FLIP ROBO

FLIGHT PRICE PREDICTION

Submitted by:

P. MANIVANNAN

INTRODUCTION

Problem Statement:

Anyone who has booked a flight ticket knows how unexpectedly the prices vary. So, we have to work on a project where we 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

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)

- Review of Literature

The Research is done on Flight Price scrapped from website and how each Features plays a role in Flight Price with Data Science as tool the research is done. First Brief Exploratory Data Analysis is done which includes Handling Null values, Visualizing the Data to Understand Correlation with Flight Ticket price and understanding Relation between various input features. Then Data is Fed to Machine Learning Algorithms. after Evaluating and Testing Different Algorithms Best Algorithm is Selected. Based on Evaluation Metrics such as Training Score, R2 Score, Cross Validation Score etc.

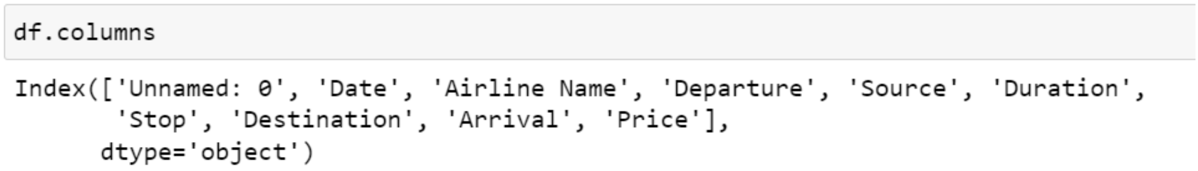
- Motivation for the Problem Undertaken

The Flight price is always related to Time of purchase everyone knows this. but as a data analyst I am not going to make compromise with what others say. I will analyse what other features are contribute to the Flight Price using data analytics.

Analytical Problem Framing

- Data Sources and their formats

The data contains 9910 rows and 10 columns. The columns are



```
df.columns
```

```
Index(['Unnamed: 0', 'Date', 'Airline Name', 'Departure', 'Source', 'Duration',
       'Stop', 'Destination', 'Arrival', 'Price'],
      dtype='object')
```

The Flight Price data is collected from Make My Trip website using web scrapping. data is collected from different cities in India and combined.

The data is in xlsx format.

- Data Pre-processing Done

1. The data has no null values .
2. We had categorical data which needs to be encoded. for this approach I used pd.get_dummies method of pandas library as categorical data are not ordinal in nature.
3. Then Skewness Step is Skipped as the data are categorical in nature.
4. Then outliers Step is also Skipped as the data are categorical in nature.
5. As Multicollinearity is higher we used PCA analysis and reduced the No of columns and handle multicollinearity as well.

- Data Inputs- Logic- Output Relationships

The price of Indigo starts from very low compared to other airline companies. Flights with no stops have less price compared to flights with stops. Based on destination the price of flight varies. Most importantly the price of flight is higher when booking date is too close to departure date and price low when flight ticket is booked in advance about 2 to 3 months. And the Price of very early morning flights before 4.0 AM are comparatively low.

- State the set of assumptions (if any) related to the problem under consideration
    1. For handling the categorical data dummies method is used assuming the categorical data do not have ordinal relationship with each other. as more insight is need to conclude which category is ordinal or not. its safe to use dummies method.
    2. Principal component analysis technique is used to reduce the column count. But the variance of features are retained by using pca.explained_variance_ratio_ method to find the variance exhibited by different features.

- Hardware and Software Requirements and Tools Used

The Hardware requirements are 16 Gb RAM,500 GB SSD, at least 6 Cores processor

The software requirements are Windows, Anaconda Framework and libraries used are Pandas, NumPy, Sklearn, SciPy, Matplotlib, Seaborn

We use Pandas for reading the Dataset, Creating Dataframe, much more to handle data in dataset

Certain functions of NumPy are used like log , cbrt ,sqrt skewness transformation . absolute (abs) function used along with Zscore.

SKlearn Library is used for Machine Learning Algorithms like SVC, Random Forest, Linear Regression etc and Metrics for analysing them.

Scipy is scientific python library used for Zscore method etc.

Matplotlib and Seaborn are used for visualizations plotting graphs charts, etc

Finally joblib module is used for saving our model for future use.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

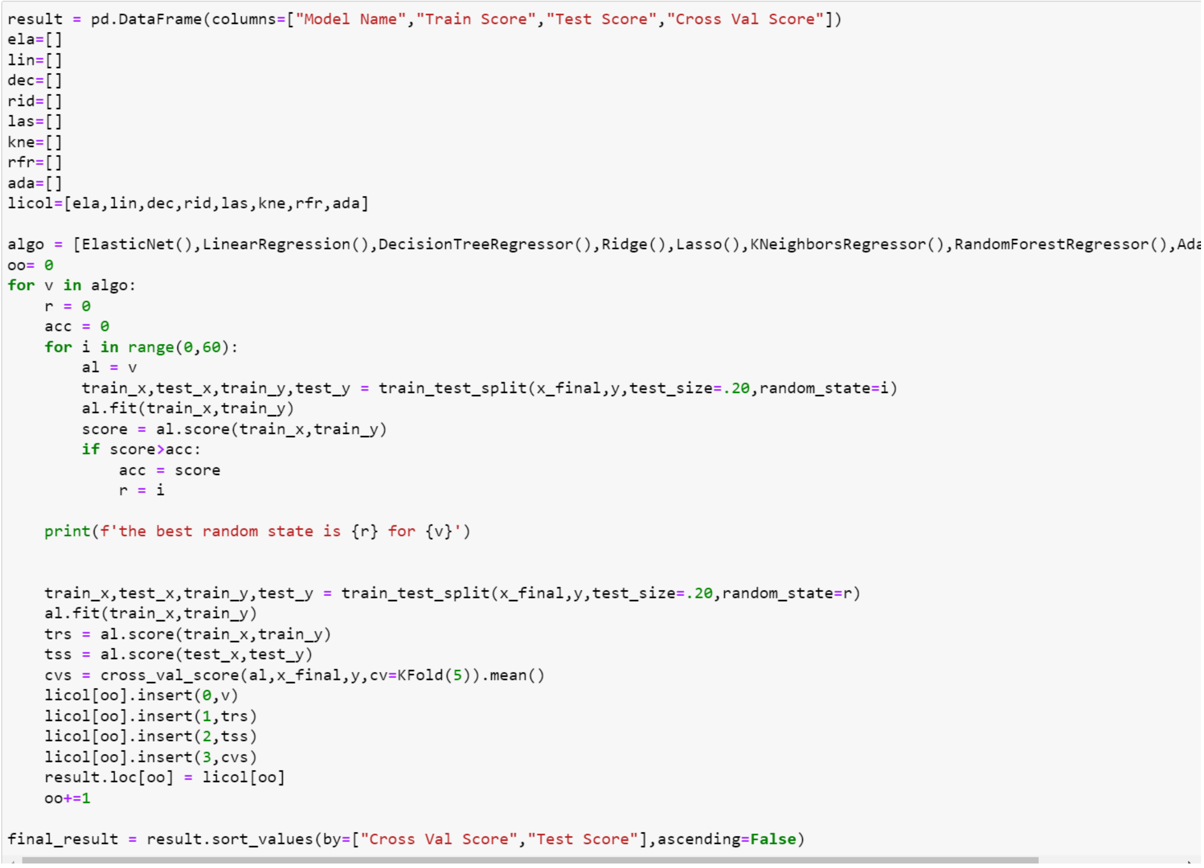
  Since the Target Variable is Continuous in nature I should Use Regression Approach to Solve this Problem and build a Regression Model

- Testing of Identified Approaches (Algorithms)
  1. Linear Regression
  2. Ridge Regression
  3. Lasso Regression
  4. Elastic Net Regression
  5. Decision Tree Regressor
  6. Random Forest Regressor
  7. Ada Boost Regressor
  8. KNeighbours Regressor

