Olist E-Commerce

Predicting Customer Satisfaction

Project Report

Group 18

Shreyaskumar Raghavendra Bhat
(502) 751-2480 | bhat.shre@northeastern.edu

Prateek Srigiri Revadi
(857) 576 - 9377 | revadi.p@northeastern.edu

Percentage of Effort Contributed by Shreyaskumar R Bhat: 50%
Percentage of effort Contributed by Prateek S Revadi: 50%

Signature of Shreyaskumar R Bhat: Shreyaskumar Raghavendra Bhat
Signature of Prateek S Revadi: Prateek Srigiri Revadi

Submission Date: 04/30/2022

INTRODUCTION

DATA MINING AND PROJECT SCOPE

The fields of statistics and machine learning converge in the field of data mining. In the realm of statistics, there are several strategies for investigating data and developing models: linear regression, logistic regression, classification models, discriminant analysis, and principal components analysis, to name a few. In contrast to statistics, data mining deals with big datasets in an open-ended manner, making it hard to create the strict restrictions that inference would require around the subject being addressed.

The typical approach to data mining risks overfitting, which occurs when a model is fitted so closely to the available sample of data that it reflects not just structural properties of the data but also random idiosyncrasies. The goal of data mining is to build a model that would explain the trend and predict the outcome of the target variable with comparable accuracy.

The scope of the project is to primarily test our ability to interpret and execute the different data mining techniques and principles on to real world scenarios.

PROBLEM AT HAND

Customer satisfaction is one of the key aspects of running a successful business model. Feedback based business decisions have seen remarkable growth in-terms of business reach, product sales, and optimization of supply networks. For this reason, we chose to select a dataset provided by Olist Stores, a prominent e-commerce website based in Brazil. Olist connects small businesses all over Brazil to channels with a single contract and no trouble. Those businesses can sell their goods on the Olist Store and have them shipped straight to customers through Olist logistics partners. A seller is notified after a buyer orders a product through Olist Store and is responsible for fulfilling that order. The customer receives an email with a satisfaction survey where they can rate his purchase experience and leave some remarks.

Problem Definition and Approach

The goal of this project is to utilize the sales data along with the customer feedback to understand possible correlations and build a classification model to understand the trend in the data. For this, we used multiple machine learning classification algorithms to assess the predicted data to the actual data. Finally, evaluating each classification model based on their performance parameters and selecting the best model that would represent the data.

Data Source

This dataset was generously provided by Olist, which was made publicly accessible on kaggle.com. The dataset has information of 100,000 orders from 2016 to 2018 made at multiple marketplaces in Brazil. Its features allow you to evaluate an order from a variety of perspectives, including order status, pricing, payment, and freight performance, as well as customer location, product qualities, and customer feedback.

Source: Olist, & André Sionek. (2018). Brazilian E-Commerce Public Dataset by Olist. Kaggle. https://doi.org/10.34740/KAGGLE/DSV/195341

License: CC BY-NC-SA 4.0

Data Exploration & Feature Engineering

Datasets

The data includes 9 separate datasets that are linked with a primary key. This is a large dataset that covers several facets of the Brazilian e-commerce industry. It contains information on a variety of businesses that are currently operating on the market. It contains information about 100k orders scheduled from 2016 to 2018. The dataset list is shown in the figure below –

	Dataset	Numeric features	Numeric features name	Object features	Object features name	Bool features
0	Customers	1	customer_zip_code_prefix	4	customer_id, customer_unique_id, customer_city, customer_state	0
1	Geolocations	3	geolocation_zip_code_prefix, geolocation_lat, geolocation_lng	2	geolocation_city, geolocation_state	0
2	Items	3	order_item_id, price, freight_value	4	order_id, product_id, seller_id, shipping_limit_date	0
3	Payments	3	payment_sequential, payment_installments, payment_value	2	order_id, payment_type	0
4	Orders	0		8	order_id, customer_id, order_status, order_purchase_timestamp, order_approved_at, order_delivered_carrier_date, order_delivered_customer_date, order_estimated_delivery_date	0
5	Products	7	product_name_lenght, product_description_lenght, product_photos_qty, product_weight_g, product_length_cm, product_height_cm, product_width_cm	2	product_id, product_category_name	0
6	Reviews	1	review_score	6	review_id, order_id, review_comment_title, review_comment_message, review_creation_date, review_answer_timestamp	0
7	Sellers	1	seller_zip_code_prefix	3	seller_id, seller_city, seller_state	0
8	Category translation	0		2	product_category_name, product_category_name_english	0

***Fig 1** – List of datasets and their associated variables*

A database is a logically ordered collection of data that can be easily accessed, controlled, and updated. The dataset comprises a wealth of information on customer reviews, geolocation, sellers, orders, order items, and much more. The data is in csv format. It also includes product ratings, comments, and metadata from other firms. In order to analyze

the data in a wholistic fashion and for ease of analysis, we merge the datasets based on common attributes.

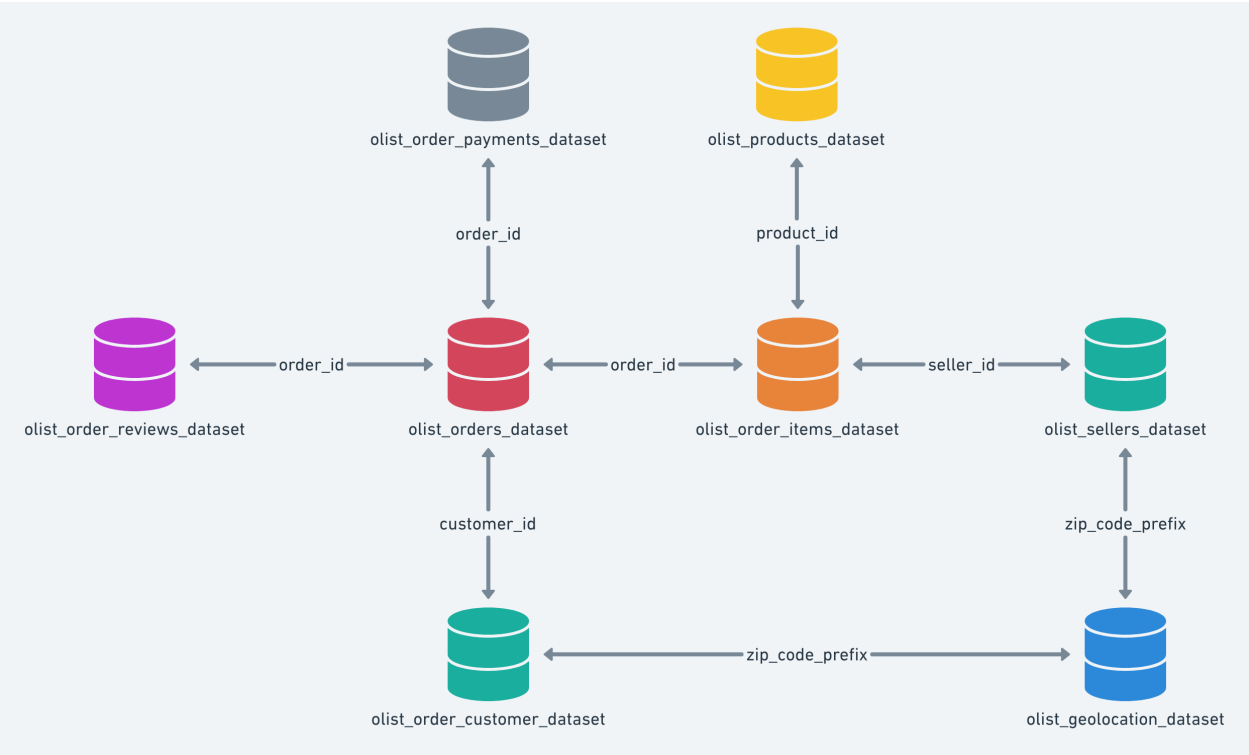


Fig 2 *– Data-map for the Olist dataset. The branches show the common attributes that connect the datasets*

Data Exploration

To understand the skewness and possible correlations we explore the datasets individually. The datasets are imported into JupyterLabs Notebook, and we use Python as our primary programming language. The packages used for data exploration and feature engineering are *Pandas, Matplotlib* and *Seaborn*. We start off with understanding the domain of our business activities. Let us look at the geolocation dataset and use *plotly* package to plot the geolocation.

