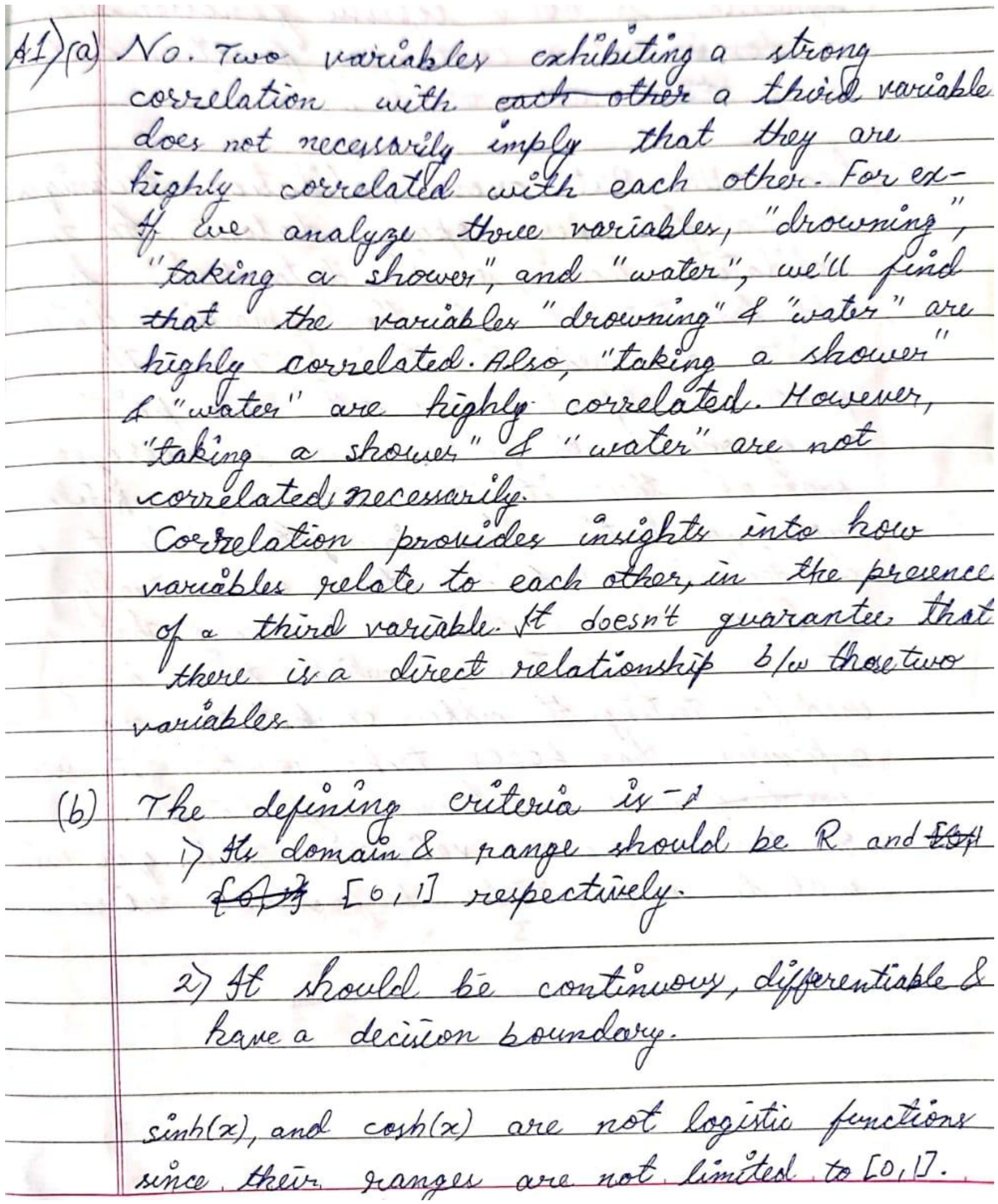
ML Assignment 1

Mohammad Sufyan Azam
2020312

Q1)



A1) (a) No. Two variables exhibiting a strong correlation with ~~each other~~ a third variable does not necessarily imply that they are highly correlated with each other. For ex- If we analyze three variables, "drowning", "taking a shower", and "water", we'll find that the variables "drowning" & "water" are highly correlated. Also, "taking a shower" & "water" are highly correlated. However, "taking a shower" & "water" are not correlated necessarily.
Correlation provides insights into how variables relate to each other, in the preence of a third variable. It doesn't guarantee that there is a direct relationship b/w those two variables.

(b) The defining criteria is -
1) Its domain & range should be R and ~~[0,1]~~ ~~[0,1]~~ [0,1] respectively.

