FLIGHT PRICE PREDICTION PROJECT

Submitted by:

TAMALI SAHA

FlipRobo SME:

GULSHANA CHAUDHARY

ACKNOWLEDGMENT

I would like to express my special gratitude to Flip Robo Technologies team, who has given me this opportunity to deal with this dataset during my internship. It helped me to improve my analyzation skills. I want to express my gratitude to Ms. Gulshana Chaudhary (SME, Flip Robo) as she has helped me to get out of all the difficulties I faced while doing the project. I also want to give huge thanks to entire DataTrained team.

Bibliography:

Reference used in this project:

1. Hands on Machine learning with scikit learn and tensor flow by Aurelien Geron.
2. Andrew Ng Notes on Machine Learning (GitHub).
3. Different projects on Github and Kaggle.
4. Different conference papers on Recharchgate.
5. B. Smith, J. Leimkuhler, R. Darrow, and Samuels, “Yield management at American airlines, “Interfaces, vol. 22, pp. 8–31, 1992
6. Yeamduan Narangajavana, Fernando.J. Garrigos-Simon, Javier Sanchez García, Santiago Forgas-Coll, “Prices, prices and prices: A study in the airline sector”, Tourism Manage., 41 (2014), pp. 28-42
7. K. Tziridis, T. Kalampokas, G. A. Papakostas, and K. I. Diamantaras, “Airfare prices prediction using machine learning techniques,” in the 25th IEEE European signal processing conference, 2017, pp. 1036– 1039.
8. G.A. Papakostas, K.I. Diamantaras and T. Papadimitriou, “Parallel pattern classification utilizing GPU-Based kernelized slackmin algorithm,” doi:10.1016/j.jpdc.2016.09.001.
9. T. Janssen, “A linear quantile mixed regression model for prediction of airline ticket prices,” Bachelor Thesis, Radboud University, 2014.

Table of Contents

1. Introduction

1.1 Business Problem Framing

The airline industry is regarded as one of the most progressive, employing sophisticated strategies and ways to dynamically allocate airline rates. These sectors strive to maintain the highest feasible total income and increase profit. While airline businesses work to maintain their overall income as high as possible and optimise their profit, customers are looking for the best deal on their ticket. The client typically pays more or the airline firm loses money when there is a mismatch between the number of seats available and the number of passengers who want to book them. The majority of airline corporations have cutting-edge technologies and skills that allow them to manage the pricing process.

With the creation of numerous internet tools to compare prices across multiple airline companies, buyers are also getting smarter. Furthermore, the task of determining on optimal pricing is challenging for everyone due to airline competition. Anyone who has purchased a plane ticket is aware of how costs may change suddenly. The price of the least costly ticket on a specific flight decreases with time. This normally happens in an effort to increase revenue based on:

- Time of purchase patterns (making sure last-minute purchases are expensive)
- Keeping the flight as full as they want it (raising prices on a flight which is filling up in order to reduce sales and hold back inventory for those expensive last-minute expensive purchases)

So, this project involves collection of data for flight fares with other features and building a model to predict fares of flights.

1.2 Conceptual Background of the Domain Problem

According to a research, India's attractive aerospace sector is experiencing rapid growth. By 2030, India would be the largest avionics market, moving up from third place in 2020. Considering that there were only 81 million passengers in 2015, it is expected that Indian aviation traffic will surpass 100 million by 2017. According to Google, India is the country where "Cheap Air Tickets" is most frequently searched. When air travel is made available to India's white-collar class, consumers start looking for affordable options.

B. Smith, J. Leimkuhler, R. Darrow, and Samuels, “Yield management at American airlines, “Interfaces, vol. 22, pp. 8–31, 1992 says that, Anyone who has ever purchased a plane ticket is aware of how quickly prices may change. A unique valuation method is carried out by aircraft using cutting-edge techniques known as revenue management. The least priced accessible ticket can be expensive or inexpensive depending on the time period. This valuation approach automatically changes the toll according to the time of day, such as morning, afternoon, or night. The cost may also vary with the seasons, such as winter, summer, and holiday seasons. The carrier's primary objective is to increase its revenue, whereas the buyer is always looking for the best deal. In general, buyers try to buy the ticket before the scheduled departure day.

An airline business may be able to pre-plan flights and set reasonable prices for a route with the use of early demand forecasting. Current demand prediction models often aim to forecast passenger demand for a certain flight or route as well as the market share of a specific airline. An airline firm may classify consumers according to their willingness to pay and subsequently charge them various fees. This is known as price discrimination. For instance, business clients are more likely to spend more than leisure clients since they place more value on the calibre of the service than the cost.

1.3 Review of Literature

On the airlines side, the main goal is increasing revenue and maximizing profit. According to (Narangajavana et al., 2014), airlines utilize various kinds of pricing strategies to determine optimal ticket prices: long-term pricing policies, yield pricing which describes the impact of production conditions on ticket prices, and dynamic pricing which is mainly associated with dynamic adjustment of ticket prices in response to various influencing factors.[6]

To anticipate ticket prices, Tziridis et al. used eight machine learning models, including ANNs, RF, SVM, and LR, and compared their effectiveness. The most accurate regression model had an 88% accuracy rate. The best model in their comparison is a bagging regression tree since it is reliable and unaffected by the use of various input feature sets.[7]

Up to 75% of the time, the logistic regression model is accurate. The model's conclusion is that the majority of airline ticket prices change daily. According to G.A. Papakostas, K.I. Diamantaras and T. Papadimitriou, the cost of the ticket is high for a while before gradually dropping to a particular level. The cost of the ticket starts to rise once more when the flight is more than 2-3 days away [8].

With current daily airfares provided by www.infare.com, Janssen developed an expectation model for the San Francisco to New York route using the Linear Quantile Blended Regression method. The number of days till departure and whether the flight is on a weekday or the end of the week were used as the model's two highlights. The algorithm accurately predicts airfare for days that are distant from the departure date, but for a significant amount of time close to the departure day, the prediction isn't convincing [9].

1.4 Motivation for the Problem Undertaken

The project is provided to me by Flip Robo Technologies as a part of the internship programme (Internship Batch No-31). This problem is a real world dataset. The exposure of this data gives me the opportunity to locate my skills in solving a real time problem. It is the primary motivation to solve this problem.

Early prediction of the demand along a given route could help an airline company pre-plan the flights and determine appropriate pricing for the route. In addition, competition between airlines makes the task of determining optimal pricing is hard for everyone. Here firstly web scrapped the data to make a dataset for us. Then built a ML model with respect to that dataset. The web scraping done on 8th January, 2023. The date closest to the departure date is chosen (9th January, 2023), and then some random dates within a three-month range are chosen. From February 1 to February 4, 2023, four consecutive days are also chosen to observe the range of prices for Economy, Premium Economy, and Business class.

2. Analytical Problem Framing

2.1 Mathematical/ Analytical Modelling of the Problem

First phase of problem modelling involves data scraping of flights from internet. For that purpose, flight data is scrap from www.yatra.com for different timeframe of 9th Jan 2023 to 21st April 2023. Data is scrape for flights on route of Kolkata to Bangalore. Data is scrap for Economy class, Premium Economy class & Business class flights.

Next phase is data cleaning & pre-processing for building ML Model. Our objective is to predict flight prices which can be resolve by use of regression-based algorithm. Further Hyper parameter tuning performed to build more accurate model out of best model.

