ConfigurationManual

MScResearchProject
Data Analytics

Vishan Lal
StudentID: x21120803

SchoolofComputing
National College of Ireland

Supervisor: Prof. Cristina Muntean

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Vishan Lal
Student ID:	X21120803
Programme:	Data Analytics
Year:	2022-23
Module:	MSc Research Project
Supervisor:	Prof. Cristina Muntean
Submission Due Date:	15/12/2022
Project Title:	Configuration Manual
Word Count:	xxx
Page Count:	15

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	VIshan Lal
Date:	15th December 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Vishan lal
X21120803

1 Introduction

This research aims to address about how can Machine learning models like Linear Regression, Decision Tree Regressor, Bagging Regressor, XG Boost Regressor, K neighbours Regressor, Extra Trees Regressor, Ridge Regressor and Lasso Regressor are able to accurately predict the price of the airfare for a specific departure date, and how does flight price in India vary based on various factors that determines the flight price using the different exploratory data analysis techniques to make a better customer experience in booking the flight at the cheapest cost.

2 System Specification

Processor : Apple M1 Pro
Memory(RAM) Installed : 16 GB
Storage: 1 TB SSD

3 Software Tools

The tools required for this project are:

- Anaconda Navigator
- Python
- Jupyter Notebook

4 Software Installation

This is a step by step explanation of the implementation.
Download and install python from https://www.python.org/downloads/

Downloading and installing anaconda from https://www.anaconda.com/

5 Implementation

The following packages and libraries are utilized:

