



Flight Price Prediction

Submitted by:

M SAI PUJITH

REDDY

ACKNOWLEDGMENT

A unique opportunity like this comes very rarely. It is indeed a pleasure for me to have worked on this project. The satisfaction that accompanies the successful completion of this project is incomplete without the mention of the people whose guidance has made it possible for me to complete this project. I am grateful to my internship company **Flip Robo Technologies** with its ideals and inspiration for providing me with facilities that has made this project a success.

I am grateful to **Vaishali Singh** for providing the opportunity to work as a intern in **Flip Robo Technologies**

I like to acknowledge the effort put in by our SME **Sapna Verma** in helping me understand the relevant concepts related to Data preprocessing and providing guidance when necessary.

M SAI PUJITH REDDY

INTRODUCTION

- **Business Problem Framing**
- Anyone who has booked a flight ticket knows how unexpectedly the prices vary. The cheapest available ticket on a given flight gets more and less expensive over time. This usually happens as an attempt to maximize revenue based on -
 1. Time of purchase patterns (making sure last-minute purchases are expensive)
 2. 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, you have to work on a project where you collect data of flight fares with other features and work to make a model to predict fares of flights.
- **Conceptual Background of the Domain Problem**

Airline companies use complex algorithms to calculate flight prices given various conditions present at that particular time. These methods take financial, marketing, and various social factors into account to predict flight prices. Nowadays, the number of people using flights has increased significantly. It is difficult for airlines to maintain prices since prices change dynamically due to different conditions. That's why we will try to use machine learning to solve this problem. This can help airlines by predicting what prices they can maintain. It can also help customers to predict future flight prices and plan their journey accordingly.
- **Review of Literature**

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 paper, we present a review of customer side and airlines side prediction models. Our review 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. Features extracted from external factors such as social media data and search engine query are not considered. Therefore, we introduce and discuss the concept of using social media data for ticket/demand prediction.

- **Motivation for the Problem Undertaken**

Motivated by previous studies, we can think of various additional useful features from social media that can possibly forecast airlines passenger demand and or ticket prices. For example, sentiment analysis of different twitter hash tags could convey the presence of some event at a flight origin/destination city that improves the prediction of ticket price/demand. This kind of feature extraction might involve searching for special keywords or group of terms, determining the number of times they appear, understanding the location and the date, their context etc.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem
- Mathematical Summary:

Dimensions of Dataset: There are 9 columns and 1611 rows in this dataset

Null Values: There are no null values in this dataset

Skewness: There is no skewness in columns.

- Data Sources and their formats

Standard deviation is very normal in most of the columns.

There is not much difference between mean and 50th percentile, which means data is skewed

There is not much difference between 75th percentile and max, which means there are outliers

- Data Sources and their formats

All the objects were object data type after scraping before pre-processing.

- Data Pre-processing Done

- Feature Extraction:

At first we had only 9 columns in which Date column consists of Day of the week and Days prior booking. After extracting the required fields using feature extraction we got 11 columns.

Date Column (before):

Date
13-10-2021
13-10-2021
13-10-2021
13-10-2021
13-10-2021

After extracting week day and days prior booking:

Note: I have scraped data of 13th October.

Day	Days_priors_booking
Wednesday	4
Wednesday	4
Wednesday	4
Wednesday	4
Wednesday	4

1. Extracting Day

```
#from Dates column we can extract day and days prior to booking

def day(x):
    if x == 0:
        return "Monday"
    elif x == 1:
        return "Tuesday"
    elif x == 2:
        return "Wednesday"
    elif x == 3:
        return "Thursday"
    elif x == 4:
        return "Friday"
    elif x == 5:
        return "Saturday"
    elif x == 6:
        return "Sunday"
```

```
#extracting the day of the week
df["Day"] = df["week_num"].apply(day)
```

2. Extracting the days prior booking

```
#extraxting the days prior to booking column
#since i scaped the data on 9th
#for 9th days prior booking is zero
df["Days_priors_booking"] = df["only_date"] - 9
```

3. Converting Duration columns in minutes

```
#Converting duration columns to minutes

def minutes(x):
    hour, minute = x.split("h")
    minutes = (60 * int(hour)) + int(minute.strip("m"))
    return float(minutes)
```

```
#applying the above function
df['Duration_in_Minutes'] = df.Duration.apply(minutes)
```

4. Feature extraction of next day arrival column from arrival date

```
def next_day(x):
    if "\n" in x:
        return "Yes"
    else:
        return "No"
```

```
#creating a new column using the existing information
#if flight reaches the destination next day i.e after 12 am
df["Next_day_Arrival"] = df["Arrival Time"].apply(next_day)
```

5. Extracting date from Arrival date column using Regular expressions

```
import re
def Arrival(x):
    a = re.search(r"\d\d:\d\d", x)
    return a.group()
```

```
#removing the "\n+1 day" from arrival date
df["Arrival_Time"] = df["Arrival Time"].apply(Arrival)
```

6. Feature engineering of Arrival part of the day and departure part of the day

```
#lets extract the time of the day from this column Dep_hour and Arrival_hour column
#lets write a function for this
def part_of_the_day(x):
    if (x > 4) and (x <= 8):
        return 'Early Morning'
    elif (x > 8) and (x <= 12):
        return 'Morning'
    elif (x > 12) and (x <= 16):
        return 'Noon'
    elif (x > 16) and (x <= 20):
        return 'Eve'
    elif (x > 20) and (x <= 24):
        return 'Night'
    elif (x <= 4):
        return 'Late Night'
```

```
#applying the above fuction to get part of the day column
df['Departure_Part_of_the_day'] = df.Depature_hour.apply(part_of_the_day)
```

```
#applying the above fuction to get part of the day column
df['Arrival_Part_of_the_day'] = df.Arrival_hour.apply(part_of_the_day)
```

7. Converting week number to weekday

```
#exatrcting week number
df["week_num"] = pd.DatetimeIndex(df["Dates"]).weekday
```

```
#from Dates column we can extarct day and days prior to booking
def day(x):
    if x == 0:
        return "Monday"
    elif x == 1:
        return "Tuesday"
    elif x == 2:
        return "Wednesday"
    elif x == 3:
        return "Thursday"
    elif x == 4:
        return "Friday"
    elif x == 5:
        return "Saturday"
    elif x == 6:
        return "Sunday"
```


8. Final Data Frame after Data Pre-processing

df												
	Airline	Source City	Destination City	Stops	Price	Duration_in_Minutes	Next_day_Arrival	Depature_hour	Arrival_hour	Departure_Part_of_the_day	Arrival_Pai	
0	Air Asia	Mumbai	Kolkata	1 Stop	7410.0	475.0	Yes	23	7	Night		
1	Air Asia	Mumbai	Kolkata	1 Stop	7410.0	685.0	Yes	20	8	Eve		
2	Air Asia	Mumbai	Kolkata	2 Stop (s)	7410.0	925.0	No	5	21	Early Morning		
3	SpiceJet	Mumbai	Kolkata	Non Stop	7412.0	155.0	No	5	7	Early Morning		
4	Go First	Mumbai	Kolkata	Non Stop	7412.0	160.0	No	7	10	Early Morning		
...
1606	Vistara	Lucknow	Goa	2 Stop (s)	14690.0	1210.0	Yes	15	11	Noon		
1607	Vistara	Lucknow	Goa	2 Stop (s)	14690.0	1210.0	Yes	15	11	Noon		
1608	Vistara	Lucknow	Goa	2 Stop (s)	14690.0	1210.0	Yes	15	11	Noon		
1609	Vistara	Lucknow	Goa	2 Stop (s)	14690.0	1210.0	Yes	15	11	Noon		
1610	IndiGo	Lucknow	Goa	1 Stop	14864.0	310.0	No	9	14	Morning		

1611 rows × 13 columns

9. Scaling the Input data

```
#Lets scale the data using standard scaler
scaler = StandardScaler()
scaled_X = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)
scaled_X.head()
```

	Duration_in_Minutes	Depature_hour	Arrival_hour	Days_priors_booking	Airline_Air Asia	Airline_Air India	Airline_Go First	Airline_IndiGo	Airline_SpiceJet	Airline_Vistara	...
0	-0.463273	2.125164	-1.370853	-1.074835	3.280110	-0.544112	-0.307294	-0.647598	-0.288772	-0.543154	...
1	0.051397	1.548211	-1.182788	-1.074835	3.280110	-0.544112	-0.307294	-0.647598	-0.288772	-0.543154	...
2	0.639591	-1.336554	1.262053	-1.074835	3.280110	-0.544112	-0.307294	-0.647598	-0.288772	-0.543154	...
3	-1.247531	-1.336554	-1.370853	-1.074835	-0.304868	-0.544112	-0.307294	-0.647598	3.462937	-0.543154	...
4	-1.235277	-0.951919	-0.806659	-1.074835	-0.304868	-0.544112	3.254217	-0.647598	-0.288772	-0.543154	...

