CUSTOMER REVIEW PREDICTION

Table of Contents

Framing the Problem

1. Exploring the dataset

Upon careful assessment of the dataset and considering all the features present, I have decided to use machine learning models to predict the review of customers by utilizing relevant information to see which key features affect it and how.

The review scores vary from 1-5 but I have opted to convert it into a binary classification where a review score of 1, 2, or 3 will be considered to be poor review thus mapping it to class 0 and review scores of 4 and 5 will be considered to be good review thus mapping it to class 1. I will be implementing Logistic Regression, Random Forest, KNN, XGBoost and Multinomial Naïve Bayes.

2. Users of my solution

The users of my solution would be the stakeholders of the company who could use this information to gauge the performance of the products and the satisfaction of the customers with the products guiding them to make more informed decisions.

3. Performance Measurement of Models

As this is a classification problem, I shall use Accuracy, Precision, Recall and F1-Score as the key performance metrics to assess the performance of the models.

4. Availability of Human Expertise

Human expertise is definitely available in this domain such as business analysts and customer experience managers. However, it would be very resource intensive to do it manually by collecting surveys and feedback and manually analyzing over a hundred thousand data.

5. Assumptions and Limitations

The assumptions for this project are as follows:

1. The data is clean and accurate.
2. Considerations pertaining to temporal consistency regarding customer reviews.

Data Preparation

1. Joining Tables

I started off by joining the tables orders, customers, order items, order payments, and order reviews. This helped me create a more comprehensive single dataset that is easier to work on rather than separate smaller dataset.

2. Missing Values

I checked for missing values in the dataset using the isnull() function from pandas and then used dropna() to drop all the missing values.

3. Dropping columns and values

I dropped the unnecessary columns such as the id columns, pre-aggregated columns used to calculate aggregated features, orders that had a status other than delivered as those values did not have any reviews.

4. Outliers

I have removed outliers from the price column using the conditional statement of all values greater than 300 based on the visualization of price by reviews. I also removed outliers from fulfilment time less than 55 and payment installments greater than 8 as they were outliers as per the boxplots.

5. Aggregating Features

I have aggregated some features into one to better represent the data. I calculated the fulfillment time by aggregating order purchase timestamp and order delivery date. I also calculated the estimated delivery time by aggregating the estimated delivery date and the order purchase timestamp. The aggregated data serves as a better and comparable feature compared to the individual features alone.

6. Data Imbalance

I handled the class imbalance based on the bar plot of the count of the classes. I downsampled the 1 class which had almost 4 times as many samples as the 0 class.

7. Type Conversion

I converted the categorical string values to int using the label encoder function from scikit learn.

8. Discretizing continuous features & One-hot Encoding

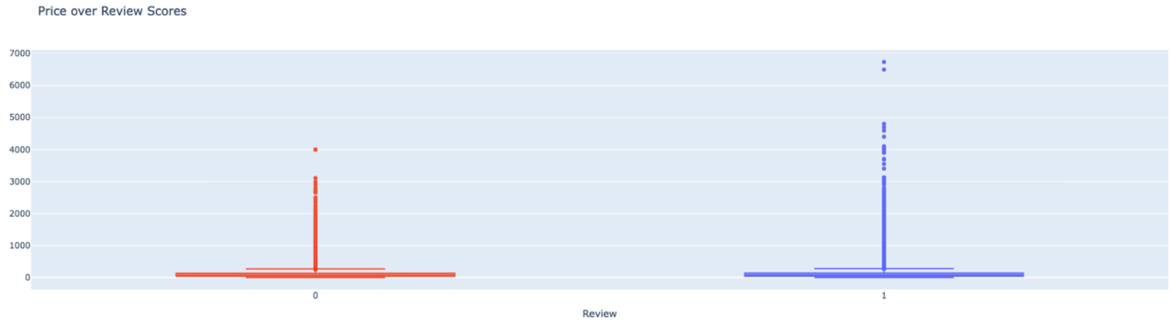
These steps were not necessary for my preprocessing to fit it to my models.

