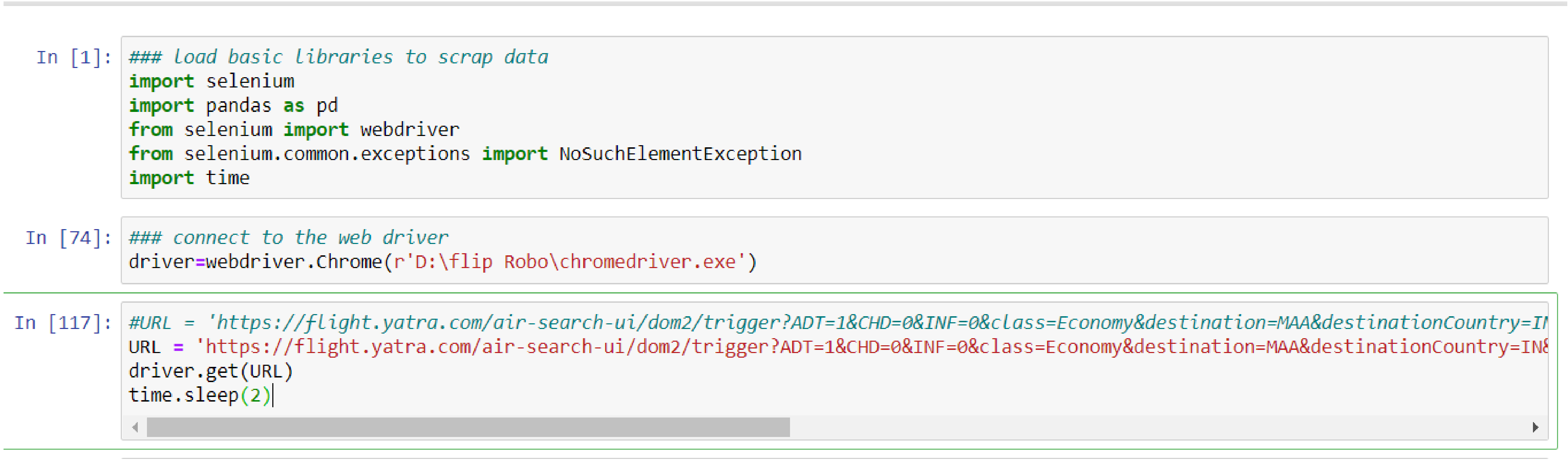
Flight Price

Optimal timing for airline ticket purchasing from the consumer’s perspective is challenging principally because buyers have insufficient information for reasoning about future price movements. In this project we simulate various models for computing expected future prices and classifying whether this is the best time to buy the ticket.

Pipeline Follows

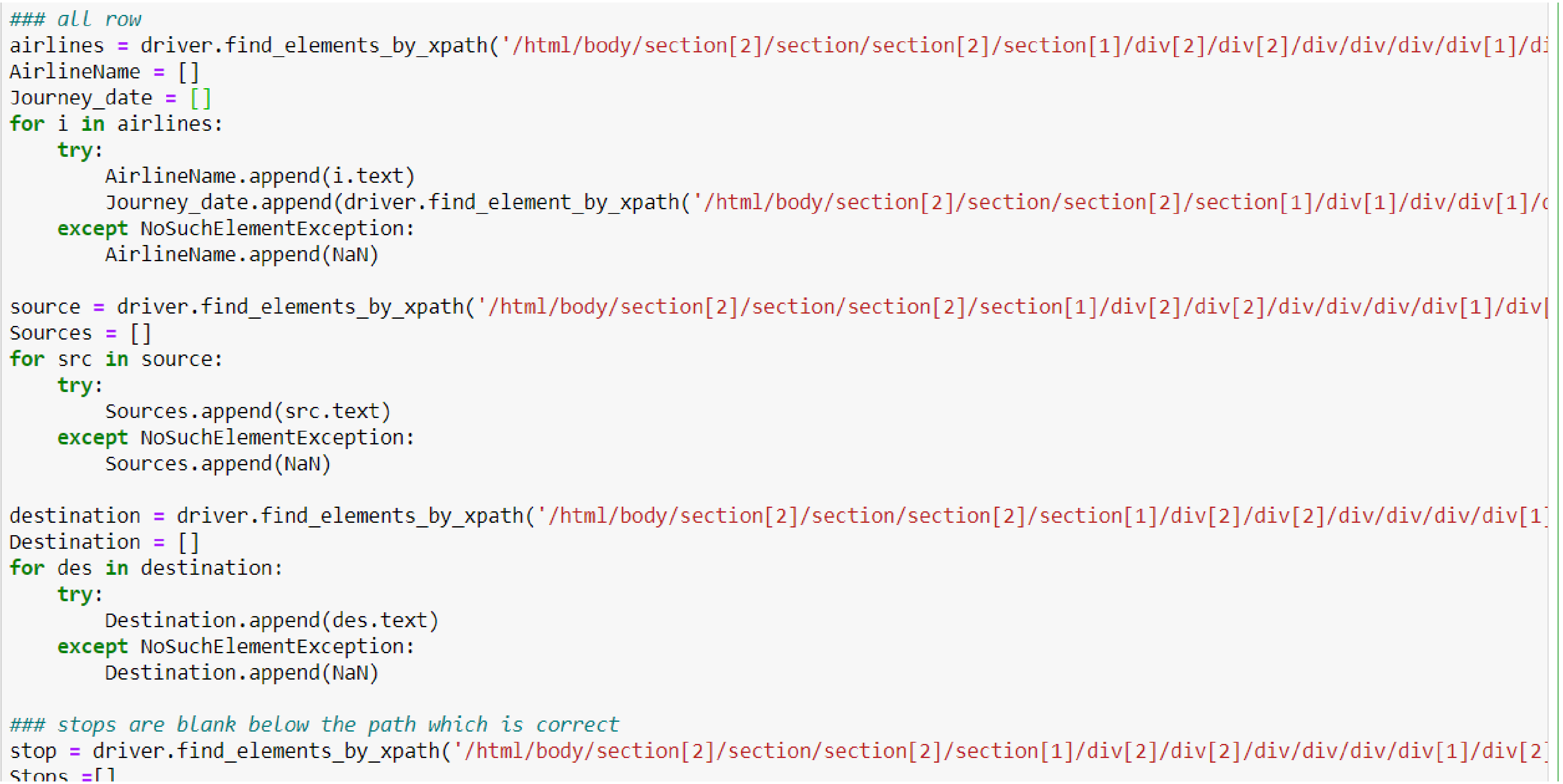
- Scrapping.
- Load Dataset
- EDA
- Visualization
- Data Cleaning
- Scaling
- Model Building
- Save

Scrap the data using Selenium



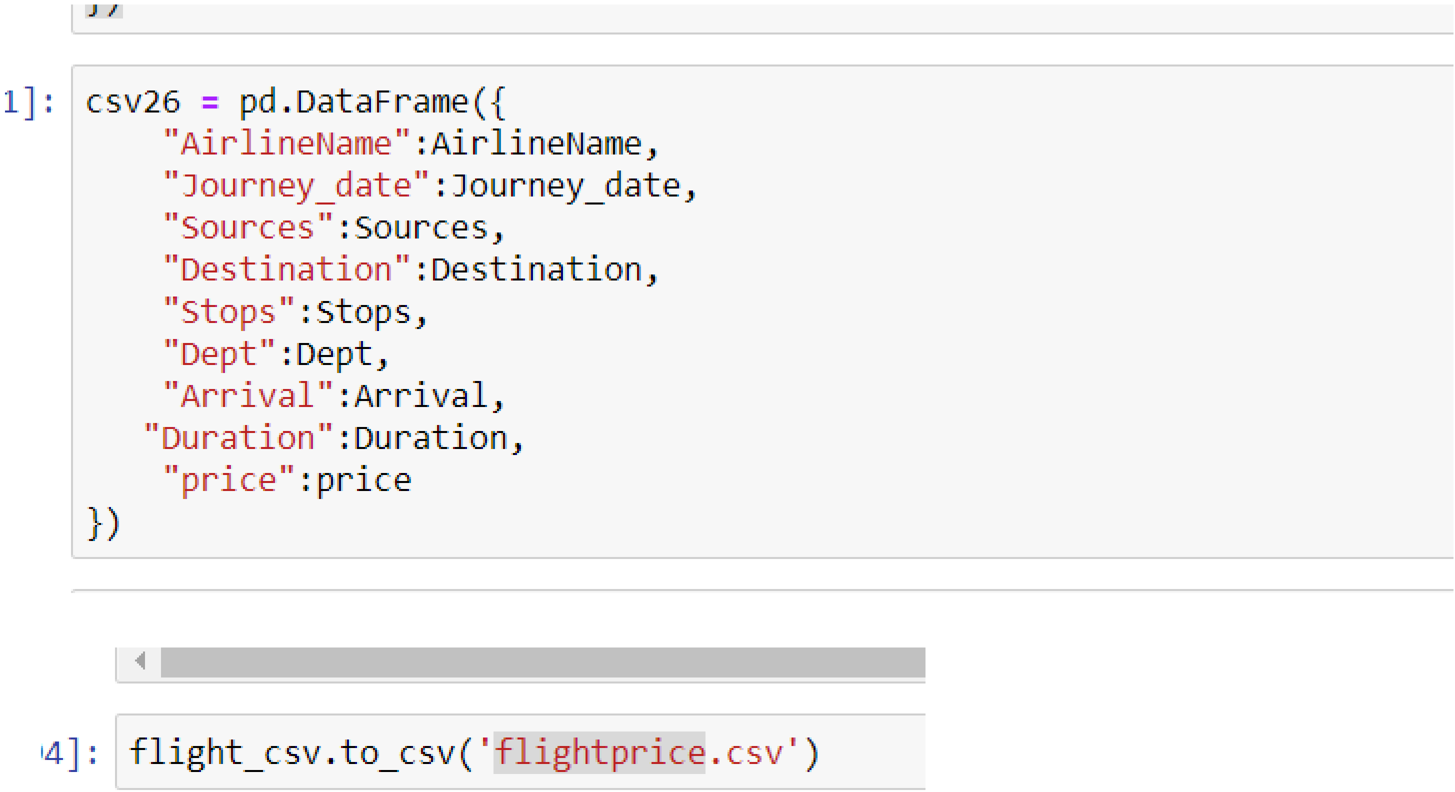
```
In [1]:  ### load basic libraries to scrap data
         import selenium
         import pandas as pd
         from selenium import webdriver
         from selenium.common.exceptions import NoSuchElementException
         import time

In [74]: ### connect to the web driver
         driver=webdriver.Chrome(r'D:\flip Robo\chromedriver.exe')

In [117]: #URL = 'https://flight.yatra.com/air-search-ui/dom2/trigger?ADT=1&CHD=0&INF=0&class=Economy&destination=MAA&destinationCountry=I
          URL = 'https://flight.yatra.com/air-search-ui/dom2/trigger?ADT=1&CHD=0&INF=0&class=Economy&destination=MAA&destinationCountry=IN
          driver.get(URL)
          time.sleep(2)
```

