FLIGHT-PRICE-PROJECT

SUBMITTED BY: - TEJENDRA SONI

ACKNOWLEDGMENT

I would like to express my thanks of gratitude to SME SWATI MAHASETH as well as Flip Robo who gave me the golden opportunity to do this wonderful project on the topic FLIGHT PRICE PROJECT, which also helped me in doing a lot of research and I came to know about so many new things. I am very thankful to them.

INTRODUCTION

OBJECTIVE:

1. Data Collection You must scrape at least 1500 rows of data. You can scrape more data as well, it’s up to you, More the data better the model in this section you must scrape the data of flights from different websites (yatra.com, skyscanner.com, official websites of airlines, etc.). The number of columns for data doesn’t have limit, it’s up to you and your creativity. Generally, these columns are airline name, date of journey, source, destination, route, departure time, arrival time, duration, total stops and the target variable price. You can make changes to it, you can add, or you can remove some columns, it completely depends on the website from which you are fetching the data.

2. Data Analysis After cleaning the data, you must do some analysis on the data. Do airfares change frequently? Do they move in small increments or in large jumps? Do they tend to go up or down over time? What is the best time to buy so that the

consumer can save the most by taking the least risk? Does price increase as we get near to departure date? Is Indigo cheaper than Jet Airways? Are morning flights expensive?

3. Model Building After collecting the data, you need to build a machine learning model. Before model building do all data pre-processing steps. Try different models with different hyper parameters and select the best model. Follow the complete life cycle of data science. Include all the steps like 1. Data Cleaning 2. Exploratory Data Analysis 3. Data Pre-processing 4. Model Building 5. Model Evaluation 6. Selecting the best mode

Firstly, we will start by importing required libraries and databases.

In [1]:

```
#Importing the needed libraries

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

In [2]:

```
#Storing the data into the dataframe & printing it

df=pd.read_csv('FLIGHT-DATA-SCRAPING.csv')
df
```

Out[2]:

	Unnamed: 0	Airline	Departure_time	Time_of_arrival	Duration	Source	Destination	Meal_availability	Number_of_stops	Price
0	0	Go First	12:50	22:05	9h 15m	New Delhi	Mumbai	eCash 250	1 Stop	5954
1	1	Air India	18:00	20:00	2h 00m	New Delhi	Mumbai	Free Meal	Non Stop	5955
2	2	Air India	07:00	09:05	2h 05m	New Delhi	Mumbai	Free Meal	Non Stop	5955
3	3	Vistara	05:45	07:55	2h 10m	New Delhi	Mumbai	No Meal Fare	Non Stop	5955
4	4	IndiGo	06:30	08:40	2h 10m	New Delhi	Mumbai	Emissions: 142 Kg CO2	Non Stop	5955
...	...	...	...	...	...	...	...	...	...	...
3155	3155	Air India	17:30	09:00	15h 30m	Goa	New Delhi	Free Meal	2 Stop(s)	20910
3156	3156	Air India	21:00	10:15	13h 15m	Goa	New Delhi	Free Meal	1 Stop	22590
3157	3157	Air India	17:30	13:50	20h 20m	Goa	New Delhi	Free Meal	2 Stop(s)	23745
3158	3158	Air India	23:35	10:15	10h 40m	Goa	New Delhi	Free Meal	1 Stop	25162
3159	3159	Air India	17:30	18:30	25h 00m	Goa	New Delhi	Free Meal	2 Stop(s)	25320

3160 rows × 10 columns

Our dataset has 3160 rows and 10 columns.

In [3]:
```
#Printing the shape of the dataset it stats that there are 3160 rows & 10 columns

df.shape
```

Out[3]:
```
(3160, 10)
```

In [4]:
```
#Getting the information about the dataset

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3160 entries, 0 to 3159
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Unnamed: 0         3160 non-null   int64 
 1   Airline            3160 non-null   object
 2   Departure_time     3160 non-null   object
 3   Time_of_arrival    3160 non-null   object
 4   Duration           3160 non-null   object
 5   Source             3160 non-null   object
 6   Destination        3160 non-null   object
 7   Meal_availability  3160 non-null   object
 8   Number_of_stops    3160 non-null   object
 9   Price              3160 non-null   int64 
dtypes: int64(2), object(8)
memory usage: 247.0+ KB
```

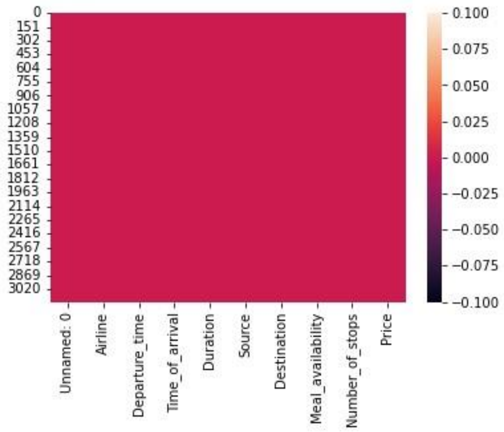
Printing the shape of the dataset & printing the information about the dataset Only unnamed: 0 & Price column are of integer type rest all the variables are of object type

In [5]:
```
#Checking the null values
#We can see that there are no null valuies in our dataset

