Machine Learning CSE 343/543

Assignment-1 Report

Chaitanya Garg 2021248

Section - A

a) No, If two variables exhibit a strong correlation with a third variable, then, its not necessary for them to showcase a strong correlation with each other. The reason is for a correlation to exist between the two variables, it should occur even when we disregard the third variable.
Eg Consider the case, where the 1st variable is the breadth, 2nd variables is the length and 3rd variables is the area(length x breadth). Now a change in length causes a change in area and similarly a change in breadth causes a change in area but they themselves are independent of each other and hence, not correlated.

b) Defining criterion for a mathematical function to be classified as a Logistic Function is should have the following:
i)It should have domain : R/(-inf,inf)
ii)It should be continuous and gradually inc/descr
iii) It should be give bounded values as outputs like (x,y) where x,y belong to R(Real Numbers)

Classifying Valid Logistic Functions:
i) sin(h(x)) : Not valid as it’s range is not bounded

ii) cos(h(x)) : Not valid as it’s range is not bounded
iii) tan(h(x)) : Valid as its fulfills the criterion, its range is (-1,1) which can be converted if needed to (0,1) by
f(x)= (tan(h(x))+1)/2
iv)signum(x) : Not valid as its not a continuous function

c) For very sparse datasets, leave one out technique is beneficial because it is a validation technique such that each observation is considered as the validation set while the rest are considered as the training set. So, for a dataset having n values. The model is fitted for the training set and prediction is made on the validation set n times. It helps to reduce bias and randomness and prevents the problem of overfitting. It is different from the traditional K-fold technique as its a special case wherein k=n. In traditional technique, data is divided in k sets of almost equal length then, one is chosen as the validation set and the rest is the testing set. Then, training happens and predictions are made on the validation set. This procedure is followed k times, until the model is not trained and tested for all subsets. While in leave one out, each set is equal and has length/size 1.

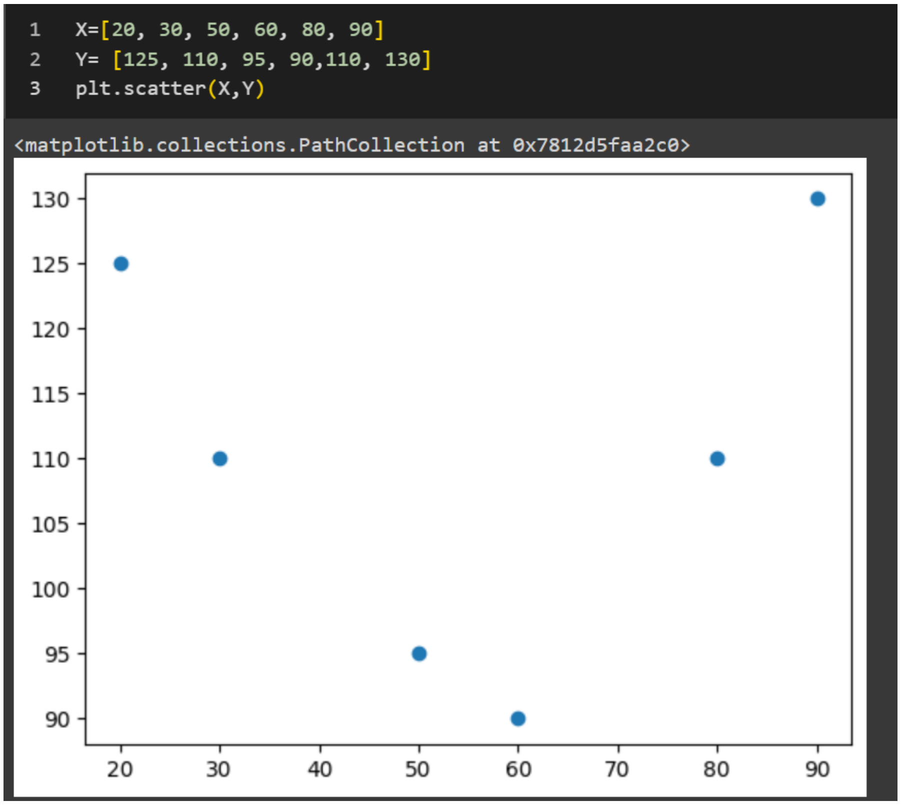
d) The general equation for a line in slope intercept form is
y = mx + c
Our goal is to find the coefficients m,c for a set of n data-points (xi,yi) using Least Square Regression.
i) We calculate xMean and yMean
xMean = 1/n(sum (i = 1 to n):xi)
yMean = 1/n(sum (i = 1 to n):yi)
ii) Calculating the mean deviation for x and y,i.e.,(x-xMean)
xmi = x - xMean
ymi = y - yMean
Val = (sum (i = 1 to n): xmi * ymi)/ (sum (i = 1 to n): xmi * xmi)

This Val is the much needed slope(m) of the avg line of the given points.
iii) To get c we use,
c = yMean - m * xMean

Following, this procedure, we get the values of the coefficients of m and c using Least Square Regression.

e) Parameters to be measured to estimated for the linear Regression model $Y = \alpha+\beta x+\varepsilon$ ε $N(0,\sigma)$ are: a) α, β, σ
Since, α, β are the coefficients of the model and noise ε depends on the standard deviation σ

f) The answer is (d) $Y= \alpha + \beta 1x + \beta 2x$^2 $+\varepsilon$ $\beta 2 > 0$, since on plotting we get an upward opening parabola with a positive intercept