2.2 Data Sources and their formats

Data is collected from www.yatra.com for timeframe of 9th Jan 2023 to 21st April 2023 using selenium and saved in CSV file. Data is scrape for flights on route of Kolkata to Bangalore. Data is scrap for Economy class, Premium Economy class & Business class flights. Around 2300 flights details are collected for this project. Later the training dataset is divided into two parts, training and testing. After determine the proper model, the model is applied to predict the target variable for the test dataset. Here the web scraping done on 8th Jan, 2023. So the earliest departure date is January 9, 2023, and the latest departure date is April 21, 2023, which is approximately 3-4 months after the date of web scraping (that is data collection date).

```
data = pd.read_excel('Final_Flight_Price_Details.xlsx')
data.sample(n=5)
```

	Airlines_name	Journey_class	Aeroplane_model	Date	Source	Destination	Departure_Time	Arrival_Time	Duration	Total_Stops	Price
1739	Vistara Business	Business	UK-774/851	Wed, 18 Jan	Kolkata	Bangalore	20:40	08:45\n+ 1 day	12h 05m	1 Stop	47158
2330	Air India Business	Business	AI-402/815	Fri, 21 Apr	Kolkata	Bangalore	10:30	18:20	7h 50m	1 Stop	60996
1312	Air India	Economy	AI-763/807	Fri, 21 Apr	Kolkata	Bangalore	06:50	20:15	13h 25m	1 Stop	12408
378	IndiGo	Economy	6E-6513	Sat, 28 Jan	Kolkata	Bangalore	08:25	11:10	2h 45m	Non Stop	8712
495	IndiGo	Economy	6E-647/605	Wed, 1 Feb	Kolkata	Bangalore	04:45	13:55	9h 10m	1 Stop	6507

```
print('No. of Rows :',data.shape[0])
print('No. of Columns :',data.shape[1])
```

```
No. of Rows : 2350
No. of Columns : 11
```

2.3 Data Pre-processing Done:

2.3.1 Feature Engineering:

' --', 'null', 'NA', ' ' are not present in the training and testing dataset.

```
data.isin(['--','null','NA',' ']).sum().any()
```

```
False
```

```
data.isnull().sum()
```

```
Airlines_name      0
Journey_class      0
Aeroplane_model    0
Date               0
Source             0
Destination        0
Departure_Time     0
Arrival_Time       0
Duration           0
Total_Stops        0
Price              0
dtype: int64
```

2.3.2 Drop unnecessary columns:

Let's drop the unnecessary column 'Source', 'Destination' from the dataset as this two has one unique value.

```
# Dropping Unnecessary columns
data.drop(columns=['Source','Destination'], inplace= True)
```

2.3.3 Conversion of Duration column from hr & Minutes format into Minutes:

By default, Duration of flights are given in format of [(hh) hours: (mm) minute] which need to convert into uniform unit of time. Here we have written code to convert duration in terms of minute. So here we convert 05 minutes into 5 minutes before starting the conversion.

```
data['Duration'] = data['Duration'].map(lambda x : x.replace('05m','5m'))
```

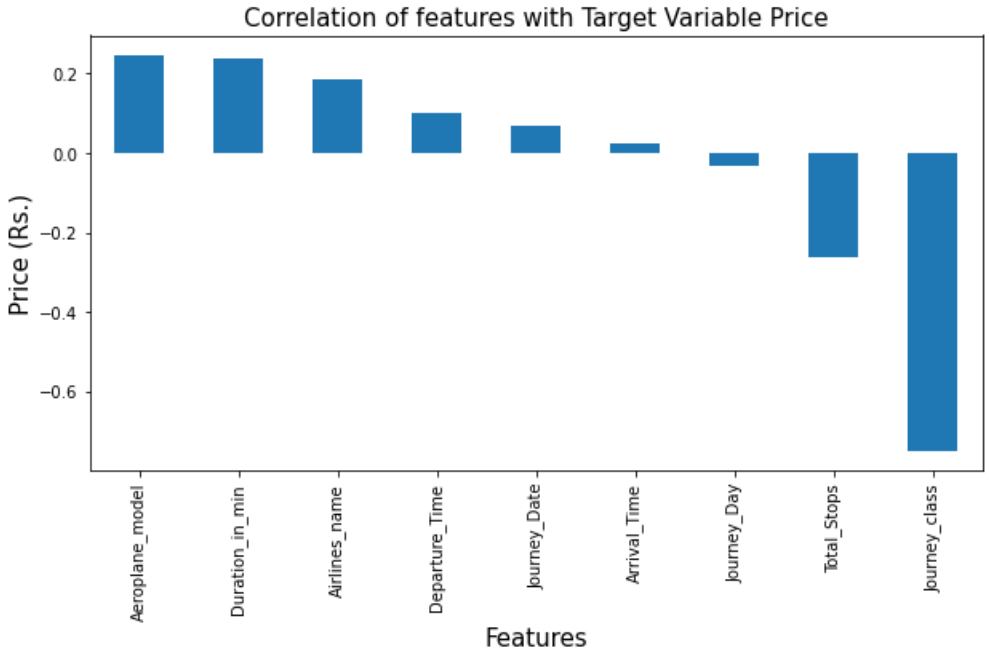
```
data['Duration_in_min'] = data['Duration'].str.replace("h", '*60').str.replace(' ','+').str.replace('m','*1').apply(eval)
data.drop(['Duration'], inplace= True, axis=1)
```

2.3.4 Create new column for day and date:

New column for 'Journey_Day' & 'Journey_Date' is extracted from Date column and then delete the Date column.

2.3.5 Correlation:

Observations of the correlation:
1. We can class is highly correlated with target variable Price.
2. Most of features are moderately & poorly correlated with each other.



2.3.6 Skewness:

1. We can see Price, Total_Stops are skewed features.
2. We cannot transform Price features as it is target variable while stops is categorical variable so concept of skewness does not applicable to it.

2.4 Data Inputs- Logic- Output Relationships

We can see in the correlation that every features are correlated with each other moderately & poorly and also they are correlated with target variable label.

2.5 State the set of assumptions (if any) related to the problem under consideration

No such assumptions are taken for this case.

2.6 Hardware and Software Requirements and Tools Used

Processor: Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz 2.00 GHz
RAM: 4.00 GB
System Type: 64-bit operating system, x64-based processor
Window: Windows 10 Pro
Anaconda – Jupyter Notebook

Libraries Used –

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split
```

For Webscraping the following libraries are used.

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
import time
import selenium
from selenium import webdriver
from selenium.common.exceptions import StaleElementReferenceException, NoSuchElementException
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import WebDriverWait
```

Except this, different libraries are used for machine learning model building from sklearn.

3. Model/s Development and Evaluation

3.1 Identification of possible problem-solving approaches (methods):

Data extraction from the www.yatra.com website is the first step in problem solving, which we have already completed. Building a machine learning model to anticipate flight prices is the next step in the problem-solving process. A regression-based machine learning approach, such as linear regression, can be used to tackle this problem. The initial task is to translate categorical variables into numerical characteristics for that purpose. Data is scaled using a standard scalar when data encoding is complete. This scaled data is used to build the final model. Data is split into training and test sets using the train test split function in the model_selection module of the Sklearn library in order to generate an ML model before applying a regression technique. The model is then trained using different regression algorithms, and 5-fold cross validation is carried out. Further Hyper parameter tuning performed to build more accurate model out of best model.

Then the best model is chosen from 7 different algorithm.

3.2 Testing of Identified Approaches (Algorithms)

Part_1 Web Scraping Strategy employed in this project as follow:

- Selenium will be used for web scraping data from www.yatra.com.
- Flights on route of Kolkata to Bangalore in duration of 9th Jan 2023 to 21st April 2023.
- Data is scrap in three parts:
    1. Economy class flight price extraction
    2. Premium Economy class flight price extraction
    3. Business class price extraction
- Selecting features to be scrap from website.
- Web scraping code executed for above mention details.
- Exporting final data in Excel file.