- Run and Evaluate selected models

  I have custom written a Code Block for Running all Algorithms and Storing them in DataFrame and Ranking them based on their Accuracy Score. The best part is the code finds the best random state for each algorithm and prints them

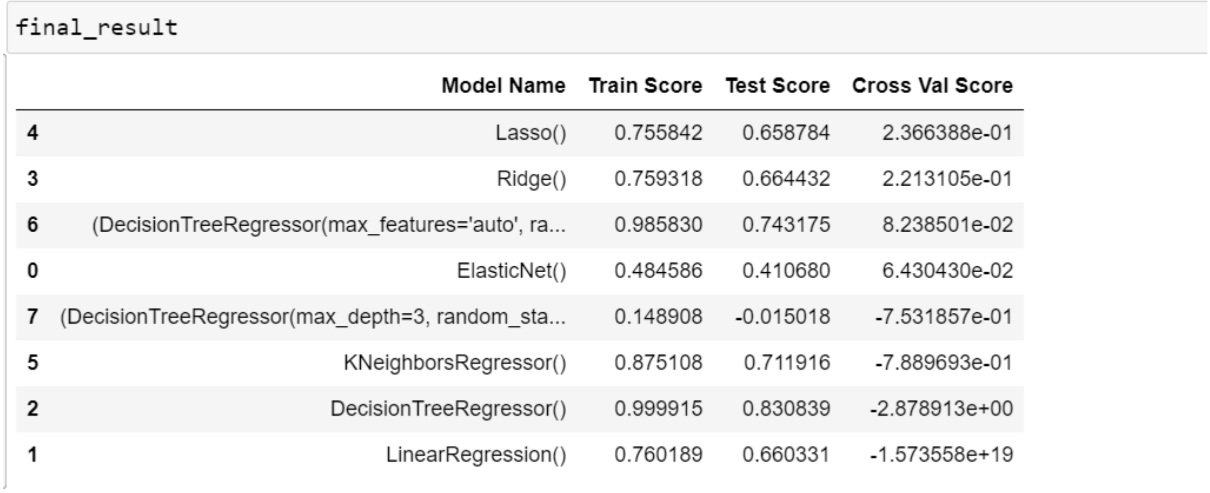
Code:



```
result = pd.DataFrame(columns=["Model Name","Train Score","Test Score","Cross Val Score"])
ela=[]
lin=[]
dec=[]
rid=[]
las=[]
kne=[]
rfr=[]
ada=[]
licol=[ela,lin,dec,rid,las,kne,rfr,ada]

algo = [ElasticNet(),LinearRegression(),DecisionTreeRegressor(),Ridge(),Lasso(),KNeighborsRegressor(),RandomForestRegressor(),Ada
oo= 0
for v in algo:
    r = 0
    acc = 0
    for i in range(0,60):
        al = v
        train_x,test_x,train_y,test_y = train_test_split(x_final,y,test_size=.20,random_state=i)
        al.fit(train_x,train_y)
        score = al.score(train_x,train_y)
        if score>acc:
            acc = score
            r = i

    print(f'the best random state is {r} for {v}')


    train_x,test_x,train_y,test_y = train_test_split(x_final,y,test_size=.20,random_state=r)
    al.fit(train_x,train_y)
    trs = al.score(train_x,train_y)
    tss = al.score(test_x,test_y)
    cvs = cross_val_score(al,x_final,y,cv=KFold(5)).mean()
    licol[oo].insert(0,v)
    licol[oo].insert(1,trs)
    licol[oo].insert(2,tss)
    licol[oo].insert(3,cvs)
    result.loc[oo] = licol[oo]
    oo+=1

final_result = result.sort_values(by=["Cross Val Score","Test Score"],ascending=False)
```

If you see the code I have created a list of algorithm which I need to run. And here is the output:

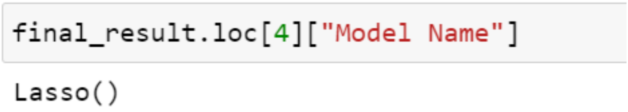
Output1:



```
final_result
```

	Model Name	Train Score	Test Score	Cross Val Score
4	Lasso()	0.755842	0.658784	2.366388e-01
3	Ridge()	0.759318	0.664432	2.213105e-01
6	(DecisionTreeRegressor(max_features='auto', ra...	0.985830	0.743175	8.238501e-02
0	ElasticNet()	0.484586	0.410680	6.430430e-02
7	(DecisionTreeRegressor(max_depth=3, random_sta...	0.148908	-0.015018	-7.531857e-01
5	KNeighborsRegressor()	0.875108	0.711916	-7.889693e-01
2	DecisionTreeRegressor()	0.999915	0.830839	-2.878913e+00
1	LinearRegression()	0.760189	0.660331	-1.573558e+19

Output2:



```
final_result.loc[4]["Model Name"]
```

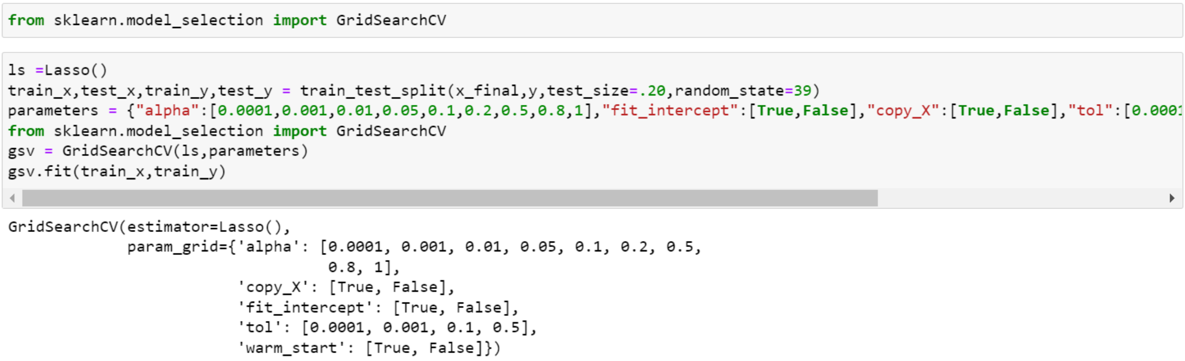
```
Lasso()
```

Here you can see the Model Name, Training Score, Testing Score.

Based on this best model is selected and further proceeded for hyperparameter tuning. Here Lasso Regression performed best compared to others. Even though training and testing score are better for Decision Tree Regressor Corss validation Score is too low for them so Lasso is selected

Hyperparameter Tuning

Code:

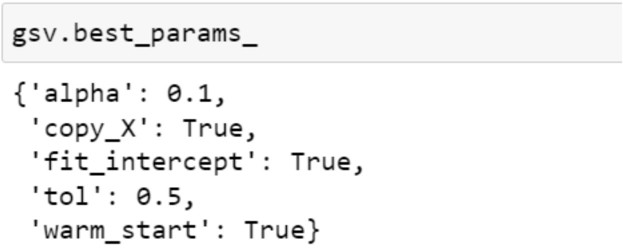


```
from sklearn.model_selection import GridSearchCV
```

```
ls =Lasso()
train_x,test_x,train_y,test_y = train_test_split(x_final,y,test_size=.20,random_state=39)
parameters = {"alpha":[0.0001,0.001,0.01,0.05,0.1,0.2,0.5,0.8,1],"fit_intercept":[True,False],"copy_X":[True,False],"tol":[0.0001
from sklearn.model_selection import GridSearchCV
gsv = GridSearchCV(ls,parameters)
gsv.fit(train_x,train_y)
```

```
GridSearchCV(estimator=Lasso(),
             param_grid={'alpha': [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.5,
                                   0.8, 1],
                         'copy_X': [True, False],
                         'fit_intercept': [True, False],
                         'tol': [0.0001, 0.001, 0.1, 0.5],
                         'warm_start': [True, False]})
```

Output:

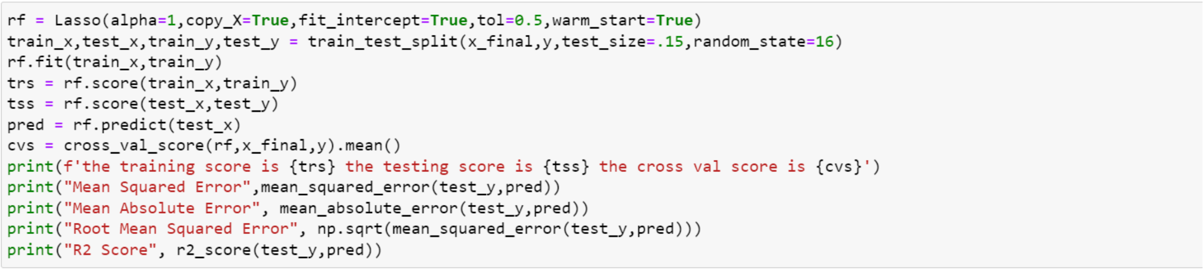


```
gsv.best_params_
```

```
{'alpha': 0.1,
 'copy_X': True,
 'fit_intercept': True,
 'tol': 0.5,
 'warm_start': True}
```

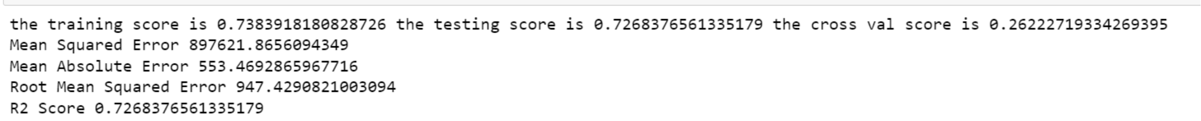
The Best Parameters are listed now we need to run the model with the best parameters

Code:



```
rf = Lasso(alpha=1,copy_X=True,fit_intercept=True,tol=0.5,warm_start=True)
train_x,test_x,train_y,test_y = train_test_split(x_final,y,test_size=.15,random_state=16)
rf.fit(train_x,train_y)
trs = rf.score(train_x,train_y)
tss = rf.score(test_x,test_y)
pred = rf.predict(test_x)
cvs = cross_val_score(rf,x_final,y).mean()
print(f'the training score is {trs} the testing score is {tss} the cross val score is {cvs}')
print("Mean Squared Error",mean_squared_error(test_y,pred))
print("Mean Absolute Error", mean_absolute_error(test_y,pred))
print("Root Mean Squared Error", np.sqrt(mean_squared_error(test_y,pred)))
print("R2 Score", r2_score(test_y,pred))
```

Output:



```
the training score is 0.7383918180828726 the testing score is 0.7268376561335179 the cross val score is 0.26222719334269395
Mean Squared Error 897621.8656094349
Mean Absolute Error 553.4692865967716
Root Mean Squared Error 947.4290821003094
R2 Score 0.7268376561335179
```

The Best Training Score for Lasso Regression After Hyper parameter tuning for Training is 74% , Testing is 73% and Cross Validation Score is 27%

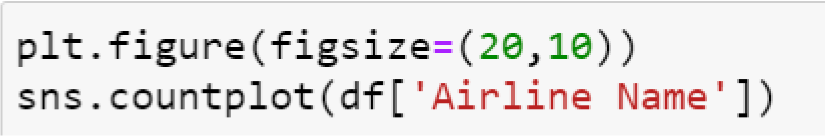
- ## Key Metrics for success in solving problem under consideration

Since it’s a Regression problem The key metrics used are Score method in Regression for training and testing, R2 Score, Mean Squared error, Mean Absolute Error, Root Mean Squared Error.

Visualizations

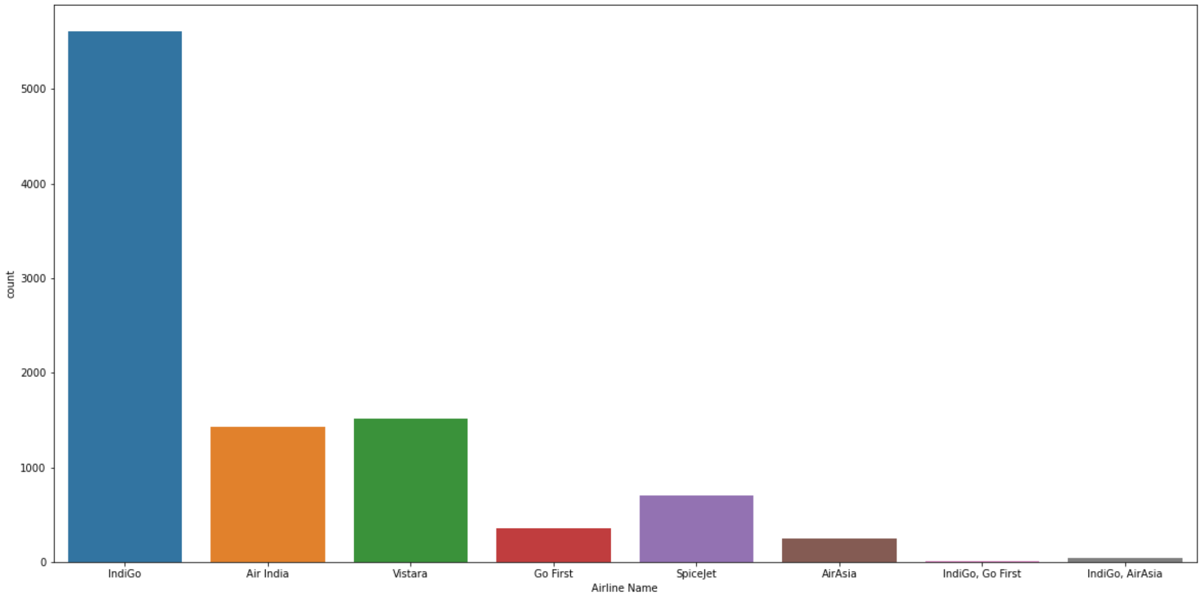
For visualizations the Matplotlib & Seaborn Library are used.

Code:



```
plt.figure(figsize=(20,10))
sns.countplot(df['Airline Name'])
```

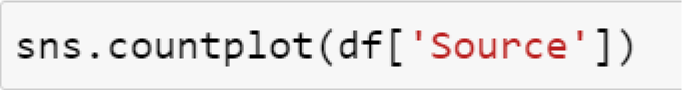
Output:



Observation:

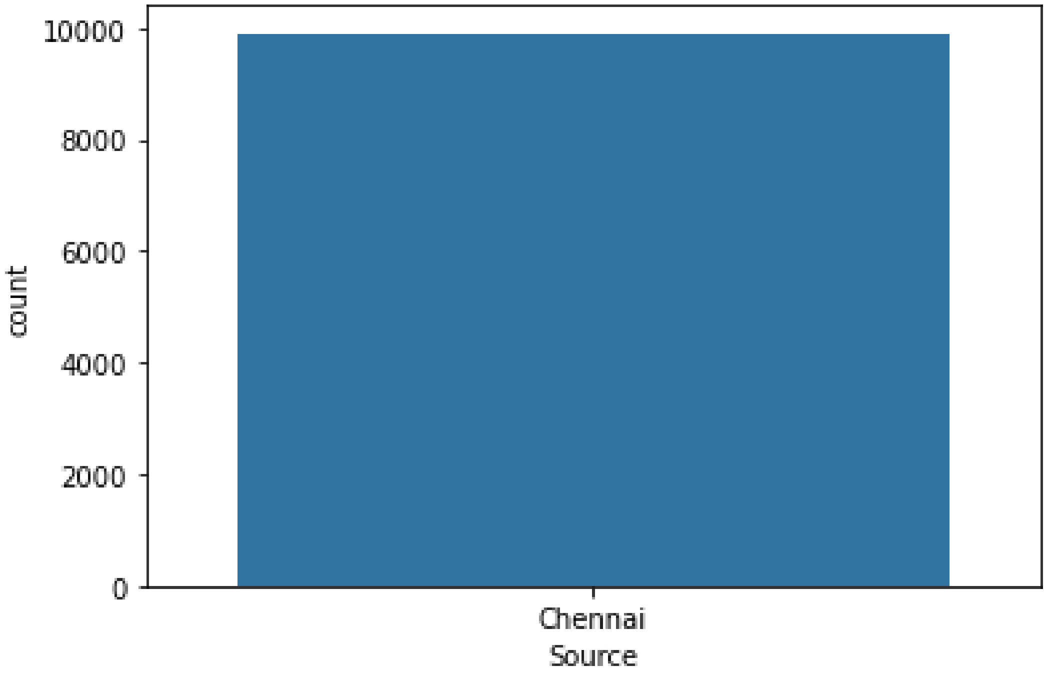
IndiGo seems to have more frequent flights compared to other airlines