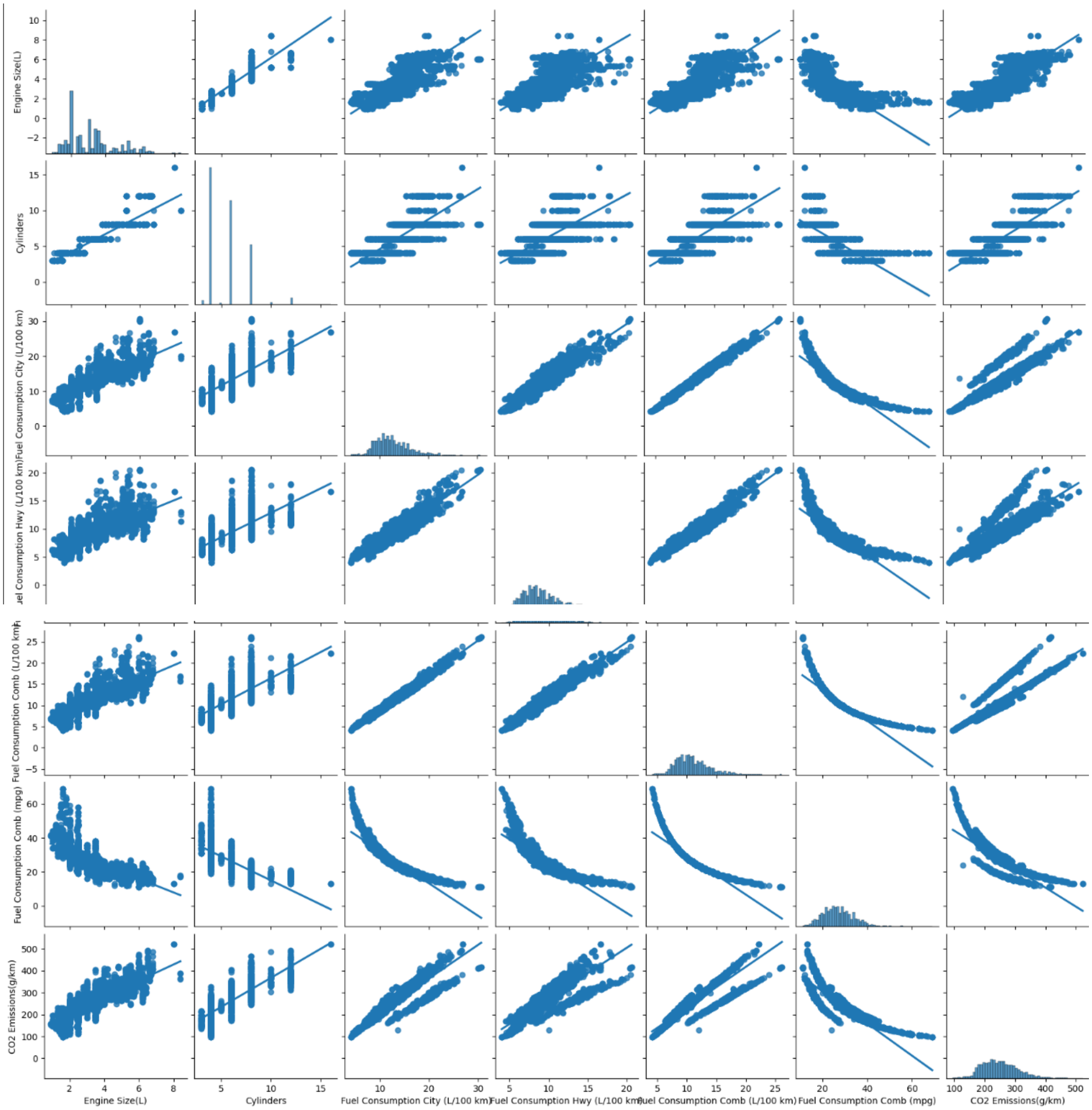


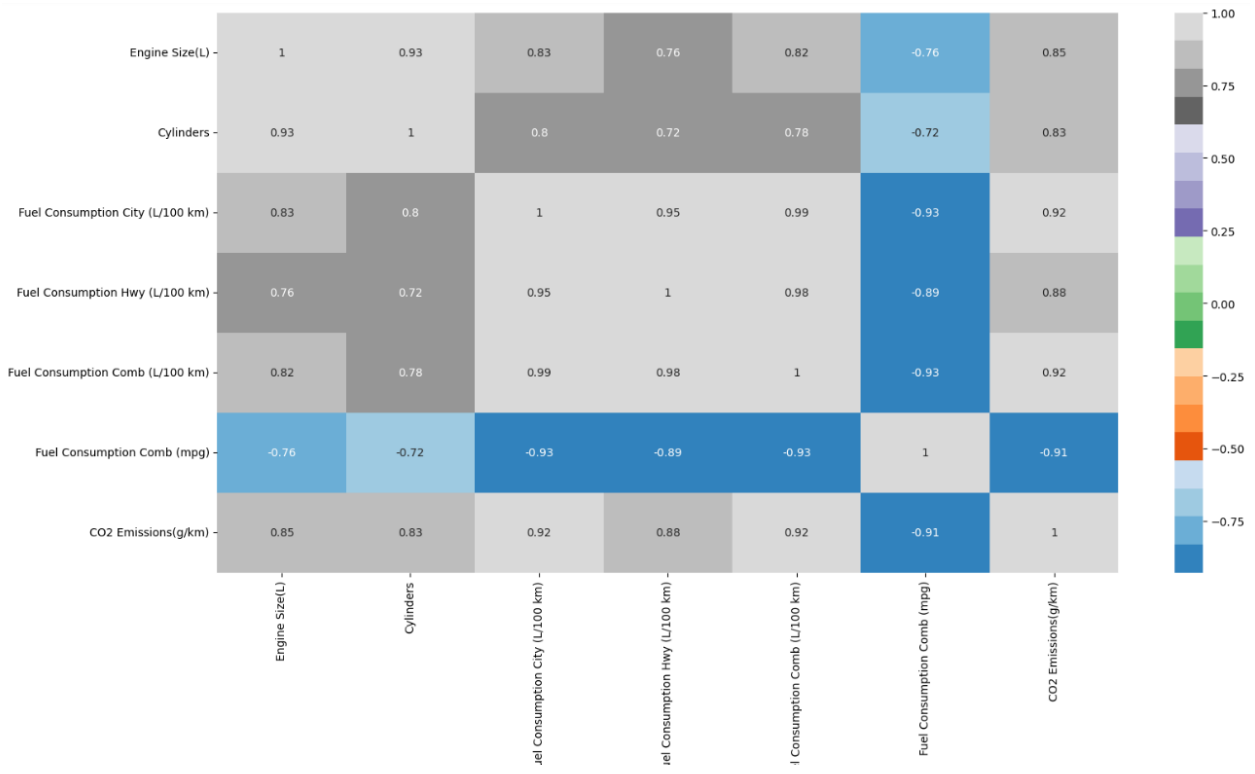
```
X=[20, 30, 50, 60, 80, 90]
Y= [125, 110, 95, 90,110, 130]
plt.scatter(X,Y)
```

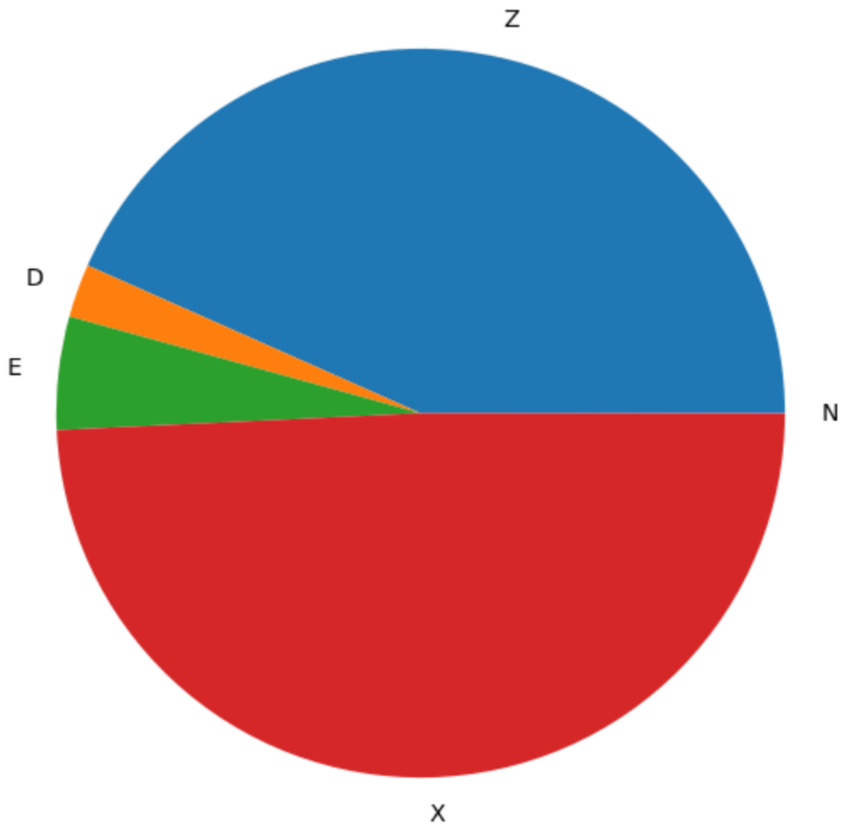
```
<matplotlib.collections.PathCollection at 0x7812d5faa2c0>
```

