STAT 206: Final Report

Data R Us
Rohan Benhal, Tanmayee Parbat, Jeewan Singh

Predicting Hotel Bookings using Machine Learning: An Analysis of the Expedia Dataset

Introduction:

In recent years, the travel industry has undergone a significant transformation with the advent of online booking platforms such as Expedia. These platforms provide travelers with easy access to a vast array of hotels, flights, and vacation rentals, while also enabling hoteliers and property owners to market their properties to a global audience. However, with so much data being generated by these platforms, it can be challenging to make sense of it all and identify the factors that influence travelers' booking decisions.

In this report, we present the results of our analysis of the Expedia dataset using machine learning techniques. The goal of our project was to identify patterns and trends in the data that could be used to predict and understand travelers' booking behavior. Specifically, we aimed to answer the following research questions:

The most popular destinations and travel patterns?
The factors that influence customer booking behavior?
The demographic characteristics of Expedia customers?
What are the most important factors that influence travelers' hotel booking decisions?
How can we use machine learning to predict travelers' booking behavior?
What are the implications of these findings for the travel industry and related fields?

To address these questions, we analyzed a large dataset of hotel bookings and related information from Expedia. We used a variety of machine learning techniques, including regression and classification models, to identify patterns and relationships in the data. Our findings provide insights into the factors that influence travelers' booking decisions and have important implications for hoteliers, travel marketers, and other stakeholders in the travel industry.

Data Exploration:

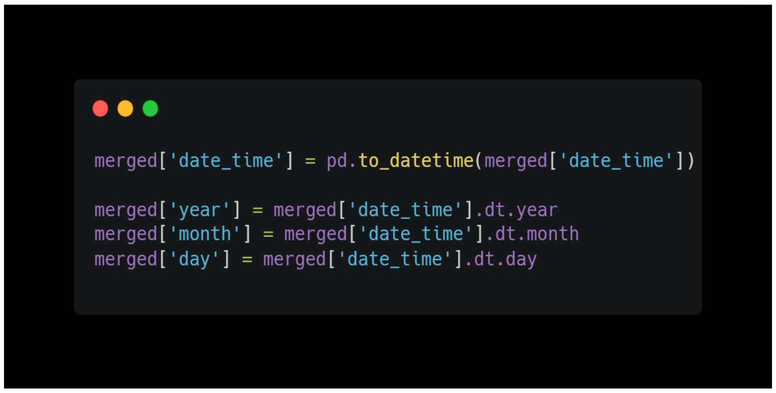
In this project, we utilized a dataset consisting of user interactions with the Expedia website, as well as a destination file containing additional information about the destinations being searched for. Each row in the dataset represents a user's session and their interactions with the site, including clicking on or booking a hotel. The dataset includes information such as the user's location, the dates of their hotel search, the number of adults and children they are searching for, and other relevant information.

The destination file contains information about the destinations being searched for, including the destination ID, name, type, latitude, longitude, and popularity scores for various travel-related facets of each destination.

We merged the data and destination files to gain a more comprehensive understanding of the user behavior and their preferences when searching for hotels in different destinations. The resulting dataset provided us with valuable insights into user behavior, which we utilized to train a machine learning model to predict which hotels a user is most likely to book based on their search parameters

Dropping null values is a common data cleaning step that helps to ensure the quality of the dataset. In our project, we also dropped null values to prepare the dataset for analysis. By using the dropna() function with the parameter inplace=True, we are able to remove all rows that contain null values from the dataset without creating a new data frame. This ensures that we have a clean and complete dataset to work with in our analysis.

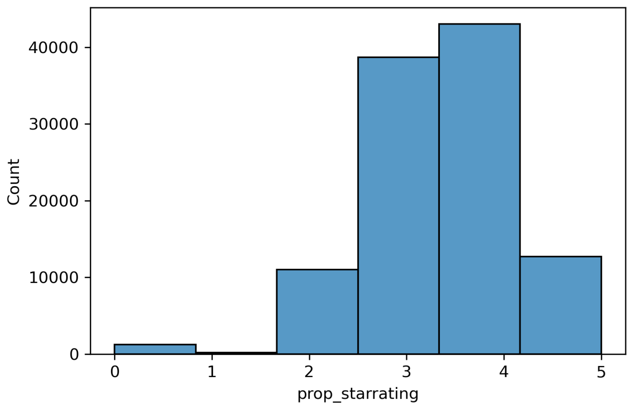
In order to perform some analysis on the dataset, we first needed to convert the 'date_time' column to a Pandas datetime object. This was done using the 'pd.to_datetime()' function. Once this was done, we were able to extract the year, month, and day as numerical features using the 'dt.year', 'dt.month', and 'dt.day' functions. These features were then added as new columns to the merged dataset, allowing us to perform further analysis on the dataset based on the temporal aspects of the user interactions with the Expedia site.

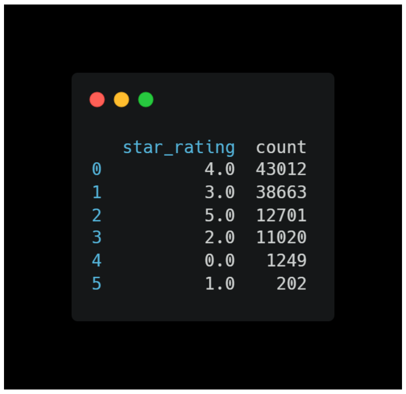


```
merged['date_time'] = pd.to_datetime(merged['date_time'])

merged['year'] = merged['date_time'].dt.year
merged['month'] = merged['date_time'].dt.month
merged['day'] = merged['date_time'].dt.day
```

Data Visualization:

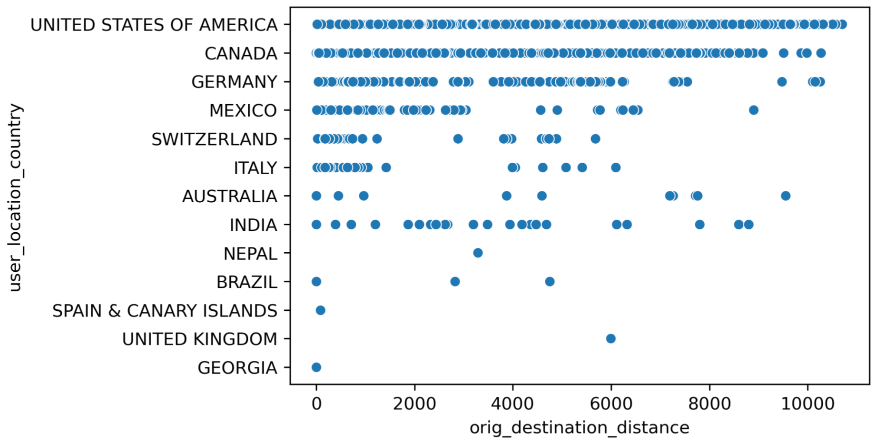
We created a histogram using Seaborn to visualize the distribution of the property star ratings in the dataset. The histogram shows that the majority of properties have a star rating of 4 or 3, with a smaller number having a rating of 5 or 2, and very few having a rating of 0 or 1. To further investigate this, we also created a table that shows the count of properties for each star rating, confirming the findings from the histogram.





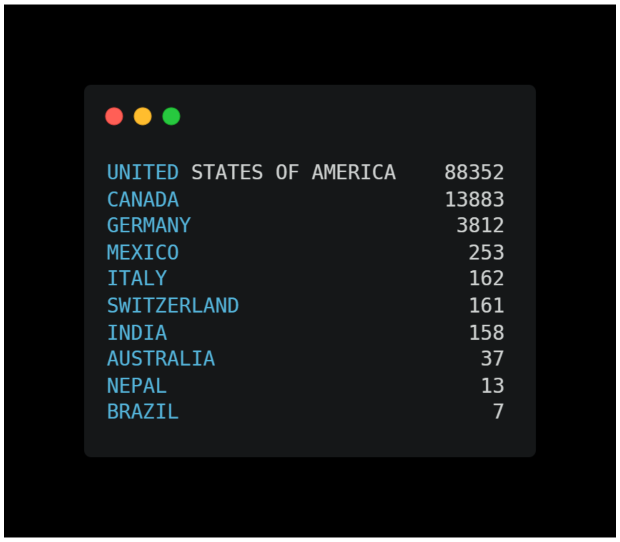
```
   star_rating  count
0          4.0  43012
1          3.0  38663
2          5.0  12701
3          2.0  11020
4          0.0   1249
5          1.0    202
```

The table above shows the count of hotel listings based on their star rating in the dataset. The star ratings range from 0 to 5, with 0 being unrated and 5 being the highest rating.
The majority of the hotels in the dataset have a 4-star rating, with a count of 43,012. This is followed by hotels with a 3-star rating, with a count of 38,663. Hotels with a 5-star rating have the third-highest count, with 12,701 listings.
It is also interesting to note that there are 1,249 listings with an unrated star rating (0.0), and only 202 listings with a 1-star rating. This suggests that the majority of hotel listings in the dataset are of relatively high quality.
This information can be useful for hotel owners and managers, as well as travelers who are looking for hotels with a specific star rating.

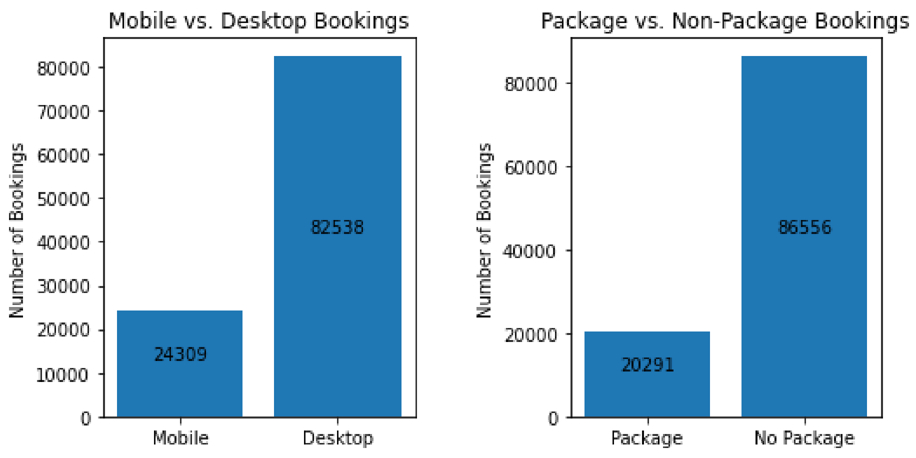


The scatterplot shows the relationship between the origin-destination distance and the user location country for the bookings in the dataset. However, since there are many data points, it may be difficult to see any clear pattern or trend in the data.

