MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY JAIPUR

MLDA Final Project

To forecast whether the client will uphold their booking or terminate it

Submitted by
Harish (2022PPD5236)
Rajendra Jangid (2022PPD5232)
Om Singh (2022PES5272)

Supervised by
Prof. Dr. Rajesh Kumar
Department of Electrical Engineering

Abstract

The online hotel reservation channels have dramatically changed booking possibilities and customers' behavior. A significant number of hotel reservations are called-off due to cancellations or no-shows. The typical reasons for cancellations include change of plans, scheduling conflicts, etc. This is often made easier by the option to do so free of charge or preferably at a low cost which is beneficial to hotel guests but it is a less desirable and possibly revenue-diminishing factor for hotels to deal with.

This study involved the analysis of data using various regression and classification-based techniques implemented with PYTHON's sklearn libraries. The resulting accuracy was approximately 90%.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The online hotel reservation channels have dramatically changed booking possibilities and customers' behavior. A significant number of hotel reservations are called-off due to cancellations or no-shows. The typical reasons for cancellations include change of plans, scheduling conflicts, etc. This is often made easier by the option to do so free of charge or preferably at a low cost which is beneficial to hotel guests but it is a less desirable and possibly revenue-diminishing factor for hotels to deal with.

1.1 Problem statement

Can we predict if the costumer is going to honor the reservation or cancel it.

1.2 Aim and objectives

1. Classify whether person keep up its reservation after doing it.
2. Visualization the working of different ML Algorithms.
3. Evaluating the results.

1.3 Scope and limitations

Future scopes include:

- Using the Bayesian approach through one condition we have more certainty toward decision-making.
- Visualization of working of all ML algorithms.

Chapter 2

Methodology

2.1 Pre-Processing

2.1.1 Taking Care of Missing Values

There is no missing value in given dataset and information about the data is mention below.

TABLE 2.1.1

```
Data columns (total 19 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Booking_ID                            36275 non-null  object
 1   no_of_adults                          36275 non-null  int64
 2   no_of_children                        36275 non-null  int64
 3   no_of_weekend_nights                  36275 non-null  int64
 4   no_of_week_nights                     36275 non-null  int64
 5   type_of_meal_plan                     36275 non-null  object
 6   required_car_parking_space            36275 non-null  int64
 7   room_type_reserved                    36275 non-null  object
 8   lead_time                             36275 non-null  int64
 9   arrival_year                          36275 non-null  int64
 10  arrival_month                         36275 non-null  int64
 11  arrival_date                          36275 non-null  int64
 12  market_segment_type                   36275 non-null  object
 13  repeated_guest                        36275 non-null  int64
 14  no_of_previous_cancellations          36275 non-null  int64
 15  no_of_previous_bookings_not_canceled  36275 non-null  int64
 16  avg_price_per_room                    36275 non-null  float64
 17  no_of_special_requests                36275 non-null  int64
 18  booking_status                        36275 non-null  object
dtypes: float64(1), int64(13), object(5)
```

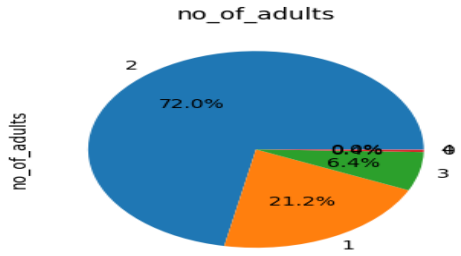
And unique values in each column are:

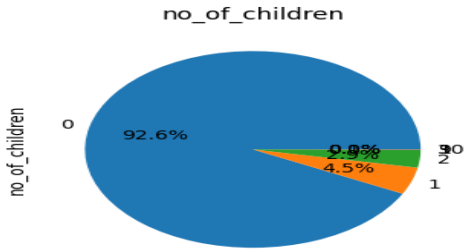
TABLE 2.1.2

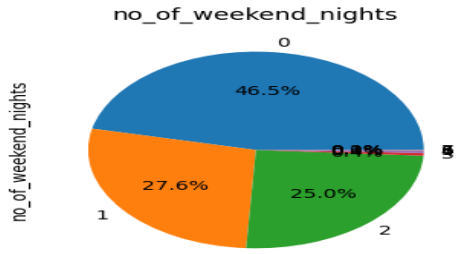
	unique value count
Booking_ID	36275
no_of_adults	5
no_of_children	6
no_of_weekend_nights	8
no_of_week_nights	18
type_of_meal_plan	4
required_car_parking_space	2
room_type_reserved	7
lead_time	352
arrival_year	2
arrival_month	12
arrival_date	31
market_segment_type	5
repeated_guest	2
no_of_previous_cancellations	9
no_of_previous_bookings_not_canceled	59
avg_price_per_room	3930
no_of_special_requests	6
booking_status	2

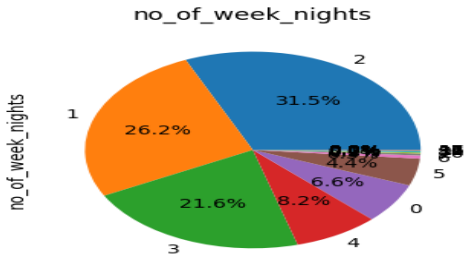
2.1.2 Data Visualization

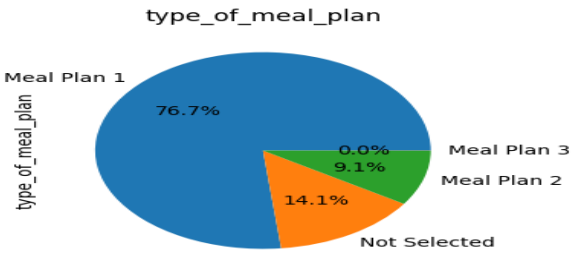
Frequency of every unique sample in each column can be analyzed through pie plot, for every column feature we can identify the proverbiality of occurrence of every unique value.

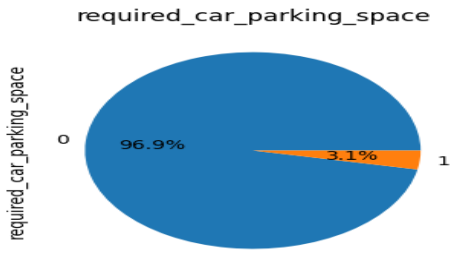

no_of_adults
no_of_adults
2
72.0%
6.4%
3
21.2%
1

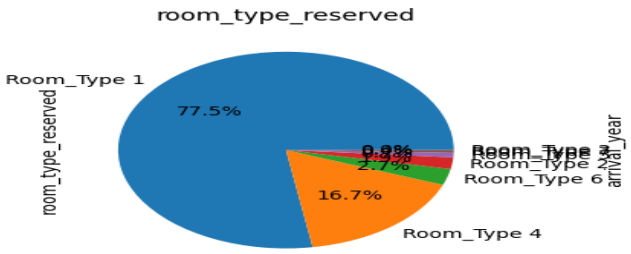

no_of_children
no_of_children
0
92.6%
4.5%
1

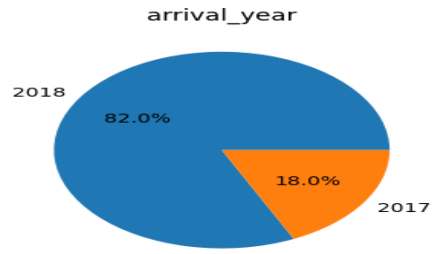

no_of_weekend_nights
no_of_weekend_nights
0
46.5%
27.6%
1
25.0%
2


no_of_week_nights
no_of_week_nights
2
31.5%
1
26.2%
6.6%
8.2%
21.6%
3
4
0


type_of_meal_plan
type_of_meal_plan
Meal Plan 1
76.7%
0.0%
Meal Plan 3
9.1%
Meal Plan 2
14.1%
Not Selected


required_car_parking_space
required_car_parking_space
0
96.9%
3.1%
1


room_type_reserved
room_type_reserved
Room_Type 1
77.5%
Room_Type 6
16.7%
Room_Type 4


arrival_year
arrival_year
2018
82.0%
18.0%
2017

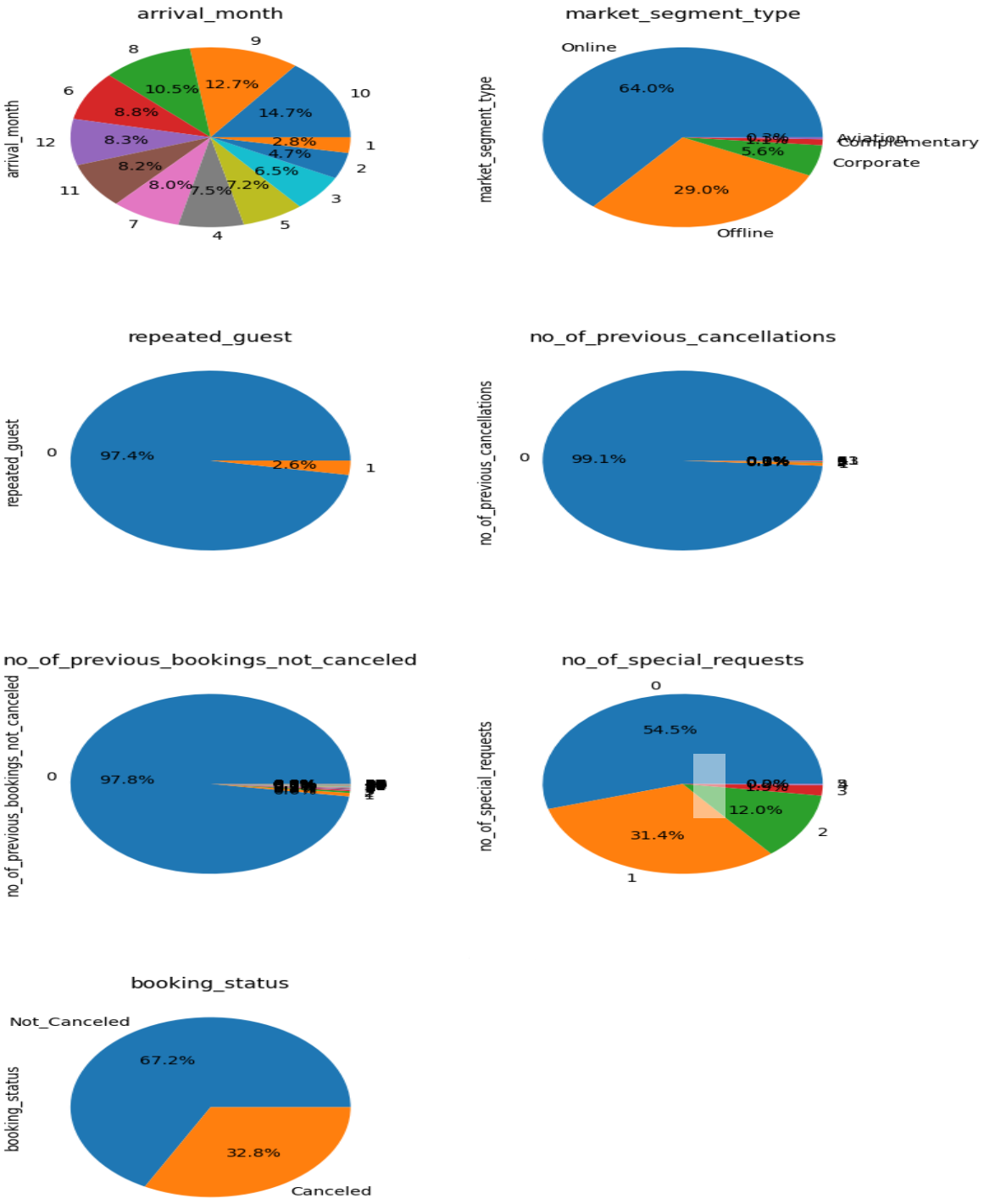


Fig. 2.1.2 Data Arrangement

2.1.3 Outlier Removal

Science most of the columns are categorical variables except leading time and average price, we don't need to be concerned about other columns.