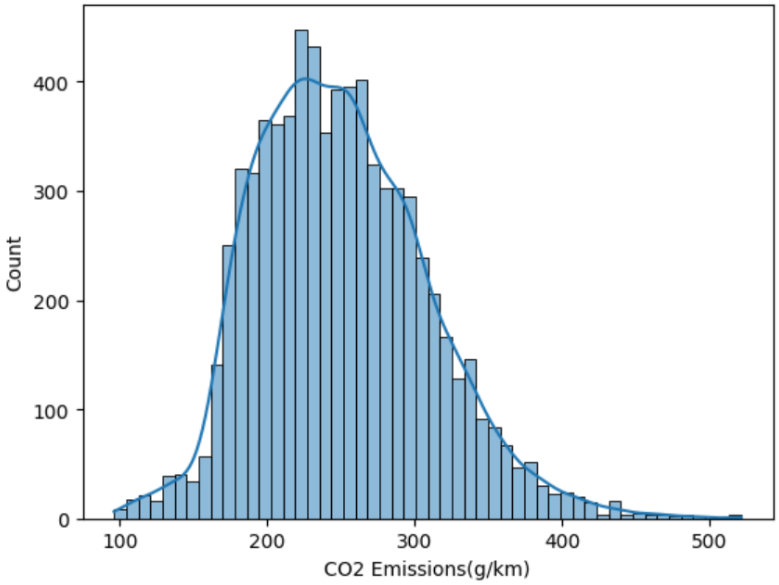
Section - C

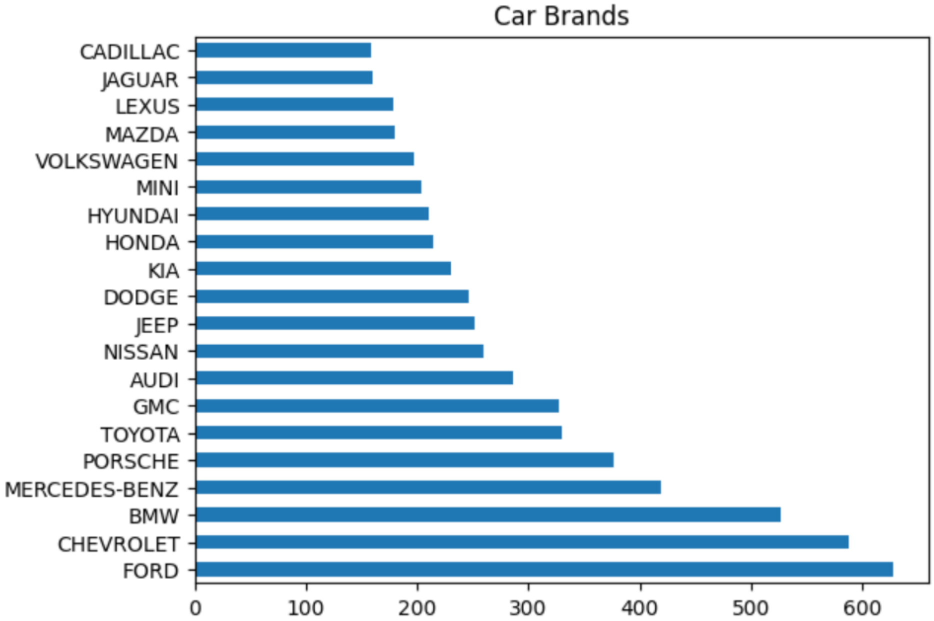
a) After making scatter plots, histograms, barplots, heatmaps and piecharts, the following inferences were made

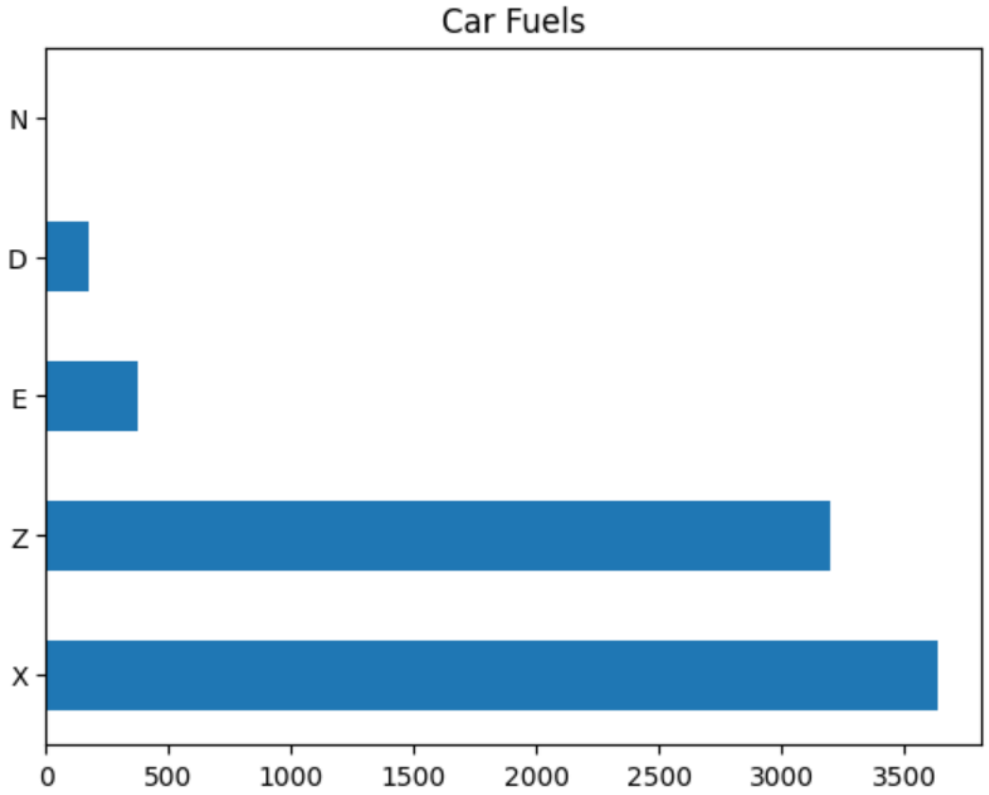



Engine Size(L)
Cylinders
Fuel Consumption City (L/100 km)
Fuel Consumption Hwy (L/100 km)
Fuel Consumption Comb (L/100 km)
Fuel Consumption Comb (mpg)
CO2 Emissions(g/km)
1 0.93 0.83 0.76 0.82 -0.76 0.85
0.93 1 0.8 0.72 0.78 -0.72 0.83
0.83 0.8 1 0.95 0.99 -0.93 0.92
0.76 0.72 0.95 1 0.98 -0.89 0.88
0.82 0.78 0.99 0.98 1 -0.93 0.92
-0.76 -0.72 -0.93 -0.89 -0.93 1 -0.91
0.85 0.83 0.92 0.88 0.92 -0.91 1
Engine Size(L)
Cylinders
uel Consumption City (L/100 km)
uel Consumption Hwy (L/100 km)
l Consumption Comb (L/100 km)
Fuel Consumption Comb (mpg)
CO2 Emissions(g/km)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