2) It should be continuous, differentiable & have a decision boundary.

sinh(x), and cosh(x) are not logistic functions since their ranges are not limited to [0,1].

tanh(x) is a logistic function since its range lies in [-1,1] which can be shifted to [0,1] & its domain is R. Also it's continuous & differentiable too.

signum(x) is not a logistic function because its ~~domain~~ not a continuous function & its range ~~[illegible]~~ ~~does not~~ is {-1,0,1}.

(c) Leave-One-Out cross validation technique is used for very sparse datasets. In this validation technique, one random data point is held for testing while the remaining data points are used for training. Then this process is repeated for all data points one by one & the performance is averaged from all these iterations. However, in K-fold cross validation technique, the dataset is divided into K-folds that are mutually exclusive of each other and among them, K-1 sets are used for training while the remaining is used for testing. It makes computation more expensive than LOOCV technique. Also it's useful ~~for larg~~ when we have larger datasets.

In this case, LOOCV validation technique will provide more stable performance metric.

DELTA
Date / /
Pg No.

(d) Let the hypothesis function be $h(x) = mx + c$ for the linear regression model.
To find the parameters m & c, we need to minimise: $\frac{1}{n}\times\sum_{i=1}^{n}(y_i - (mx_i + c))^2 = J(y,x) = MSE$

Taking partial derivates of it wrt m & c,

$$\frac{\delta(J)}{\delta(m)} = 2\sum_{i=1}^{n}(y_i - (mx_i + c))(-x_i) = 0$$

$$\Rightarrow \sum_{i=1}^{n}(-x_i y_i + m x_i^2 + c x_i) = 0$$

$$\Rightarrow m\sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i y_i - c\sum_{i=1}^{n} x_i$$

$$\Rightarrow m = \frac{\Sigma x_i y_i - c\Sigma x_i}{\Sigma x_i^2} \quad — ①$$

$$\frac{\delta(J)}{\delta(c)} = 2\sum_{i=1}^{n}(y_i - (mx_i + c))(-1) = 0$$

$$\Rightarrow \Sigma y_i - m\Sigma x_i - cn = 0$$

$$\Rightarrow m = \frac{\Sigma y_i - cn}{\Sigma x_i} \quad — ②$$

From ① & ②,

$$\boxed{m = \frac{n\Sigma x_i y_i - \Sigma x_i \Sigma y_i}{n\Sigma x_i^2 - (\Sigma x_i)^2}}$$

$$\boxed{c = \frac{\Sigma y_i - m\Sigma x_i}{n}}$$

DELTA
Date / /
Pg No.

~~Ca~~ Putting m in c,

$$\therefore \boxed{c = \frac{\Sigma y_i}{n} - \left(\frac{\Sigma x_i}{n}\right)\left(\frac{n\Sigma x_i y_i - \Sigma x_i \Sigma y_i}{n\Sigma x_i^2 - (\Sigma x_i)^2}\right)}$$

(e) (a) α, β, σ

α, β are the weights of the model, so it needs to be estimated.
σ is the standard deviation of the noise parameter ϵ which represents the variability of data around our predicted line. We need to estimate σ so that we can ensure the accuracy of the predictions is good.

(f) (d) $Y = \alpha + \beta_1 x + \beta_2 x^2 + \epsilon \quad \beta_2 > 0$

On analyzing the relationship b/w X & Y we see that the Y values first ~~increase~~ decreases then increases, suggesting that there is a parabolic relationship of X & Y & there is a point of local minima.
Hence, (d) is correct.

Q2)

Q3)

(a) These are the insights which I got from the code visualisations-

Insight 1: Through the scatter plots we can clearly see that as the engine size increases, the CO2 Emissions directly increase. Also, on increasing the no of cylinders the CO2 Emissions directly increase. Thus they have a linear dependence on each other. This is also verified by the Correlation Heatmap.

Insight 2: Features such as Fuel Consumption Hwy (L/100 km), Fuel Consumption City (L/100 km), and Fuel Consumption Comb (L/100 km) are also linearly dependent on CO2 Emissions. As the values of these features increase, the CO2 Emissions also increases.

Insight 3: Fuel Consumption Comb (mpg) decreases as the CO2 Emissions increase. It can also be approximated by a linear relationship with CO2 Emissions (although it actually follows a non-linear relationship) which was verified by making their scatterplot and pairplots too.

Insight 4: Through the heatmap, we see that the cylinders are more correlated to the Engine Size, rather than CO2 Emissions which in turn is highly correlated to the CO2 Emissions. So, they can be considered as a derived feature of Engine Size thus having an indirect relationship to CO2 Emissions.