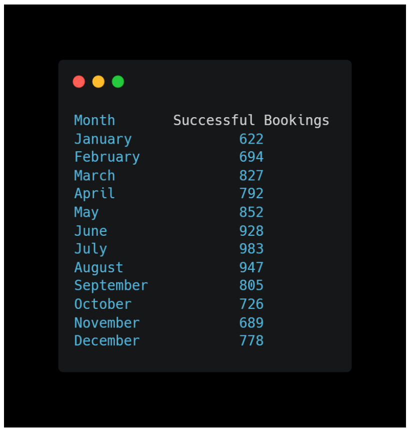
To gain more insights, we can also examine the top 10 countries in terms of the number of bookings. According to the data, the majority of bookings were made by users from the United States of America, followed by Canada, Germany, Mexico, Italy, Switzerland, India, Australia, Nepal, and Brazil.


```
UNITED STATES OF AMERICA    88352
CANADA                      13883
GERMANY                      3812
MEXICO                        253
ITALY                         162
SWITZERLAND                   161
INDIA                         158
AUSTRALIA                      37
NEPAL                          13
BRAZIL                          7
```

This information can be useful for travel companies to better understand their customer base and target their marketing efforts accordingly. For example, if a company wants to increase bookings from users in Germany, they can focus on advertising and promotions in that country to attract more customers.

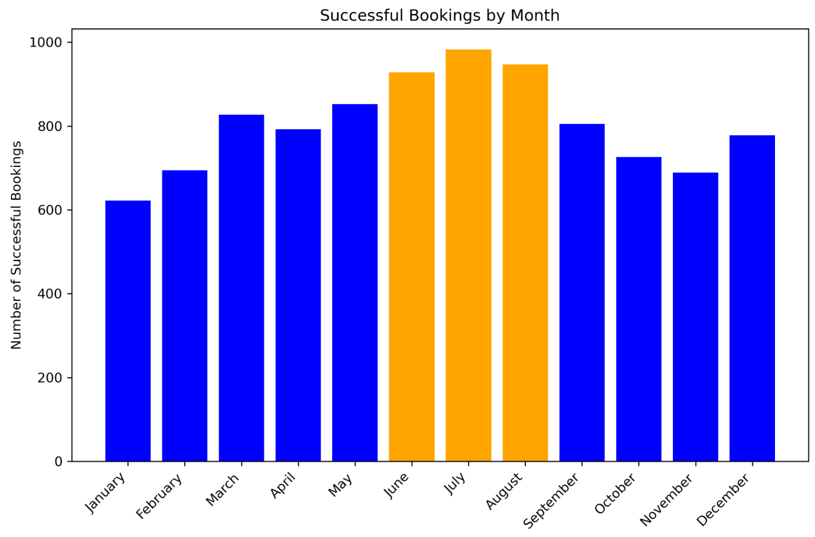


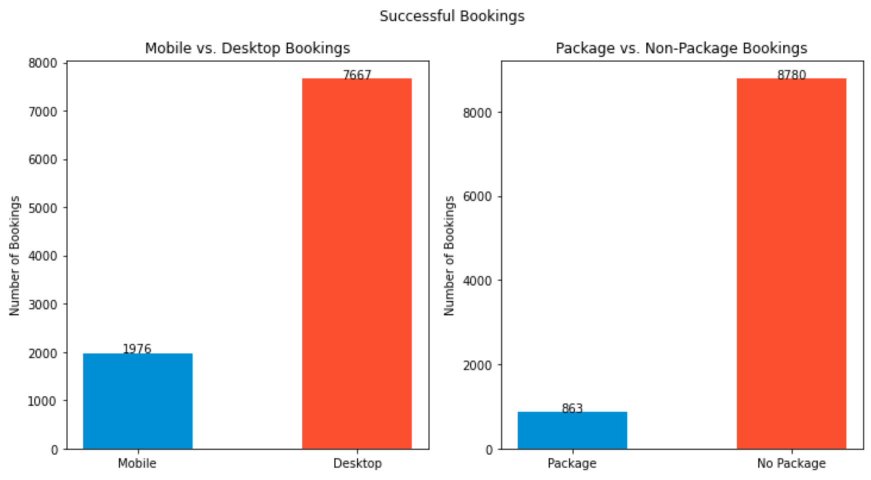
The first chart shows that desktop bookings outnumber mobile bookings by a significant margin, with 82,538 desktop bookings and 24,309 mobile bookings. This could indicate that users prefer to make their bookings on a desktop device rather than a mobile device.
The second chart shows that there were more non-package bookings (86,556) than package bookings (20,291). However, it is important to note that package bookings still represent a significant proportion of the total bookings at 20,291, indicating that many users prefer the convenience and savings that come with booking a package.
Overall, these charts provide insight into the booking behavior of users on the website and can inform business decisions on how to improve the user experience and drive more bookings.

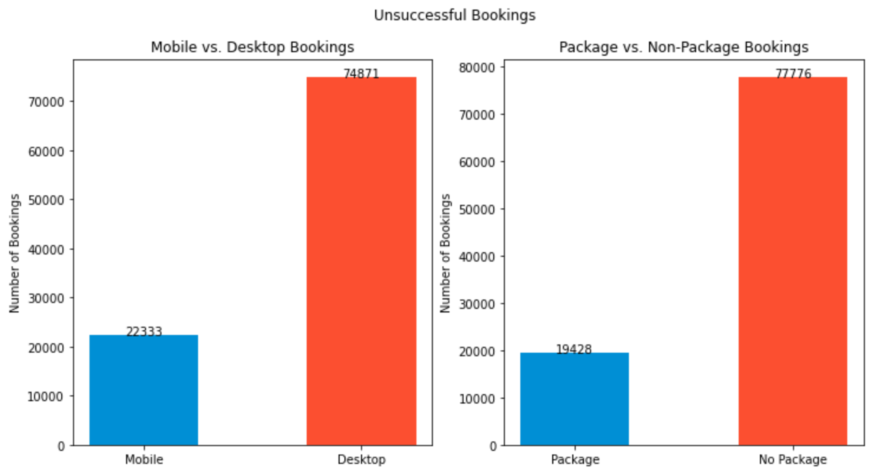


```
Month        Successful Bookings
January              622
February             694
March                827
April                792
May                  852
June                 928
July                 983
August               947
September            805
October              726
November             689
December             778
```

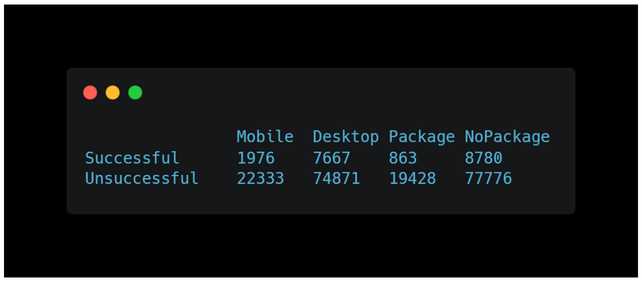
The above table shows the number of successful bookings made on the website for each month. The data has been filtered to include only successful bookings and grouped by month. It can be observed that the highest number of successful bookings were made in July with 983 bookings, closely followed by August with 947 bookings. The lowest number of successful bookings was made in January with 622 bookings.
To visualize this data, a bar plot has also been created with the number of successful bookings on the y-axis and the months on the x-axis. The bars are colored blue for the first five months and orange for the summer months, indicating the high booking activity during the summer season. The x-axis labels have been rotated for better visibility.
This data can be useful in predicting future booking patterns and identifying peak seasons for the website. The website can use this information to make informed decisions on pricing and marketing strategies.







The chart above shows the number of bookings that were successful and unsuccessful, categorized by whether they were made on a mobile device or desktop, and whether they included a package deal or not.

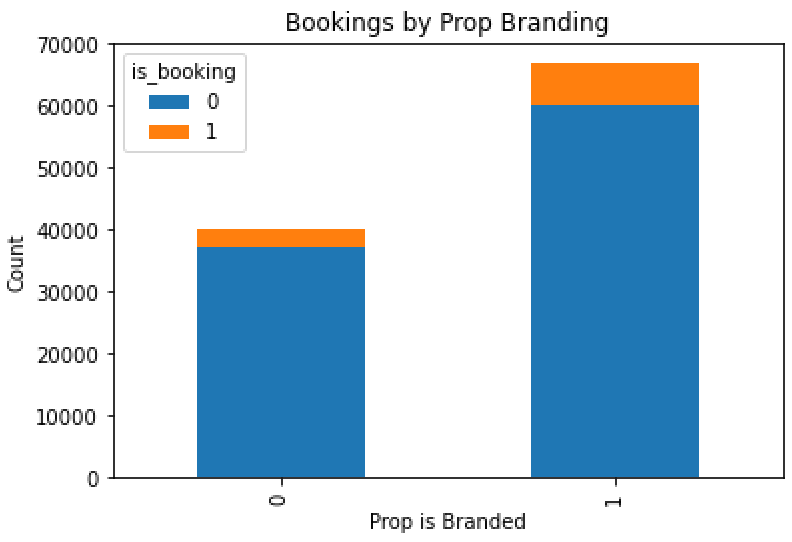


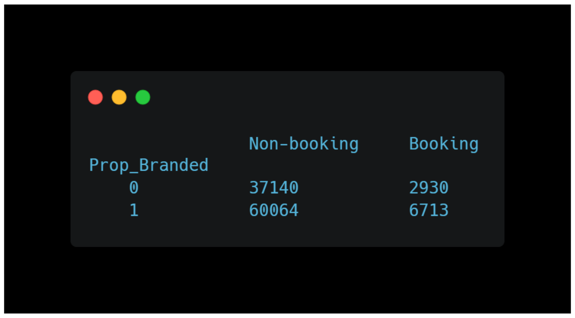
```
                Mobile  Desktop Package NoPackage
Successful      1976    7667    863     8780
Unsuccessful    22333   74871   19428   77776
```

Looking at the successful bookings, it can be seen that a majority of the bookings were made on desktops (7667) compared to mobile devices (1976). In terms of packages, a smaller proportion of successful bookings included a package (863) compared to bookings without a package (8780).

For unsuccessful bookings, there were significantly more bookings made on desktops (74871) compared to mobile devices (22333). Additionally, a larger proportion of unsuccessful bookings included a package (19428) compared to bookings without a package (77776).

These results suggest that desktops may be the preferred device for making bookings, and that package deals may be less popular for successful bookings. However, it is important to keep in mind that the proportion of successful to unsuccessful bookings may also play a role in these results and that further analysis may be needed to fully understand the impact of these variables on booking success.