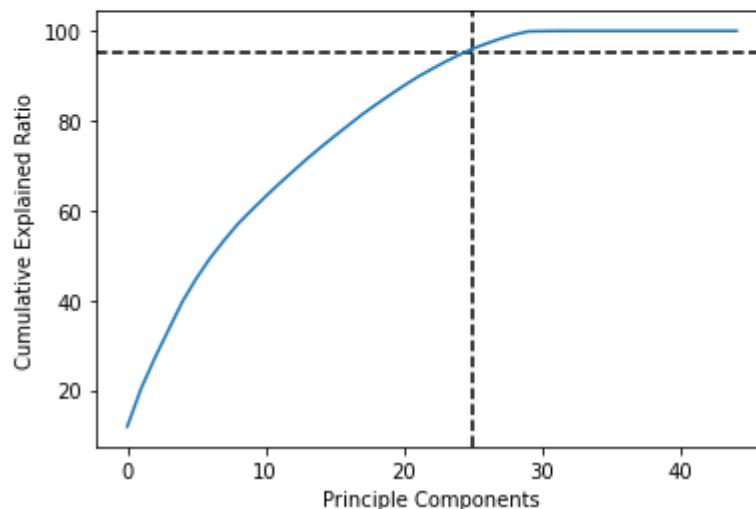
5 rows × 45 columns

10. PCA (principle component Analysis)

```
#lets plot the graph for graphical understanding
plt.ylabel('Cumulative Explained Ratio')
plt.xlabel('Principle Components')

plt.axvline(x = s, color = 'k', linestyle = '--')
plt.axhline(y = 95, color = 'k', linestyle = '--')

plt.plot(cum_score)
plt.show()
```



- Data Inputs- Logic- Output Relationships
 - We see good correlation between day's prior booking and Price as well as weekday and price.
 - Airfares change very frequently
 - Airfares tend to go down over time
-
- Hardware and Software Requirements and Tools Used
 - Machine: Can use a laptop/desktop.
 - Operating system: Windows 8 or 10, Mac OS X 10.9 Mavericks or Higher

- RAM & Processor: 4 GB+ RAM, i3 5th Generation 2.2 Ghz or equivalent/higher
- Tools Used: Jupyter Notebook, Microsoft Excel.
- Libraries Used : Sklearn,Seaborn,Matplotlib,Pandas,Numpy ,warnings

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)
- I started by extracting all the random flights data from Yatra website from five different location which include Delhi, Mumbai, Kolkata and Bangalore.
- After extracting the required information, I had column called Date which consist of three different inputs which days prior booking and day of the week.
- I used Regular Expression and other methods to extract the information from the existing column.
- I then dropped the old columns after extracting the required information.
- Converted Object data type like price to INT after replacing Symbols.

- Testing of Identified Approaches (Algorithms)

Below are the algorithms used for the training and testing.

Lr = LinearRegression()

dtc = DecisionTreeRegressor()

knn = KNeighborsRegressor(n_neighbors=5)

rf = RandomForestRegressor()

ada = AdaBoostRegressor()

1. Linear Regression

```

lrg = LinearRegression()
lrg.fit(x_train,y_train)
pr =lrg.predict(x_test)
print("r2_score of linear refression is :", r2_score(y_test,pr))
print('Error :')
print('mean absolute error :',mean_absolute_error(y_test,pred))
print('mean squared error : ', mean_squared_error(y_test,pred))
print('root mean squared error :',np.sqrt(mean_squared_error(y_test,pred)))

r2_score of linear refression is : 0.6951706950402602
Error :
mean absolute error : 3597.4374685695016
mean squared error : 19357960.20887996
root mean squared error : 4399.768199448689

```

Main Code for Running Multiple model at a time

```

#lets Choose r2 score of below four Models
dtc = DecisionTreeRegressor()
knn = KNeighborsRegressor(n_neighbors=5)
rf = RandomForestRegressor()
ada = AdaBoostRegressor()

```

```

#checking each model with Cross val score
model_list = [dtc,knn,rf,ada]
least_difference = []
for m in model_list:
    m.fit(x_train,y_train)
    pred = m.predict(x_test)
    cvs = cross_val_score(m,pca_x,Y,cv =5)
    print('\n')
    print(m)
    print('Scores :')
    print('r2 score:',r2_score(y_test,pred))
    print('Cross Val score :',cvs.mean())
    print('Error :')
    print('mean absolute error :',mean_absolute_error(y_test,pred))
    print('mean squared error : ', mean_squared_error(y_test,pred))
    print('root mean squared error :',np.sqrt(mean_squared_error(y_test,pred)))
    print('Difference :')
    difference = np.abs(r2_score(y_test,pred) - cvs.mean())
    print('Difference between cross val score and r2 score is : {0:.2f}'.format(difference))
    least_difference.append((m,'Difference between cross val score and r2 score error is : {0:.2f}'.format(difference)))

```

2. Decision Tree Regressor

```

DecisionTreeRegressor()
Scores :
r2 score: 0.8127532004570739
Cross Val score : -2.617234370957825
Error :
mean absolute error : 643.9859154929577
mean squared error : 2300128.842253521
root mean squared error : 1516.6175662484993
Difference :
Difference between cross val score and r2 score is : 3.43

```

3. KNN Regressor

```
KNeighborsRegressor()  
Scores :  
r2 score: 0.8405881495596268  
Cross Val score : -0.1285775537056941  
Error :  
mean absolute error : 880.4056338028169  
mean squared error : 1958205.9393802818  
root mean squared error : 1399.359117374908  
Difference :  
Difference between cross val score and r2 score is : 0.97
```

4. Random Forest Regressor

```
RandomForestRegressor()  
Scores :  
r2 score: 0.9112200542997836  
Cross Val score : -0.9324741693884183  
Error :  
mean absolute error : 651.2223661971831  
mean squared error : 1090567.71179662  
root mean squared error : 1044.3025001390258  
Difference :  
Difference between cross val score and r2 score is : 1.84
```

5. AdaBoost Regressor

```
AdaBoostRegressor()  
Scores :  
r2 score: 0.7440735156979998  
Cross Val score : -0.652817374716703  
Error :  
mean absolute error : 1474.0280186149187  
mean squared error : 3143786.112641262  
root mean squared error : 1773.072506312492  
Difference :  
Difference between cross val score and r2 score is : 1.40
```

- Run and Evaluate selected models

These are the algorithms used along with the snapshot of their code.

- Key Metrics for success in solving problem under consideration
- **R2 Score:**

R-squared is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model.

Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y , disregarding the input features, would get a R^2 score of 0.0.

We obtained the r^2 score of 91.12 % which is very good.

- **Root Mean Squared Error (RMSE):**

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points. RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit.