df.isnull().sum()
```

Out[5]:
```
Unnamed: 0           0
Airline              0
Departure_time       0
Time_of_arrival      0
Duration             0
Source               0
Destination          0
Meal_availability    0
Number_of_stops      0
Price                0
dtype: int64
```

In [6]:
```
#Plotting the graph with the help of seaborn library
#As the graph is fully orange & there are no lines means our dataset is clean & there are no null values

sns.heatmap(df.isnull())
```

Out[6]:
```
<AxesSubplot:>
```



DROPPING THE COLUMNS WE DON'T NEED

In [7]:
```
#Dropping the Unnamed column as it does'nt contribute towards the analysis part

df=df.drop(['Unnamed: 0'],axis=1)
```

In [9]:
```
#Printing the dataset

df
```

Out[9]:

	Airline	Departure_time	Time_of_arrival	Duration	Source	Destination	Meal_availability	Number_of_stops	Price
0	Go First	12:50	22:05	9h 15m	New Delhi	Mumbai	eCash 250	1 Stop	5954
1	Air India	18:00	20:00	2h 00m	New Delhi	Mumbai	Free Meal	Non Stop	5955
2	Air India	07:00	09:05	2h 05m	New Delhi	Mumbai	Free Meal	Non Stop	5955
3	Vistara	05:45	07:55	2h 10m	New Delhi	Mumbai	No Meal Fare	Non Stop	5955
4	IndiGo	06:30	08:40	2h 10m	New Delhi	Mumbai	Emissions: 142 Kg CO2	Non Stop	5955
...	...	...	...	...	...	...	...	...	...
3155	Air India	17:30	09:00	15h 30m	Goa	New Delhi	Free Meal	2 Stop(s)	20910
3156	Air India	21:00	10:15	13h 15m	Goa	New Delhi	Free Meal	1 Stop	22590
3157	Air India	17:30	13:50	20h 20m	Goa	New Delhi	Free Meal	2 Stop(s)	23745
3158	Air India	23:35	10:15	10h 40m	Goa	New Delhi	Free Meal	1 Stop	25162
3159	Air India	17:30	18:30	25h 00m	Goa	New Delhi	Free Meal	2 Stop(s)	25320

3160 rows × 9 columns

ENCODING THE COLUMNS

In [10]:
```
#Converting the columns with LabelEncoder

from sklearn.preprocessing import LabelEncoder
categorical_features = list(df.select_dtypes(include=['object']).columns)
label_encoder_feat = {}
for i, feature in enumerate(categorical_features):
    label_encoder_feat[feature] = LabelEncoder()
    df[feature] = label_encoder_feat[feature].fit_transform(df[feature])
```

In [20]:
```
#Getting the description of the dataset

df.describe()
```

Out[20]:

	Airline	Departure_time	Time_of_arrival	Duration	Source	Destination	Meal_availability	Number_of_stops	Price
count	3160.000000	3160.000000	3160.000000	3160.000000	3160.000000	3160.000000	3160.000000	3160.000000	3160.000000
mean	3.118038	112.156329	138.378481	208.619304	2.689873	4.512975	9.836392	0.818987	14454.162975
std	1.721967	62.518502	67.756120	125.448712	1.889649	2.721275	1.371996	1.439556	6385.523113
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	3363.000000
25%	1.000000	53.000000	70.000000	94.000000	1.000000	2.000000	9.000000	0.000000	9850.500000
50%	3.000000	106.000000	150.000000	214.000000	3.000000	5.000000	10.000000	0.000000	13636.500000
75%	5.000000	169.000000	196.000000	339.000000	4.000000	7.000000	11.000000	1.000000	17759.750000
max	5.000000	235.000000	241.000000	389.000000	5.000000	8.000000	11.000000	4.000000	48597.000000

VISUALIZATION:

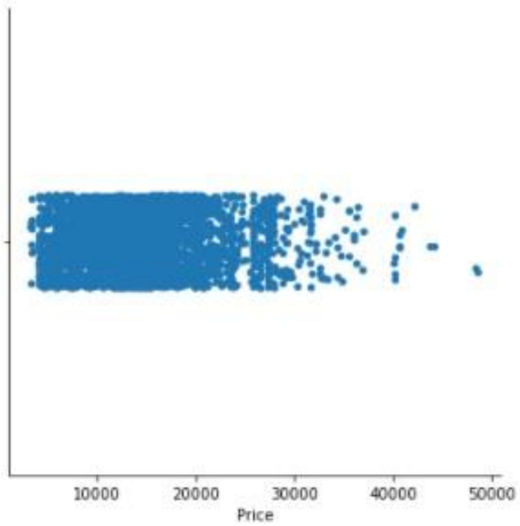
Plotting the cat plot & distribution plot of our target variable PRICE

In the graph 1 we can determine that the PRICE is more concentrated towards 0-25000 approx.

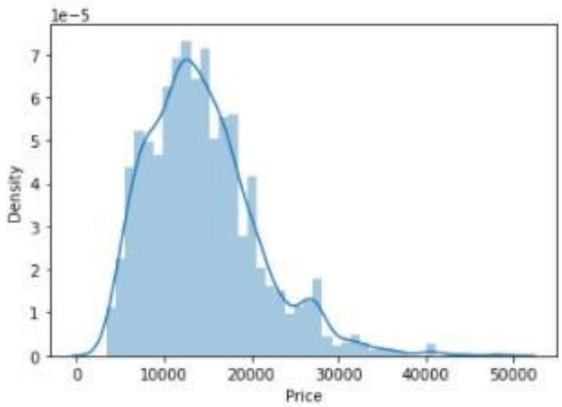
In the graph 2 we can determine that our target variable PRICE column

is almost normally distributed