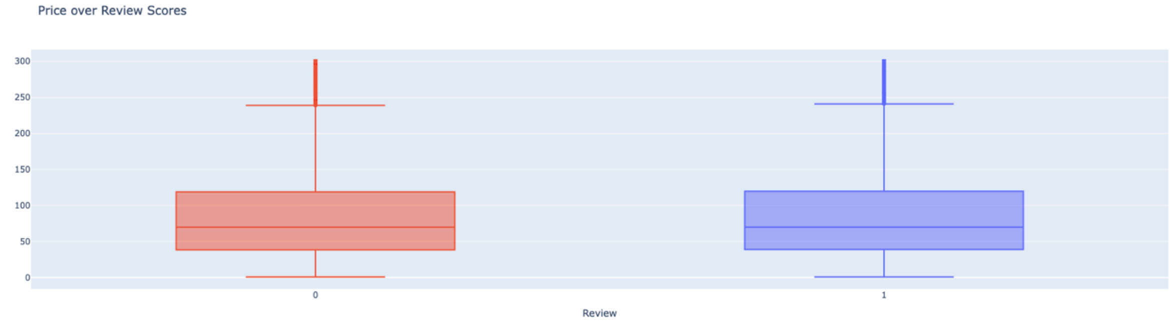
Exploratory Analysis

1. Numeric Variables

I made 3 visualizations using box plot – payment value by reviews, fulfillment time by reviews, and payment installment by reviews. Based on the visualizations, I carried out outlier removal.

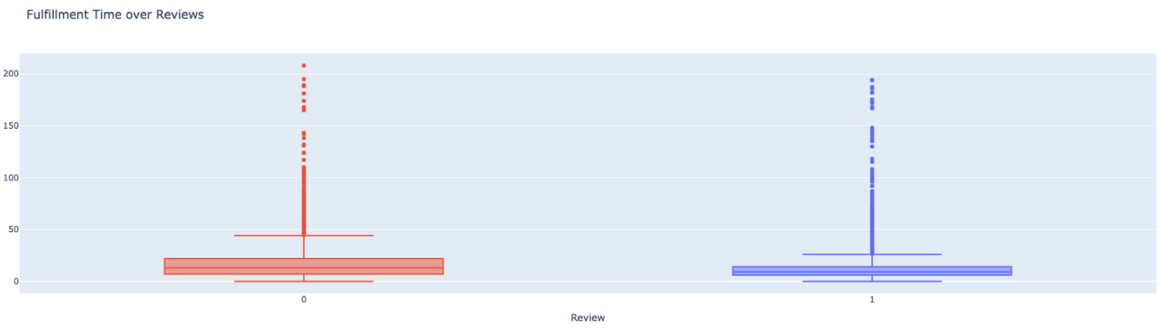
The first visualization is of price over review scores as is, and the second visualization is after removing outliers.

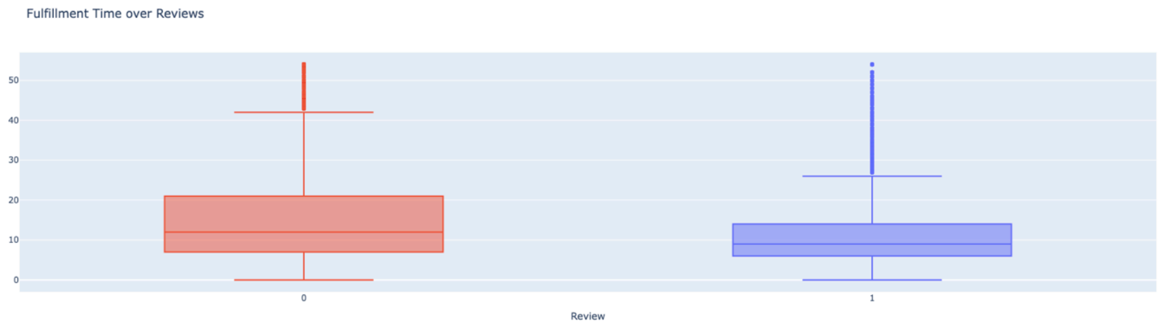




This shows that there isn't any particular effect of price on reviews.

