**Mileston 2 :**

- Reading data and preprocess:

- Previewing data info:

* Noticed from below that there are non-null elements.
* Price is object, which indicates there is something wrong with target.
* Some columns like date, dep\_time, time\_taken, and arr\_time they are all relevant to the same thing which is time, and need to be adjusted.

- For date:

* The date is split into 3 different sections: day, month, and year. All took the format int8 except year took int16 to save memory. Then dropped date.
* def date\_handel(d, cols):  
   d['date\_year'] = d[cols].str.split('-|/', expand=True)[2].astype(np.int16)  
   d['date\_month'] = d[cols].str.split('-|/', expand=True)[1].astype(np.int8)  
   d['date\_day'] = d[cols].str.split('-|/', expand=True)[0].astype(np.int8)  
   d.drop(columns=[cols], inplace=True)  
   # we can also Extract Month Name, Day of Week-Name , Extract Day of Week

- For stop:

* It had a problem with its format is previewed above in the data.head()
* def stop\_fun(x, cols):  
   x[cols] = x[cols].str.split('p', expand=True)[0] + 'p'

- For route:

* It was in dictionary format, so the keys are columns, and the values are the elements of columns.
* def route(x, cols):  
   x['source'] = x[cols].str.split('\'', expand=True)[3]  
   x['destination'] = x[cols].str.split('\'', expand=True)[7]  
   x.drop(columns=[cols], inplace=True)

- For dep\_time and arr\_time:

* It’s split into hours and minutes then add them together.

def time\_handel(d2, cols2):  
  
 # converting to datatime datatype  
 d2[cols2] = pd.to\_datetime(d2[cols2])  
 # all dep\_time in minute  
 d2[cols2] = d2[cols2].dt.minute + d2[cols2].dt.hour\*100

- For time\_taken:

Is the difference between dep\_time and arr\_time

def time\_taken(d4, time1, time2):  
 d4['time\_taken'] = abs(d4[time1]-d4[time2])

-For the categorical data

cols = ('ch\_code', 'type', 'airline', 'source', 'destination', 'stop')

* For each categorical column, feature encoder was used on them.

def feature\_encoder(x, cols):  
 for c in cols:  
 lbl = LabelEncoder()  
 lbl.fit(list(x[c].values))  
 x[c] = lbl.transform(list(x[c].values))  
 return x

for Ticket Category :

we set to each category a certain value

# {cheap = 0 , moderate = 1, expensive = 2 or very expensive = 3}  
def classify\_TicketCategory(x, cols):  
 for val in cols:  
 if (val == 'cheap'):  
 x.append(0)  
 elif(val == 'moderate'):  
 x.append(1)  
 elif(val == 'expensive'):  
 x.append(2)  
 else:  
 x.append(4)

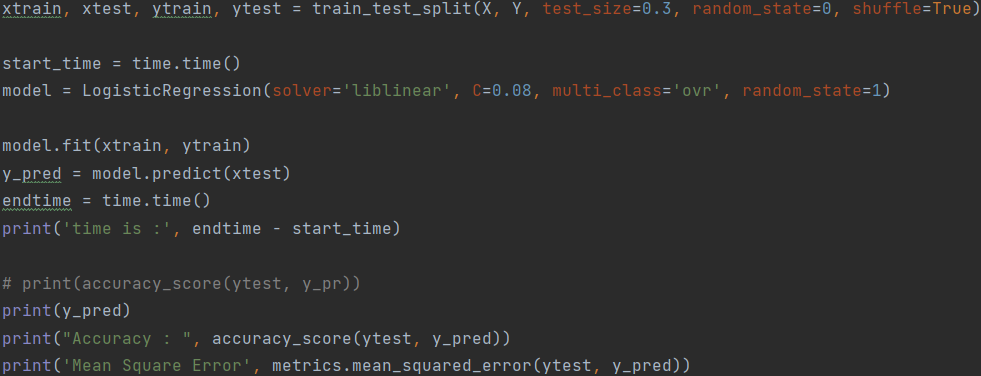
**Analysis the dataset :**

Cleaning data through preprocessing then apply the feature scaling(choose the feature and test and train it ) to get best regression and visualization model.

There are three models :

1. logistic regression
2. Random forest
3. Decision tree

logistic regression :



- Train test split :

- to train X , Y make test size =0.3 and random = 0 and the data shuffle.