NumPy
Pandas
Matplotlib
ScikitLearn
seaborn

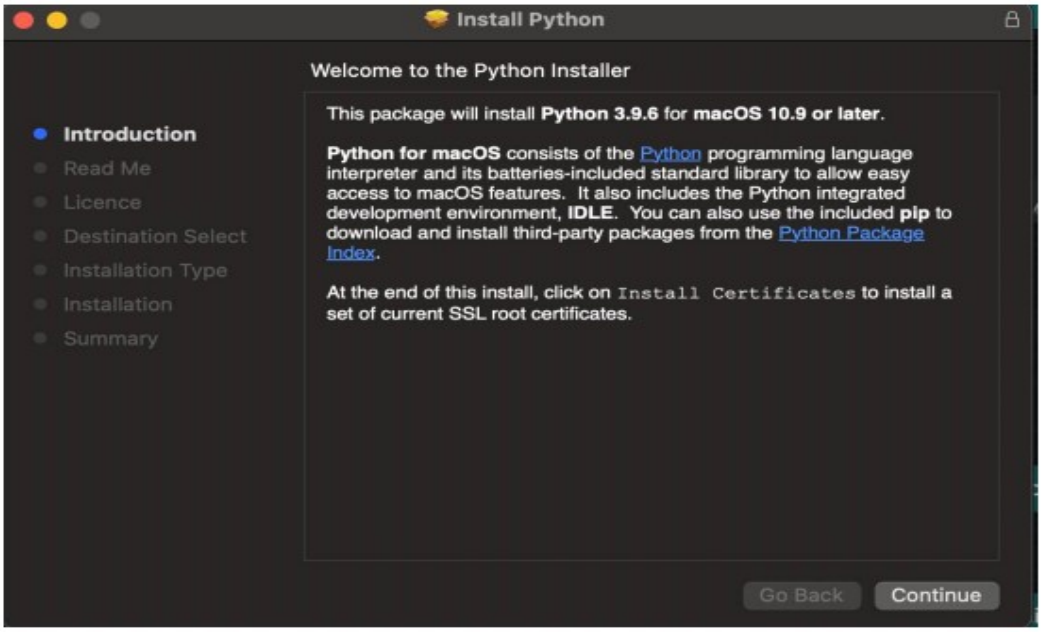

Figure 1: Python setup introduction

Figure 2: Readme in the python setup

Figure 3: Licence in Python setup

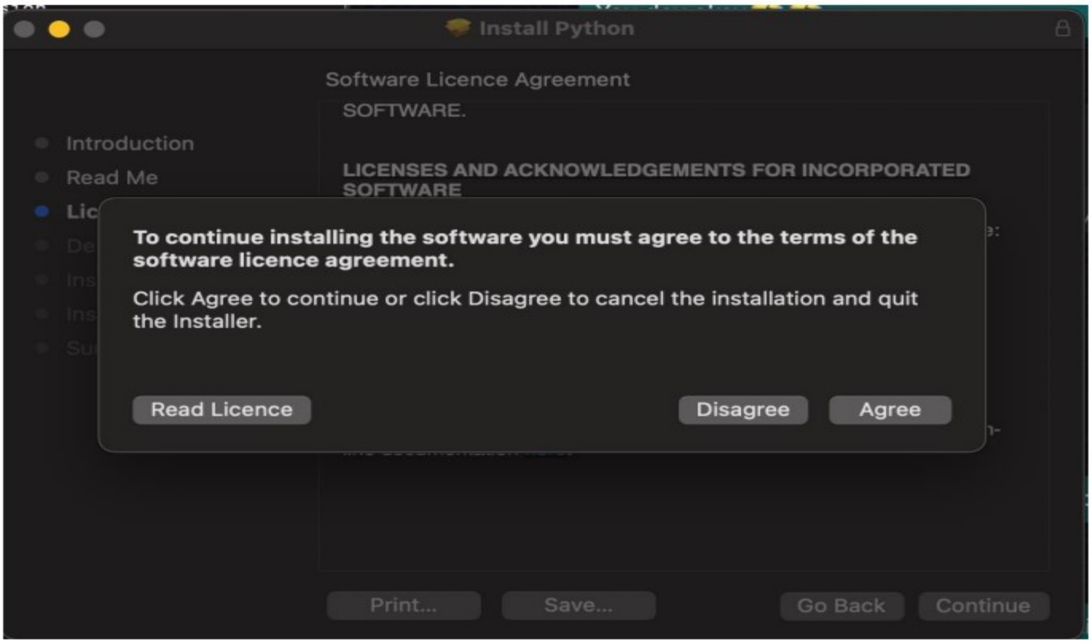

Figure 4: License aggrement in python setup

Figure 5: Installation type in setup

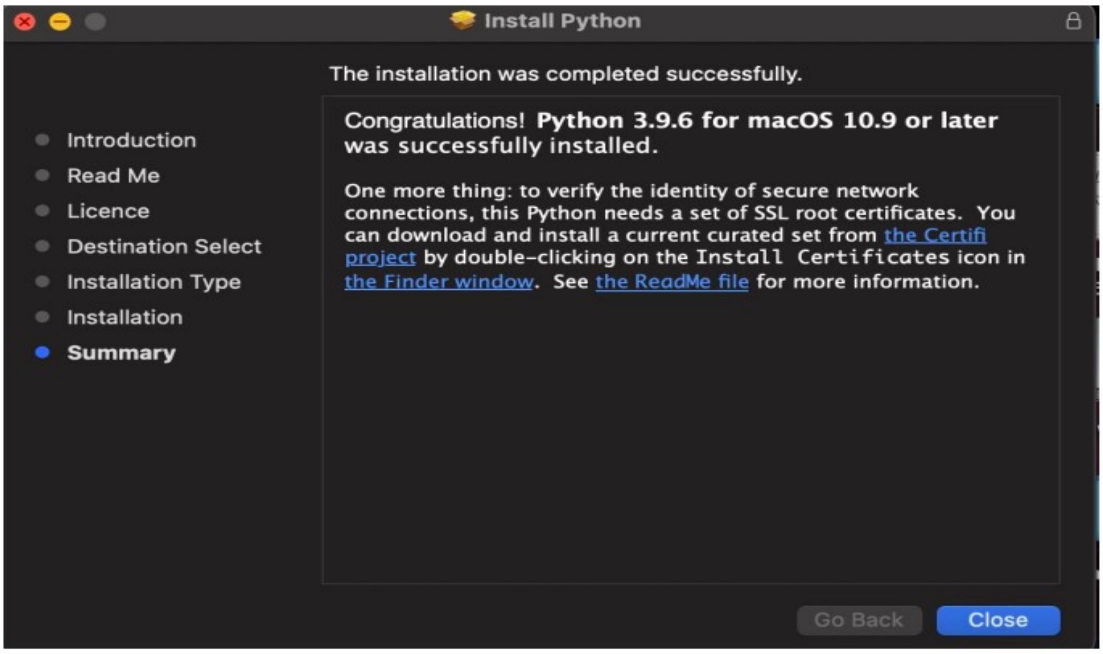

Figure 6: Summary of installation completion

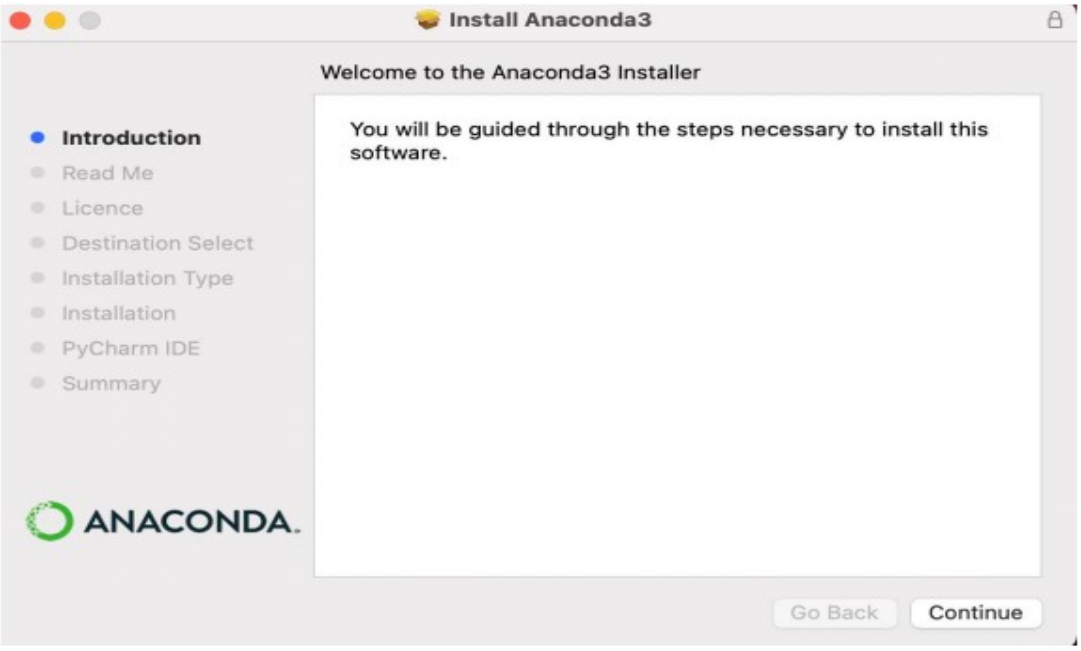

Figure 7: Introduction to anaconda installer setup

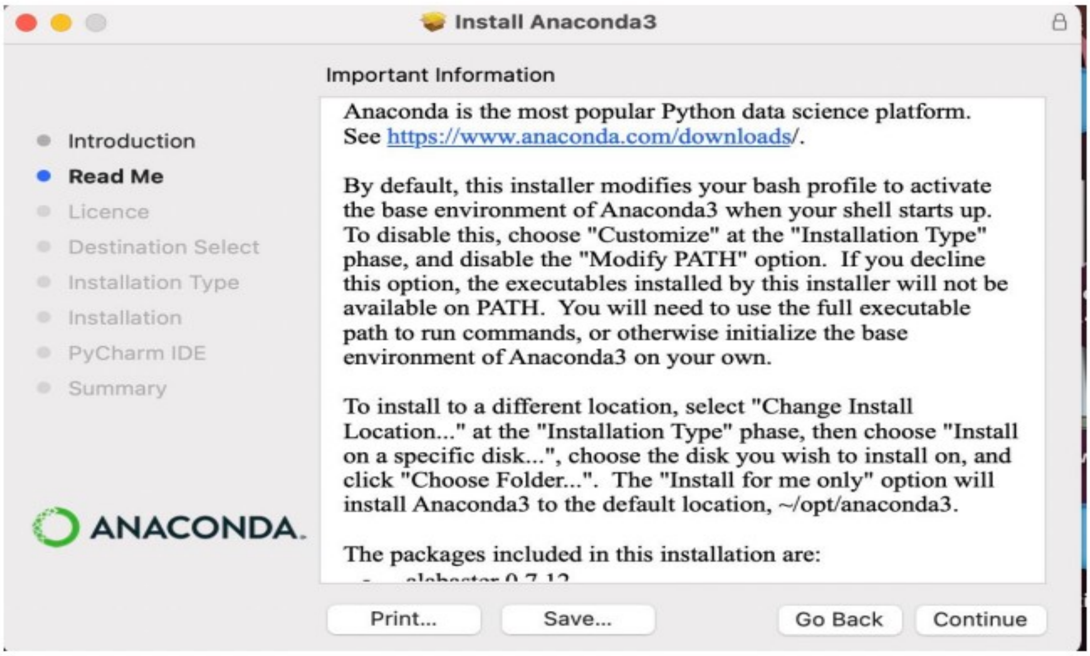

Figure 8: Readme in Anaconda installer setup

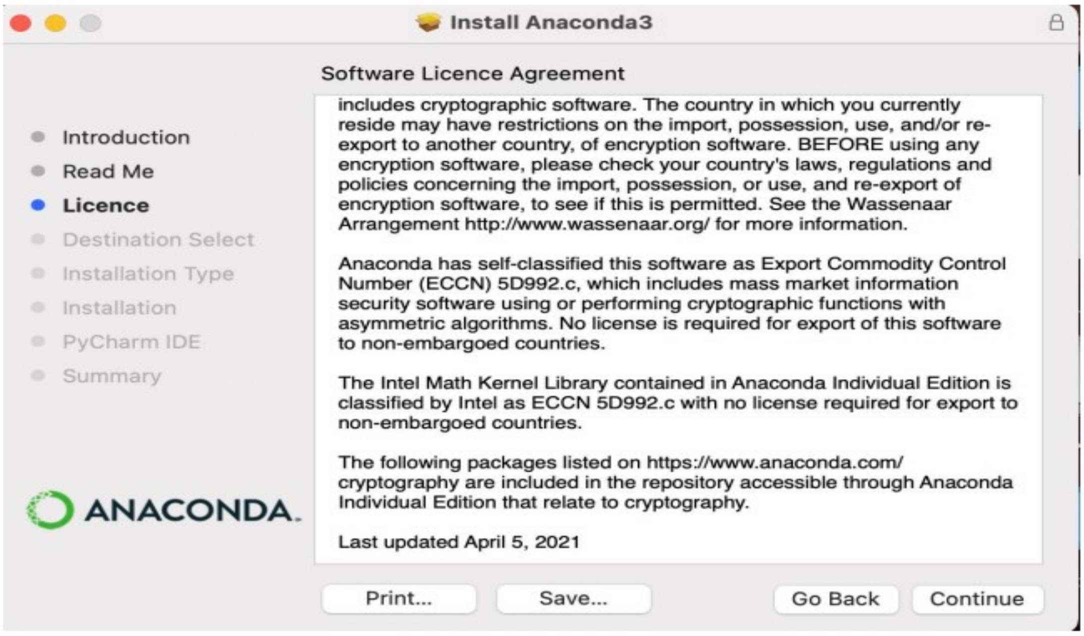

Figure 9: Licence agreement of anaconda setup

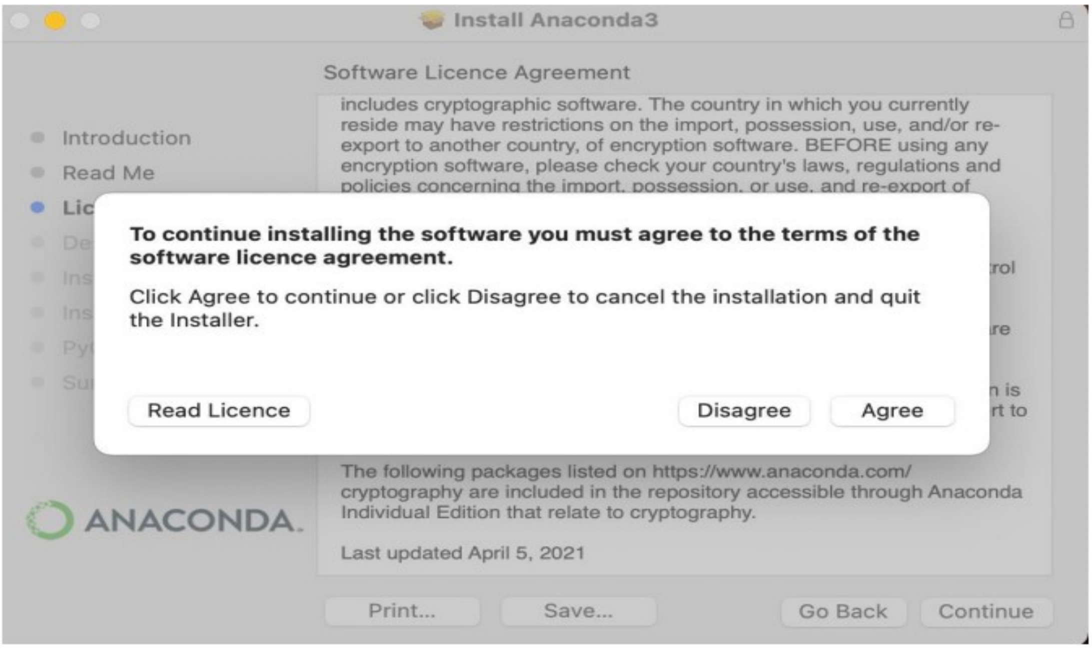