Fig 3 *– Geolocation plot gives us an idea about our business domain*

As seen from the figure above, our domain of interest in primarily in Brazil, although there are a few skew points around Portugal. This might be due to the trade affiliations between the two Portuguese speaking countries.

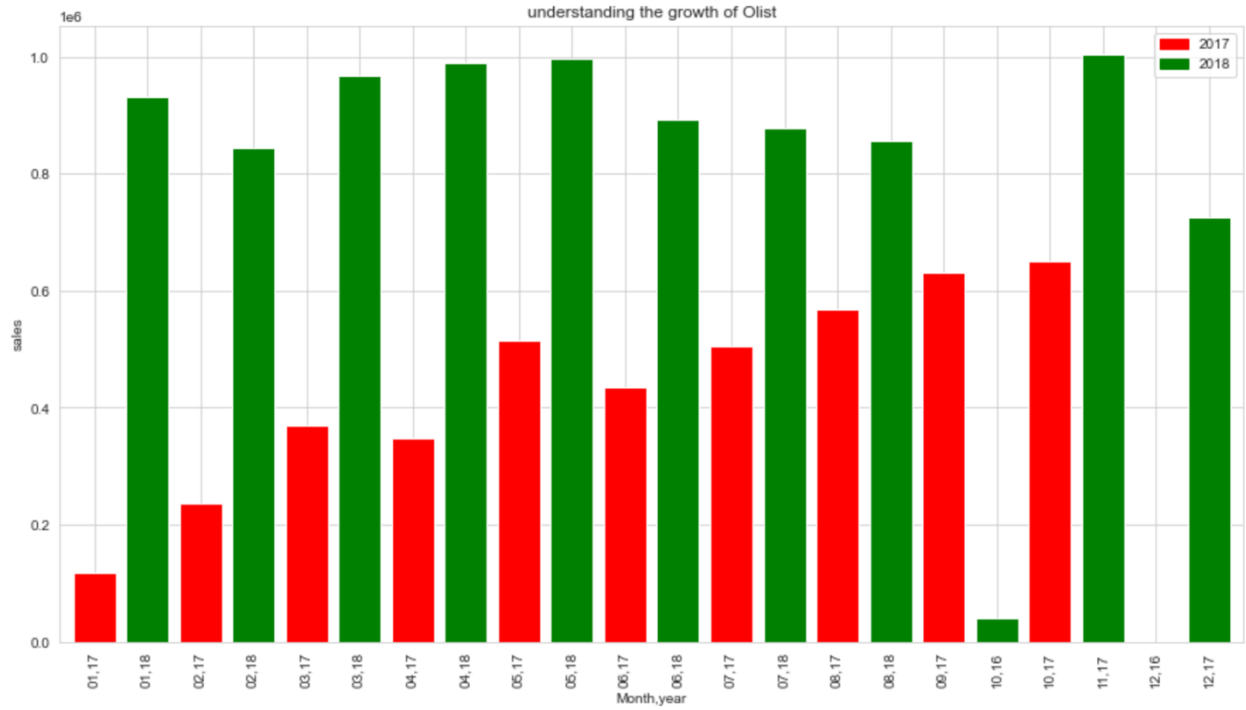
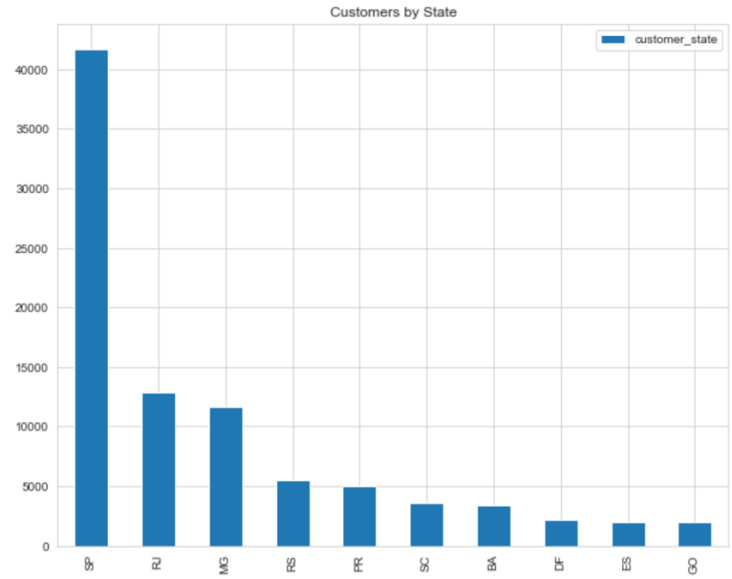


Fig 4 *– Overall Sales comparison of the years 2017 and 2018*




```
sao paulo                15540
rio de janeiro            6882
belo horizonte            2773
brasilia                  2131
curitiba                  1521
campinas                  1444
porto alegre              1379
salvador                  1245
guarulhos                 1189
sao bernardo do campo      938
Name: customer_city, dtype: int64
```

Fig 5 *- Top ten states that represent maximum number of purchasing*

It is seen that 90% of all the business activities are being done in the top ten states. This gives us a good lead in reducing the number of records during feature engineering.

Let us look at the customer reviews as it is predicted variable. We plot a pie chart to get a bird's eye view of the score distribution. This will give us an idea of the skewness in the data, which would influence the kind of classification model that would be deployed.

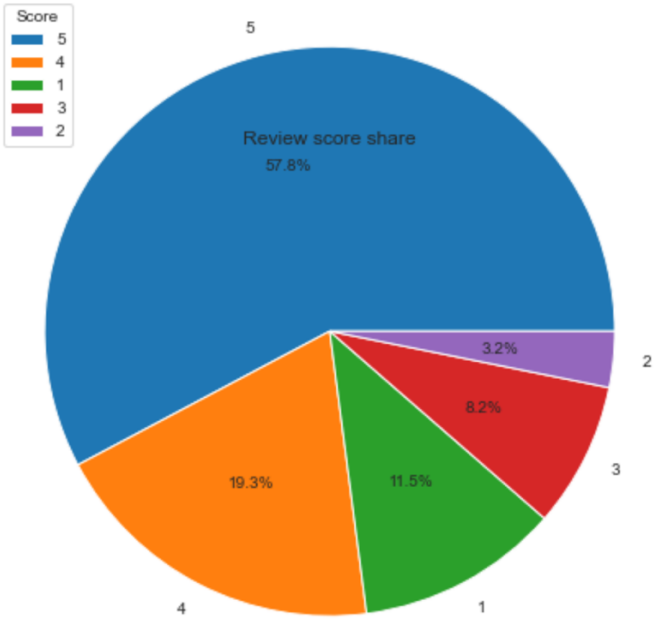


Fig 6 *– Pie chart plot for customer review score*

FEATURE ENGINEERING

In order for the classification model to run efficiently, one needs to input variables that agree with the algorithm. What that means is the variables that are being predicted have to have a clear ability to be classified and the variables that are used to predict the target variable has to have some amount of correlation with each other. This can be achieved by feature engineering.