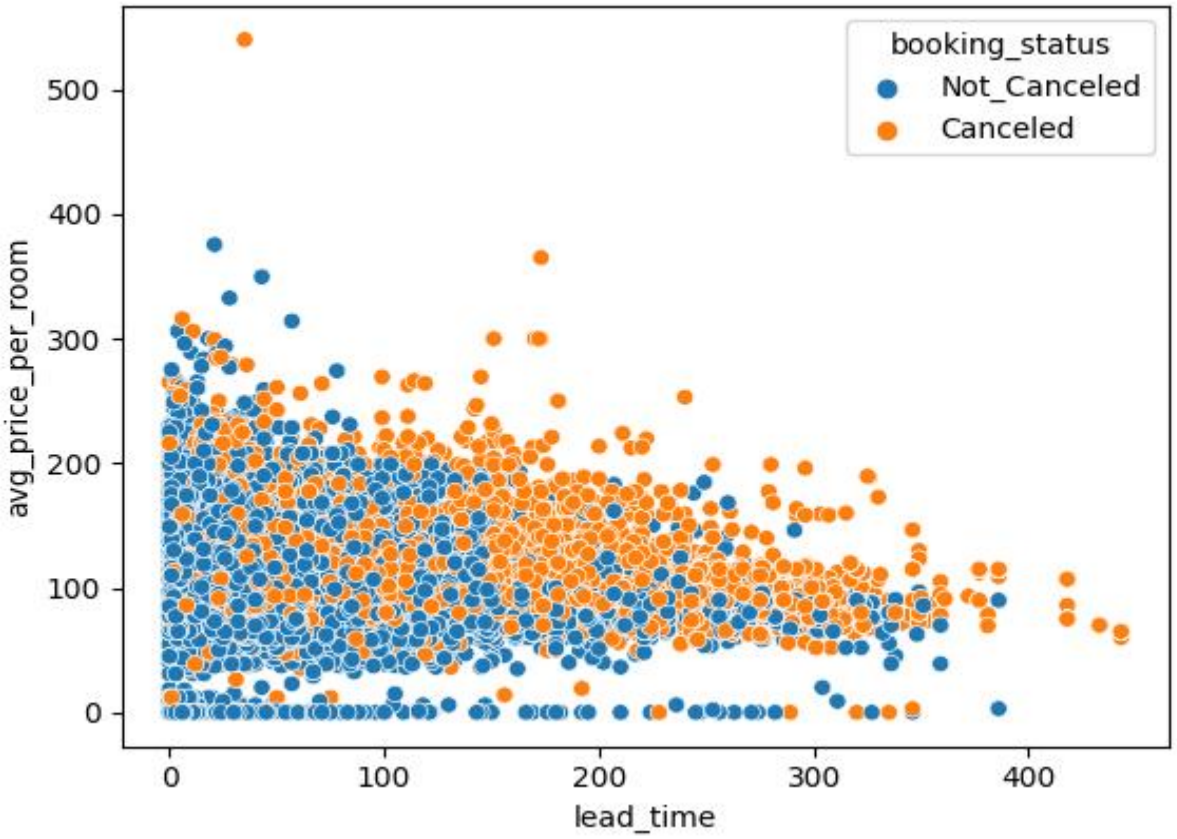


Fig. 2.1.3 Average price per room vs lead time

We can see that in leading time there is no outlier but for average price we removed outlier manually by applying condition such that remove all the entries above 400RS price.

df = df[df['avg_price_per_room'] <= 400]

2.1.4 Data Interpterion

The behavior of data can be studied using diverse methods. Mean, RMS, skewness, and kurtosis are some of the techniques employed. These approaches provide insights into different aspects

of data behavior. Analyzing data using such methods helps in understanding its characteristics comprehensively.

Feature	Formula
Mean value	$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$
Standard deviation	$\sigma = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(x_i - \overline{x})^2}$
Kurtosis	$\mathrm{K} = \frac{1}{N}\sum_{i=1}^{N}\frac{(x_i-\overline{x})^4}{\sigma^4}$
Skewness	$\mathrm{Sk} = \frac{1}{N}\sum_{i=1}^{N}\frac{(x_i-\overline{x})^3}{\sigma^3}$
Root mean square	$\mathrm{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$
Crest factor	$\mathrm{Crf} = \frac{\max value}{\mathrm{RMS}}$
Peak to Peak value	$\mathrm{PPV} = \max value - \min value$

the skewness value for each numerical column in the DataFrame, with a value of 0 indicating no skew, negative values indicating left skew (long tail to the left), and positive values indicating right skew (long tail to the right).
In dataset we found the skewness for each column:

```
no_of_adults: -0.33
no_of_children: 3.94
no_of_weekend_nights: 0.74
no_of_week_nights: 1.60
type_of_meal_plan: 1.76
required_car_parking_space: 5.41
room_type_reserved: 3.09
lead_time: 1.29
arrival_year: -1.67
arrival_month: -0.35
market_segment_type: 1.24
```

```
repeated_guest: 6.00
no_of_previous_cancellations: 25.19
no_of_previous_bookings_not_canceled: 19.24
avg_price_per_room: 0.61
no_of_special_requests: 1.15
booking_status: 0.73
```

Table 2.1.4

the mean and standard deviation of the dataset is:

Columns	count	mean	std	min	25%	50%	75%	max
no_of_adults	36252	1.845167	0.518397	0	2	2	2	4
no_of_children	36252	0.103001	0.388285	0	0	0	0	2
no_of_weekend_nights	36252	0.810907	0.870672	0	0	1	2	7
no_of_week_nights	36252	2.20432	1.410812	0	1	2	3	17
type_of_meal_plan	36252	0.324092	0.634086	0	0	0	0	3
required_car_parking_space	36252	0.031005	0.173334	0	0	0	0	1
room_type_reserved	36252	0.335513	0.772092	0	0	0	0	6
lead_time	36252	85.252427	85.941121	0	17	57	126	443
arrival_year	36252	2017.82053	0.383747	2017	2018	2018	2018	2018
arrival_month	36252	7.423425	3.069705	1	5	8	10	12
arrival_date	36252	15.597981	8.74041	1	8	16	23	31
market_segment_type	36252	0.804342	0.646578	0	0	1	1	4
repeated_guest	36252	0.025654	0.158102	0	0	0	0	1
no_of_previous_cancellations	36252	0.023364	0.368448	0	0	0	0	13
no_of_previous_bookings_not_canceled	36252	0.153509	1.754723	0	0	0	0	58
avg_price_per_room	36252	103.386175	34.942027	0	80.3	99.4	120	375.5
no_of_special_requests	36252	0.619387	0.786115	0	0	0	1	5
booking_status	36252	0.327651	0.469363	0	0	0	1	1

2.1.5 Data analysis with respect to the Output

By examining the plots, it becomes evident that the probability of customer confirmation is higher during the months of October, November, and December. This conclusion can be drawn from a clear visual representation.