Scrapping And Storing in Variables/Array



```
### all row
airlines = driver.find_elements_by_xpath('/html/body/section[2]/section/section[2]/section[1]/div[2]/div[2]/div/div/div/div[1]/di
AirlineName = []
Journey_date = []
for i in airlines:
    try:
        AirlineName.append(i.text)
        Journey_date.append(driver.find_element_by_xpath('/html/body/section[2]/section/section[2]/section[1]/div[1]/div/div[1]/d
    except NoSuchElementException:
        AirlineName.append(NaN)

source = driver.find_elements_by_xpath('/html/body/section[2]/section/section[2]/section[1]/div[2]/div[2]/div/div/div/div[1]/div[
Sources = []
for src in source:
    try:
        Sources.append(src.text)
    except NoSuchElementException:
        Sources.append(NaN)

destination = driver.find_elements_by_xpath('/html/body/section[2]/section/section[2]/section[1]/div[2]/div[2]/div/div/div/div[1]
Destination = []
for des in destination:
    try:
        Destination.append(des.text)
    except NoSuchElementException:
        Destination.append(NaN)

### stops are blank below the path which is correct
stop = driver.find_elements_by_xpath('/html/body/section[2]/section/section[2]/section[1]/div[2]/div[2]/div/div/div/div[1]/div[2]
Stops =[]
```

Save Into Csv format

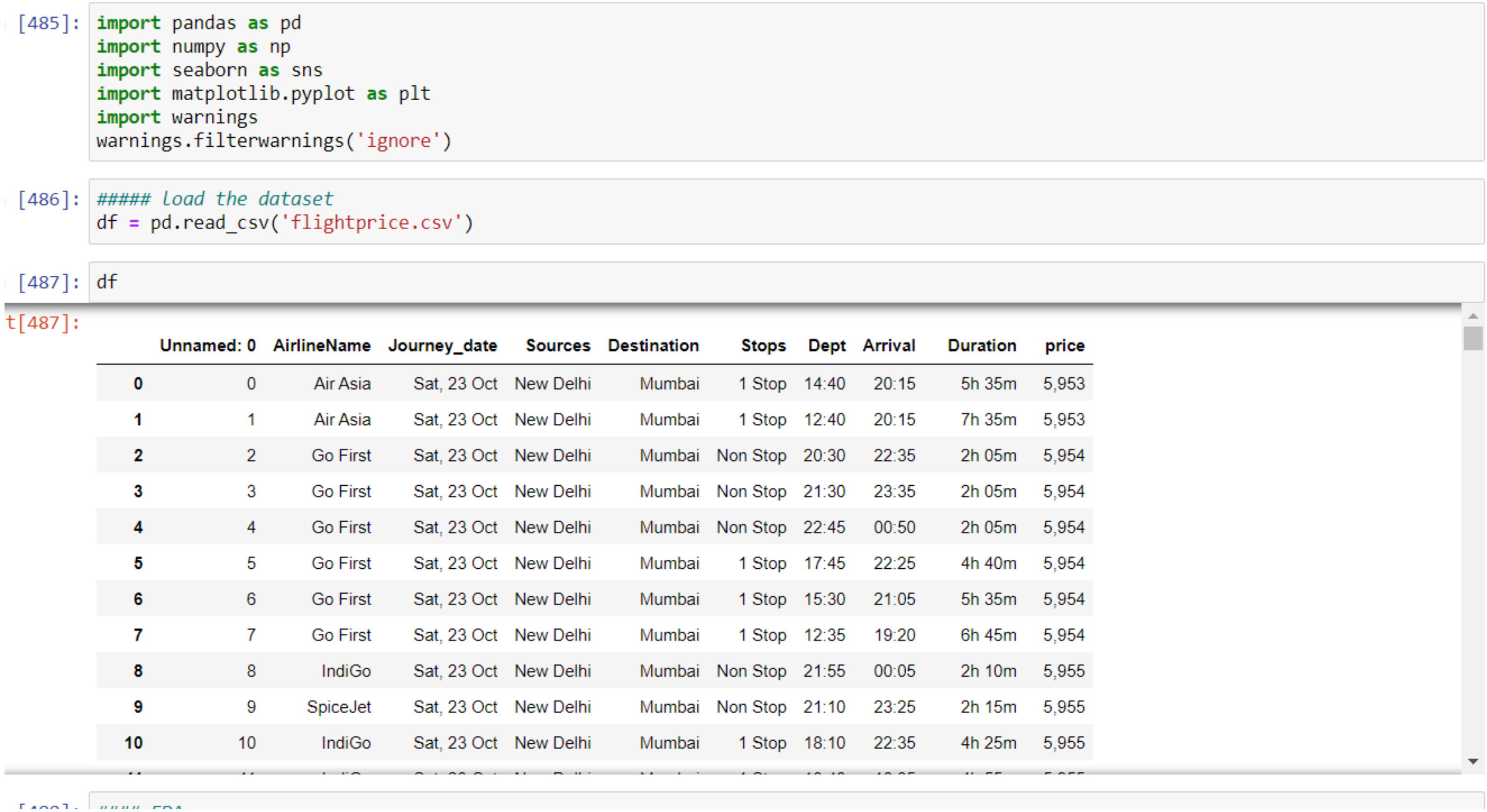


```
})
```

```
1]: csv26 = pd.DataFrame({
        "AirlineName":AirlineName,
        "Journey_date":Journey_date,
        "Sources":Sources,
        "Destination":Destination,
        "Stops":Stops,
        "Dept":Dept,
        "Arrival":Arrival,
       "Duration":Duration,
        "price":price
    })
```

```
4]: flight_csv.to_csv('flightprice.csv')
```

Flight Prediction – Load Dataset



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

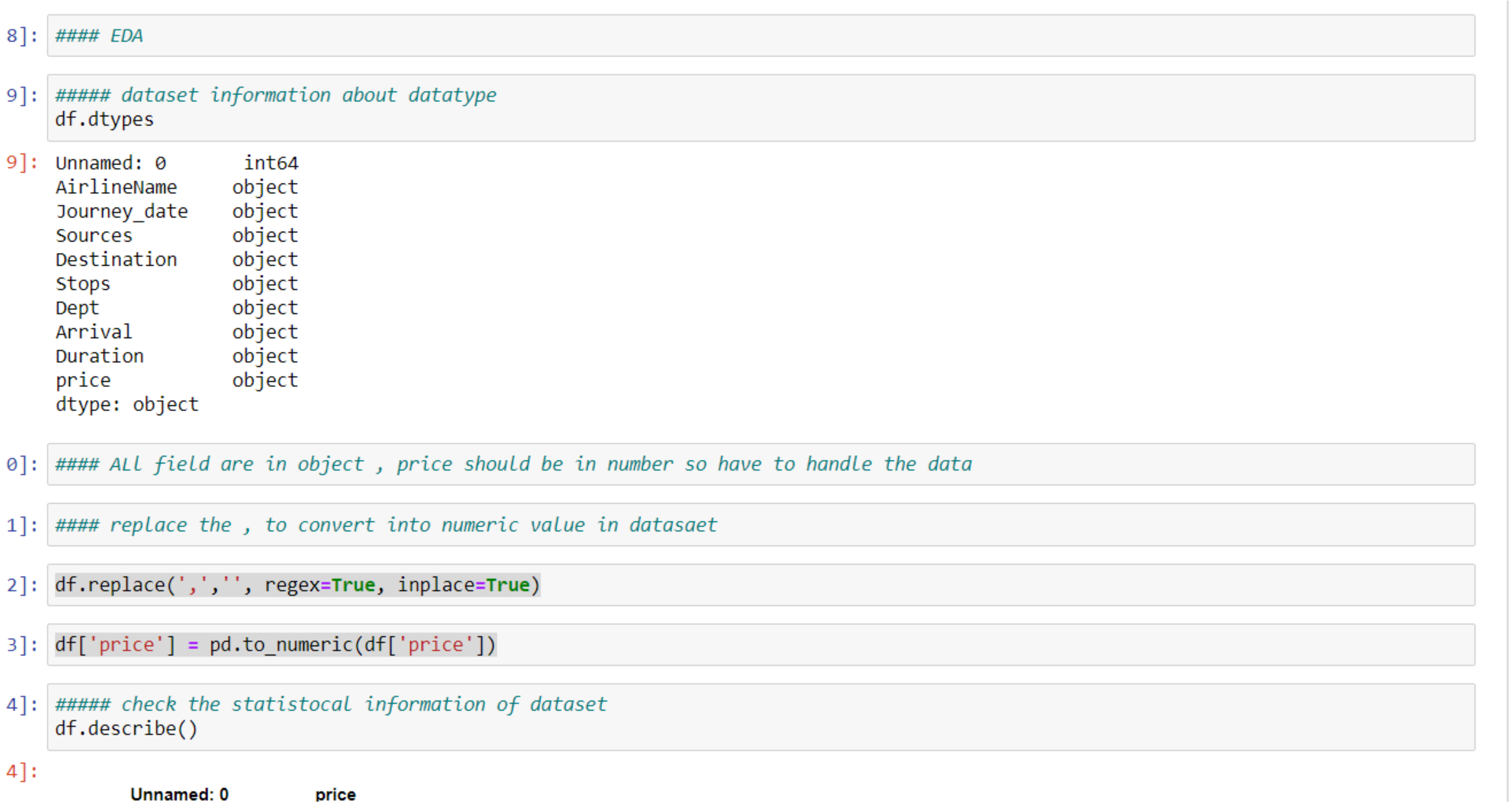
```
[486]: ##### load the dataset
       df = pd.read_csv('flightprice.csv')
```

```
[487]: df
```

t[487]:

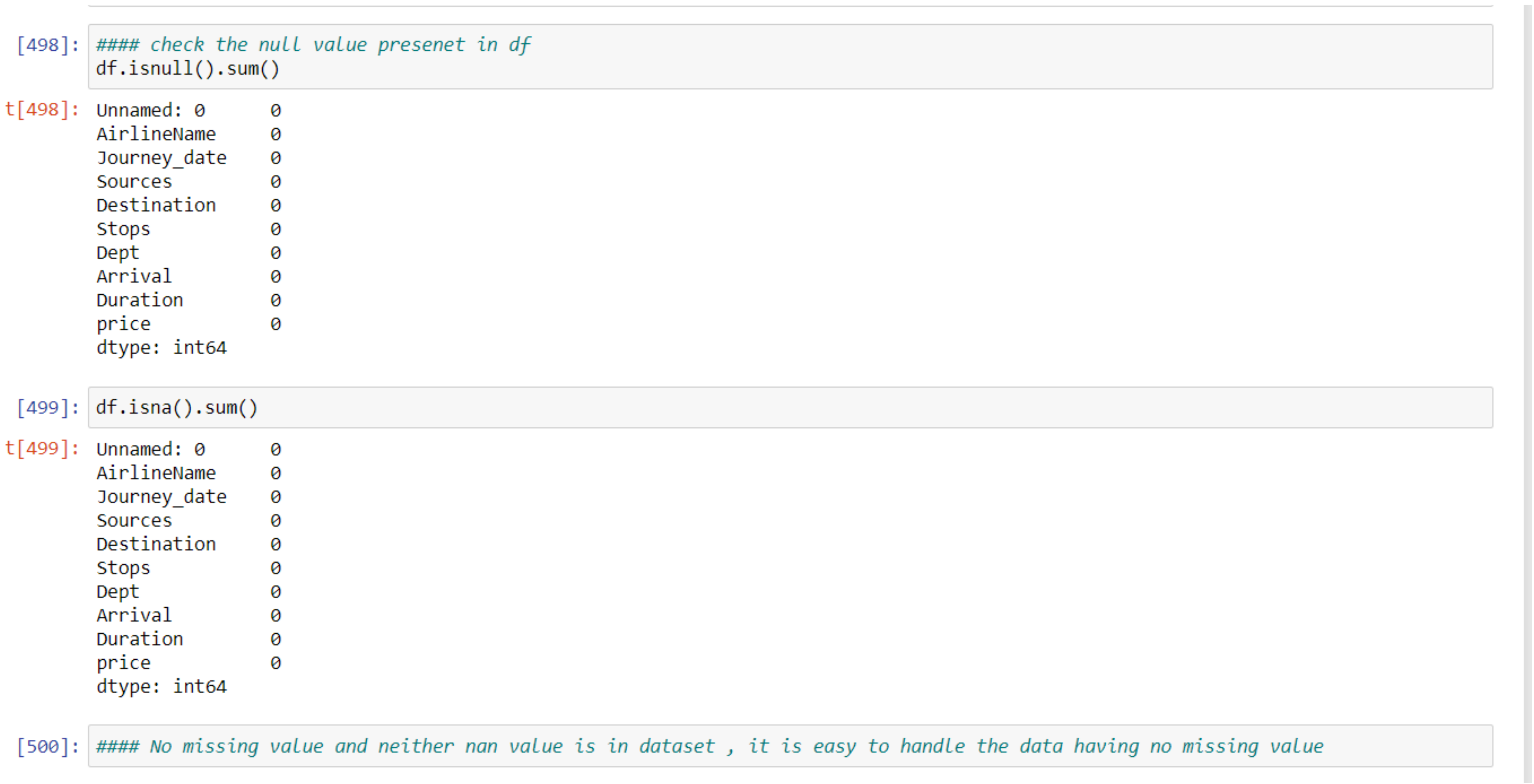
	Unnamed: 0	AirlineName	Journey_date	Sources	Destination	Stops	Dept	Arrival	Duration	price
0	0	Air Asia	Sat, 23 Oct	New Delhi	Mumbai	1 Stop	14:40	20:15	5h 35m	5,953
1	1	Air Asia	Sat, 23 Oct	New Delhi	Mumbai	1 Stop	12:40	20:15	7h 35m	5,953
2	2	Go First	Sat, 23 Oct	New Delhi	Mumbai	Non Stop	20:30	22:35	2h 05m	5,954
3	3	Go First	Sat, 23 Oct	New Delhi	Mumbai	Non Stop	21:30	23:35	2h 05m	5,954
4	4	Go First	Sat, 23 Oct	New Delhi	Mumbai	Non Stop	22:45	00:50	2h 05m	5,954
5	5	Go First	Sat, 23 Oct	New Delhi	Mumbai	1 Stop	17:45	22:25	4h 40m	5,954
6	6	Go First	Sat, 23 Oct	New Delhi	Mumbai	1 Stop	15:30	21:05	5h 35m	5,954
7	7	Go First	Sat, 23 Oct	New Delhi	Mumbai	1 Stop	12:35	19:20	6h 45m	5,954
8	8	IndiGo	Sat, 23 Oct	New Delhi	Mumbai	Non Stop	21:55	00:05	2h 10m	5,955
9	9	SpiceJet	Sat, 23 Oct	New Delhi	Mumbai	Non Stop	21:10	23:25	2h 15m	5,955
10	10	IndiGo	Sat, 23 Oct	New Delhi	Mumbai	1 Stop	18:10	22:35	4h 25m	5,955

EDA – It is technique to know what is in my data how the data is behaving as per domain.
Through EDA I know the shape size and description of data set.



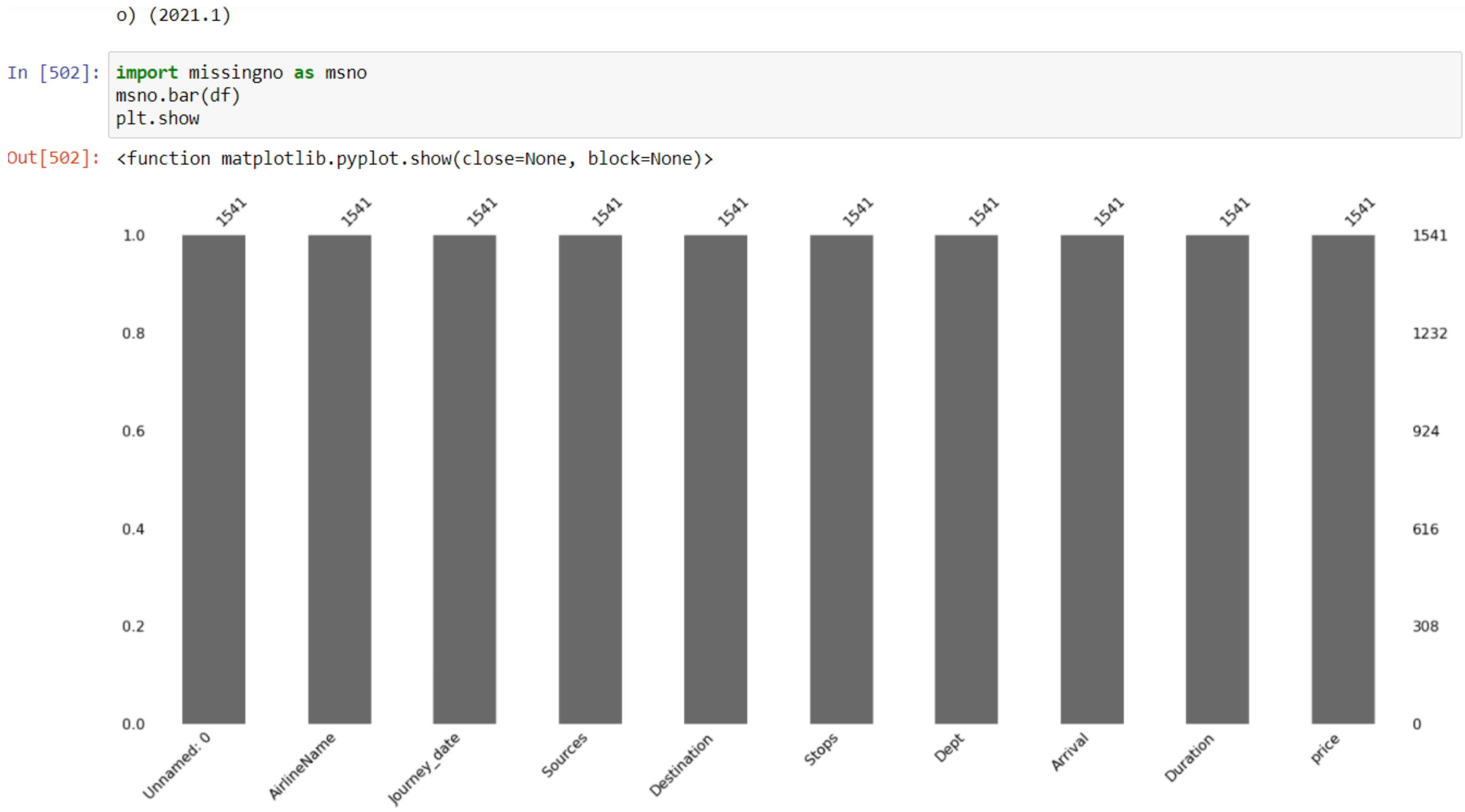
```
8]:  #### EDA

9]:  ##### dataset information about datatype
     df.dtypes

9]:  Unnamed: 0        int64
     AirlineName      object
     Journey_date     object
     Sources          object
     Destination      object
     Stops            object
     Dept             object
     Arrival          object
     Duration         object
     price            object
     dtype: object

0]:  #### ALL field are in object , price should be in number so have to handle the data

1]:  #### replace the , to convert into numeric value in datasaet

2]:  df.replace(',','', regex=True, inplace=True)

3]:  df['price'] = pd.to_numeric(df['price'])

4]:  ##### check the statistocal information of dataset
     df.describe()

4]:
          Unnamed: 0          price
```