```
In [12]: #Plotting catplot with our target column

         sns.catplot(data=df,x='Price')

Out[12]: <seaborn.axisgrid.FacetGrid at 0x274d6618340>
```




```
In [13]: #Plotting the distribution plot of the target variable

         sns.distplot(df['Price'])

Out[13]: <AxesSubplot:xlabel='Price', ylabel='Density'>
```



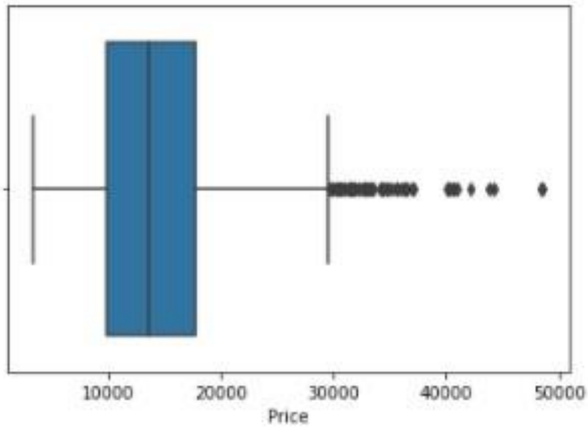
In the graph 1 we can determine that there are some outliers in our target variable PRICE

We will remove it afterwards

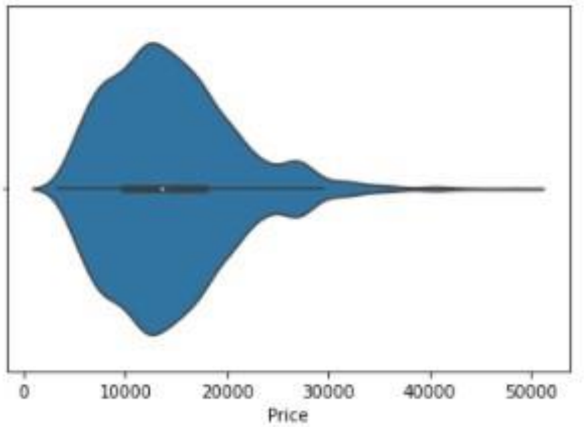
&

In the graph 2 we can see the distribution of PRICE through VIOLINPLOT

```
In [14]: #There are outliers present in the Price column

         sns.boxplot(data=df,x='Price')

Out[14]: <AxesSubplot:xlabel='Price'>
```



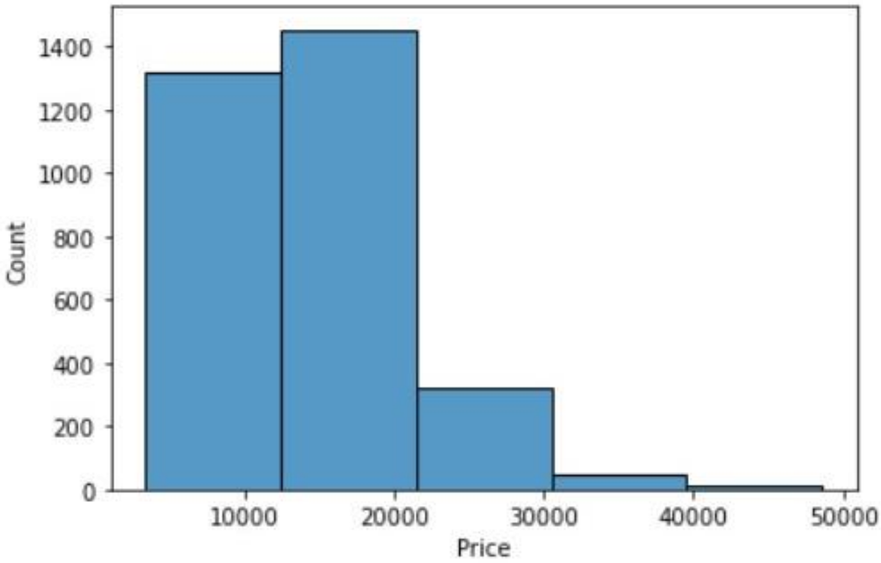

```
In [15]: #Plotting violin plot

         sns.violinplot(df['Price'])