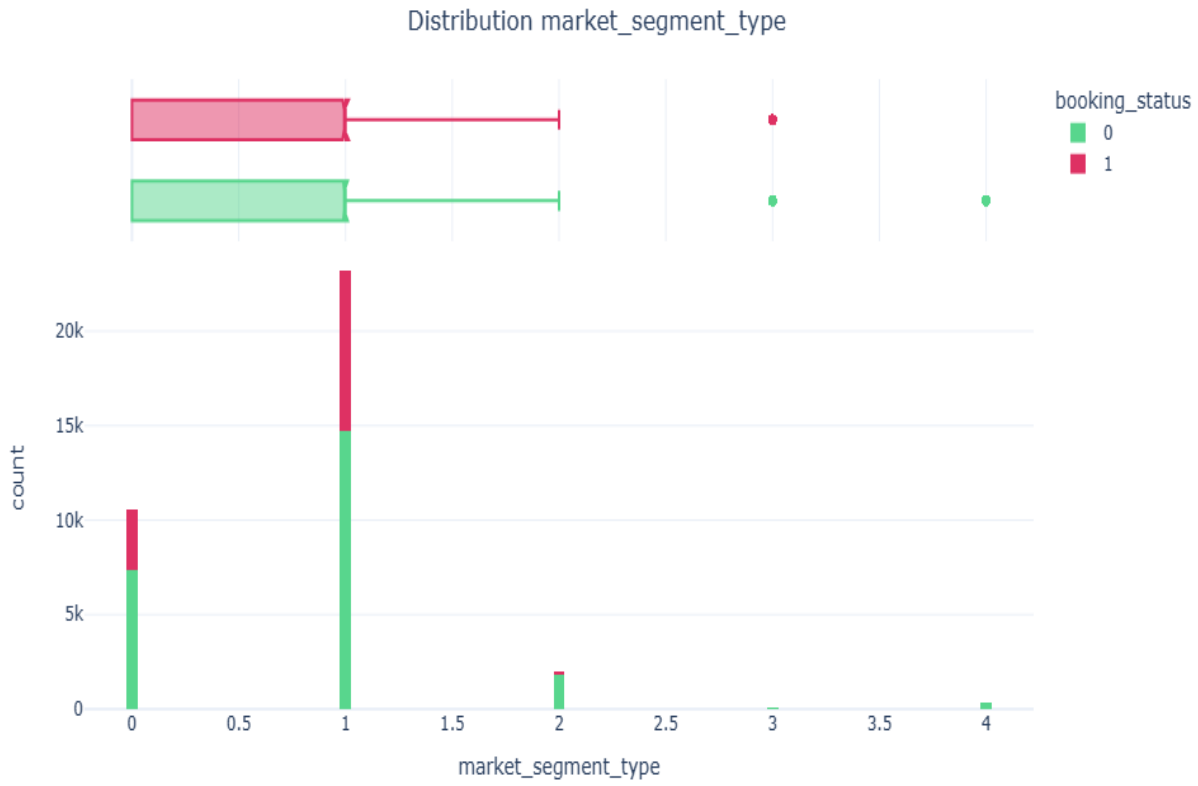


Fig 2.1.5(a) Distribution market segment type

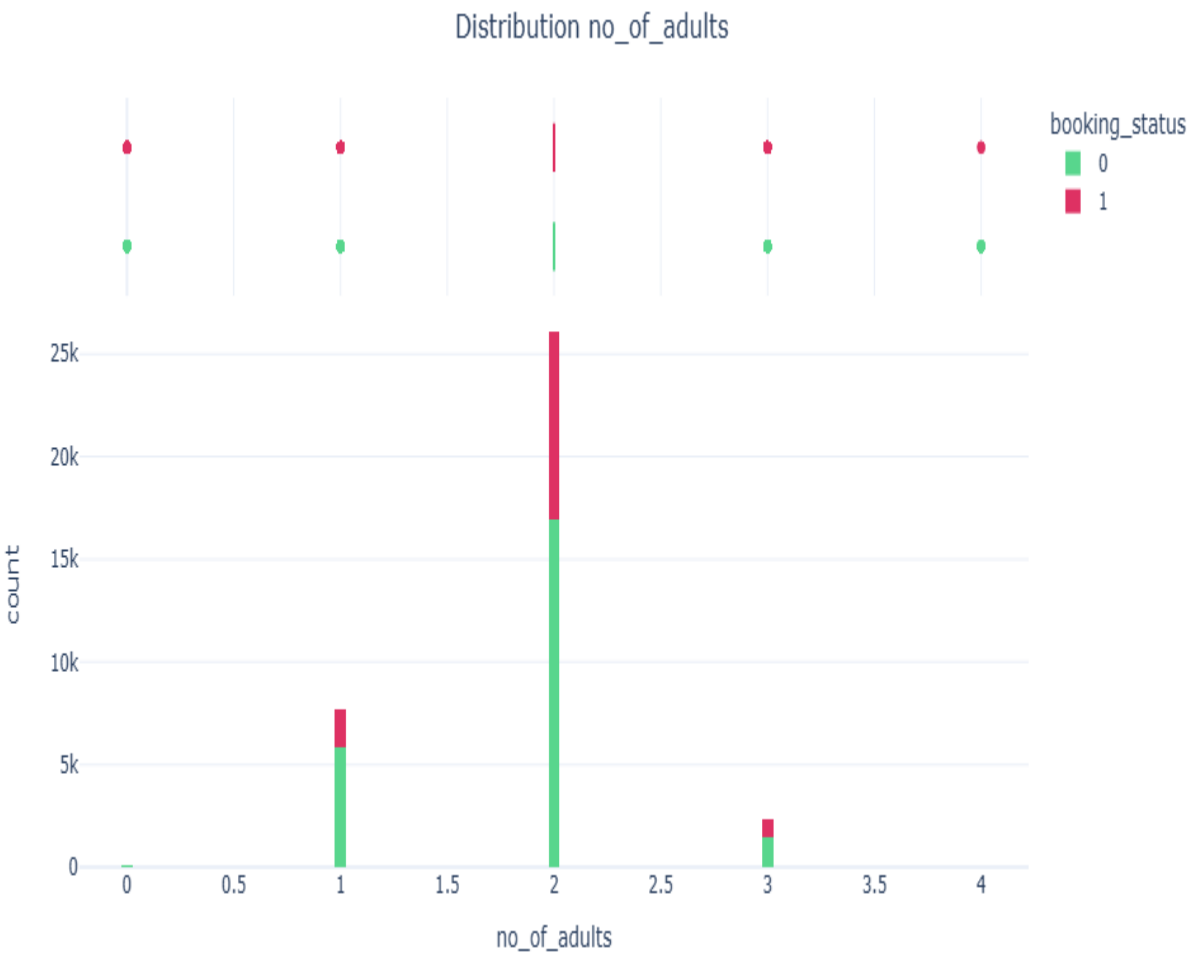


Fig 2.1.5(b) Distribution no of adults

Fig 2.1.5(c) Distribution of no of children

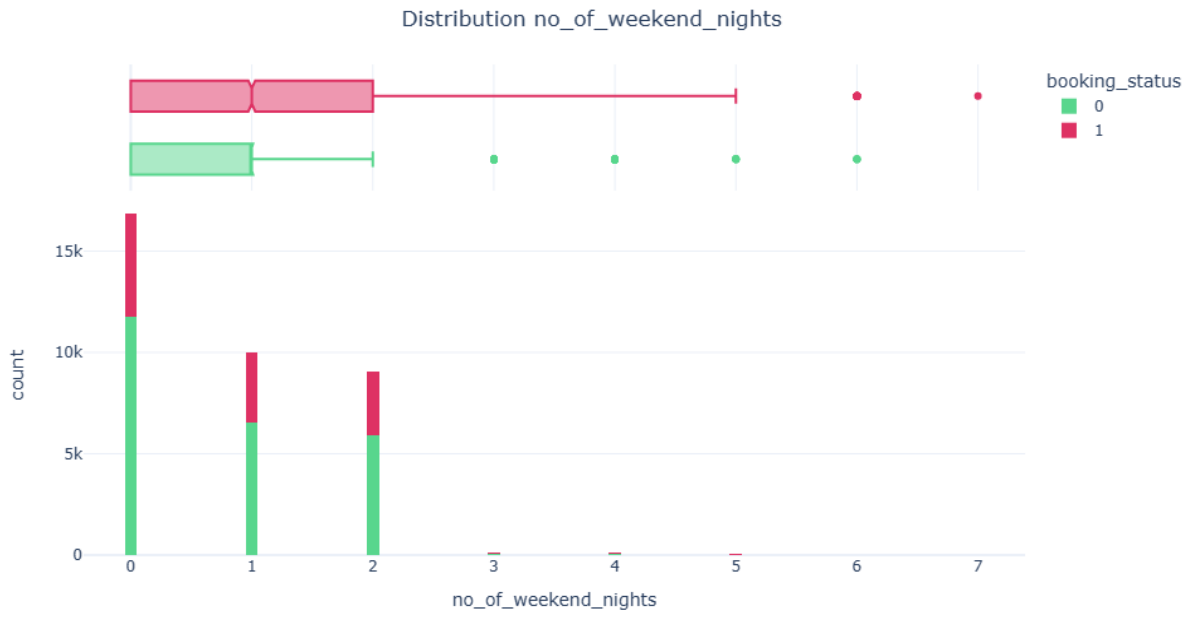


Fig 2.1.5(d). Distribution of no of weekend nights

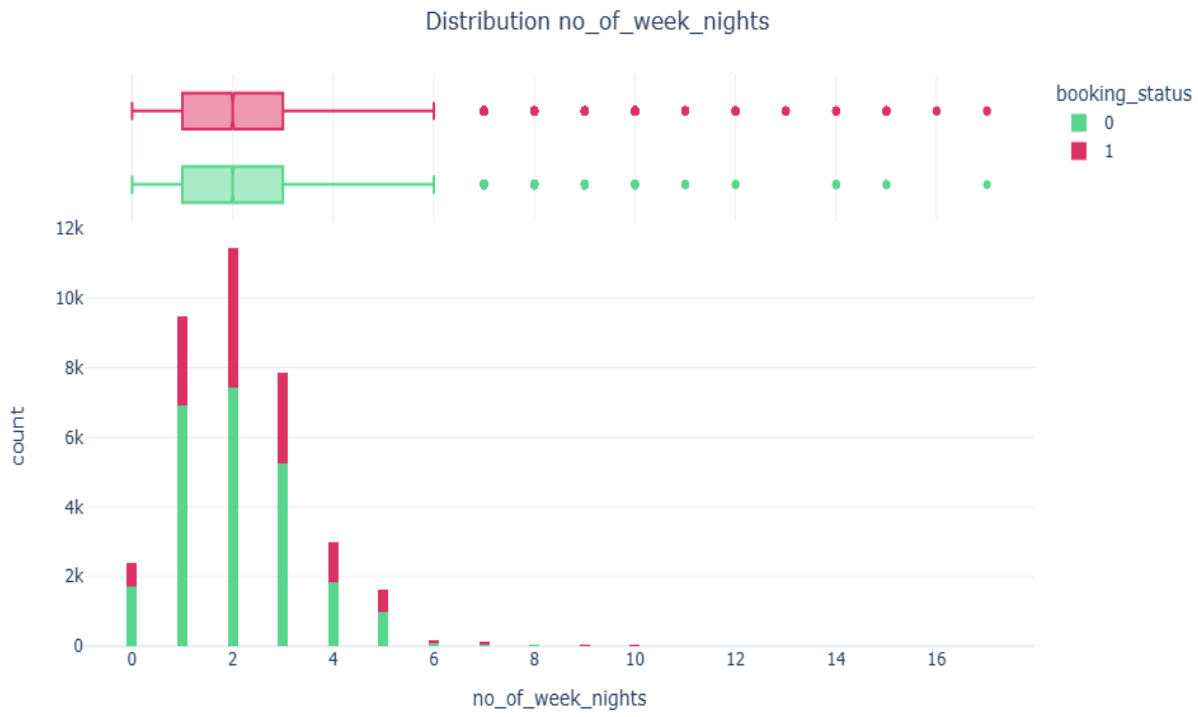


Fig.2.1.5(e) Distribution of no of week nights

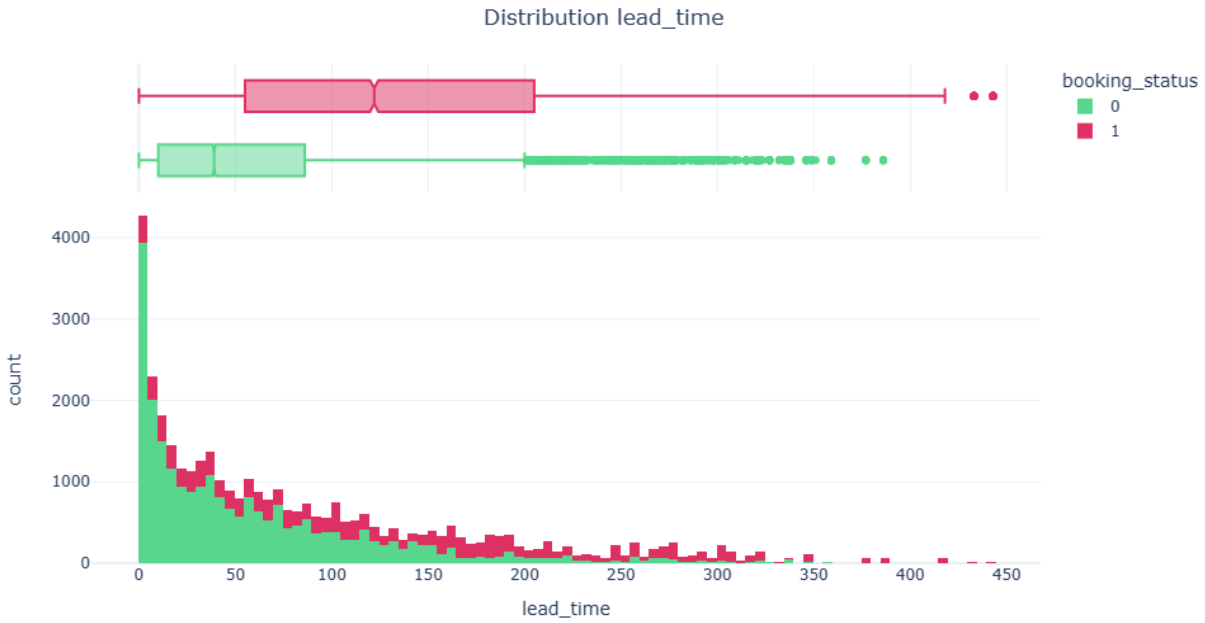


Fig 2.1.5(f) Distribution of lead time

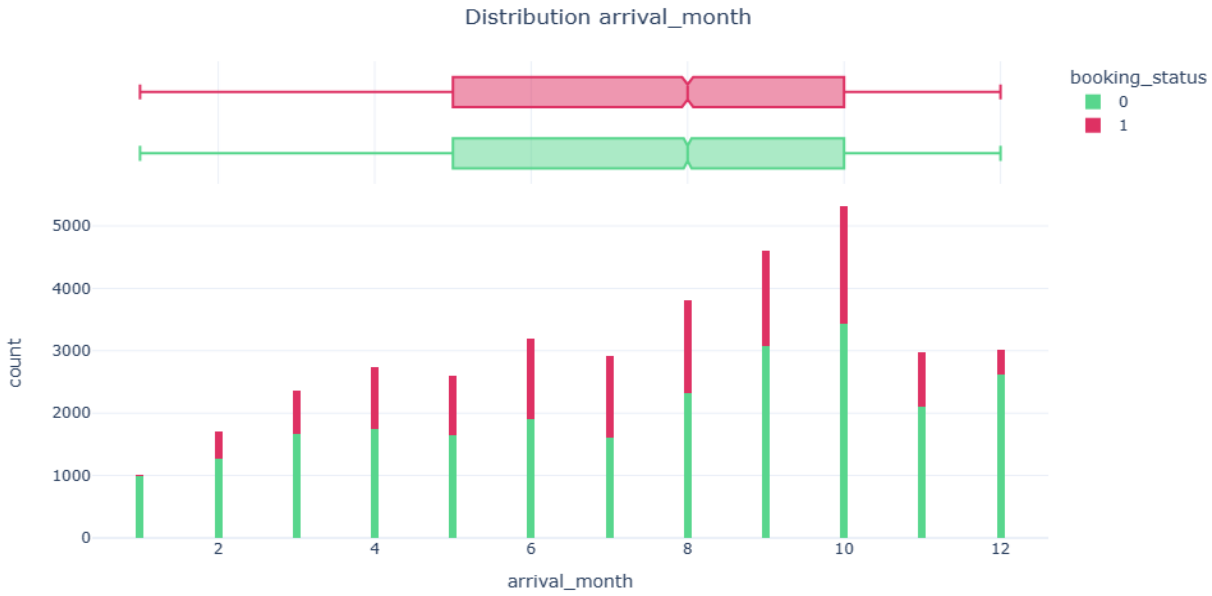


Fig.2.1.5(g) Distribution Arrival month

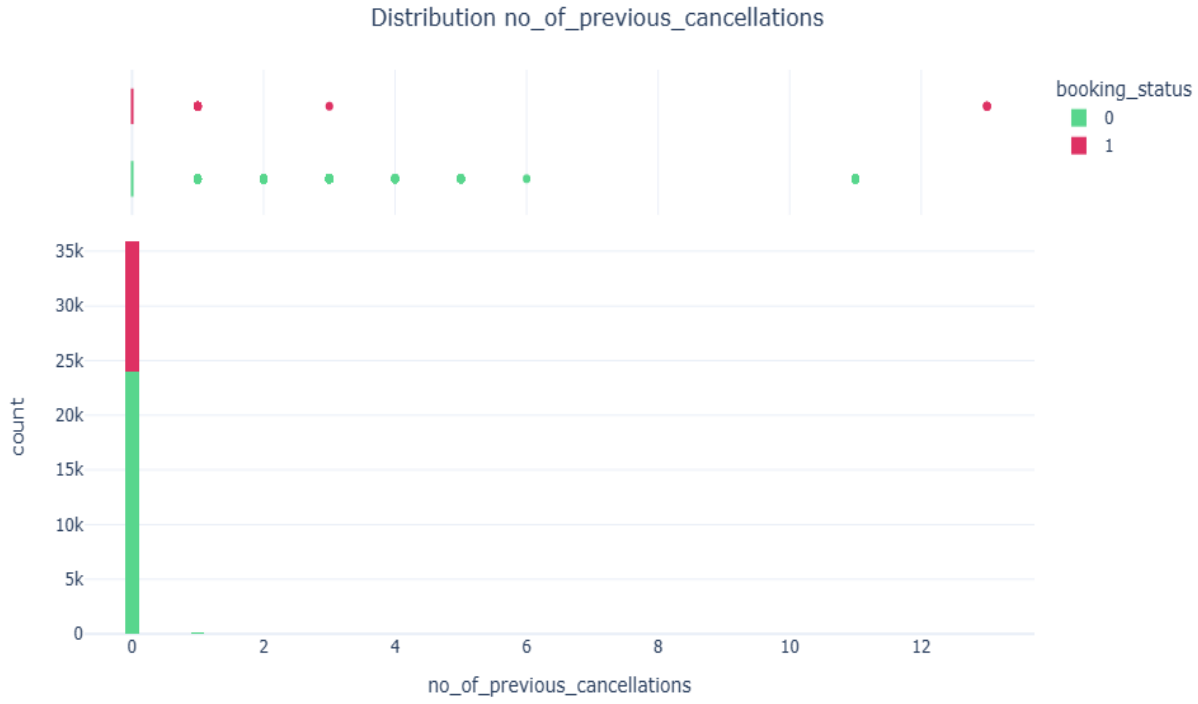


Fig. 2.1.5(h) Distriution of no of previous cancellations

Fig. 2.1.5(i) Distribution of previous booking not cancelled

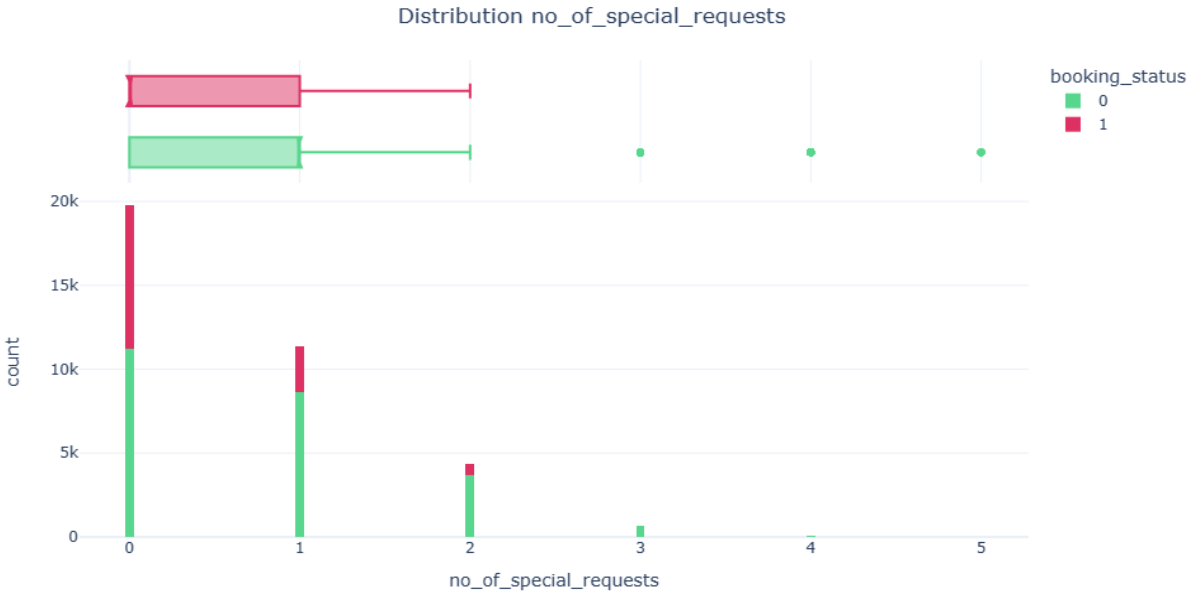


Fig. 2.1.5(j) Distribution of special requests

2.1.6 Scaling the data

scaling refers to the process of transforming the numerical values of features in a dataset to fall within a specific range or scale. This is often done to ensure that all features have a similar magnitude and do not disproportionately influence the learning algorithm during machine learning or statistical analysis. There are several different types of scaling techniques that can be used, including:

Min-max scaling: Also known as normalization, this technique scales the features to a specific range, typically [0, 1], by subtracting the minimum value of the feature from all data points and then dividing the result by the range (i.e., the difference between the maximum and minimum values of the feature). The formula for min-max scaling is:

X_scaled = (X - X_min) / (X_max - X_min)

where X is the original feature, X_scaled is the scaled feature, X_min is the minimum value of the feature, and X_max is the maximum value of the feature.

Z-score scaling: Also known as standardization, this technique scales the features to have zero mean and unit variance. It involves subtracting the mean of the feature from all data points and then dividing the result by the standard deviation of the feature. The formula for z-score scaling is:

X_scaled = (X - X_mean) / X_std

2.1.7 Correlation Matrix