Code:



```
sns.countplot(df['Source'])
```

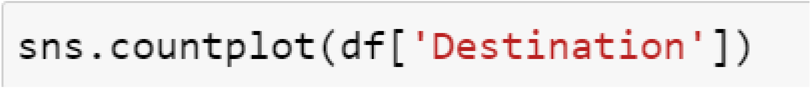
Output:



Observation:

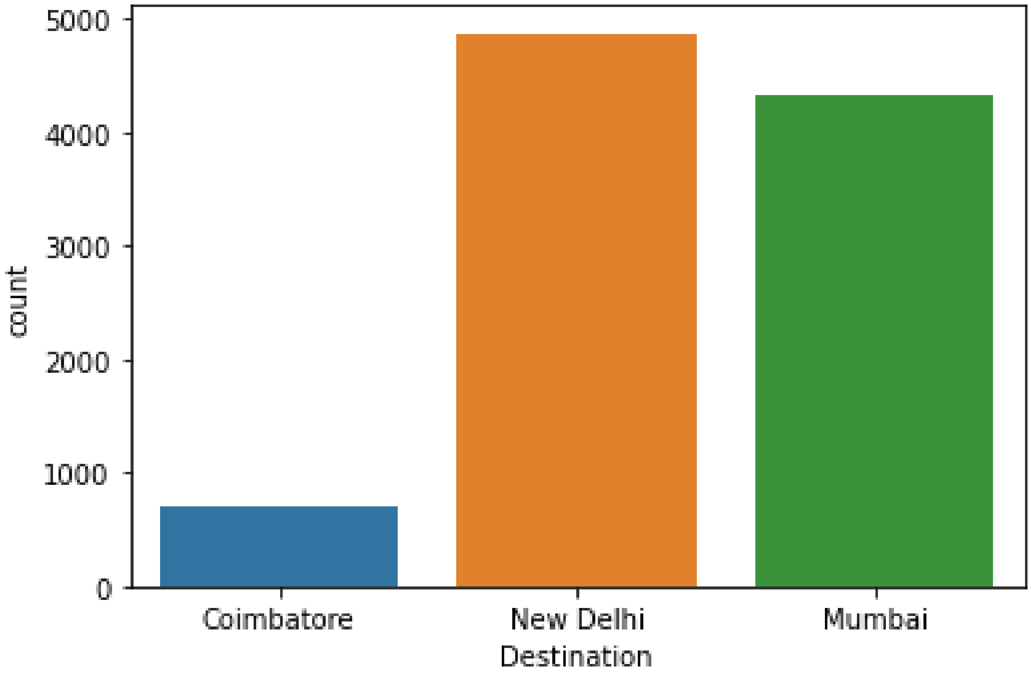
This Data Set Contains Flights Starting From Chennai

Code:



```
sns.countplot(df['Destination'])
```

Output:



Observation:

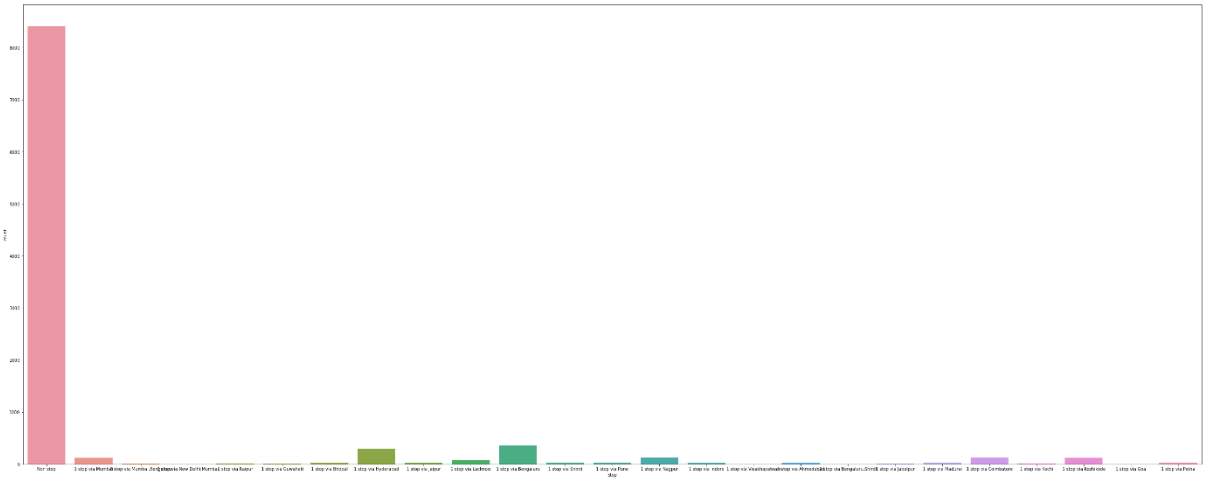
This Dataset contains Flights Going to Coimbatore New Delhi and Mumbai.

The Flights Going to New Delhi is higher compared to other Two Destinations

Code:

```
plt.figure(figsize=(30,20))
sns.countplot(df['Stop'])
```

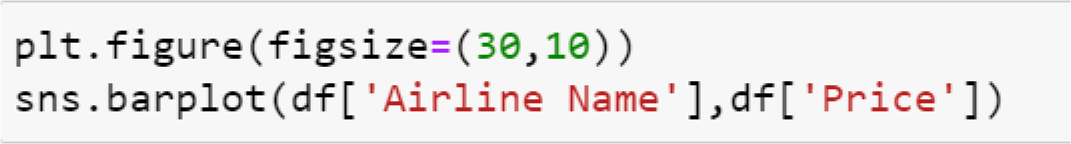
Output



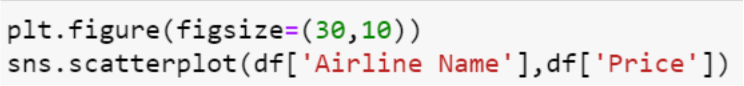
Observation:

This No stop flights are higher in number compared to others

Code:

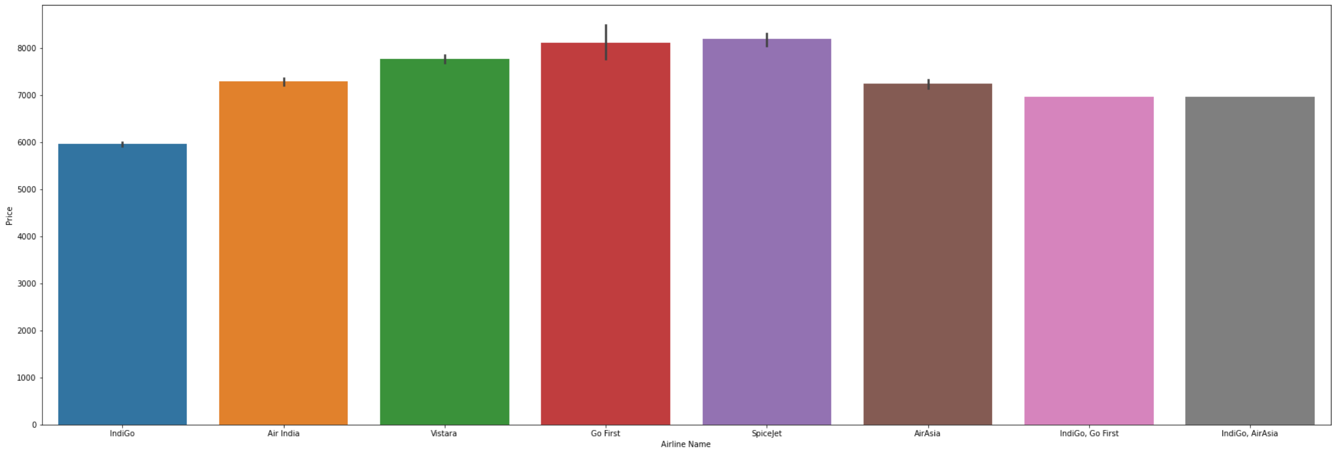


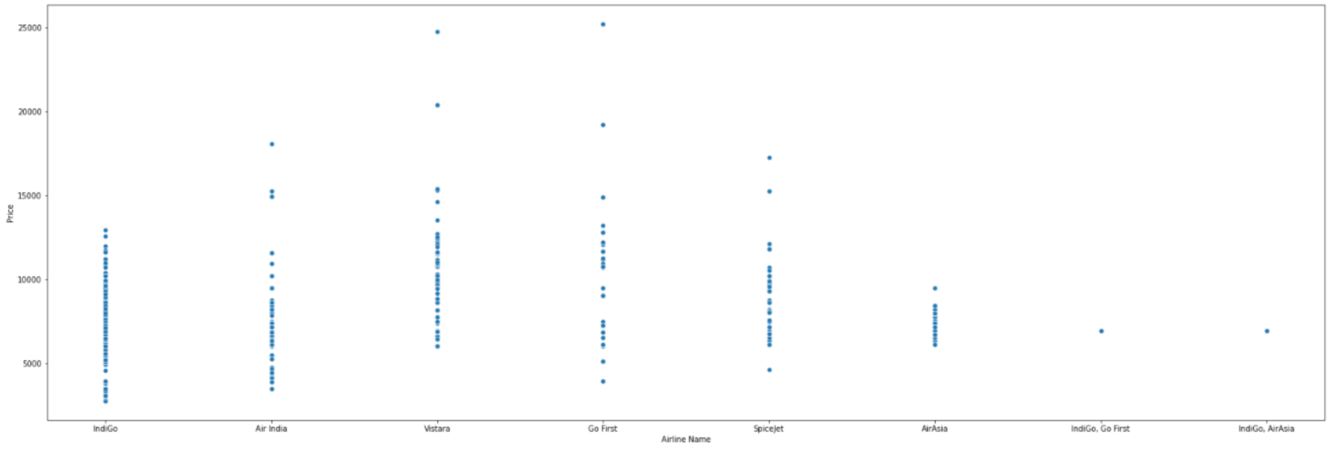
```
plt.figure(figsize=(30,10))
sns.barplot(df['Airline Name'],df['Price'])
```



```
plt.figure(figsize=(30,10))
sns.scatterplot(df['Airline Name'],df['Price'])
```

Output:

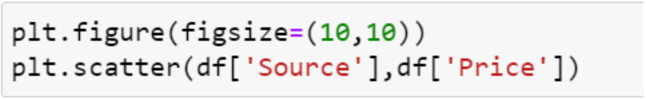




Observation:

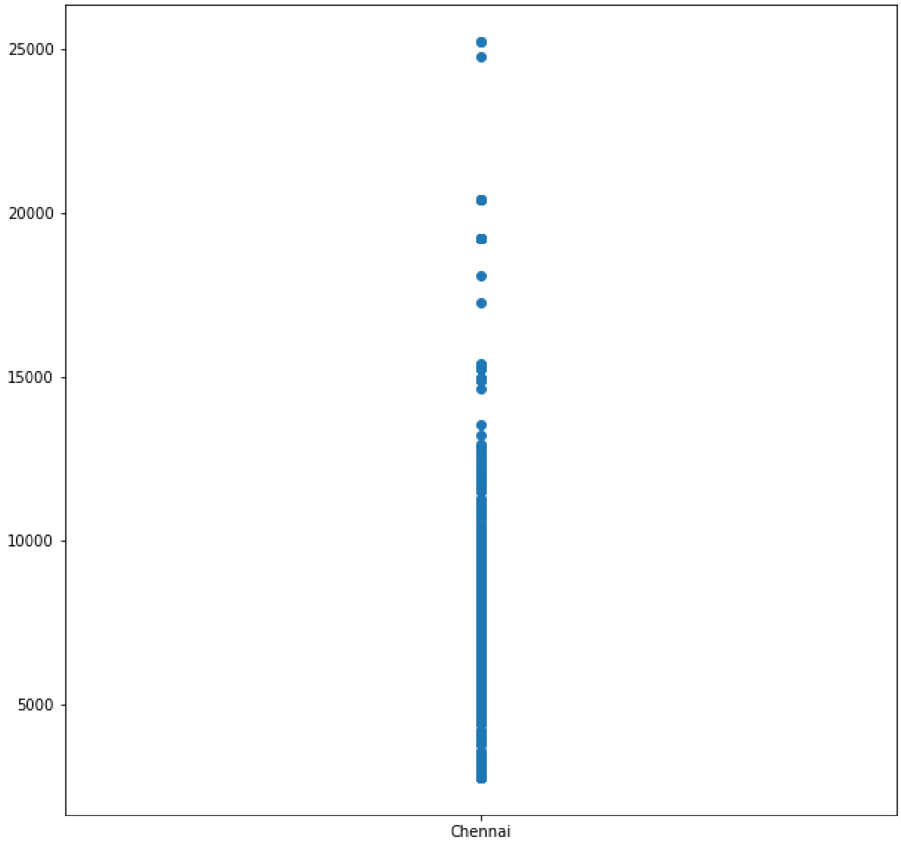
The Price of Indigo is low compared to all other airlines.

Code:



```
plt.figure(figsize=(10,10))
plt.scatter(df['Source'],df['Price'])
```

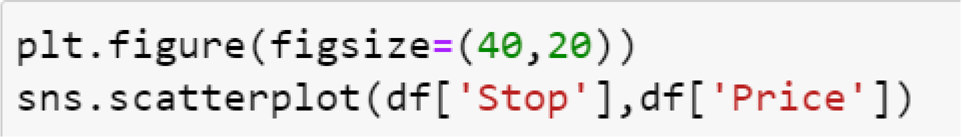
Output:



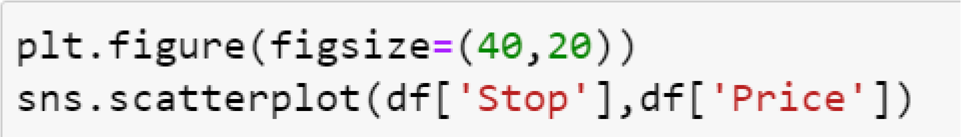
Observation:

The Price of Flights from Chennai to 3 Destinations are below Rs 5000 to Rs 25,000

Code:

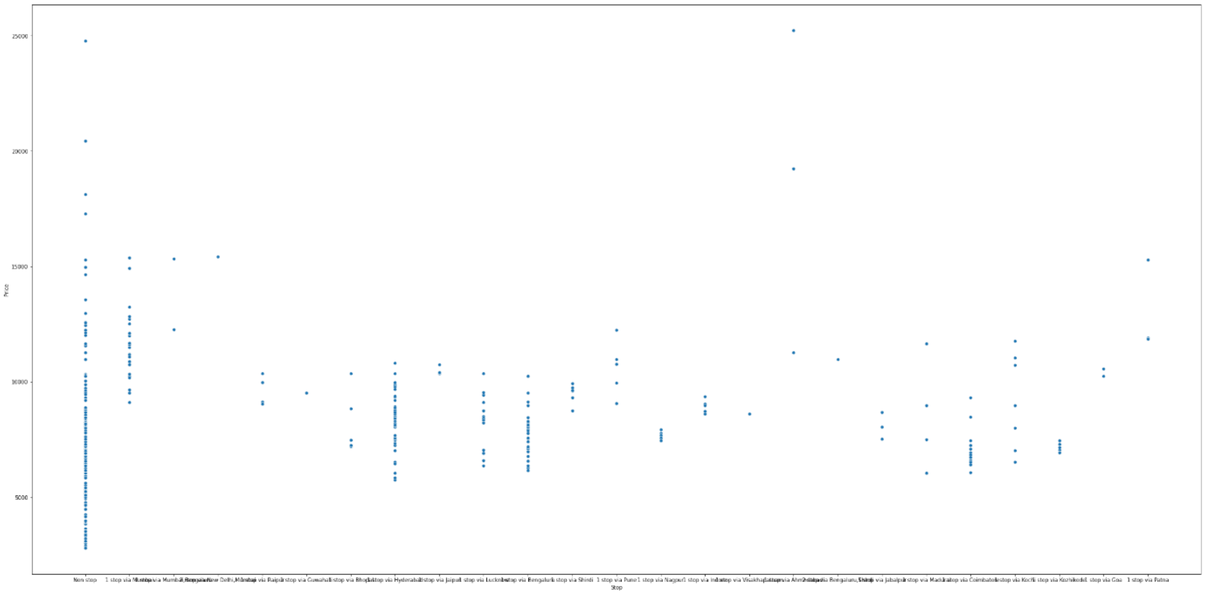


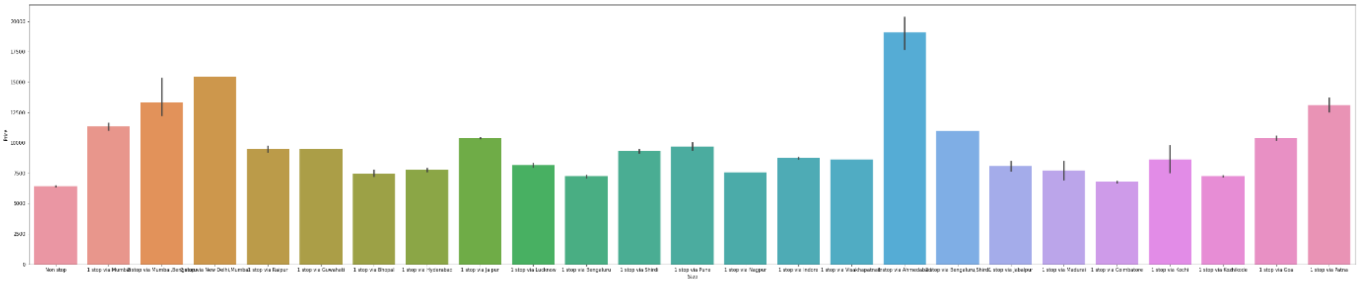
```
plt.figure(figsize=(40,20))
sns.scatterplot(df['Stop'],df['Price'])
```



```
plt.figure(figsize=(40,20))
sns.scatterplot(df['Stop'],df['Price'])
```

Output:

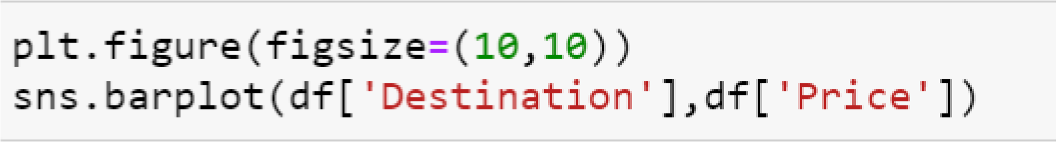




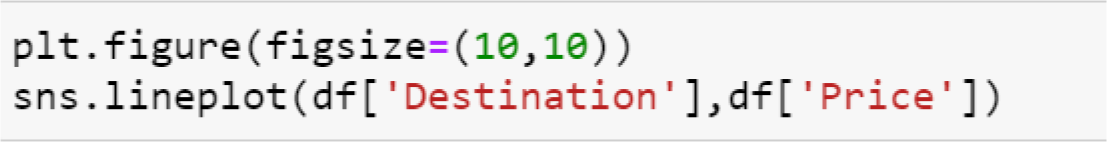
Observation:

For No Stop flight the Price is Lower. as the stop increases the price also increases

Code:

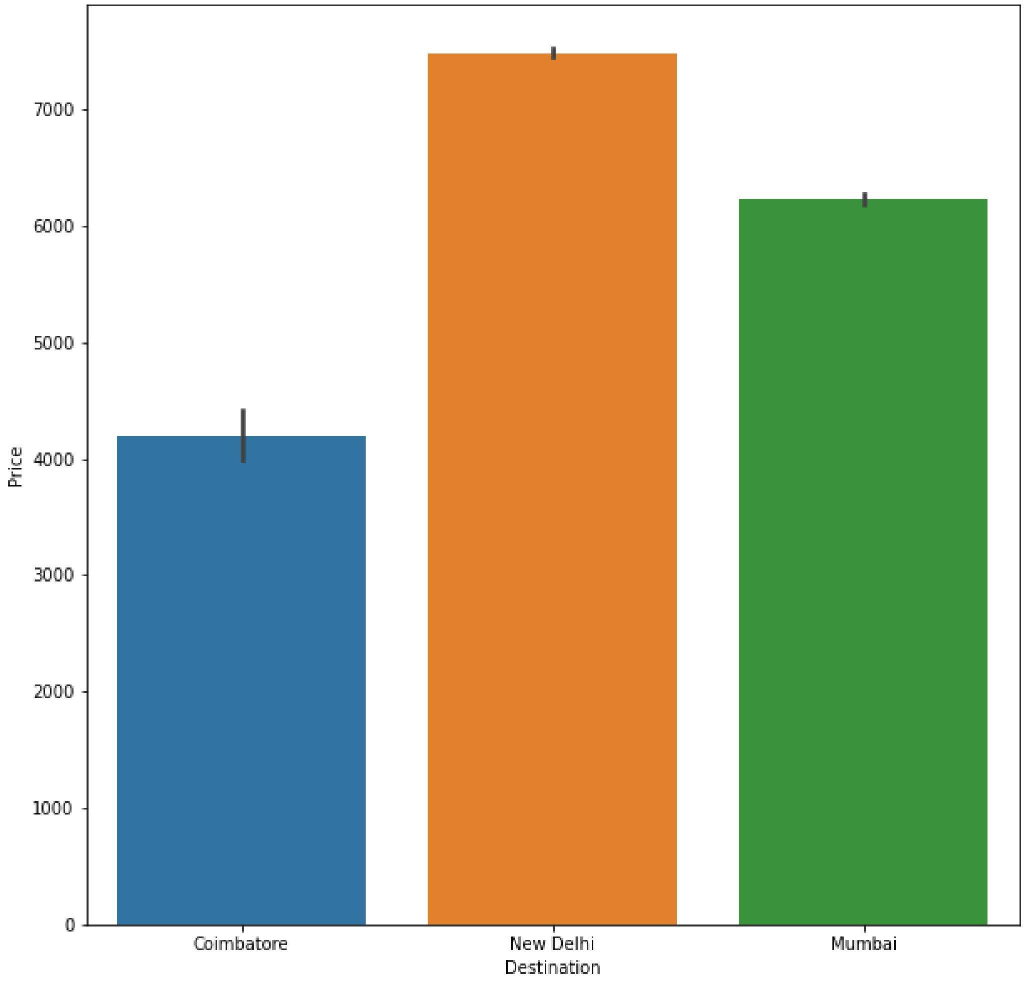


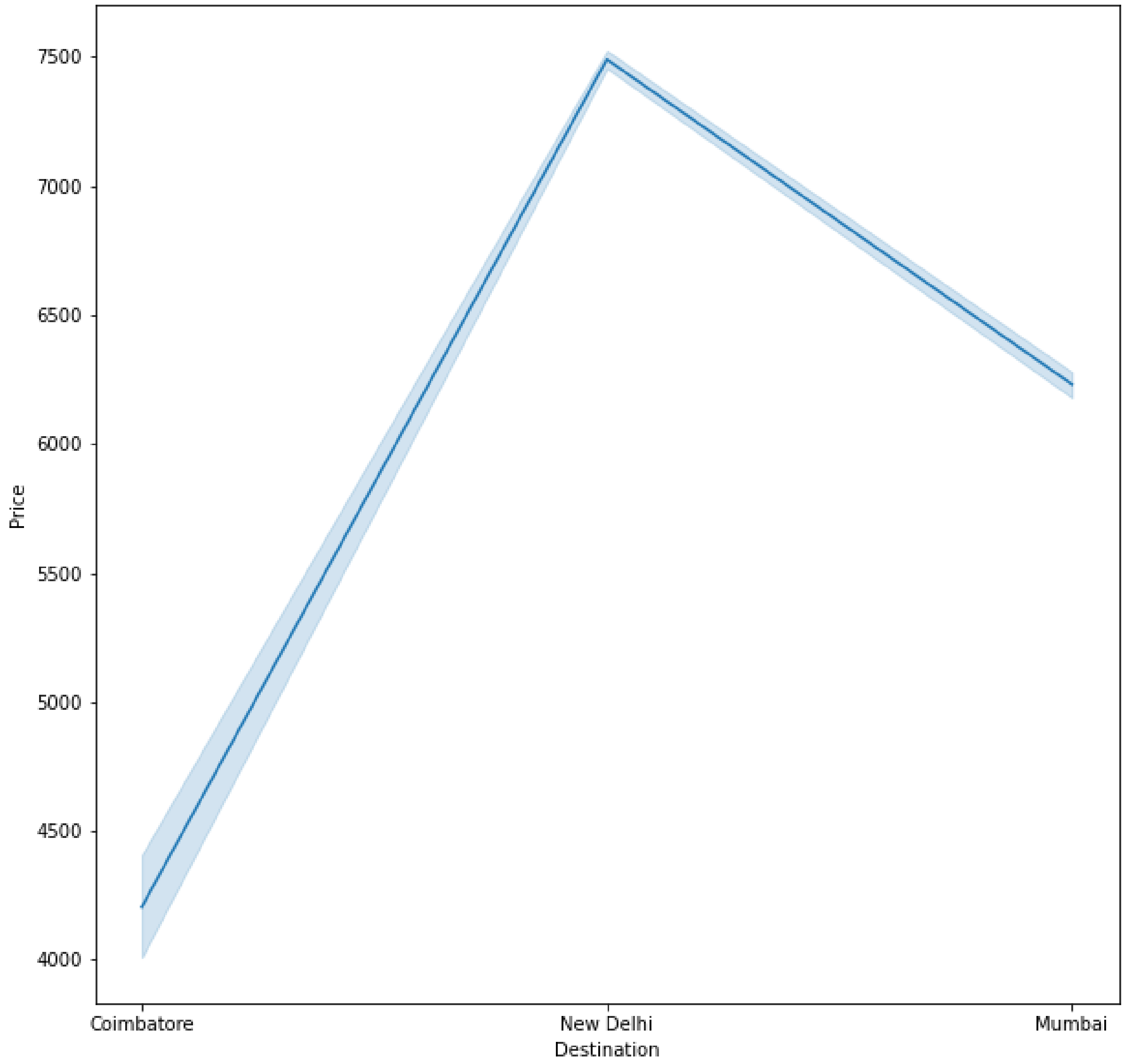
```
plt.figure(figsize=(10,10))
sns.barplot(df['Destination'],df['Price'])
```



```
plt.figure(figsize=(10,10))
sns.lineplot(df['Destination'],df['Price'])
```

Output:

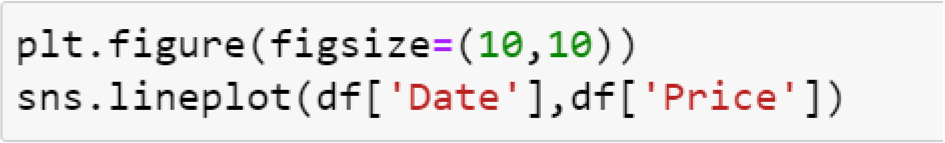




Observation:

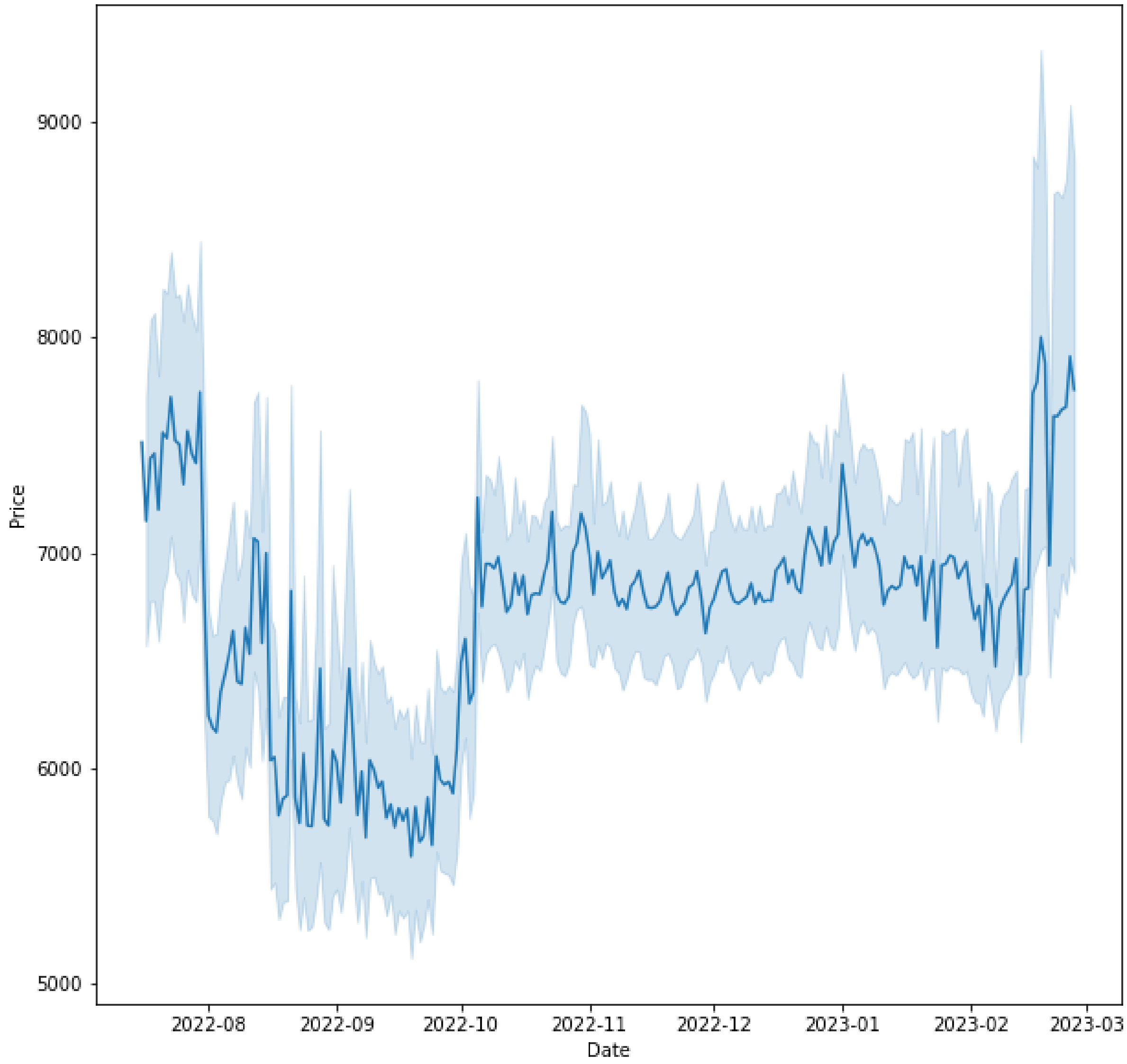
Flights arrival at New Delhi is higher price compared to others