- Time start

- Make the model to logistic regression that have Solver to the linear type

and multi class send to it One-vs-rest ‘ovr’ strategy splits a multi-class

classification into one binary classification problem per class

and make random = 1 .

- Fit the model

- Predict the data

- End time

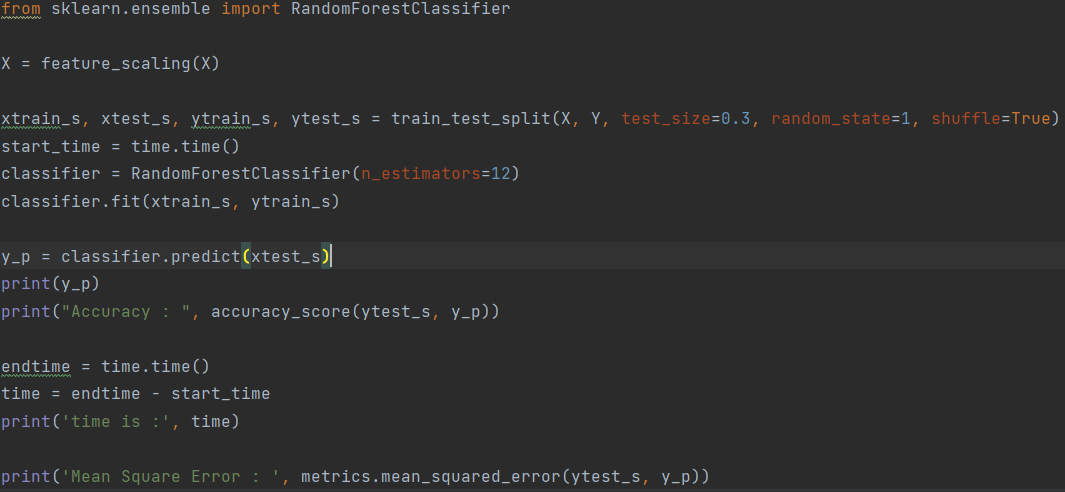
- print the predict

- Print the Accuracy : 0.7140002497814413

- Print the time : 14.524606943130493

- Print the MSE : 0.5567767092682792

Random Forest :



- Feature Scaling For X

- Train test split :

- to train X , Y make test size =0.3 and random = 1 and the data shuffle.

- Time start

- RandomForestClassifier using 12 estimators

- Fit the model

- Predict the data

- End time

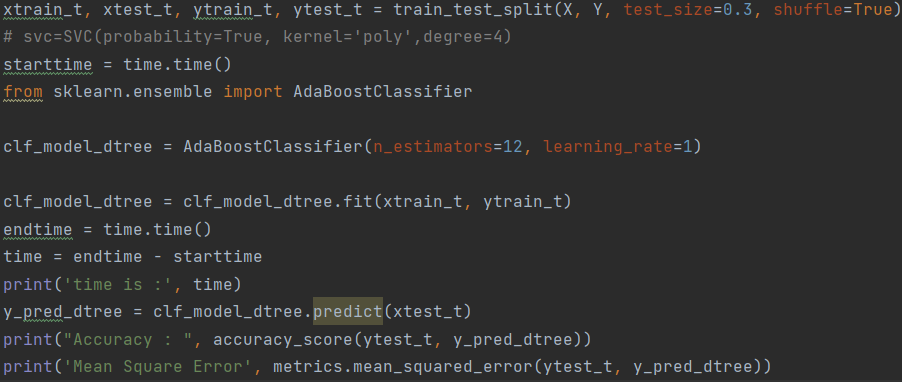
- Print the predict

- Print the Accuracy : 0.9602847508430123

- Print the time : 2.201885223388672

- Print the MSE : 0.05835172002275787

Decision tree :



- Train test split :

- to train X , Y make test size =0.3 and random = 1 and the data shuffle.

- Time start

- AdaBoostClassifier using 12 estimators and learning rate 1.

- Fit the model

- Predict the data

- End time

- Print the predict

- Print the Accuracy : 0.625411098622039

- Print the time : 1.3911468982696533

- Print the MSE : 0.6447691603180551

|  |  |  |  |
| --- | --- | --- | --- |
| models | logistic regression | Random Forest | Decision tree |
| MSE | 0.5567767092682792 | 0.05835172002275787 | 0.6447691603180551 |

**The Conclusion**

The accuracy improved by 96% in the Random Forest model.