In machine learning, a correlation matrix is a matrix that shows the correlation coefficients between pairs of variables in a dataset. Correlation is a statistical measure that quantifies the strength and direction of association between two variables. A correlation coefficient is a value between -1 and 1, where -1 indicates a perfect negative correlation (as one variable increases, the other decreases), 1 indicates a perfect positive correlation (as one variable increases, the other also increases), and 0 indicates no correlation (the variables are unrelated).

A correlation matrix is commonly used in machine learning for various purposes, including:

Feature selection: Correlation matrix can be used to identify highly correlated variables in a dataset. When two or more variables are highly correlated, they may provide redundant information to a machine learning algorithm, and using all of them as features in a model can result in multicollinearity. In such cases, one of the correlated variables may be removed to reduce redundancy and improve model performance.

Feature engineering: Correlation matrix can help identify relationships between variables that can be used to create new features. For example, if two variables are positively correlated, a new feature that represents their sum or average can be created as a potential predictor in a machine learning model.

Data exploration and visualization: A correlation matrix can be used to gain insights into the relationships between variables in a dataset. It can be visualized as a heatmap, where the color's intensity indicates the correlation's strength. Heatmaps can help identify patterns and trends in the data, such as clusters of variables with high positive or negative correlations.

It's important to note that correlation does not imply causation. The correlation only measures the strength and direction of association between variables but does not establish a cause-and-effect relationship. Correlation should be used in conjunction with other statistical techniques and domain knowledge to draw meaningful conclusions and make informed decisions in machine learning tasks.

