Paytm Flight Price – Web Scraper & Price Prediction Using Python & Machine Learning

❖ Introduction:

Nowadays, airline ticket prices can vary dynamically and significantly for the same flight, even for nearby seats within the same cabin. Customers are seeking to get the lowest price while airlines are trying to keep their overall revenue as high as possible and maximize their profit. Airlines use various kinds of computational techniques to increase their revenue such as demand prediction and price discrimination. From the customer side, two kinds of models are proposed by different researchers to save money for customers: models that predict the optimal time to buy a ticket and models that predict the minimum ticket price. In this project, i present a review of customer side and airlines side prediction models. My analysis shows that models on both sides rely on limited set of features such as historical ticket price data, ticket purchase date and departure date.

❖ Problem statement:

The objective of this project is to predict the price of flight ticket based on criteria. The use case of this project is to study those factors that play an important role to fluctuate flight price.

In this study, Data is collected by web scraping from paytm.com. Paytm provide an interface to customers to make a decision to buy flight ticket.

❖ About Dataset:

	Flight name	Departure_time	Arrival_time	Price	Source	Destination	Duration	No of stop	Date
0	SpiceJet	21:45	23:45	7308	Mumbai	Delhi	2h	Non	2022-03-22
1	IndiGo	17:15	19:20	7308	Mumbai	Delhi	2h 5m	Non	2022-03-22
2	IndiGo	18:30	20:35	7308	Mumbai	Delhi	2h 5m	Non	2022-03-22
3	IndiGo	20:30	22:35	7308	Mumbai	Delhi	2h 5m	Non	2022-03-22
4	IndiGo	22:20	00:25	7308	Mumbai	Delhi	2h 5m	Non	2022-03-22

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9598 entries, 0 to 9597
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   Flight name     9598 non-null   object
 1   Departure_time  9598 non-null   object
 2   Arrival_time    9598 non-null   object
 3   Price           9598 non-null   int64 
 4   Source          9598 non-null   object
 5   Destination     9598 non-null   object
 6   Duration        9598 non-null   object
 7   No of stop      9598 non-null   object
 8   Date            9598 non-null   object
dtypes: int64(1), object(8)
memory usage: 675.0+ KB
```

❖ Descriptive Statistical Analysis:

	Duration	Journey_Day	Journey_Month	Arrival_Hours	Arrival_Minutes	Dep_Hours	Dep_Minutes	Price
count	9598.000000	9598.000000	9598.000000	9598.000000	9598.000000	9598.000000	9598.000000	9598.000000
mean	9.000834	16.496874	4.371327	14.554074	28.910190	12.975099	24.540529	9441.779225
std	5.872988	9.580638	0.881578	6.426518	18.049203	5.563122	18.436387	3631.258119
min	0.000000	1.000000	3.000000	0.000000	0.000000	0.000000	0.000000	3077.000000
25%	4.166667	7.000000	4.000000	9.000000	15.000000	8.000000	5.000000	6984.000000
50%	8.333333	15.000000	4.000000	16.000000	30.000000	13.000000	25.000000	9198.000000
75%	13.000000	23.000000	5.000000	20.000000	45.000000	18.000000	40.000000	11209.000000
max	23.916667	30.000000	6.000000	23.000000	55.000000	23.000000	55.000000	32322.000000

Outcome:

✓ Analysis clearly describes that the Outliers and skewness is present in the target variable.

❖ Data Pre-processing:

- Data pre-processing includes:
  - **Treating null values:** There is no null values in dataset.
  - **Data Cleaning:** Cleaned the data like converting the datatype of those features that must be in numeric datatype by using **pandas function**. Dropped the unnecessary and unwanted columns from the data frame by using **drop ()**.
  - **Features extraction:** like column 'Departure Time', 'Arrival Time', 'Price', 'Duration', and 'Date', need to extract different important values into other columns like 'Dept_Hours', 'Dept_Minutes', 'Arrival_Hours', 'Arrival_Minutes', 'Time Duration of Flight', 'Journey Date', 'Journey Month', 'Journey Year' by taking **time delta** and **to_datetime** function of pandas dataframe.

❖ Features Engineering:

- Dealing with outlier's and skewness:
  - Outliers & Skewness is present in 'Price' which is target variable.
- Encoding categorical column: As machine learning model understands numerical values or we can say that for a good model we need data in numerical form. To deal with this, machine learning has different type of encoder which converts object type of columns/variable into numeric by encoding them or we can handle it by replace/custom mapping.
  - Few encoding techniques are:
    - Label Encoding
    - Ordinal Encoding

    - One Hot Encoding
    - Dummy Encoding
    - Binary Encoding
  - Label encoder is used for encoding categorical variable.

- Scaling the independent variable: Scaling is a method to normalize independent variable. In machine learning different technique are available to normalize the data.
  - Few techniques used for scaling:
    - Absolute Maximum Scaling
    - Min-Max Scaling
    - Standardization
  - After selecting X and y, Standard Scaler is used to scale independent variable.

- Dealing with multicollinearity: The best way to deal with multicollinearity is VIF. VIF is variance inflation factor which is member of stats.outliers_influence of statsmodels.
  - For each feature calculate VIF score and try to drop those independent features which have highest VIF. Check VIF again.
  - The standard threshold for VIF is 5. However, sometimes it depends upon the dataset.

❖ Libraries:

- numpy
- pandas
- matplotlib
- seaborn
- model_selection: train_test_split, cross_val_score, GridSearchCV
- datetime
- preprocessing : StandardScaler
- score: r2 score, MSE score, RMSE score

❖ Testing of Identified Approaches (Algorithms)

As the problem is regression problem. In machine learning there are several algorithms for regression problem. I used total 9 algorithm to build best model on this dataset. The algorithms are:

- Linear Regression
- K-Neighbors Regressor

- Decision Tree Regressor
- Random Forest Regressor
- Support Vector Machine Regressor
- Bagging Regressor
- Gradient Boosting Regressor
- AdaBoost Regressor
- XGB Regressor

❖ Run and evaluate selected models

Here are the coding and insights of all models:

▪ Linear Regression