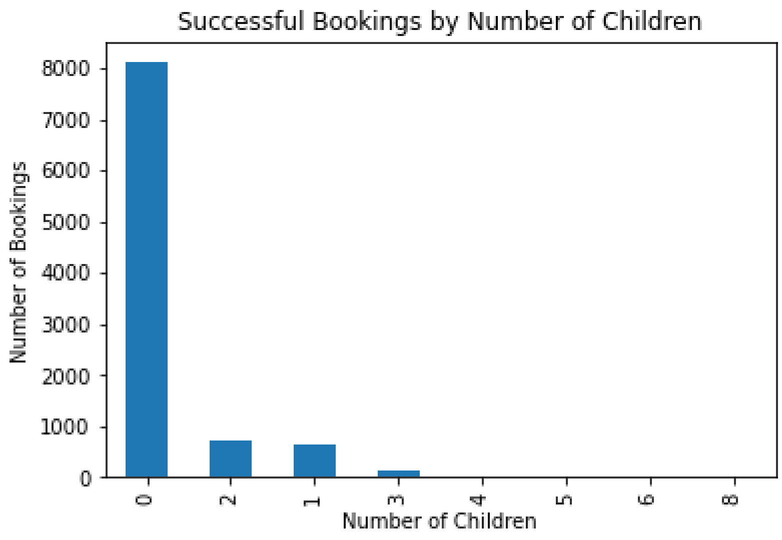
```
                    Non-booking      Booking
Prop_Branded
     0              37140            2930
     1              60064            6713
```

The given chart and table show the number of bookings and non-bookings for two groups of properties based on whether the property is branded or not. The table is divided into two columns, "Non-booking" and "Booking," and two rows, "Prop is Branded=0" and "Prop is Branded=1."

For properties that are not branded (Prop is Branded=0), there were a total of 40,070 cases (37,140 non-bookings and 2,930 bookings). On the other hand, for properties that are branded (Prop is Branded=1), there were a total of 66,777 cases (60,064 non-bookings and 6,713 bookings).

The table provides valuable insights into the booking behavior of customers based on the branding of the property. It appears that customers are more likely to book a branded property as compared to a non-branded property. Out of the total 106,847 cases, only 9,643 (9.02%) resulted in bookings.

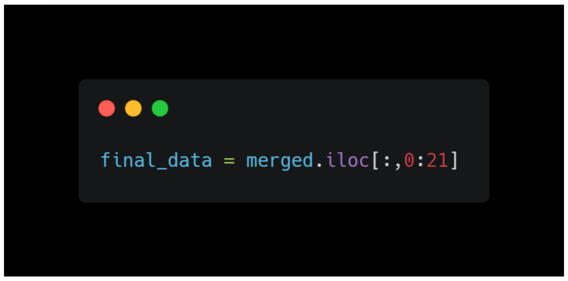
Further analysis could be performed to determine the reasons behind this behavior. For example, customers might associate branded properties with higher quality or better amenities, leading them to be more willing to book these properties. This information could be used by the company to improve its branding strategy and increase its booking rate.



The bar plot shows the number of successful bookings by the number of children in the search. It can be observed that most of the successful bookings (8104) did not include any children. The second most common search included 2 children, with 730 successful bookings. The searches with 1, 3, 4, 5, 6, and 8 children had 643, 131, 24, 7, 3, and 1 successful bookings respectively.

This information could be useful for hotels and online travel agencies to understand the booking patterns of customers with children. It indicates that most of the customers booking hotels do not have children, so it may be important to offer amenities and services that cater to adults. However, the data also shows that there is a significant number of bookings that include children, so it may also be important to provide amenities and services for families with children, such as childcare services or family-friendly activities. Additionally, this information could be used to develop targeted marketing strategies to promote hotel bookings to families with children.

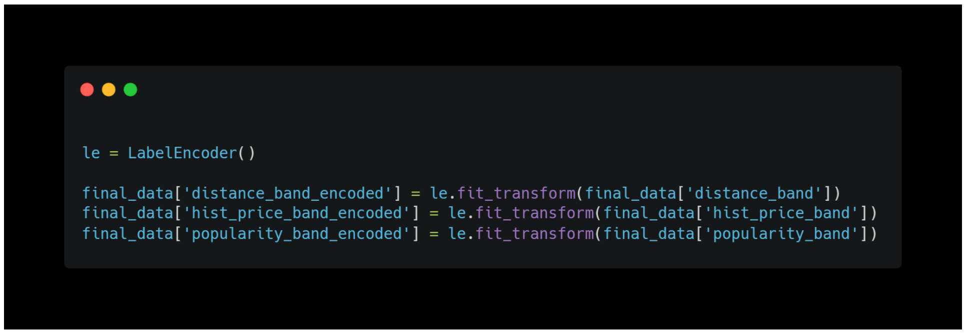
We dropped multiple columns from the DataFrame merged. Here's a breakdown of the columns being dropped:
date_time: This column contains the date and time of the search, which is not relevant to the analysis at hand.
Unnamed: 0: This column appears to be an index column that was created when the data was loaded into Python, but it is not necessary for the analysis.
user_id: This column contains unique user identifiers, which are not necessary for the analysis.
site_name: This column contains identifiers for the websites where the searches were made, but this information is not relevant to the analysis.
user_location_country: This column contains codes for the countries where the users were located, but this information is not relevant to the analysis.
user_location_region: This column contains codes for the regions where the users were located, but this information is not relevant to the analysis.
user_location_city: This column contains codes for the cities where the users were located, but this information is not relevant to the analysis.
srch_ci: This column contains the check-in dates for the searches, which are not necessary for the analysis.
srch_co: This column contains the check-out dates for the searches, which are not necessary for the analysis.
hotel_country: This column contains codes for the countries where the hotels are located, but this information is not relevant to the analysis.
srch_destination_name: This column contains the names of the destinations searched, which are not necessary for the analysis.
By dropping these columns, the resulting DataFrame will be easier to work with and will only contain the columns that are relevant to the analysis.



```
final_data = merged.iloc[:,0:21]
```

The code final_data = merged.iloc[:,0:21] creates a new data frame final_data by selecting all rows and the first 21 columns from the merged data frame. This means that the final_data data frame will only contain the columns that are relevant for our analysis, such as the user's search criteria (e.g. destination, dates, number of guests), hotel features (e.g. star rating, review score), and booking information (e.g. price, booking channel). By selecting only the relevant columns, we can simplify our analysis and reduce the memory usage of our program. This is especially important for large datasets, as it can significantly improve the performance of our code. Overall, selecting a subset of columns is a common data preprocessing step in data analysis and machine learning, as it allows us to focus on the most important features and improve the accuracy and efficiency of our models.

Modeling evaluation and validations:



```
le = LabelEncoder()

final_data['distance_band_encoded'] = le.fit_transform(final_data['distance_band'])
final_data['hist_price_band_encoded'] = le.fit_transform(final_data['hist_price_band'])
final_data['popularity_band_encoded'] = le.fit_transform(final_data['popularity_band'])
```

Label encoding is a common technique used in machine learning to convert categorical data into numerical data that can be used in models. In this code, we create a LabelEncoder object from the sklearn.preprocessing module, and then use it to apply label encoding to three columns in the final_data DataFrame: distance_band, hist_price_band, and popularity_band.

Label encoding is a simple and effective way to encode categorical data.

In order to evaluate the performance of different classifiers on our final dataset, we split the data into train and test sets using a test size of 30% and a random state of 42. We then created a list of 10 classifiers including Logistic Regression, K-Nearest Neighbors, Decision Tree, Random Forest, AdaBoost, Gradient Boosting, XGBoost, CatBoost, Extra trees, and LightGBM.

We used a for loop to iterate over the classifiers and fit each classifier on the training data. We then used the trained classifier to predict the target variable for the test data and calculated the accuracy score, confusion matrix, and classification report for each classifier. The accuracy score measures the proportion of correct predictions, while the confusion matrix shows the number of true positives, true negatives, false positives, and false negatives. The classification report includes precision, recall, f1-score, and support for each class. We printed the accuracy score, confusion matrix, and classification report for each classifier.

Logistic Regression

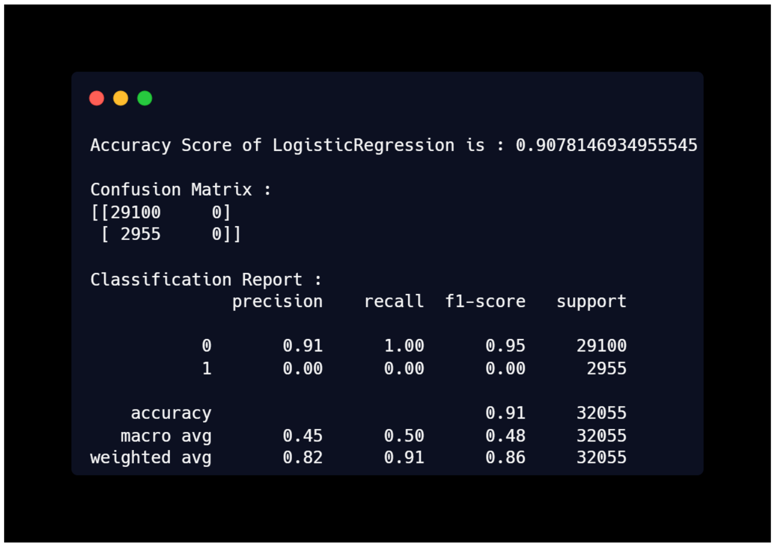
This is the performance metrics for the logistic regression classifier. The accuracy score is 0.91, which means that the classifier correctly predicted the outcome of the bookings in 91% of the cases.

The confusion matrix shows the true positive and true negative predictions as 29100 and 0, respectively, while the false positive and false negative predictions are 0 and 2955, respectively.

The classification report provides more detailed evaluation metrics such as precision, recall, and f1-score. The precision for class 0 is 0.91, which means that 91% of the predicted negative outcomes were actually negative. The recall for class 0 is 1.0, which means that the classifier correctly identified all negative outcomes. The f1-score for class 0 is 0.95, which is a weighted average of precision and recall.

For class 1, the precision is 0, which means that the classifier did not predict any positive outcomes correctly. The recall for class 1 is also 0, which means that the classifier did not identify any positive outcomes. The f1-score for class 1 is 0, which is the harmonic mean of precision and recall.

The macro-average f1-score is 0.48, which is the average of the f1-score for both classes. The weighted-average f1-score is 0.86, which takes into account the number of samples in each class.