The RMSE is the square root of the variance of the residuals. It indicates the absolute fit of the model to the data—how close the observed data points are to the model's predicted values. Whereas R-squared is a relative measure of fit, RMSE is an absolute measure of fit. Lower values of RMSE indicate better fit.

Root Mean Squared of this Random forest model is 1044 which is low compared to others.

- **Mean Squared Error:**

Mean squared error is the average of the squared error that is used as the loss function for least squares regression. It is the sum, over all the data points, of the square of the difference between the predicted and actual target variables, divided by the number of data points.

Mean Squared of this Random forest model is 1090567.711779662 which is low compared to others.

- **Mean Absolute Error:**

In the context of machine learning, absolute error refers to the magnitude of difference between the prediction of an observation and the true value of that observation. MAE takes the average of absolute errors for a group of predictions and observations as a measurement of the magnitude of errors for the entire group.

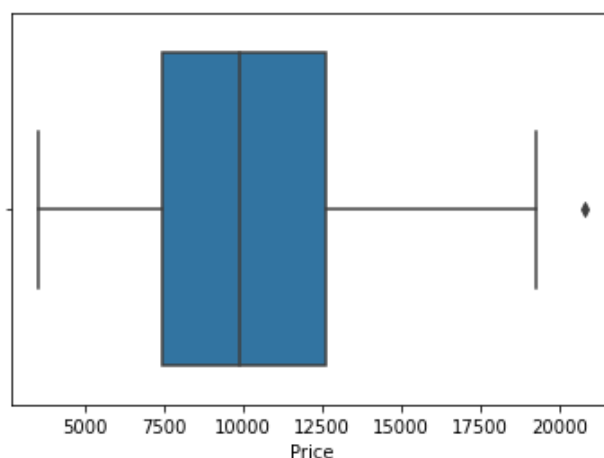
Mean Absolute error of this random forest model is 651.2223661971831.

- **Visualizations**

The plots made along with their pictures and the inferences and observations obtained from those.

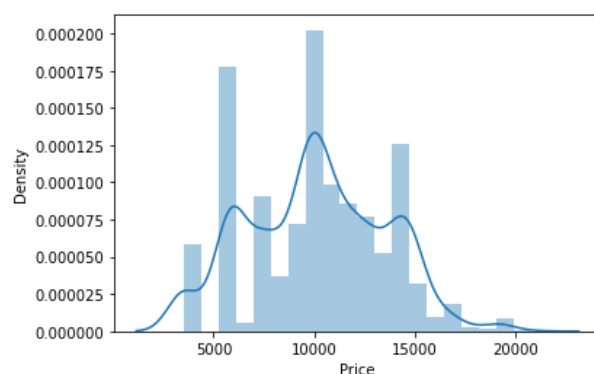
Outliers in Target column:

```
#lets check the box plot of our target column, to check if there are outliers  
sns.boxplot(df.Price)  
plt.show()
```



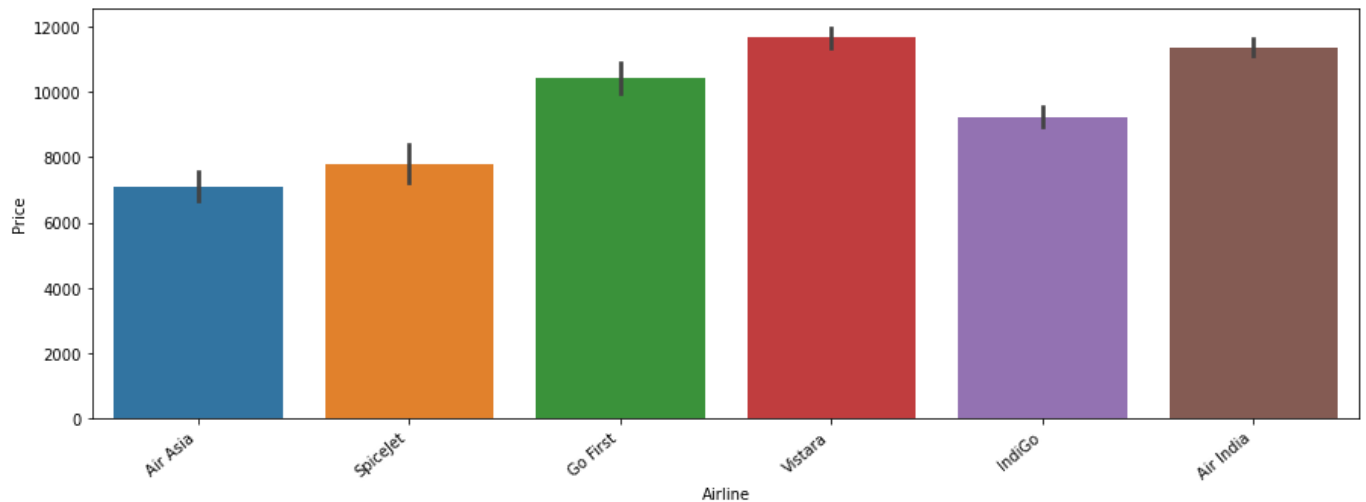
Distribution of the target column:

```
#lets check the distribution of the target column  
sns.distplot(df.Price)  
plt.show()
```



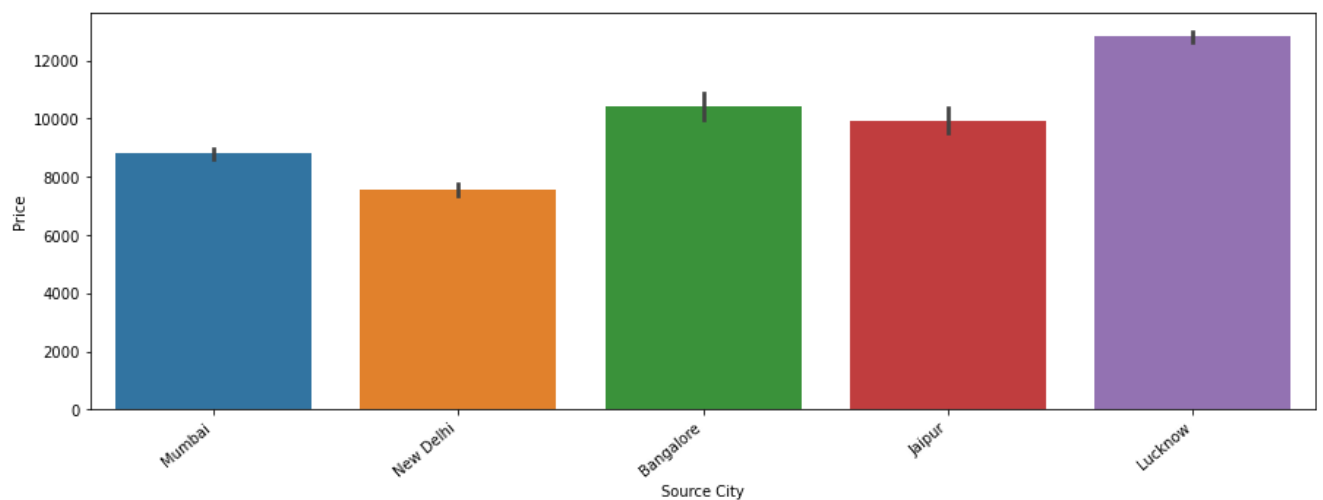
Bivariate Analysis with Target Variable:

```
#Lets check this column againt our target variable
plt.figure(figsize= (15,5))
coll = sns.barplot(x = df['Airline'] , y =df['Price'] )
coll.set_xticklabels(coll.get_xticklabels(), rotation=40, ha="right")
plt.show()
```



