FLIGHT PRICE PREDICTION

MADE BY:

SHUBHAM SHUKLA

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).

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.

WEB SCRAPING FOR DATA COLLECTION

```
airline_name=[]
date_of_journey=[]
source=[]
destination=[]
departure_time=[]
arrival_time=[]
duration=[]
total_stops=[]
price=[]



an=driver.find_elements_by_xpath('//span[@class="i-b text ellipsis"]')
for i in an:
    try:
        airline_name.append(i.text)
    except NoSuchElementException:
        airline_name.append('-')


s = driver.find_elements_by_xpath('//div[@class="i-b col-4 no-wrap text-right dtime col-3"]/p')
for i in s:
    try:

        source.append(i.text)
    except NoSuchElementException:
        source.append('-')


d = driver.find_elements_by_xpath("//div[@class='i-b pdd-0 text-left atime col-5']/p[2]")
for i in d:
    try:
        destination.append(i.text)
    except NoSuchElementException:
        destination.append('-')
```

```
de = driver.find_elements_by_xpath("//div[@class='i-b col-4 no-wrap text-right dtime col-3']/div")
for i in de:
    try:

        departure_time.append(i.text)
    except NoSuchElementException:
        departure_time.append('-')


a = driver.find_elements_by_xpath("//div[@class='i-b pdd-0 text-left atime col-5']/p[1]")
for i in a:
    try:

        arrival_time.append(i.text)
    except NoSuchElementException:
        arrival_time.append('-')


du = driver.find_elements_by_xpath('//div[@class="stop-cont pl-13"]/p')
for i in du:
    try:

        duration.append(i.text)
    except NoSuchElementException:
        duration.append('-')



ts = driver.find_elements_by_xpath('//div[@class="stop-cont pl-13"]/div')
for i in ts:
    try:

        total_stops.append(i.text)
    except NoSuchElementException:
        total_stops.append('-')
```

HERE I HAVE USED SELENIUM FOR WEB SCRAPING.

DATA LOADING

```
data=pd.read_csv('flight_data.csv')
```

```
data
```

	Unnamed: 0	Unnamed: 0.1	Airline	Source	Destination	Dep_Time	Arrival_Time	Duration	Total_Stops	Price
0	0	0	Jet Airways	Banglore	Delhi	18:55	22:00	3h 5m	non-stop	7229
1	1	1	Multiple carriers	Delhi	Cochin	10:20	01:30 22 May	15h 10m	1 stop	7485
2	2	2	IndiGo	Banglore	Delhi	18:55	21:50	2h 55m	non-stop	4823
3	3	3	Air India	Delhi	Cochin	05:55	07:40 07 Mar	25h 45m	2 stops	14641
4	4	4	SpiceJet	Kolkata	Banglore	06:55	09:30	2h 35m	non-stop	3841
...	...	...	...	...	...	...	...	...	...	...
1669	1669	1669	IndiGo	Banglore	Delhi	04:00	06:50	2h 50m	non-stop	4423
1670	1670	1670	Jet Airways	Kolkata	Banglore	08:25	18:15	9h 50m	1 stop	10844
1671	1671	1671	Jet Airways	Delhi	Cochin	19:30	12:35 28 Jun	17h 5m	2 stops	13764
1672	1672	1672	Air India	Delhi	Cochin	23:00	19:15 10 Mar	20h 15m	1 stop	11260
1673	1673	1673	Jet Airways	Kolkata	Banglore	16:30	20:45 13 May	28h 15m	1 stop	10844

1674 rows × 10 columns

```
data.describe()
```

	Unnamed: 0	Unnamed: 0.1	Price
count	1674.000000	1674.000000	1674.000000
mean	836.500000	836.500000	9083.091995
std	483.386491	483.386491	4475.845590
min	0.000000	0.000000	1965.000000
25%	418.250000	418.250000	5403.000000
50%	836.500000	836.500000	8529.000000
75%	1254.750000	1254.750000	12361.500000
max	1673.000000	1673.000000	57209.000000

```
data.isna().sum()
```

```
Unnamed: 0      0
Unnamed: 0.1    0
Airline         0
Source          0
Destination     0
Dep_Time        0
Arrival_Time    0
Duration        0
Total_Stops     0
Price           0
dtype: int64
```