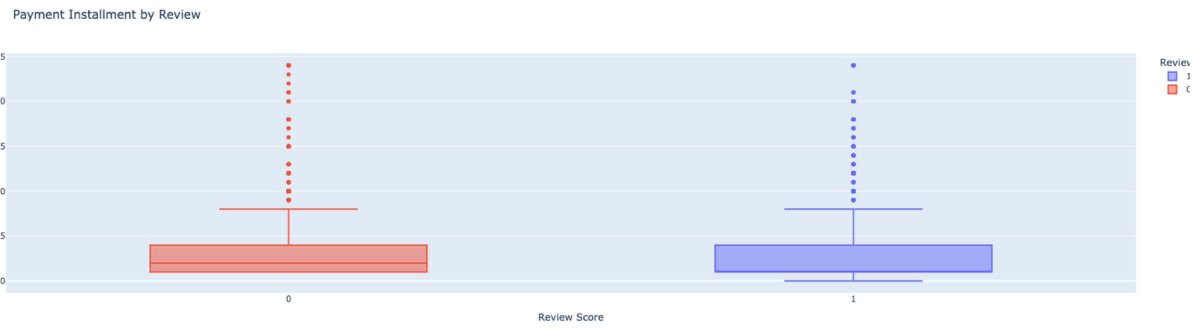
Next, I visualized fulfillment time over reviews followed by the same visualizations with outliers removed.

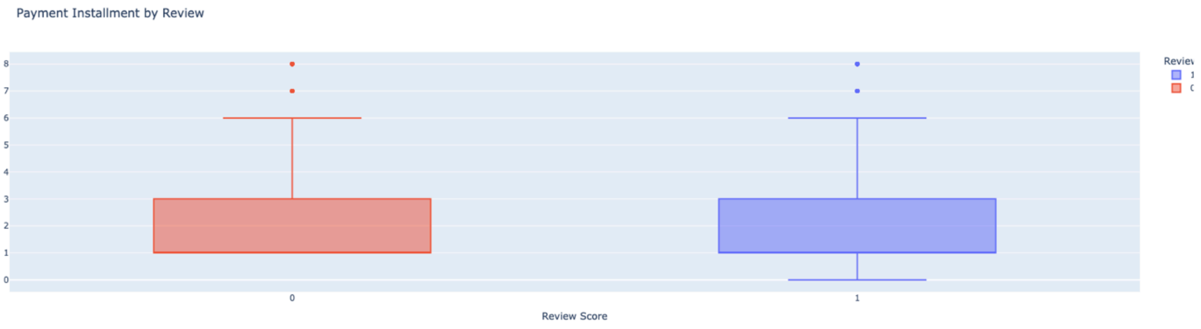




There is an inverse relationship with fulfillment time and reviews with higher time warranting lower review scores.

I visualized payment installment by reviews followed by the same visualization with the outliers removed.

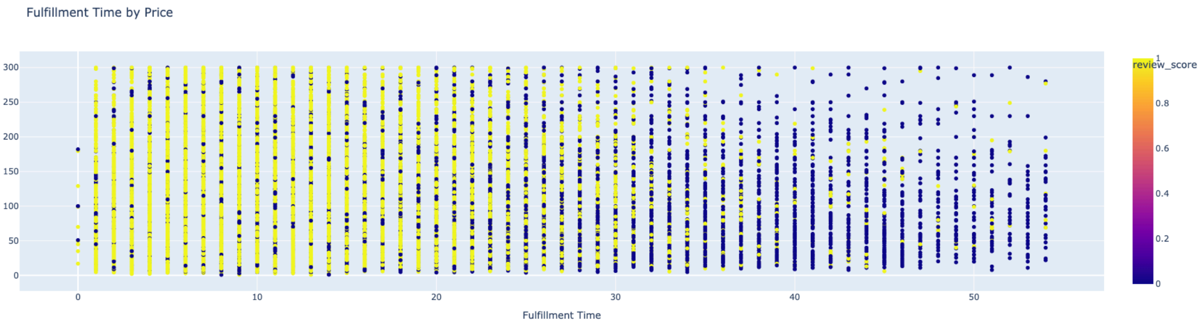




This goes to show that there is not much correlation between these variables.