Check for Null Value



```
[498]: #### check the null value presenet in df
       df.isnull().sum()

t[498]: Unnamed: 0      0
        AirlineName     0
        Journey_date    0
        Sources         0
        Destination     0
        Stops           0
        Dept            0
        Arrival         0
        Duration        0
        price           0
        dtype: int64

[499]: df.isna().sum()

t[499]: Unnamed: 0      0
        AirlineName     0
        Journey_date    0
        Sources         0
        Destination     0
        Stops           0
        Dept            0
        Arrival         0
        Duration        0
        price           0
        dtype: int64

[500]: #### No missing value and neither nan value is in dataset , it is easy to handle the data having no missing value
```

Visualization – It is process of showing data in chart format which make easy to know the data information in graphical representation.

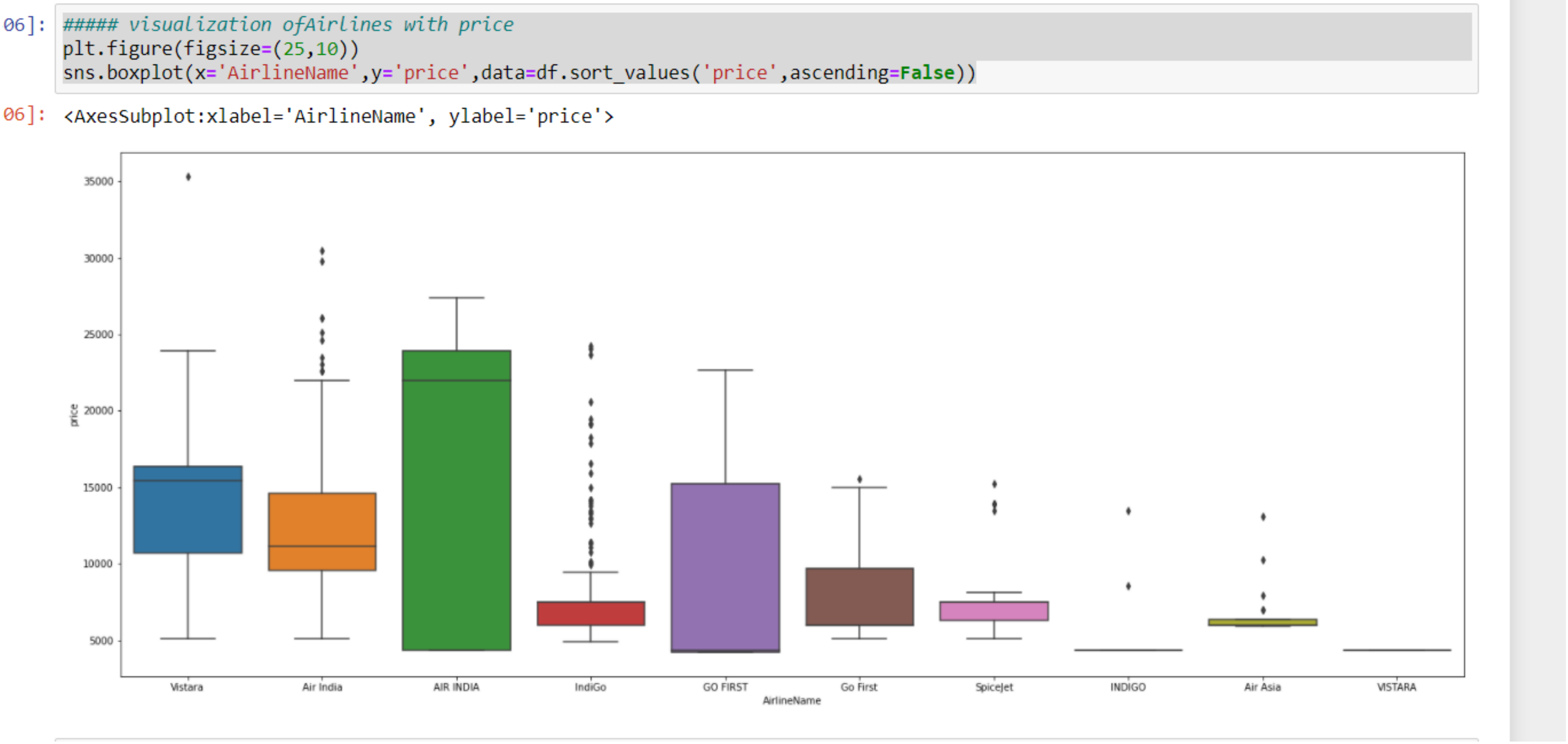


```
o) (2021.1)
```

```
In [502]: import missingno as msno
          msno.bar(df)
          plt.show
```

```
Out[502]: <function matplotlib.pyplot.show(close=None, block=None)>
```

Price Vary with Airlines



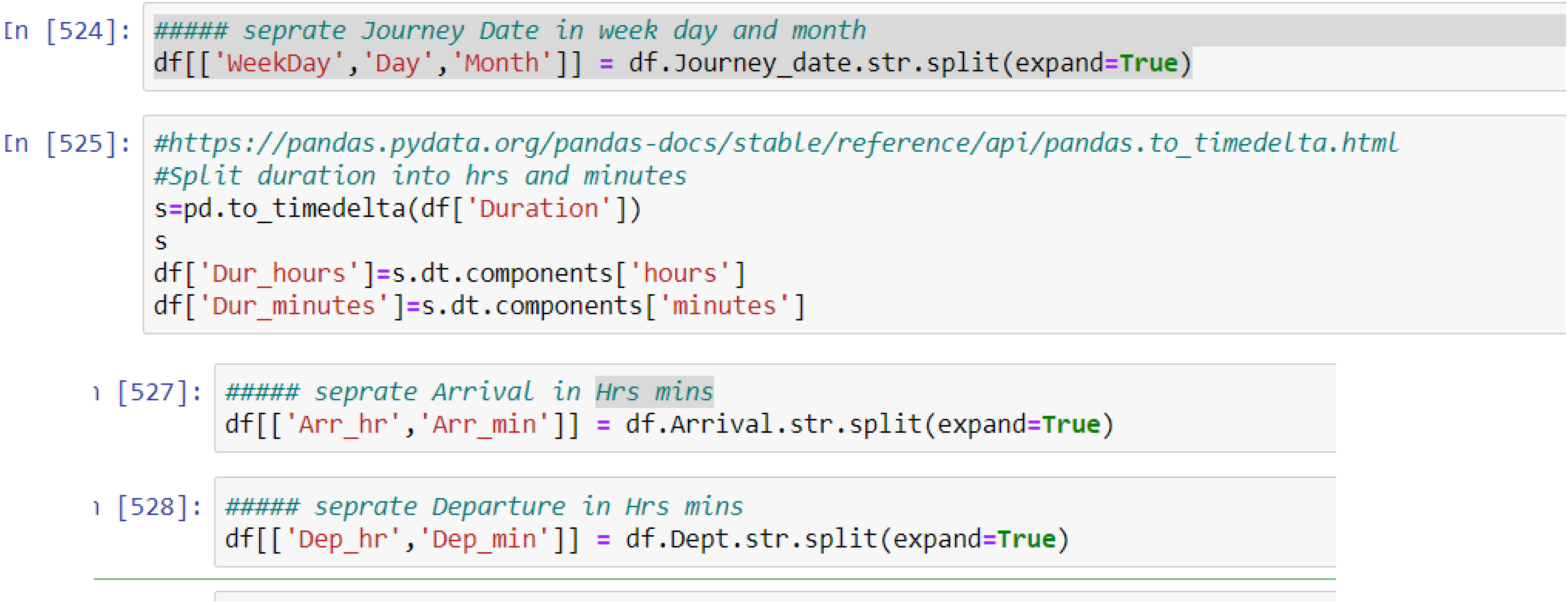
```
06]: ##### visualization ofAirlines with price
     plt.figure(figsize=(25,10))
     sns.boxplot(x='AirlineName',y='price',data=df.sort_values('price',ascending=False))
```

```
06]: <AxesSubplot:xlabel='AirlineName', ylabel='price'>
```

Pre-Processing and Data Cleaning

- It is the most important part of ML.
- Data cleaning decide the Accuracy of Model.
- Most of the time spent to cleaning of data.
- I have used various techniques and it comes when we can see and judge each field and its use.