```
lr = LinearRegression()
lr.fit(X_train,y_train)
score_pred_lr = lr.score(X_test,y_test)*100
y_pred_lr = lr.predict(X_test)
mse_lr=mean_squared_error(y_test,y_pred_lr)
score_lr=np.sqrt(mse_lr)
r2_score_lr = r2_score(y_test,y_pred_lr)
print('The score by Linear Regression on test set is :',score_pred_lr)
print('The MSE score is :',mse_lr)
print('The RMSE score is :',score_lr)
print('The r2 score is :',r2_score_lr)
```

```
The score by Linear Regression on test set is : 35.62018035637657
The MSE score is : 8508996.088078374
The RMSE score is : 2917.0183558007266
The r2 score is : 0.35620180356376574
```

Cross Validation on Linear Regression Model

```
score=cross_val_score(lr,X_train,y_train,cv=10,scoring='neg_mean_squared_error')
score_cross_lr=np.sqrt(-score)
print('The cross validation score :',np.mean(score_cross_lr),np.std(score_cross_lr))
```

```
The cross validation score : 2963.7476998003885 109.58516267317867
```

▪ K-Neighbors Regressor

```
kn = KNeighborsRegressor()
kn.fit(X_train,y_train)
score_pred_kn = kn.score(X_test,y_test)*100
y_pred_kn = kn.predict(X_test)
mse_kn=mean_squared_error(y_test,y_pred_kn)
score_kn=np.sqrt(mse_kn)
r2_score_kn = r2_score(y_test,y_pred_kn)
print('The score by K-Neighbors Regressor on test set is :',score_pred_kn)
print('The MSE score is :',mse_kn)
print('The RMSE score is :',score_kn)
print('The r2 score is :',r2_score_kn)
```

```
The score by K-Neighbors Regressor on test set is : 73.50935368353765
The MSE score is : 3501233.882375
The RMSE score is : 1871.1584332640034
The r2 score is : 0.7350935368353766
```

Cross Validation K-Neighbors Regressor

```
score=cross_val_score(kn,X_train,y_train,cv=10,scoring='neg_mean_squared_error')
score_cross_kn=np.sqrt(-score)
print('The cross validation score :',np.mean(score_cross_kn),np.std(score_cross_kn))
```

```
The cross validation score : 2078.484649312992 129.8347309129714
```

Decision Tree Regressor

```
dt = DecisionTreeRegressor()
dt.fit(X_train,y_train)
score_pred_dt = dt.score(X_test,y_test)*100
y_pred_dt = dt.predict(X_test)
mse_dt=mean_squared_error(y_test,y_pred_dt)
score_dt=np.sqrt(mse_dt)
r2_score_dt = r2_score(y_test,y_pred_dt)
print('The score by  Decision TreeRegressor on test set is :',score_pred_dt)
print('The MSE score is :',mse_dt)
print('The RMSE score is :',score_dt)
print('The r2 score is :',r2_score_dt)
```

```
The score by  Decision TreeRegressor on test set is : 79.02658850331356
The MSE score is : 2772028.212673611
The RMSE score is : 1664.940903658028
The r2 score is : 0.7902658850331357
```

Cross Validation Decision Tree Regressor

```
score=cross_val_score(dt,X_train,y_train,cv=10,scoring='neg_mean_squared_error')
score_cross_dt=np.sqrt(-score)
print('The cross validation score :',np.mean(score_cross_dt),np.std(score_cross_dt))
```

```
The cross validation score : 1825.7452717499116 101.63773529263787
```

Random Forest Regressor

```
rf = RandomForestRegressor()
rf.fit(X_train,y_train)
score_pred_rf = rf.score(X_test,y_test)*100
y_pred_rf = rf.predict(X_test)
mse_rf=mean_squared_error(y_test,y_pred_rf)
score_rf=np.sqrt(mse_rf)
r2_score_rf = r2_score(y_test,y_pred_rf)
print('The score by  Random Forest Regressor on test set is :',score_pred_rf)
print('The MSE score is :',mse_rf)
print('The RMSE score is :',score_rf)
print('The r2 score is :',r2_score_rf)
```

```
The score by  Random Forest Regressor on test set is : 88.8337972761411
The MSE score is : 1475822.3278964427
The RMSE score is : 1214.8342800137157
The r2 score is : 0.888337972761411
```

Cross Validation Random Forest Regressor

```
score=cross_val_score(rf,X_train,y_train,cv=10,scoring='neg_mean_squared_error')
score_cross_rf=np.sqrt(-score)
print('The cross validation score :',np.mean(score_cross_rf),np.std(score_cross_rf))
```

```
The cross validation score : 1419.4787291126345 61.21270034863392
```

Support Vector Machine Regressor

```
sr = SVR()
sr.fit(X_train,y_train)
score_pred_sr = sr.score(X_test,y_test)*100
y_pred_sr = sr.predict(X_test)
mse_sr=mean_squared_error(y_test,y_pred_sr)
score_sr=np.sqrt(mse_sr)
r2_score_sr = r2_score(y_test,y_pred_sr)
print('The score by  Support Vector Machine Regressor on test set is :',score_pred_sr)
print('The MSE score is :',mse_sr)
print('The RMSE score is :',score_sr)
print('The r2 score is :',r2_score_sr)
```