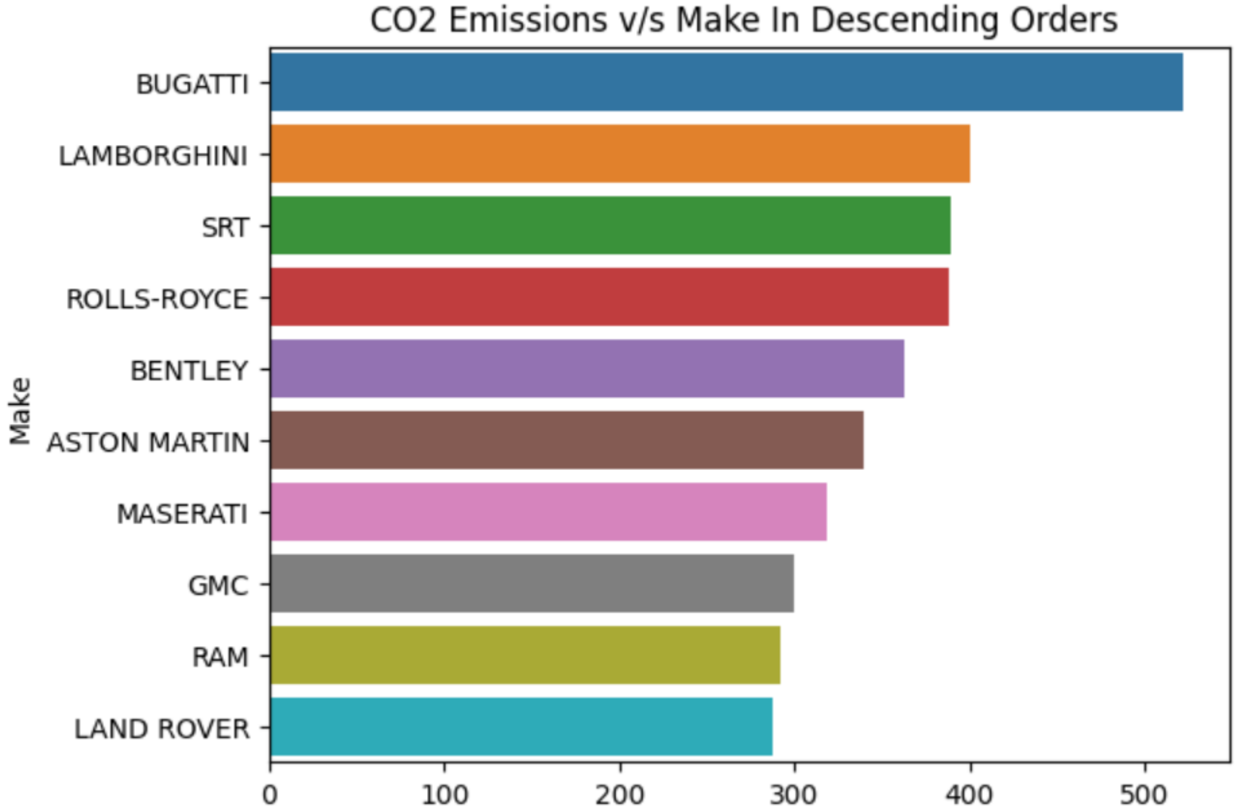

Z
D
E
N
X


Count
0
100
200
300
400
100
200
300
400
500
CO2 Emissions(g/km)

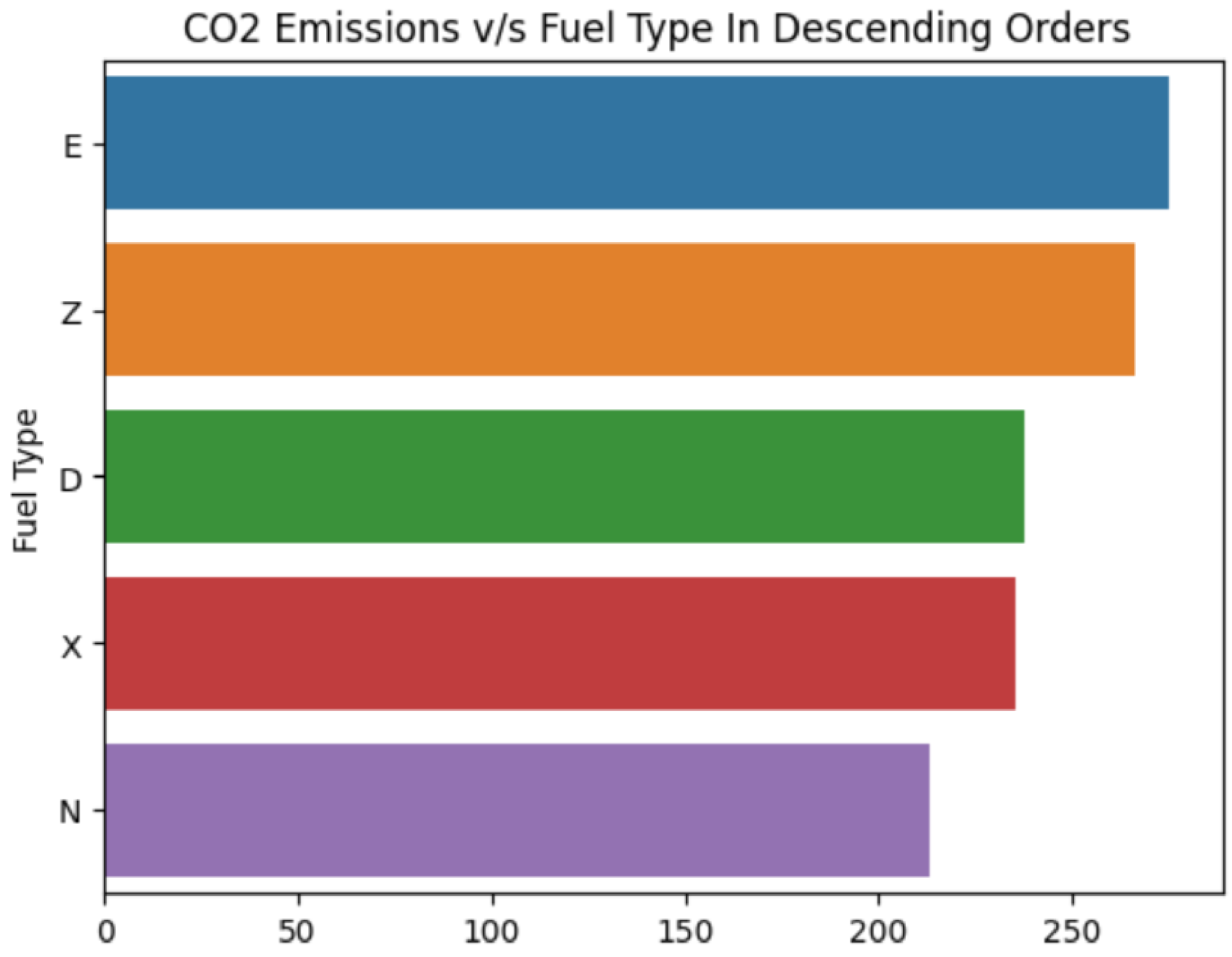

Car Brands
CADILLAC
JAGUAR
LEXUS
MAZDA
VOLKSWAGEN
MINI
HYUNDAI
HONDA
KIA
DODGE
JEEP
NISSAN
AUDI
GMC
TOYOTA
PORSCHE
MERCEDES-BENZ
BMW
CHEVROLET
FORD
0
100
200
300
400
500
600