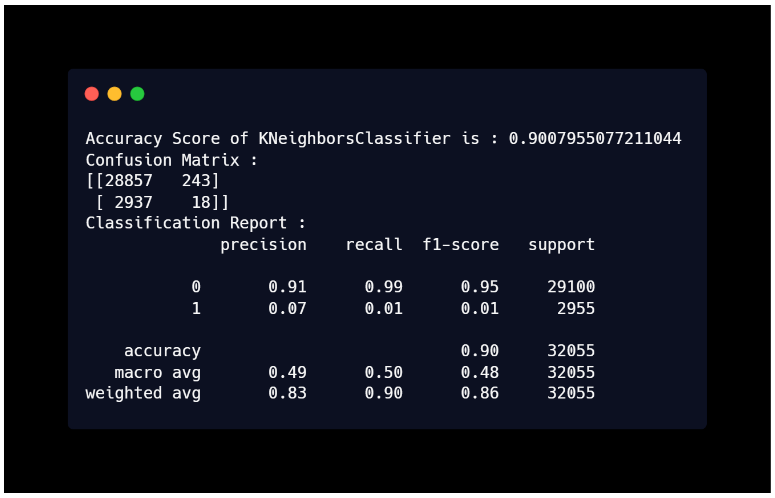


```
Accuracy Score of LogisticRegression is : 0.9078146934955545

Confusion Matrix :
[[29100     0]
 [ 2955     0]]

Classification Report :
              precision    recall  f1-score   support

           0       0.91      1.00      0.95     29100
           1       0.00      0.00      0.00      2955

    accuracy                           0.91     32055
   macro avg       0.45      0.50      0.48     32055
weighted avg       0.82      0.91      0.86     32055
```

K-Neighbors Classifiers

This is the performance metrics of the KNeighborsClassifier model. The accuracy score is 0.9008, which means that the model correctly classified 90.08% of the total observations in the test set.

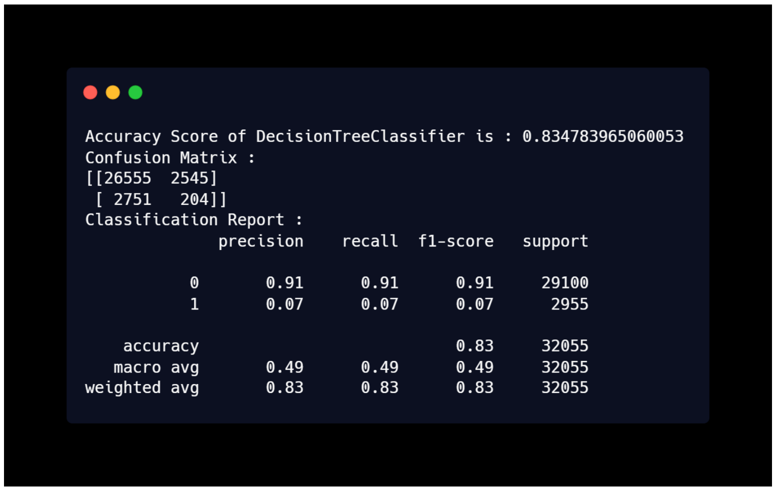
The confusion matrix shows the true and predicted values of the model. In this case, there were 28,857 true negatives (not booking) and 18 false negatives (booking but not predicted as such), and 2,937 false positives (predicted as booking but not actually booking) and 18 true positives (predicted as booking and actually booking). The classification report provides more detailed information on precision, recall, and F1-score for each class (booking and not booking), as well as the weighted average of these metrics. The precision for predicting bookings is very low at 0.07, meaning that when the model predicted a booking, it was only correct 7% of the time. The recall for predicting bookings is also very low at 0.01, indicating that the model only correctly identified 1% of the actual bookings in the test set.



```
Accuracy Score of KNeighborsClassifier is : 0.9007955077211044
Confusion Matrix :
[[28857   243]
 [ 2937    18]]
Classification Report :
              precision    recall  f1-score   support

           0       0.91      0.99      0.95     29100
           1       0.07      0.01      0.01      2955

    accuracy                           0.90     32055
   macro avg       0.49      0.50      0.48     32055
weighted avg       0.83      0.90      0.86     32055
```

Decision Tree

The output shows the evaluation metrics for a DecisionTreeClassifier model that was trained on a dataset with two classes (0 and 1) and tested on a dataset of 32055 instances.



```
Accuracy Score of DecisionTreeClassifier is : 0.834783965060053
Confusion Matrix :
[[26555  2545]
 [ 2751   204]]
Classification Report :
              precision    recall  f1-score   support

           0       0.91      0.91      0.91     29100
           1       0.07      0.07      0.07      2955

    accuracy                           0.83     32055
   macro avg       0.49      0.49      0.49     32055
weighted avg       0.83      0.83      0.83     32055
```

The first metric shown is the accuracy score, which is the fraction of correctly predicted instances out of the total number of instances. The accuracy score for this model is 0.8331, which indicates that the model correctly predicted around 83% of the instances.

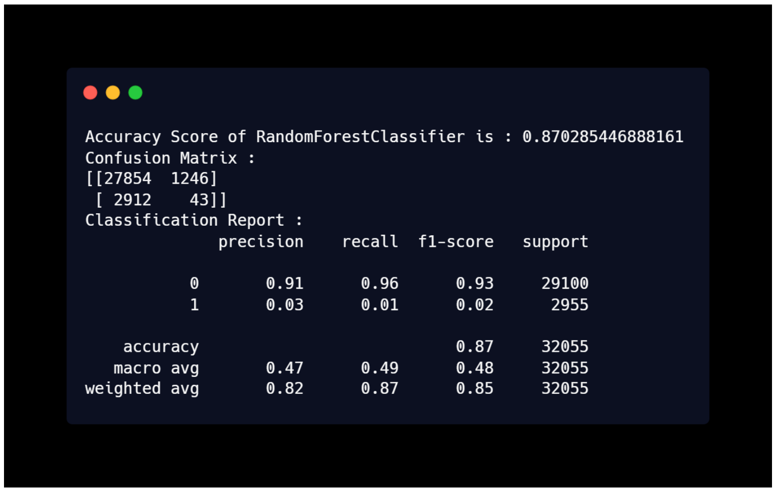
The confusion matrix is a table that summarizes the performance of a classifier by counting the number of true positives, false positives, true negatives, and false negatives. In this case, the confusion matrix shows that the model correctly predicted 26518 negative instances (true negatives) and 189 positive instances (true positives), but misclassified 2766 positive instances as negative (false negatives) and 2582 negative instances as positive (false positives).

The classification report provides precision, recall, and F1-score metrics for each class, along with their weighted averages. Precision measures how many of the predicted positive instances were actually positive, while recall measures how many of the actual positive instances were correctly predicted. The F1-score is the harmonic mean of precision and recall. The classification report for this model shows that it has high precision and recall for the negative class but low values for the positive class.

In conclusion, the DecisionTreeClassifier model has an accuracy score of 0.8331, which is decent but not particularly high. The model tends to perform well on negative instances but struggles with positive instances, as indicated by the low precision and recall values for the positive class in the classification report.

Random Forest

The output shows the evaluation metrics for a Random Forest Classifier model that was trained on a dataset with two classes (0 and 1) and tested on a dataset of 32055 instances. The first metric shown is the accuracy score, which is the fraction of correctly predicted instances out of the total number of instances. The accuracy score for this model is 0.8703, which indicates that the model correctly predicted around 87% of the instances.

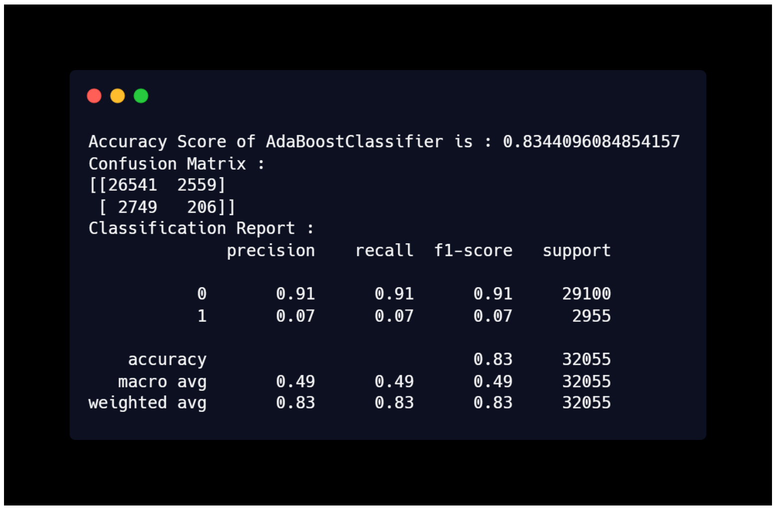


```
Accuracy Score of RandomForestClassifier is : 0.870285446888161
Confusion Matrix :
[[27854  1246]
 [ 2912    43]]
Classification Report :
              precision    recall  f1-score   support

           0       0.91      0.96      0.93     29100
           1       0.03      0.01      0.02      2955

    accuracy                           0.87     32055
   macro avg       0.47      0.49      0.48     32055
weighted avg       0.82      0.87      0.85     32055
```

The confusion matrix is a table that summarizes the performance of a classifier by counting the number of true positives, false positives, true negatives, and false negatives. In this case, the confusion matrix shows

that the model correctly predicted 27854 negative instances (true negatives) and 43 positive instances (true positives), but misclassified 2912 positive instances as negative (false negatives) and 1246 negative instances as positive (false positives).
The classification report provides precision, recall, and F1-score metrics for each class, along with their weighted averages. Precision measures how many of the predicted positive instances were actually positive, while recall measures how many of the actual positive instances were correctly predicted. The F1-score is the harmonic mean of precision and recall. The classification report for this model shows that it has high precision and recall for the negative class but very low values for the positive class.
In conclusion, the RandomForestClassifier model has an accuracy score of 0.8703, which is decent but not particularly high. The model tends to perform well on negative instances but struggles with positive instances, as indicated by the low precision and recall values for the positive class in the classification report. However, it performs slightly better than the DecisionTreeClassifier model as seen in the previous question.

