

Hotel Bookings Data Analysis

Phase 2

Vrashi Shrivastava

vrashish@buffalo.edu

Rohit Ramichetty

rohithra@buffalo.edu

1. Problem Statement

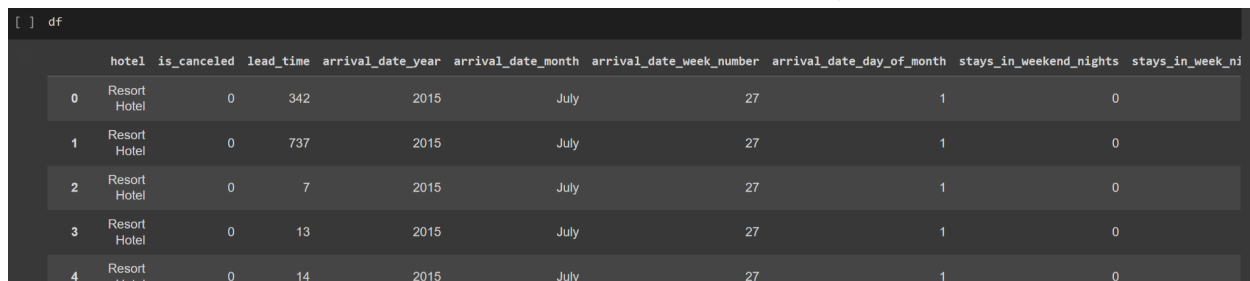
For this Project, we are going to take a Hotel Booking dataset, clean it by handling missing values and managing the format of the data, perform EDA on it to understand it better and run it through multiple models to draw intelligent information from it regarding the profits revenue, the possible bookings along with cancellations, and customer satisfaction of the hotel to ultimately predict the demand of the hotel and how can it be benefitted through data-driven decision-making.

2. Dataset

We picked the dataset from Kaggle([Hotel booking demand | Kaggle](#))

The dataset is called "Hotel Booking Demand". It contains information about hotel bookings, including the type of hotel, the number of guests, the duration of stay, and whether the booking was canceled or not. The dataset is meant to be used for predictive modeling and analysis and is available for download in several formats, including CSV and SQL.

There are 32 columns in the dataset Majority of which is in integer or string format. The data is originally from the article [Hotel booking demand datasets - ScienceDirect](#), written by Nuno Antonio, Ana Almeida, and Luis Nunes for Data in Brief, Volume 22, February 2019.



	hotel	is_canceled	lead_time	arrival_date_year	arrival_date_month	arrival_date_week_number	arrival_date_day_of_month	stays_in_weekend_nights	stays_in_week_night
0	Resort Hotel	0	342	2015	July	27	1	0	
1	Resort Hotel	0	737	2015	July	27	1	0	
2	Resort Hotel	0	7	2015	July	27	1	0	
3	Resort Hotel	0	13	2015	July	27	1	0	
4	Resort Hotel	0	14	2015	July	27	1	0	

Fig 1. Sample of the data

3. Cancellation Prediction using kNN Algorithm

The purpose of this prediction:

The reason for predicting cancellations from the booking data, from a hotel's point of view, is to know what bookings to cancellations ratio they can expect. This gives them some insight into

whether there is a need for change in management or services. If there are a large number of cancellations then we can perform further data training to analyze the reasons for such high cancellation prediction.

The hotels can also manage their resources better if they have an idea about the number of cancellations that they can expect.

We are performing this prediction based on different attributes of the data such as the type of user that did the booking, the date of booking, the deposit, previous cancellations by the users, booked room type, etc.

Description of the algorithm and why we chose it:

K-nearest neighbors (KNN) is a type of supervised learning algorithm used for both regression and classification. KNN tries to predict the correct class for the test data by calculating the distance between the test data and all the training points. Then select the K number of points which is closest to the test data. The KNN algorithm calculates the probability of the test data belonging to the classes of 'K' training data and the class that holds the highest probability will be selected. In the case of regression, the value is the mean of the 'K' selected training points. KNN is also flexible in terms of the choice of distance metric and the number of nearest neighbors to consider. Moreover, it can be easily implemented and scaled for large datasets.