```
The score by  Support Vector Machine Regressor on test set is : 3.1511902870367425
The MSE score is : 12800379.80137886
The RMSE score is : 3577.761842462248
The r2 score is : 0.031511902870367425
```

Cross Validation On Support Vector Machine Regressor

```
score=cross_val_score(sr,X_train,y_train,cv=10,scoring='neg_mean_squared_error')
score_cross_sr=np.sqrt(-score)
print('The cross validation score :',np.mean(score_cross_sr),np.std(score_cross_sr))
```

```
The cross validation score : 3577.5230351601226 128.47422336768145
```

- Bagging Regressor

```
br = BaggingRegressor()
br.fit(X_train,y_train)
score_pred_br = br.score(X_test,y_test)*100
y_pred_br = br.predict(X_test)
mse_br=mean_squared_error(y_test,y_pred_br)
score_br=np.sqrt(mse_br)
r2_score_br = r2_score(y_test,y_pred_br)
print('The score by  Bagging Regressor on test set is :',score_pred_br)
print('The MSE score is :',mse_br)
print('The RMSE score is :',score_br)
print('The r2 score is :',r2_score_br)
```

```
The score by  Bagging Regressor on test set is : 86.84329703200208
The MSE score is : 1738904.1272002324
The RMSE score is : 1318.6751408896098
The r2 score is : 0.8684329703200209
```

Cross Validation on Bagging Regressor

```
score=cross_val_score(br,X_train,y_train,cv=10,scoring='neg_mean_squared_error')
score_cross_br=np.sqrt(-score)
print('The cross validation score :',np.mean(score_cross_br),np.std(score_cross_br))
```

```
The cross validation score : 1516.4683210235528 78.67065104596101
```

- Gradient Boosting Regressor

```
gr = GradientBoostingRegressor()
gr.fit(X_train,y_train)
score_pred_gr = gr.score(X_test,y_test)*100
y_pred_gr = gr.predict(X_test)
mse_gr=mean_squared_error(y_test,y_pred_gr)
score_gr=np.sqrt(mse_gr)
r2_score_gr = r2_score(y_test,y_pred_gr)
print('The score by  Gradient Boosting Regressor on test set is :',score_pred_gr)
print('The MSE score is :',mse_gr)
print('The RMSE score is :',score_gr)
print('The r2 score is :',r2_score_gr)
```

```
The score by  Gradient Boosting Regressor on test set is : 77.6244594921053
The MSE score is : 2957345.759963975
The RMSE score is : 1719.693507565803
The r2 score is : 0.776244594921053
```

Corss Validation on Gradient Boosting Regressor

```
score=cross_val_score(gr,X_train,y_train,cv=10,scoring='neg_mean_squared_error')
score_cross_gr=np.sqrt(-score)
print('The cross validation score :',np.mean(score_cross_gr),np.std(score_cross_gr))
```

```
The cross validation score : 1860.797738726719 101.33041095683322
```

- AdaBoost Regressor

```
ar = AdaBoostRegressor()
ar.fit(X_train,y_train)
score_pred_ar = ar.score(X_test,y_test)*100
y_pred_ar = ar.predict(X_test)
mse_ar=mean_squared_error(y_test,y_pred_ar)
score_ar=np.sqrt(mse_ar)
r2_score_ar = r2_score(y_test,y_pred_ar)
print('The score by  Ada Boost Regressor on test set is :',score_pred_ar)
print('The MSE score is :',mse_ar)
print('The RMSE score is :',score_ar)
print('The r2 score is :',r2_score_ar)
```

```
The score by  Ada Boost Regressor on test set is : 23.608939657694272
The MSE score is : 10096505.973688599
The RMSE score is : 3177.499956520629
The r2 score is : 0.23608939657694272
```

Corss Validation on Ada Boost Regressor

```
score=cross_val_score(ar,X_train,y_train,cv=10,scoring='neg_mean_squared_error')
score_cross_ar=np.sqrt(-score)
print('The cross validation score :',np.mean(score_cross_ar),np.std(score_cross_ar))
```

```
The cross validation score : 3125.906779886339 157.83758845387365
```

- XGB Regressor

```
xr = xb.XGBRegressor()
xr.fit(X_train,y_train)
score_pred_xr = xr.score(X_test,y_test)*100
y_pred_xr = xr.predict(X_test)
mse_xr=mean_squared_error(y_test,y_pred_xr)
score_xr=np.sqrt(mse_xr)
r2_score_xr = r2_score(y_test,y_pred_xr)
print('The score by  XGB Boost Regressor on test set is :',score_pred_xr)
print('The MSE score is :',mse_xr)
print('The RMSE score is :',score_xr)
print('The r2 score is :',r2_score_xr)
```

```
The score by  XGB Boost Regressor on test set is : 88.33015699271559
The MSE score is : 1542387.8017543806
The RMSE score is : 1241.9290647031257
The r2 score is : 0.8833015699271559
```

Corss Validation on XGB Regressor

```
score=cross_val_score(xr,X_train,y_train,cv=10,scoring='neg_mean_squared_error')
score_cross_xr=np.sqrt(-score)
print('The cross validation score :',np.mean(score_cross_xr),np.std(score_cross_xr))
```