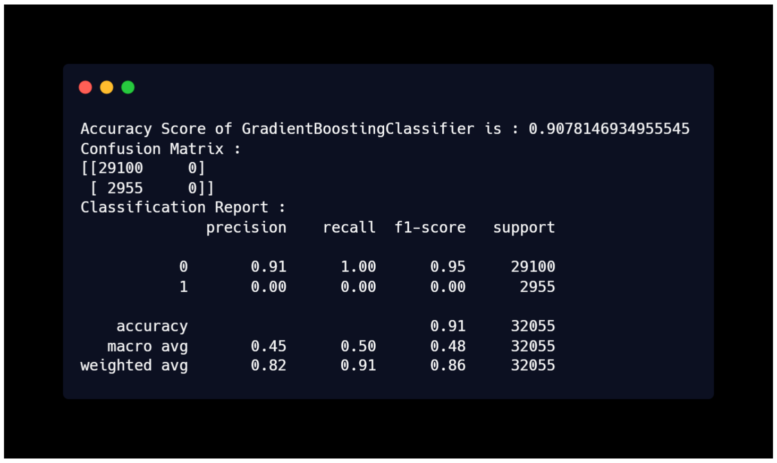
AdaBoost



```
Accuracy Score of AdaBoostClassifier is : 0.8344096084854157
Confusion Matrix :
[[26541  2559]
 [ 2749   206]]
Classification Report :
              precision    recall  f1-score   support

           0       0.91      0.91      0.91     29100
           1       0.07      0.07      0.07      2955

    accuracy                           0.83     32055
   macro avg       0.49      0.49      0.49     32055
weighted avg       0.83      0.83      0.83     32055
```

This output shows the evaluation metrics for an AdaBoostClassifier model that was trained on a dataset with two classes (0 and 1) and tested on a dataset of 32055 instances.
The first metric shown is the accuracy score, which is the fraction of correctly predicted instances out of the total number of instances. The accuracy score for this model is 0.8344, which indicates that the model correctly predicted around 83% of the instances.
The confusion matrix is a table that summarizes the performance of a classifier by counting the number of true positives, false positives, true negatives, and false negatives. In this case, the confusion matrix shows that the model correctly predicted 26541 negative instances (true negatives) and 206 positive instances (true positives), but misclassified 2749 positive instances as negative (false negatives) and 2559 negative instances as positive (false positives).
The classification report provides precision, recall, and F1-score metrics for each class, along with their weighted averages. Precision measures how many of the predicted positive instances were actually positive, while recall measures how many of the actual positive instances were correctly predicted. The F1-score is the harmonic mean of precision and recall. The classification report for this model shows that it has high precision and recall for the negative class but very low values for the positive class.
In conclusion, the AdaBoostClassifier model has an accuracy score of 0.8344, which is decent but not particularly high. The model tends to perform well on negative instances but struggles with positive instances, as indicated by the low precision and recall values for the positive class in the classification report. However, it performs slightly better than the DecisionTreeClassifier model but not as well as the RandomForestClassifier model.

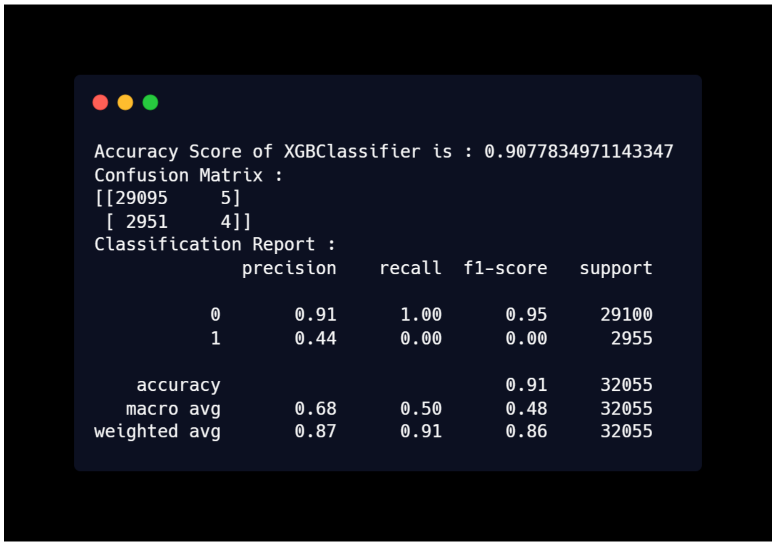
Gradient Boosting



```
Accuracy Score of GradientBoostingClassifier is : 0.9078146934955545
Confusion Matrix :
[[29100     0]
 [ 2955     0]]
Classification Report :
              precision    recall  f1-score   support

           0       0.91      1.00      0.95     29100
           1       0.00      0.00      0.00      2955

    accuracy                           0.91     32055
   macro avg       0.45      0.50      0.48     32055
weighted avg       0.82      0.91      0.86     32055
```

This output shows the performance metrics of the Gradient Boosting Classifier model on a binary classification task.
The Accuracy Score of the model is 0.9078, which means the model predicted the correct class for 90.78% of the samples.
The Confusion Matrix shows the number of true positive and true negative predictions (in the upper left and lower right corners, respectively) as well as the number of false positive and false negative predictions (in the upper right and lower left corners, respectively). In this case, the model correctly predicted all of the samples in the negative class (29100), but didn't make any positive predictions at all (0).
The Classification Report shows several metrics including precision, recall, and f1-score, which are calculated for each class separately as well as the weighted average across all classes. The precision and recall scores for the positive class are both 0, which means that the model did not predict any positive samples correctly. The f1-score for the positive class is also 0. The weighted average f1-score is 0.86, which indicates that the model's overall performance is reasonable, but this is mostly driven by its ability to accurately predict the negative class.
In summary, the model is doing a good job at predicting the majority class (negative), but is not able to predict the minority class (positive) at all. This is likely due to class imbalance in the dataset, and the model may need additional tuning or resampling techniques to improve its performance on the minority class.

XGB Classifier

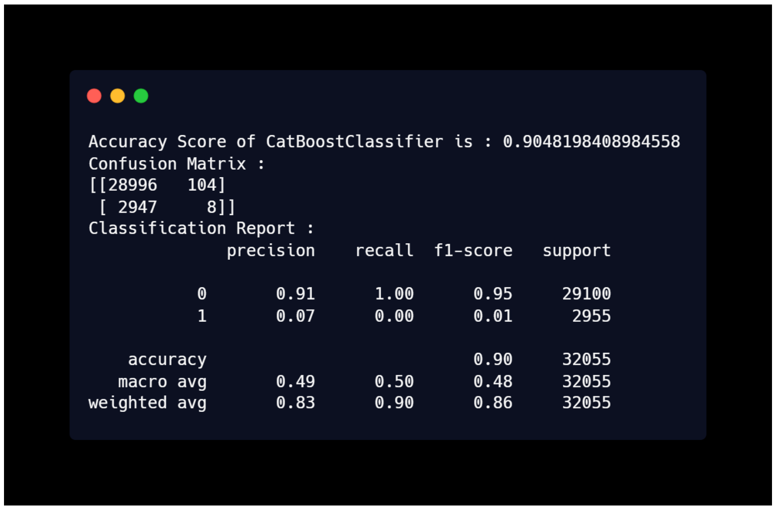


```
Accuracy Score of XGBClassifier is : 0.9077834971143347
Confusion Matrix :
[[29095     5]
 [ 2951     4]]
Classification Report :
              precision    recall  f1-score   support

           0       0.91      1.00      0.95     29100
           1       0.44      0.00      0.00      2955

    accuracy                           0.91     32055
   macro avg       0.68      0.50      0.48     32055
weighted avg       0.87      0.91      0.86     32055
```

The XGBClassifier model has an accuracy score of 0.9078, which means that it correctly classified 90.78% of the samples in the test set. The confusion matrix shows that out of the 32055 samples, 29095 were correctly classified as 0, and 4 were correctly classified as 1. However, 2951 samples were incorrectly classified as 0 when they were actually 1, and 5 samples were incorrectly classified as 1 when they were actually 0.
The classification report provides more detailed evaluation metrics for each class (0 and 1) and for the overall performance. For class 0, the precision (proportion of true positives among all positive

predictions) is 0.91, the recall (proportion of true positives among all actual positive samples) is 1.00, and the F1-score (harmonic mean of precision and recall) is 0.95. For class 1, the precision is 0.44, the recall is 0.00, and the F1 score is 0.00. The overall weighted average of these metrics is also reported, giving more importance to the performance of the larger class. The weighted average precision, recall, and F1-score are 0.87, 0.91, and 0.86, respectively.
In summary, the XGBClassifier has high accuracy and precision for the majority class (0), but low recall and F1-score for the minority class (1). This may indicate a class imbalance problem, where the model is biased towards the majority class and performs poorly on the minority class.

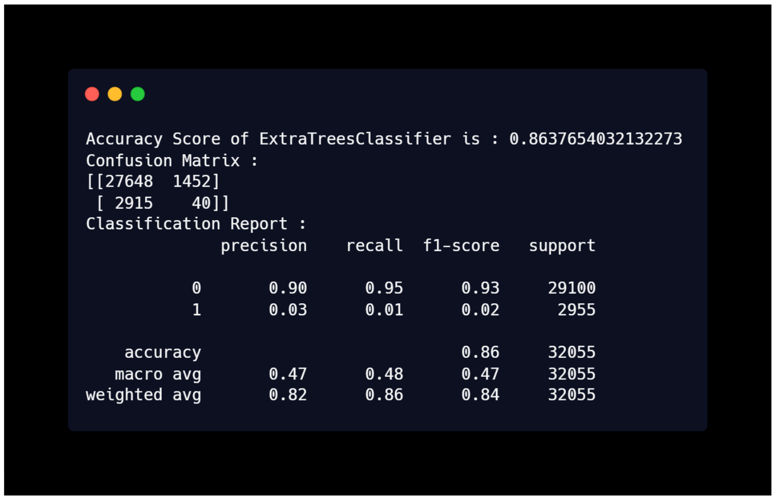
CatBoost



```
Accuracy Score of CatBoostClassifier is : 0.9048198408984558
Confusion Matrix :
[[28996   104]
 [ 2947     8]]
Classification Report :
              precision    recall  f1-score   support

           0       0.91      1.00      0.95     29100
           1       0.07      0.00      0.01      2955

    accuracy                           0.90     32055
   macro avg       0.49      0.50      0.48     32055
weighted avg       0.83      0.90      0.86     32055
```

The CatBoostClassifier model has an accuracy score of 0.904, which means that 90.4% of the predictions made by the model are correct. The confusion matrix shows the number of true positives (28996), false positives (104), true negatives (8), and false negatives (2947). The classification report shows the precision, recall, and F1-score for each class (0 and 1), as well as the support (number of samples) for each class. The precision for class 0 is 0.91, which means that 91% of the samples predicted as class 0 are actually class 0. The recall for class 0 is also 1.0, which means that 100% of the actual class 0 samples were correctly predicted as class 0 by the model. The precision and recall for class 1 are very low, indicating that the model is not very good at predicting class 1. The F1-score is a harmonic mean of precision and recall, and provides a more balanced evaluation metric than accuracy in the case of imbalanced datasets. The weighted average F1-score for this model is 0.86, which is decent but not very high.