The process of feature engineering follows the following sequence –

1. **Merging datasets** – Datasets are merged into a single master dataset based on the existing common attributes.
2. **Eliminating null values** – Upon merging, it is possible that there are a lot of missing data (might be genuinely missing or empty spaces owing to merging operation). It was seen that the *review_comment_title* and *review_comment_message* columns had over 85% missing values. These columns cannot be saved by filling the null values. Hence, we drop these variables. This also gives us an insight on customer behavior while filling the feedback forms. Most of the customers just tend to rate their purchase in stars and not type in detailed feedback.
3. **Dropping duplicates** – It is quite possible too have duplicate entries in a data set of this size. Hence, we run a row wise search for duplicate values and drop then from the dataset.
4. **Minimizing the number of variables** – The master dataset has a dimension of 115609 x 48. That's a lot of variables, and it is possible that few of the data might be redundant. One of the variables that can be looked into are the time series data. Time series data are not useful for classification unless they are converted to a numerical value (in terms of interval time) or categorical variables (Early, Standard Delivery, Delayed etc.). The operations below show the methods of achieving new meaningful data from existing data.

```python
start = data['order_purchase_timestamp']
end1 = data['order_estimated_delivery_date']
end2 = data['order_delivered_customer_date']
ship = data['shipping_limit_date']
review_start = data['review_creation_date']
review_end = data['review_answer_timestamp']
```

```python
#subtracting the order_purchase_time to rest time based feature
data['delivery_days'] = end2 -  start
data['estimated_days'] = end1 - start
data['ships_in'] = ship - start
```

```python
#subtracting the review_creation_date from review_answer_timestamp to find review_time
data['review_time'] = review_end - review_start
```

```python
#replacing the time stamp notation and converting type to int
data['delivery_days'] = pd.to_numeric(data['delivery_days'].dt.days, downcast='integer')
data['estimated_days'] = pd.to_numeric(data['estimated_days'].dt.days, downcast='integer')
data['ships_in'] = pd.to_numeric(data['ships_in'].dt.days, downcast='integer')
data['review_time'] = pd.to_numeric(data['review_time'].dt.days, downcast='integer')
```

```python
data['arrival_time'] = (data['estimated_days'] - data['delivery_days']).apply(lambda x: 'Early/OnTime' if x > 0 else 'Late')
```

Fig 6 *– Codes to extract useful numerical and categorical data from timeseries data*

The list of mew variables introduced are –

i) **delivery_days** (Actual period of delivery)

ii) **estimated_delivery_days** (Estimated delivery time as per the seller)

iii) **ships_in** (Time for the order to be shipped at the distributer)

iv) **arrival_time** (The difference between the estimated interval and the actual interval. Converted to a categorical variable)

v) **review_time** (Time taken by the customer to submit their review)

Another useful data is the **seller_popularity,** which was achieved by assigning a score for each unique seller based on the number of orders from the same seller. We also analyzed the product price, the freight price and the total payment done by the customer. We created a new variable called the **freight_ratio**, which is the proportion of the total payment that went into covering the delivery charges. Finally, we use the customer and seller geolocation data to get the distance between the points of purchase and delivery. This resulted in a new numerical variable **distance.** The **review_score** is converted to a categorical variable, with review scores of 4 and above being *Good* and lower than 4 being *Bad*.

After creating new variables, we drop the old ones as they are no longer of interest. The correlation plot shows the relationship.

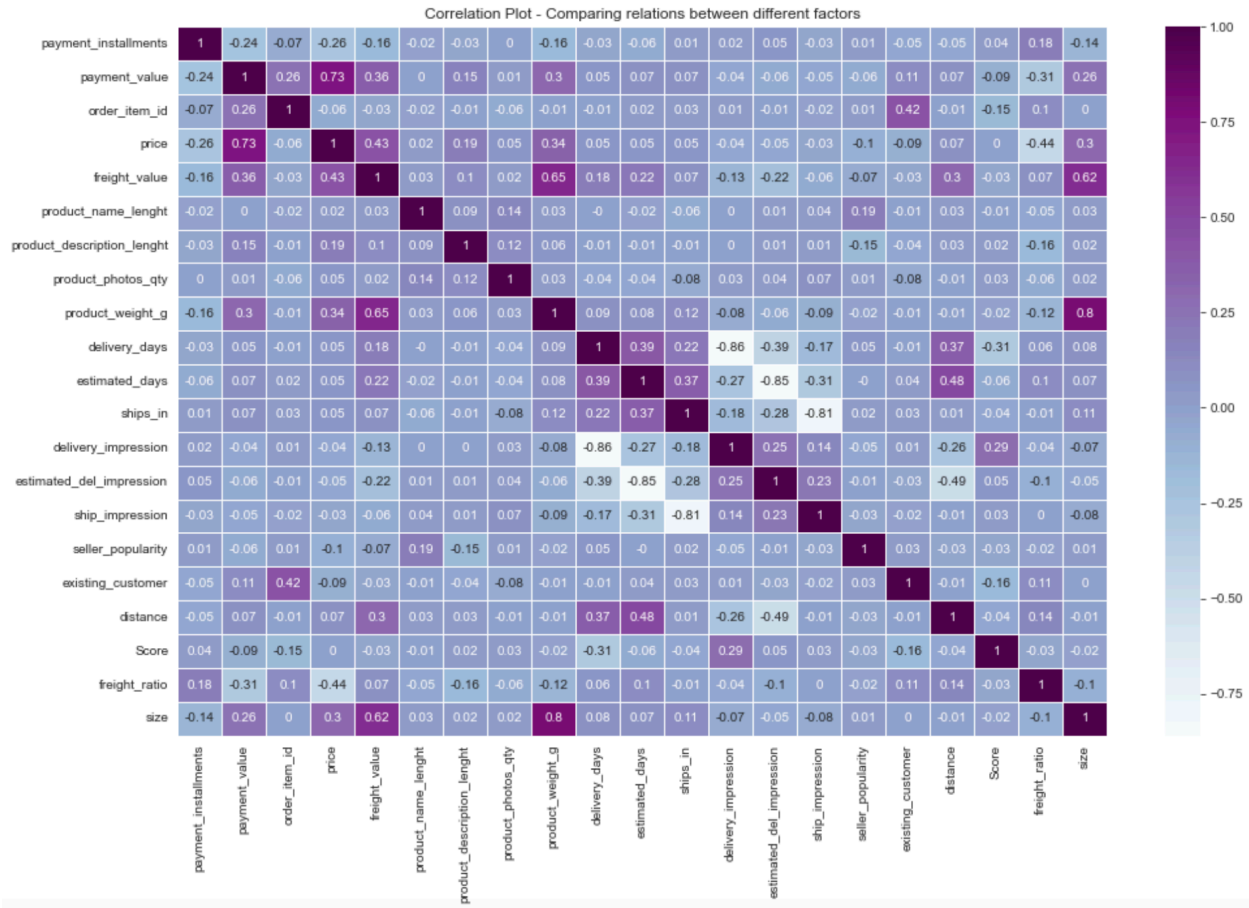
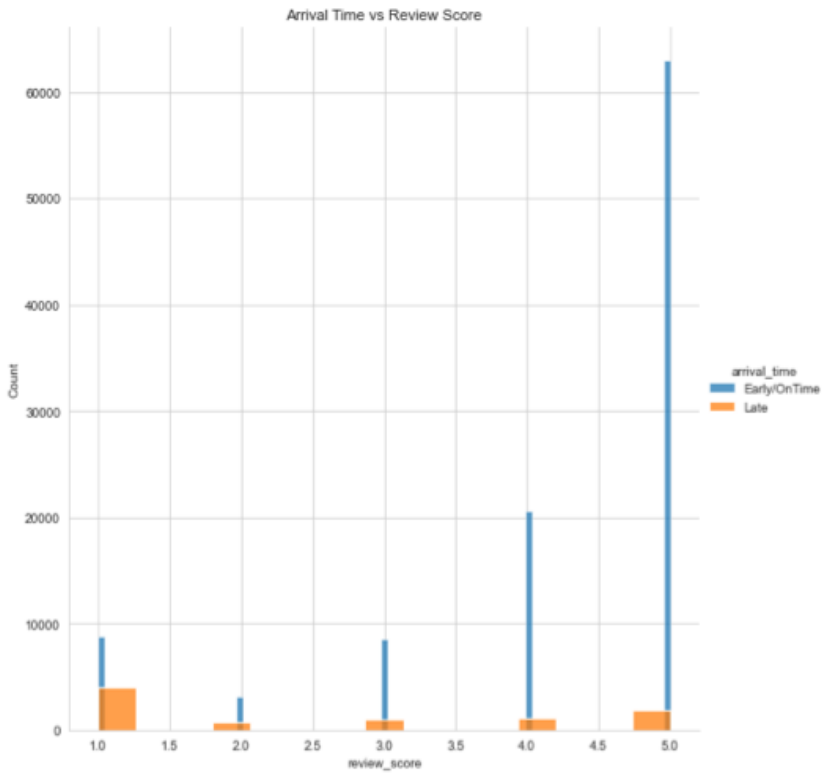