There are no missing values

DATA PRE-PROCESSING

```
def preprocess1(df):
  df['Total_Stops']=df['Total_Stops'].fillna(df['Total_Stops'].mode()[0])
  df=df.drop(['Duration'],axis=1)
  return df
data=preprocess1(data)
data
```

	Unnamed: 0	Unnamed: 0.1	Airline	Source	Destination	Dep_Time	Arrival_Time	Total_Stops	Price
0	0	0	Jet Airways	Banglore	Delhi	18:55	22:00	non-stop	7229
1	1	1	Multiple carriers	Delhi	Cochin	10:20	01:30 22 May	1 stop	7485
2	2	2	IndiGo	Banglore	Delhi	18:55	21:50	non-stop	4823
3	3	3	Air India	Delhi	Cochin	05:55	07:40 07 Mar	2 stops	14641
4	4	4	SpiceJet	Kolkata	Banglore	06:55	09:30	non-stop	3841
...	...	...	...	...	...	...	...	...	...
1669	1669	1669	IndiGo	Banglore	Delhi	04:00	06:50	non-stop	4423
1670	1670	1670	Jet Airways	Kolkata	Banglore	08:25	18:15	1 stop	10844
1671	1671	1671	Jet Airways	Delhi	Cochin	19:30	12:35 28 Jun	2 stops	13764
1672	1672	1672	Air India	Delhi	Cochin	23:00	19:15 10 Mar	1 stop	11260
1673	1673	1673	Jet Airways	Kolkata	Banglore	16:30	20:45 13 May	1 stop	10844

1674 rows × 9 columns

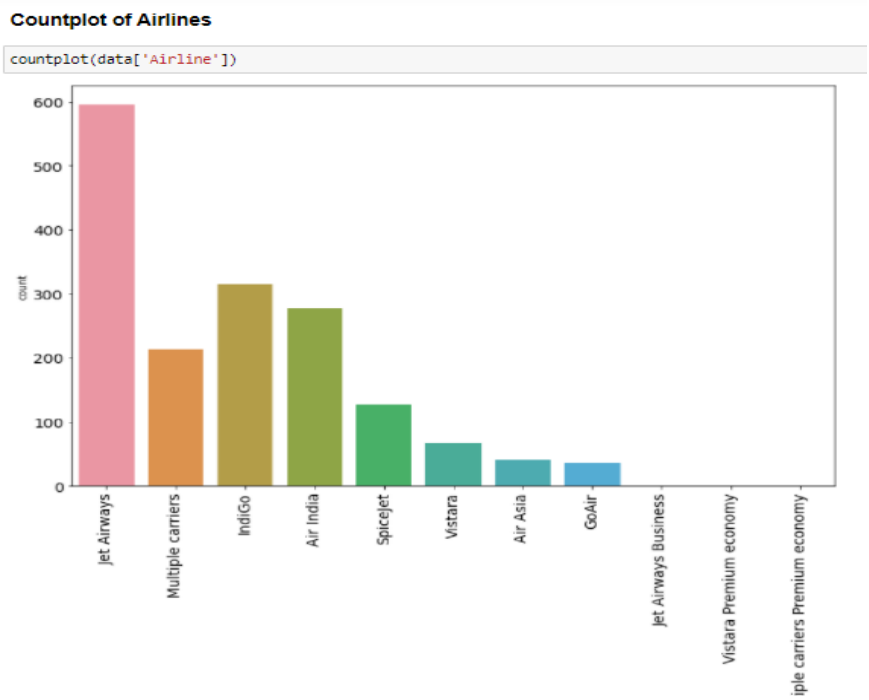
```
def preprocess2(df):
  df['Dep_hour']=pd.to_datetime(df['Dep_Time']).dt.hour
  df['Dep_minute']=pd.to_datetime(df['Dep_Time']).dt.minute
  df=df.drop(['Dep_Time'],axis=1)
  df['arrival_hour']=pd.to_datetime(df['Arrival_Time']).dt.hour
  df['arrival_minute']=pd.to_datetime(df['Arrival_Time']).dt.minute
  df=df.drop(['Arrival_Time'],axis=1)
  return df
data=preprocess2(data)
data
```