Figure 10: Agree to the License of anaconda installer setup

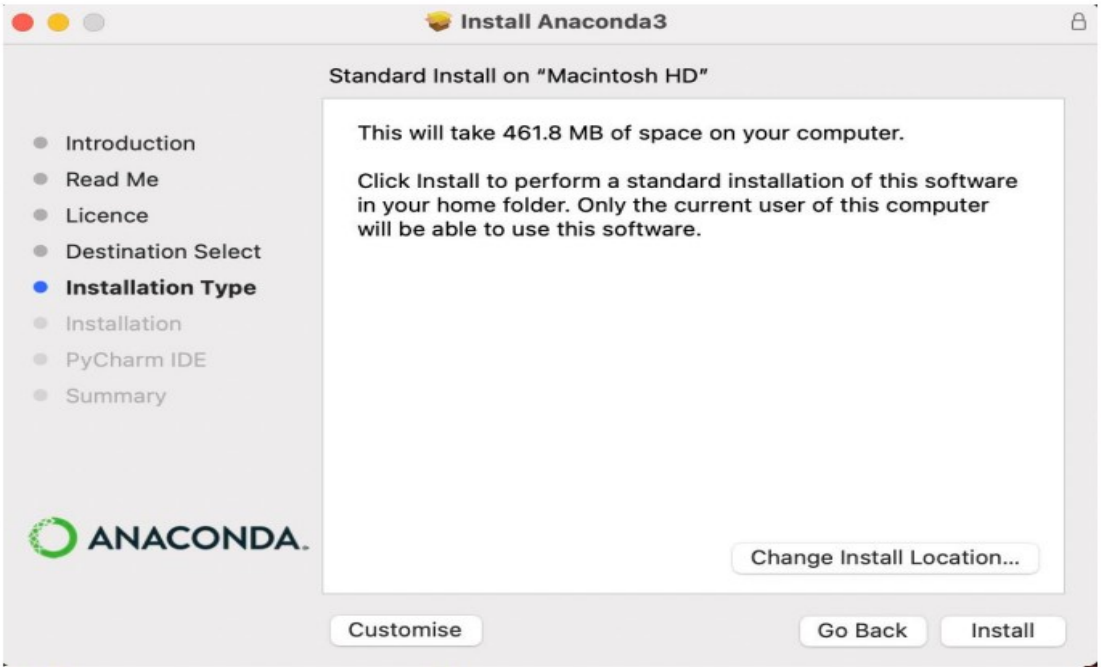

Figure 11: Choosing Anaconda type of installation

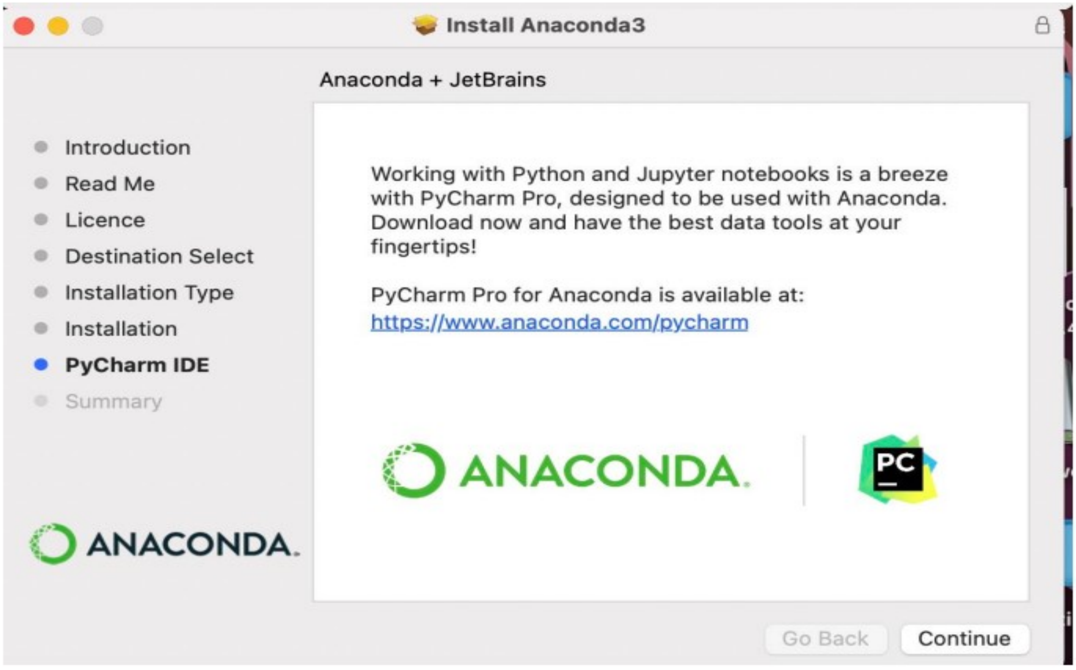

Figure 12: Installation completion of anaconda

5.1 Importing Libraries

The following stpes were taken to import the libraries:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
import os
import plotly.express as px
import plotly.graph_objects as go
from sklearn import preprocessing
from datetime import date, datetime, time
from babel.dates import format_date, format_datetime, format_time
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
from sklearn.feature_selection import RFE
```

Figure 13: Importing required Libraries and Packages

5.2 Loading The data

```
flights_df = pd.read_csv("../data/Clean_Dataset.csv")
bussiness_df = pd.read_csv("../data/business.csv")
economy_df = pd.read_csv("../data/economy.csv")
```

Figure 14: Loading the data

5.3 Data Analysis, preparation and visualisation

5.3.1 Columns/Features in Dataset

	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price
0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumbai	Economy	2.17	1	5953
1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.33	1	5953
2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.17	1	5956
3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.25	1	5955
4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.33	1	5955

Figure 15: 5 Top Records of the dataset

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300153 entries, 0 to 300152
Data columns (total 11 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   airline           300153 non-null  object 
 1   flight            300153 non-null  object 
 2   source_city       300153 non-null  object 
 3   departure_time    300153 non-null  object 
 4   stops             300153 non-null  object 
 5   arrival_time      300153 non-null  object 
 6   destination_city  300153 non-null  object 
 7   class             300153 non-null  object 
 8   duration          300153 non-null  float64
 9   days_left         300153 non-null  int64  
 10  price             300153 non-null  int64  
dtypes: float64(1), int64(2), object(8)
memory usage: 25.2+ MB
```