Out[15]: <AxesSubplot:xlabel='Price'>
```



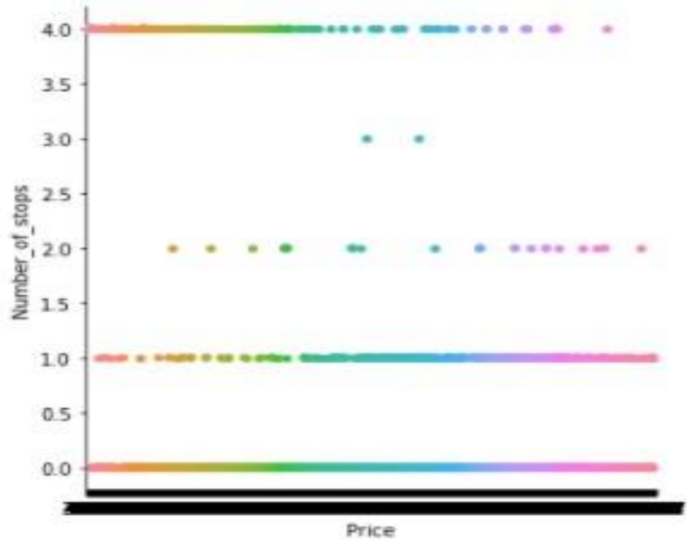
In the graph we are plotting the histogram plot of the PRICE & we can see that at ₹20000 most of the fares occurred i.e., 1400+

In [16]:
```
#Plotting histogram

sns.histplot(df['Price'],bins=5)
```

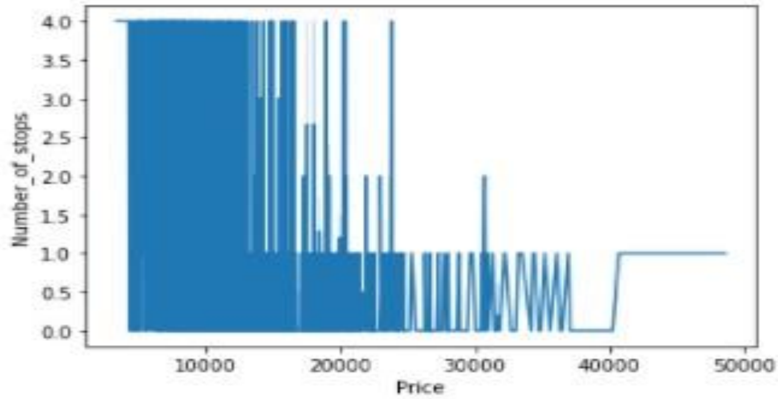
Out[16]: <AxesSubplot:xlabel='Price', ylabel='Count'>



In [22]:
```
#PLotting catpLot
sns.catplot(x='Price',y='Number_of_stops',data=df)
```
Out[22]: <seaborn.axisgrid.FacetGrid at 0x274d6ed4f10>



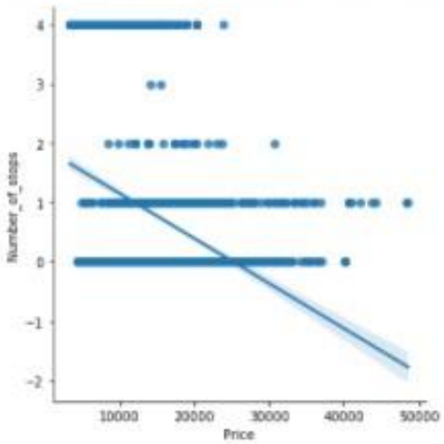
In [23]:
```
#PLoting Line pLot
sns.lineplot(x='Price',y='Number_of_stops',data=df)
```
Out[23]: <AxesSubplot:xlabel='Price', ylabel='Number_of_stops'>



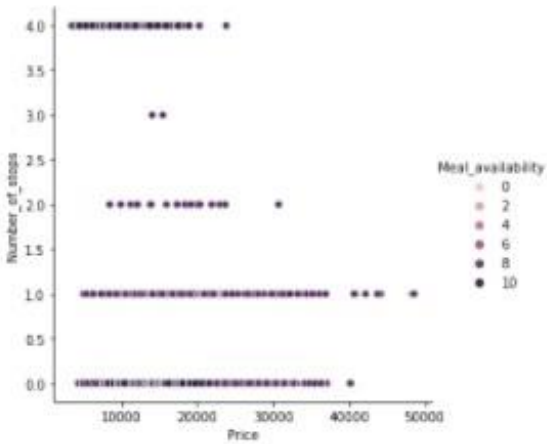
With this graph we can determine that as the price increases the number of stops of FLIGHTS decreases

As we can see when the graph goes from 15000 to 30000 we can clearly see that the number of stops has been decreased

```
In [24]: #Ploting lm plot
         sns.lmplot('Price','Number_of_stops',data=df)
Out[24]: <seaborn.axisgrid.FacetGrid at 0x274d6ce4070>
```