2. Relationship between Numeric Variables

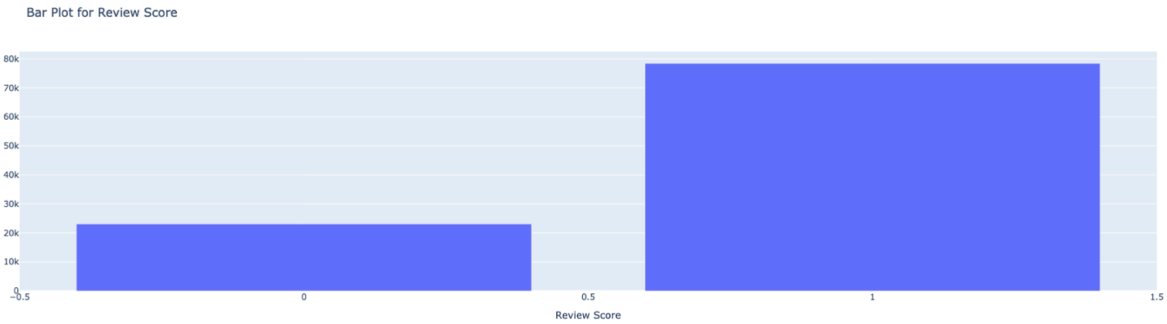
I visualized Fulfillment time by price with the points color coded by reviews.



It can be seen that there is no general trend of fulfillment time and price but it seems clear that increase in fulfillment times decreased the review scores.