Here I have split the Journey Date into Day week and Month and same for min and Hrs



In [524]:

```
##### seprate Journey Date in week day and month
df[['WeekDay','Day','Month']] = df.Journey_date.str.split(expand=True)
```

In [525]:

```
#https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.to_timedelta.html
#Split duration into hrs and minutes
s=pd.to_timedelta(df['Duration'])
s
df['Dur_hours']=s.dt.components['hours']
df['Dur_minutes']=s.dt.components['minutes']
```

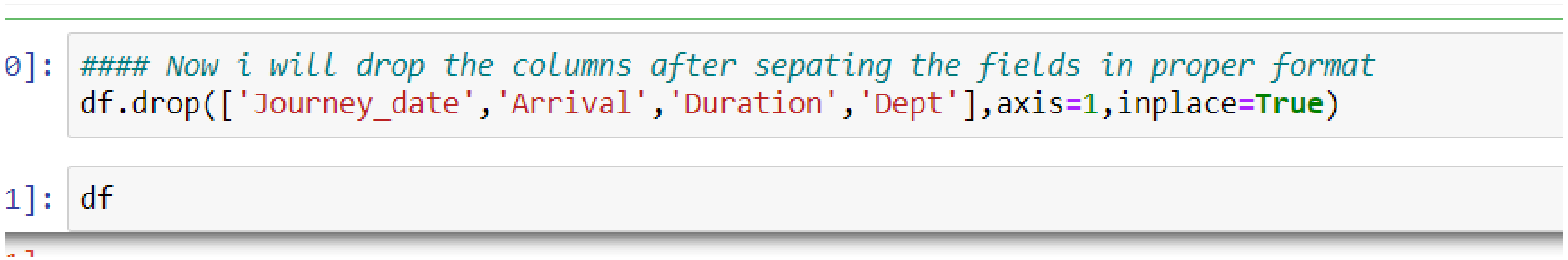
n [527]:

```
##### seprate Arrival in Hrs mins
df[['Arr_hr','Arr_min']] = df.Arrival.str.split(expand=True)
```

n [528]:

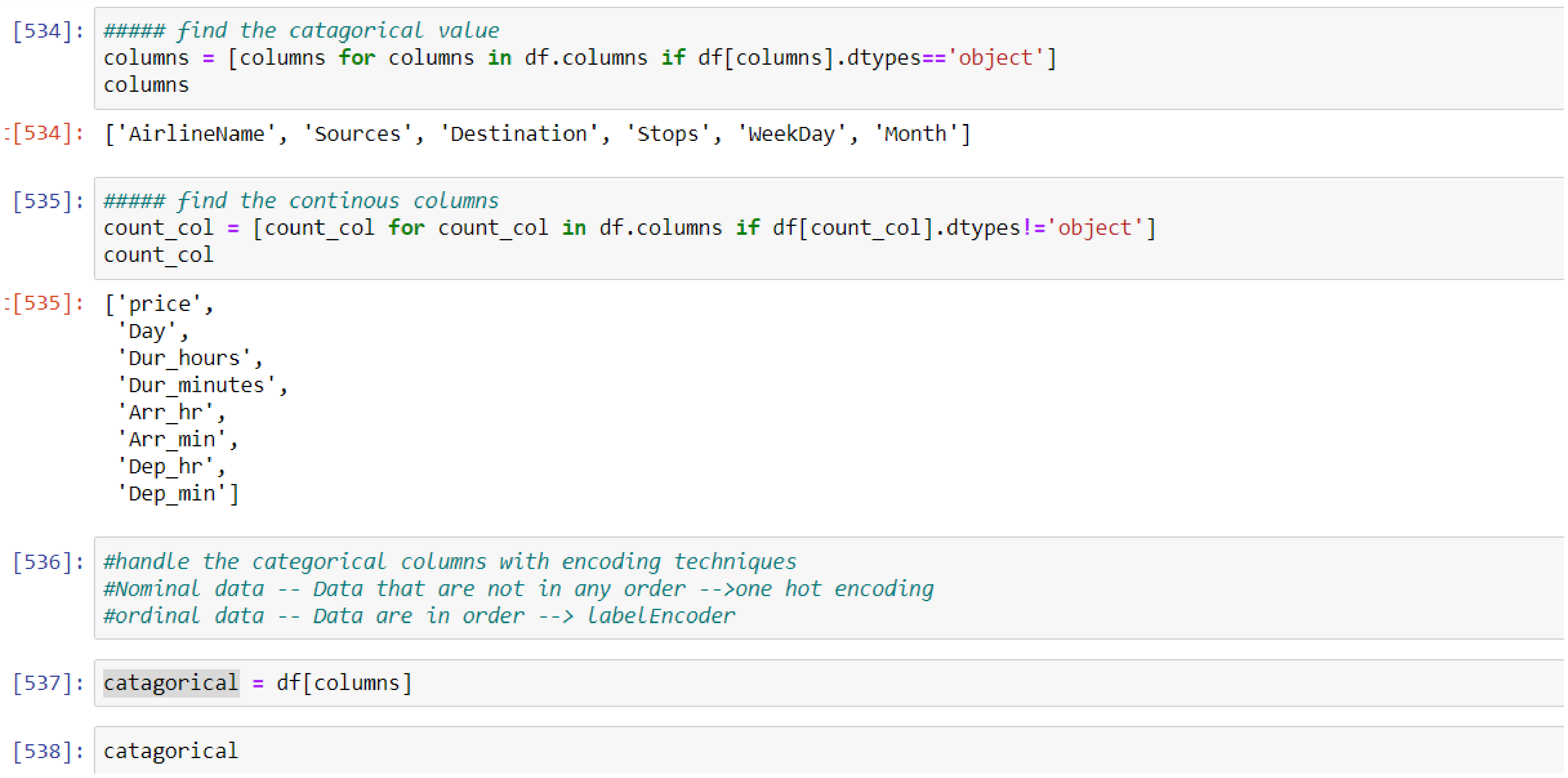
```
##### seprate Departure in Hrs mins
df[['Dep_hr','Dep_min']] = df.Dept.str.split(expand=True)
```

Drop the columns after splitting of Data



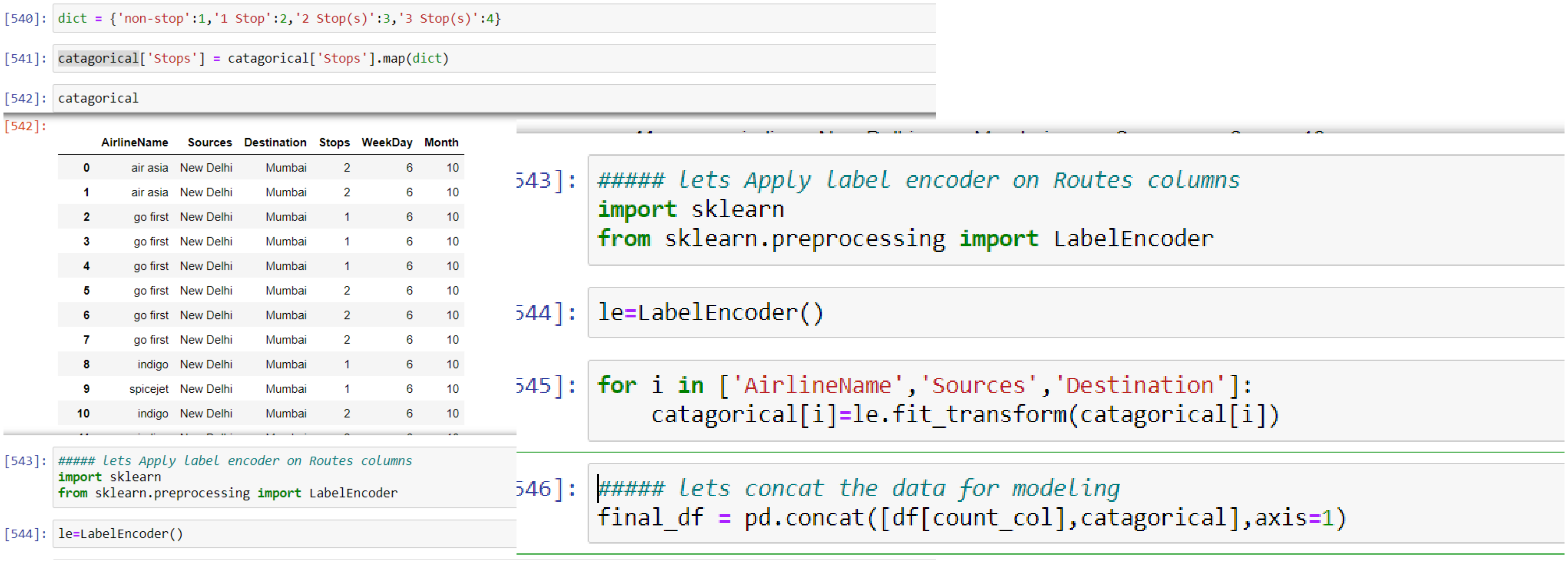
```
0]: #### Now i will drop the columns after sepating the fields in proper format
    df.drop(['Journey_date','Arrival','Duration','Dept'],axis=1,inplace=True)

1]: df
```

Separate Categorical And Continous Columns



```
[534]: ##### find the catagorical value
       columns = [columns for columns in df.columns if df[columns].dtypes=='object']
       columns

:[534]: ['AirlineName', 'Sources', 'Destination', 'Stops', 'WeekDay', 'Month']

[535]: ##### find the continous columns
       count_col = [count_col for count_col in df.columns if df[count_col].dtypes!='object']
       count_col

:[535]: ['price',
         'Day',
         'Dur_hours',
         'Dur_minutes',
         'Arr_hr',
         'Arr_min',
         'Dep_hr',
         'Dep_min']

[536]: #handle the categorical columns with encoding techniques
       #Nominal data -- Data that are not in any order -->one hot encoding
       #ordinal data -- Data are in order --> labelEncoder

[537]: catagorical = df[columns]

[538]: catagorical
```