Figure 16: Printing the features of dataset

5.3.2 Descriptive stats of flights dataset

```
flights_df.describe()
```

	duration	days_left	price
count	300153.000000	300153.000000	300153.000000
mean	12.221021	26.004751	20889.660523
std	7.191997	13.561004	22697.767366
min	0.830000	1.000000	1105.000000
25%	6.830000	15.000000	4783.000000
50%	11.250000	26.000000	7425.000000
75%	16.170000	38.000000	42521.000000
max	49.830000	49.000000	123071.000000

Figure 17: Descriptive stats of flight dataset

5.3.3 Generating the one hot encoded values for the categorical values of flight dataframe

```
In [39]: dummies_data = pd.get_dummies(flights_df[['airline','source_city','departure_time','stops','arrival_time','class',
                                                    'destination_city']])
         dummies_data.head()

         final_data = pd.concat([flights_df.drop(['airline','source_city','departure_time','stops','arrival_time','class',
                                                  'destination_city'], axis=1), dummies_data], axis=1)
```

Figure 18: Generating the one hot encoded values

5.3.4 Features after doing one hot encoding

```
final_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300153 entries, 0 to 300152
Data columns (total 31 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   flight                        300153 non-null  object
 1   duration                      300153 non-null  float64
 2   days_left                     300153 non-null  int64
 3   price                         300153 non-null  int64
 4   airline                       300153 non-null  int64
 5   stops                         300153 non-null  int64
 6   class                         300153 non-null  int64
 7   source_city_Bangalore         300153 non-null  uint8
 8   source_city_Chennai           300153 non-null  uint8
 9   source_city_Delhi             300153 non-null  uint8
 10  source_city_Hyderabad         300153 non-null  uint8
 11  source_city_Kolkata           300153 non-null  uint8
 12  source_city_Mumbai            300153 non-null  uint8
 13  departure_time_Afternoon      300153 non-null  uint8
 14  departure_time_Early_Morning  300153 non-null  uint8
 15  departure_time_Evening        300153 non-null  uint8
 16  departure_time_Late_Night     300153 non-null  uint8
 17  departure_time_Morning        300153 non-null  uint8
 18  departure_time_Night          300153 non-null  uint8
 19  arrival_time_Afternoon        300153 non-null  uint8
 20  arrival_time_Early_Morning    300153 non-null  uint8
 21  arrival_time_Evening          300153 non-null  uint8
 22  arrival_time_Late_Night       300153 non-null  uint8
 23  arrival_time_Morning          300153 non-null  uint8
 24  arrival_time_Night            300153 non-null  uint8
 25  destination_city_Bangalore    300153 non-null  uint8
 26  destination_city_Chennai      300153 non-null  uint8
 27  destination_city_Delhi        300153 non-null  uint8
 28  destination_city_Hyderabad    300153 non-null  uint8
 29  destination_city_Kolkata      300153 non-null  uint8
 30  destination_city_Mumbai       300153 non-null  uint8
dtypes: float64(1), int64(5), object(1), uint8(24)
memory usage: 22.9+ MB
```

Figure 20: Feature set for flight data after doing one hot encoding

5.3.5 Plotting Correlation matrix including one hot encoded variable

```
plt.figure(figsize=(28,13))
cor = final_data.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
```

```
<AxesSubplot:>
```