Fig 7 – Correlation plot for the numerical variables

5. **Filtering –** As specified before, we filter out data for the top ten states that represent maximum number of business activities. We also drop extreme outliers for the delivery period.
6. **Standardizing –** For the classification model to be robust, we standardize the numerical data and create dummies for categorical variables.

After feature engineering, we explore the data to understand the nuances in the data and also analyze the usefulness of newly generated variables.



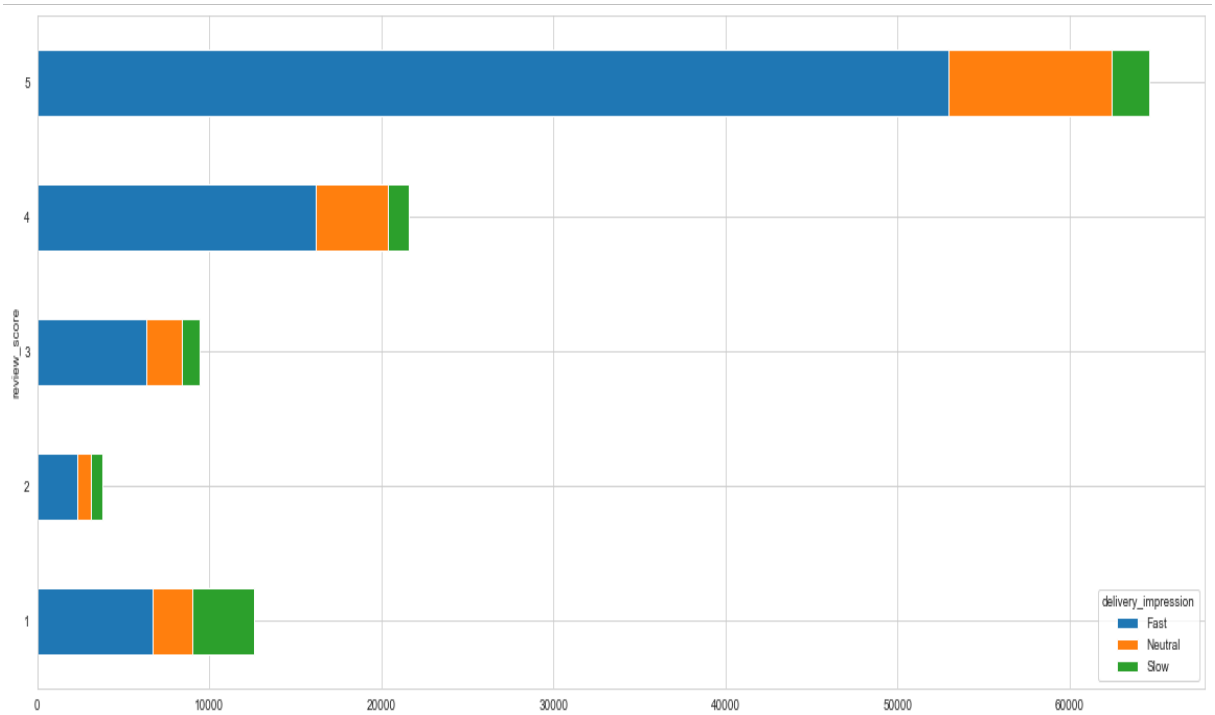


Fig 8 *– The graph shows a clear affinity of high review scores towards fast arrival time*

DATA MINING MODELS AND METHODS

INITIATION

The master data set has been cleaned, filtered, and standardized. We run different classification models too build a predictive model for customer satisfaction. The packages used for analysis are *sci-kit learn* and *dmba*.

Dataset has been evaluated using the following models:

- Naïve-Bayes Classification
- k-Nearest Neighbors Classification
- Random Forest classification
- Decision Tree classification
- Boosted Tree Classification
- Logistic Regression
- Neural Networks

A Principal Component Analysis was run on the fourteen numerical variable which gave us the following cumulative variance –

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13	PC14	PC15
Standard deviation	1.779	1.4311	1.2077	1.1262	1.0931	1.0235	0.9881	0.9186	0.8423	0.8151	0.7191	0.6315	0.5490	0.4651	0.4365
Proportion of variance	0.211	0.1365	0.0972	0.0846	0.0797	0.0698	0.0651	0.0563	0.0473	0.0443	0.0345	0.0266	0.0201	0.0144	0.0127
Cummulative proportion	0.211	0.3475	0.4448	0.5293	0.6090	0.6788	0.7439	0.8001	0.8474	0.8917	0.9262	0.9528	0.9729	0.9873	1.0000

The table shows that it takes all the fourteen variables to explain the variance in the data. Hence, we chose not to drop any variables.

In order to build a model, we need two datasets, the **training set** and the **test set**. We split the master dataset into a 66% training and 33% test set, with stratified sampling. ***GridSearchCV*** from *sklearn* was used to select the best parameters for every classifier.

For categorical variables, correlation was observed after creating dummies and ordinal variables. We chose not to drop any categorical variables unless required.