```
The cross validation score : 1397.1167559400465 112.1900729735818
```

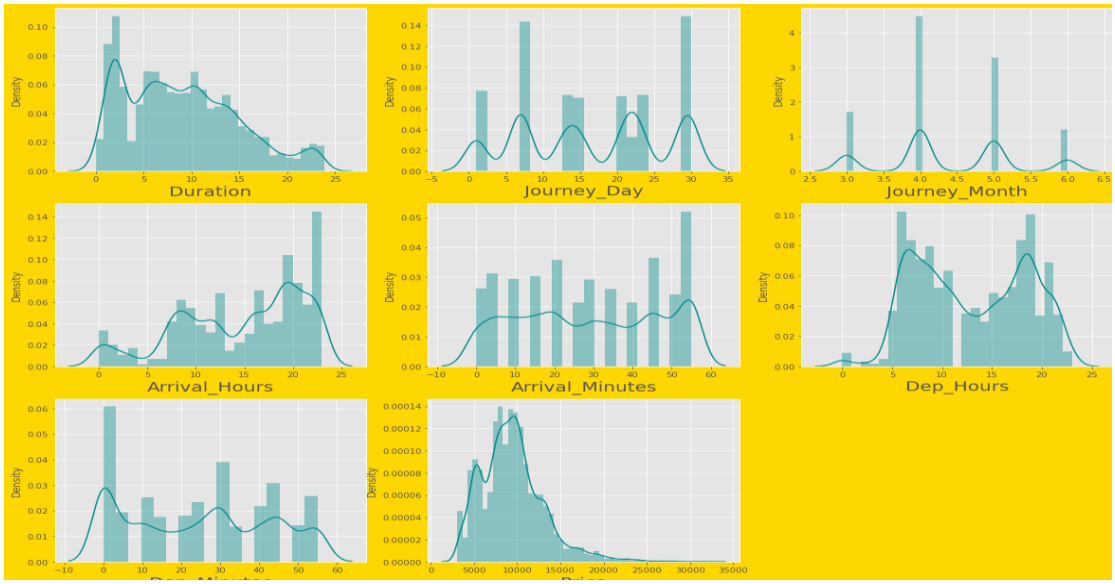
❖ Key Metrics for success in solving problem under consideration

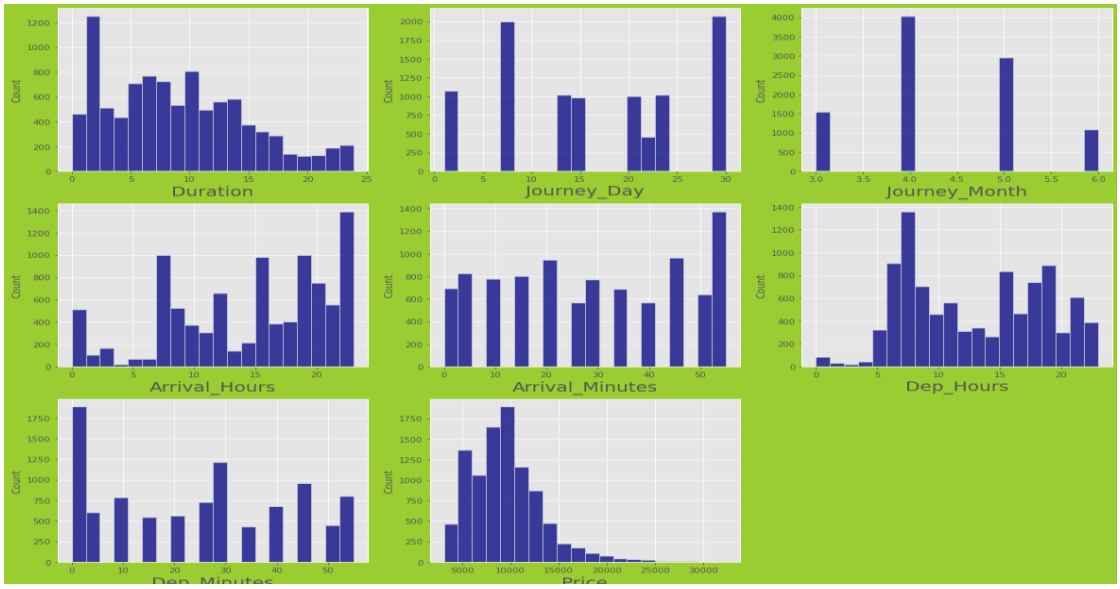
Following are the metrics to observe good model:

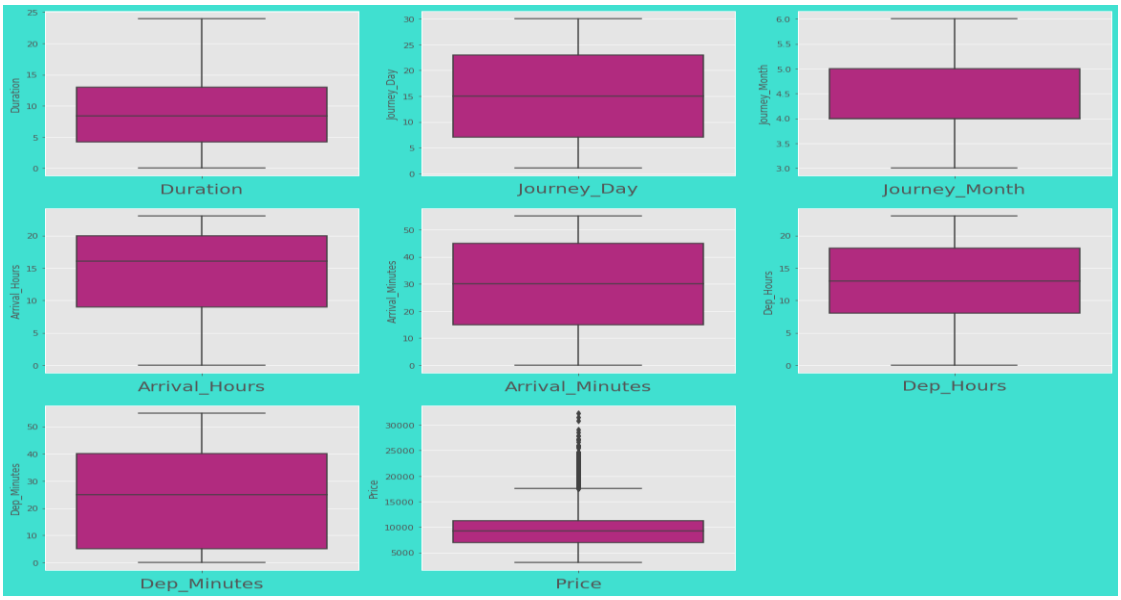
- Mean absolute error: MAE is a very simple metric which calculates the absolute difference between actual and predicted values. It should be minimum.
- Mean squared error : MSE is a very simple metric with a little bit of change in mean absolute error. Mean squared error states that finding the squared difference between actual and predicted value.
- Root mean absolute error: It is a simple square root of mean squared error.
- R2 score: R2 score is a metric that tells the performance of your model, not the loss in an absolute sense that how many wells did your model perform.