Implementation Steps:

1. Load the dataset: Load the dataset you want to work with into a pandas dataframe or a NumPy array.
2. We are encoding the data which will contain all numerical values, with each categorical feature encoded as a sequence of integers allowing it to be used in machine learning algorithms

```
encoder=LabelEncoder()
dict_df={}
for feature in df.columns:
    dict_df[feature]=encoder.fit_transform(df[feature])
#converting back the encoded feature into dataframe
df=pd.DataFrame(dict_df)
```

3. Feature scaling: Scale the features of the dataset to ensure that they are on the same scale. This can be done using normalization or standardization. We used MinMaxScaler method which standardize the range of features in the input dataset

```
scaler=MinMaxScaler()
df=pd.DataFrame(scaler.fit_transform(df),columns=df.columns)
```

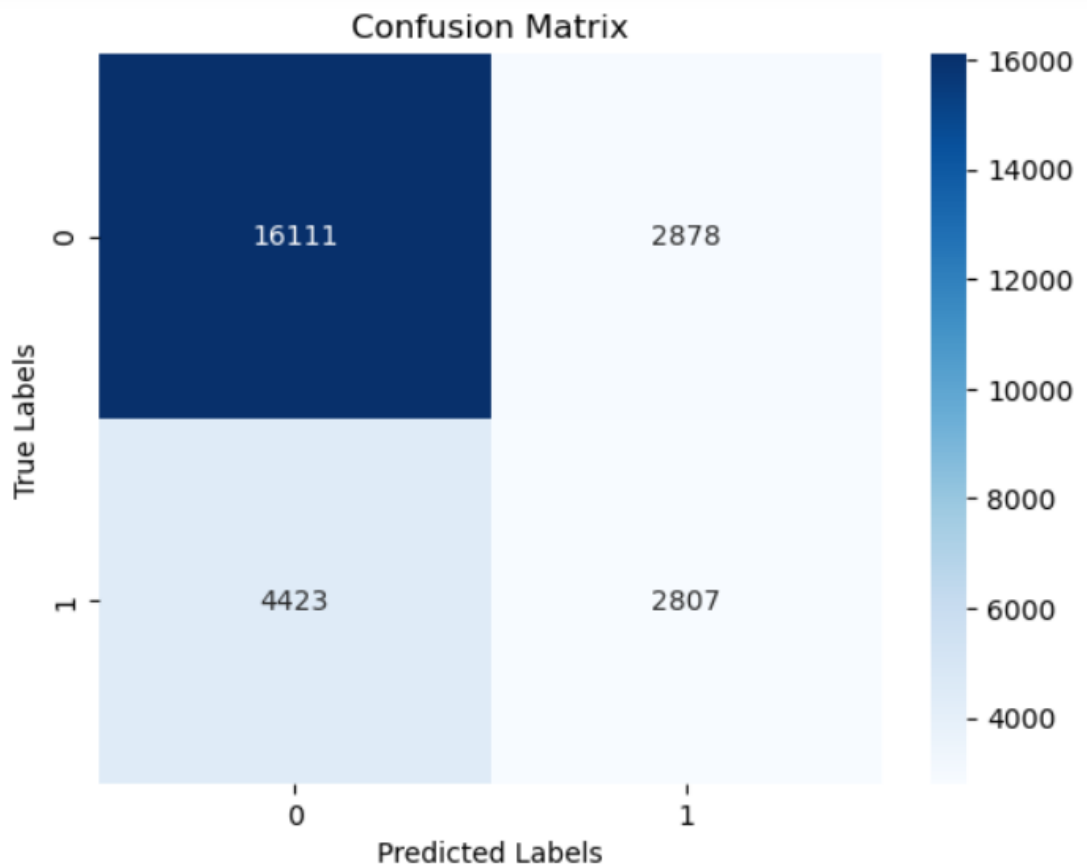
4. Split the dataset: Split the dataset into a training set and a testing set. This can be done using the train_test_split function in scikit-learn. We are using 30% as test data and 70% as training data.

```
y=df_new['is_canceled']
x = df_new.drop('is_canceled', axis = 1)
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.30)
```