Encoding of columns



```
[540]: dict = {'non-stop':1,'1 Stop':2,'2 Stop(s)':3,'3 Stop(s)':4}
```

```
[541]: catagorical['Stops'] = catagorical['Stops'].map(dict)
```

```
[542]: catagorical
```

[542]:

	AirlineName	Sources	Destination	Stops	WeekDay	Month
0	air asia	New Delhi	Mumbai	2	6	10
1	air asia	New Delhi	Mumbai	2	6	10
2	go first	New Delhi	Mumbai	1	6	10
3	go first	New Delhi	Mumbai	1	6	10
4	go first	New Delhi	Mumbai	1	6	10
5	go first	New Delhi	Mumbai	2	6	10
6	go first	New Delhi	Mumbai	2	6	10
7	go first	New Delhi	Mumbai	2	6	10
8	indigo	New Delhi	Mumbai	1	6	10
9	spicejet	New Delhi	Mumbai	1	6	10
10	indigo	New Delhi	Mumbai	2	6	10

```
[543]: ##### lets Apply label encoder on Routes columns
       import sklearn
       from sklearn.preprocessing import LabelEncoder
```

```
[544]: le=LabelEncoder()
```

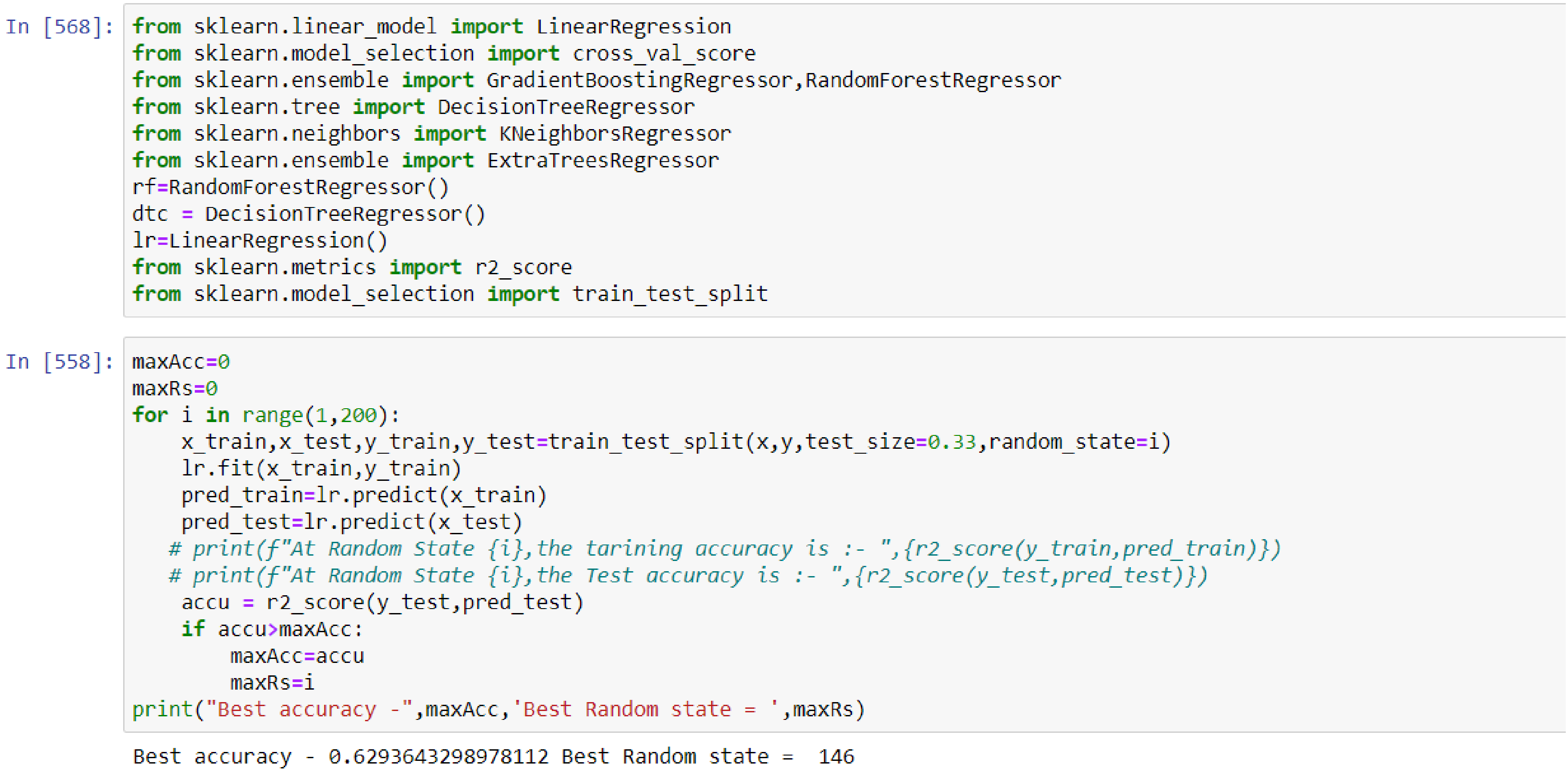
```
543]: ##### lets Apply label encoder on Routes columns
      import sklearn
      from sklearn.preprocessing import LabelEncoder
```

```
544]: le=LabelEncoder()
```

```
545]: for i in ['AirlineName','Sources','Destination']:
          catagorical[i]=le.fit_transform(catagorical[i])
```

```
546]: ##### lets concat the data for modeling
      final_df = pd.concat([df[count_col],catagorical],axis=1)
```

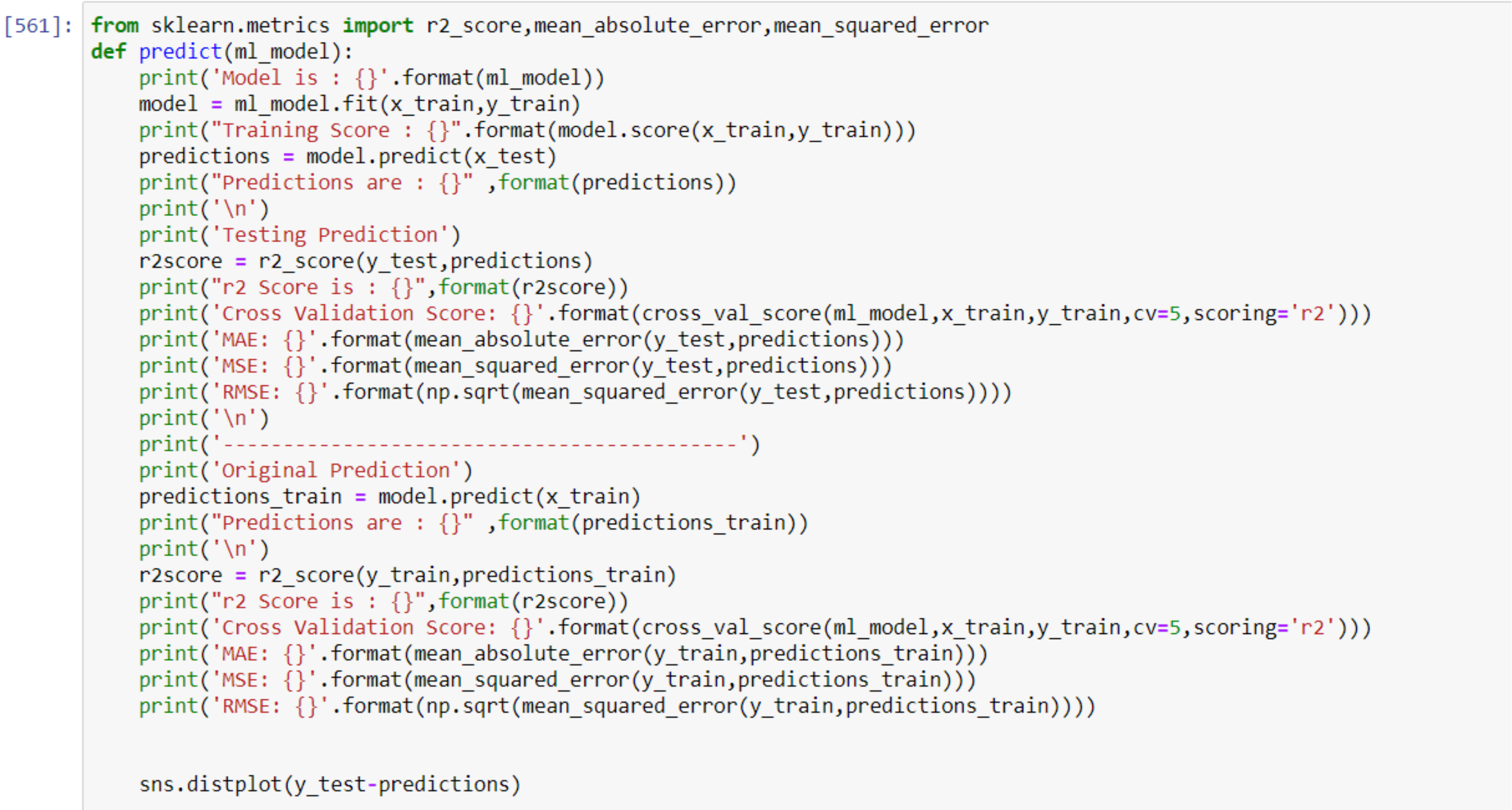
Model Building and Find the Best Random State



```
In [568]: from sklearn.linear_model import LinearRegression
          from sklearn.model_selection import cross_val_score
          from sklearn.ensemble import GradientBoostingRegressor,RandomForestRegressor
          from sklearn.tree import DecisionTreeRegressor
          from sklearn.neighbors import KNeighborsRegressor
          from sklearn.ensemble import ExtraTreesRegressor
          rf=RandomForestRegressor()
          dtc = DecisionTreeRegressor()
          lr=LinearRegression()
          from sklearn.metrics import r2_score
          from sklearn.model_selection import train_test_split
```

```
In [558]: maxAcc=0
          maxRs=0
          for i in range(1,200):
              x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=i)
              lr.fit(x_train,y_train)
              pred_train=lr.predict(x_train)
              pred_test=lr.predict(x_test)
             # print(f"At Random State {i},the tarining accuracy is :- ",{r2_score(y_train,pred_train)})
             # print(f"At Random State {i},the Test accuracy is :- ",{r2_score(y_test,pred_test)})
              accu = r2_score(y_test,pred_test)
              if accu>maxAcc:
                  maxAcc=accu
                  maxRs=i
          print("Best accuracy -",maxAcc,'Best Random state = ',maxRs)

          Best accuracy - 0.6293643298978112 Best Random state =  146
```