```
In [26]: #Plotting the relational plot of Price & Number of stops with respect to the meanl availability in the flights
         sns.relplot(x='Price',y='Number_of_stops',hue='Meal_availability',data=df)
Out[26]: <seaborn.axisgrid.FacetGrid at 0x274d849f880>
```

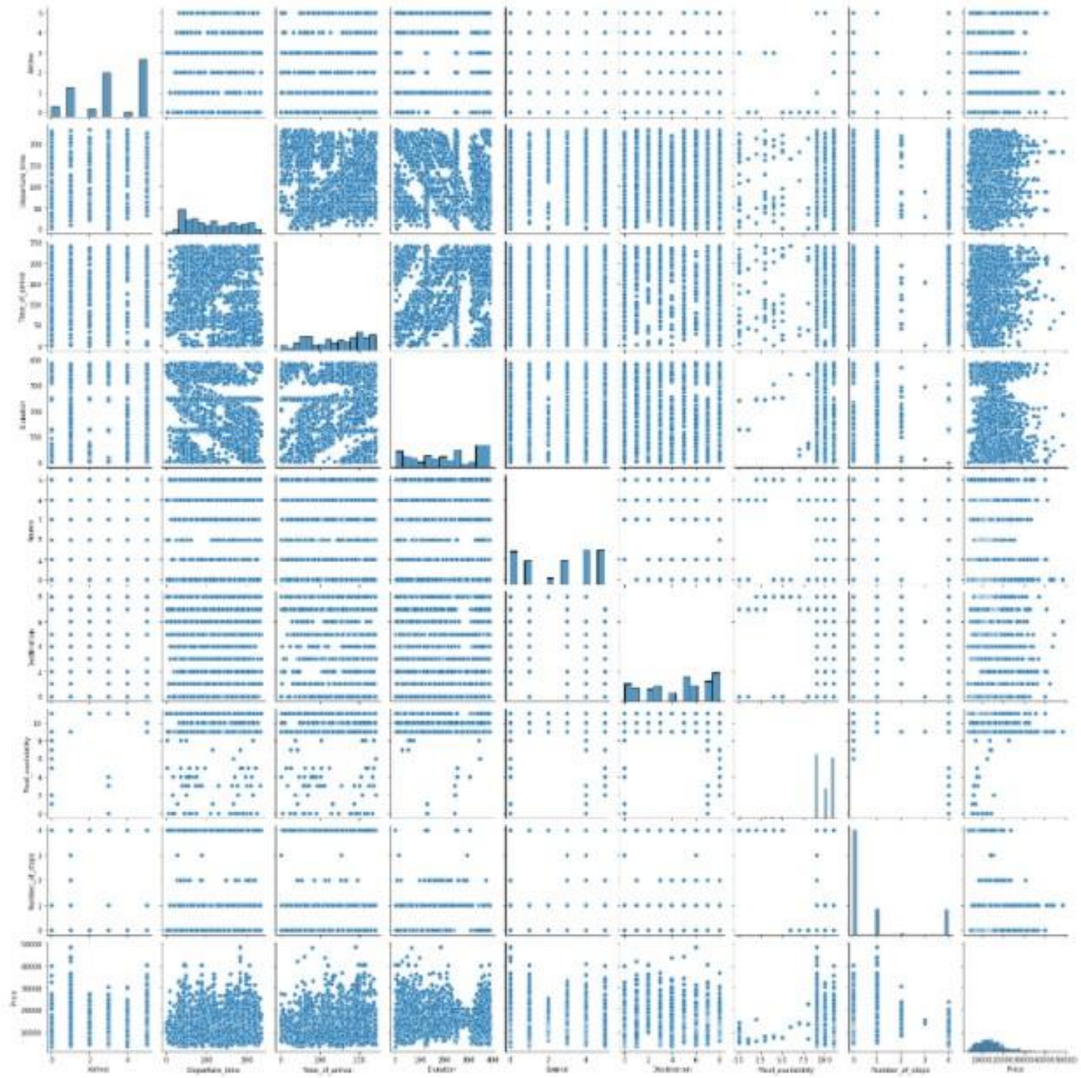


As we have seen in the above graphs

This graph also states that there is negative relationship b/w PRICE & NUMBER OF STOPS

Printing the relational plot of PRICE & NUMBER OF STOPS with respect to the MEAL AVAILABILITY in the FLIGHT

```
In [29]: #Plotting the pairplpot of the dataset
         #Printing the all the rows & columns with respect to each other

         sns.pairplot(df)

Out[29]: <seaborn.axisgrid.PairGrid at 0x274e0e66f10>
```



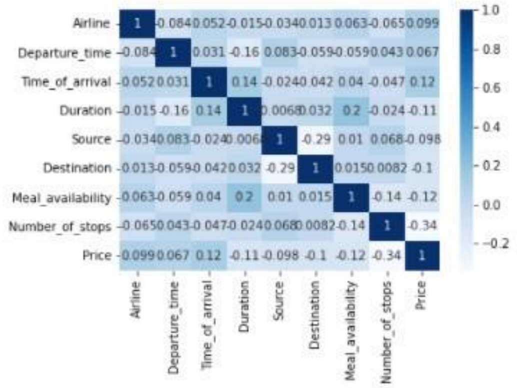
Plotting all the columns with respect to the rows

Plotting the pair plot of the whole dataset

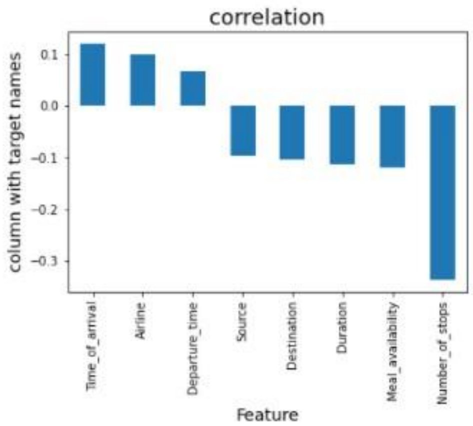
CORRELATION:

In [18]:
```
#Showing corelation
#Here the darker shades are highly corelated