Total 7 algorithms used for the training and testing are:

1. Linear Regression
2. Decision Tree Regressor
3. KNN Regressor
4. Random Forest Regressor
5. SVM Regressor
6. Gradient Boosting Regressor
7. Ada Boost Regressor

3.3 Key Metrics for success in solving problem under consideration:

From metrics module of sklearn library import mean_absolute_error, mean_squared_error, r2_score. From model_selection also, we use cross_val_score. Those are the matrices use to validate the model's quality. Let's discuss every metrics shortly.

- R2 Score: R-squared (R2) is a statistical measure that shows how much of a dependent variable's variance is explained by one or more independent variables in a regression model.
- Mean Squared Error: You can determine how closely a regression line resembles a set of points using the mean squared error (MSE). This is accomplished by squaring the distances between the points and the regression line (also known as the "errors").
- Root Mean Squared Error: **Root Mean Square Error** is the measure of how well a regression line fits the data points. RMSE can also be construed as Standard Deviation in the residuals.
- Mean Absolute Error: The average discrepancy between the calculated and real values is calculated using mean absolute error. As it calculates inaccuracy in observations made on the same scale, it is also known as scale-dependent accuracy. It serves as a machine learning evaluation metric for regression models.

3.4 Run and Evaluate selected models

First find the best random state of train_test_split to get best accuracy. Here the random state is 1454. Then after splitting the data into 4 different part and check the shape of the data.

```
print('Training feature shape:',x_train.shape)
print('Training target shape:',y_train.shape)
print('Test feature shape:',x_test.shape)
print('Test target shape:',y_test.shape)
```

```
Training feature shape: (1762, 9)
Training target shape: (1762,)
Test feature shape: (588, 9)
Test target shape: (588,)
```

A Linear Regression:

```
x_train,x_test,y_train,y_test = train_test_split(x_scale,y,test_size = 0.25, random_state= 1454)

lin_reg= LinearRegression()

lin_reg.fit(x_train, y_train)

y_pred = lin_reg.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.7557632728766102
Mean absolute error: 9218.753892938974
Mean square error: 122447731.40484563
Root mean square error: 11065.610304219357
```

```
from sklearn.model_selection import GridSearchCV
grid = dict(fit_intercept=['True', 'False'], n_jobs=[1,-1])
grid_lin = GridSearchCV(estimator=lin_reg, param_grid= grid, cv=9)
grid_lin.fit(x_train, y_train)
grid_lin.best_params_
```

```
{'fit_intercept': 'True', 'n_jobs': 1}
```

```
grid_lin_best = LinearRegression(fit_intercept= 'True', n_jobs= 1)
grid_lin_best.fit(x_train, y_train)
y_pred = grid_lin_best.predict(x_test)
print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.7557632728766102
Mean absolute error: 9218.753892938974
Mean square error: 122447731.40484563
Root mean square error: 11065.610304219357
```

No such improvement seen after GridSearchCV.

B DecisionTree Regressor:

First we use decision tree and then we apply hyperparameter tuning on it. After using Gridseaech CV, R2 is slightly improved.

```
from sklearn.tree import DecisionTreeRegressor
dt = DecisionTreeRegressor()
dt.fit(x_train, y_train)
y_pred = dt.predict(x_test)
print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.9482622765368676
Mean absolute error: 2092.2976190476193
Mean square error: 25938633.147959184
Root mean square error: 5092.9984437420735
```

```
param = {'criterion' : ["squared_error", "absolute_error", "friedman_mse"], 'min_samples_split' : range(1,5),
    'splitter' : ["best", "random"], 'max_features':["auto", "sqrt", "log2"],'min_samples_leaf' : range(1,5)}
grid_search = GridSearchCV(estimator = dt,cv=5,param_grid = param)
grid_search.fit(x_train, y_train)
print("Best Parameters:" , grid_search.best_params_)
```

```
Best Parameters: {'criterion': 'squared_error', 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 4, 'splitte
r': 'best'}
```

```
grid_dt_best = grid_search.best_estimator_
grid_dt_best.fit(x_train, y_train)
y_pred = grid_dt_best.predict(x_test)
print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.9574294591488841
Mean absolute error: 2308.3312580984775
Mean square error: 21342679.347578093
Root mean square error: 4619.813778452341
```

In this way we apply hyper parameter tuning on every model. Then we get the following results after applying Hyper parameter tuning.

C KNeighbors Regressor:

For k=3 we get the best RMSE value for KNeighborsRegressor().

```
knn =KNeighborsRegressor(n_neighbors=3)
knn.fit(x_train, y_train)
y_pred = knn.predict(x_test)
print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.9491565879888462
Mean absolute error: 2859.391723356009
Mean square error: 25490271.389455784
Root mean square error: 5048.789101304964
```

```
param = {'algorithm' : ['auto', 'ball_tree', 'kd_tree'], 'leaf_size' : [30,40,25,50],
         'n_neighbors' : [3,4],'weights': ['uniform', 'distance'], 'p':[1,2,3,0]}

gridsearchknn = GridSearchCV(estimator = knn, param_grid=param, cv= 9)
gridsearchknn.fit(x_train, y_train)
print("Best Parameters:" , gridsearchknn.best_params_)
```

```
Best Parameters: {'algorithm': 'auto', 'leaf_size': 50, 'n_neighbors': 4, 'p': 1, 'weights': 'distance'}
```

```
grid_knn_best = gridsearchknn.best_estimator_
grid_knn_best.fit(x_train, y_train)
y_pred = grid_knn_best.predict(x_test)
print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.9495017301866335
Mean absolute error: 2829.272231654459
Mean square error: 25317234.845652957
Root mean square error: 5031.623480115832
```

D *Random Forest Regressor:*

R2 score, RMSE are slightly improved after GridSearch CV.

```
from sklearn.ensemble import RandomForestRegressor

rf= RandomForestRegressor()
rf.fit(x_train, y_train)
y_pred = rf.predict(x_test)
print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.9665228612913833
Mean absolute error: 2144.809149659864
Mean square error: 16783715.279334355
Root mean square error: 4096.793292238986
```

```
from sklearn.model_selection import GridSearchCV
params = {'n_estimators' : [100,110,80, 120], 'criterion' : ["squared_error", "absolute_error", "poisson"],
         'min_samples_split':[1,2,3,4],'ccp_alpha':[0,1.0,1.5,0.5]}
rf_grd = GridSearchCV(rf, param_grid = params,cv=5)
rf_grd.fit(x_train, y_train)
print('best params : ', rf_grd.best_params_)
```

```
best params :  {'ccp_alpha': 1.0, 'criterion': 'squared_error', 'min_samples_split': 2, 'n_estimators': 110}
```

```
grid_rf_best = rf_grd.best_estimator_
grid_rf_best.fit(x_train, y_train)
y_pred = grid_rf_best.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.9667869366077368
Mean absolute error: 2127.4171017682747
Mean square error: 16651321.499789609
Root mean square error: 4080.6030804024067
```

E SVM Regressor:

R2 score is very poor after using svm. So let's skip this.

```
from sklearn.svm import SVR
svr_rbf = SVR(kernel='rbf')
svr_rbf.fit(x_train, y_train)

y_pred = svr_rbf.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
```

```
R2 Score: -0.3412579736537018
Mean absolute error: 16183.352746828412
Mean square error: 672437753.4734323
```

F Gradient Boosting Regressor:

R2 score, RMSE are slightly improved after GridSearchCV.

```
from sklearn.ensemble import  GradientBoostingRegressor
gbdt= GradientBoostingRegressor()
gbdt.fit(x_train, y_train)
y_pred = gbdt.predict(x_test)
print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.9583532719724313
Mean absolute error: 2766.409916084396
Mean square error: 20879527.118925393
Root mean square error: 4569.41211961948
```

```
params = {'loss': ['squared_error', 'absolute_error'], 'n_estimators':[100,120,80],
          'criterion':['squared_error', 'mse'],'max_features': ['auto', 'sqrt']}
gbdt_grd = GridSearchCV(gbdt, param_grid = params, cv=5)
gbdt_grd.fit(x_train, y_train)
print('best params : ', gbdt_grd.best_params_)
```

```
best params :  {'criterion': 'mse', 'loss': 'squared_error', 'max_features': 'auto', 'n_estimators': 120}
```

```
grid_gbdt_best = gbdt_grd.best_estimator_
grid_gbdt_best.fit(x_train, y_train)
y_pred = grid_gbdt_best.predict(x_test)
print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.9591788090160343
Mean absolute error: 2737.6473000339356
Mean square error: 20465645.311015382
Root mean square error: 4523.897137536991
```

G AdaBoost Regressor:

R2 score, RMSE are slightly improved after GridSearchCV.

```
from sklearn.ensemble import AdaBoostRegressor

ada= AdaBoostRegressor()
ada.fit(x_train, y_train)

y_pred = ada.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.9198177075141332
Mean absolute error: 4457.388851686235
Mean square error: 40199276.8629511
Root mean square error: 6340.289966787884
```

```
params = { 'loss' : ['linear', 'square'], 'learning_rate': [0.1,0.001,1,0.01] ,'n_estimators':[50,60,40] }
ada_grd = GridSearchCV(ada, param_grid = params, cv=7)
ada_grd.fit(x_train, y_train)
print('best params : ', ada_grd.best_params_)
```

```
best params :  {'learning_rate': 0.1, 'loss': 'linear', 'n_estimators': 40}
```

```
grid_ada_best = ada_grd.best_estimator_
grid_ada_best.fit(x_train, y_train)
y_pred = grid_ada_best.predict(x_test)

print('R2 Score:', r2_score(y_test, y_pred))
print('Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('Mean square error:', mean_squared_error(y_test, y_pred))
print('Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
R2 Score: 0.9319404686639019
Mean absolute error: 3939.3634616158565
Mean square error: 34121548.01915585
Root mean square error: 5841.3652530171275
```

As per 7 different regression model we can see that the model with maximum R2 score and minimum RMSE value is GradientBoostingRegressor() and RandomForestRegressor(). Let's check the cross validation score before final prediction.

3.5 Cross Validation:

```
from sklearn.model_selection import cross_val_score

all_models = [lin_reg , grid_dt_best , grid_knn_best , grid_rf_best , grid_gbdt_best, grid_ada_best]
for i in all_models:
    cvscore = cross_val_score(i, x_scale,y, cv = 5)
    print('Cross Validation Score of :',i)
    print("\n Cross Validation Score : " ,cvscore)
    print("\nMean CV Score :",cvscore.mean())
    print("\nStd deviation :",cvscore.std())
    print("\n-----------")
    print("-----------")
```

Here, the cross validation score is very poor for all of the different models. But Gradient Boosting Regressor gives better CV Score compare to others. So let's take it.

3.6 Overfitting checking:

```
from sklearn.linear_model import Ridge, Lasso, RidgeCV, LassoCV
#LASSO Regression
lassocv= LassoCV()
lassocv.fit(x_train, y_train)
alpha= lassocv.alpha_   #best learning rate for LASSO
alpha
```

```
16.305806173640438
```

```
lasso_reg=Lasso(alpha)
lasso_reg.fit(x_train,y_train)
print ("Score after applying LASSO regression on the model is :", lasso_reg.score(x_test, y_test))
```

```
Score after applying LASSO regression on the model is : 0.7556291216144208
```

For Lasso and Ridge CV, the score is quite similar and around 75%. This dataset is free from overfitting problem.

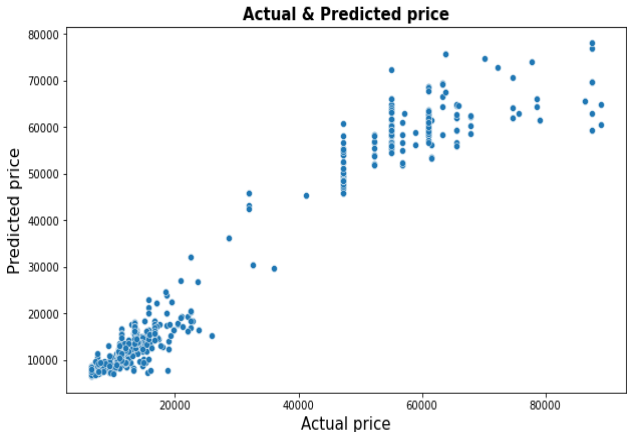
3.7 Final Model:

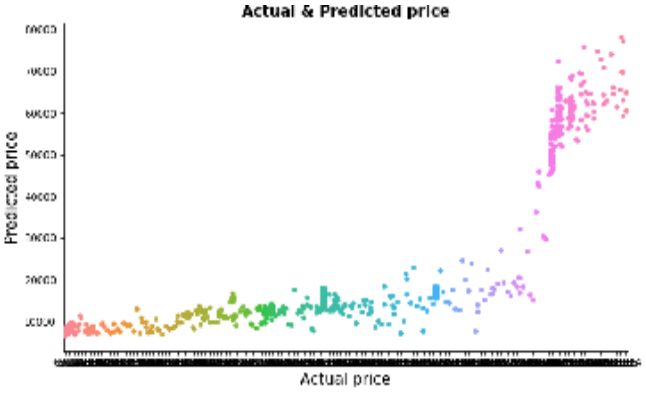
Here Gradient Boosting Regressor is the best model for this dataset.

```
print('Final R2 Score:', r2_score(y_test, y_pred))
print('\nFinal Mean absolute error:', mean_absolute_error(y_test, y_pred))
print('\nFinal Mean square error:', mean_squared_error(y_test, y_pred))
print('\nFinal Root mean square error:',np.sqrt( mean_squared_error(y_test, y_pred)))
```

```
Final R2 Score: 0.9591788090160343
Final Mean absolute error: 2737.6473000339356
Final Mean square error: 20465645.311015382
Final Root mean square error: 4523.897137536991
```

3.8 Visualize the variation of actual test data and predicted data:

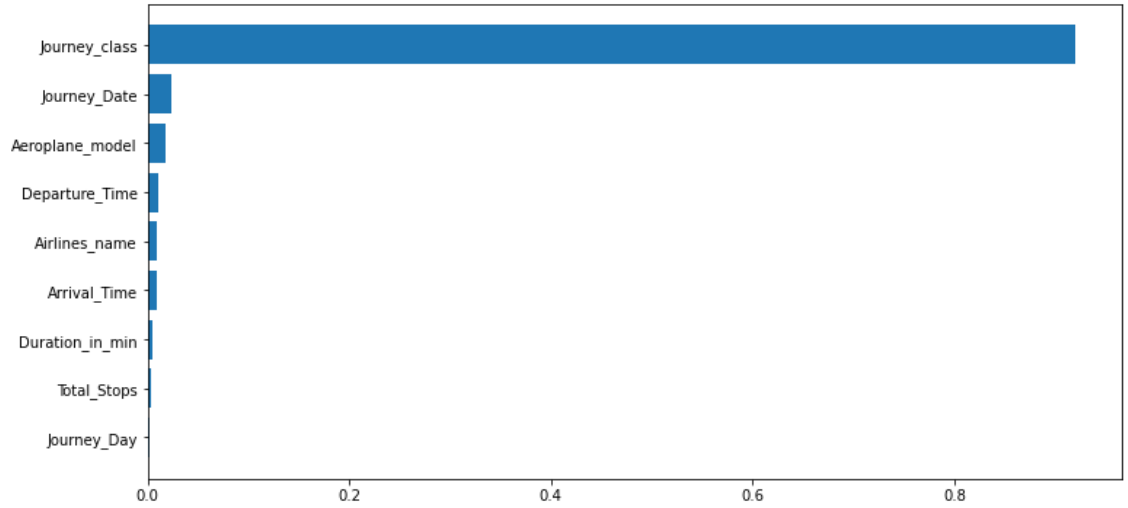




3.9 Important Feature Selection:

The observations are as followes:

1. Journey class is the most important feature for predicting price.
2. Total duration of journey,total stops, journey day are the less importance feature.