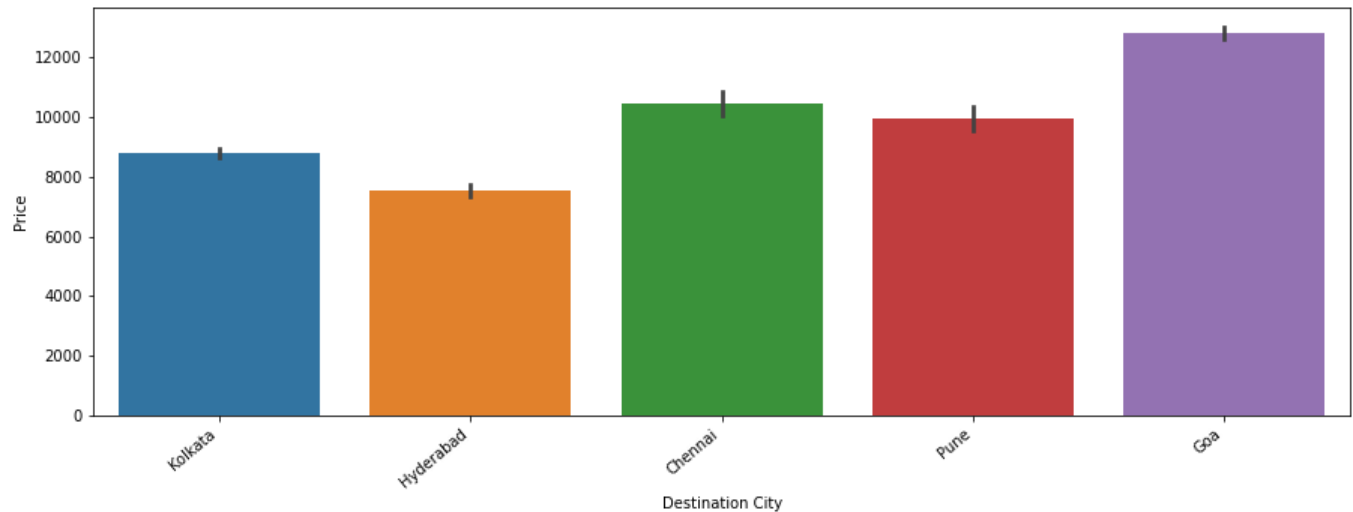
#from the above plot we can see that AirAsia is very cheap and vistara is more expensive

```
#Lets check this column againt our target variable
plt.figure(figsize= (15,5))
coll = sns.barplot(x = df['Source City'] , y =df['Price'] )
coll.set_xticklabels(coll.get_xticklabels(), rotation=40, ha="right")
plt.show()
```



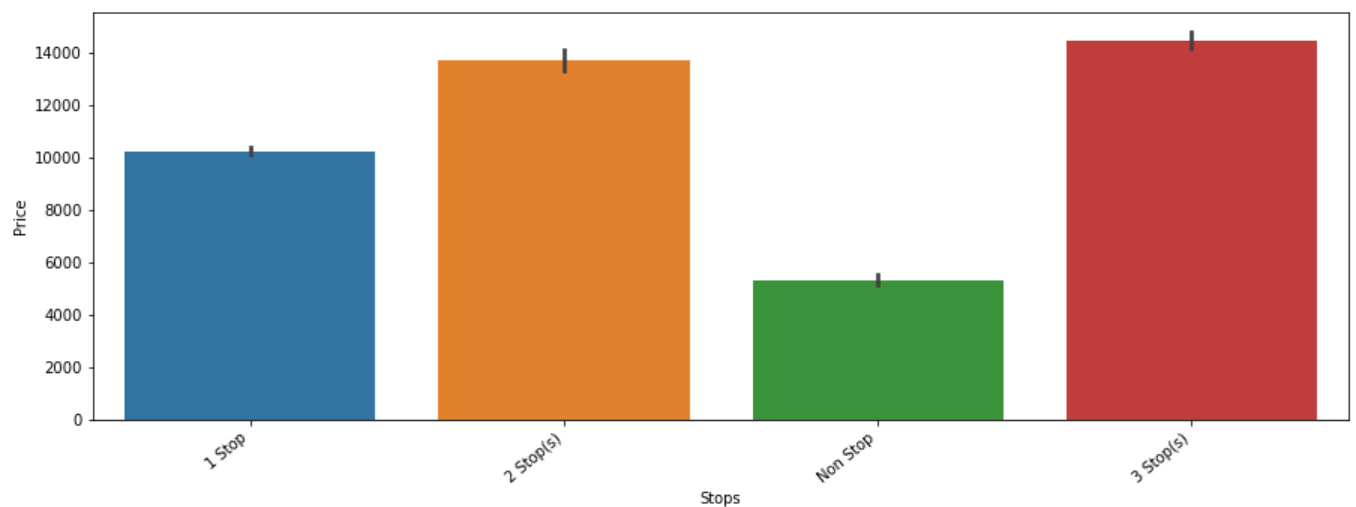
#we can see that New Delhi have lowest prices and Lucknow have highest prices.

```
#Lets check this column against our target variable
plt.figure(figsize= (15,5))
coll = sns.barplot(x = df['Destination City'] , y =df['Price'] )
coll.set_xticklabels(coll.get_xticklabels(), rotation=40, ha="right")
plt.show()
```



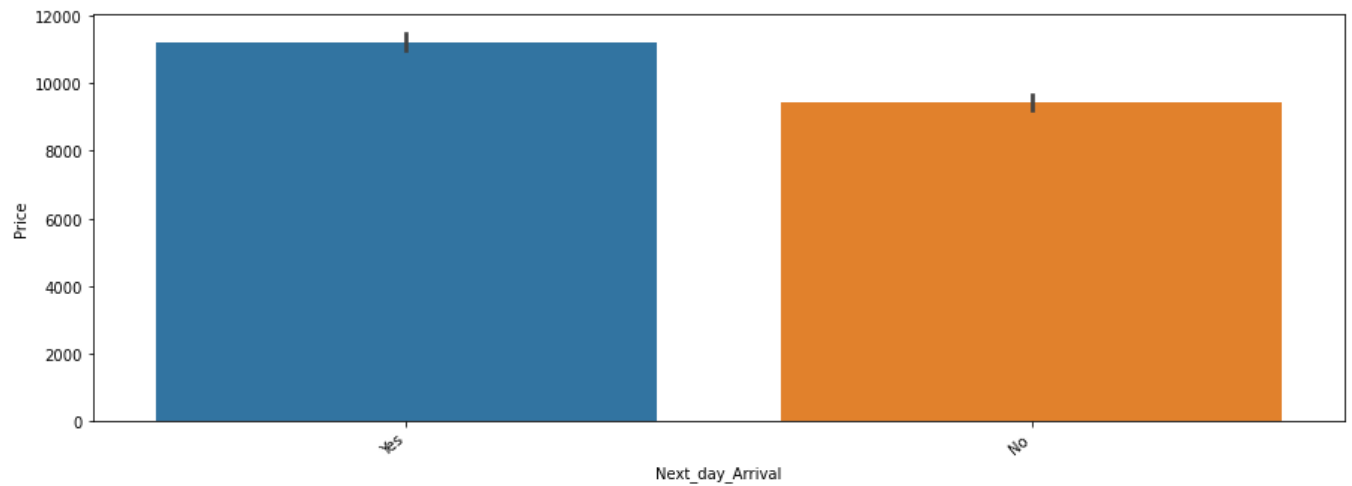
we can see that Hyderabad have less prices and Goa have high prices

```
#Lets check this column against our target variable
plt.figure(figsize= (15,5))
coll = sns.barplot(x = df['Stops'] , y =df['Price'] )
coll.set_xticklabels(coll.get_xticklabels(), rotation=40, ha="right")
plt.show()
```