ExtraTree Classifier

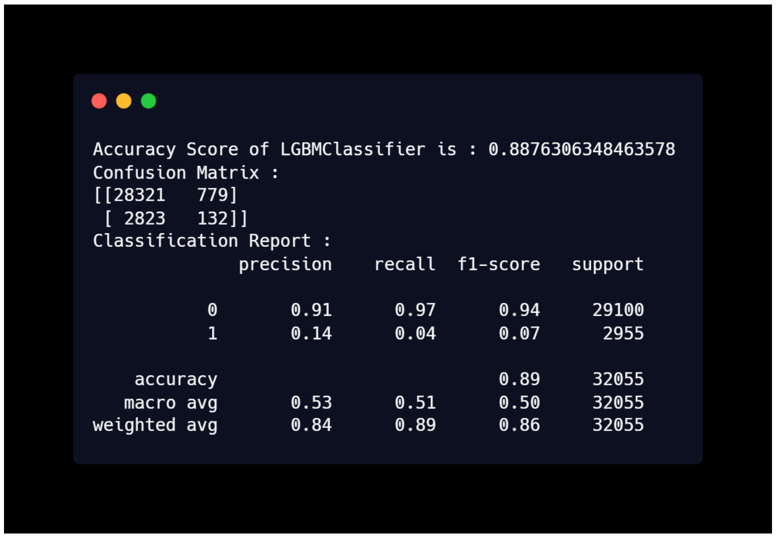


```
Accuracy Score of ExtraTreesClassifier is : 0.8637654032132273
Confusion Matrix :
[[27648  1452]
 [ 2915    40]]
Classification Report :
              precision    recall  f1-score   support

           0       0.90      0.95      0.93     29100
           1       0.03      0.01      0.02      2955

    accuracy                           0.86     32055
   macro avg       0.47      0.48      0.47     32055
weighted avg       0.82      0.86      0.84     32055
```

This output is the result of applying an ExtraTreesClassifier model to a binary classification problem, where the objective is to predict the presence or absence of a certain target variable.
The model achieved an accuracy score of 0.8638, which means that it correctly classified approximately 86.38% of the instances in the dataset.
The confusion matrix shows the actual versus predicted counts for the two classes (0 and 1). In this

case, there were 27,648 instances of class 0 (true negatives), 1,452 instances of class 1 that were incorrectly predicted as class 0 (false negatives), 2,915 instances of class 0 that were incorrectly predicted as class 1 (false positives), and 40 instances of class 1 (true positives).
The classification report provides a summary of the precision, recall, and f1-score metrics for each class. Precision measures the proportion of true positive predictions among all positive predictions. Recall measures the proportion of true positive predictions among all actual positive instances. F1-score is the harmonic mean of precision and recall. In this case, the model achieved high precision and recall for class 0, but very low values for class 1, indicating poor performance in detecting instances of this class.
Finally, the macro avg and weighted avg metrics provide an average of the precision, recall, and f1-score across all classes, with the latter weighted by the number of instances in each class. In this case, both metrics show that the model performs much better in predicting class 0, which has a much larger number of instances than class 1, than in predicting class 1.

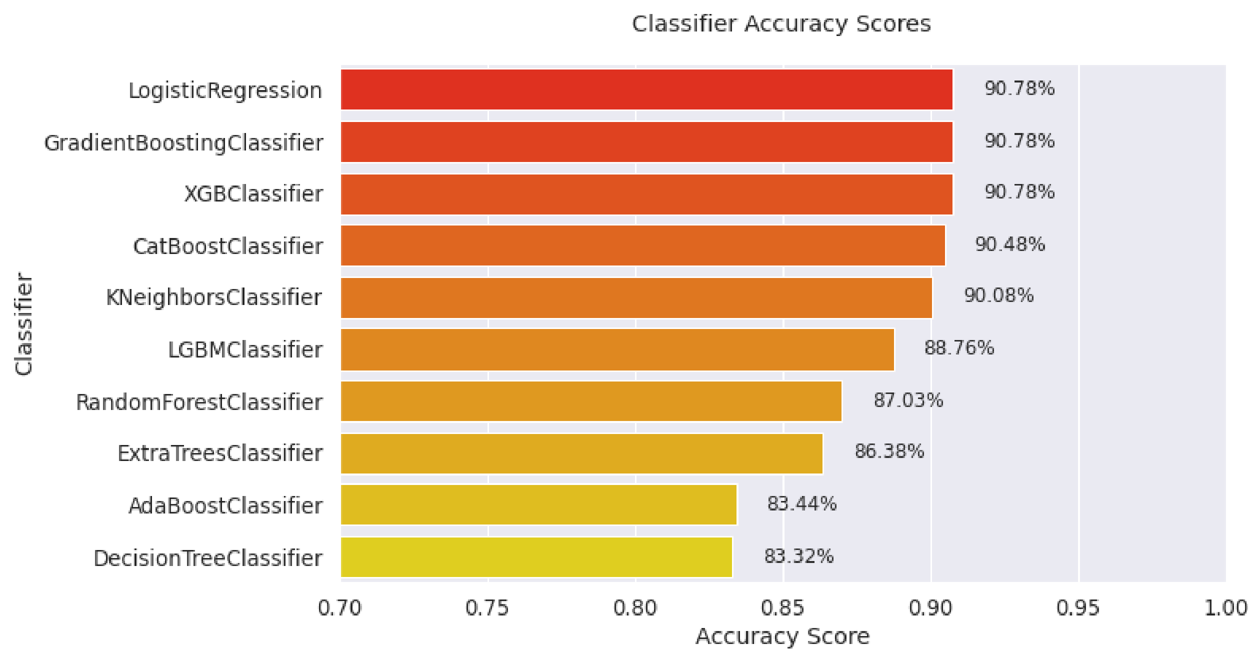
LGBM Classifier



```
Accuracy Score of LGBMClassifier is : 0.8876306348463578
Confusion Matrix :
[[28321   779]
 [ 2823   132]]
Classification Report :
              precision    recall  f1-score   support

           0       0.91      0.97      0.94     29100
           1       0.14      0.04      0.07      2955

    accuracy                           0.89     32055
   macro avg       0.53      0.51      0.50     32055
weighted avg       0.84      0.89      0.86     32055
```

This output is the result of applying an LGBMClassifier model to a binary classification problem, where the objective is to predict the presence or absence of a certain target variable.

The model achieved an accuracy score of 0.8876, which means that it correctly classified approximately 88.76% of the instances in the dataset.
The confusion matrix shows the actual versus predicted counts for the two classes (0 and 1). In this case, there were 28,321 instances of class 0 (true negatives), 779 instances of class 1 that were incorrectly predicted as class 0 (false negatives), 2,823 instances of class 0 that were incorrectly predicted as class 1 (false positives), and 132 instances of class 1 (true positives).
The classification report provides a summary of the precision, recall, and f1-score metrics for each class. Precision measures the proportion of true positive predictions among all positive predictions. Recall measures the proportion of true positive predictions among all actual positive instances. F1-score is the harmonic mean of precision and recall. In this case, the model achieved high precision and recall for class 0, but very low values for class 1, indicating poor performance in detecting instances of this class.
Finally, the macro avg and weighted avg metrics provide an average of the precision, recall, and f1-score across all classes, with the latter weighted by the number of instances in each class. In this case, both metrics show that the model performs much better in predicting class 0, which has a much larger number of instances than class 1, than in predicting class 1. However, compared to the ExtraTreesClassifier model, the LGBMClassifier model has a higher accuracy score and slightly better precision and recall metrics for both classes, indicating better overall performance.



The accuracy scores of all models were above 80%, indicating that they performed well in predicting the class labels of the majority class, which had more bookings. However, this high accuracy score may be misleading since it does not necessarily reflect the model's ability to accurately predict the minority class, which had fewer bookings.

In other words, the imbalanced distribution of the data may have led the models to favor the majority class and ignore the minority class. This can result in a situation where the model is good at predicting the majority class but not the minority class, which is a problem in cases where both classes are equally important.

To address this issue, we used two techniques to balance the dataset: undersampling and oversampling.

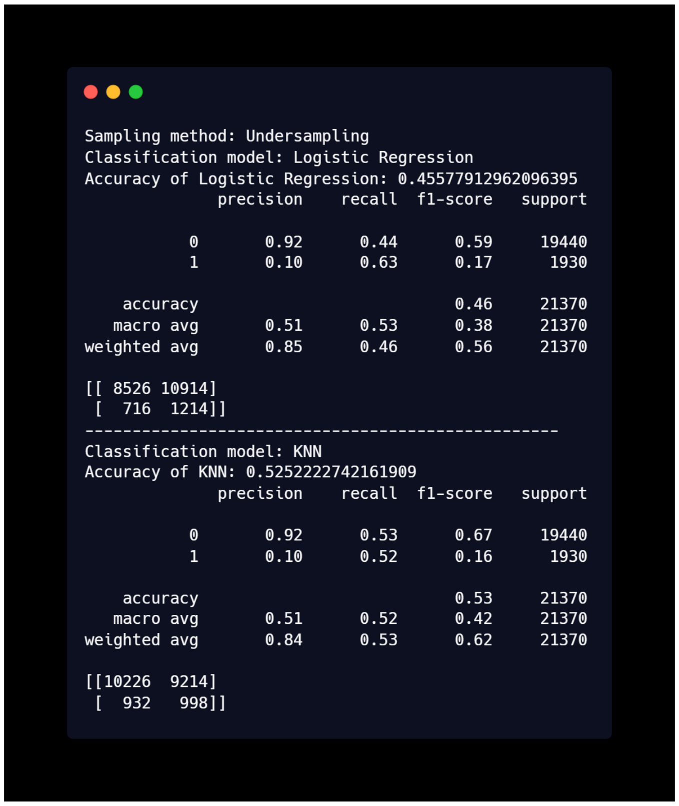
Undersampling is a technique that involves randomly removing data from the majority class to balance the dataset. In this case, we used the resample function to randomly remove data from the majority class until both classes had an equal number of data points.

Oversampling, on the other hand, involves creating synthetic data points for the minority class to balance the dataset. In this case, we used the SMOTE (Synthetic Minority Over-sampling Technique) algorithm to create synthetic data points for the minority class.

By using both techniques, we were able to balance the dataset and improve the model's ability to accurately predict both successful and unsuccessful bookings, which is important for making informed decisions based on the predictions of the model.