- Cross_val_score: Cross-validation is a resampling method that uses different portions of the data to test and train a model on different iterations. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice.

❖ Visualizations

- Univariate

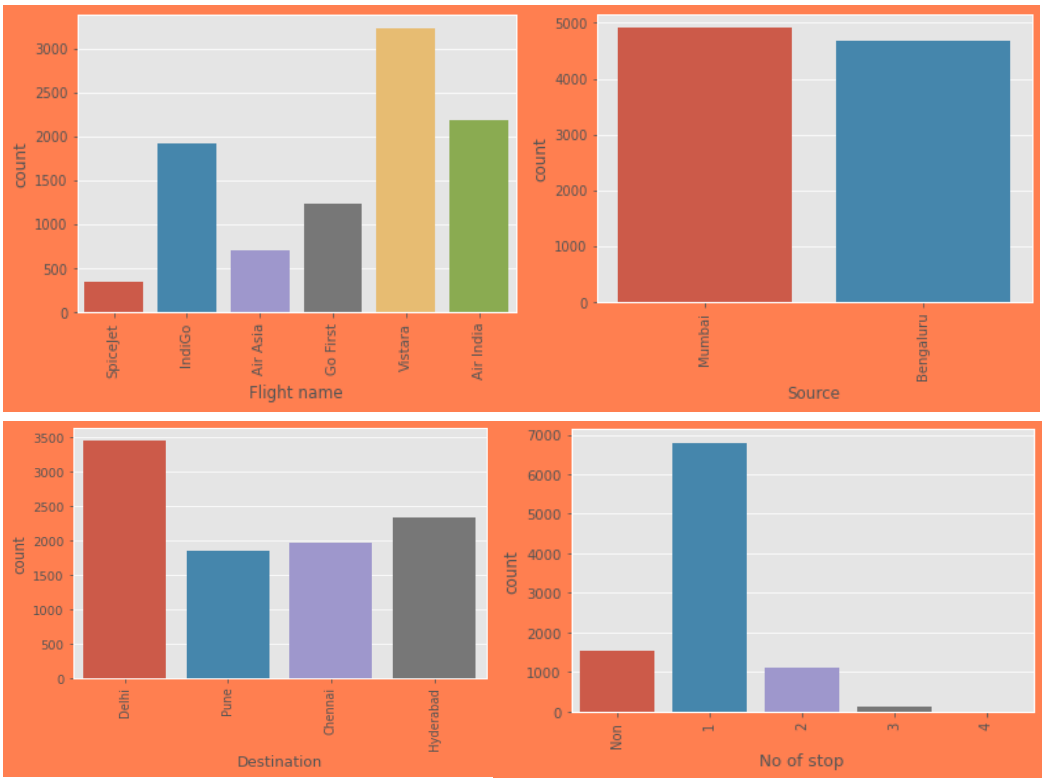






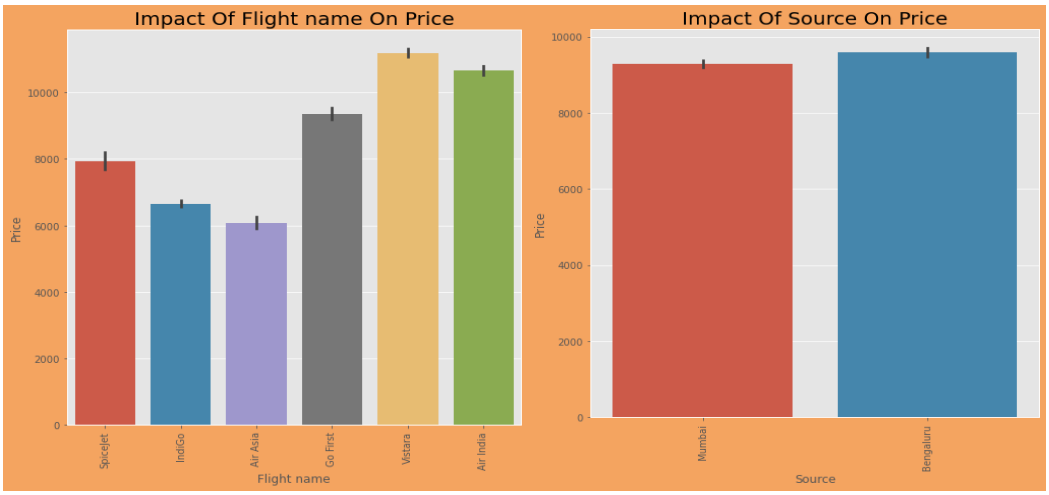
- Outcome:
  - ✓ Skewness and Outliers are present in target variable.

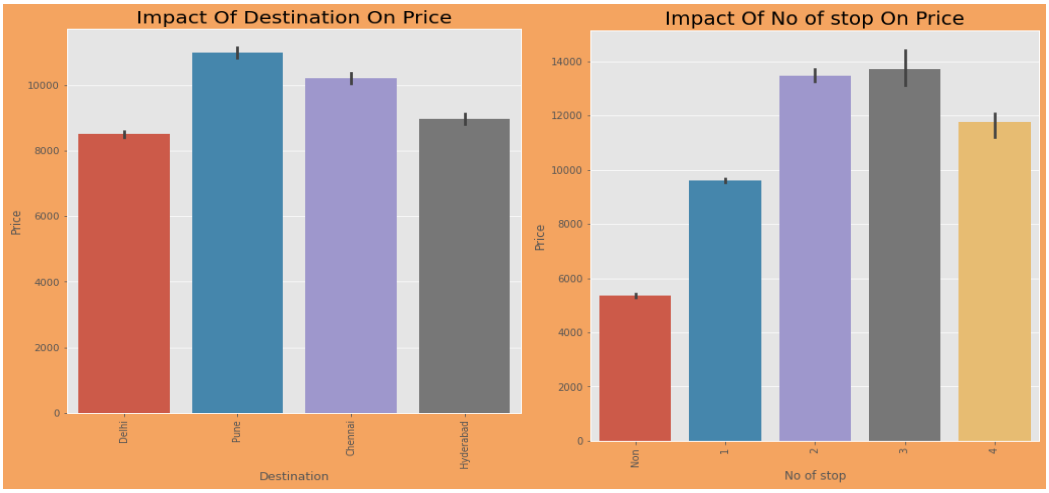
- Visualization of object column



- Outcome :
  - ✓ Most of the flight in the dataset is of 'Vistara' followed by 'Air India'.
  - ✓ The flight with Source location as 'Mumbai' are most in counting in compare to 'Bengaluru'.
  - ✓ The flight with Destination location as 'Delhi' are most in counting.
  - ✓ The flight with 1 stop are most in counting.

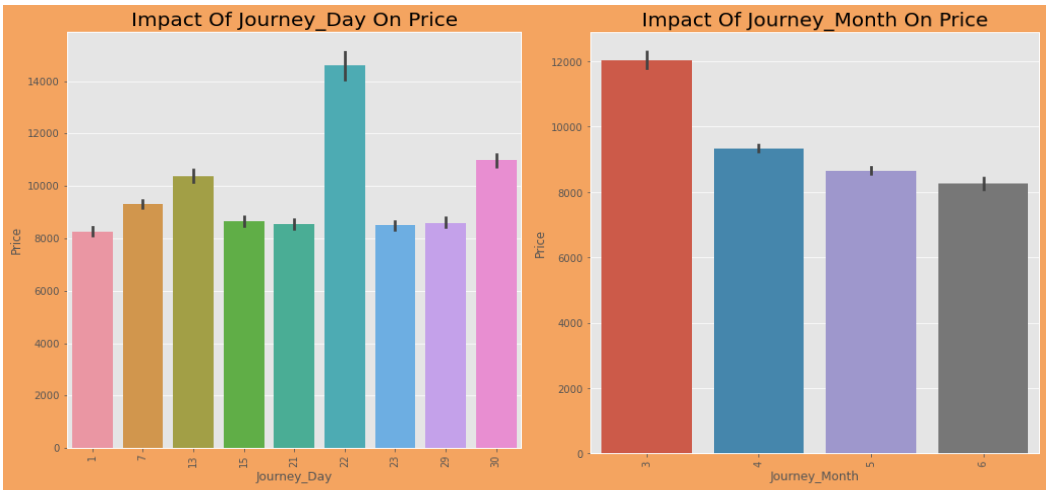
- Bivariate

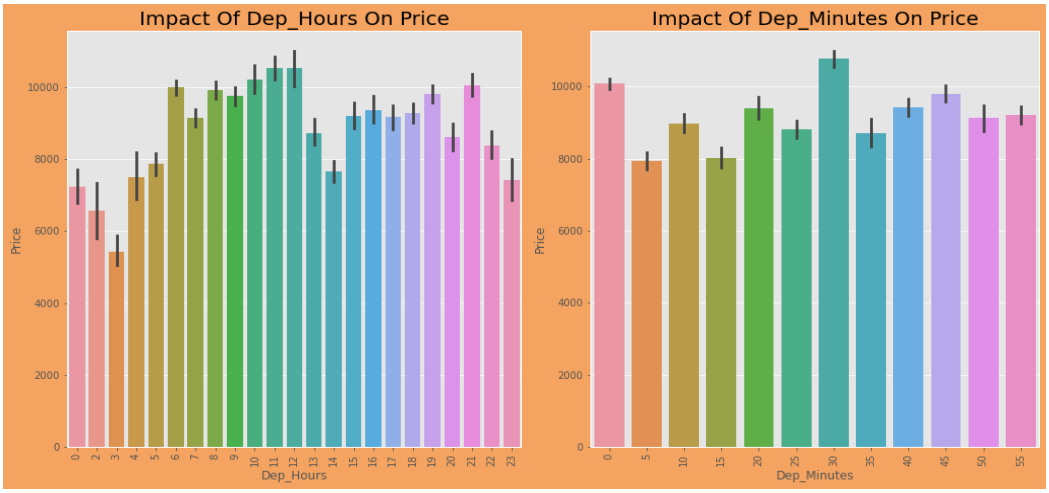


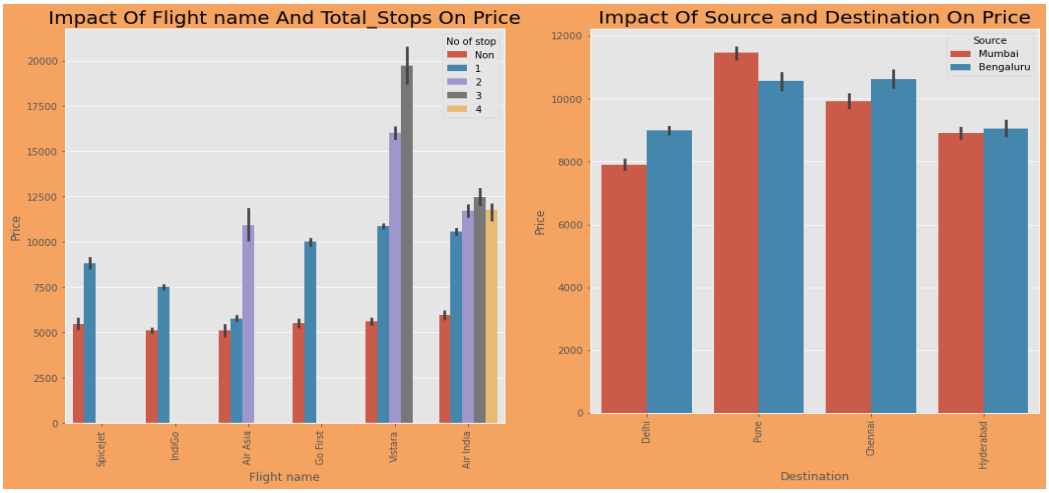


Outcome :