	Unnamed: 0	Unnamed: 0.1	Airline	Source	Destination	Total_Stops	Price	Dep_hour	Dep_minute	arrival_hour	arrival_minute
0	0	0	Jet Airways	Banglore	Delhi	non-stop	7229	18	55	22	0
1	1	1	Multiple carriers	Delhi	Cochin	1 stop	7485	10	20	1	30
2	2	2	IndiGo	Banglore	Delhi	non-stop	4823	18	55	21	50
3	3	3	Air India	Delhi	Cochin	2 stops	14641	5	55	7	40
4	4	4	SpiceJet	Kolkata	Banglore	non-stop	3841	6	55	9	30
...	...	...	...	...	...	...	...	...	...	...	...
1669	1669	1669	IndiGo	Banglore	Delhi	non-stop	4423	4	0	6	50
1670	1670	1670	Jet Airways	Kolkata	Banglore	1 stop	10844	8	25	18	15
1671	1671	1671	Jet Airways	Delhi	Cochin	2 stops	13764	19	30	12	35
1672	1672	1672	Air India	Delhi	Cochin	1 stop	11260	23	0	19	15
1673	1673	1673	Jet Airways	Kolkata	Banglore	1 stop	10844	16	30	20	45

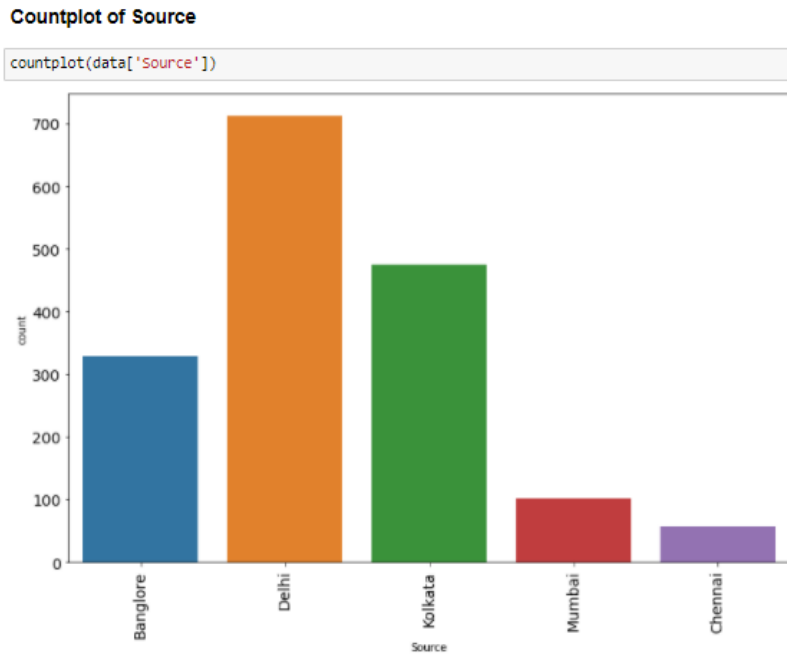
1674 rows × 11 columns

Here I have done feature engineering using the present columns only and making my data more stable for building machine learning model.

Exploratory Data Analysis



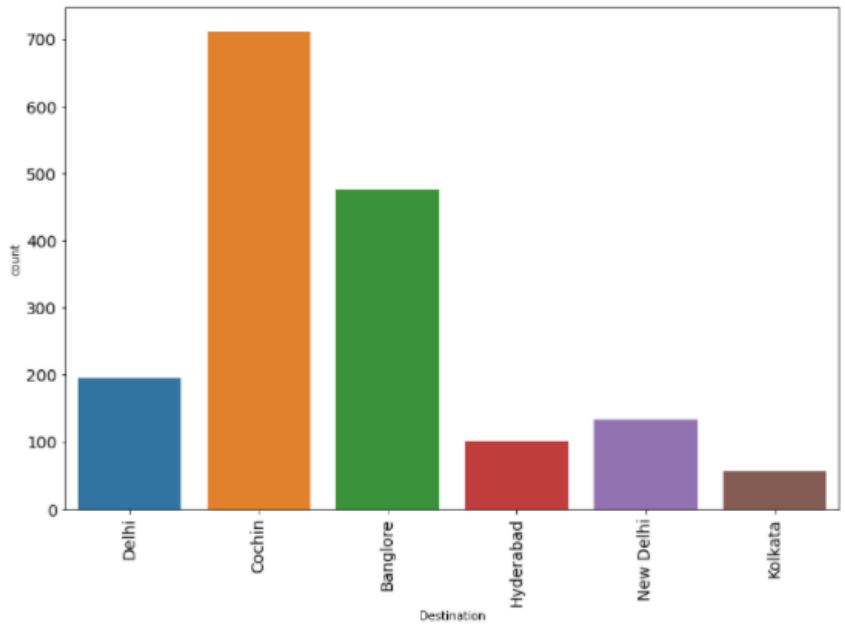
Maximum passengers travel with Jet Airways.