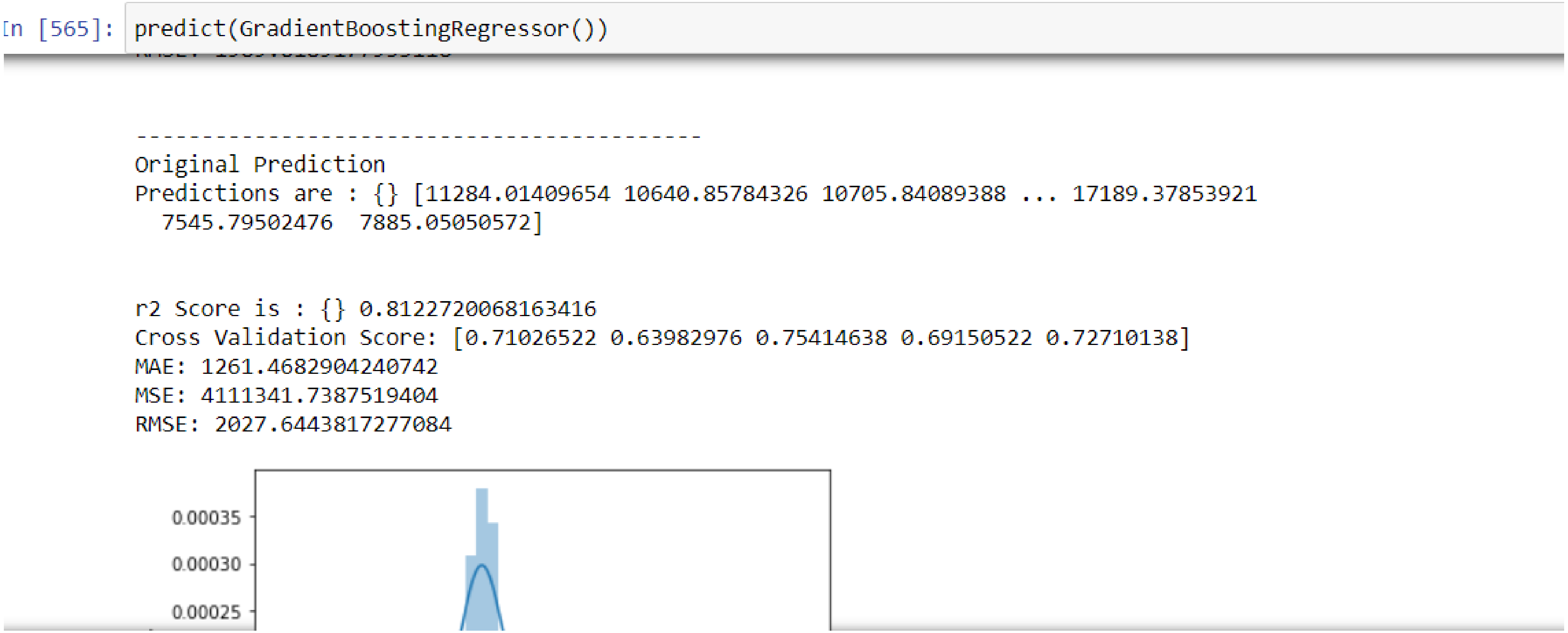
Function to Predict The Model



```
[561]: from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
       def predict(ml_model):
           print('Model is : {}'.format(ml_model))
           model = ml_model.fit(x_train,y_train)
           print("Training Score : {}".format(model.score(x_train,y_train)))
           predictions = model.predict(x_test)
           print("Predictions are : {}" ,format(predictions))
           print('\n')
           print('Testing Prediction')
           r2score = r2_score(y_test,predictions)
           print("r2 Score is : {}",format(r2score))
           print('Cross Validation Score: {}'.format(cross_val_score(ml_model,x_train,y_train,cv=5,scoring='r2')))
           print('MAE: {}'.format(mean_absolute_error(y_test,predictions)))
           print('MSE: {}'.format(mean_squared_error(y_test,predictions)))
           print('RMSE: {}'.format(np.sqrt(mean_squared_error(y_test,predictions))))
           print('\n')
           print('--------------------------------------------')
           print('Original Prediction')
           predictions_train = model.predict(x_train)
           print("Predictions are : {}" ,format(predictions_train))
           print('\n')
           r2score = r2_score(y_train,predictions_train)
           print("r2 Score is : {}",format(r2score))
           print('Cross Validation Score: {}'.format(cross_val_score(ml_model,x_train,y_train,cv=5,scoring='r2')))
           print('MAE: {}'.format(mean_absolute_error(y_train,predictions_train)))
           print('MSE: {}'.format(mean_squared_error(y_train,predictions_train)))
           print('RMSE: {}'.format(np.sqrt(mean_squared_error(y_train,predictions_train))))


           sns.distplot(y_test-predictions)
```

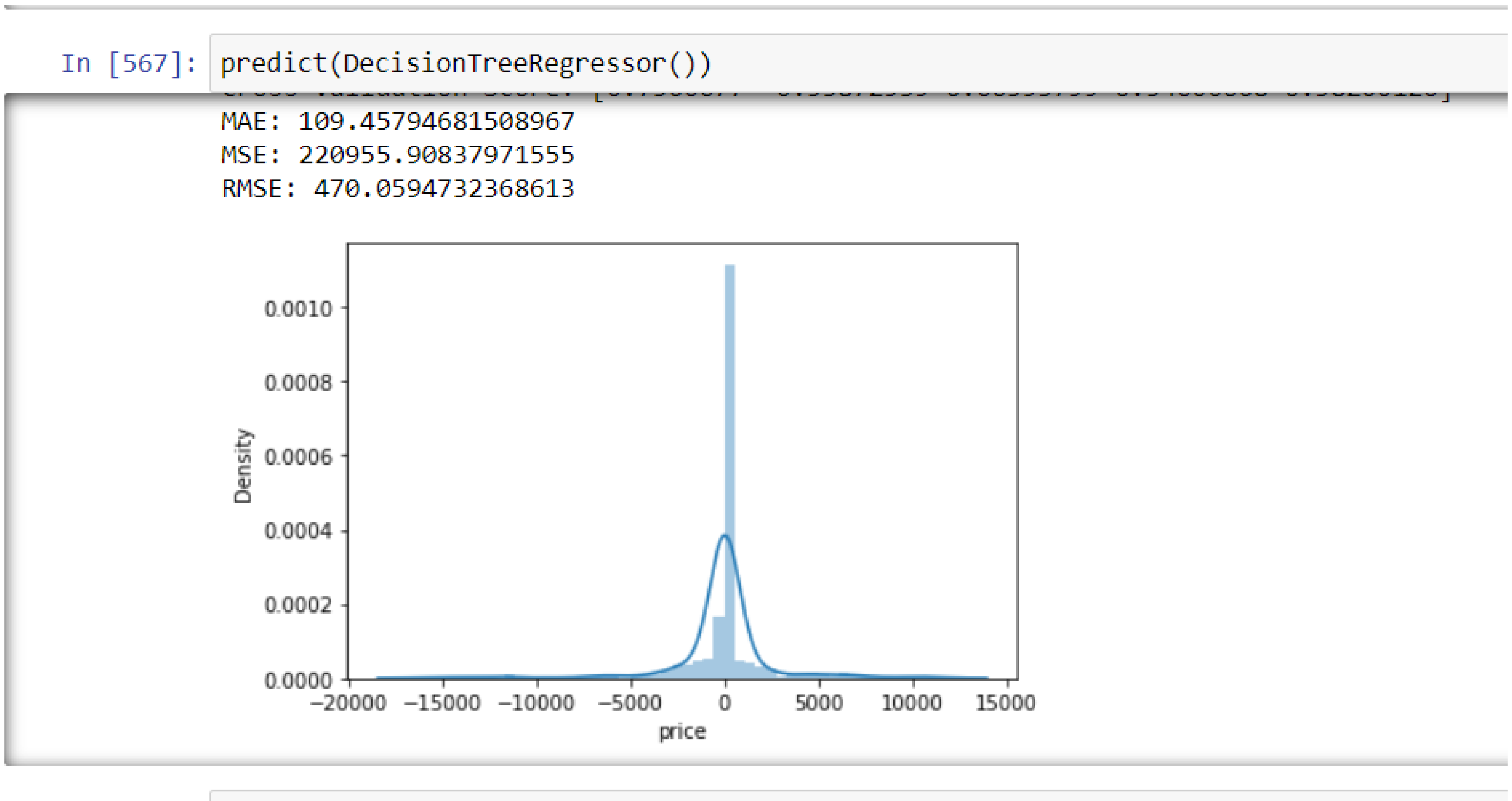
Gradient Bosting -Technique

price



```
In [565]: predict(GradientBoostingRegressor())

          ------------------------------------------
          Original Prediction
          Predictions are : {} [11284.01409654 10640.85784326 10705.84089388 ... 17189.37853921
            7545.79502476  7885.05050572]


          r2 Score is : {} 0.8122720068163416
          Cross Validation Score: [0.71026522 0.63982976 0.75414638 0.69150522 0.72710138]
          MAE: 1261.4682904240742
          MSE: 4111341.7387519404
          RMSE: 2027.6443817277084
```

