Negative: 3882

False Positive: 1250

The accuracy of the base model is: 88.62840682472857

The precision of the base model is: 92.42424242424242

The re-call value of the base mode is: 79.70938741375706

The f1-score of the base model is: 85.5972159856309

1. **Bagging with Hyperparameter tuning:**

Now, for hyperparameter tuning for bagging we have used the base estimators we got for knn, decision tree and random forest with the best parameters.

Best Parameters for Bagging:

{'base\_estimator': RandomizedSearchCV(cv=3, estimator=DecisionTreeClassifier(random\_state=0),

n\_jobs=-1,

param\_distributions={'criterion': ['gini', 'entropy'],

random\_state=0, scoring='roc\_auc'),

'max\_features': 9,

'max\_samples': 42,

'n\_estimators': 120}

Chart, treemap chart

Description automatically generated

True negative: 20633

True Positive: 15700

False Negative: 3432

False Positive: 5365

The accuracy of the base model is: 80.50742300022158

The precision of the base model is: 74.53121291241396

The re-call value of the base mode is: 82.06146769809742

The f1-score of the base model is: 78.11528223499266

1. **Boosting using AdaBoost:**

We, have used base model for AdaBoost.

Chart, treemap chart

Description automatically generated

True negative: 25998

True Positive: 10503

False Negative: 8629

False Positive: 0

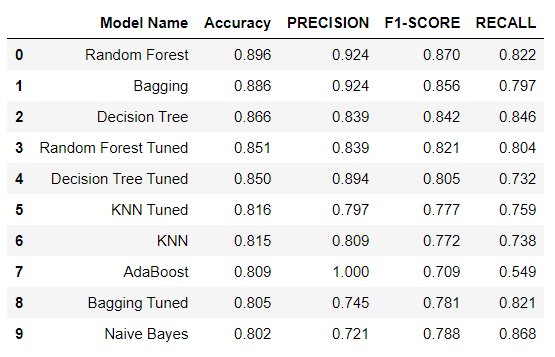
The accuracy of the base model is: 80.87968092178151

The precision of the base model is: 100.0

The re-call value of the base mode is: 54.89755383650429

The f1-score of the base model is: 70.88240256453518

# **Final Results:**



**ROC Curve for the top 4 performing models:**

**A picture containing diagram

Description automatically generated**

From the above table and the results that we have achieved here, we see that **RANDOM FOREST** will give us the best accuracy to detect a FRAUD activity.

# Model Implications

We see that Random Forest is the best model among all the models that have been built here. Compared to the base model which only gave us a accuracy of 68%, we are getting a accuracy of 89% through random forest.

Some details about the base model:

The accuracy of the model = TP+TN / (TP+TN+FP+FN) = 0.6862028214786912

The Miss-classification = 1-Accuracy = 0.31379717852130884

Sensitivity or True Positive Rate = TP / (TP+FN) = 0.5875104152962329

Specificity or True Negative Rate = TN / (TN+FP) = 0.7579816923420406

Positive Predictive value = TP / (TP+FP) = 0.6384083869430546

Negative predictive Value = TN / (TN+FN) = 0.7164390582134997

Positive Likelihood Ratio = Sensitivity / (1-Specificity) = 2.427545341431575

Negative likelihood Ratio = (1-Sensitivity) / Specificity = 0.5441946538698594

###### From the above statistics it is clear that the model has a very high difference between sensitivity and specificity.

Some details about the model built using random forest

The accuracy of the model = TP+TN / (TP+TN+FP+FN) = 0.8958120983824507

The Miss-classification = 1-Accuracy = 0.10418790161754932

Sensitivity or True Positive Rate = TP / (TP+FN) = 0.8218691197992891

Specificity or True Negative Rate = TN / (TN+FP) = 0.9502269405338872

Positive Predictive value = TP / (TP+FP) = 0.9239628628510989

Negative predictive Value = TN / (TN+FN) = 0.8787706317586795

Positive Likelihood Ratio = Sensitivity / (1-Specificity) = 2.427545341431575

Negative likelihood Ratio = (1-Sensitivity) / Specificity = 0.5441946538698594

###### From the above statistics it is clear that the model has reduced the difference between sensitivity and specificity, that was seen in the base model.

Inconclusion, we see that if the company has to predict fraud accurately, a model with better accuracy, precision is required.

To improve the model performance, since there are many categorical columns present in the dataset, with many categories in each, we can for further better model building apply some technique to segment the categories into smaller categories and then proceed with better mode building to get more accurate results.

# RFM ANALYSIS TO ANALIZE CUSTOMER RETENTION

We have used the original dataset to analyse the RFM for the DataCO company.

Understanding customer needs and targeting specific clusters of customers based on their need is one way for a supply chain company to increase number of customers and also to gain more profits.Since,purchase history of customers is already available in the dataset, it can use RFM analysis for customer segmentation. Even though there are so many different methods for customer segmentation, RFM analysis is being used because it utilizes numerical values to show Customer recency,frequency and monetary values and also the output results are easy to interpret

After analysis we get the below table:

Graphical user interface, text, application

Description automatically generated

**Plotting the distribution for Recency, Frequency and Monetary value:**

Diagram

Description automatically generated

The total data is divided into 4 quantiles. The R\_Value should be low because it indicates recent customer activity and F\_value, M\_Value should be high since they indicate frequency and total value of purchase. Function is defined to indicate quantiles as numerical values.

Graphical user interface, text, application

Description automatically generated

The RFM score is then calculated from the above table:

#Adding R,F,M Scores to one new column

Customer\_seg['RFM\_Score'] = Customer\_seg.R\_Score.astype(str)+ Customer\_seg.F\_Score.astype(str)+Customer\_seg.M\_Score.astype(str)

A picture containing table

Description automatically generated

It can be seen that there are 33 different customer segments. To make it easier for segmentation individual R,F,M scores are added together

Customer\_seg['RFM\_Total\_Score'] = Customer\_seg[['R\_Score','F\_Score','M\_Score']].sum(axis=1)

There are 9 values in total for customer segmentation. Appropriate names were assigned for each value separately.

Graphical user interface

Description automatically generated

Visually, checking the segmentation, using pie chart.

Chart, pie chart

Description automatically generated

Since total customers are divided into 9 segments it can be seen that, 11.4% customers are at risk of losing them as customers and 11% customers needs attention else even they will be lost eventually.It can be seen that 4.4% of customers are already lost.

Our Top 10 Churned best customers who has not purchased anything in a while

Graphical user interface, application, table

Description automatically generated

Top 10 new best customers who place costly orders often.

Graphical user interface, application, table

Description automatically generated

## THANK YOU!!