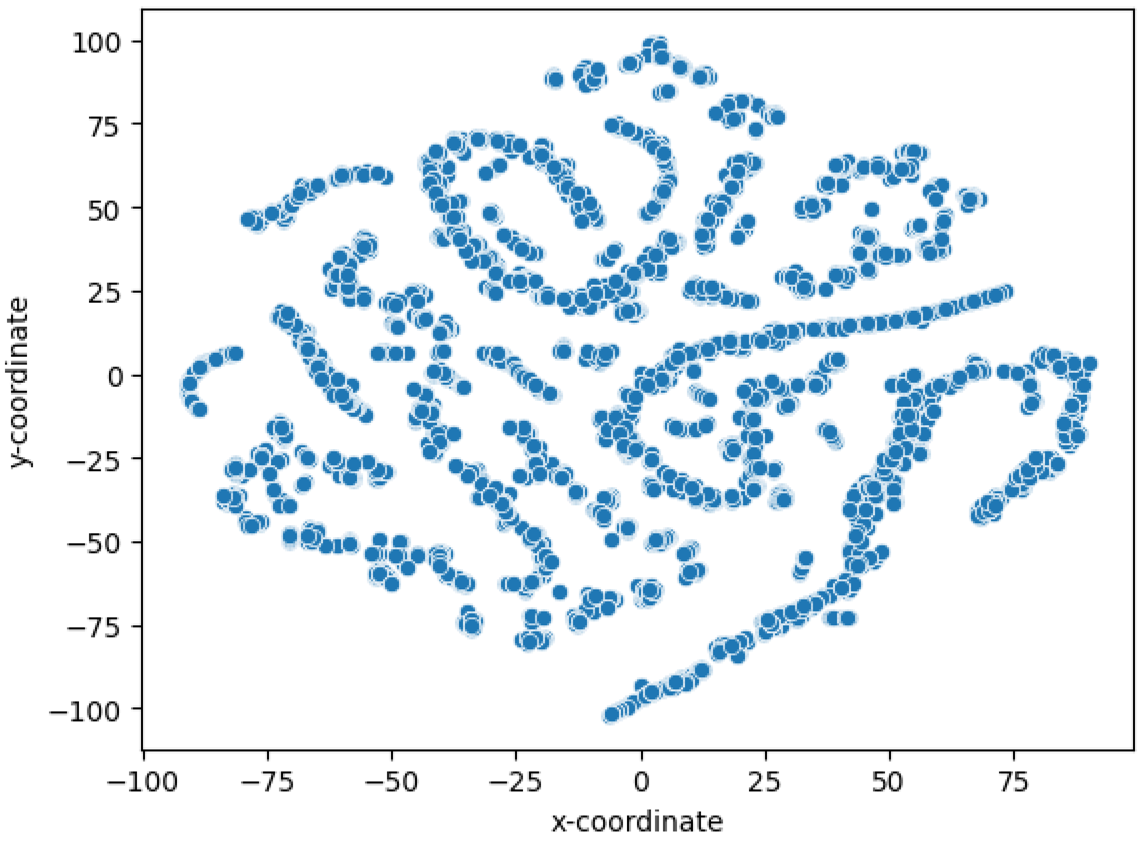
Insight 5: We can see that vehicles using fuel type E generally is more efficient than other fuels because over the range of fuel consumption values the increase in CO2 Emissions change at a smaller pace as compared to other fuel types (Slope is smaller for fuel vs co2 emissions graph).

Insight 6: Maximum no of outliers present in the data for COMPACT cars and cars with Z fuel types. This was found by their boxplots.