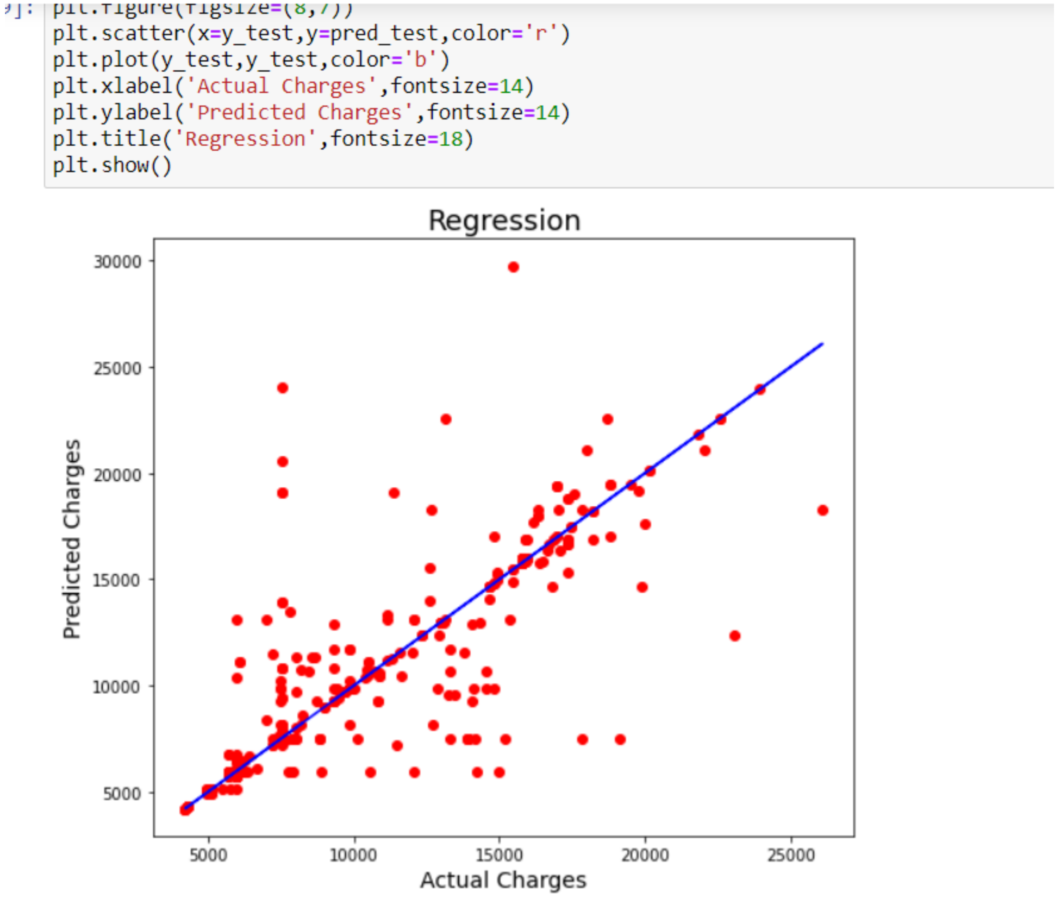
Decision Tree Method



```
In [567]: predict(DecisionTreeRegressor())
```

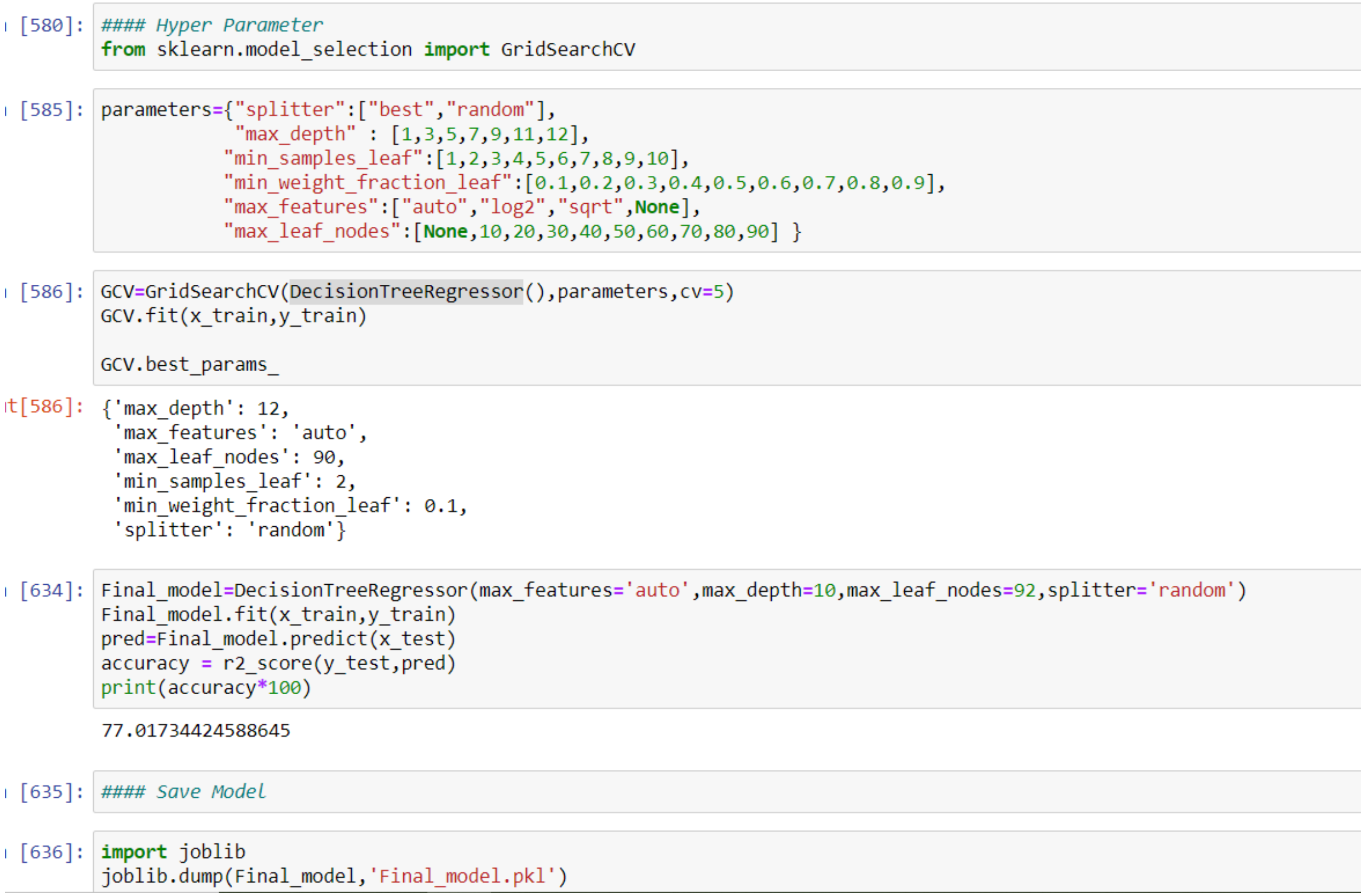
```
Cross Validation Score: [illegible]
MAE: 109.45794681508967
MSE: 220955.90837971555
RMSE: 470.0594732368613
```

Best Fit Line



```
plt.figure(figsize=(8,7))
plt.scatter(x=y_test,y=pred_test,color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel('Actual Charges',fontsize=14)
plt.ylabel('Predicted Charges',fontsize=14)
plt.title('Regression',fontsize=18)
plt.show()
```

Tunning of Model – Hyper Parameter Tunning



[580]:
```
#### Hyper Parameter
from sklearn.model_selection import GridSearchCV
```

[585]:
```
parameters={"splitter":["best","random"],
            "max_depth" : [1,3,5,7,9,11,12],
           "min_samples_leaf":[1,2,3,4,5,6,7,8,9,10],
           "min_weight_fraction_leaf":[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9],
           "max_features":["auto","log2","sqrt",None],
           "max_leaf_nodes":[None,10,20,30,40,50,60,70,80,90] }
```

[586]:
```
GCV=GridSearchCV(DecisionTreeRegressor(),parameters,cv=5)
GCV.fit(x_train,y_train)

GCV.best_params_
```

t[586]:
```
{'max_depth': 12,
 'max_features': 'auto',
 'max_leaf_nodes': 90,
 'min_samples_leaf': 2,
 'min_weight_fraction_leaf': 0.1,
 'splitter': 'random'}
```

[634]:
```
Final_model=DecisionTreeRegressor(max_features='auto',max_depth=10,max_leaf_nodes=92,splitter='random')
Final_model.fit(x_train,y_train)
pred=Final_model.predict(x_test)
accuracy = r2_score(y_test,pred)
print(accuracy*100)
```

```
77.01734424588645
```

[635]:
```
#### Save Model
```

[636]:
```
import joblib
joblib.dump(Final_model,'Final_model.pkl')
```

- Thank You