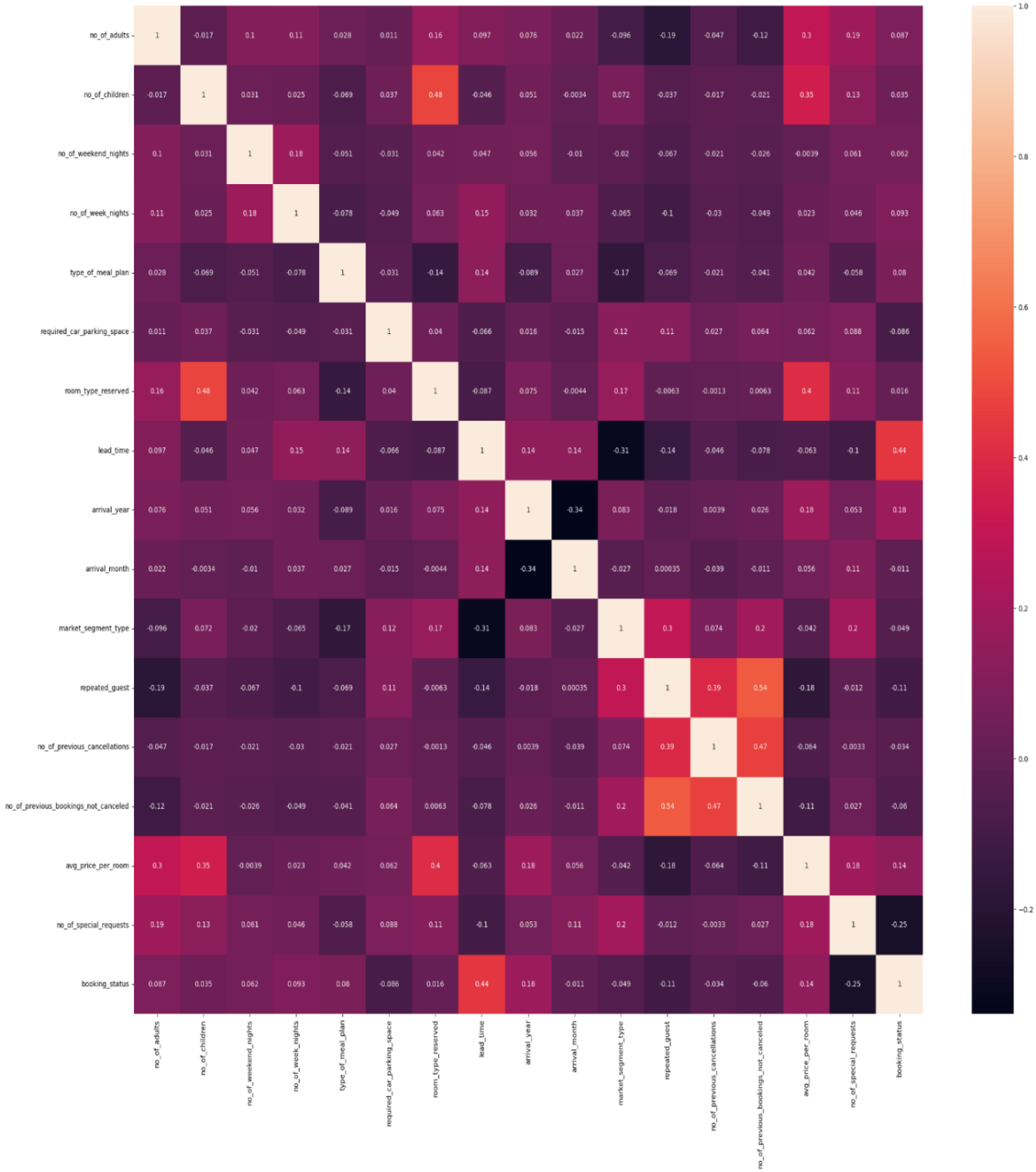


Fig. 2.1.7 Correlation Matrix

2.1.8 Encoding

Encoding is the process of converting data from one representation to another in a machine-readable format. In machine learning, encoding is often used to convert categorical or text data into numerical representations that can be easily processed by algorithms. Here are some common types of encoding techniques used in machine learning:

Label Encoding: Label encoding is a simple technique where each unique category in a categorical feature is assigned a unique numerical label. For example, if a feature has three categories: "red," "green," and "blue," label encoding might assign them the numerical labels 0, 1, and 2, respectively. However, it's important to note that label encoding introduces an arbitrary ordinal relationship between the categories, which may not necessarily reflect their true relationship.

One-Hot Encoding: One-hot encoding is a binary encoding technique where each unique category is represented as a binary vector with all zeros except for one "hot" or "active" bit, which represents the presence of that category. For example, using one-hot encoding, the categories "red," "green," and "blue" would be represented as [1, 0, 0], [0, 1, 0], and [0, 0, 1], respectively. One-hot encoding is useful when there is no ordinal relationship between categories, and it allows algorithms to treat categories as independent and equally weighted.

Binary Encoding: Binary encoding is a combination of label encoding and one-hot encoding, where each unique category is first assigned a numerical label and then represented as a binary vector. However, unlike one-hot encoding, binary encoding uses binary digits (0s and 1s) to represent the presence or absence of a category, instead of a single "hot" bit. This can result in more compact representations compared to one-hot encoding, particularly for features with a large number of categories.

Count Encoding: Count encoding is a technique that replaces each category with the count of its occurrences in the dataset. This can be useful for handling categorical features with high cardinality, where the number of unique categories is large. Count encoding can capture information about the frequency or prevalence of categories in the data, which can be relevant for some machine learning algorithms.

Target Encoding: Target encoding, also known as mean encoding, is a technique that replaces each category with the mean of the target variable for that category. It is particularly useful for classification problems, where the target variable is a categorical variable. Target encoding can capture the relationship between the categorical feature and the target variable, which can help improve the performance of some machine learning models.

Embedding: Embedding is a technique commonly used for text data, where words or phrases are represented as dense vectors of continuous values. Embeddings are learned representations that capture the semantic meaning or context of words or phrases, and they can be used as input features for machine learning models. Embeddings are typically learned using algorithms such as Word2Vec, GloVe, or FastText.

These are some common types of encoding techniques used in machine learning to convert categorical or text data into numerical representations that can be effectively processed by algorithms. The choice of encoding technique depends on the specific characteristics of the data and the requirements of the machine learning model being used.

2.2 Data Training And Testing

We split the data into training and testing

```
X = df_scaled.drop(["booking_status"] , axis =1 )

y = df_scaled["booking_status"]X_train , X_test , y_train , y_test = train_test_split(X , y,
random_state = 42 ,test_size =0.33)
```

Next, we utilized multiple regression methods to determine the viability of a regression-based approach for the data at hand.

Regression is a popular supervised machine learning technique that aims to model the relationship between dependent variables and one or more independent variables. There are several types of regression techniques commonly used in machine learning, including:

Linear Regression: This is the most basic type of regression, where a linear equation is used to model the relationship between the independent and dependent variables. It assumes a linear relationship between the variables, and the goal is to find the best-fit line that minimizes the residuals between the observed and predicted values.

Logistic Regression: Despite its name, logistic regression is actually used for classification tasks. It models the probability of an input belonging to a particular class, and is commonly used for binary classification problems where there are only two possible outcomes.

Polynomial Regression: This type of regression allows for the modeling of nonlinear relationships between the variables by including higher-order terms of the independent variables in the regression equation. It can capture more complex patterns in the data beyond linear relationships.

Ridge Regression: Ridge regression is a type of regularized regression that introduces a penalty term to the linear regression equation to mitigate multicollinearity (high correlation among independent variables). It can help prevent overfitting in cases where there are multiple highly correlated independent variables.

Lasso Regression: Lasso regression is another regularized regression technique that introduces a penalty term, but it uses a different type of penalty that can result in sparse solutions. It is often used for feature selection, as it can force some of the coefficients of the independent variables to exactly zero, effectively excluding them from the model.

Elastic Net Regression: This is a combination of Ridge and Lasso regression that incorporates both types of penalties. It allows for a balance between Ridge and Lasso regularization, and is useful when dealing with datasets that have high multicollinearity and a large number of features.

Support Vector Regression (SVR): SVR is a regression technique that uses support vector machines (SVM) to model the relationship between variables. It can handle nonlinear relationships and is particularly useful for datasets with outliers.

Decision Tree Regression: Decision tree regression is a non-parametric technique that models the relationship between variables using a tree-like structure. It can capture nonlinear relationships and interactions between variables.

Random Forest Regression: It is an ensemble learning method that combines multiple decision trees to create a more accurate and robust regression model. It involves building a collection of decision trees, where each tree is trained on a random subset of the training data and a random subset of the features. The predictions from all the individual trees are then averaged to obtain the final prediction.

These are just some of the many regression techniques available in machine learning. The choice of regression method depends on the specific characteristics of the data and the problem at hand, and experimentation with different techniques may be necessary to determine the most appropriate approach.

TABLE 2.2

Model	Accuracy	R-Squared	RMSE	Time Taken
Random Forest Regressor	0.64	0.64	0.28	10.12
Extra Trees Regressor	0.64	0.64	0.28	2.81
XGB Regressor	0.61	0.61	0.29	4.66
Bagging Regressor	0.61	0.61	0.29	0.44
LGBM Regressor	0.6	0.6	0.3	1.15
Hist Gradient Boosting Regressor	0.59	0.6	0.3	1.77
K Neighbours Regressor	0.5	0.5	0.33	4.56
MLP Regressor	0.49	0.49	0.34	6.93
Gradient Boosting Regressor	0.48	0.48	0.34	2.09
SVR	0.45	0.45	0.35	30.83
Nu SVR	0.43	0.44	0.35	54.8
Decision Tree Regressor	0.38	0.38	0.37	0.08
Extra Tree Regressor	0.35	0.35	0.38	0.06
Bayesian Ridge	0.32	0.32	0.39	0.06
Ridge CV	0.32	0.32	0.39	0.72
Ridge	0.32	0.32	0.39	1.12
Transformed Target Regressor	0.32	0.32	0.39	0.48
Linear Regression	0.32	0.32	0.39	0.05
Lasso Lars IC	0.32	0.32	0.39	0.38

Lasso Lars CV	0.32	0.32	0.39	0.1
Lars CV	0.32	0.32	0.39	0.53
Lars	0.32	0.32	0.39	0.51
Elastic Net CV	0.32	0.32	0.39	0.24
Lasso CV	0.32	0.32	0.39	0.23
Ada Boost Regressor	0.32	0.32	0.39	0.22
SGD Regressor	0.31	0.31	0.39	0.36
Orthogonal Matching Pursuit CV	0.3	0.3	0.39	0.14
Huber Regressor	0.29	0.29	0.4	0.39
Tweedie Regressor	0.23	0.23	0.41	0.3
Linear SVR	0.2	0.2	0.42	1.02
Orthogonal Matching Pursuit	0.18	0.18	0.42	0.07
Poisson Regressor	0.14	0.14	0.43	0.18
Dummy Regressor	0	0	0.47	0.03
Elastic Net	0	0	0.47	0.12
Lasso Lars	0	0	0.47	0.03
Lasso	0	0	0.47	0.16
Kernel Ridge	-0.19	-0.18	0.51	47.64
Passive Aggressive Regressor	-1.03	-1.03	0.67	0.11
RANSAC Regressor	-1.04	-1.03	0.67	22.09
Gaussian Process Regressor	-43541.1	-43482.9	98	162.57

The accuracy plot for different methods is shown below.