sns.heatmap(df.corr(),annot=True,cmap='Blues')
```

Out[18]: <AxesSubplot:>



In [19]:
```
#Showing +vely & -vely corelation of the columns

df.corr()['Price'].sort_values(ascending=False).drop(['Price']).plot(kind='bar')
plt.xlabel('Feature',fontsize=14)
plt.ylabel('column with target names',fontsize=14)
plt.title('correlation',fontsize=18)
plt.show()
```



Plotting the graph of the correlation

Also, we can here determine that all the darker shade of columns are highly

Correlated

Time of arrival is most positively correlated

Then, Airline

Then, Departure time

& The remaining columns are negatively correlated

Number of stops is most negatively correlated

Then, Meal availability

Then, Duration

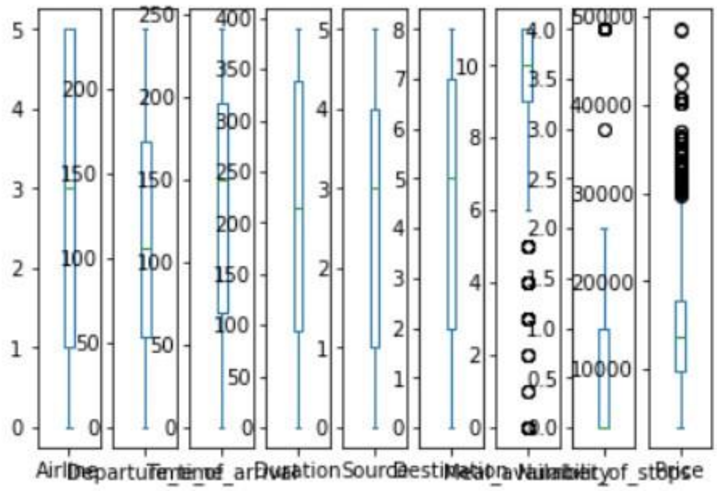
Then, Destination

Then, Source

OUTLIERS:

```
In [30]: #Showing boxplots of all the variables to determine outliers

         df.plot(kind='box',subplots=True)
```

```
Out[30]: Airline                 AxesSubplot(0.125,0.125;0.0731132x0.755)
         Departure_time       AxesSubplot(0.212736,0.125;0.0731132x0.755)
         Time_of_arrival      AxesSubplot(0.300472,0.125;0.0731132x0.755)
         Duration             AxesSubplot(0.388208,0.125;0.0731132x0.755)
         Source               AxesSubplot(0.475943,0.125;0.0731132x0.755)
         Destination          AxesSubplot(0.563679,0.125;0.0731132x0.755)
         Meal_availability    AxesSubplot(0.651415,0.125;0.0731132x0.755)
         Number_of_stops      AxesSubplot(0.739151,0.125;0.0731132x0.755)
         Price                AxesSubplot(0.826887,0.125;0.0731132x0.755)
         dtype: object
```



There are some outliers in all the columns

So, we will remove them & will make our dataset ready for analysis

In [36]:
```
#Removing outliers

from scipy.stats import zscore
z=np.abs(zscore(df))
threshold=3
np.where(z>3)
```

Out[36]:
```
(array([   4,    5,   11,   14,   20,   24,   41,   98,  148,  149,  150,
         154,  155,  156,  157,  164,  165,  269,  541,  650,  675,  707,
         708,  717,  724,  725,  742,  743,  799,  840,  844,  851,  852,
         853,  859,  861,  864,  928,  929, 1042, 1178, 1179, 1180, 1181,
        1182, 1414, 1418, 1420, 1423, 1424, 1429, 1430, 1432, 1433, 1435,
        1535, 1539, 1544, 1548, 1552, 1578, 1579, 1585, 1603, 1612, 1613,
        1627, 1701, 1702, 1778, 1779, 1825, 1938, 1998, 2083, 2084, 2267,
        2268, 2269, 2388, 2389, 2390, 2391, 2392, 2393, 2394, 2395],
       dtype=int64),
 array([6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 8, 8, 8, 8, 6,
        6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 8, 8, 8, 8, 8,
        8, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 8, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
        6, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8],
       dtype=int64))
```

In [37]:
```
#Storing the dataframe after removing outlier