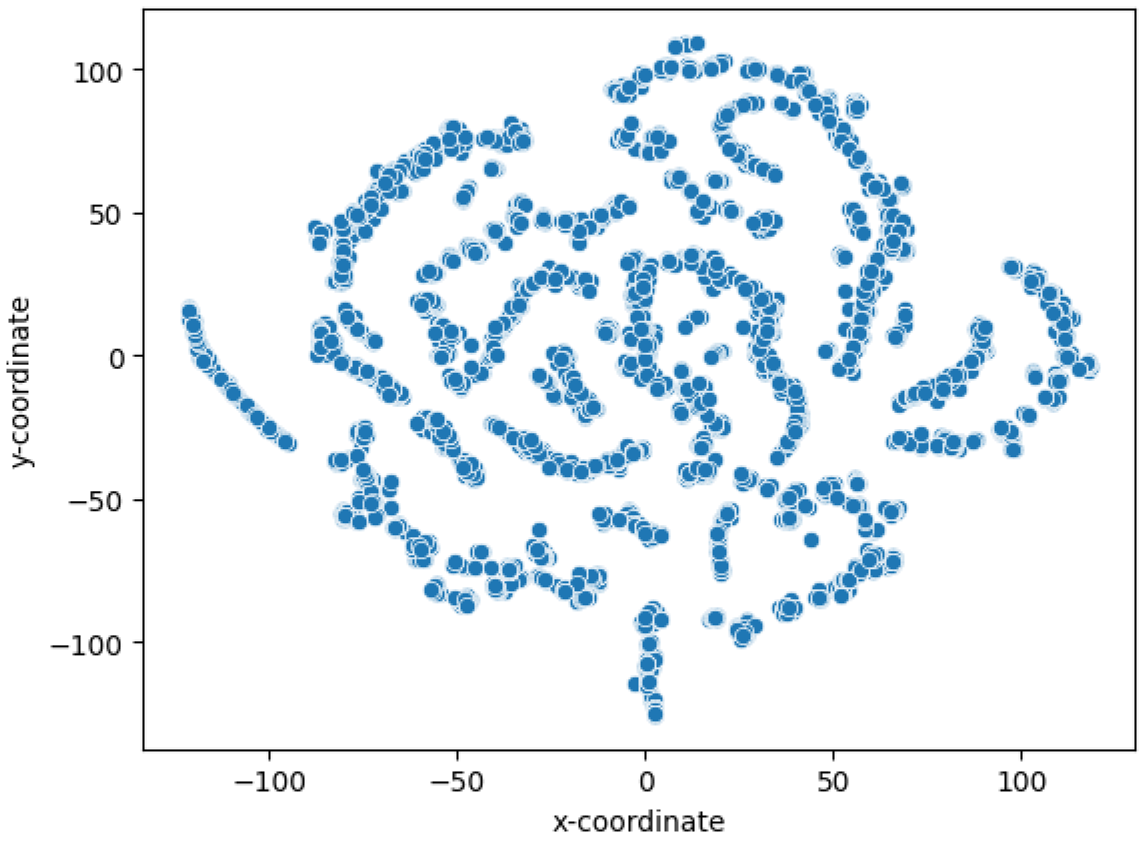
Insight 7: Through the histogram and the distribution plot of CO2 Emissions, we can see that the output labels (h(x)) is disproportionate. It is heavily imbalanced in the sense that most of the labels correspond to the first 25-30 classes in CO2 Emissions. Thus, an equal amount of test labels for each output value is not present in this data.

Insight 8: We can see from the histograms and displot() that the output label CO2 Emissions follows the normal distribution (approximately). Also, all fuel consumption features follow the normal distribution too.

(b)
TSNE scatterplot with n=1000 iterations:



TSNE scatterplot with n=2000 iterations:

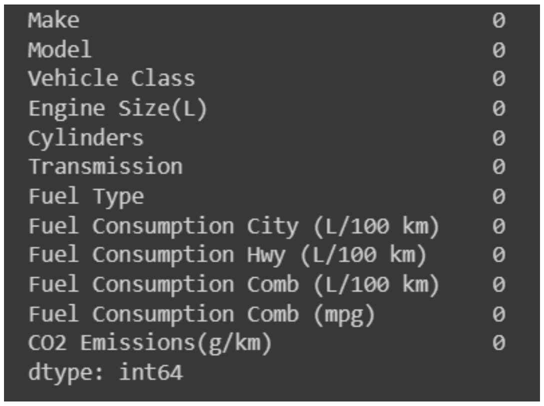


From these TSNE plots, we can see that the default parameters in the TSNE() function work well and the data looks quite seperable. If we double the no of iterations to 2000 for TSNE, we can see that the clusters are not very much different. Thus we can conclude that the data is saturated at 1000 iterations.

(c)