Car Fuels



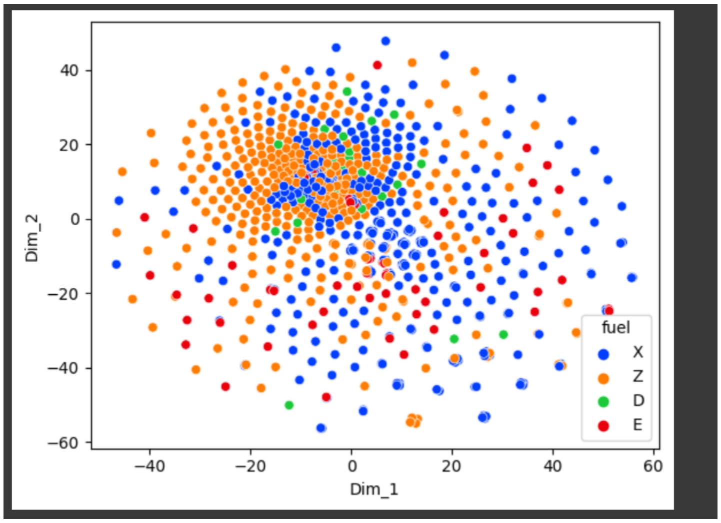
CO2 Emissions v/s Make In Descending Orders

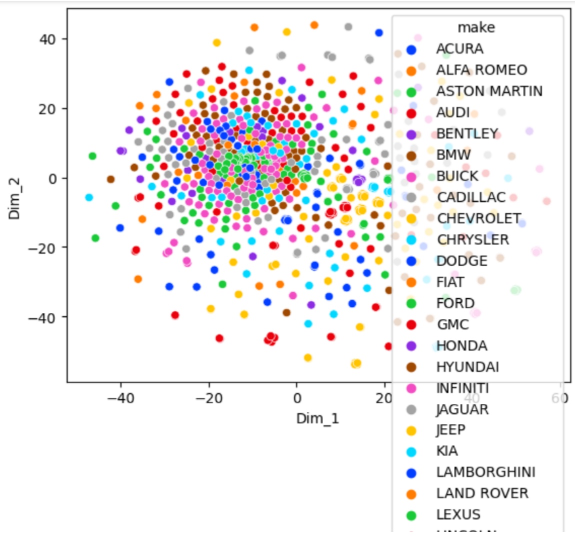


i) CO2 varies linearly with Fuel Consumption parameters (L/100km) ones
ii) CO2 varies inversely with Fuel Consumption(mpg)
iii) CO2 variation with cylinders and engines is disproportionate
iv) Fuel Type X is the most common type of fuel in vehicles
v) CO2 emissions have a Guassian like distribution with them having 250g/km as the most common emission
vi) Ford Vehicles were found to be the most in the dataset
vii) Bugatti Vehicles are responsible for the most mean CO2 emissions while Smart vehicles have the least
viii) E type fuel lends to the most CO2 emissions while N has the least

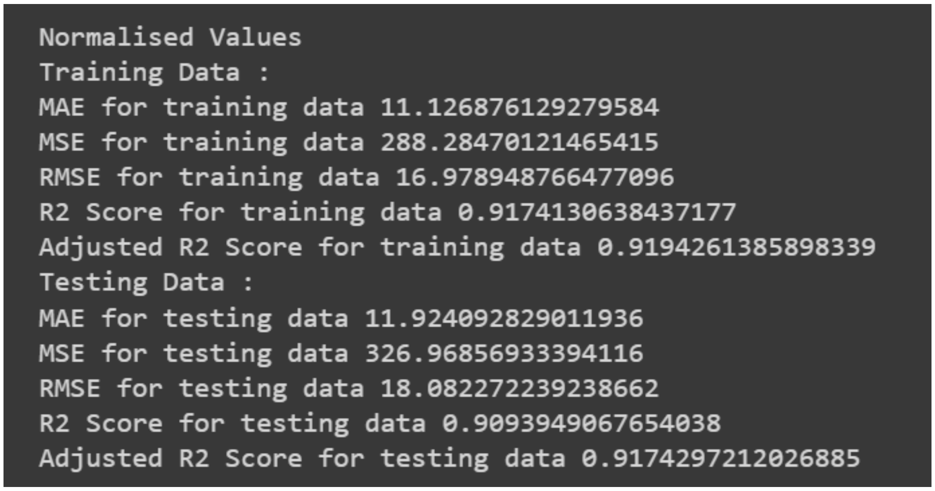
All these deductions have been made from the graphs plotted in Part-a of the code file.

b) T-SNE plots are used to reduce multi dimensional data to just 2 dimensions for simpler visualization. In the plotted TSNE graphs, the observation was that partial clustering was observed but otherwise the data was largely inseparable.





c) In this part, first we check for missing values to remove/fill them. Since there are none, we move onto the next step.
Next, we check for duplicates, and we find that the data does consist of duplicates so we remove those.
Next, we normalize our data for better accuracy and speed by using Standard_scaler().
Now, we make functions to calculate the MAE, MSE, RMSE, R2 score and Adjusted R2 score.
All these functions are fitted in another function PrintAll_() for easy functionality and simpler calling.
Now, we split the data set into the required 80:20 split and do our work.

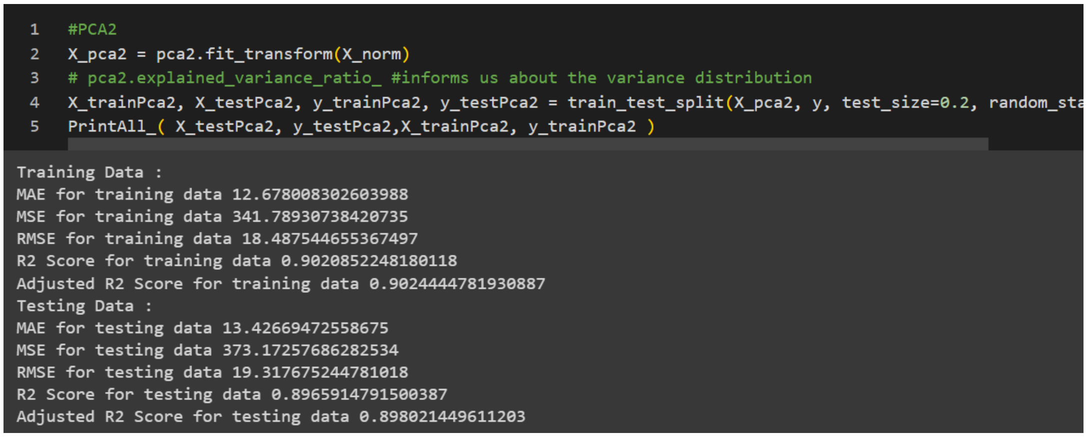


```
Normalised Values
Training Data :
MAE for training data 11.126876129279584
MSE for training data 288.28470121465415
RMSE for training data 16.978948766477096
R2 Score for training data 0.9174130638437177
Adjusted R2 Score for training data 0.9194261385898339
Testing Data :
MAE for testing data 11.924092829011936
MSE for testing data 326.96856933394116
RMSE for testing data 18.082272239238662
R2 Score for testing data 0.9093949067654038
Adjusted R2 Score for testing data 0.9174297212026885
```

d) We now make use of PCA(Principal Component Analysis), its a technique wherein we choose the parameters that have the most changing variance. In other words, preference is given to the most changing parameters. PCA is performed for n=2,4,6,8 &10.