df_new=df[(z<3).all(axis=1)]
df_new
```

Out[37]:

	Airline	Departure_time	Time_of_arrival	Duration	Source	Destination	Meal_availability	Number_of_stops	Price
0	2	111	219	381	5	7	11	0	5954
1	1	171	194	243	5	7	9	4	5955
2	1	45	63	244	5	7	9	4	5955
3	5	30	49	245	5	7	10	4	5955
6	1	57	76	245	5	7	9	4	5955
...	...	...	...	...	...	...	...	...	...
3155	1	165	62	67	2	8	9	1	20910
3156	1	206	77	40	2	8	9	0	22590
3157	1	165	120	137	2	8	9	1	23745
3158	1	233	77	9	2	8	9	0	25162
3159	1	165	176	191	2	8	9	1	25320

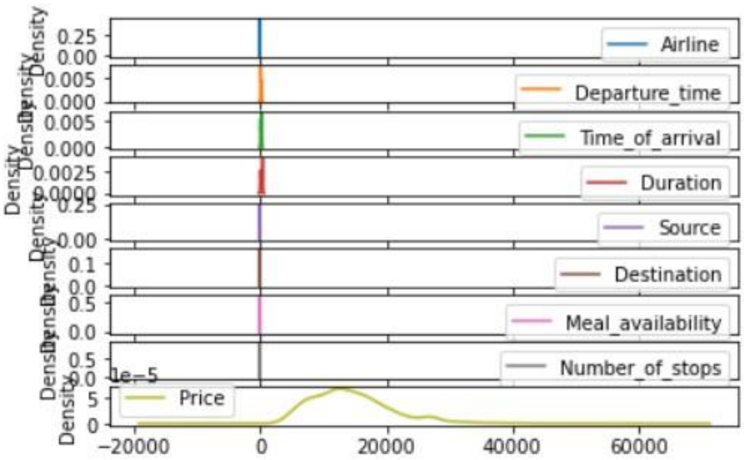
3073 rows × 9 columns

Removing the outliers & storing the filtered data into df_new

SKEWNESS:

In [31]:
```python
#Determining the skewness present in all the columns

df.plot(kind='kde',subplots=True)
```

Out[31]:
```
array([<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
       <AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
       <AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
       <AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
       <AxesSubplot:ylabel='Density'>], dtype=object)
```



There is no high skewness is present in our

Dataset

Still, we will check on it individually & will try to

remove the skewness till the maximum extent

that we can

In [42]:
```
#Removnig skewness with the help of power transformation

from sklearn.preprocessing import power_transform
df_new=power_transform(x)

df_new=pd.DataFrame(df_new,columns=x.columns)
```

In [43]:
```
#Here we can see skewness has been removed from the data

df_new.skew()
```

Out[43]:
```
Airline              -0.278453
Departure_time       -0.110872
Time_of_arrival      -0.321332
Duration             -0.379470
Source               -0.310849
Destination          -0.313039
Meal_availability    -0.031782
Number_of_stops       0.833876
dtype: float64
```

Using power transformation technique, we are trying to remove skewness

```
In [49]: #Using for loop determining the best accuracy for the model at the best random state

maxAccu=0
maxRS=0
for i in range(1,200):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.10,random_state=i)
    DTC=DecisionTreeClassifier()
    DTC.fit(x_train,y_train)
    pred=DTC.predict(x_test)
    acc=accuracy_score(y_test,pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Best accuracy is ",maxAccu, " on Random State ",maxRS)
```

```
Best accuracy is  0.4253246753246753  on Random State  145
```

```
In [50]: #Sending the data for training & testing phase

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=maxRS)
```

```
In [51]: #Created a list in which we have stored all the instances of the model
#Using for loop we will determine the accuracy of all the models

model=[DecisionTreeClassifier(),SVC(),AdaBoostClassifier(),RandomForestClassifier(),LogisticRegression()]

for m in model:
    m.fit(x_train,y_train)
    #m.score(x_train,y_train)
    pred=m.predict(x_test)
    acc=accuracy_score(y_test,pred)
    print('Accuracy Score of',m,'is:', acc)
    print(confusion_matrix(y_test,pred))
    print(classification_report(y_test,pred))
    print('\n')
```

```
Accuracy Score of DecisionTreeClassifier() is: 0.36097560975609755
[[0 0 0 ... 0 0 0]
 [0 0 1 ... 0 0 0]
 [0 0 1 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
              precision    recall  f1-score   support

        3363       0.00      0.00      0.00         1
        3497       0.00      0.00      0.00         1
        3498       0.50      1.00      0.67         1
        3499       0.00      0.00      0.00         0
        4202       0.00      0.00      0.00         1
        4263       1.00      0.50      0.67         2
        4275       1.00      1.00      1.00         2
        4338       0.00      0.00      0.00         1
        4339       0.00      0.00      0.00         1
```

MODELS	ACCURACY
DTC	36.09
SVC	01.46
ABC	00.48
RFC	36.58
LOR	03.73

WE CAN SEE RFC HAS 36.58% ACCURACY

In [52]:
```python
#Again using for loop we will now determine the cross validation of each model

model=[DecisionTreeClassifier(),SVC(),AdaBoostClassifier(),RandomForestClassifier(),LogisticRegression()]
for m in model:
    score=cross_val_score(m,x,y,cv=5)
    print("Score for ",m," is : ",score.mean())
```