Code:



```
plt.figure(figsize=(10,10))
sns.lineplot(df['Date'],df['Price'])
```

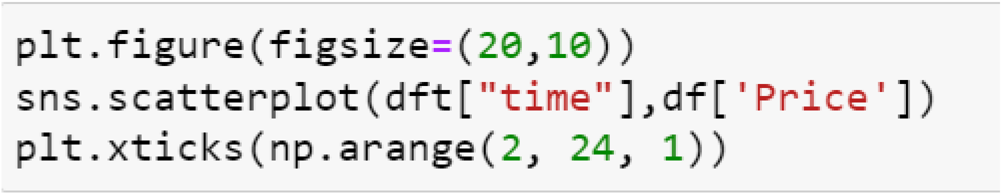
Output:



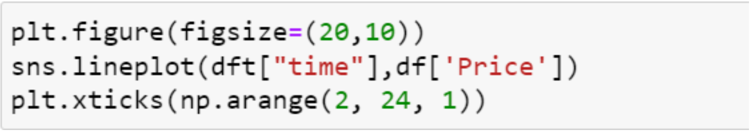
Observation:

The Price of Flight Increases when booked at last minute. the price of flight is low when booked 2 to 3 months before in advance. Interestingly the price is higher when booked too advance also we can see the graph when booked 4-5 months in advance the price is high and price is so high when booked at 7 to 8 months in Early.

Code:

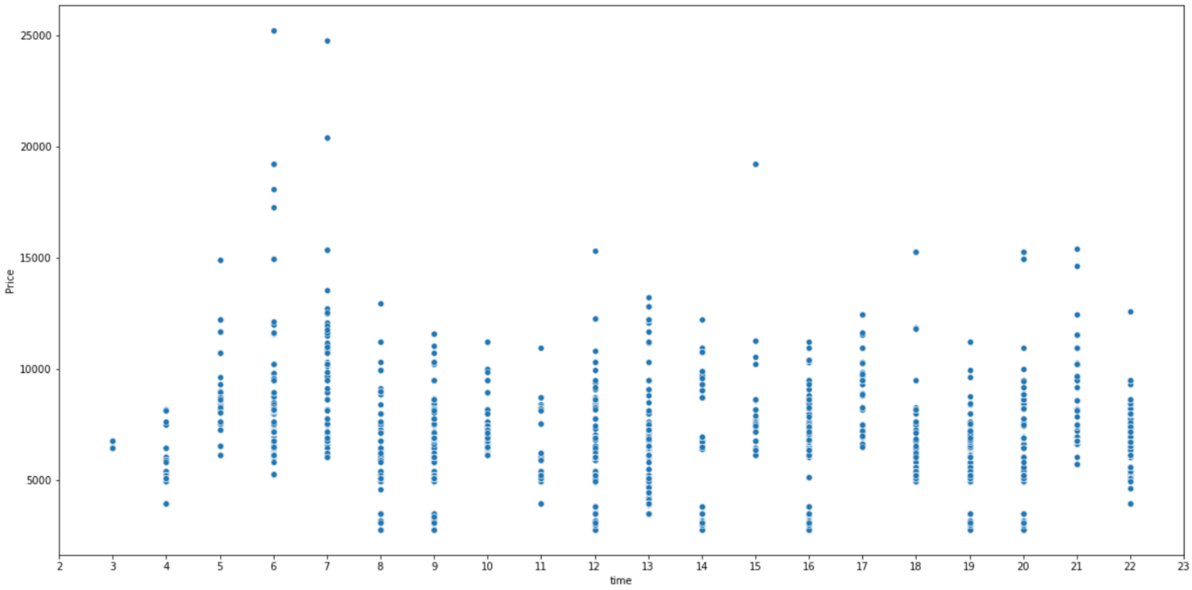


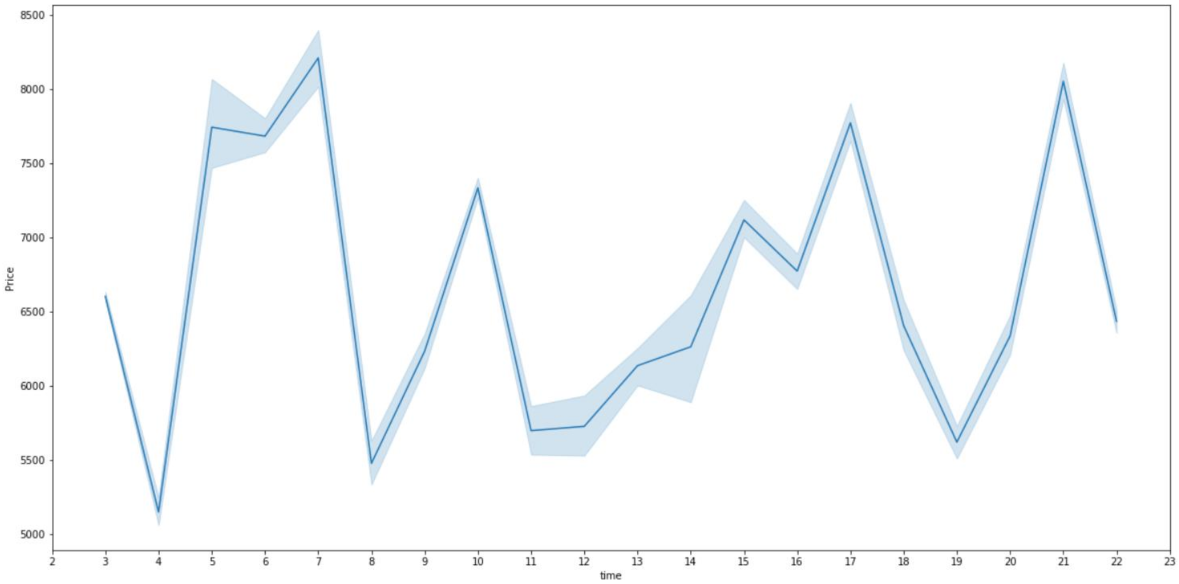
```
plt.figure(figsize=(20,10))
sns.scatterplot(dft["time"],df['Price'])
plt.xticks(np.arange(2, 24, 1))
```



```
plt.figure(figsize=(20,10))
sns.lineplot(dft["time"],df['Price'])
plt.xticks(np.arange(2, 24, 1))
```

Output





Observation:

The Price of Ticket is lower for very early morning and late-Night flights before 4.00AM and after 10.00PM.

- Interpretation of the Results

  Based on the Visualizations I can see when flights booked in advance at 2-3 months the price is low. last minute flight fares are nearly 35% higher compared to advance booking price at 2-3 months. early morning flights are cheap compared to morning flights. before 4.00 AM and after 10.00 PM the flight prices are cheap. Flights with No stop price is low compared to Flights with Stops. Indigo Flight Prices are low compared to other Airlines.

CONCLUSION

- Key Findings and Conclusions of the Study

  Price of Flights when booked at last minute are so high. when flights booked at advance of 2-3 months the price is low. so, the best time to book flight is in advance to 2 to 3 months. The flights without Stops price are less so it is advisable to book flights without stops. Very early morning flights are cheap before 4.00AM or after 10.00PM. Indigo Flight is cheap compared to other Airlines.

- Learning Outcomes of the Study in respect of Data Science

  When flights booked at advance at 2-3 months the price is low. last minute flight fares are nearly 35% higher compared to advance booking price at 2-3 months. also, when booking too early before 4-8 months the price is 20% higher compared to advance booking at 2-3 months. interestingly when booking at 10 - 12 months in advance the price is higher up to 35%. so, the best time to book flight is in advance to 2 to 3 months.

- Limitations of this work and Scope for Future Work

  The Data Collected is not diversified across all places in India. So in depth analysis can be done after collecting data across India.