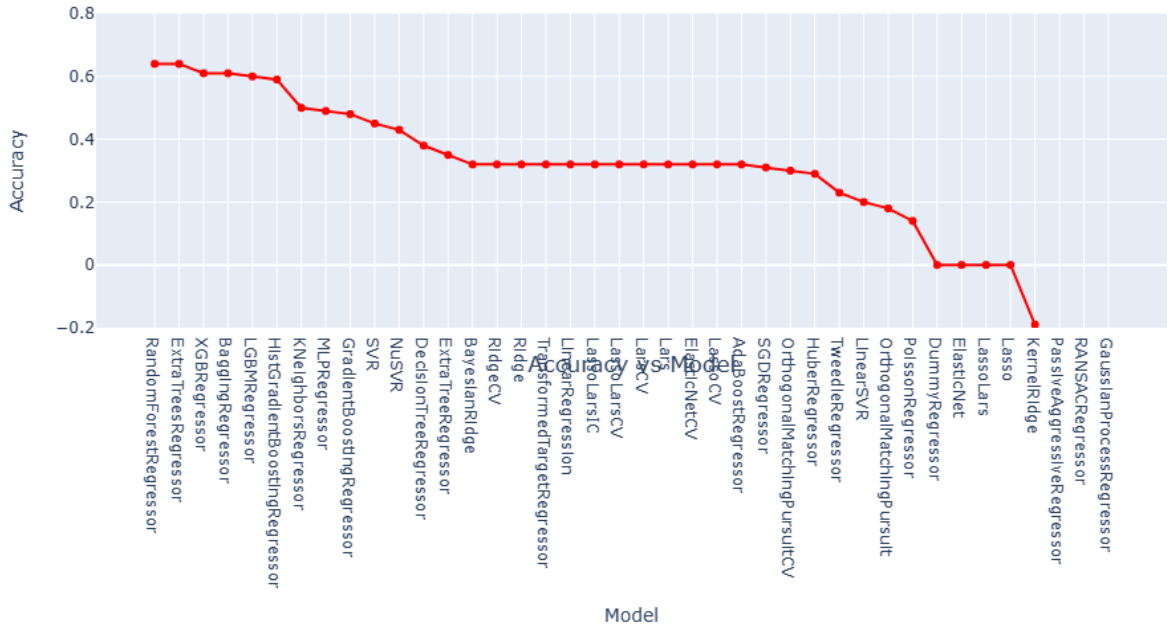


Fig. 2.2.1 Accuracy plot for different methods

Additionally, we can depict the time taken by each process in a graphical format, as illustrated in the graph presented below.

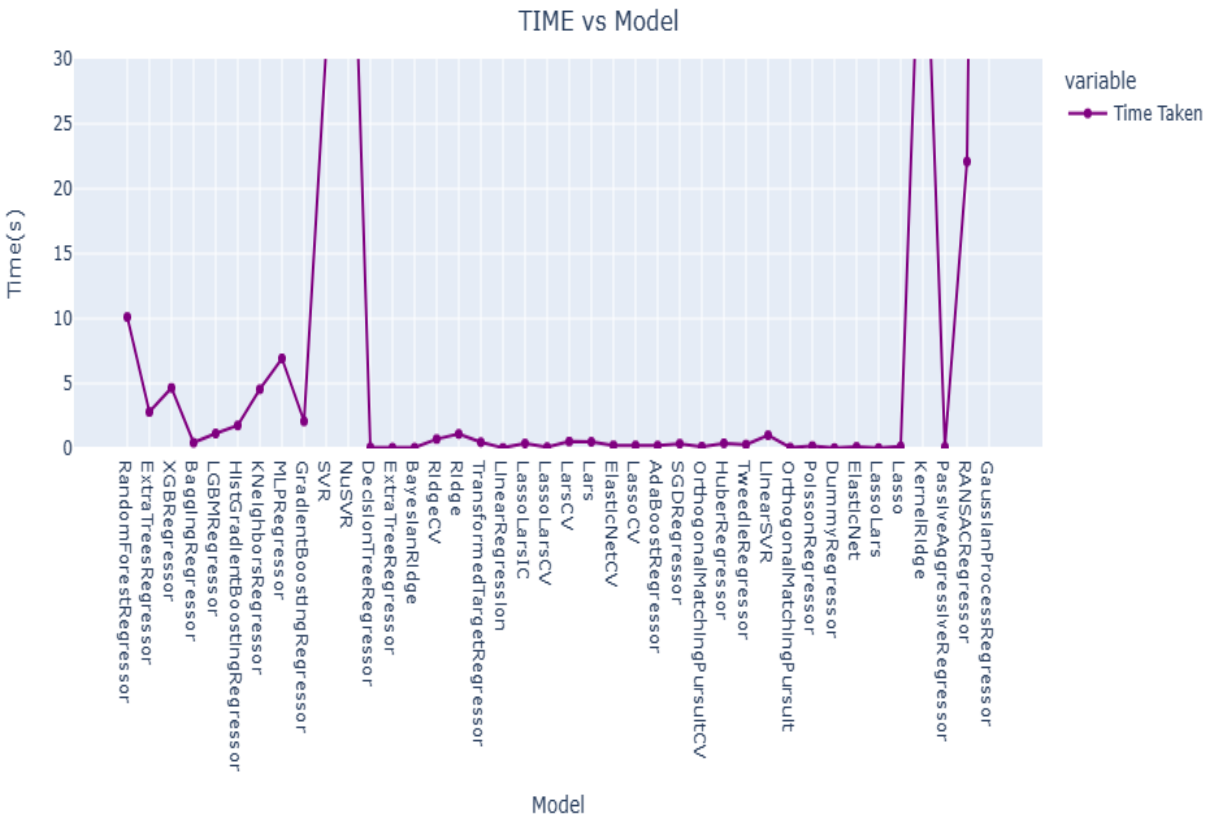


Fig. 2.2.2 Time Vs Model

2.2.2 Classification

Classification techniques are a type of supervised machine learning algorithms that are used to predict the class or category of an object or instance based on its features or attributes. There are several popular classification techniques used in machine learning, including:

Logistic Regression: It is a binary classification technique that uses a logistic function to model the probability of an instance belonging to a particular class.

Decision Trees: This technique involves constructing a tree-like structure where decisions

are made based on feature values, leading to the assignment of instances to different classes.

Random Forests: This is an ensemble technique that combines multiple decision trees to improve classification accuracy and reduce overfitting.

Support Vector Machines (SVM): SVM is a powerful technique for binary classification that finds an optimal hyperplane to separate instances from different classes in a higher-dimensional space.

Naive Bayes: This technique is based on Bayes' theorem and assumes that features are conditionally independent, making it computationally efficient for text classification and spam detection.

K-Nearest Neighbors (KNN): KNN assigns instances to classes based on the majority class of its k-nearest neighbors in the feature space.

Neural Networks: These are deep learning models that can be used for classification tasks with complex data, such as images or speech.

Gradient Boosting Methods: Techniques such as AdaBoost and Gradient Boosting Machines (GBM) combine weak learners to create a strong classifier for improved accuracy.

Deep Learning Techniques: Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) are commonly used for image, text, and time series data classification tasks.

These are just a few examples of the many classification techniques available in machine learning, and the choice of technique depends on the nature of the data and the specific problem at hand.

TABLE- 2.2.2

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
Random Forest Classifier	0.9	0.87	0.87	0.89	3.82
Extra Trees Classifier	0.89	0.86	0.86	0.89	2.76
XGB Classifier	0.89	0.86	0.86	0.89	2.52
Bagging Classifier	0.88	0.86	0.86	0.88	0.66

LGBM Classifier	0.88	0.86	0.86	0.88	0.67
Decision Tree Classifier	0.86	0.85	0.85	0.86	0.2
Label Spreading	0.85	0.83	0.83	0.85	551.45
Label Propagation	0.85	0.83	0.83	0.85	270.02
K Neighbours Classifier	0.85	0.83	0.83	0.85	10.71
Extra Tree Classifier	0.85	0.82	0.82	0.84	0.08
SVC	0.84	0.79	0.79	0.83	53.82
Ada Boost Classifier	0.82	0.78	0.78	0.81	1.01
Nu SVC	0.81	0.75	0.75	0.8	80.97
Logistic Regression	0.79	0.74	0.74	0.78	0.19
Nearest Centroid	0.74	0.74	0.74	0.75	0.09
Calibrated Classifier CV	0.79	0.74	0.74	0.78	13.66
Linear SVC	0.79	0.73	0.73	0.78	5.2
Linear Discriminant Analysis	0.79	0.73	0.73	0.78	0.36
Ridge Classifier CV	0.79	0.73	0.73	0.78	0.12
Ridge Classifier	0.79	0.73	0.73	0.78	0.11
SGD Classifier	0.78	0.71	0.71	0.77	0.33
Bernoulli NB	0.76	0.7	0.7	0.75	0.14
Passive Aggressive Classifier	0.73	0.7	0.7	0.73	0.09
Perceptron	0.7	0.69	0.69	0.7	0.11
Quadratic Discriminant Analysis	0.47	0.6	0.6	0.42	0.11
Gaussian NB	0.44	0.58	0.58	0.37	0.08
Dummy Classifier	0.67	0.5	0.5	0.54	0.06

And the accuracy curve for each classification technique is shown below for the given dataset:

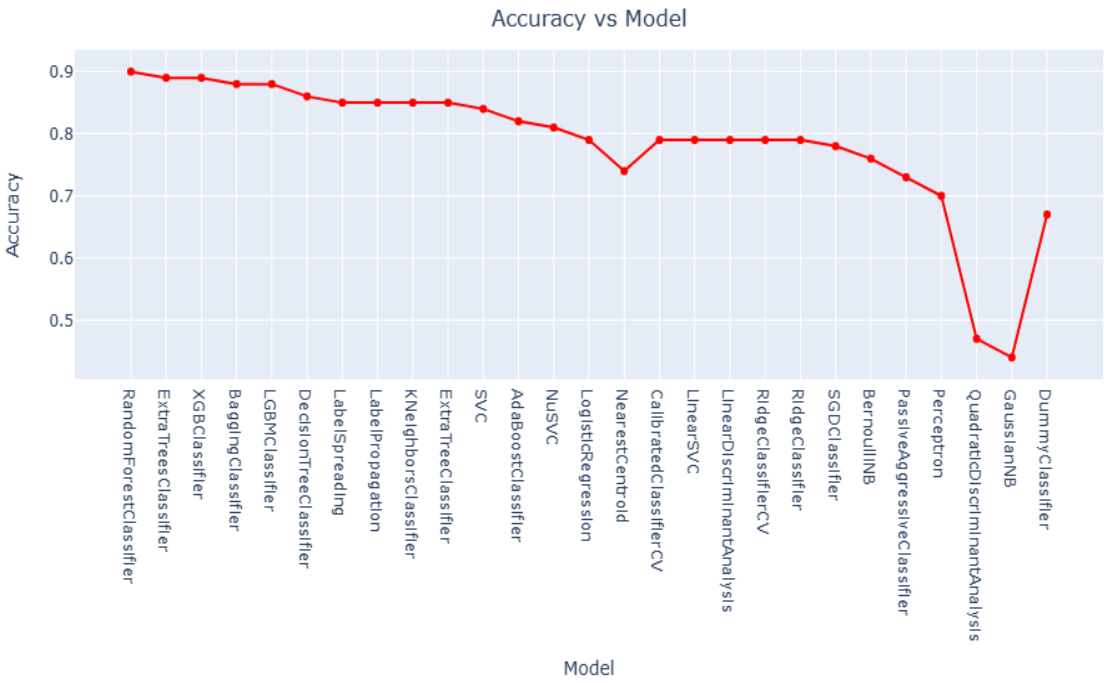


Fig. 2.2.3 Accuracy curve for each classification technique

The time taken by each classification method can be visualized through the diagram presented below.