NAÏVE BAYES AND K-NN CLASSIFIERS

We start off with the simplest and the most primitive form of classification, that is the Naïve Bayes. We consider the default cut-off probability of 0.5 as it is a binary classification. For the K-NN Classifier, we run a search algorithm to detect the best 'k' value that would give maximum accuracy. We chose this k value (k = 21) to solve the model. The model performance is shown below –

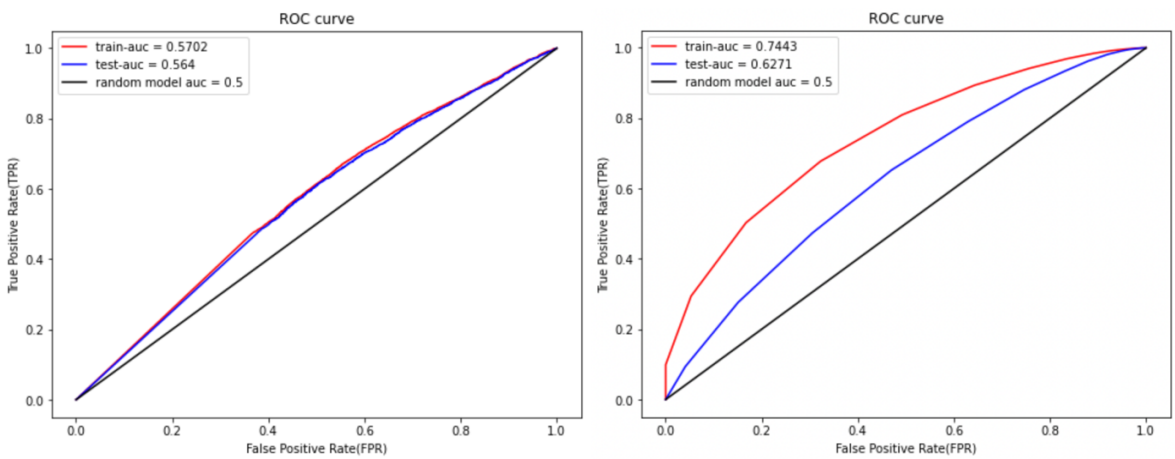


Fig 9 *–ROC Curve for Naïve Bayes (left) and K-NN Classifier (right)*

As expected, the models are not a good fit to represent the trend and predict values of the test dataset. Naive Bayes assumes that if a predictor category is not present in the training data, a new record containing that predictor category has zero probability. If this rare predictor value is crucial, this can be a problem.

K-NN. although slightly better, still fails as a good model owing to the number of neighbors the target variable must have (curse of dimensionality). The model is also time consuming and computationally not efficient for this dataset. The model fails to capture local trends under the approach to fit for global scale.

DECISION TREE CLASSIFIER

Decision tree classifiers have proven to be effective in a wide range of applications. The ability to capture descriptive decision-making information from provided data is their most essential attribute. We start at the root of the tree and split the data on the feature that yields the most information gain (IG) using the decision method (reduction in uncertainty towards the final decision). The model performance is shown below.

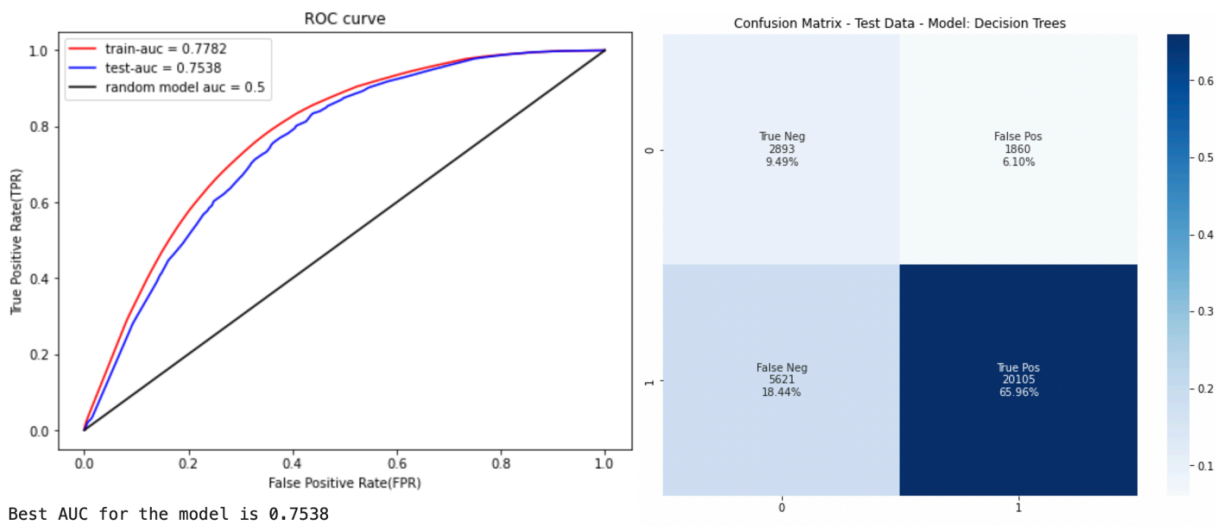


Fig 10 *– The AUC score is much better than the previous models. The confusion matrix on the right shows the effectiveness of the model*

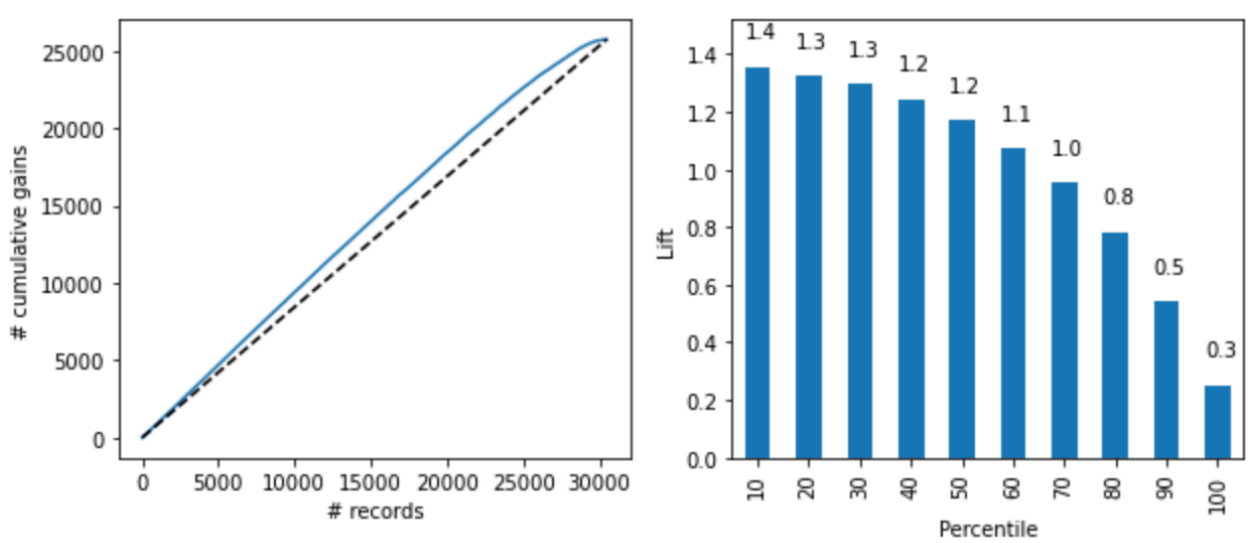
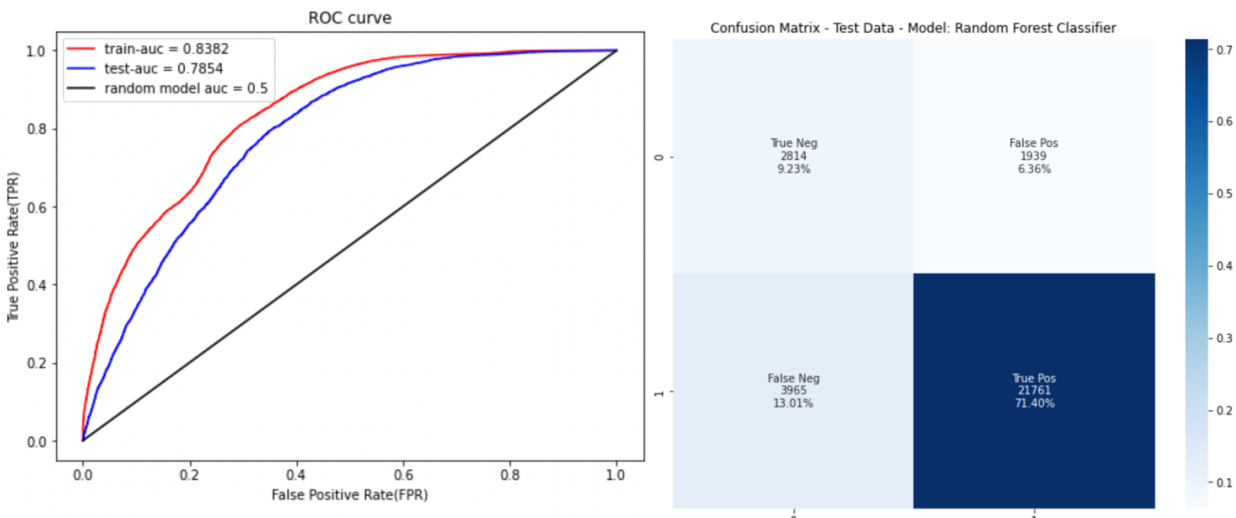