- ✓ 'Vistara' flight price are high, followed by 'Air India'.
- ✓ Those flight which has Source as 'Bengaluru' having high price.
- ✓ Those flight which has Destination as 'Pune' having high price.
- ✓ The fligh with 2 and 3 stop having high price.

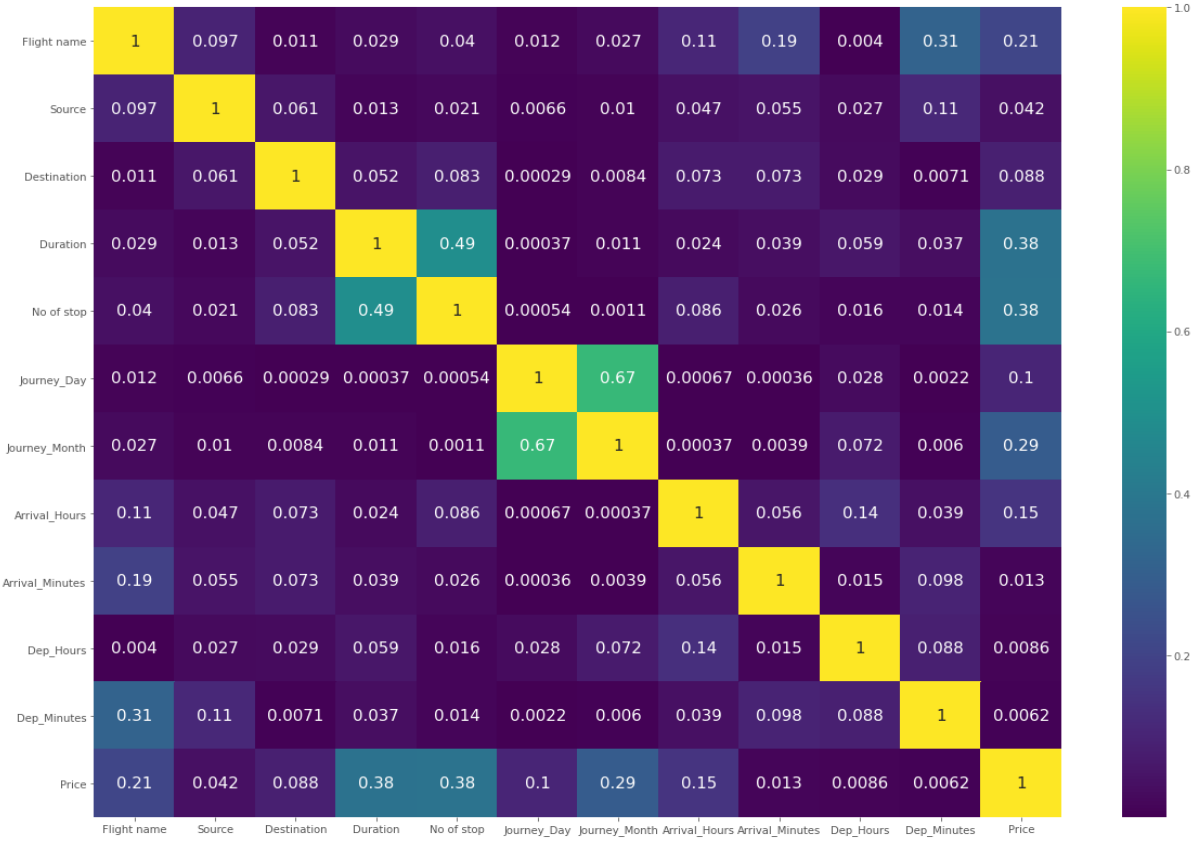






- Outcome :
  - ✓ The price are high on date 22.
  - ✓ The price are high if booking is done on within one week of journey date.
  - ✓ In the Morning and Evening time flight tickets are expensive.
  - ✓ 'Vistara' airline price is high when flight has 3 stop.
  - ✓ Mumbai to Pune flight ticket are most expensive ticket.

- Correlation matrix



- Outcome:
  - ✓ Multicollinearity is not present in the dataset.

❖ Interpretation of the Results

- Score of all model

	Name Of Model	Score On Test	R2 Score	RMSE	MSE	CV Score
0	Linear Regression	35.620180	0.356202	2917.018356	8.508996e+06	2963.747700
1	K-Neighbors Regressor	73.509354	0.735094	1871.158433	3.501234e+06	2078.484649
2	Decision Tree Regressor	79.026589	0.790266	1664.940904	2.772028e+06	1825.745272
3	Random Forest Regressor	88.833797	0.888338	1214.834280	1.475822e+06	1419.478729
4	Support Vector Machine Regressor	3.151190	0.031512	3577.761842	1.280038e+07	3577.523035
5	Bagging Regressor	86.843297	0.868433	1318.675141	1.738904e+06	1516.468321
6	Ada Boost Regressor	23.608940	0.236089	3177.499957	1.009651e+07	3125.906780
7	Gradient Boosting Regressor	77.624459	0.776245	1719.693508	2.957346e+06	1860.797739
8	XGB Regressor	88.330157	0.883302	1241.929065	1.542388e+06	1397.116756

- Outcome:
  - ✓ R2 score of Random Forest Regressor and XGB Regressor are also best. The RMSE score of Random Forest Regressor and XGB Regressor is less. Cross validation score is less for XGB Regressor as compared with Random Forest Regressor. So, XGB Regressor is final model on this dataset.

❖ Hyperparameter Tuning

