**Preprocessing techniques:**

- 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

- Finally the target “price”.

data['date'] = data['date'].replace('[-]', '/', regex=True)  
data['price'] = data['price'].replace('[,]', '', regex=True).astype(int)

**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

**Difference between two models:**

Model1🡪 linear regression of degree 1

Model2🡪 linear regression of polynomial feature degree 3

Training time 🡪the polynomial take more time than the multi-featured model

MSE🡪model 2 improve and get small value than the MSE in model 1

**Feature affect each other :**

the best feature that has more than 50% correlation which affect the price 5 feature

**Features:**

Airline-ch. code-num code-stop -type

**Size of training :**

80% of the dataset

**Size of testing :**

20% of the dataset

**Regression techniques:**

Split the data into training and testing set where testing size is 20% and training size is 80%

Then applying linear regression on data to get the best prediction

**Regression line plot :**

Model1:

Graphical user interface

Description automatically generated

Model2:

**Graphical user interface, chart

Description automatically generated**

**The Conclusion**

The accuracy improved by 92% in the second model