Fig 11 *– The Lift Chart and Gains Chart for the decision tree classification model*

RANDOM FOREST CLASSIFIER

Random forest is a supervised machine learning algorithm that is commonly used to solve classification and regression problems. Because random forests are built from subsets of data and the final output is based on average or majority rating, overfitting is avoided. One of the most essential characteristics of the Random Forest Algorithm is that it can handle data sets with both continuous and categorical variables, as in regression and classification. The model performance is shown below -



Best AUC for the model is 0.7854

Fig 12 *– The ROC curve and Confusion matrix for Random Forest Classifier*

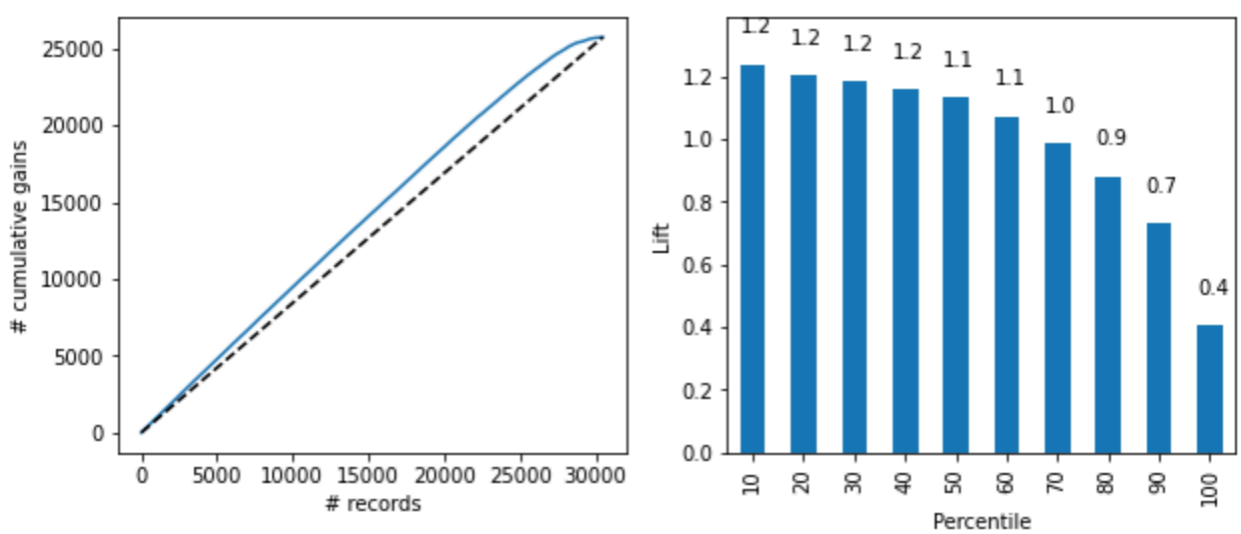


Fig 13 *– Lift Chart and Gains Chart for Random Forest Classifier*

Boosted Tree Classifier

Random forests construct each tree separately, whereas Boosted tree constructs one tree at a time. This additive model works in a stage-by-stage manner, introducing a weak learner to help existing weak learners develop. When we have unbalanced data, such as in real-time risk assessment, boosting works well.