```
clf = xb.XGBRegressor()
param = {'n_estimators':[100,150,200],
         'learning_rate' : [0.1,0.01,0.8],
         'max_depth' : [3,5,7,10],
         'subsample' : [0.1,0.5,0.9,1]
         }

grd = GridSearchCV(clf,param_grid=param,scoring='neg_mean_squared_error',cv=10)
grd.fit(X_train,y_train)


clf = grd.best_estimator_

clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)

print("Best parameter",grd.best_params_)
print("r2 score after hyper parameter tunning",r2_score(y_test,y_pred))
```

```
Best parameter {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 200, 'subsample': 0.9}
r2 score after hyper parameter tunning 0.8975886479120276
```

- Outcome:
  - ✓ After hyperparameter tuning i am able to improve the performance of my model. So fitted the model using best parameter to improve the performance of model.

CONCLUSION

❖ Key Findings and Conclusions of the Study

- In this project report, I used machine learning algorithms to predict the flight ticket prices. I describe the flow of process to understand the dataset and finding the correlation between the features. After that, I observed that features are very less correlated with target and there is no multicollinearity problem exist in the dataset. All feature are used as input for model for nine regression algorithms and by these algorithm I am able to predict the price for flight ticket and save the model. I also observed that the performance of model and compared different model performance. Also perform hyper -tuning on finalize model. To analysis model, used different metrics and perform cross validation to ensure the performance of model.

❖ Findings are :

- Do airfares change frequently?
  - ✓ Airfares change during the morning and evening time of the day.

- Do they move in small increments or in large jumps?
  - ✓ Airfares move in large jumps in the morning and evening time of the day.

- Do they tend to go up or down over time?
  - ✓ Airfares are smallest when flight depart at 3 AM, after that fares go up.

- What is the best time to buy so that the consumer can save the most by taking the least risk?
  - ✓ From the study I am on conclusion that early morning and late night time are the best time to buy cheapest ticket.

- Does price increase as we get near to departure date?
  - ✓ From the study I conclude that price increase as customers booked the ticket near to departure date i.e. last minute ticket are most expensive one.

- Is Indigo cheaper than Vistara and Air India Airways?
  - ✓ Indigo is cheaper than Vistara and Air India Airways. AirAsia is cheapest flight.

- Are morning flights expensive?
  - ✓ Not all morning flights are expensive because 3 AM flights are smallest fare flights.

❖ Learning Outcomes of the Study in respect of Data Science

- During the study on this dataset and building the model, I observed and obtained factors about the different criteria of ticket booking of flight. I also got the knowledge about the platform by which I can book the ticket of flight, how the ticket booking process worked, how price is dynamically changed, what are the best timings and departure date to book the ticket. Based on this observation, I tried to build a best model. I found that dataset was interesting but very small in size and did not cover maximum source, destination and time slot. All available data is for future reference, no past data is available. Different types of visualization graph are used like distplot, histplot, boxplot, countplot, barplot to visualize features, target and features impact on target. These visualizations told about features and target. Data cleaning was one of the important and crucial thing in this project where I deal with features having string values, features extraction and selection. After that, I used nine machine learning regression algorithms to build the model. I used different metrices to compare the model. Finally, I can conclude that XGB Regressor is the best model on this dataset. The big challenge I faced in this project during web scraping is to obtain the data. Initially, I tried web scraping on yatra.com, but after some scraping my request was denied by yatra.com server. Then, I tried to do web scraping on paytm.com and I got succeed in scraping the data.
- Finally, I achieved my goal i.e. to predict the price of flight ticket by machine learning algorithm, which in turn help both airline industry and customer.

❖ Limitations of this work and Scope for Future Work

- The limitation of this project is the small size of data.
- In future, this machine learning model may bind with various websites which in turn provide real time data for price prediction. Also, we may add large historical data of flight price which can help to improve accuracy of the machine learning model. We can build an android app as user interface for interacting with user. For better performance, we plan to judiciously design deep learning network structures, use adaptive learning rates and train on clusters of data rather than the whole dataset.

Written by: Rohit Kachhal