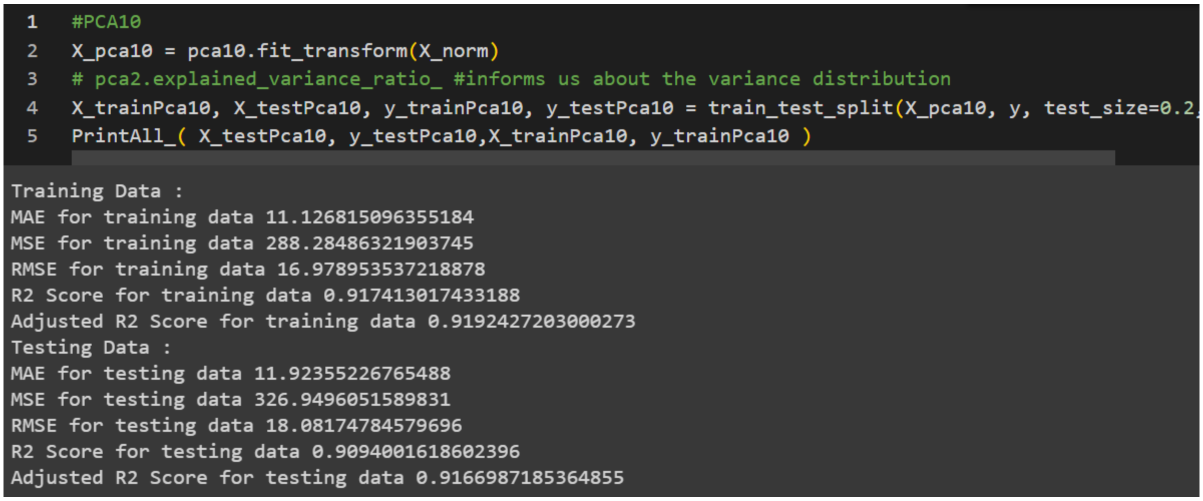
The observation is that the MAE, MSE, RMSE for n=2 are more but the difference is not too much while both n=8 and 10 close in to the actual value obtained on the original model.

The R2 score for n=2 is also very credible with a value of 0.90 wherein the original value had an initial R2 score 0.917 so, even with just considering 2 parameters the model performs pretty decently.



```
#PCA2
X_pca2 = pca2.fit_transform(X_norm)
# pca2.explained_variance_ratio_ #informs us about the variance distribution
X_trainPca2, X_testPca2, y_trainPca2, y_testPca2 = train_test_split(X_pca2, y, test_size=0.2, random_sta
PrintAll_( X_testPca2, y_testPca2,X_trainPca2, y_trainPca2 )
```

```
Training Data :
MAE for training data 12.678008302603988
MSE for training data 341.78930738420735
RMSE for training data 18.487544655367497
R2 Score for training data 0.9020852248180118
Adjusted R2 Score for training data 0.9024444781930887
Testing Data :
MAE for testing data 13.42669472558675
MSE for testing data 373.17257686282534
RMSE for testing data 19.317675244781018
R2 Score for testing data 0.8965914791500387
Adjusted R2 Score for testing data 0.898021449611203
```



```
#PCA10
X_pca10 = pca10.fit_transform(X_norm)
# pca2.explained_variance_ratio_ #informs us about the variance distribution
X_trainPca10, X_testPca10, y_trainPca10, y_testPca10 = train_test_split(X_pca10, y, test_size=0.2,
PrintAll_( X_testPca10, y_testPca10,X_trainPca10, y_trainPca10 )
```

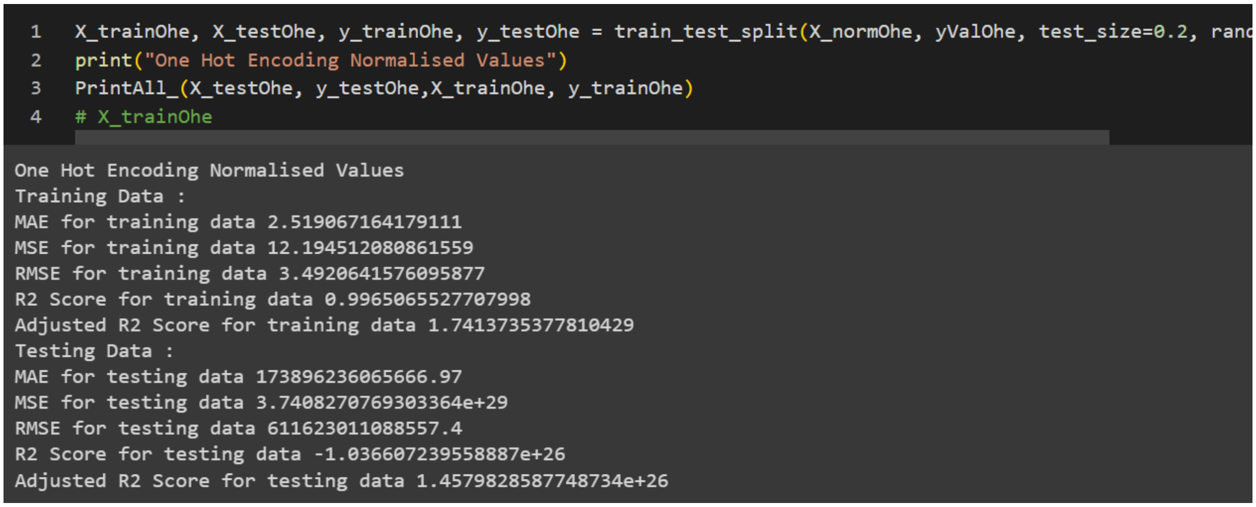
```
Training Data :
MAE for training data 11.126815096355184
MSE for training data 288.28486321903745
RMSE for training data 16.978953537218878
R2 Score for training data 0.917413017433188
Adjusted R2 Score for training data 0.9192427203000273
Testing Data :
MAE for testing data 11.92355226765488
MSE for testing data 326.9496051589831
RMSE for testing data 18.08174784579696
R2 Score for testing data 0.9094001618602396
Adjusted R2 Score for testing data 0.9166987185364855
```

e) We make a new dataframe for One Hot Encoding, so as to retain the other data for repetitive calculations on part-c) prepared dataset. So, first we remove duplicates.

To do, OHE, we convert the string categorical features into number and then apply the encoding function, the not needed columns are then deleted.

Upon applying OHE, the size of the dataset increases rapidly to about 2150 columns.

Next, we train the model, and see the predictions. Due to there being so many parameters the model performs extremely poorly on the test data, MAE, MSE, RMSE are off the charts and R2 score is very negative.