```
Score for  DecisionTreeClassifier()  is :  0.3312708879531792
Score for  SVC()  is :  0.01464156139932735
Score for  AdaBoostClassifier()  is :  0.007810704165673579
Score for  RandomForestClassifier()  is :  0.35827970657556735
Score for  LogisticRegression()  is :  0.02733508116840126
```

THE LEAST DIFFERENCE THAT WE CAN SEE IS IN RFC MODEL SO WE WILL SELECT RFC MODEL AS OUR BEST & NOW WILL HYPERPARAMETER TUNNING

We are cross validating each model & then we will select the best model for our dataset

Again, we are selecting the RFC as our best model

In [53]:
```python
#Importing the grid search cv for hypertune the model

from sklearn.model_selection import GridSearchCV
```

In [54]:
```python
#Passing the parameters for the model

parameters = {'max_depth':np.arange(2,10),
              'criterion':['gini','entropy'],
              'max_features':['auto','sqrt','log2'],
              'min_samples_split':[2,3,4]}
```

In [55]:
```python
#Creating the instance of the model

GCV=GridSearchCV(RandomForestClassifier(),parameters,cv=5)
```

In [56]:
```python
#Fetting the model

GCV.fit(x_train,y_train)
```

Out[56]:
```
GridSearchCV(cv=5, estimator=RandomForestClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': array([2, 3, 4, 5, 6, 7, 8, 9]),
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_split': [2, 3, 4]})
```

In [57]:
```python
#Getting the best parameters

GCV.best_params_
```

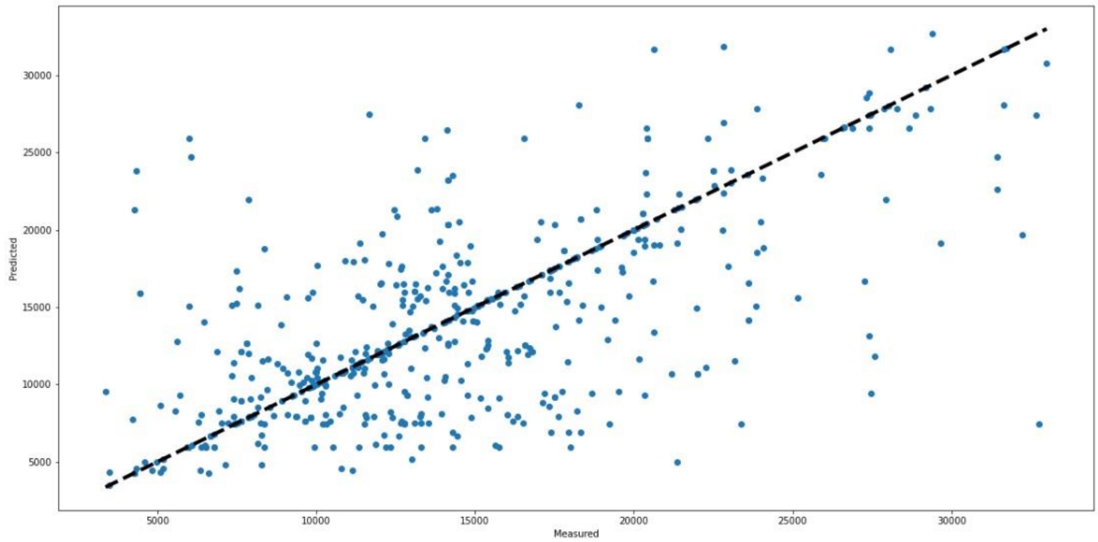
Out[57]:
```
{'criterion': 'entropy',
 'max_depth': 9,
 'max_features': 'log2',
 'min_samples_split': 3}
```

In [58]:
```python
#Passing the best parameters & printing the final accuracy

Final_mod= RandomForestClassifier(criterion="entropy",max_depth=9,max_features="log2",min_samples_split=3)
Final_mod.fit(x_train,y_train)
pred=Final_mod.predict(x_test)
acc=accuracy_score(y_test,pred)
print(acc*100)
```

```
34.959349593495936
```

Hyper parameter tunning our best model

```python
In [59]: #Ploting a diagram of the predicted & measured results

fig_dims = (20, 10)
fig, ax = plt.subplots(figsize=fig_dims)
ax.scatter(y_test, pred)
ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=4)
ax.set_xlabel('Measured')
ax.set_ylabel('Predicted')
plt.show()
```



The line shows the predicted & measured Results

SAVING & CONCLUSION:

```python
In [60]: #Importing pickle for saving the model
#Saving it in the pickle file

import pickle
filename= 'FLIGHT-PRICE-FOR-FLIPROBO.pkl'
pickle.dump(Final_mod, open(filename, 'wb'))
```

```python
In [61]: #load the model from the disk

loaded_model = pickle.load(open('FLIGHT-PRICE-FOR-FLIPROBO.pkl', 'rb'))
result = loaded_model.score(x_test,y_test)
print(result)
```

```
0.34959349593495936
```

```python
In [62]: #Printing the dataframe of the predicted & measured

conclusion=pd.DataFrame([loaded_model.predict(x_test)[:],pred[:]],index=["Predicted","Orginal"])
conclusion
```

Out[62]:

	0	1	2	3	4	5	6	7	8	9	...	605	606	607	608	609	610	611	612	613	614
Predicted	9277	21948	16677	17905	16571	5955	12213	26568	20694	7518	...	5943	22618	16657	13499	11973	19961	12544	7424	5943	21285
Orginal	9277	21948	16677	17905	16571	5955	12213	26568	20694	7518	...	5943	22618	16657	13499	11973	19961	12544	7424	5943	21285

2 rows × 615 columns