Delhi has the maximum count.

Countplot of Destination

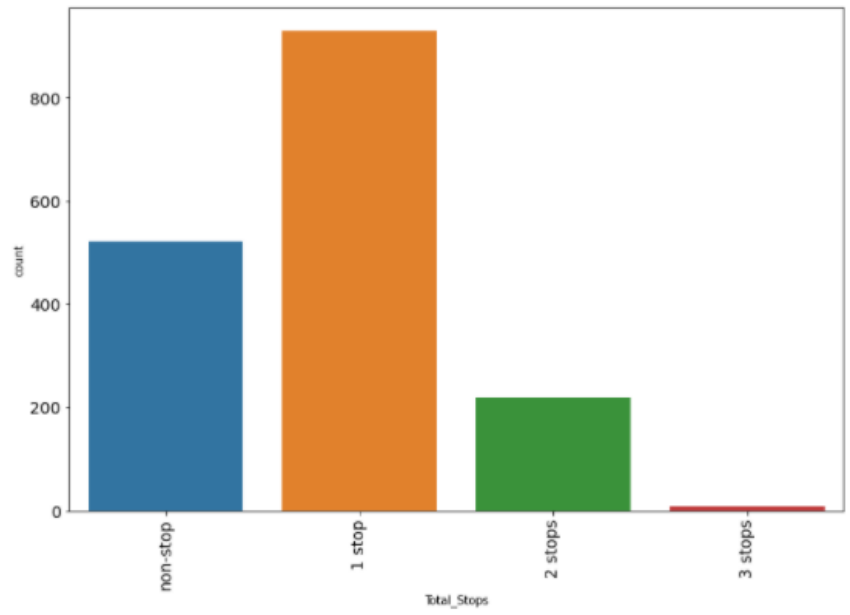
```
countplot(data['Destination'])
```



Maximum people are going to Cochin.