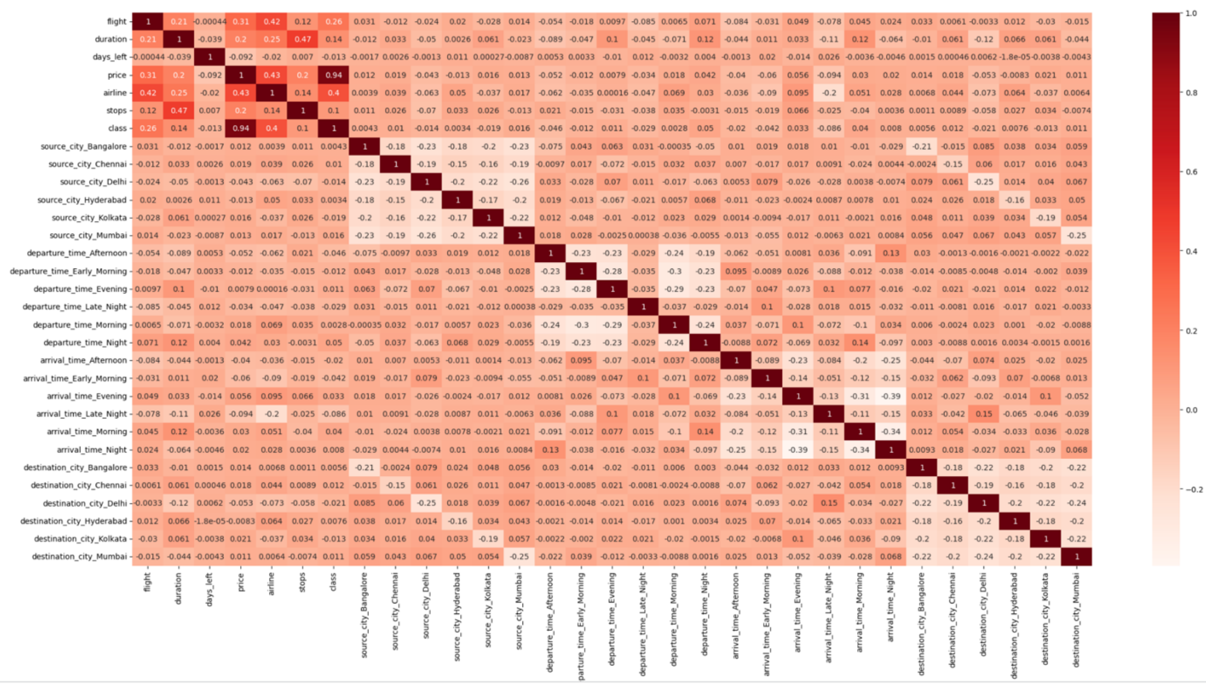

Figure 21: Correlation matrix including one hot encoded variable

5.6 Spliting the data into 70 percent training and 30 percent testing

Separating the features and the target values into separate dataframe variables

```
In [44]: # storing the dependent variables in X and Independent Variable in Y
x = final_data.drop(['price'], axis = 1)
y = final_data['price']
```

Splitting the data into training set and testing set in the ratio of 70% and 30% respectively

```
In [45]: from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.30, random_state = 42)
x_train.shape, y_train.shape, x_test.shape, y_test.shape

Out[45]: ((210107, 30), (210107,), (90046, 30), (90046,))
```

Figure 22: Split of data into training and testing

5.7 Performing the Min Max scaling for the preprocessed dataset

Scaling the values to convert the int values to fit into Machine Learning models

```
from sklearn.preprocessing import MinMaxScaler
mmscaler = MinMaxScaler(feature_range = (0, 1))
x_train = mmscaler.fit_transform(x_train)
x_test = mmscaler.fit_transform(x_test)
x_train = pd.DataFrame(x_train)
x_test = pd.DataFrame(x_test)
```

5.8 Machine Learning Algorithms

Implementing all the Regression models for predicting prices of flight

5.8.1 Importing libraries for models

```
# Build the regression / regressor models

from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
import xgboost as xgb
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import GradientBoostingRegressor
```

Figure 23: Library imports

5.8.2 Create objects of regression models with default hyper-parameters

```
modelmlg = LinearRegression()
modeldcr = DecisionTreeRegressor()
modelbag = BaggingRegressor()
#modelSVR = SVR()
modelXGR = xgb.XGBRegressor()
modelKNN = KNeighborsRegressor()
modelETR = ExtraTreesRegressor()
modelRE = Ridge()
modelL = Lasso(alpha = 0.1)

modelGBR = GradientBoostingRegressor()
```

Figure 24: **Creating objects of regression models**

5.8.3 Evaluation matrix for all algorithms

```
MM = [modelmlg, modeldcr, modelbag, modelXGR, modelKNN, modelETR, modelRE, modelL, modelGBR]
for model in MM:

    # fitting model
    model.fit(x_train, y_train)

    # predicting model with test data
    y_pred = model.predict(x_test)

    # print the model name
    print('Model name', model)

    # Evaluation metrics for Regression analysis

    from sklearn import metrics

    print('Mean Absolute Error (MAE): ', round(metrics.mean_absolute_error(y_test, y_pred), 3))
    print('Mean Squared Error (MSE): ', round(metrics.mean_squared_error(y_test, y_pred), 3))
    print('Root Mean Squared Error (RMSE): ', round(np.sqrt(metrics.mean_squared_error(y_test, y_pred)), 3))
    print('R2 Score: ', round(metrics.r2_score(y_test, y_pred), 5))
    print('Root Mean Squared Log Error (RMSLE): ', round(np.log(np.sqrt(metrics.mean_squared_error(y_test, y_pred))), 3

    # Define the function to calculate the MAPE - Mean Absolute Percentage Error

    def MAPE(y_test, y_pred):
        y_test, y_pred = np.array(y_test), np.array(y_pred)
        return np.mean(np.abs((y_test - y_pred) / y_test)) * 100

    # Evaluation of MAPE
    result = MAPE(y_test, y_pred)
    print('Mean Absolute Percentage Error (MAPE): ', round(result, 2), '%')

    # Calculate Adjusted R Squared values
    r_squared = round(metrics.r2_score(y_test, y_pred), 6)
    adjusted_r_squared = round(1-(1-r_squared)*(len(y)-1) / (len(y)-x.shape[1]-1), 6)
    print('Adjusted R Square: ', adjusted_r_squared)
    print("------------------------------------------------------------------------")
```

```
new_row = {'Model Name' : model,
           'Mean_Absolute_Error_MAE' : metrics.mean_absolute_error(y_test, y_pred),
           'Adj_R_Square' : adjusted_r_squared,
           'Root_Mean_Squared_Error_RMSE' : np.sqrt(metrics.mean_squared_error(y_test, y_pred)),
           'Mean_Absolute_Percentage_Error_MAPE' : MAPE(y_test, y_pred),
           'Mean_Squared_Error_MSE' : metrics.mean_squared_error(y_test, y_pred),
           'Root_Mean_Squared_Log_Error_RMSLE' : np.log(np.sqrt(metrics.mean_squared_error(y_test, y_pred))),
           'R2_Score' : metrics.r2_score(y_test, y_pred)}
results = results.append(new_row, ignore_index = True)
```