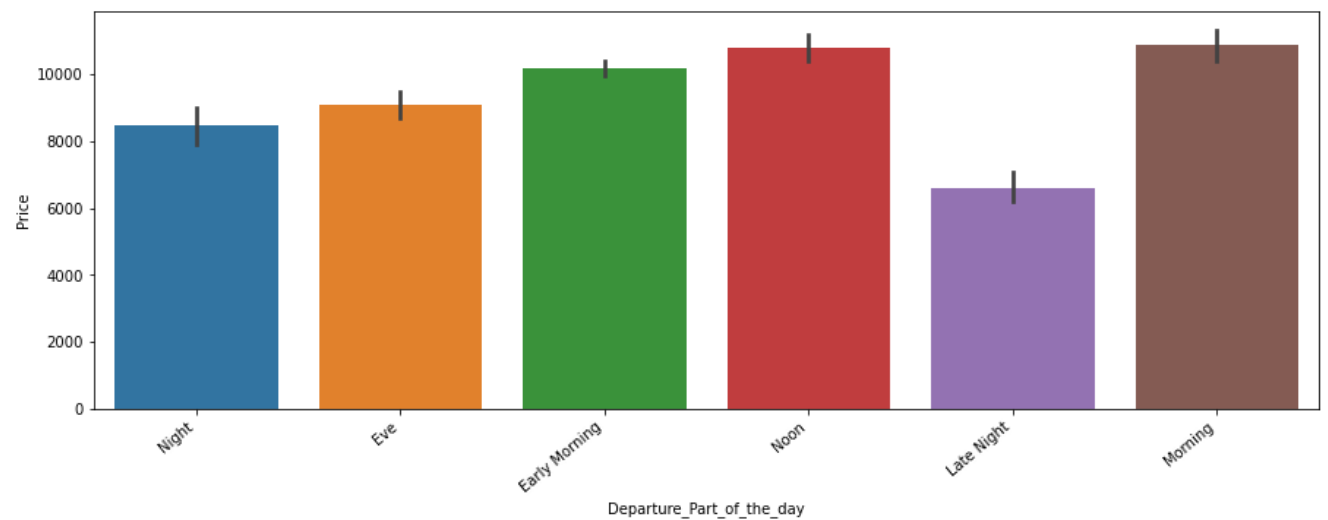
#from the above plot we can see that flights having more spots are more expensive

```
#Lets check this column againt our target variable
plt.figure(figsize= (15,5))
coll = sns.barplot(x = df['Next_day_Arrival'] , y =df['Price'] )
coll.set_xticklabels(coll.get_xticklabels(), rotation=40, ha="right")
plt.show()
```



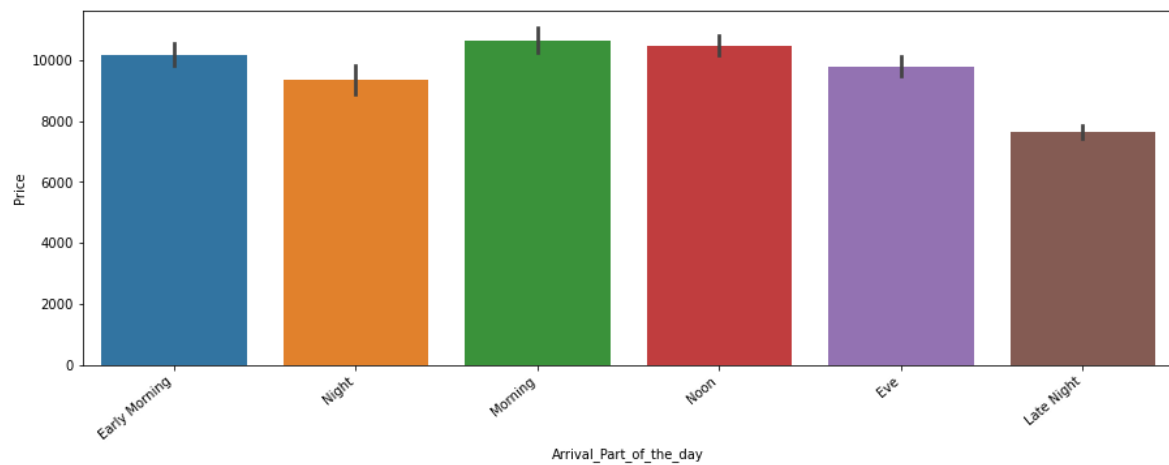
#from the above plot we can see that if arrival is not in same day then the flight price is expensive

```
#Lets check this column againt our target variable
plt.figure(figsize= (15,5))
coll = sns.barplot(x = df['Departure_Part_of_the_day'] , y =df['Price'] )
coll.set_xticklabels(coll.get_xticklabels(), rotation=40, ha="right")
plt.show()
```



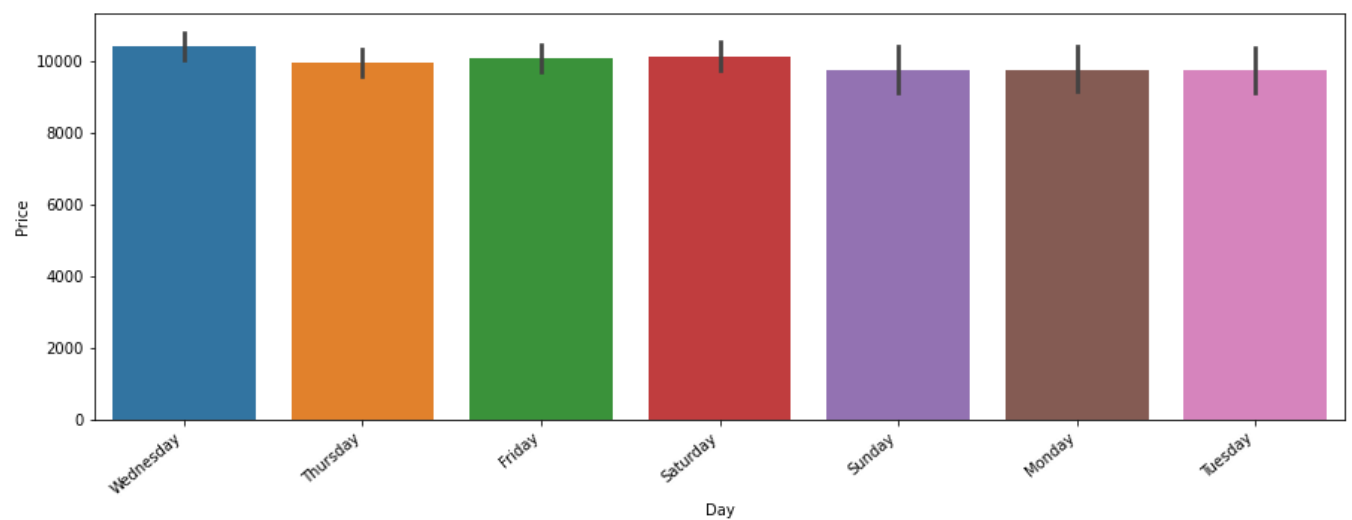
#from the above plot we can see that Morning and Noon flights are expensive and Late Night flights are cheaper

```
#Lets check this column againt our target variable
plt.figure(figsize= (15,5))
coll = sns.barplot(x = df['Arrival_Part_of_the_day'] , y =df['Price'] )
coll.set_xticklabels(coll.get_xticklabels(), rotation=40, ha="right")
plt.show()
```