5. Train the model and Predict target variable: Create an instance of the KNN algorithm with the desired value of K and fit it to the training set. Use the predict method of the KNN model to predict the target variable of the test set.

```
KNN=KNeighborsClassifier()  
KNN.fit(X_train,y_train)  
y_pred=KNN.predict(X_test)
```

6. Visualization of the model: Confusion matrix for the KNN algorithm.



	precision	recall	f1-score	support
0.0	0.78	0.85	0.82	18989
1.0	0.49	0.39	0.43	7230
accuracy			0.72	26219
macro avg	0.64	0.62	0.62	26219
weighted avg	0.70	0.72	0.71	26219

7. we can see that the model achieved an overall accuracy of 0.72, which means that it correctly predicted the outcome for 72% of the test set instances. Overall, these results

suggest that the K- Nearest Neighbors algorithm applied to the given data has reasonable performance in predicting canceled bookings

4. Booking waitlist prediction using Logistic Regression

The purpose of this prediction:

Through waitlist prediction, we want to predict the popularity of a hotel. Based on the past trends of the data like at what time of the year does a particular type of hotel have a big waitlist? This will give the hotels some idea as to what they can expect from the coming months. Or If a hotel's waitlist numbers start going down, then its relevance in the market would also be decreasing, so there might be a requirement for some change.

This prediction combined with our last prediction about cancellations can give hotels a lot of information about their standing in the market, and the scope for improvement. Also, it will help them manage their resources better.

Description of the algorithm and why we chose it:

Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Unlike linear regression which outputs continuous number values, logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes.

Logistic regression can capture nonlinear relationships between the predictor variables and the outcome variable, through the use of higher-order terms. Logistic regression is computationally efficient and can be applied to large datasets.

Implementation Steps:

1. Load the dataset: Load the dataset you want to work with into a pandas dataframe or a NumPy array.
2. We are encoding the data which will contain all numerical values, with each categorical feature encoded as a sequence of integers allowing it to be used in machine learning algorithms

```
encoder=LabelEncoder()
dict_df_clean={}
for feature in df_cleaned.columns:
    dict_df_clean[feature]=encoder.fit_transform(df_cleaned[feature])
#converting back the encoded feature into dataframe
df_cleaned=pd.DataFrame(dict_df_clean)
```

3. Split the dataset: Split the dataset into a training set and a testing set. This can be done using the `train_test_split` function in scikit-learn.