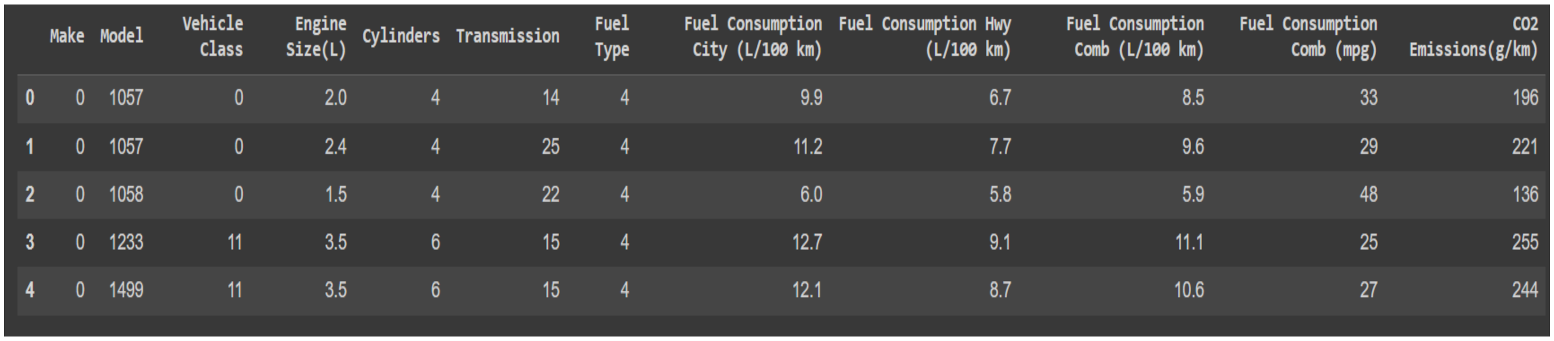
Preprocessing steps done -

- Checked for null/missing values in the dataset and found none



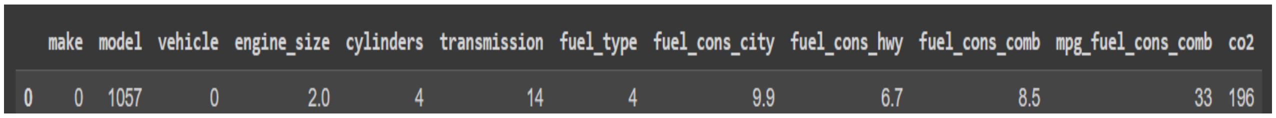
```
Make                                 0
Model                                0
Vehicle Class                        0
Engine Size(L)                       0
Cylinders                            0
Transmission                         0
Fuel Type                            0
Fuel Consumption City (L/100 km)     0
Fuel Consumption Hwy (L/100 km)      0
Fuel Consumption Comb (L/100 km)     0
Fuel Consumption Comb (mpg)          0
CO2 Emissions(g/km)                  0
dtype: int64
```

- Used Label Based Encoding for 5 categorical features present in the dataset



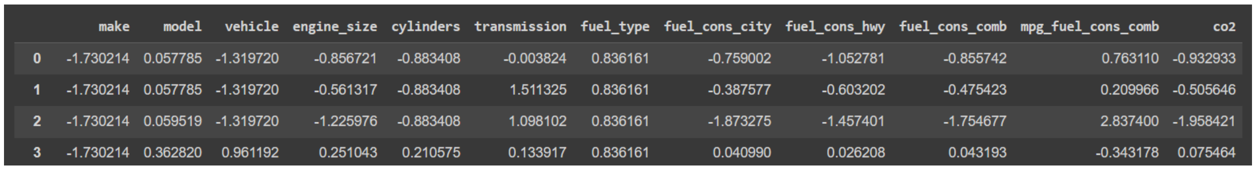
	Make	Model	Vehicle Class	Engine Size(L)	Cylinders	Transmission	Fuel Type	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	Fuel Consumption Comb (L/100 km)	Fuel Consumption Comb (mpg)	CO2 Emissions(g/km)
0	0	1057	0	2.0	4	14	4	9.9	6.7	8.5	33	196
1	0	1057	0	2.4	4	25	4	11.2	7.7	9.6	29	221
2	0	1058	0	1.5	4	22	4	6.0	5.8	5.9	48	136
3	0	1233	11	3.5	6	15	4	12.7	9.1	11.1	25	255
4	0	1499	11	3.5	6	15	4	12.1	8.7	10.6	27	244

- Preprocessed the headers to keep the header names simple and removed the units.



	make	model	vehicle	engine_size	cylinders	transmission	fuel_type	fuel_cons_city	fuel_cons_hwy	fuel_cons_comb	mpg_fuel_cons_comb	co2
0	0	1057	0	2.0	4	14	4	9.9	6.7	8.5	33	196

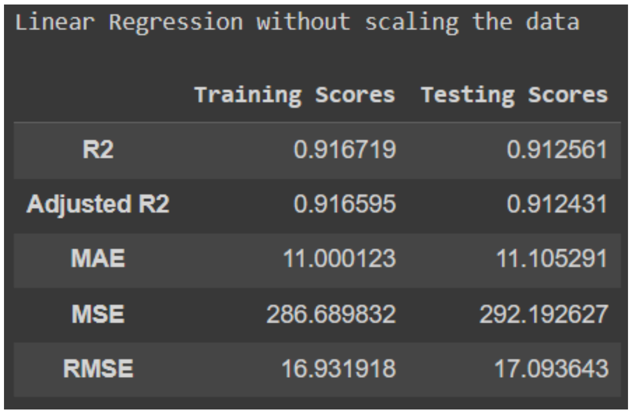
- Found out that some of the features have high values (in thousands) and some of the features have very low values (1-10). So, scaled the data using StandardSccaler()