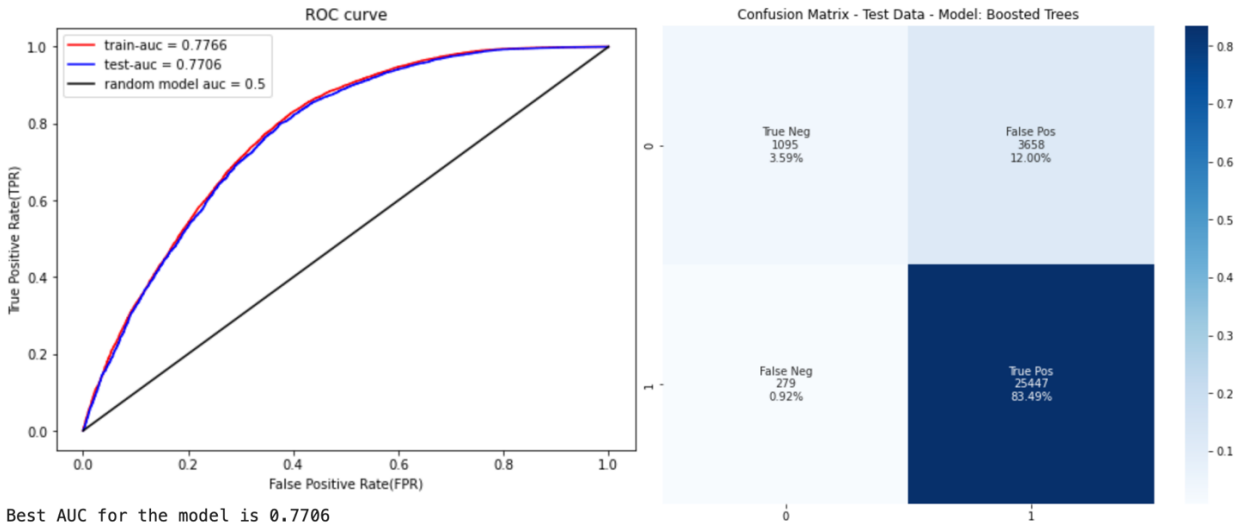


Fig 14 *- The ROC curve and Confusion matrix for Boosted Tree Classifier*

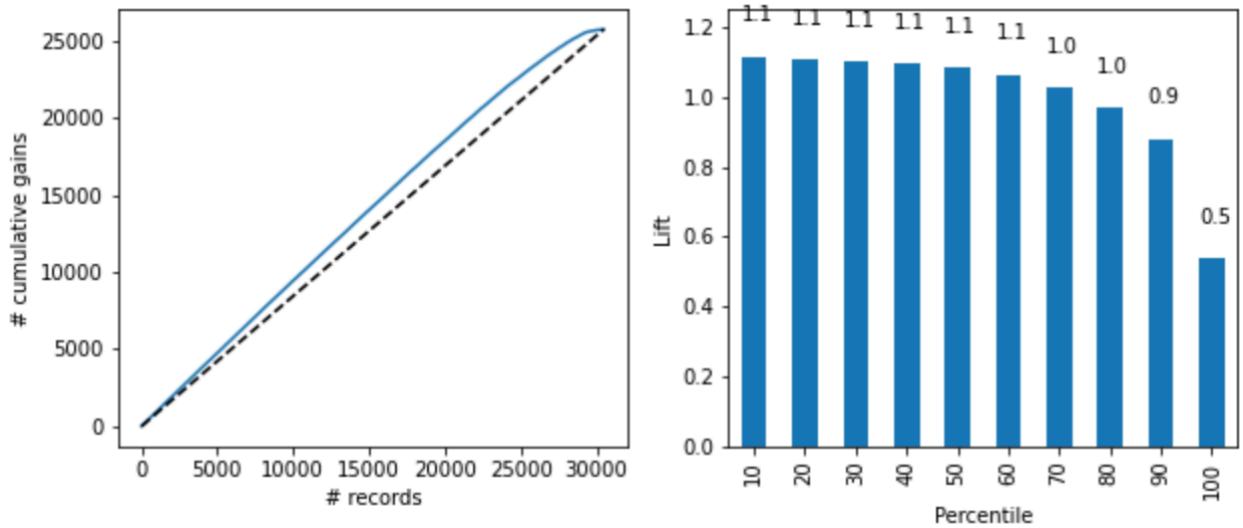


Fig 15 *- Lift Chart and Gains Chart for Boosted Tree Classifier*

Although Boosted trees perform worse compared to random forest in the presence of noise, the model outperforms its competitor for our dataset.

Logistics Regression

The concept underlying logistic regression is simple: rather than utilizing Y as the outcome variable directly, we utilize a function of it called the logit. The logit may be described as a linear function of the predictors, it turns out. It is possible to map the logit back to a probability once it has been anticipated. When a structured model is needed to describe or forecast categorical (in particular, binary) outcomes, logistic regression comes in handy.

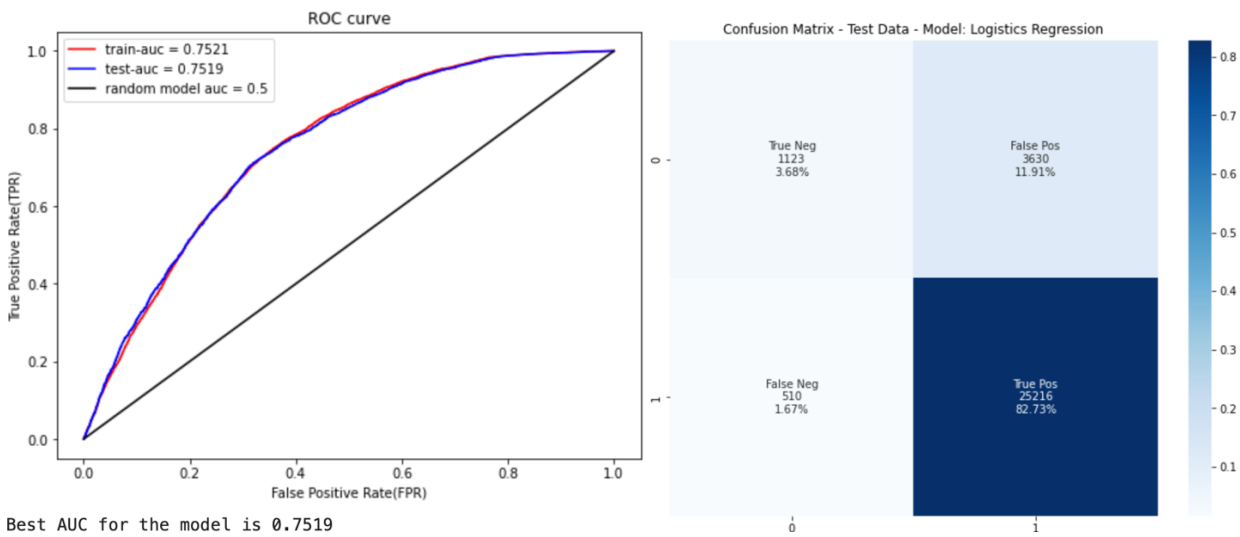


Fig 16 *- The ROC curve and Confusion matrix for Boosted Tree Classifier*

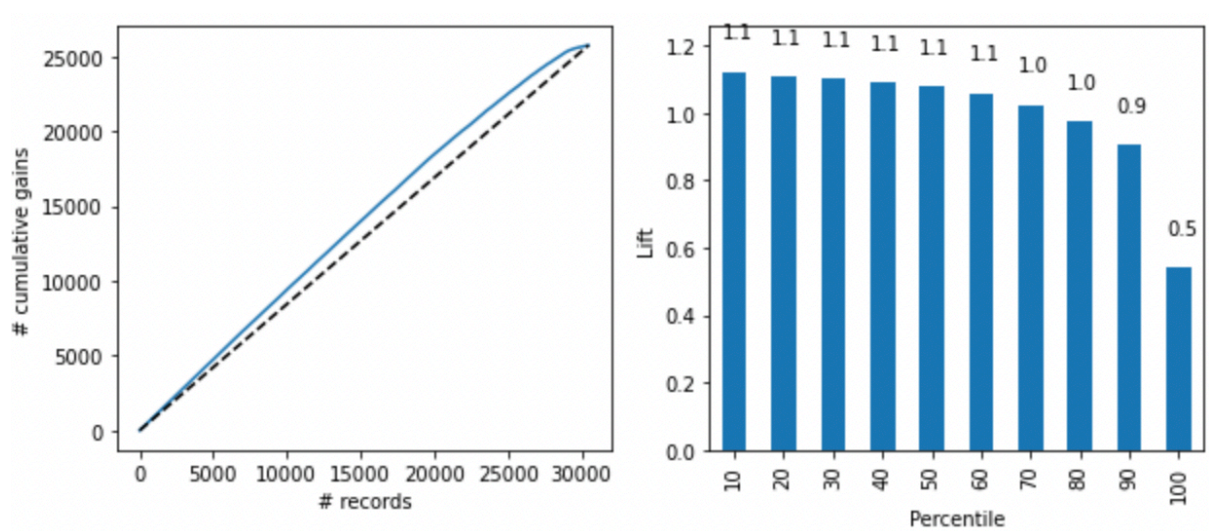


Fig 17 *- Lift Chart and Gains Chart for Boosted Tree Classifier*

Although Logistic Regression can be a robust model, it still does not perform better than boosted trees.