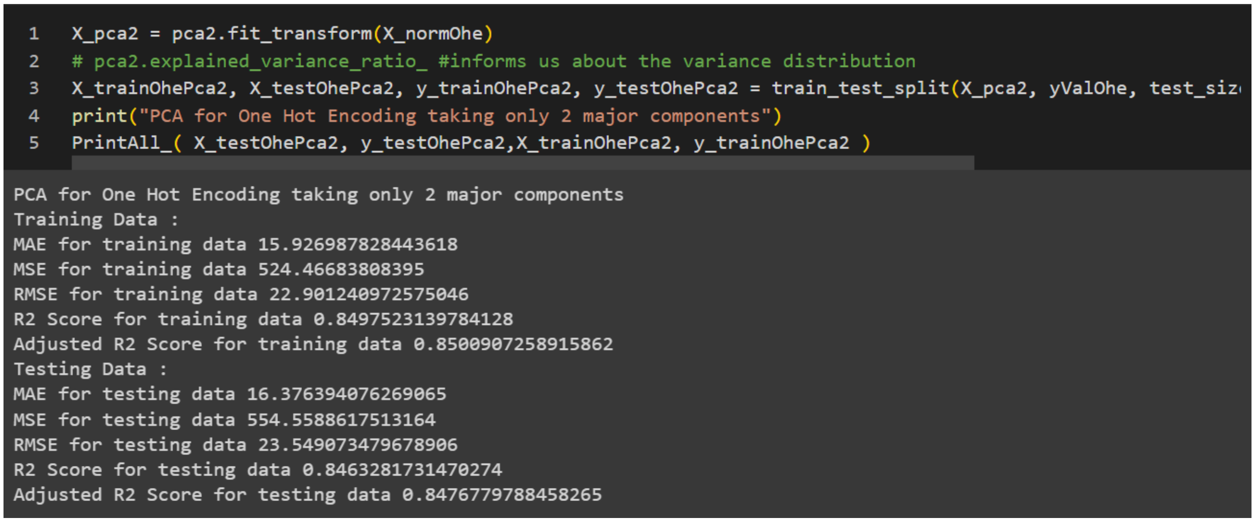


```
X_trainOhe, X_testOhe, y_trainOhe, y_testOhe = train_test_split(X_normOhe, yValOhe, test_size=0.2, rand
print("One Hot Encoding Normalised Values")
PrintAll_(X_testOhe, y_testOhe,X_trainOhe, y_trainOhe)
# X_trainOhe
```

```
One Hot Encoding Normalised Values
Training Data :
MAE for training data 2.519067164179111
MSE for training data 12.194512080861559
RMSE for training data 3.4920641576095877
R2 Score for training data 0.9965065527707998
Adjusted R2 Score for training data 1.7413735377810429
Testing Data :
MAE for testing data 173896236065666.97
MSE for testing data 3.7408270769303364e+29
RMSE for testing data 611623011088557.4
R2 Score for testing data -1.036607239558887e+26
Adjusted R2 Score for testing data 1.4579828587748734e+26
```

f) Now, we apply PCA for various values of n

For n=2, the model performs very decently the MAE, RMSE are close to the original model. MSE is almost double while R2 score is 0.84



```
X_pca2 = pca2.fit_transform(X_normOhe)
# pca2.explained_variance_ratio_ #informs us about the variance distribution
X_trainOhePca2, X_testOhePca2, y_trainOhePca2, y_testOhePca2 = train_test_split(X_pca2, yValOhe, test_siz
print("PCA for One Hot Encoding taking only 2 major components")
PrintAll_( X_testOhePca2, y_testOhePca2,X_trainOhePca2, y_trainOhePca2 )
```

```
PCA for One Hot Encoding taking only 2 major components
Training Data :
MAE for training data 15.926987828443618
MSE for training data 524.46683808395
RMSE for training data 22.901240972575046
R2 Score for training data 0.8497523139784128
Adjusted R2 Score for training data 0.8500907258915862
Testing Data :
MAE for testing data 16.376394076269065
MSE for testing data 554.5588617513164
RMSE for testing data 23.549073479678906
R2 Score for testing data 0.8463281731470274
Adjusted R2 Score for testing data 0.8476779788458265
```

Observations Table for Training Data

n	MAE	MSE	RMSE	R2	ADJR2
2	15.92	524.46	22.90	0.849	0.85
10	15.30	486.52	22.05	0.86	0.862
25	14.13	387.39	19.9	0.88	0.89
50	13.02	324.72	18.02	0.90	0.91
100	10.04	183.25	13.5	0.94	0.966
200	9.06	146.77	12.11	0.957	0.99
400	9.32	151.31	12.30	0.956	1.03
500	8.93	139.80	11.82	0.959	1.06
1000	7.518	103.3	10.16	0.97	1.21