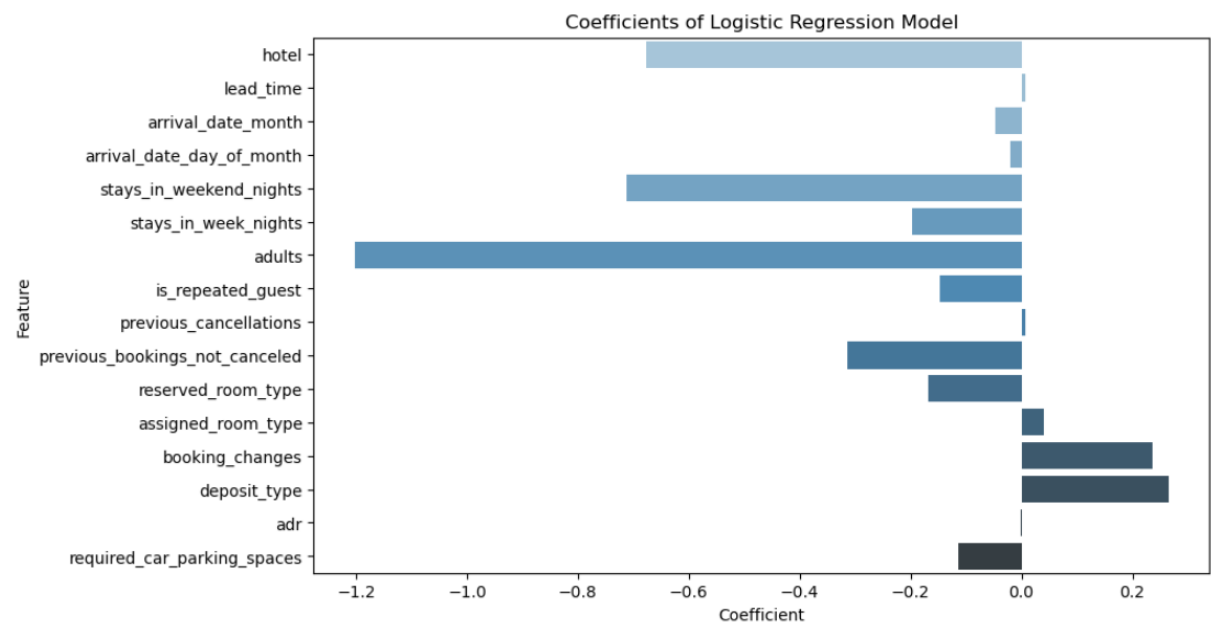
```
y=df_logistic['days_in_waiting_list']
x = df_logistic.drop('days_in_waiting_list', axis = 1)
x_log_train, X_test, y_log_train, y_test = train_test_split(x, y, test_size = 0.30)
```

4. Train the model and Predict the target variable: Create an instance of the Logistic Regression Model and fit it to the training set. Use the predict method of the Logistic Regression model to predict the target variable of the test set

```
lr_model = LogisticRegression()

# Train the logistic regression model
lr_model.fit(x_log_train, y_log_train)
y_pred=lr_model.predict(X_test)
# Get the coefficients and feature names
coef = lr_model.coef_[0]
features = x.columns
```

5. Visualization of the model: The y-axis shows the features used in the model, and the x-axis shows the coefficient values. The color of the bars represents the magnitude of the coefficients, with darker bars indicating larger coefficients. The plot can be used to determine which features are most important in predicting the target variable.



	precision	recall	f1-score	support
0	0.99	1.00	0.99	25971
1	0.06	0.00	0.01	248
accuracy			0.99	26219
macro avg	0.53	0.50	0.50	26219
weighted avg	0.98	0.99	0.99	26219

6. We can see that the number of previous cancellations has the strongest relationship with the outcome variable, followed by the type of deposit made, assigned room type and the lead time. This information can be used to inform decisions about how to reduce waiting times for future bookings

5. Customer loyalty prediction using Naive Bayes

The purpose of this prediction:

Through this prediction, we want to find out how loyal of a customer base a hotel has. To gain this knowledge, we are training the model on the data attributes like previous cancellations, if the booking was on the waitlist or not, type of customer, date of the booking, number of children and babies, changes made in booking, deposit, etc.

Then we are predicting whether the booking was canceled after being on the waitlist. We are trying to find out how many customers will be willing to be on the waitlist to stay in the hotel.

Description of the algorithm and why we chose it:

While performing classification tasks, the Naive Bayes method makes the simple assumption that features are independent of one another given the class label. It computes the posterior probability of each class given a fresh data point by estimating probability distributions of the features. Although Naive Bayes is effective with tiny datasets, if the independence assumption is broken, performance may not be as good as it may be.

We chose Naive Bayes for the classification task because of its simplicity, efficiency, and ability to handle small datasets. It is less prone to overfitting compared to more complex algorithms and can handle both binary and multi-class classification problems. Moreover, Naive Bayes can provide valuable insights into the relationships between features and the target variable, and non-experts can easily interpret it.