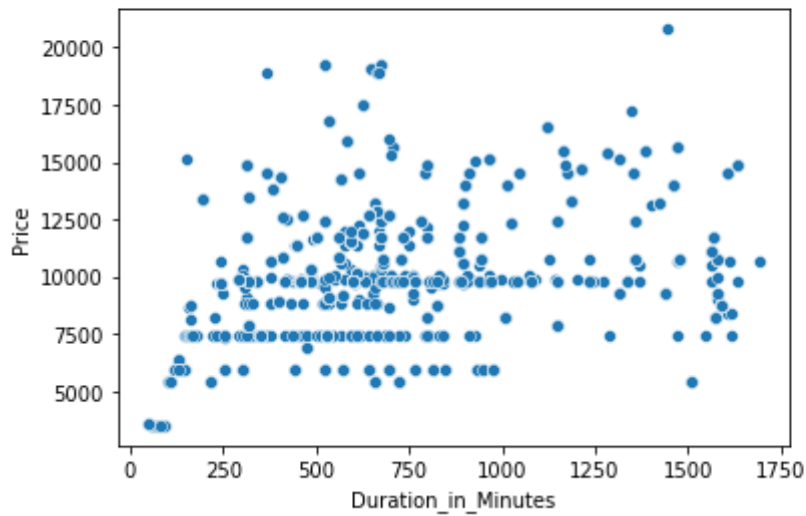
#Late flights are cheap and morning flight prices are high

```
#Lets check this column againt our target variable
plt.figure(figsize= (15,5))
coll = sns.barplot(x = df['Day'] , y =df['Price'] )
coll.set_xticklabels(coll.get_xticklabels(), rotation=40, ha="right")
plt.show()
```



#from the above plot ve can see that there is not much difference in prices

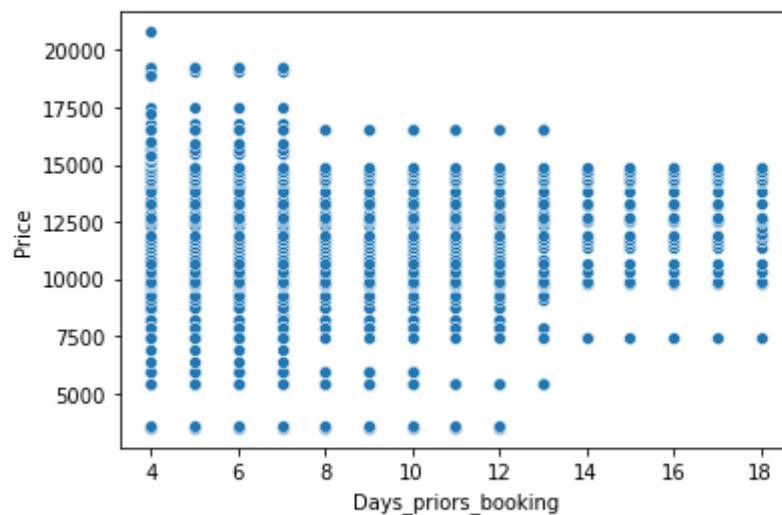
```
#Lets check this column againt our target variable
sns.scatterplot(x = 'Duration_in_Minutes', y = 'Price' , data = df)
plt.show()
```



1. Do airfares change frequently?

```
#Lets check this column againt our target variable
```

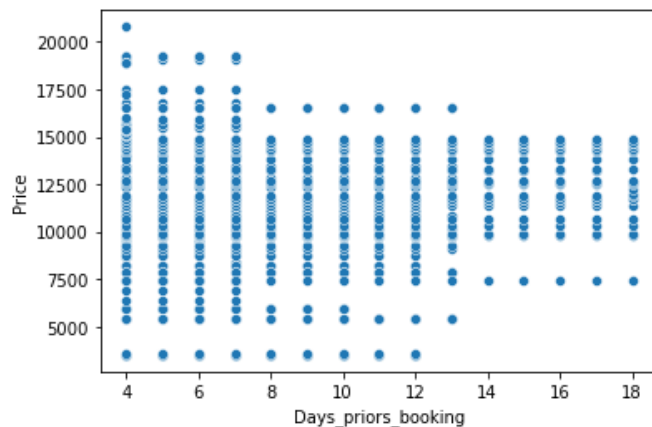
```
#Lets check this column againt our target variable
sns.scatterplot(x = 'Days_priors_booking', y = 'Price' , data = df)
plt.show()
```



From the above plot we can see that airfares change frequently

2. Do they move in small increments or in large jumps?

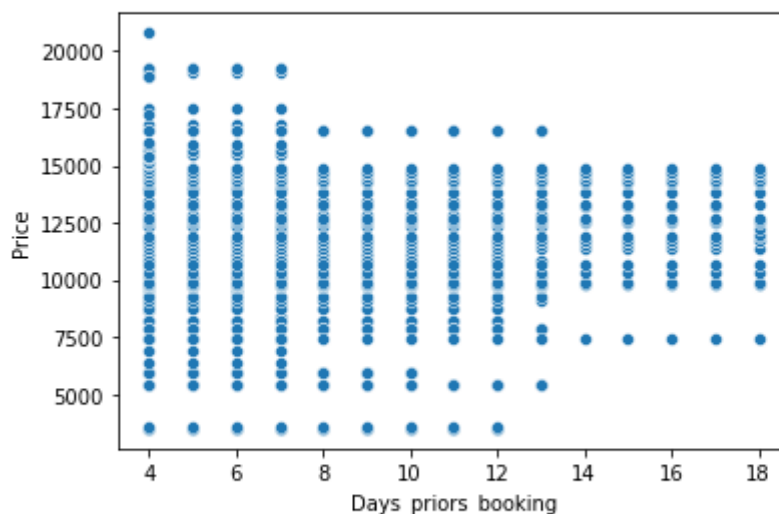
```
#Lets check this column against our target variable
sns.scatterplot(x = 'Days_priors_booking', y = 'Price' , data = df)
plt.show()
```



Yes, they move in large jumps

3. Do they tend to go up or down over time?

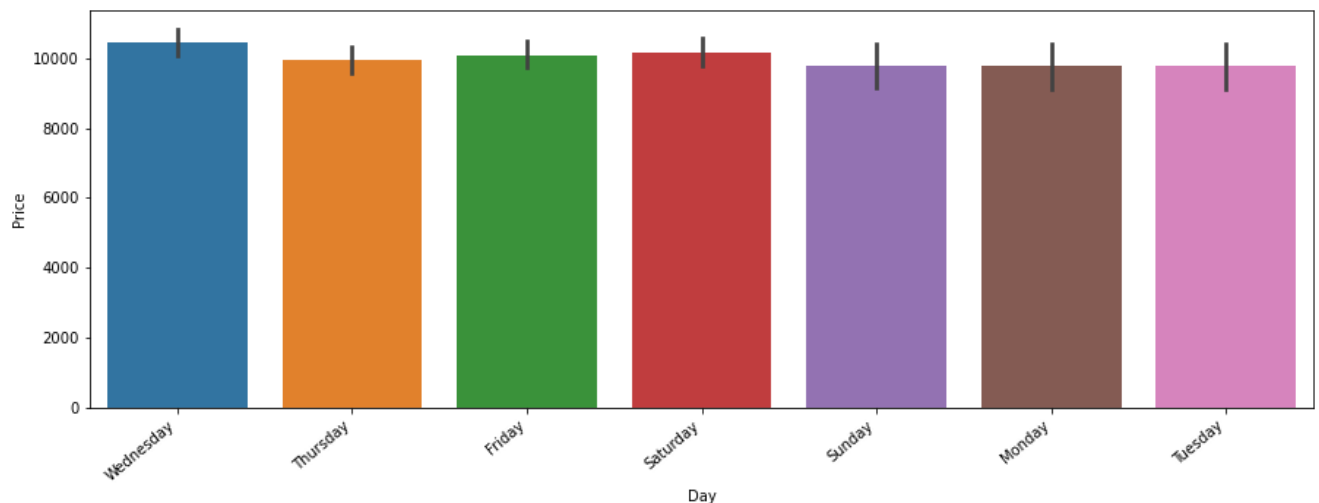
```
#Lets check this column against our target variable
sns.scatterplot(x = 'Days_priors_booking', y = 'Price' , data = df)
plt.show()
```