	make	model	vehicle	engine_size	cylinders	transmission	fuel_type	fuel_cons_city	fuel_cons_hwy	fuel_cons_comb	mpg_fuel_cons_comb	co2
0	-1.730214	0.057785	-1.319720	-0.856721	-0.883408	-0.003824	0.836161	-0.759002	-1.052781	-0.855742	0.763110	-0.932933
1	-1.730214	0.057785	-1.319720	-0.561317	-0.883408	1.511325	0.836161	-0.387577	-0.603202	-0.475423	0.209966	-0.505646
2	-1.730214	0.059519	-1.319720	-1.225976	-0.883408	1.098102	0.836161	-1.873275	-1.457401	-1.754677	2.837400	-1.958421
3	-1.730214	0.362820	0.961192	0.251043	0.210575	0.133917	0.836161	0.040990	0.026208	0.043193	-0.343178	0.075464

Applied Linear regression on three datasets -

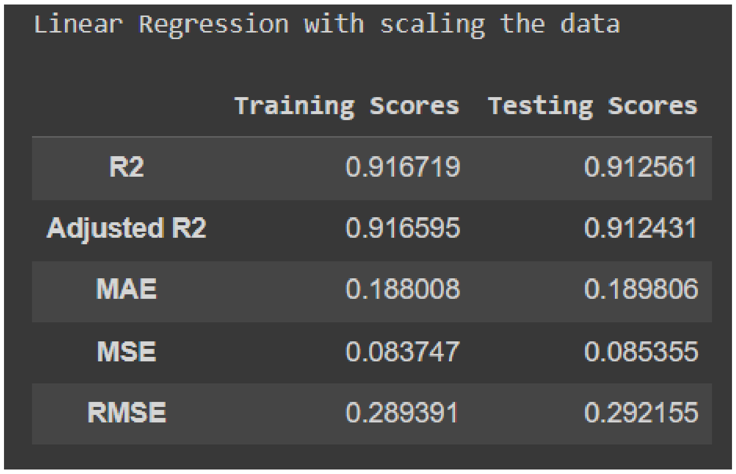
1) Dataset without scaling the attributes



```
Linear Regression without scaling the data
```

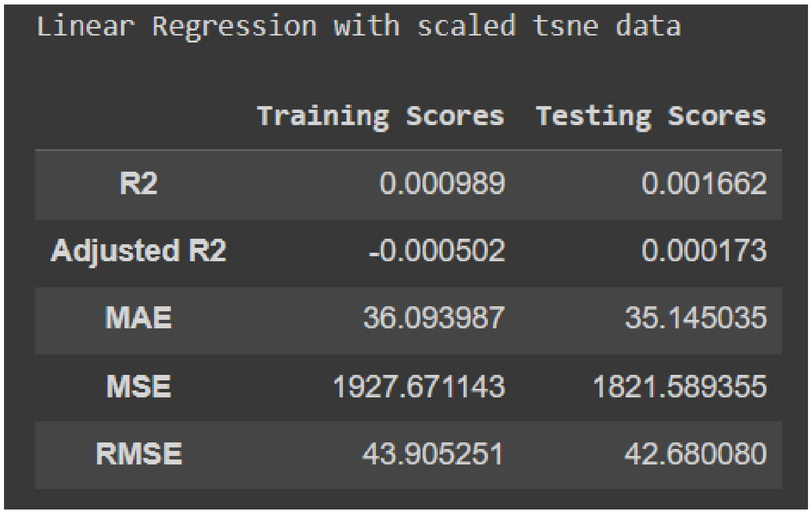
	Training Scores	Testing Scores
R2	0.916719	0.912561
Adjusted R2	0.916595	0.912431
MAE	11.000123	11.105291
MSE	286.689832	292.192627
RMSE	16.931918	17.093643

2) Dataset after scaling the attributes


Linear Regression with scaling the data

	Training Scores	Testing Scores
R2	0.916719	0.912561
Adjusted R2	0.916595	0.912431
MAE	0.188008	0.189806
MSE	0.083747	0.085355
RMSE	0.289391	0.292155