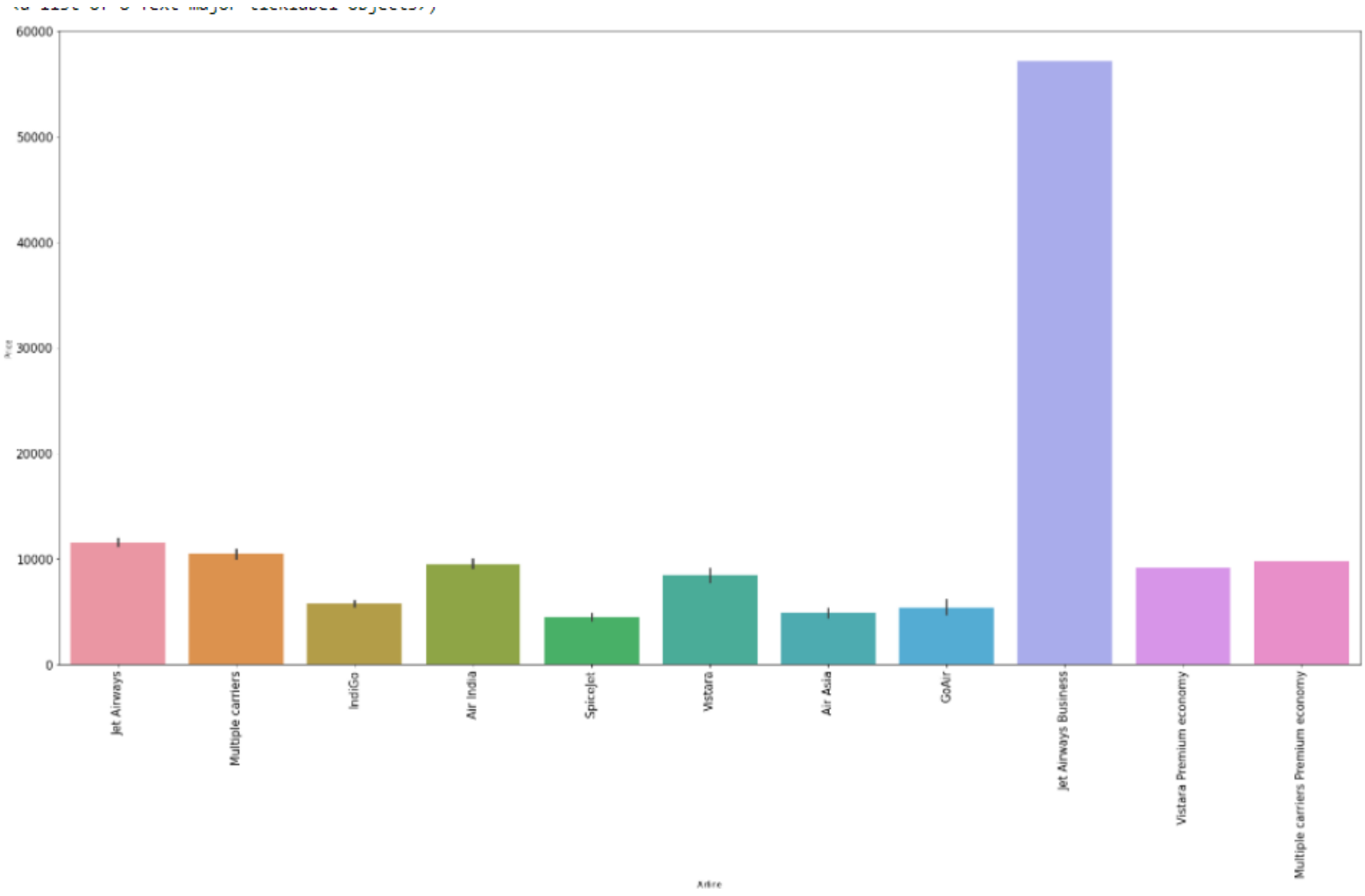
Countplot of Total_Stops

```
countplot(data['Total_Stops'])
```



Most of the flights have only 1 stop.

AIRLINE VS PRICE



Jet Airways Business has the highest price.

PREPARAING DATA FOR MACHINE LEARNING

Feature Transformation

```
oe=OrdinalEncoder()
def ordinal_encoder(df,col):
  df[col]=oe.fit_transform(df[col])
  return df
```

USing ordinal encoder because i have features who have values in order

```
data=ordinal_encoder(data,['Airline','Source','Destination','Total_Stops'])
```

Splitting data:

```
def preprocess3(df):
  df=df.copy()
  X=df.drop(['Price'],axis=1)
  y=df['Price']
  return X,y
```