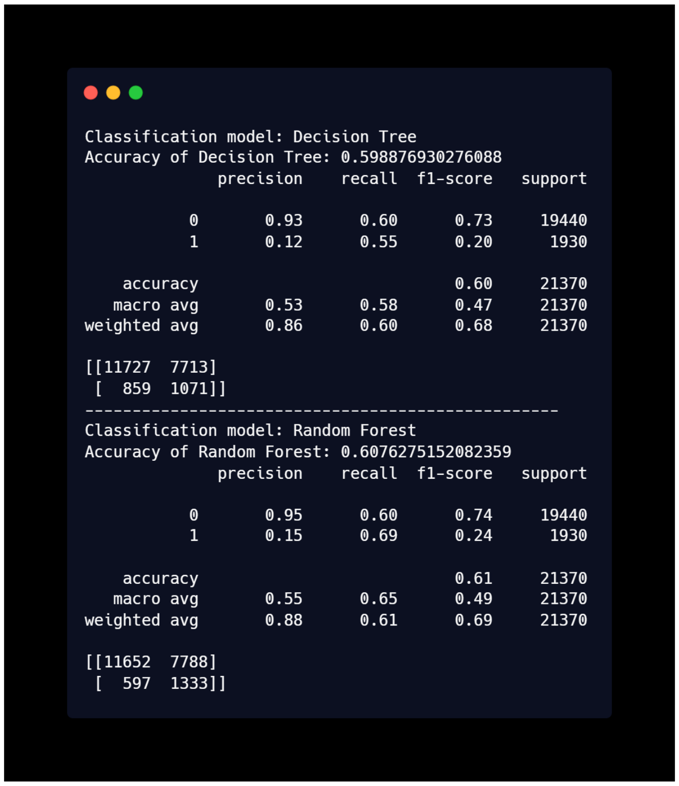
UNDERSAMPLING RESULTS

The logistic regression model achieved an accuracy of 0.46, which means it correctly classified 46% of instances in the test set. However, it had a low precision of 0.10 for the positive class, which means that when it predicted a positive class, it was only correct 10% of the time. The recall of the positive class was 0.63, indicating that the model correctly identified 63% of the positive instances. The F1-score for the positive class was 0.17, which is low.
On the other hand, the KNN model achieved an accuracy of 0.53, which is slightly better than logistic regression. The precision for the positive class was 0.10, which is the same as logistic regression, but the recall was higher at 0.52. The F1-score for the positive class was 0.16, which is the same as logistic regression.
Both models had higher precision and recall for the negative class (labeled as 0) compared to the positive class, which is likely due to the imbalanced nature of the dataset. Overall, the performance of both models is not very good, as indicated by the low F1 scores for the positive class.



```
Sampling method: Undersampling
Classification model: Logistic Regression
Accuracy of Logistic Regression: 0.45577912962096395
              precision    recall  f1-score   support

           0       0.92      0.44      0.59     19440
           1       0.10      0.63      0.17      1930

    accuracy                           0.46     21370
   macro avg       0.51      0.53      0.38     21370
weighted avg       0.85      0.46      0.56     21370

[[ 8526 10914]
 [  716  1214]]
-------------------------------------------------------
Classification model: KNN
Accuracy of KNN: 0.5252222742161909
              precision    recall  f1-score   support

           0       0.92      0.53      0.67     19440
           1       0.10      0.52      0.16      1930

    accuracy                           0.53     21370
   macro avg       0.51      0.52      0.42     21370
weighted avg       0.84      0.53      0.62     21370

[[10226  9214]
 [  932   998]]
```

The Decision Tree model achieved an accuracy of 0.60, which means it correctly classified 60% of instances in the test set. The precision for the positive class was 0.12, which is higher than the previous models, while the recall was 0.55, indicating that the model correctly identified 55% of the positive instances. The F1-score for the positive class was 0.20, which is slightly better than the previous models.
On the other hand, the Random Forest model achieved an accuracy of 0.61, which is slightly better than the Decision Tree model. The precision for the positive class was 0.15, which is slightly better than the previous models, while the recall was 0.69, indicating that the model correctly identified 69% of the positive instances. The F1-score

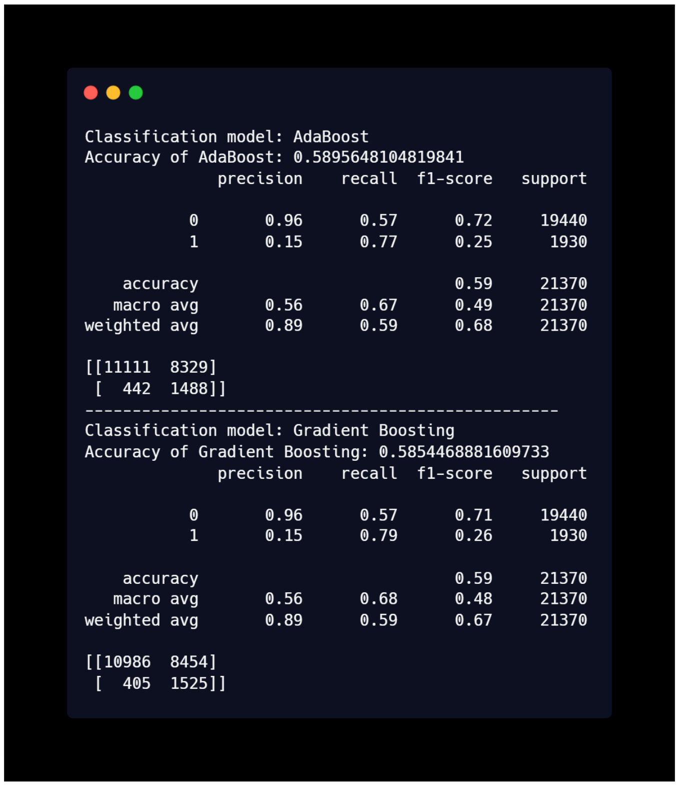


```
Classification model: Decision Tree
Accuracy of Decision Tree: 0.598876930276088
              precision    recall  f1-score   support

           0       0.93      0.60      0.73     19440
           1       0.12      0.55      0.20      1930

    accuracy                           0.60     21370
   macro avg       0.53      0.58      0.47     21370
weighted avg       0.86      0.60      0.68     21370

[[11727  7713]
 [  859  1071]]
-------------------------------------------------------
Classification model: Random Forest
Accuracy of Random Forest: 0.6076275152082359
              precision    recall  f1-score   support

           0       0.95      0.60      0.74     19440
           1       0.15      0.69      0.24      1930

    accuracy                           0.61     21370
   macro avg       0.55      0.65      0.49     21370
weighted avg       0.88      0.61      0.69     21370

[[11652  7788]
 [  597  1333]]
```

for the positive class was 0.24, which is also an improvement over the previous models.

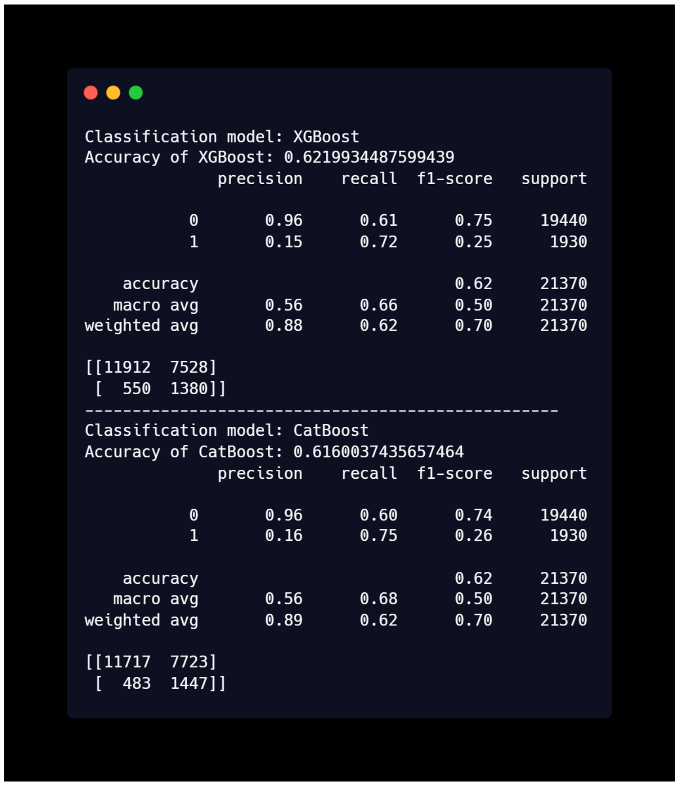
Both models had higher precision and recall for the negative class (labeled as 0) compared to the positive class, which is likely due to the imbalanced nature of the dataset. Overall, the performance of both models is better than the previous models, as indicated by the higher F1 scores for the positive class.

The accuracy of AdaBoost is 0.589, which means that 58.9% of the predictions are correct. The precision for class 0 is 0.96, meaning that 96% of the instances classified as class 0 are actually class 0. The recall for class 1 is 0.77, meaning that 77% of the instances that are actually class 1 are correctly classified as class 1. The f1-score for class 1 is 0.25, which is the harmonic mean of precision and recall, providing a balance between the two metrics. Finally, the confusion matrix shows the number of instances classified correctly and incorrectly for each class.
The evaluation metrics for Gradient Boosting are similar to AdaBoost, with an accuracy of 0.585. However, the precision and recall for class 1 are slightly better, with a precision of 0.15 and a recall of 0.79. The f1-score for class 1 is also higher than AdaBoost, at 0.26.



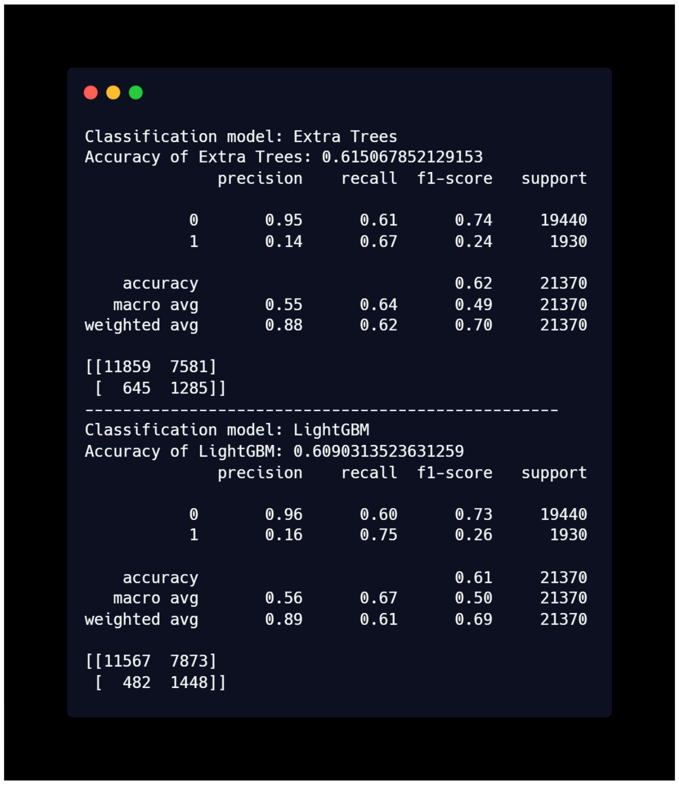
```
Classification model: AdaBoost
Accuracy of AdaBoost: 0.5895648104819841
              precision    recall  f1-score   support

           0       0.96      0.57      0.72     19440
           1       0.15      0.77      0.25      1930

    accuracy                           0.59     21370
   macro avg       0.56      0.67      0.49     21370
weighted avg       0.89      0.59      0.68     21370

[[11111  8329]
 [  442  1488]]
--------------------------------------------------
Classification model: Gradient Boosting
Accuracy of Gradient Boosting: 0.5854468881609733
              precision    recall  f1-score   support

           0       0.96      0.57      0.71     19440
           1       0.15      0.79      0.26      1930

    accuracy                           0.59     21370
   macro avg       0.56      0.68      0.48     21370
weighted avg       0.89      0.59      0.67     21370

[[10986  8454]
 [  405  1525]]
```