3.10 Load and save the model:

Let's save the model using pickle for future use. Then see the actual and predicted value of 6 random sample.

```
import pickle
pickle.dump(grid_gbdt_best, open("Flight_Price_Regression_model", "wb"))
```

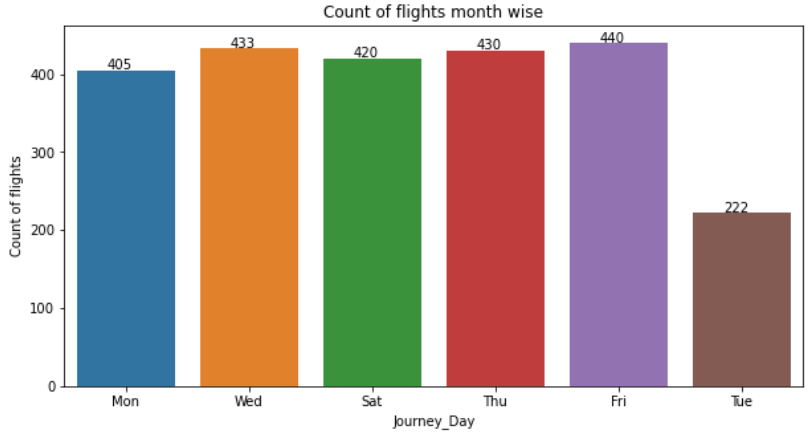
Let's see the actual and predicted data in a tabular manner.

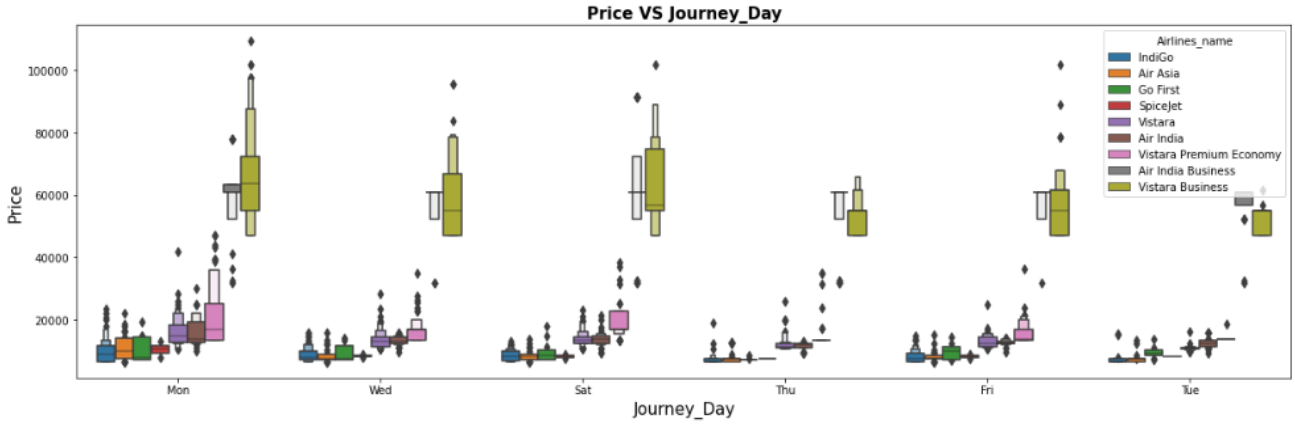
```
y_pred = load_Flight_Price_Regression_model.predict(x_test)

y_test = np.array(y_test)
data_prediction_by_model = pd.DataFrame()
data_prediction_by_model["Predicted Values"] = y_pred.round(0)
data_prediction_by_model["Actual Values"] = y_test
data_prediction_by_model.sample(n=5)
```

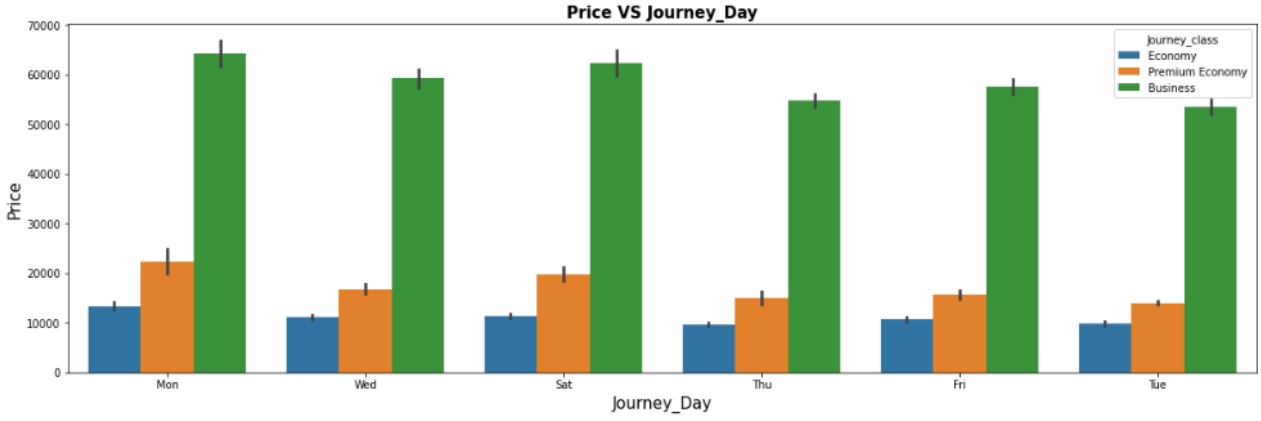
	Predicted Values	Actual Values
446	12459.0	13563
90	13933.0	13227
483	14517.0	15740
64	13766.0	15401
505	59404.0	54975

3.11 Visualizations:

Let's start the observation exploration of feature analysis.

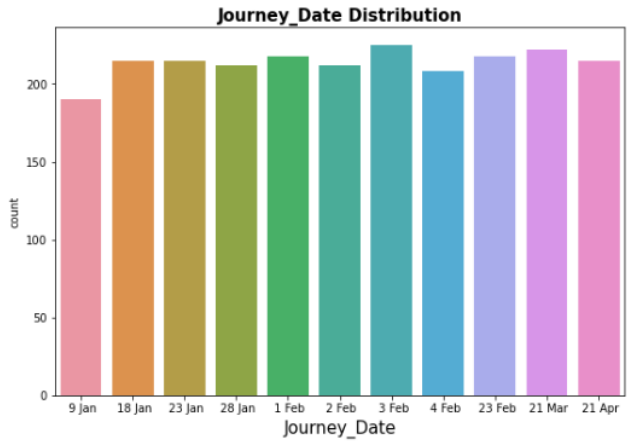


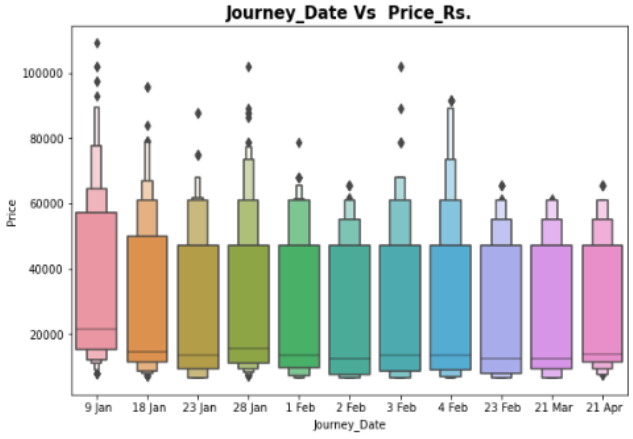




Observations:

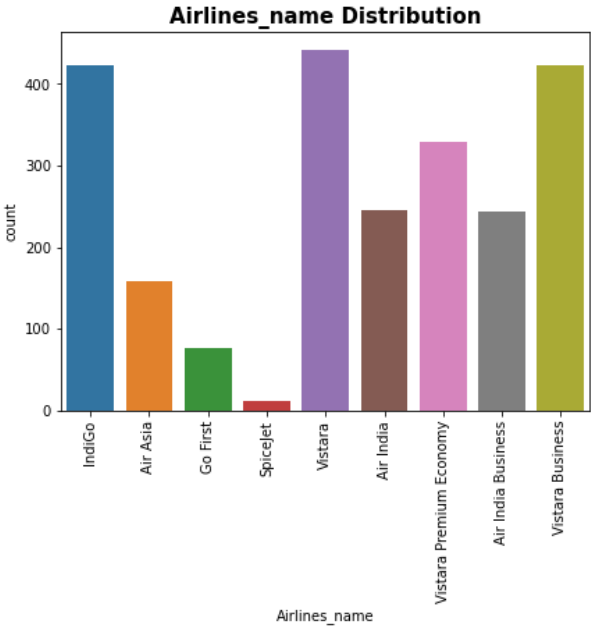
1. On Tuesday minimum flights run while on Friday maximum flights run.
2. Eveyday all of the three types of flights are running and very obviously business class's cost is high than other two for each days.
3. Maximum Avg. Fare for Business Flights is on Monday while minimum Avg. Fare for Business flights on Tuesday.
4. For Ecomony Flights & Premium Ecomony Flights: Minimum Avg. Fare on Thusday.
5. For Ecomony Flights & Premium Ecomony Flights: Maximum Avg. Fare on Monday.

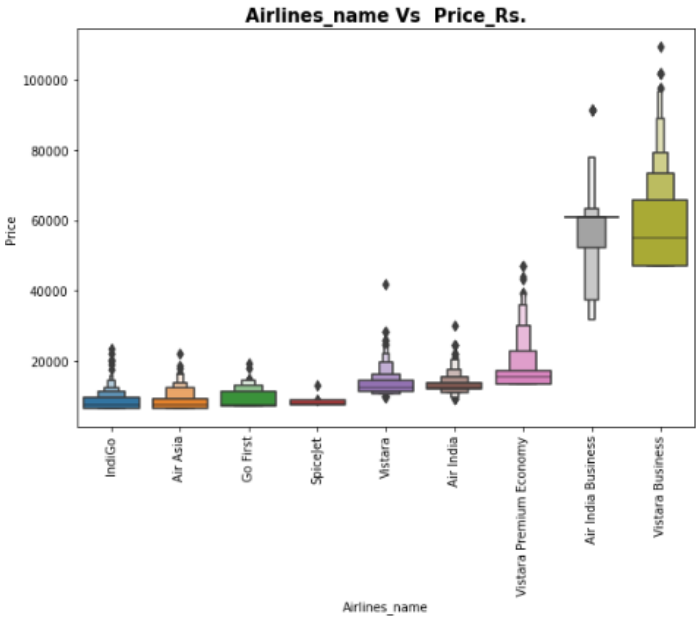




Observations:

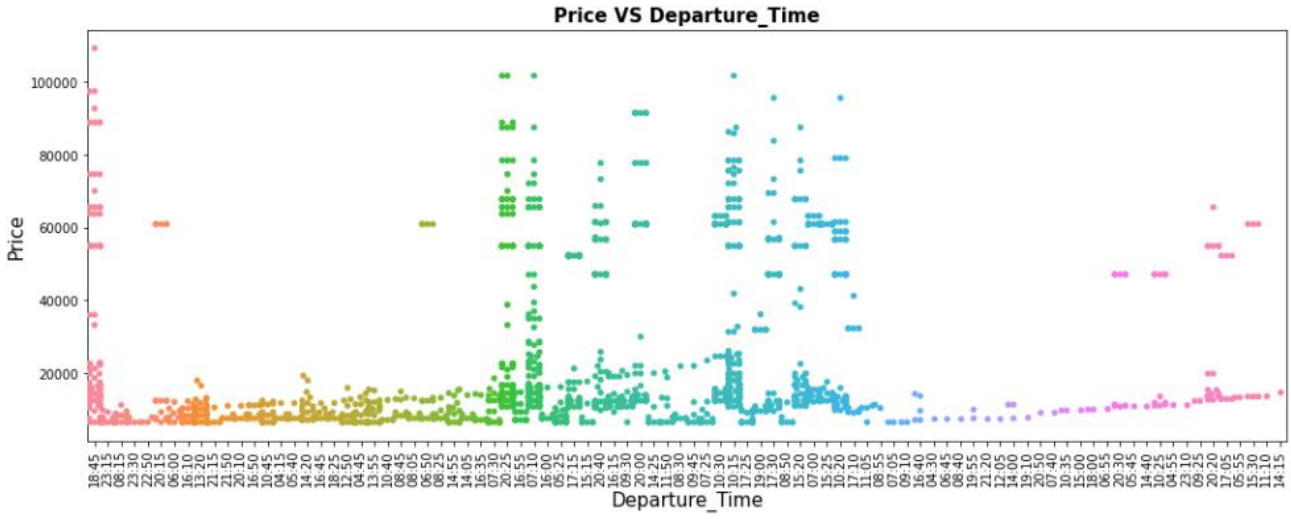
1. We have scrapped overall same percent of data in this big dataset.
2. All the data scrapped on 8th January, 2023. So for 9th January the rate is maximum and for March and April the price is least than earlier.

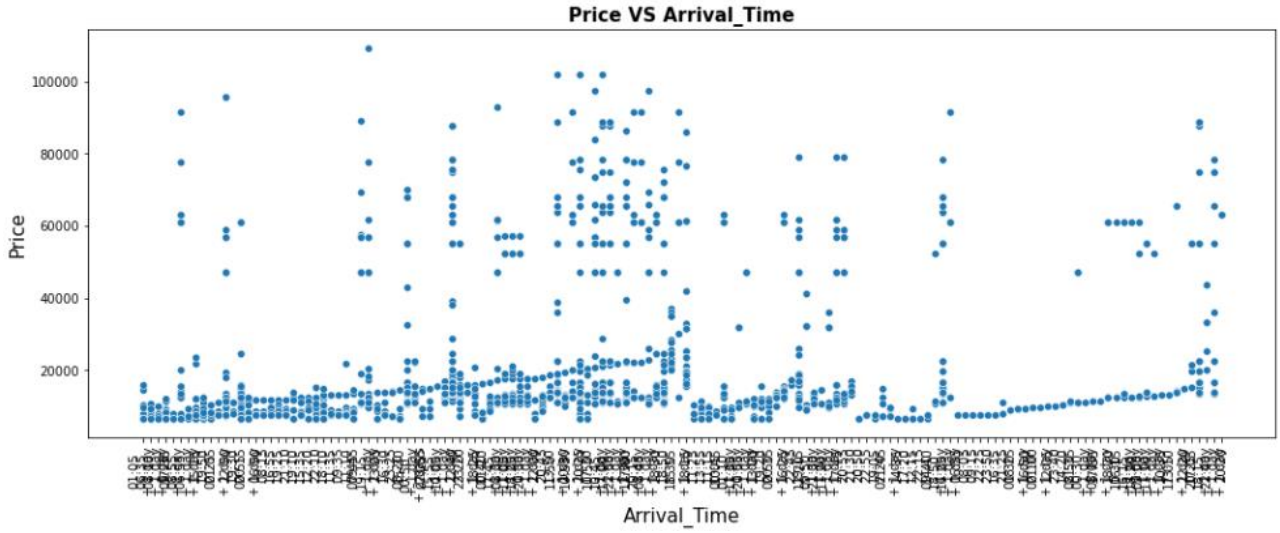




Observations:

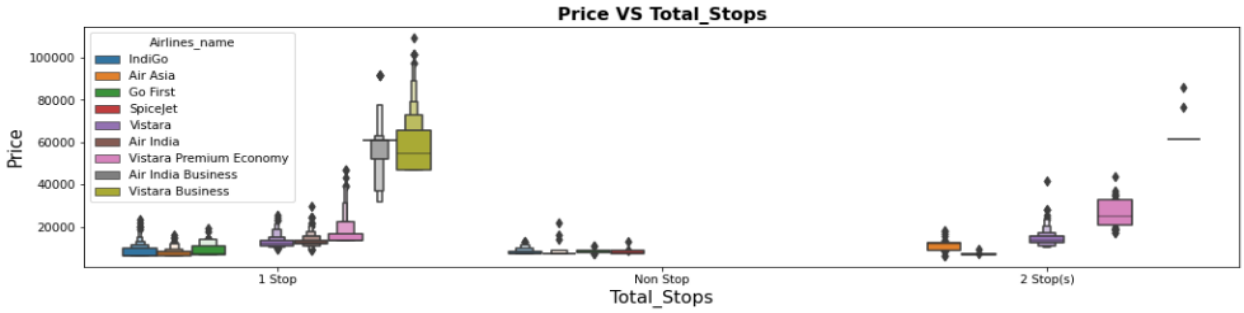
1. We can see maximum number of flights run by Vistara Economy while minimum Flights run by Spicejet.
2. Around 28% of flights of Business Class.
3. For Economy class, SpiceJet flight is cheaper than Vistara and Air India.





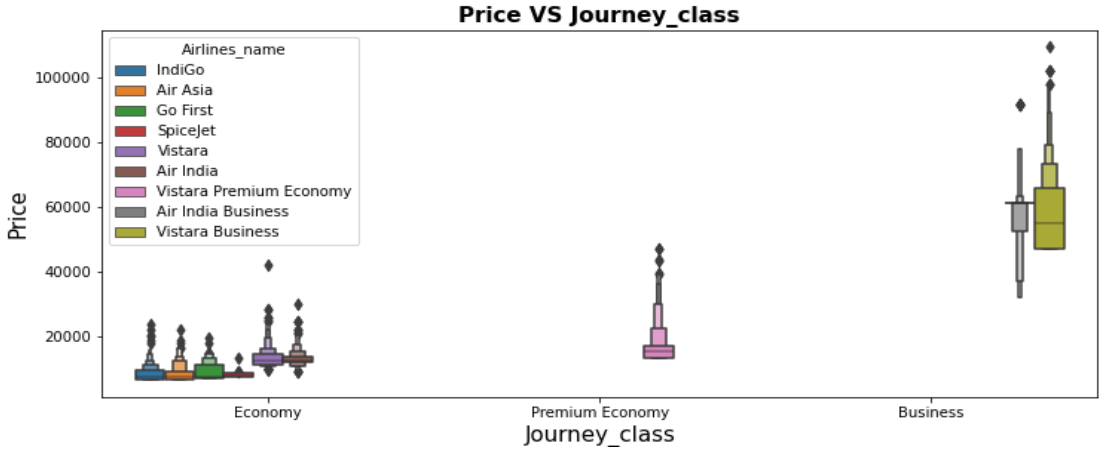
Observations:

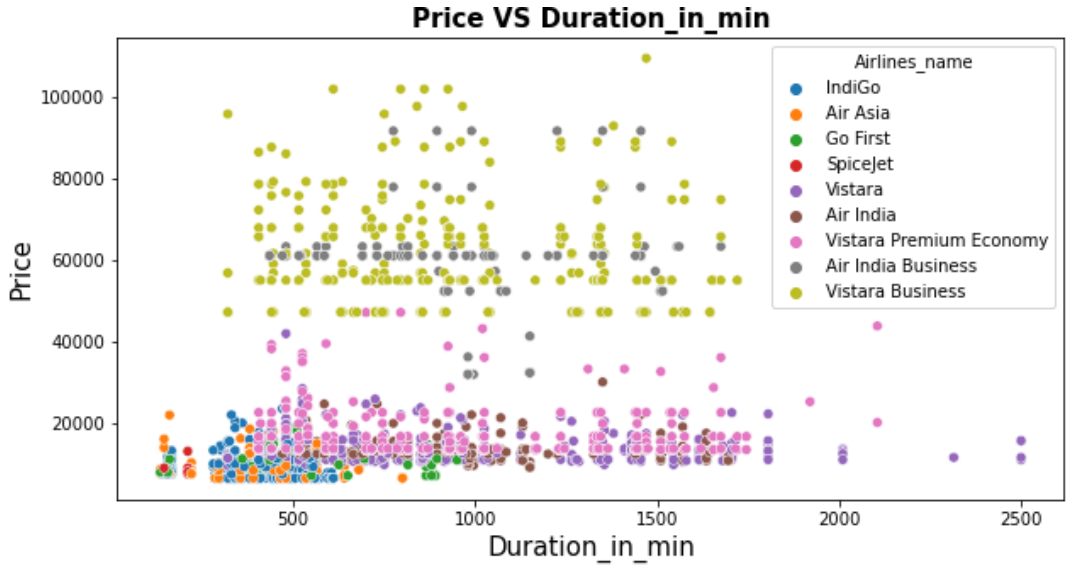
1. There is no such relation between Departure_Time and number of flights. Though maximum flights departed at 10.15 am.
2. No such relation between Arrival_Time and Departure_Time with Price of the flight.



Observations:

1. Around 82% flights are 1 stop from kolkata to Bangalore. Obviously these flights have high flight duration compare to Non-stop Flight
2. Only 7.3% of flights do not have any stop in there route.
3. Most of the cases, Vistara Business class flight is 1 stop and its price is also high than others.