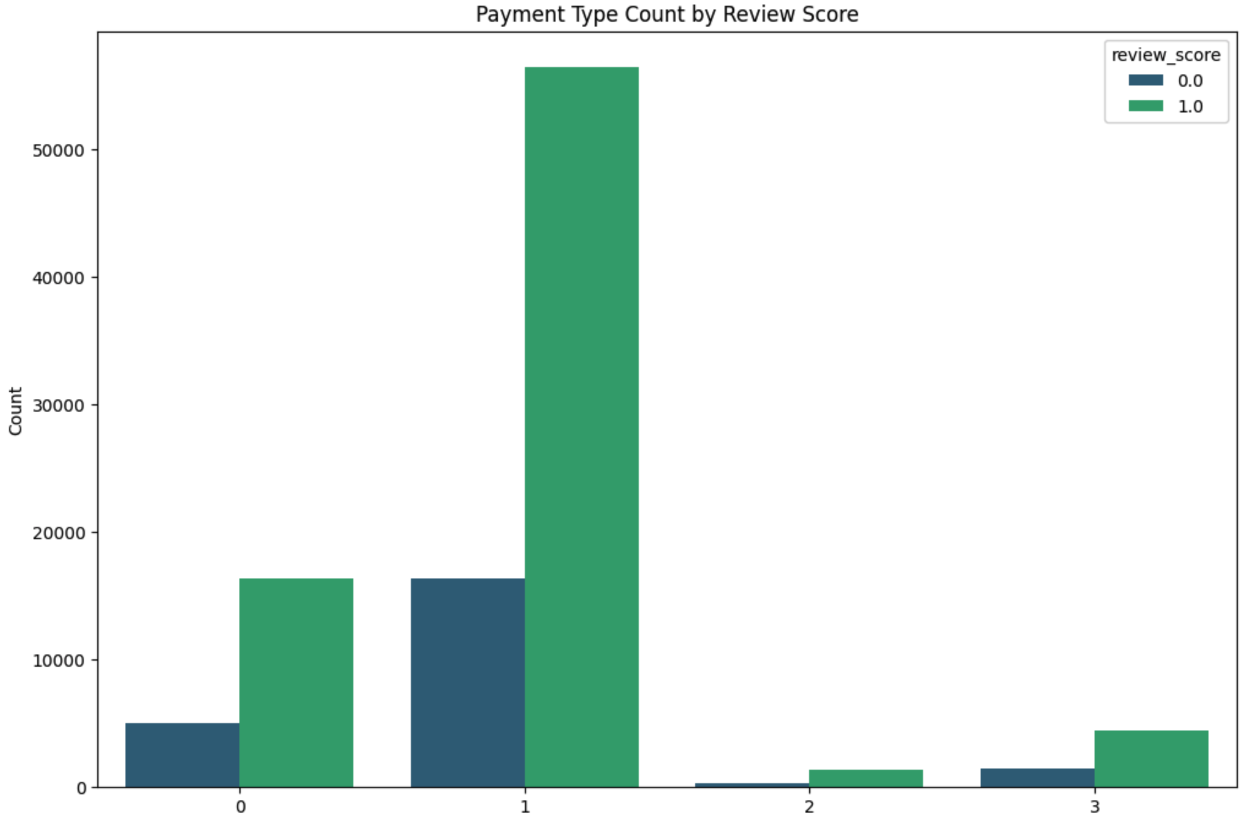
3. Categorical Features

I visualized the number of reviews to get a sense of the distribution of the 2 target classes.

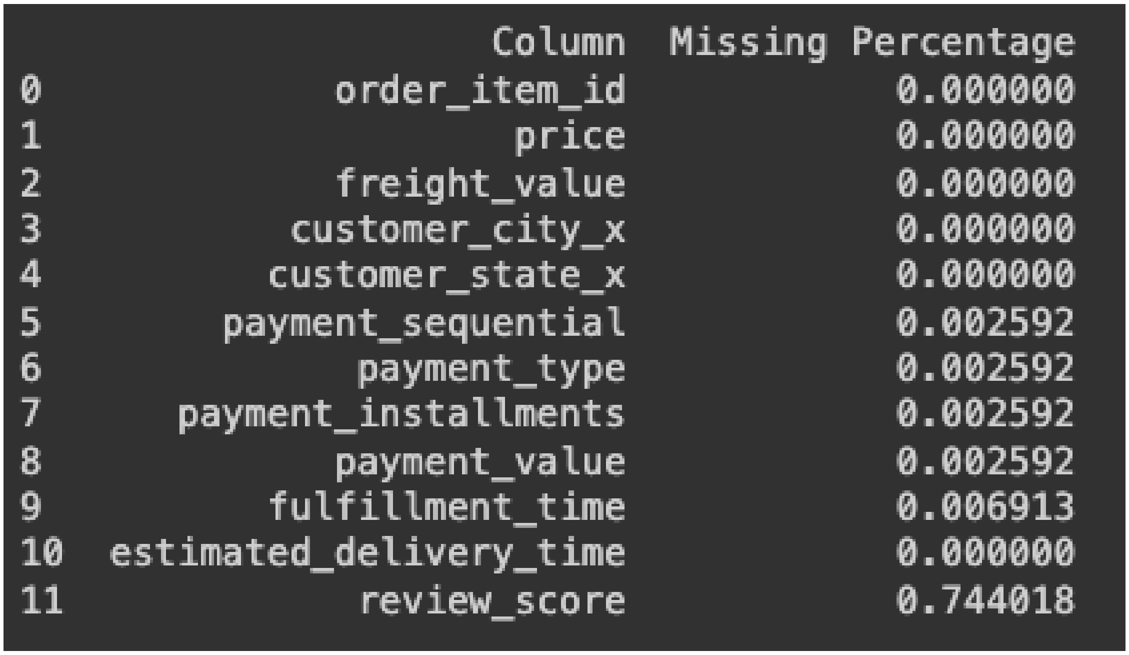


There seems to be a significant class imbalance with class 1 having almost 4 times more samples than class 0.

I also visualized the count of payment type by review from a stacked bar plot.