Figure 25: Generating evaluation metrics for all models

5.8.4 Results showcasing the prediction metrics based on regression for all the models implemented

```
models=['LinearRegression','DecisionTreeRegressor','KNeighborsRegressor','ExtraTreesRegressor',
        'GradientBoostingRegressor','XGBRegressor','BaggingRegressor',
        'Ridge Regression','Lasso Regression']
result=pd.DataFrame({'Model_Name':models})
result['Adj_R_Square']=results['Adj_R_Square']
result['Mean_Absolute_Error_MAE']=results['Mean_Absolute_Error_MAE']
result['Root_Mean_Squared_Error_RMSE']=results['Root_Mean_Squared_Error_RMSE']
result['Mean_Squared_Error_MSE']=results['Mean_Squared_Error_MSE']
result['Mean_Absolute_Percentage_Error_MAPE']=results['Mean_Absolute_Percentage_Error_MAPE']

result['Root_Mean_Squared_Log_Error_RMSLE']=results['Root_Mean_Squared_Log_Error_RMSLE']
result['R2_score']=results['R2_Score']
result=result.sort_values(by='Adj_R_Square',ascending=False).reset_index(drop=True)
result
```

	Model_Name	Adj_R_Square	Mean_Absolute_Error_MAE	Root_Mean_Squared_Error_RMSE	Mean_Squared_Error_MSE
0	XGBRegressor	0.984598	1109.516875	2815.333770	7.926104e+06
1	KNeighborsRegressor	0.982104	1197.038922	3034.699602	9.209402e+06
2	ExtraTreesRegressor	0.978741	1819.477614	3307.636674	1.094046e+07
3	DecisionTreeRegressor	0.973584	1258.952087	3686.991253	1.359390e+07
4	GradientBoostingRegressor	0.970023	1881.169731	3927.696573	1.542680e+07
5	Lasso Regression	0.957233	2785.160476	4691.359871	2.200886e+07
6	BaggingRegressor	0.906950	4572.481049	6919.907473	4.788512e+07
7	Ridge Regression	0.906950	4572.293761	6919.913344	4.788520e+07
8	LinearRegression	0.906949	4571.189881	6919.960632	4.788586e+07

Figure 26: Results and prediction metrics for all models

5.8.5 Plot showcasing the actual and the predicted prices of the flights for XG boost model

```
plt.figure(figsize=(10,5))
sns.regplot(x='Price_actual',y='Price_pred',data=result,scatter_kws={"color": "blue"}, line_kws={"color": "red"})
plt.title('Actual Price  Vs  Predicted Price ',fontsize=20)
plt.xlabel('Actual Price',fontsize=15)
plt.ylabel('Predicted Price',fontsize=15)
plt.show()
```

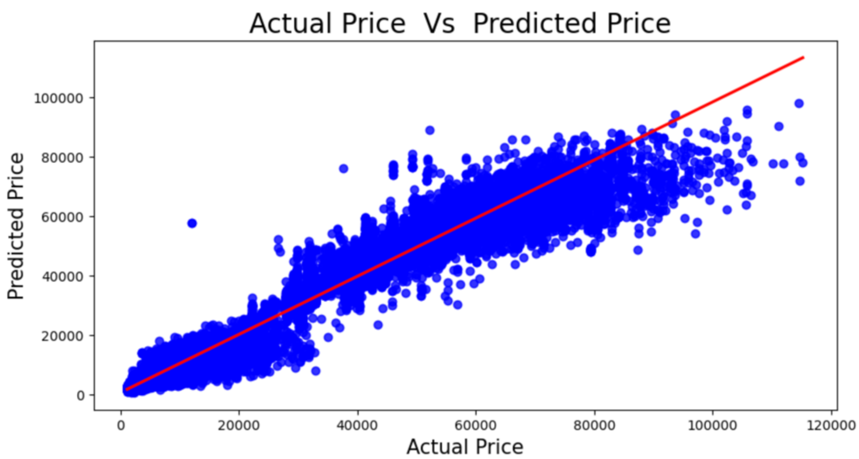

Figure 27: Actual and predicted prices of flights

6 Conclusion

The implementation of the code is shown in the document and the codes are commented for better understanding, for better readability the document is divided into sections and subsections.