:
```
X,y=preprocess3(data)
```

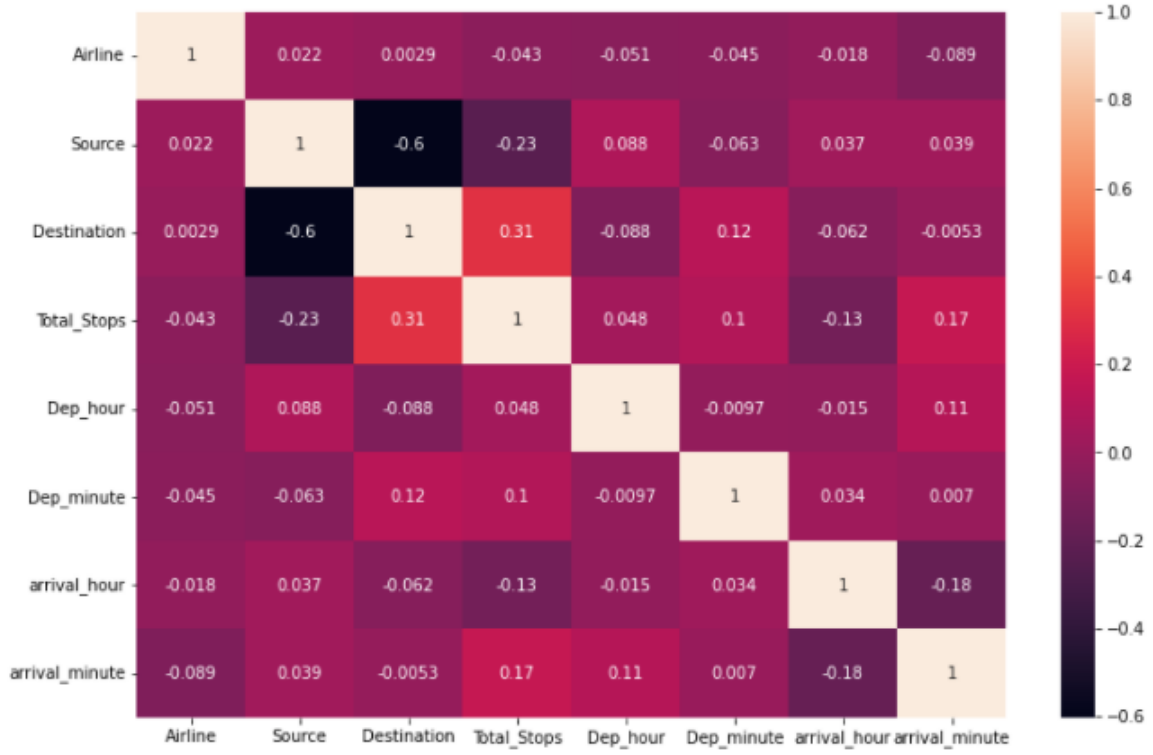
:
```
X.drop(['Unnamed: 0','Unnamed: 0.1'],axis=1,inplace=True)
```

:
```
X_train,X_test,y_train,y_test=train_test_split(X,np.log(y),test_size=0.3,random_state=42)
```

CORRELATION

```
X_corr=X.corr()
plt.figure(figsize=(12,8))
sns.heatmap(X_corr,annot=True)
```

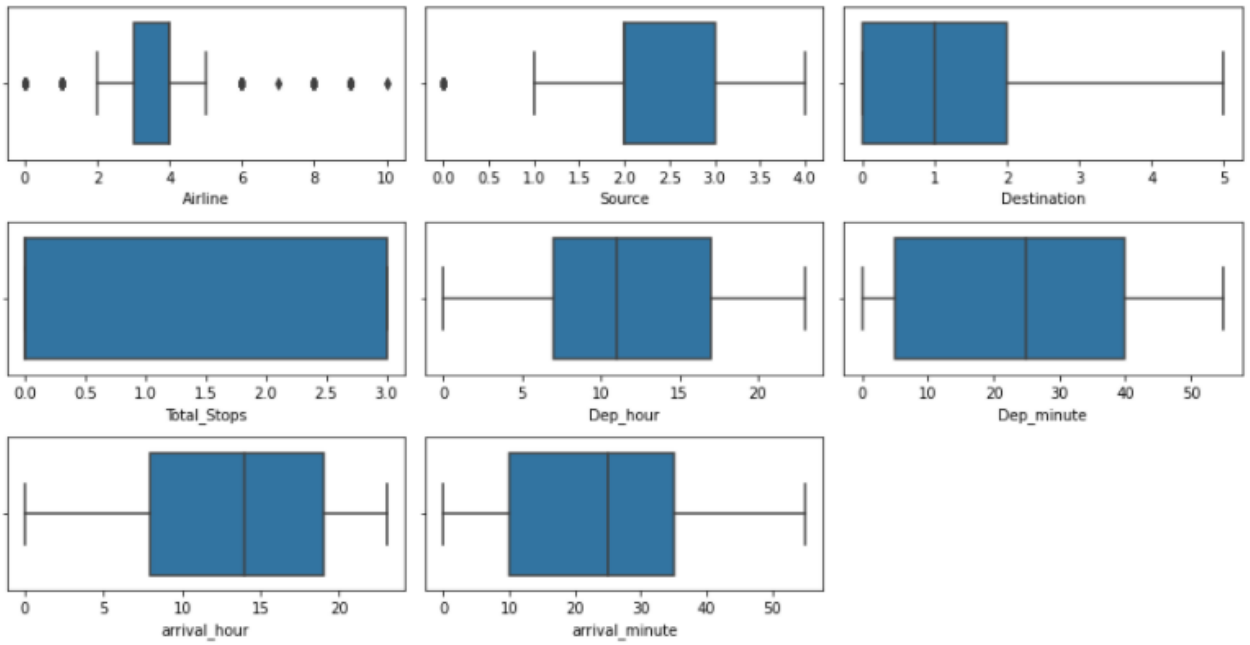
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4a9df0a050>
```



Most of the columns have less correlation in upper graph.

OUTLIERS