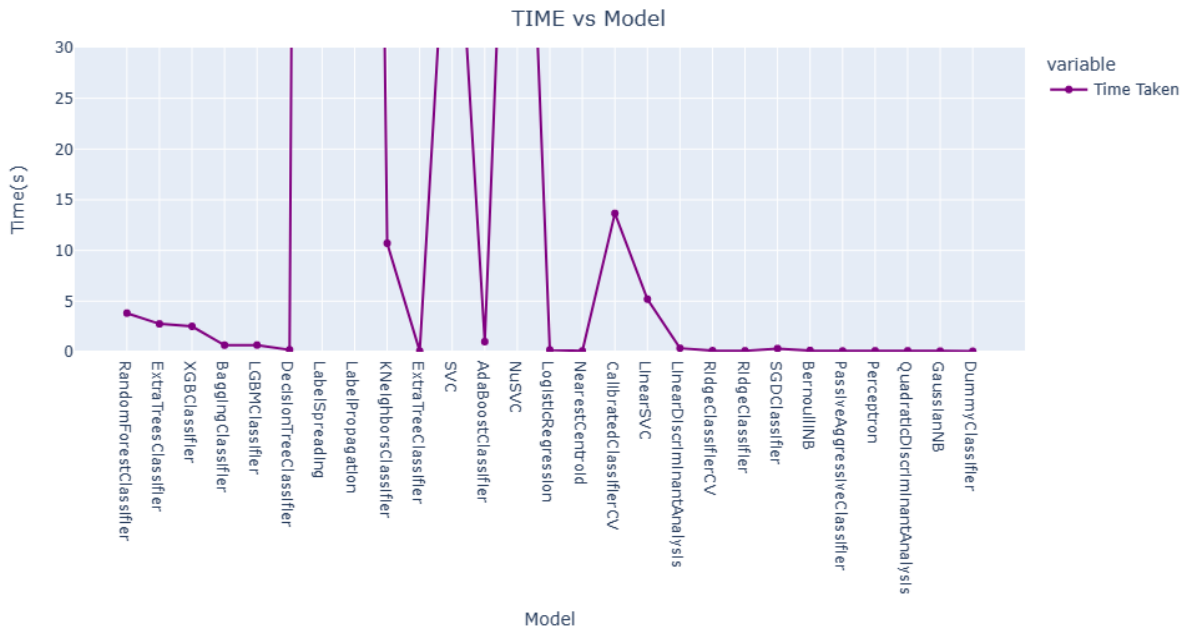


Fig. 2.2.4 time taken by each classification method

Finally, the F1-score and ROC score we can compare for each technique:

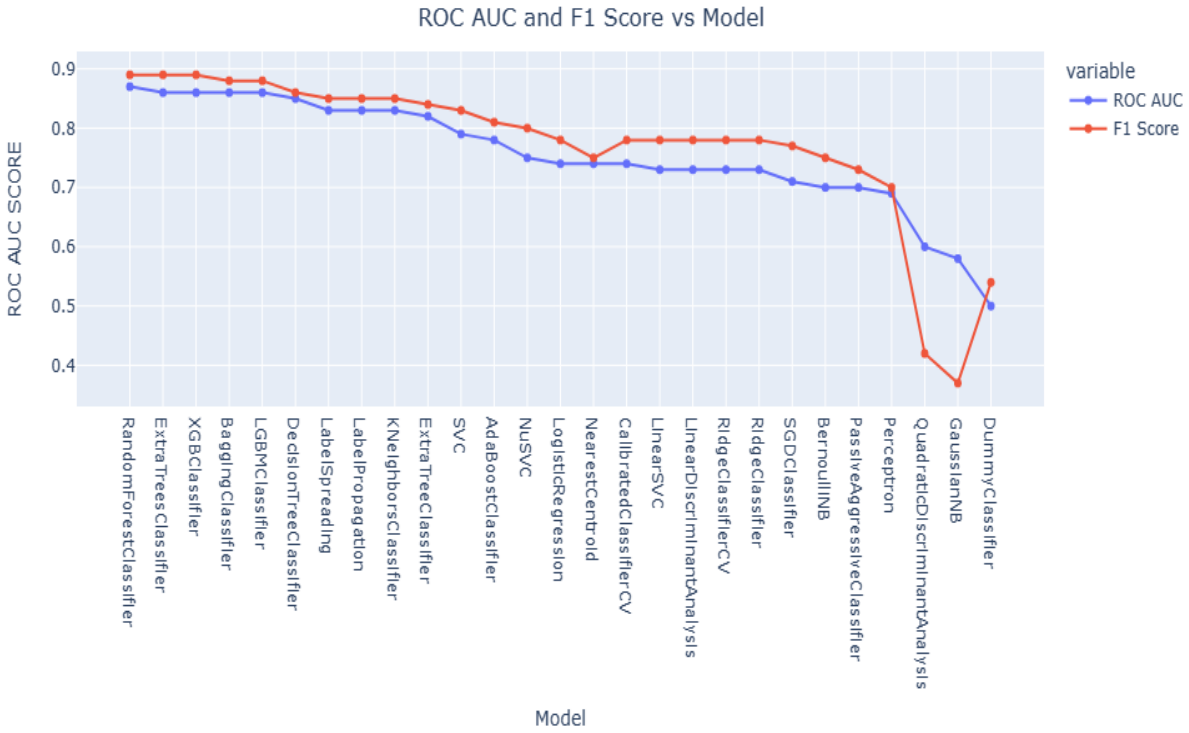


Fig. 2.2.5 F1-score and ROC score

Chapter 3

Observations & Conclusion

3.1 Analysis Of Result

Using a regression-based approach, we were able to achieve a model accuracy of approximately 64 percent with the Random Forest Regressor algorithm, but the processing time was around 10 seconds. However, when we used the Bagging Regressor algorithm, the processing time was significantly reduced (around 0.44 seconds), but the accuracy dropped slightly to around 61 percent.

When we use a classification-based approach, the Decision Tree Classifier and Extra Tree Classifier models took the least amount of time (about 0.04 seconds) and achieved an accuracy of around 85%. On the other hand, by using the Random Forest classification approach, we were able to achieve an accuracy of 90%, but it took around 2.35 seconds to complete.

3.2 Confusion Matrix

Then we adopted the Decision Tree Classifier model which is best suitable for our requirement

	precision	recall	f1-score	support
0.0	0.90	0.90	0.90	8024
1.0	0.79	0.79	0.79	3940
accuracy			0.86	11964
macro avg	0.84	0.84	0.84	11964
weighted avg	0.86	0.86	0.86	11964

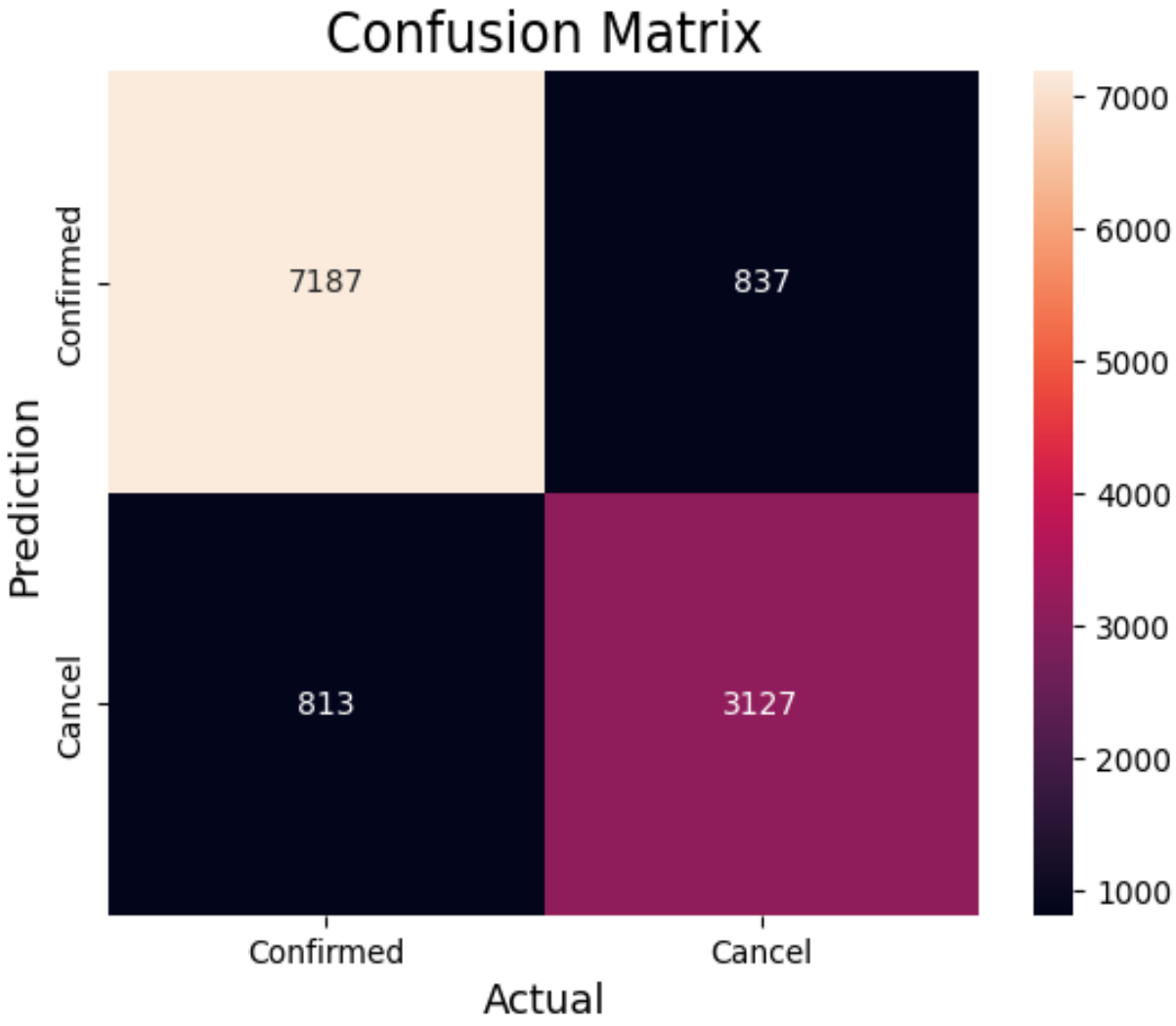


Fig. 3.2 Confusion Matrix

3.2 Decision Tree

We can visualize from the tree diagram of max depth = 3 shown below:

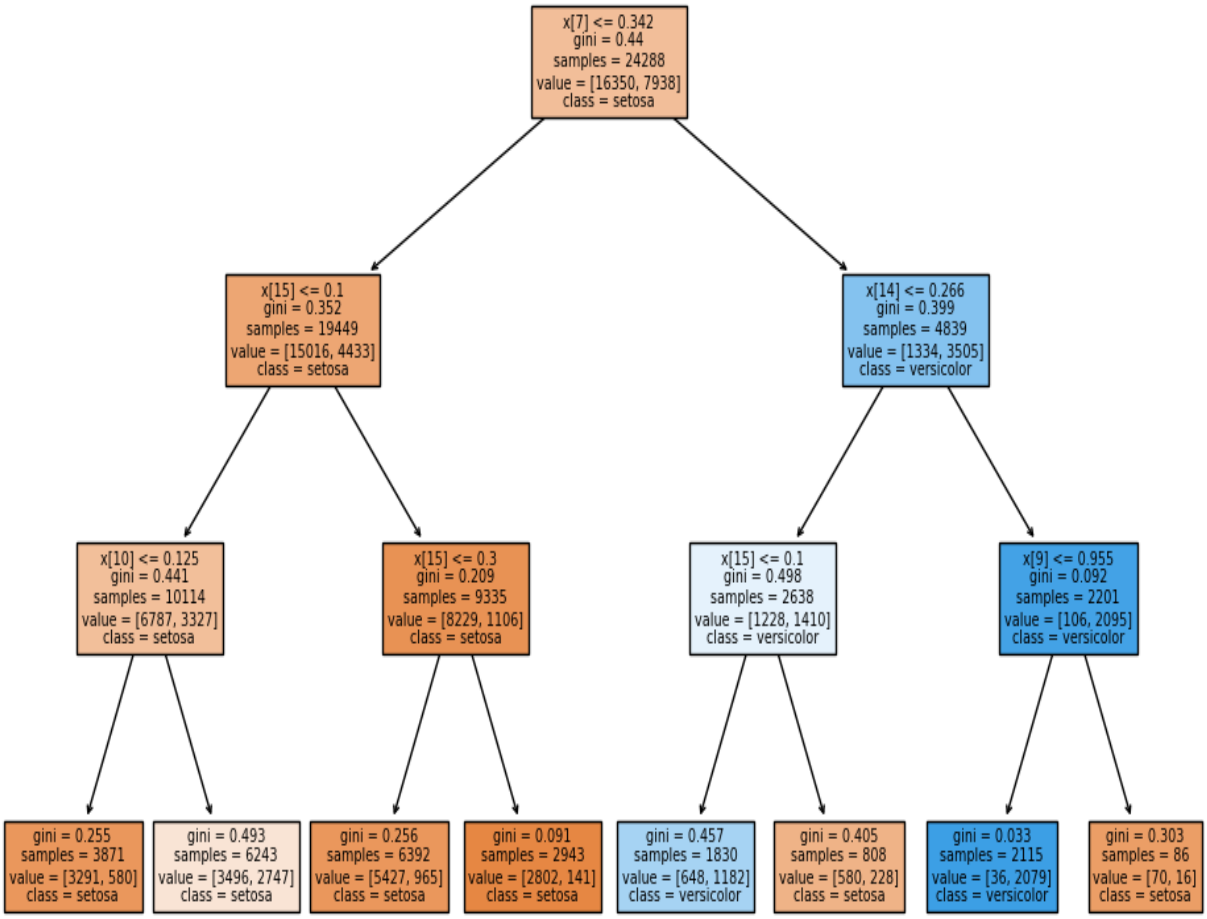


Fig. 3.3 Decision Tree

Chapter 4

Code Script

We did our complete project using Python and its libraries, and used the libraries for completing this project as mentioned below:

 # Imported Libraries

```
from ydata_profiling import ProfileReport
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from sklearn.impute import SimpleImputer
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import StratifiedKFold, train_test_split
from lightgbm import LGBMClassifier
import lazypredict
from lazypredict.Supervised import LazyClassifier

import time
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
from sklearn import linear_model
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import OrdinalEncoder
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.offline as py
import plotly.graph_objs as go
import plotly.tools as tls
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier

from sklearn.ensemble import ExtraTreesClassifier
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

#Data visualization

 # for plotting each column for an individual feature

```
unique_values = {}
for col in dp.columns:
    unique_values[col] = dp[col].value_counts().shape[0]

pd.DataFrame(unique_values, index=['unique value count']).transpose()
```

```
df = dp.drop('Booking_ID',axis=1)
df.head(2)
```

```
df.describe().T
```

```
# Calculate the number of rows and columns in the plot grid
num_cols = len(df.columns)
num_rows = int(num_cols / 2) + (num_cols % 2)

# Set the size of each subplot
fig, axs = plt.subplots(num_rows, 2, figsize=(6, 3*num_rows))

# Loop over the columns and create a bar plot for each one
for i, column in enumerate(df.columns):
    row = i // 2
    col = i % 2
    df[column].value_counts().plot.bar(ax=axs[row, col])
    axs[row, col].set_title(column)

plt.tight_layout()
plt.show()
```

#for plotting column with respect to output

```
import plotly.express as px
cont_features=['room_type_reserved','market_segment_type','no_of_adults', 'n
o_of_children', 'no_of_weekend_nights', 'no_of_week_nights', 'lead_time', 'a
rrival_month', 'no_of_previous_cancellations', 'no_of_previous_bookings_not_
canceled', 'avg_price_per_room', 'no_of_special_requests']
for col in cont_features:
    fig = px.histogram(data_frame = df,
                       x=col,
                       color= 'booking_status',
```

```
                        color_discrete_sequence =  ['#58D68D','#DE3163'],
                        marginal="box",
                        nbins= 120,
                         template="plotly_white"
                       )
    fig.update_layout(title = "Distribution " + col , title_x = 0.5)
    fig.show()

plt.figure(figsize=(28,24))
sns.heatmap(df.corr(), annot=True)
```

 # Ordinal encoding on object type varables

```
COLUMNS_FOR_FACTORISATION = df.select_dtypes('object').columns
factorization_table = {}
for column in df.columns:
    if column in COLUMNS_FOR_FACTORISATION:
        df[column], table = pd.factorize(df[column])
        factorization_table[column] = pd.DataFrame(table, columns=[column])
pd. set_option ('display.max_columns',None)
df.head(5)

df.columns
```

 # describe the data

```
df.describe(include='all').T
```

 # scaling the data

```
# define the scaler
scaler = MinMaxScaler()
df_scaled = scaler.fit_transform(df)
df_scaled = pd.DataFrame(df_scaled, columns=df.columns)
df_scaled.head()
```

 # correlation matrix

```
plt.figure(figsize=(28,24))
sns.heatmap(df.corr(), annot=True)
```

#skewness

```
from scipy.stats import skew

# Get list of numerical columns
num_cols = df_scaled.select_dtypes(include=['float64', 'int64']).columns.tol
ist()

# Calculate skewness for each numerical column
skew_vals = {}
for col in num_cols:
    skew_vals[col] = skew(df_scaled[col])

# Print skewness values
for col, val in skew_vals.items():
    print(f'{col}: {val:.2f}')
```

 # train-test split

```
X = df_scaled.drop(["booking_status"] , axis =1 )
y = df_scaled["booking_status"]
X_train , X_test , y_train , y_test = train_test_split(X , y, random_state =
42 ,test_size =0.33)
```

 # multiple regression methods

```
from lazypredict.Supervised import LazyRegressor
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
# Fit models
reg = LazyRegressor(verbose=0, ignore_warnings=True, custom_metric=None)
model, predictions = reg.fit(X_train, X_test, y_train, y_test)
```

 # multiple classification methods

```
clf = LazyClassifier(verbose=0,
                     ignore_warnings=True,
                     custom_metric=None,
                     predictions=False,
                     random_state=42,
                     classifiers='all')

models, predictions = clf.fit(X_train , X_test , y_train , y_test)
```

 # plotting output vs accuracy curve

```
x = models["Model"].unique()
y = models["Accuracy"].values
```

```
line = px.line(data_frame= models ,y =y,x=x , markers = True)
line.update_xaxes(title="Model",
                  rangeslider_visible = False)
line.update_yaxes(title = "Accuracy")
line.update_traces(line_color="red")
line.update_layout(showlegend = True,
    title = {
        'text': 'Accuracy vs Model',
        'y':0.94,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'})

line.show()
```

 # time taken by each classification method

```
line = px.line(data_frame= models ,y =["Time Taken"],x=x , markers = True)
line.update_xaxes(title="Model",
                  rangeslider_visible = False)
line.update_yaxes(title = "Time(s)",range=[0,30])
line.update_traces(line_color="purple")
line.update_layout(showlegend = True,
    title = {
        'text': 'TIME vs Model',
        'y':0.94,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'})

line.show()
```

 # F1-score and ROC score

```
line = px.line(data_frame= models ,y =["ROC AUC" , "F1 Score"] ,x=x, markers
= True)
line.update_xaxes(title="Model",
                  rangeslider_visible = False)
line.update_yaxes(title = "ROC AUC SCORE")
line.update_layout(showlegend = True,
    title = {
        'text': 'ROC AUC and F1 Score vs Model',
        'y':0.94,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'})
```

```
line.show()

# confusion matrix for Decision Tree

clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))


# heatmap for confusion matrix

cm = confusion_matrix(y_test, y_pred)

#Plot the confusion matrix.
sns.heatmap(cm,
            annot=True,
            fmt='g',
            xticklabels=['Confirmed','Cancel'],
            yticklabels=['Confirmed','Cancel'])
plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()

#PCA analysis

import numpy as np
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# Load the iris dataset
iris = load_iris()
# Instantiate PCA with the number of components you want to keep
n_components = 2  # Number of components to keep
pca = PCA(n_components=n_components)

# Fit PCA on the training dataset
pca.fit(X_train)

# Transform the training and testing datasets to the first n_components prin
cipal components
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)
```

```
# Now, X_train_pca and X_test_pca are the transformed datasets with reduced
dimensionality

# Print the explained variance ratio of the principal components
print("Explained Variance Ratio: ", pca.explained_variance_ratio_)
X_train_pca
```

 # decision Tree drawing

```
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

# Load the iris dataset
iris = load_iris()
X = X_train
y = y_train

# Instantiate a decision tree classifier
clf = DecisionTreeClassifier(random_state=42,max_depth=3)
clf.fit(X, y)

# Plot the decision tree diagram without feature names
fig, ax = plt.subplots(figsize=(12, 8))
tree.plot_tree(clf, feature_names=None, class_names=iris.target_names, fille
d=True, ax=ax)
plt.show()
```

References

[1] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[2] Sergey Lobov, Nadia Krilova, Innokentiy Kastalskiy, Victor Kazantsev, and Valeri A Makarov. Latent factors limiting the performance of semg-interfaces. *Sensors*, 18(4):1122, 2018.

[3] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[4] Introduction to Machine Learning with Python: A Guide for Data Scientists by Andreas C. Müller, Sarah Guido