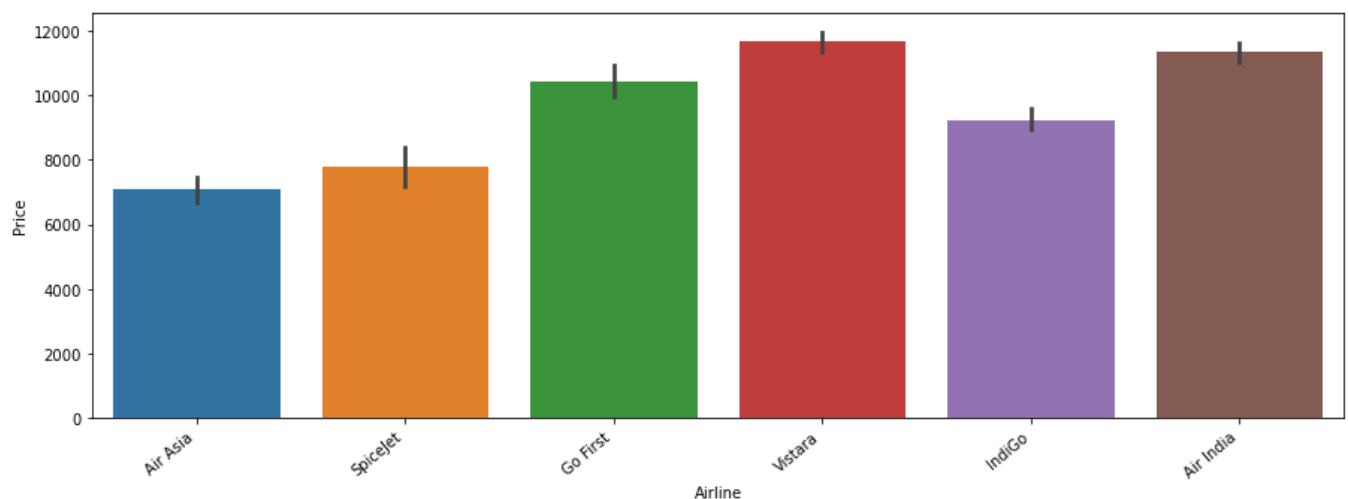
They usually tend to go down over time

4. What is the best time to buy so that the consumer can save the most by taking the least risk?

```
#Lets check this column againt our target variable
plt.figure(figsize= (15,5))
coll = sns.barplot(x = df['Day'] , y =df['Price'] )
coll.set_xticklabels(coll.get_xticklabels(), rotation=40, ha="right")
plt.show()
```



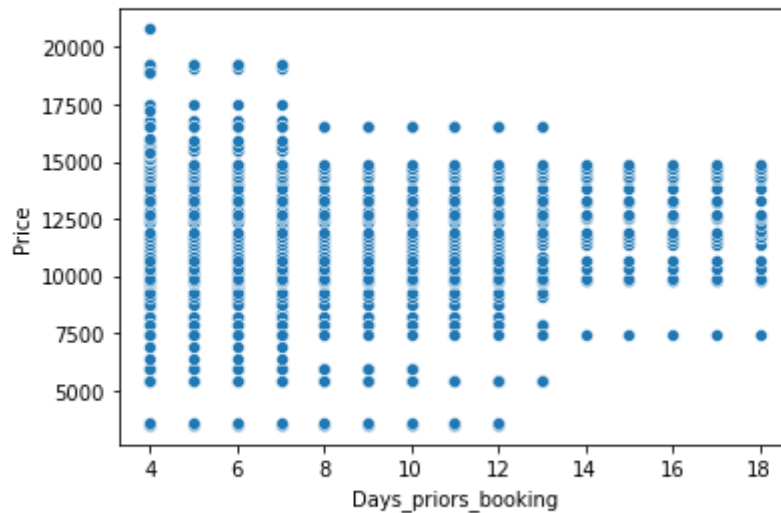
```
#Lets check this column againt our target variable
plt.figure(figsize= (15,5))
coll = sns.barplot(x = df['Airline'] , y =df['Price'] )
coll.set_xticklabels(coll.get_xticklabels(), rotation=40, ha="right")
plt.show()
```



Going on Tuesday and choosing Air Asia can reduce the money

5. Does price increase as we get near to departure date?

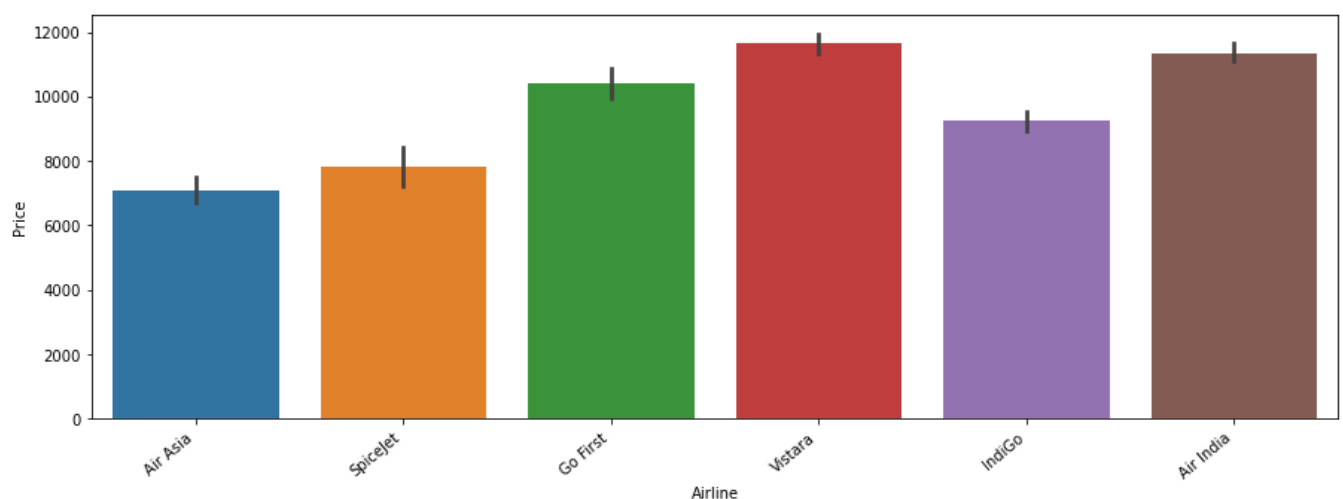
```
#Lets check this column against our target variable
sns.scatterplot(x = 'Days_priors_booking', y = 'Price' , data = df)
plt.show()
```



Yes, booking prior can save some money

6. Is Indigo cheaper than Jet Airways?

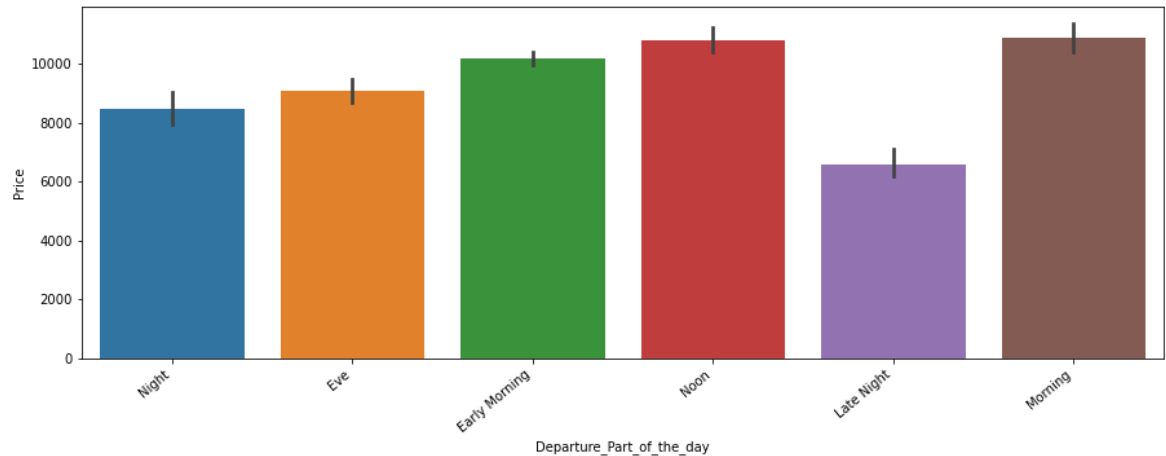
```
#Lets check this column against our target variable
plt.figure(figsize= (15,5))
coll = sns.barplot(x = df['Airline'] , y =df['Price'] )
coll.set_xticklabels(coll.get_xticklabels(), rotation=40, ha="right")
plt.show()
```