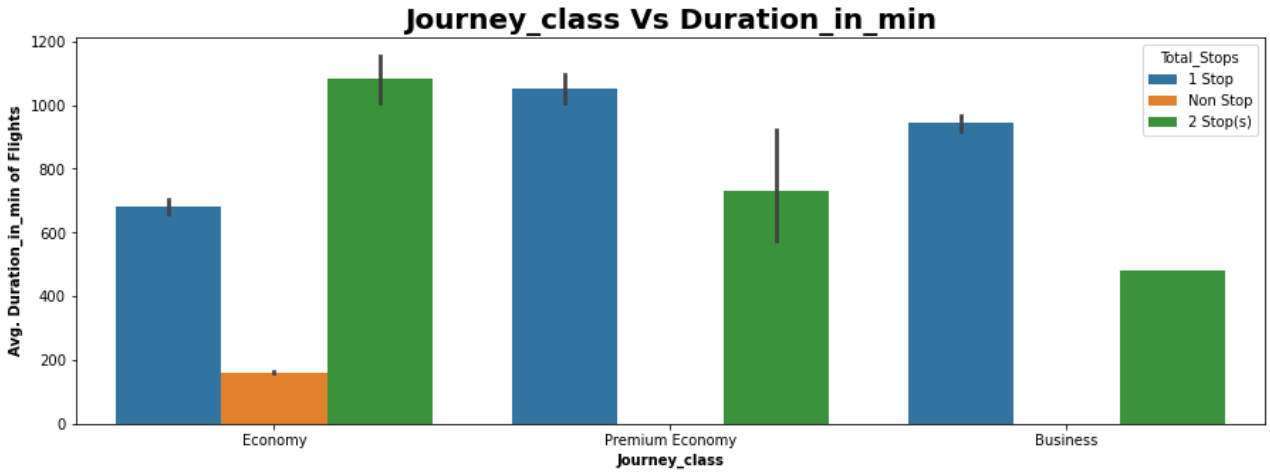


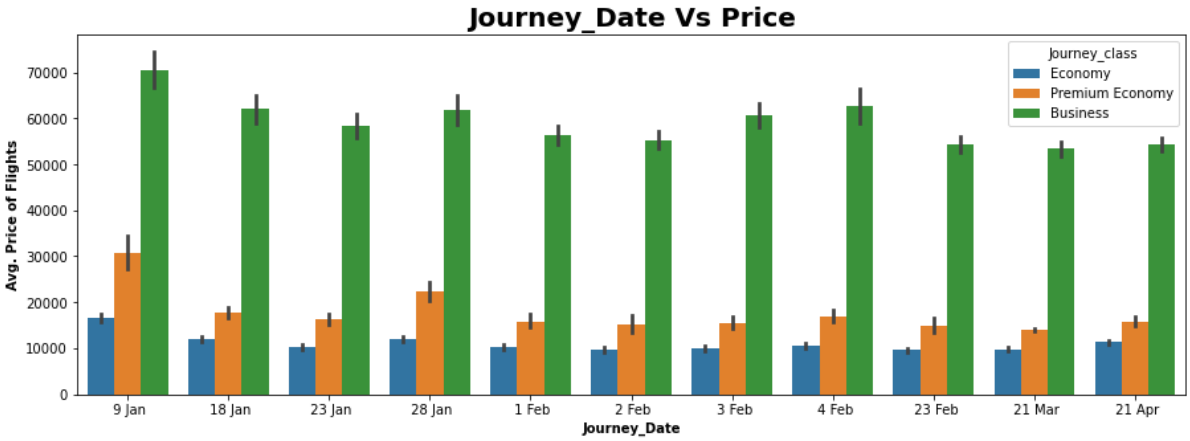


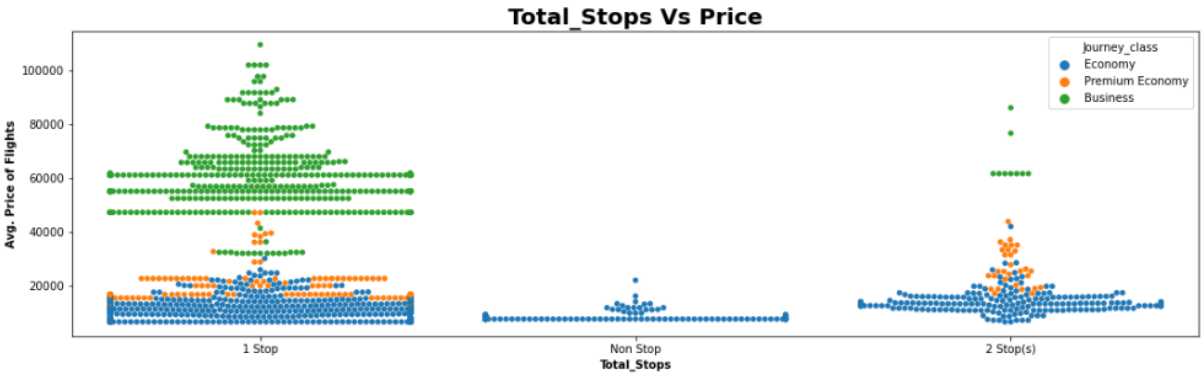
Observations:

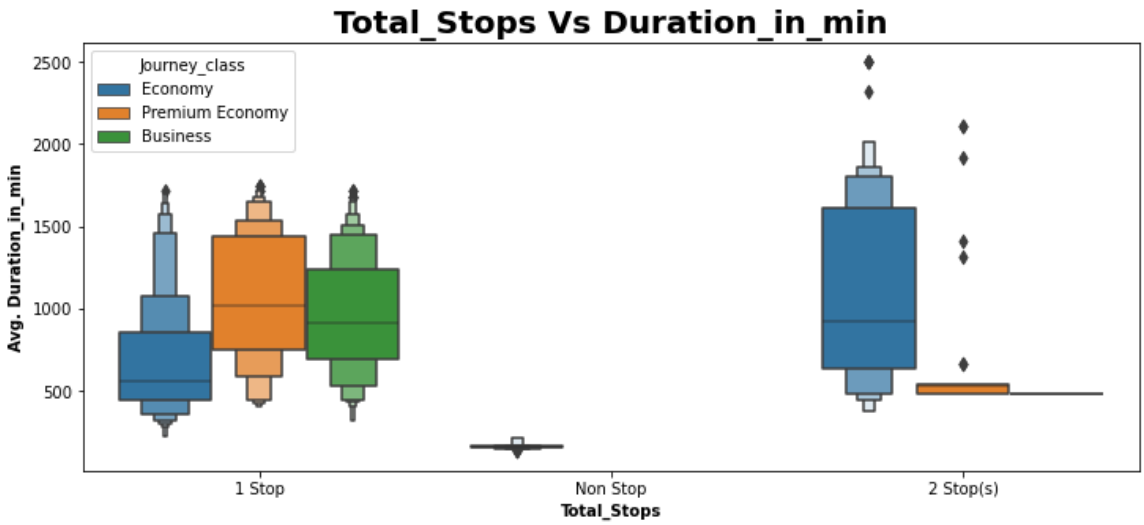
1. Outliers are present in the Duration dataset.
2. No such relation between duration and price of the flight.
3. 57.7% flights are of Ecomony class, as they are low cost of flight & most of people prefer it.
4. There are more business class flights than Premuim Ecomony flights. It strange because Business class is costlier than Premuim Ecomony class.

Multivariate Analysis (Relation between different features of the dataset):









Observations:

1. As number of Stops increase the duration of flights increases for economy class.
2. For Premium economy or Business class there are no nonstop flights.
3. Very surprisingly for Business class the duration of 1 stop flight is more than 2 stops.
4. For both maximum and minimum duration of flight, the flight type is Economy class.
5. For four consecutive day in February 2023 (1st Feb to 4th Feb), maximum average price is for 4th February 2023, Saturday.
6. For every stops, maximum average price is for Business class and minimum for economy class.
7. Every day, between three differnt journey class, maximum duration is for Premium Economy class.
8. There is no non stop flight for Business and Premium Economy class.

3.12 Interpretation of the Results:

After all the pre-processing steps, the dataset is ready to train machine learning models. All unnecessary words from comment text are deleted as they might give overfitting problem as well as it also could increase the time complexity. Now apply this dataset on different ML Classification Model (as discussed on part 3.4 - 'Run and Evaluate selected models') and check the best model for this particular dataset.

4. CONCLUSION

4.1 Key Findings and Conclusions of the Study

Finally Gradient Boosting Regressor and Random Forest Regressor giving us maximum R2 Score and minimum RMSE Value. Here, the cross validation score is very poor for all of the 6 different models. But Gradient Boosting Regressor gives better CV Score compare to others. So let's take it as our final model.

4.2 Limitations of this work and Scope for Future Work

Additionally, the following studies are examples that might be taken into account for future work in this field:

- In this study we focus on flights on route of Kolkata to Bangalore, more route can incorporate in this project to extend it beyond present investigation.
- For all 6 ML models, we receive a subpar cross-validation score in this case. This study focuses on a small number of random days. If more dates could be included or more other features were used, this CV score might have improved.
- Time series analysis can be performed over this model.