3) TSNE reduced and scaled dataset


Linear Regression with scaled tsne data

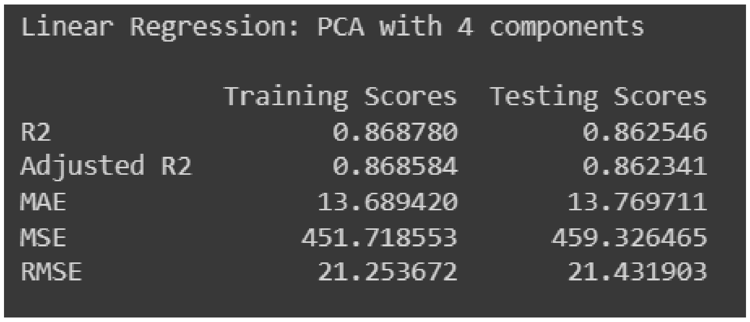
	Training Scores	Testing Scores
R2	0.000989	0.001662
Adjusted R2	-0.000502	0.000173
MAE	36.093987	35.145035
MSE	1927.671143	1821.589355
RMSE	43.905251	42.680080

From the above results, we can clearly see that the dataset that was scaled performed the best among all the three datasets. After that, the normal preprocessed dataset performed the best and the TSNE reduced and scaled dataset was the worst. This can be due to the fact that TSNE does not preserve the linear relationships between variables or maybe some important features were also discarded and hence the Linear Regression model is not performing well.

(d)

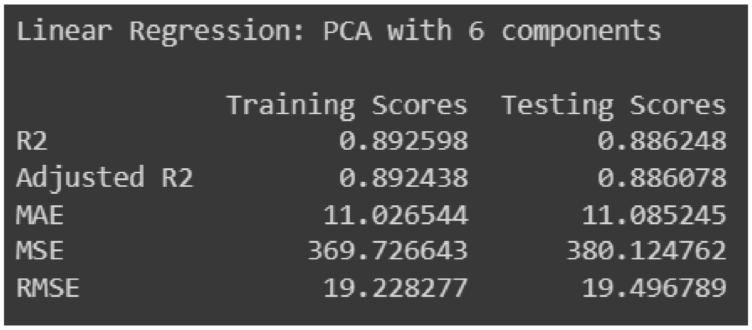
Results after applying PCA with n components -

1) n = 4



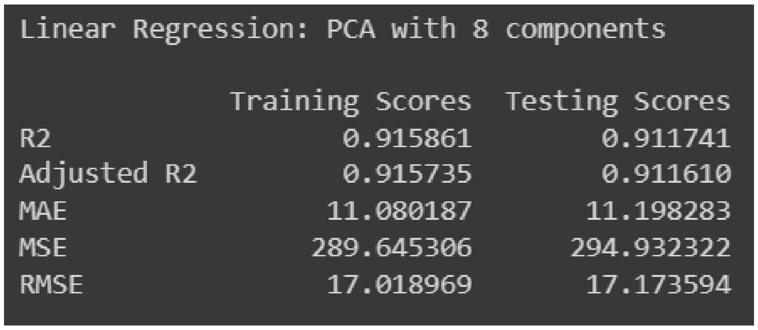
```
Linear Regression: PCA with 4 components

              Training Scores  Testing Scores
R2                   0.868780        0.862546
Adjusted R2          0.868584        0.862341
MAE                 13.689420       13.769711
MSE                451.718553      459.326465
RMSE                21.253672       21.431903
```

2) n = 6



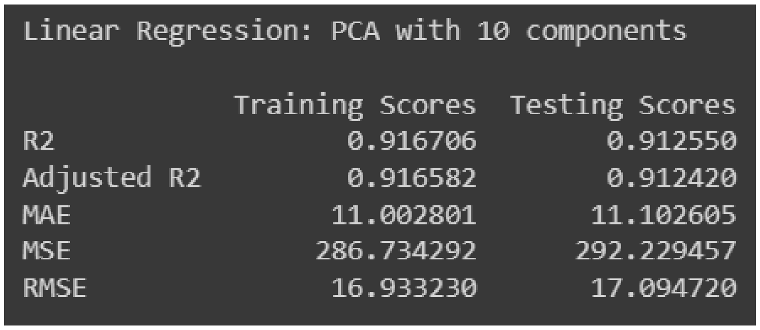
```
Linear Regression: PCA with 6 components

              Training Scores  Testing Scores
R2                   0.892598        0.886248
Adjusted R2          0.892438        0.886078
MAE                 11.026544       11.085245
MSE                369.726643      380.124762
RMSE                19.228277       19.496789
```

3) n = 8



```
Linear Regression: PCA with 8 components

              Training Scores  Testing Scores
R2                   0.915861        0.911741
Adjusted R2          0.915735        0.911610
MAE                 11.080187       11.198283
MSE                289.645306      294.932322
RMSE                17.018969       17.173594
```