4. Percentage of Missing Values



```
                         Column  Missing Percentage
0                 order_item_id            0.000000
1                         price            0.000000
2                 freight_value            0.000000
3               customer_city_x            0.000000
4              customer_state_x            0.000000
5            payment_sequential            0.002592
6                  payment_type            0.002592
7          payment_installments            0.002592
8                 payment_value            0.002592
9              fulfillment_time            0.006913
10      estimated_delivery_time            0.000000
11                 review_score            0.744018
```

Review scores had the highest percentage of missing values but overall, the percentage was so low that I dropped the rows with the missing values.

Shortlisting Promising Models

1. Models Used

I used the models Logistic Regression, K-Nearest Neighbor, Naïve Bayes, XGBoost and Random Forest using the default parameters.

2. Test Train Split

I utilized a test train split of 0.2 or 20% meaning 20% of the data would be used for testing and 80% of the data would be used for training. I used a random state of 42 to get the same split every time I ran the models.

3. Comparing Performance

I utilized 10-fold cross validation for each of the models and created a function that loops through the list of models. The best performing model was random forest with an accuracy of 68.85%. The average accuracy and the average standard deviation of the models are listed below:

Models	Accuracy	Standard Deviation
Logistic Regression	63.13%	0.64%
KNN	60.65%	0.85%
Naïve Bayes	54.35%	0.77%
XGBoost	66.95%	0.76%
Random Forest	68.85%	0.52%

4. Feature Selection

I used top 5 coefficients and top 5 feature importances to find the top 5 most significant features and I fit the models again using these 5 engineered features. The 5 most significant features are: fulfillment_time, payment_value, price, freight_value and customer_city.

Rerunning with these 5 variables yielded the following results:

Models	Accuracy	Standard Deviation
Logistic Regression	62.71%	1.00%
KNN	60.83%	0.95%
Naïve Bayes	54.31%	0.78%
XGBoost	66.58%	0.98%
Random Forest	67.26%	0.84%

It goes to show that feature selection barely affected the previous performance so it can be said that it had nominal effect.

I used 5 models in total and the top 3 models were Random Forest, XGBoost, and Logistic Regression.

Hyperparameter Tuning

I established a set of parameters as a python dictionary including important hyperparameters with different values and I used GridSearchCV to find the best ones with the best accuracy.

The best hyperparameters for each of the models are given as follows:

Models	Best Hyperparameters
Logistic Regression	C: 10
KNN	N_neighbors: 5
Naïve Bayes	Alpha: 0.1
XGBoost	Lr: 0.1, max_depth:7, n_estimators: 200
Random Forest	N_estimators: 100

Ensemble Models

I used the voting classifier as the ensemble model. I made ensembles of the top 3 best performing models logistic regression, XGBoost and Random Forest using the best hyperparameters from before.

I used both soft and hard voting classifiers and I ran both of them twice, once with the full features and once with the selected 5 features.

The performance obtained are as follows:

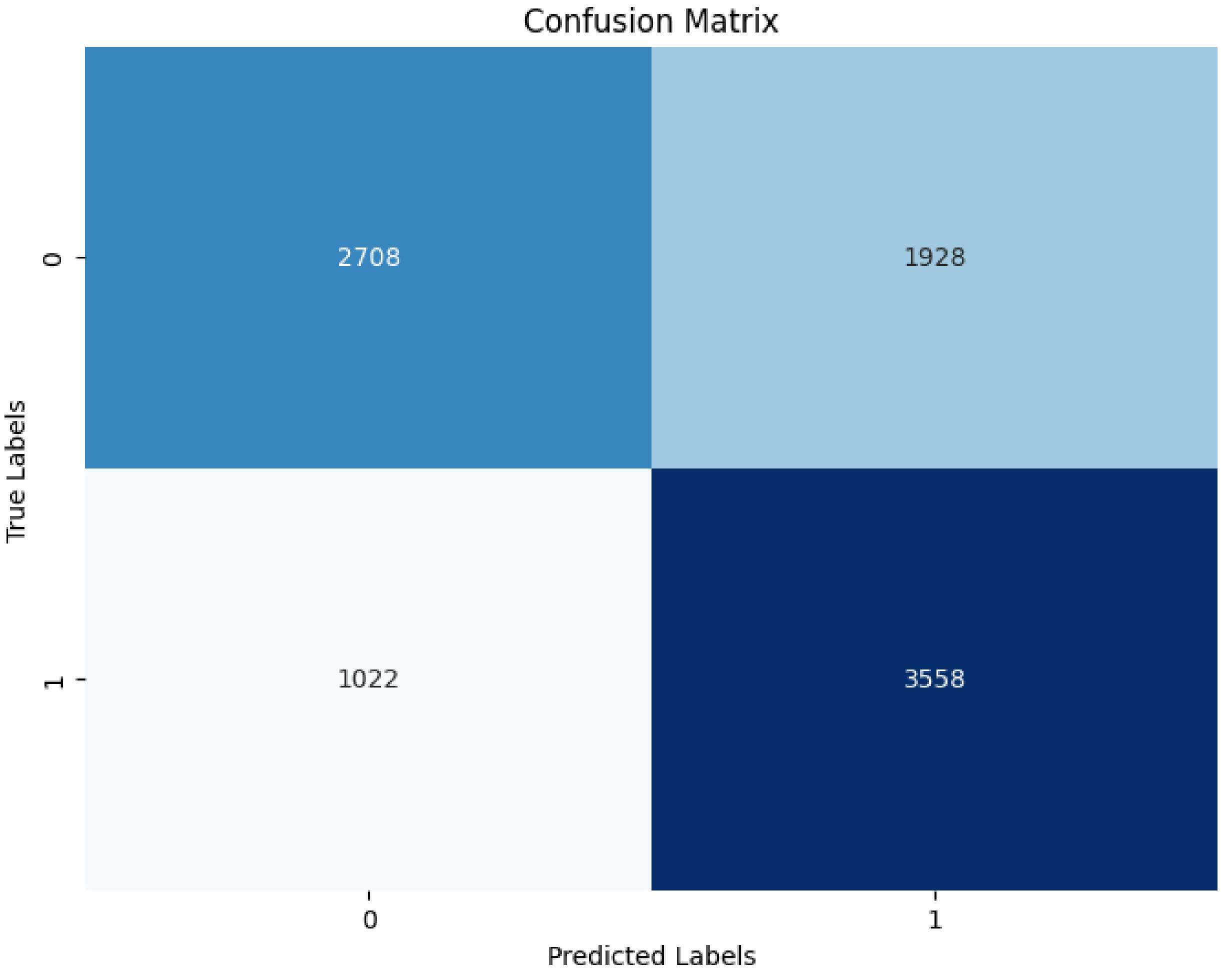
Models	Accuracy	Standard Deviation
Hard Voting without feature selection	68.47%	0.51%
Hard Voting with feature selection	67.79%	0.86%
Soft Voting without feature selection	69.30%	0.57%
Soft Voting with feature selection	68.67%	0.69%

Predictions

With all the values obtained, the best model was the soft voting ensemble without feature selection. I created predictions for X_test and saved it as y_pred and then compared y_pred with y_test.

1. Confusion Matrix

The confusion matrix is as follows:



2. Calculation of the Evaluation Metrics

The calculation of the metrics accuracy, precision, recall and f1-score are done by the equations below:

$$Accuracy\ =\ \frac{t_p+t_n}{t_p+t_n+f_p+f_n}$$

Here, t_p is the number of true positives, f_p the number of false positives, t_n is the number of true negatives and f_n is the number of false negatives. Accuracy is the percentage of correctly classified samples in the dataset.

$$Precision\ =\ \frac{t_p}{t_p+f_p}$$

The precision is a classifier’s ability to label positive samples as positive.

$$Recall = \frac{t_p}{t_p + f_n}$$

The recall is the classifier’s ability to accurately predict the positive samples.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

F1-Score is the harmonic mean of precision and recall and it shows the model’s capacity to ascertain a balance between precision and recall.

3. Obtained Results

The obtained performance metrics as per the previous equations are as follows:

Class	**Precision**	**Recall**	**F1-Score**	**Accuracy**
0	0.73	0.58	0.65	0.68
1	0.65	0.78	0.71	