```
X_val=X.columns.values
plt.figure(figsize=(12,20))
for i in range(len(X_val)):
  plt.subplot(10,3,i+1)
  sns.boxplot(X[X_val[i]])
  plt.tight_layout()
```



There is no outliers present in the data.

MODEL BUILDING

Training Multiple models

```
models={
     "XGB Regressor":XGBRegressor(),
     "ExtraTrees Regressor":ExtraTreesRegressor(),
     "RandomForest Regressor":RandomForestRegressor(),
     "Linear Regression":LinearRegression(),
     "DecisionTree Regressor":DecisionTreeRegressor(),
     "Lasso":Lasso(),
     "LIGHT GBM":LGBMRegressor()
}
```

```
CVS=[]
R2=[]
MSE=[]
MAE=[]
RMSE=[]
NAME=[]
kf=KFold(n_splits=5,shuffle=True)
for name, model in models.items():
  font=pyfiglet.figlet_format(name)
  print(font)
  NAME.append(name)
  model.fit(X_train,y_train)
  y_pred=model.predict(X_test)
  mse=mean_squared_error(y_test,y_pred)
  MSE.append(mse)
  print("MEAN SQUARED ERROR",mse)
  mae=mean_absolute_error(y_test,y_pred)
  MAE.append(mae)
  print('\n')
  print("MEAN ABSOLUTE ERROR",mae)
  cvs=cross_val_score(model,X,np.log(y),scoring='r2',cv=kf).mean()
  CVS.append(cvs)
  print('\n')
  print("CVS_SCORE",cvs)
  r2=r2_score(y_test,y_pred)
  R2.append(r2)
  print('\n')
  print("R2_SCORE",r2)
  rmse=np.sqrt(mse)
  RMSE.append(rmse)
  print('\n')
  print("RMSE",rmse)
  print('\n')
  print('MODEL PERFORMANCE CURVE')
  skplt.estimators.plot_learning_curve(model,X,np.log(y),cv=kf,scoring='r2',title=name,text_fontsize='large')
  plt.show()
```

:
```
models_result=pd.DataFrame({
    "NAME":NAME,
    "Cross_Val_Score":CVS,
    "R2_score":R2,
    "Mean_squared_error":MSE,
    "Mean_Absolute_Error":MAE,
    "RMSE":RMSE
})
models_result
```

:

	NAME	Cross_Val_Score	R2_score	Mean_squared_error	Mean_Absolute_Error	RMSE
0	XGB Regressor	0.650748	0.689063	0.078108	0.191203	0.279478
1	ExtraTrees Regressor	0.688044	0.683310	0.079553	0.185494	0.282052
2	RandomForest Regressor	0.710000	0.734377	0.066725	0.179369	0.258312
3	Linear Regression	0.446420	0.422638	0.145034	0.310727	0.380834
4	DecisionTree Regressor	0.592336	0.604512	0.099347	0.207018	0.315194
5	Lasso	-0.002568	-0.003943	0.252192	0.421478	0.502187
6	LIGHT GBM	0.711852	0.730077	0.067805	0.185964	0.260394

HYPERPARAMETER TUNING

```
params={
    'booster':['gbtree','dart'],
    'importance_type':['gain','split'],
    'max_depth':[3,4,6,5,7],
    'n_estimators':[100,200,500]
}
Grid=GridSearchCV(estimator=XGB,param_grid=params,cv=kf,n_jobs=-1,scoring='r2')
Grid.fit(X,np.log(y))
```