No, indigo flights are not cheaper than spicejet

7. Are morning flights expensive?

```
#lets check this column againt our target variable
plt.figure(figsize= (15,5))
coll = sns.barplot(x = df['Departure_Part_of_the_day'] , y =df['Price'] )
coll.set_xticklabels(coll.get_xticklabels(), rotation=40, ha="right")
plt.show()
```



Yes, morning flights are expensive

- Interpretation of the Results

I choose Linear Regression as the final model since it has the best r2 score among all the model of 91%. This model is very less difference between cross validation cross and r2 score. So, this model is not over fitted or under fitted.

Hyper parameter tuning of Random Forest Regressor

RandomForestRegressor giving the best Score of 91%

Hyperparameter Tuning:

```
#lets use random forest regressor
#it takes a lot lot of time approx 6 hours, so commented after
parameters = {'n_estimators' : [100,200,300], 'criterion': ['mse', 'mae'], 'max_features': ['auto', 'sqrt', 'log2']}
gsvrf = RandomizedSearchCV(rf, parameters, cv=5, scoring="r2")
gsvrf.fit(x_train,y_train)
print(gsvrf.best_score_)
print(gsvrf.best_params_)

0.8515586124486496
{'n_estimators': 100, 'max_features': 'auto', 'criterion': 'mse'}
```

Output of Random Forest Regressor with best parameters obtained from hyper parameter tuning:

Creating a Final Model as RandomForestRegressor

```
final_model_rf = RandomForestRegressor(n_estimators= 100,criterion= 'mse',max_features = 'auto')
final_model_rf.fit(x_train,y_train)
final_pred = final_model_rf.predict(x_test)
cvs = cross_val_score(final_model_rf,pca_x,Y,cv =5)
print('\n')
print('Scores :')
print('r2 score:',r2_score(y_test,final_pred))
print('Cross Val score :',cvs.mean())
print('Error :')
print('mean absolute error :',mean_absolute_error(y_test,final_pred))
print('mean squared error : ', mean_squared_error(y_test,final_pred))
print('root mean squared error :',np.sqrt(mean_squared_error(y_test,final_pred)))
print('Difference :')
difference = np.abs(r2_score(y_test,final_pred) - cvs.mean())
print('Difference between cross val score and r2 score is : {0:.2f}'.format(difference))
least_difference.append((m,'Difference between cross val score and r2 score is : {0:.2f}'.format(difference)))
```

```
Scores :
r2 score: 0.9112014590749229
Cross Val score : -0.7503076344554618
Error :
mean absolute error : 639.3714366197183
mean squared error : 1090796.134461972
root mean squared error : 1044.4118605521348
Difference :
Difference between cross val score and r2 score is : 1.66
```

Saving the Final model:

Serialization using joblib.

Pickled model as a file using joblib: Joblib is the replacement of pickle as it is more efficient on objects that carry large numpy arrays. These functions also accept file-like object instead of filenames.

Joblib.dump to serialize an object hierarchy.

```
#serialization using joblib
import joblib
joblib.dump(final_model_rf,'Flight_Price_Prediction.obj')
```

```
['Flight_Price_Prediction.obj']
```

```
Flight_Price = joblib.load('Flight_Price_Prediction.obj')
```

```
s_pred = Flight_Price.predict(x_test)
```

```
r2_score(y_test,s_pred)
```

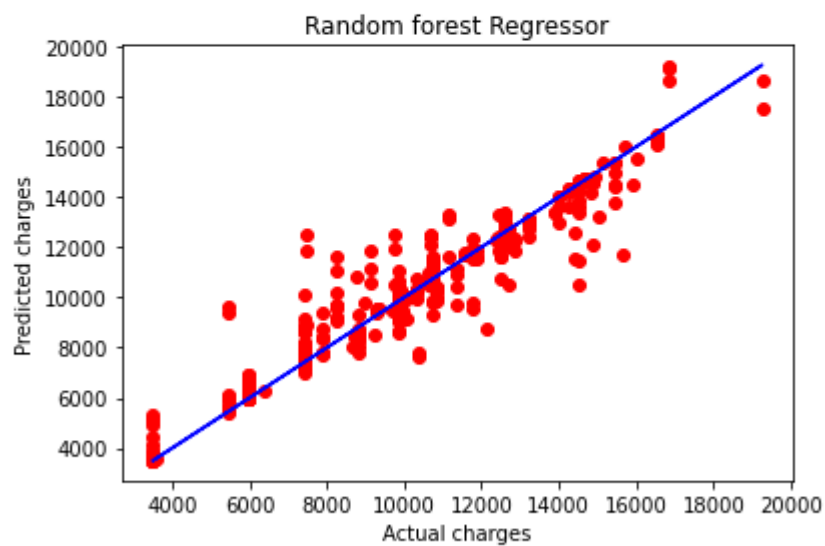
```
0.9112014590749229
```

our r2_score is 91.12% percent

Let's plot `y_test` vs predicted:

```
#lets plot y_test vs predicted

plt.figure(figsize=(6,4))
plt.scatter(x = y_test,y = s_pred,color = 'r')
plt.plot(y_test,y_test,color = 'b')
plt.xlabel('Actual charges')
plt.ylabel('Predicted charges')
plt.title('Random forest Regressor')
plt.show()
```



we can see that values are very close to the line

Final Predicted Car Prices:

```
#lets make a dataframe of actual answers vs predicted answers
conclusion = pd.DataFrame((Flight_Price.predict(x_test)[:],y_test[:]),index= ['Predicted','Actual'])
conclusion
```

	0	1	2	3	4	5	6	7	8	9	...	345	346	347	348	349	...
Predicted	13652.56	12983.39	11904.41	11029.65	9302.93	13155.54	9690.53	3703.46	3644.16	9661.23	...	9730.46	9088.37	9576.42	10442.53	8120.3	1037
Actual	14524.00	12581.00	9726.00	9830.00	10744.00	13206.00	9840.00	3498.00	3498.00	5425.00	...	9840.00	9828.00	9311.00	10744.00	8778.0	1083

CONCLUSION

- **Key Findings and Conclusions of the Study**

Though this is the simplest model we've built till now, the final predictors still seem to have high correlations. One can go ahead and remove some of these features, though that will affect the adjusted- r^2 score significantly (you should try doing that).

Thus, for now, the final model consists of the 11 variables mentioned above.

- **Learning Outcomes of the Study in respect of Data Science**

I choose Random Forest Regressor algorithm as my final model since it was having least difference between its cross validation score and r^2 score.

I learned how to use regular expression to extract the required information from the existing columns

- **Limitations of this work and Scope for Future Work**

Yes there is still room for improvement, like doing a more Web scraping from different websites, extensive feature engineering, by comparing and plotting the features against each other and identifying and removing the noisy features. The values of R-squared obtained from the algorithm give the accuracy of the model. In the future, if more data could be scraped such as the business class, so predicted results will be more accurate.