Implementation Steps:

1. Load the dataset: Load the dataset you want to work with into a pandas dataframe or a NumPy array.
2. We are using the same dataset which we performed encoding and the feature scaling using standardization
3. Split the dataset: Split the dataset into a training set and a testing set. This can be done using the `train_test_split` function in scikit-learn.

```
y=df_nb['is_canceled']  
x = df_nb.drop('is_canceled', axis = 1)  
x_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.30)
```

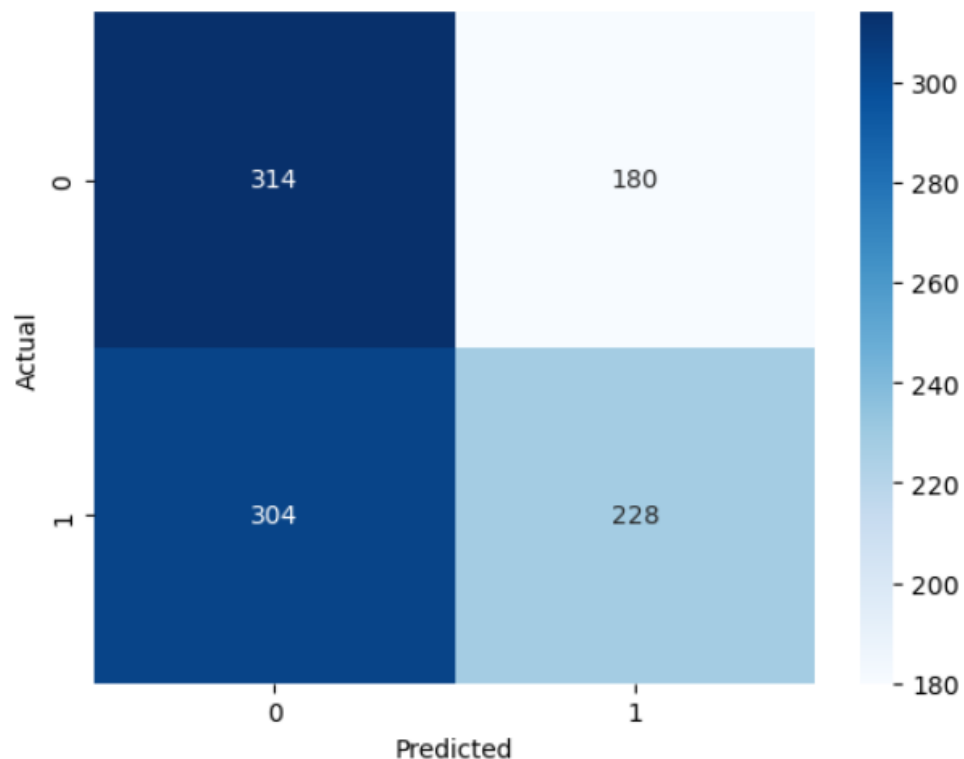
4. Train the model and Predict the target variable: Create an instance of the MultinomialNB classifier and fit it to the training set. Use the predict method of the Naive Bayes model to predict the target variable of the test set.

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score

# assuming xtrain and ytrain are your training data and labels, respectively
nb_classifier = MultinomialNB()
nb_classifier.fit(x_train, y_train)