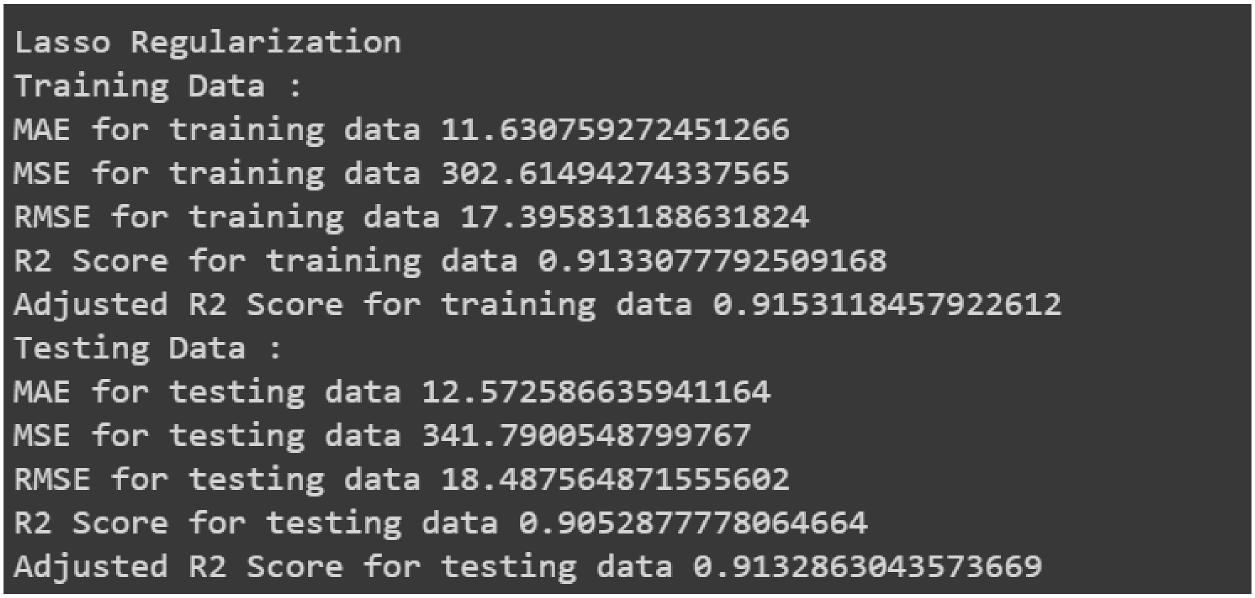
Observations Table for Testing Data

n	MAE	MSE	RMSE	R2	ADJR2
2	16.37	554.55	23.54	0.846	0.847

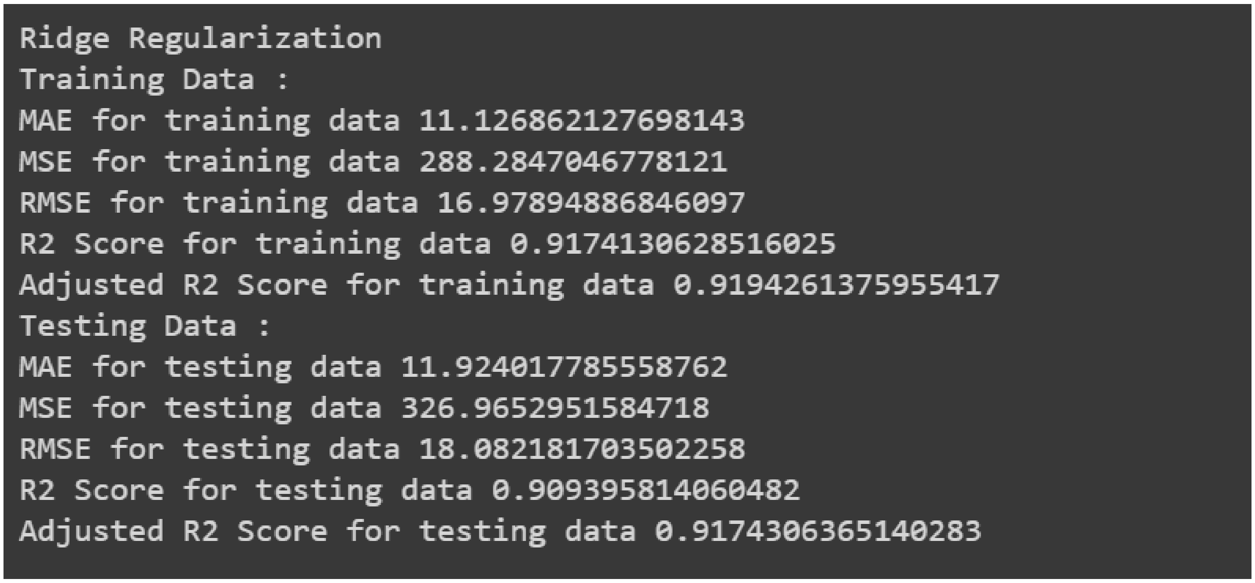
10	15.75	516.43	22.72	0.856	0.863
25	14.57	426.55	60.65	0.88	0.899
50	13.62	344.33	18.55	0.90	0.942
100	10.394	210.61	14.51	0.94	1.023
200	10.23	183.43	13.54	0.949	1.24
400	10.12	185.97	13.63	0.948	1.39
500	10.12	187.72	13.70	0.947	1.57
1000	9.80	182.02	13.49	0.949	4.65

We see a decreasing trend in the values of MAE, MSE, RMSE and increasing trend in R2 and AdjR2 except for n=400 wherein the opp. happens.
PCA on OHE is so good that gives better results than the original normalized model.

g) We apply Lasso (L1) and Ridge(L2) Regularization on the model. Lasso Regression applies an absolute penalty on the coefficients while Ridge applies a penalty on sq of coefficients thus forcing the coefficients to be less in value.



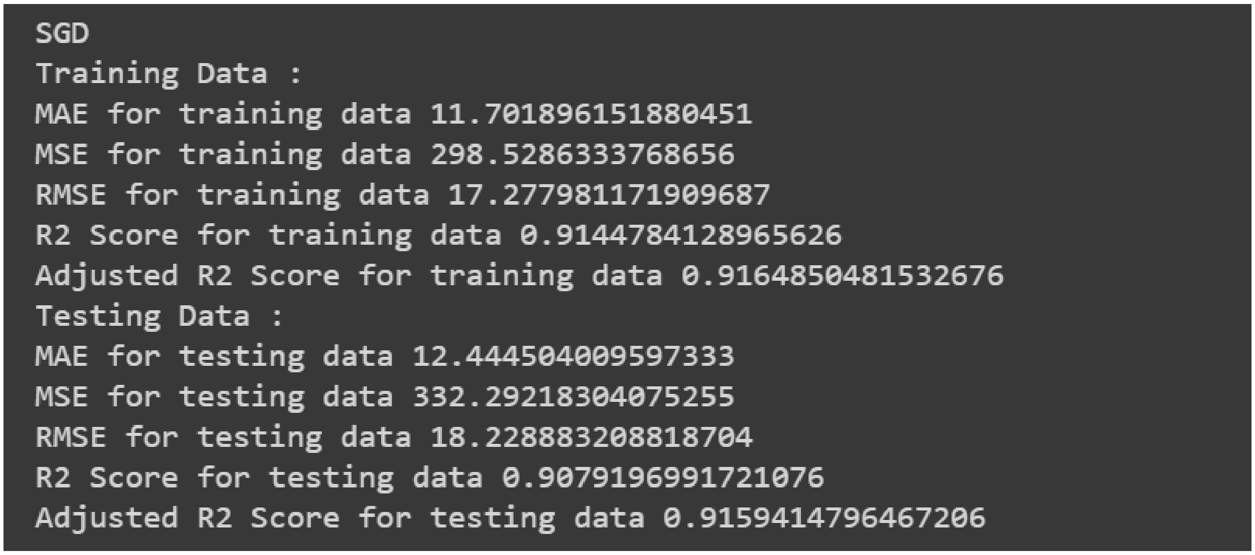
```
Lasso Regularization
Training Data :
MAE for training data 11.630759272451266
MSE for training data 302.61494274337565
RMSE for training data 17.395831188631824
R2 Score for training data 0.9133077792509168
Adjusted R2 Score for training data 0.9153118457922612
Testing Data :
MAE for testing data 12.572586635941164
MSE for testing data 341.7900548799767
RMSE for testing data 18.487564871555602
R2 Score for testing data 0.9052877778064664
Adjusted R2 Score for testing data 0.9132863043573669
```



```
Ridge Regularization
Training Data :
MAE for training data 11.126862127698143
MSE for training data 288.2847046778121
RMSE for training data 16.97894886846097
R2 Score for training data 0.9174130628516025
Adjusted R2 Score for training data 0.9194261375955417
Testing Data :
MAE for testing data 11.924017785558762
MSE for testing data 326.9652951584718
RMSE for testing data 18.082181703502258
R2 Score for testing data 0.909395814060482
Adjusted R2 Score for testing data 0.9174306365140283
```

The values obtained after regularization are very similar to the ones obtained before hinting that the coefficients were already not very high in value.

h) Lastly, we apply Stochastic Gradient Descent wherein every parameter/coefficient is altered after every iteration, wherein only row is worked on at a time.



```
SGD
Training Data :
MAE for training data 11.701896151880451
MSE for training data 298.5286333768656
RMSE for training data 17.277981171909687
R2 Score for training data 0.9144784128965626
Adjusted R2 Score for training data 0.9164850481532676
Testing Data :
MAE for testing data 12.444504009597333
MSE for testing data 332.29218304075255
RMSE for testing data 18.228883208818704
R2 Score for testing data 0.9079196991721076
Adjusted R2 Score for testing data 0.9159414796467206
```