NEURAL NETWORK

The goal of neural networks is to aggregate predictor data in a flexible way that reflects complex interactions between variables and between them and the outcome variable. It differs from logistic regression in that one or more non-linear layers might exist between the input and output levels.

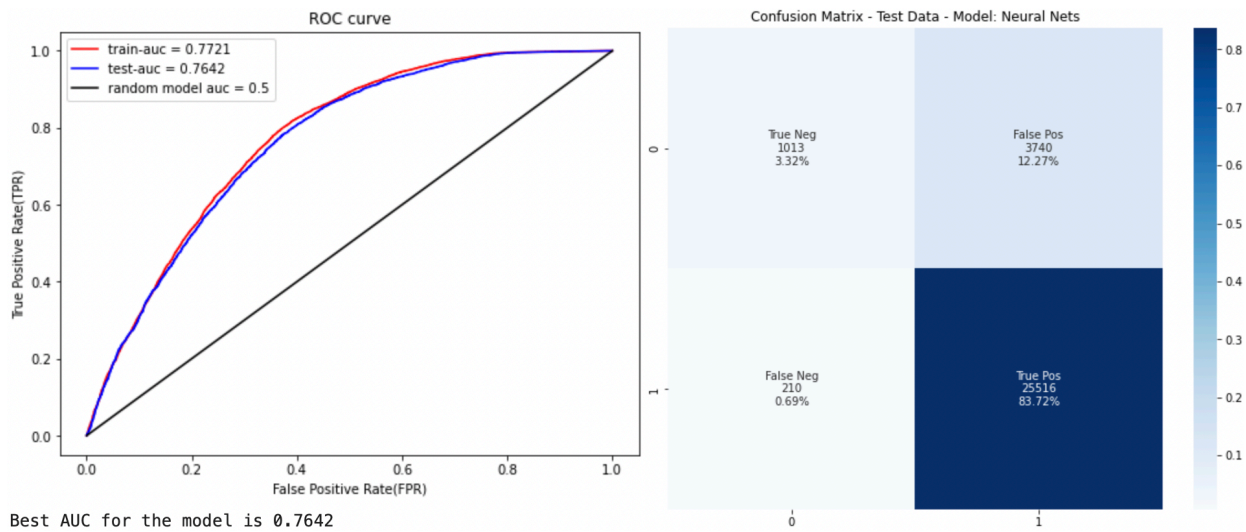


Fig 18 *- The ROC curve and Confusion matrix for Neural Network*

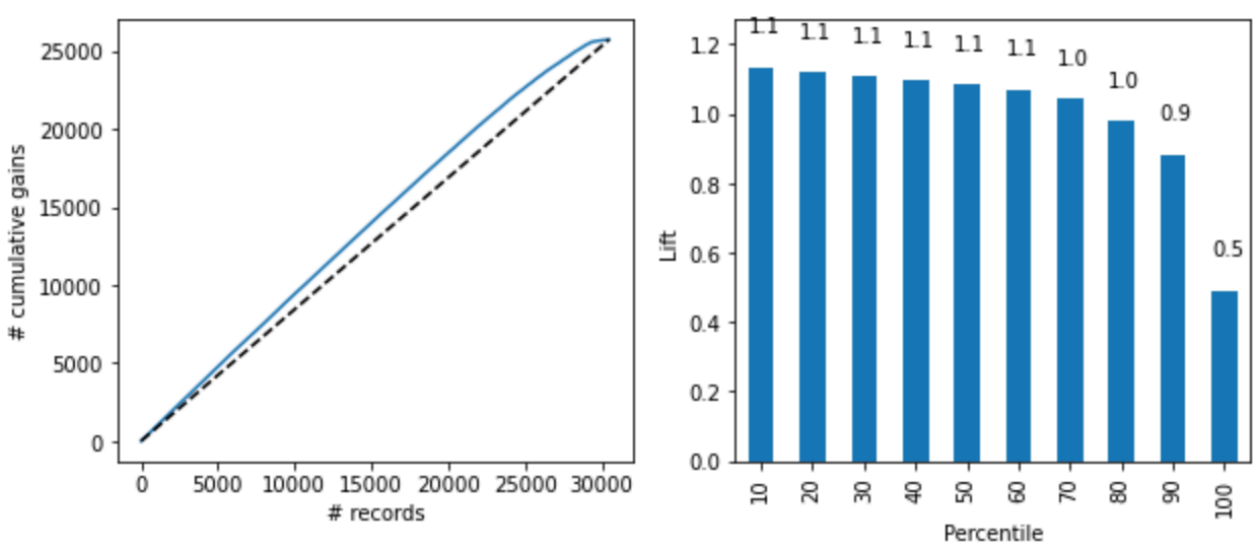


Fig 19 *- Lift Chart and Gains Chart for Neural Network*

Neural Network gives an equal predictive performance as that of Boosted trees and Random Forest Classifiers. In order to select a model we have to analyze other performance indicators.

Performance Evaluation & Results

Analyzing Performance

To pick a machine learning model out of a bunch of classifiers, we need to assess different performance parameters and the kind of data that we are working with. Performance of the models have been evaluated on the following parameters:

1. **ROC curve and AUC value** – The ROC curve displays the TPR against the FPR at different threshold levels, separating the signal from the noise. The Area Under the Curve is used to measures the classifier's ability to distinguish between classes.
2. **RMSE** - The root-mean-square error is a metric used for comparing values predicted by a model or estimate to values observed.
3. **F-1 score** - F1 Score is the weighted average of Precision and Recall. It is frequently more valuable than accuracy, especially if the distribution of classes is unequal.
4. **Misclassification Rate** - The Misclassification Rate is a performance indicator that indicates the percentage of incorrect predictions.
5. **AIC Score** - The Akaike information criterion is a predictor of prediction error and thus relative model quality for a given set of data.
6. **Matthews Correlation Coefficient** - MCC is a quality indicator for binary classifications.

Model	AUC Score	RMSE	F1 Score	Misclassification Rate	AIC Score	Matthews Correlation Coefficient
Boosted Tree	0.7706	0.3573	0.8708	0.1292	24228.7574	0.3839
Random Forest	0.7854	0.4184	0.8063	0.1937	36579.1857	0.3820
Neural Nets	0.7584	0.3589	0.8704	0.1296	24329.233	0.3789
Lasso Regression	-	0.3305	0.8651	-	17333.4869	0.3564
Decision Tree	0.7538	0.4847	0.7546	0.2454	43794.674	0.3155
k-NN	0.6271	0.392	0.8456	0.1544	29666.7985	0.1150
Naïve Bayes	0.5640	0.6190	0.6168	0.3832	57370.810	0.0799

Fig 20 *- Comparing all the parameters against each other, Booster Tree is the best fit.*

RESULTS

By comparing the performance parameters for all the models, it was a close competition between the Boosted Tree, the Random Forest Classifier, and the Neural Network Classification model. Since each model have their own advantages, boosted trees perform better with unbalanced data, which is what represents the Olist sales data. Since boosted tree also had a slightly better MCC score and AIC Score than the other models, we consider **Boosted Tree Classifier** to be the best fit model for customer satisfaction.

The selected model can be implemented in understanding -

1. Customer purchasing intent.
2. Product with highest sales potential.
3. Popular sellers based on quality of service.
4. Target Logistics Networks that would result in minimum freight prices and maximum satisfaction.