# assuming xtest is your test data
y_pred = nb_classifier.predict(X_test)
```

5. Visualization of the model: Confusion matrix for the Naive Bayes algorithm



	precision	recall	f1-score	support
0	0.51	0.64	0.56	494
1	0.56	0.43	0.49	532
accuracy			0.53	1026
macro avg	0.53	0.53	0.52	1026
weighted avg	0.53	0.53	0.52	1026

6. From the classification report, the accuracy of the algorithm in predicting whether a booking was canceled after being on the waitlist is 0.53 or 53%.

6. Target customer demographic prediction using Decision Tree

The purpose of this prediction:

Here we are trying to predict what kind of demographic a hotel can expect in the coming months. Training the data on the customer type, booking, cancellation, booking date, waitlist, etc. From that training, we will try to predict if families (adults + children + babies) will be visiting the hotels in the future. To achieve this we will be creating an additional column 'Family' and will be predicting the value of that.

This type of prediction will tell the hotels their standing in different types of potential customer base. It will also help them manage their resources better and prepare better for the predicted demographic.

Description of the algorithm and why we chose it:

A tree-like model called a decision tree is employed for classification and regression problems. Once a stopping requirement is satisfied, it divides the feature space recursively according to the most informative feature at each node. Decision trees can overfit if they are not regularized, yet they are interpretable and can handle both continuous and categorical data. Performance can be increased by using ensemble techniques like Gradient Boosted Trees and Random Forest.

We chose a decision tree for classification and regression task because it is interpretable, can handle both categorical and continuous data, and is easy to use. It also provides insights into the most important features for the task. Decision trees can be regularized to prevent overfitting, and ensemble methods like Random Forest and Gradient Boosted Trees can further enhance the model's performance.

Implementation Steps:

1. Load the dataset: Load the dataset you want to work with into a pandas dataframe or a NumPy array.
2. We are using the same dataset which we performed encoding and the feature scaling using standardization
3. We are creating family based on the count in adults and children value

```
# add new "family" column based on conditions
df_dt['family'] = ((df_dt['babies'] + df_dt['num_children'] > 0) & (df_dt['adults'] > 0)).astype(int)
```

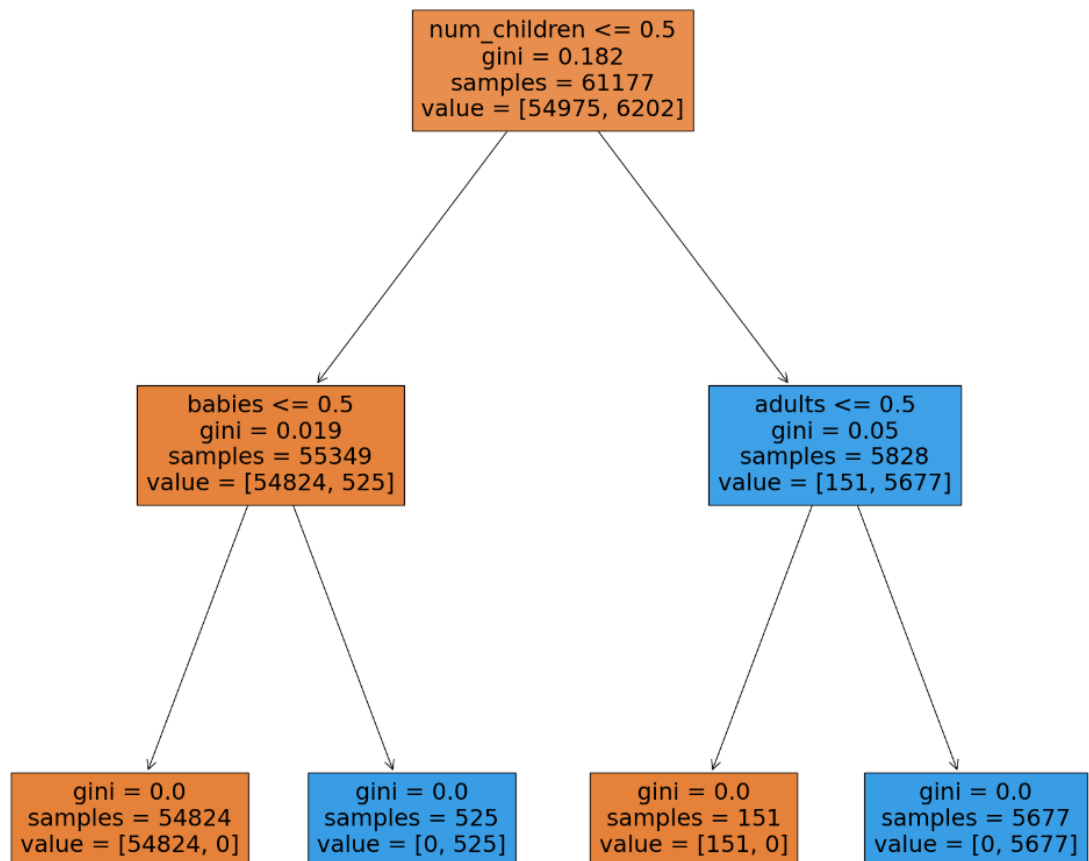

4. Split the dataset: Split the dataset into a training set and a testing set. This can be done using the `train_test_split` function in scikit-learn.