```
GridSearchCV(cv=KFold(n_splits=5, random_state=None, shuffle=True),
             estimator=XGBRegressor(base_score=0.5, booster='gbtree',
                                    colsample_bylevel=1, colsample_bynode=1,
                                    colsample_bytree=1, gamma=0, gpu_id=-1,
                                    importance_type='gain',
                                    interaction_constraints='',
                                    learning_rate=0.300000012, max_delta_step=0,
                                    max_depth=6, min_child_weight=1,
                                    missing=nan, monotone_constraints='()',
                                    n_estimators=100, n_jobs=4,
                                    num_parallel_tree=1, random_state=0,
                                    reg_alpha=0, reg_lambda=1,
                                    scale_pos_weight=1, subsample=1,
                                    tree_method='exact', validate_parameters=1,
                                    verbosity=None),
             n_jobs=-1,
             param_grid={'booster': ['gbtree', 'dart'],
                         'importance_type': ['gain', 'split'],
                         'max_depth': [3, 4, 6, 5, 7],
                         'n_estimators': [100, 200, 500]},
             scoring='r2')
```

```
Grid.best_params_
```

```
{'booster': 'gbtree',
 'importance_type': 'gain',
 'max_depth': 3,
 'n_estimators': 100}
```

```
Grid.best_score_
```

```
0.7159611434643521
```

```
XGBR=XGBRegressor(booster= 'dart',
 importance_type= 'gain',
 max_depth= 7,
 n_estimators= 200)
XGBR.fit(X_train,y_train)
```

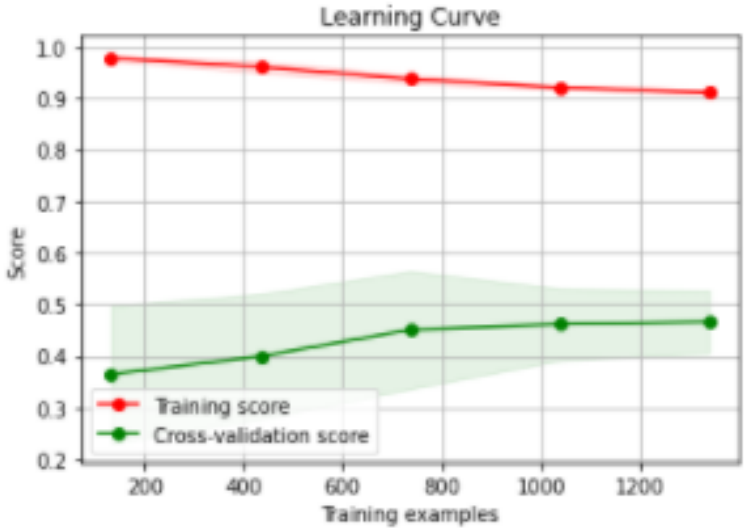
```
XGBRegressor(base_score=0.5, booster='dart', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.300000012, max_delta_step=0, max_depth=7,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=200, n_jobs=4, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='exact', validate_parameters=1, verbosity=None)
```

FINAL MODEL PERFORMANCE METRICS AND LEARNING CURVE

FINAL MODEL

```
print('MODEL PERFORMANCE CURVE')
skplt.estimators.plot_learning_curve(XGBR,X,y,cv=kf,scoring='r2')
plt.show()
```

```
MODEL PERFORMANCE CURVE
```



FINAL MODEL METRICS

```
print("MSE",mean_squared_error(y_test,y_pred))
print("MAE",mean_absolute_error(y_test,y_pred))
print("RMSE",np.sqrt(mean_squared_error(y_test,y_pred)))
print("R2_Score",r2_score(y_test,y_pred))
print("Model_Score",XGBR.score(X_test,y_test))
```

```
MSE 0.08195270738588471
MAE 0.19439573996960813
RMSE 0.2862738328696577
R2_Score 0.6737575586884916
Model_Score 0.6737575586884916
```

CONCLUSION

Learning Outcomes of the Study in respect of Data Science

The above research will help our client to study the latest flight price market and with the help of the model built he can easily predict the price ranges of the flight, and also will helps him to understand Based on what factors the fight price is decided.

THANKYOU