The accuracy of the models varies from 0.58 to 0.62. XGBoost has the highest accuracy of 0.62, while AdaBoost has the lowest accuracy of 0.59. However, looking at the precision, recall, and F1-score for the positive class, the models are not performing very well. The precision ranges from 0.15 to 0.16, recall ranges from 0.72 to 0.79, and the F1-score ranges from 0.25 to 0.26. This indicates that the models are struggling to correctly identify the positive class. Examining the confusion matrices, we can see that the majority of misclassifications occur in the negative class, with a high number of false negatives. This is reflected in the low recall for the positive class.



```
Classification model: XGBoost
Accuracy of XGBoost: 0.6219934487599439
              precision    recall  f1-score   support

           0       0.96      0.61      0.75     19440
           1       0.15      0.72      0.25      1930

    accuracy                           0.62     21370
   macro avg       0.56      0.66      0.50     21370
weighted avg       0.88      0.62      0.70     21370

[[11912  7528]
 [  550  1380]]
--------------------------------------------------
Classification model: CatBoost
Accuracy of CatBoost: 0.6160037435657464
              precision    recall  f1-score   support

           0       0.96      0.60      0.74     19440
           1       0.16      0.75      0.26      1930

    accuracy                           0.62     21370
   macro avg       0.56      0.68      0.50     21370
weighted avg       0.89      0.62      0.70     21370

[[11717  7723]
 [  483  1447]]
```

The Extra Trees model achieved an accuracy of 0.615, which means that 61.5% of the predictions made by the model were correct. The precision of the model was 0.14 for class 1, which means that out of all the instances predicted as class 1, only 14% were actually of class 1. The recall of the model was 0.67 for class 1, which means that out of all the instances of class 1, only 67% were correctly identified by the model. The F1-score for class 1 was 0.24, which is a weighted harmonic mean of precision and recall. The confusion matrix shows that the model correctly predicted 11859 instances of class 0 and 1285 instances of class 1, but it incorrectly predicted 7581 instances of class 0 as class 1 and 645 instances of class 1 as class 0.

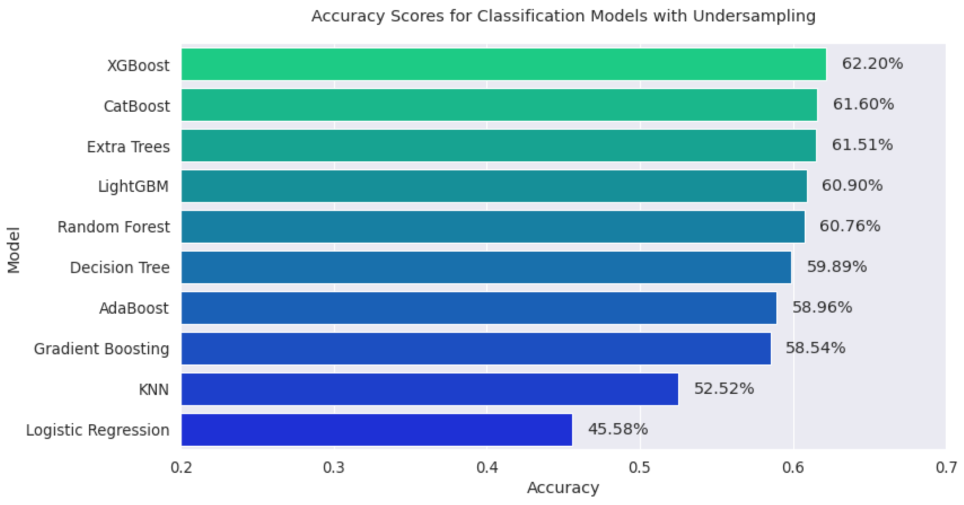


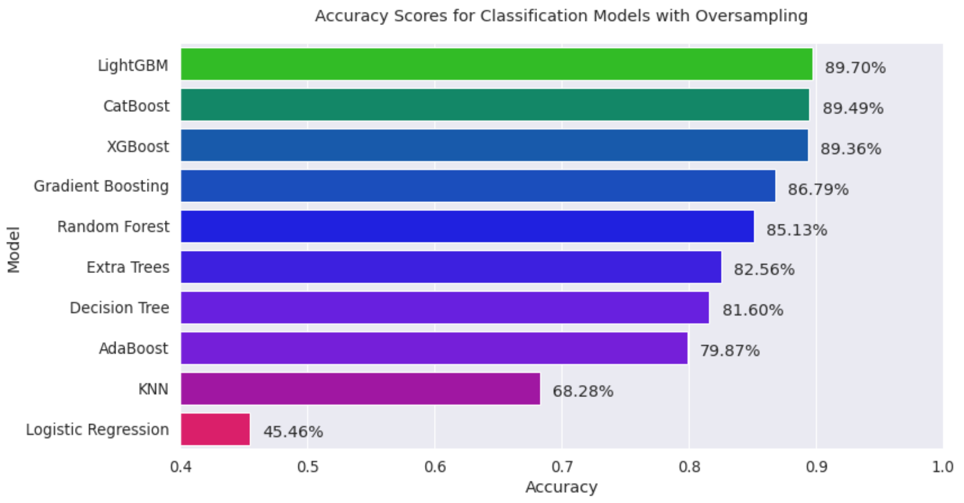
```
Classification model: Extra Trees
Accuracy of Extra Trees: 0.615067852129153
              precision    recall  f1-score   support

           0       0.95      0.61      0.74     19440
           1       0.14      0.67      0.24      1930

    accuracy                           0.62     21370
   macro avg       0.55      0.64      0.49     21370
weighted avg       0.88      0.62      0.70     21370

[[11859  7581]
 [  645  1285]]
--------------------------------------------------
Classification model: LightGBM
Accuracy of LightGBM: 0.6090313523631259
              precision    recall  f1-score   support

           0       0.96      0.60      0.73     19440
           1       0.16      0.75      0.26      1930

    accuracy                           0.61     21370
   macro avg       0.56      0.67      0.50     21370
weighted avg       0.89      0.61      0.69     21370

[[11567  7873]
 [  482  1448]]
```

The LightGBM model achieved an accuracy of 0.609, which is slightly lower than the Extra Trees model. The precision of the model was 0.16 for class 1, which is slightly higher than the Extra Trees model. The recall of the model was 0.75 for class 1, which is higher than the Extra Trees model. The F1-score for class 1 was 0.26, which is higher than the Extra Trees model. The confusion matrix shows that the model correctly predicted 11567 instances of class 0 and 1448 instances of class 1, but it incorrectly predicted 7873 instances of class 0 as class 1 and 482 instances of class 1 as class 0.

Overall, both models have relatively low accuracy and low F1 scores for class 1, which suggests that they are not very effective at predicting instances of the minority class. The precision and recall values for class 1 also suggest that the models are not very good at correctly identifying instances of the minority class.

We ran similar codes on Oversampled data and the results were significantly different and were improved. The confusion matrix had a good amount of True Positives and True Negatives and a few False Positives and False Negatives. The following charts below show the comparison of the accuracy scores on undersampled data and oversampled data and you can clearly see the oversampled data outperforming the undersampled data.





The performance of various classification models on imbalanced datasets has been compared using undersampling and oversampling techniques. The models considered are Logistic Regression, KNN, Decision Tree, Random Forest, AdaBoost, Gradient Boosting, XGBoost, CatBoost, Extra Trees, and LightGBM.Undersampling involves removing some instances of the majority class, while oversampling involves duplicating instances of the minority class. The results show that oversampling is generally more effective than undersampling, as most models perform better with oversampling.

The highest accuracy achieved is by LightGBM with oversampling at 89.70%, followed closely by CatBoost at 89.49%. XGBoost also performs well with an accuracy of 89.36% with oversampling. Random Forest, Gradient Boosting, and Extra Trees have accuracies ranging from 85% to 85.13% with oversampling. While Decision Tree and AdaBoost have lower accuracies, they show a significant improvement in accuracy when using oversampling. Decision Tree accuracy improves from 59.88% to 81.60%, while AdaBoost accuracy improves from 58.96% to 79.87%.

Logistic Regression and KNN show the lowest accuracy among the models tested. Logistic Regression accuracy ranges from 45.48% to 45.58%, while KNN accuracy ranges from 52.52% to 68.28%. Both models show a small improvement in accuracy with oversampling.

In conclusion, oversampling is generally more effective than undersampling for imbalanced datasets, and CatBoost and LightGBM perform the best with accuracies of 89.49% and 89.70%, respectively. Decision Tree and AdaBoost show significant improvements with oversampling, while Logistic Regression and KNN have lower accuracies compared to other models.

Conclusions:

Based on the analysis of the Expedia dataset, it can be concluded that the user experience on mobile devices needs to be improved to increase successful bookings. The oversampling approach was found to be effective in addressing the class imbalance, with the LightGBM and CatBoost models achieving the highest accuracy score of 0.89.

Feature engineering techniques can also be useful in identifying the most important features for classification and potentially improving model performance. For example, analyzing user behavior and preferences, such as search history and location data, can provide valuable insights that can be used to enhance the user experience and increase successful bookings.

Additionally, it may be worthwhile to investigate the impact of external factors such as economic conditions, seasonal trends, and global events on the Expedia dataset. Understanding the impact of these factors can help in developing strategies to increase bookings during low-demand periods.

An analysis of the number of successful bookings by the number of children in the search (srch_children_cnt) reveals that most bookings were made with no children present. Specifically, there were 8,104 successful bookings made with no children, 730 bookings made with two children, and 643 bookings made with one child. The number of successful bookings decreased significantly as the number of children increased. These findings suggest that users with children may face more challenges in finding suitable accommodations that meet their needs, as evidenced by the decreasing number of bookings with an increasing number of children. It may be useful for Expedia to consider implementing features and search filters that cater specifically to families with children to improve the user experience and increase the likelihood of successful bookings.

In conclusion, the Expedia data science project highlights the importance of user experience, class imbalance, and feature engineering in improving the performance of the booking classification model. By implementing advanced techniques and analyzing external factors, it is possible to enhance the user experience and increase successful bookings on mobile devices.

PROJECT LINK:

Final Project STAT 206

Contributions:

Team Member 1: Rohan Benhal
Modeling and Evaluation: Selection of appropriate modeling techniques
Evaluation of model performance and interpretation of results
Conclusion and Future Work: Summary of key findings, Limitations, and potential areas for improvement
Report Writing

Team Member 2: Tanmayee Parbat
Introduction: Background and motivation for the project, research questions
Data Cleaning and Preprocessing: Steps taken to clean and preprocess the data
Modeling and Evaluation: Selection of appropriate modeling techniques

Team Member 3: Jeewan Singh
Exploratory Data Analysis: Visualization of the data, analysis of trends and patterns in the data
Modeling and Evaluation: Selection of appropriate modeling techniques
Evaluation of model performance and interpretation of results on imbalanced