```
y=df_dt['family']
x = df_dt.drop('family', axis = 1)
x_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.30)
```

5. Train the model and Predict the target variable :Create an instance of the `DecisionTreeClassifier` class and fit it to the training set. Use the `predict` method of the `DecisionTreeClassifier` model to predict the target variable of the test set.

```
tree=DecisionTreeClassifier()
tree.fit(x_train,y_train)
y_pred = tree.predict(X_test)
```

6. Visualization of the model: We are visualizing using the `treeplot`.



7. From the tree plot the colored blue blocks indicate the bookings were classified mostly and these are the most target customers.

7. Popularity based on chronology using Adaboost

The purpose of this prediction:

Here we tried to predict the popularity of the hotels based on time (years, months, weeks). We trained the data on columns like arrival date, arrival month, year, week, etc. and we predicted the type of hotel the booking would be made in. This gives insight on what type of hotels would be more popular on what times of the year.

This will help the hotels prepare for the seasons and make marketing schemes for particular times of the year. They can work on the services they provide and tweak them according to the time of the year and predicted outcome.

Description of the algorithm and why we chose it:

The ensemble learning technique AdaBoost is employed for categorization tasks. By giving samples that were incorrectly classified more weight and training a new weak classifier on the revised dataset, it combines several weak classifiers into a strong classifier. AdaBoost is utilized in practical applications like face identification and object recognition because it can handle binary and multi-class classification problems and is less prone to overfitting. However, careful hyperparameter tweaking is necessary because it might be susceptible to noisy data and outliers.

We chose AdaBoost for the classification task because of its ability to enhance the performance of weak classifiers and handle binary and multi-class classification problems. It is less prone to overfitting compared to other complex models, and is widely used in real-world applications such as face detection and object recognition. AdaBoost is also easy to implement and can be fine-tuned for optimal performance, making it a practical choice for many machine learning problems.

Implementation Steps:

1. Load the dataset: Load the dataset you want to work with into a pandas dataframe or a NumPy array.
2. We are using the same dataset which we performed encoding and the feature scaling using standardization
3. Split the dataset: Split the dataset into a training set and a testing set. This can be done using the `train_test_split` function in scikit-learn.

```
: y=df_dt['hotel']  
x = df_dt.drop('hotel', axis = 1)  
x_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.30)
```

4. Train the model and Predict the target variable: Create an instance of the `DecisionTreeClassifier` class with `AdaBoostClassifier` and fit it to the training set. Use the `predict` method of the `AdaBoostClassifier` model to predict the target variable of the

test set.

```
# Create a Decision Tree Classifier as the base estimator
base_estimator = DecisionTreeClassifier(max_depth=1)

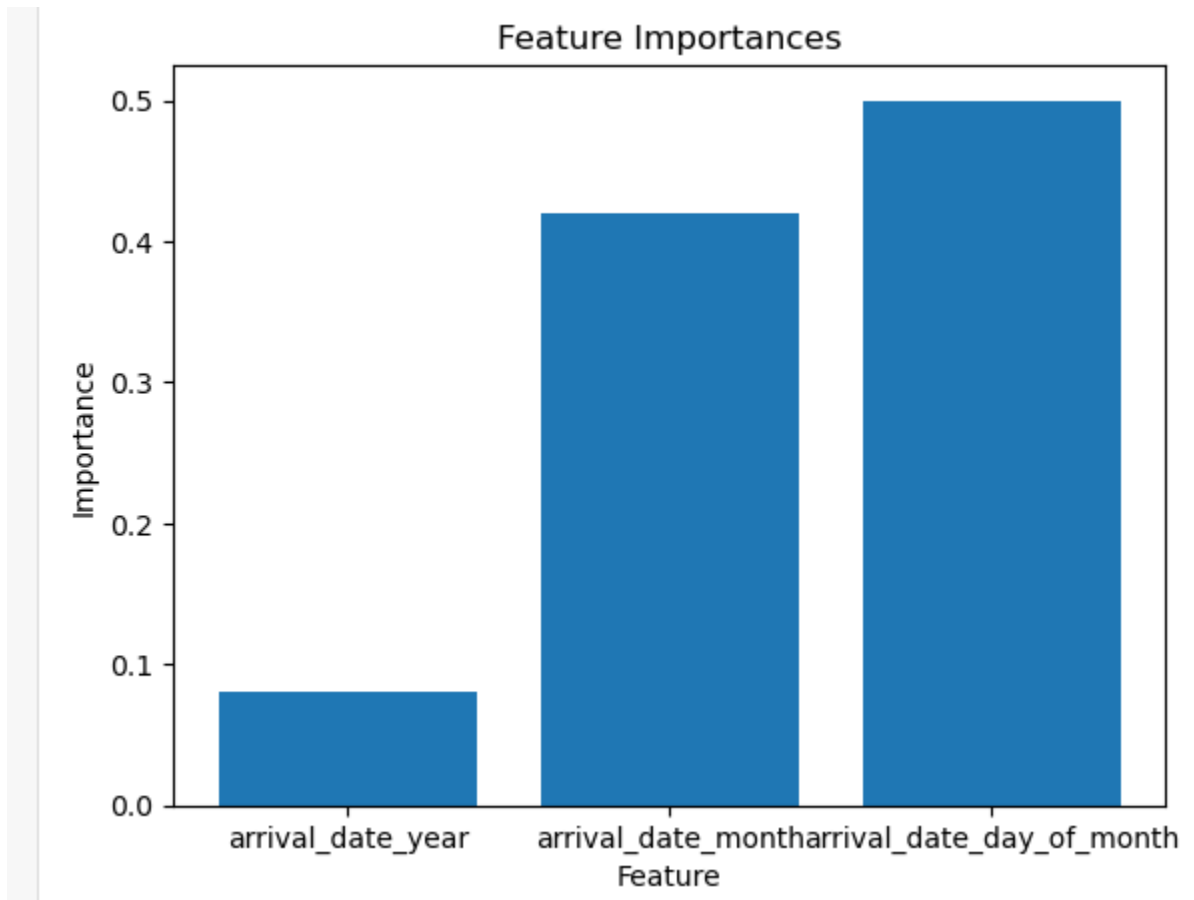
# Create an AdaBoost classifier with 50 estimators
ada = AdaBoostClassifier(base_estimator=base_estimator, n_estimators=50, random_state=42)

# Fit the AdaBoost classifier to the training data
ada.fit(x_train, y_train)

# Predict labels for the testing data
y_pred = ada.predict(X_test)

# Calculate evaluation metrics for the AdaBoost classifier
accuracy = accuracy_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
```

5. Visualization of the model: We are plotting a bar graph on x-axis we have the feature and on y-axis we have importance.



6. From the graph we can see that the arrival day of the month has the most importance so based on the day the hotel booking is more popular and also based on the month.

8. References

- Kaggle dataset:
https://www.kaggle.com/datasets/jessemostipak/hotel-booking-demand?resource=download&select=hotel_bookings.csv
- Article on Linear Regression:
<https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know-about-linear-regression/>
- Kaggle reference on further details: [Heart Attack - EDA + Prediction \(90% accuracy\) | Kaggle](#)