4) n = 10

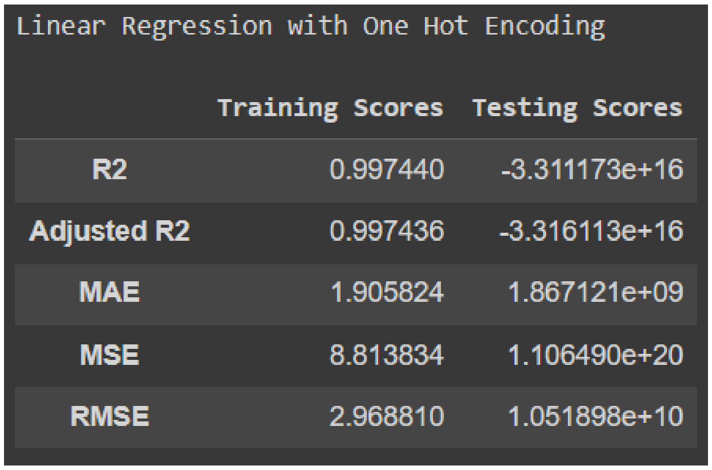


```
Linear Regression: PCA with 10 components

              Training Scores  Testing Scores
R2                   0.916706        0.912550
Adjusted R2          0.916582        0.912420
MAE                 11.002801       11.102605
MSE                286.734292      292.229457
RMSE                16.933230       17.094720
```

We can see that PCA performs best with 8+ components. Between 8 and 10 components, there isn't much variation in the results of the Linear Regression model thus, we can say that 8 components are enough for the Linear Regression model instead of the complete 11 components. The results obtained by it are also very much consistent with the results obtained in part (c) with all the components, without scaling the data. Note that here also the non-scaled version of the dataset is used.

(e)
The number of columns drastically increases to 2150 (from 12).

After applying one-hot-encoding instead of label encoding we get the following results -



```
Linear Regression with One Hot Encoding
```

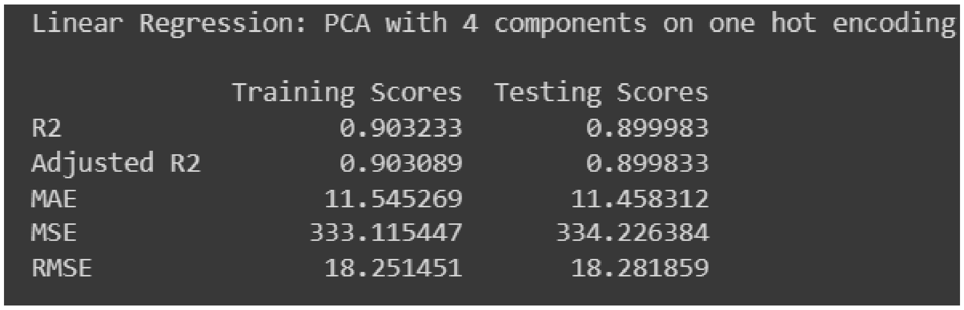
	Training Scores	Testing Scores
R2	0.997440	-3.311173e+16
Adjusted R2	0.997436	-3.316113e+16
MAE	1.905824	1.867121e+09
MSE	8.813834	1.106490e+20
RMSE	2.968810	1.051898e+10

Over here, we can clearly see that the model has overfitted the data. The training scores are good but the testing scores are very poor as compared to part (c) without scaling the dataset. This is because one-hot-encoding has overfitted the data. Due to the increased dimensionality, multicollinearity, and noise in the dataset, one-hot-encoding almost memorizes the training dataset and thus gives a very bad performance on the testing dataset.

Thus Label Based Encoding was much better than One-Hot-Encoding.

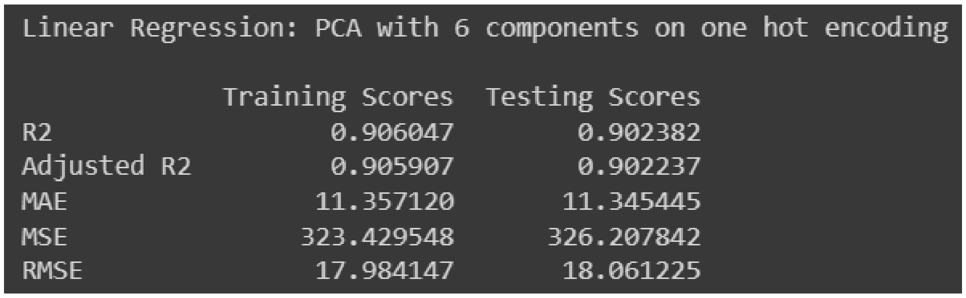
(f)
Results after applying PCA with n components on One-Hot-Encoding dataset -

1) n = 4



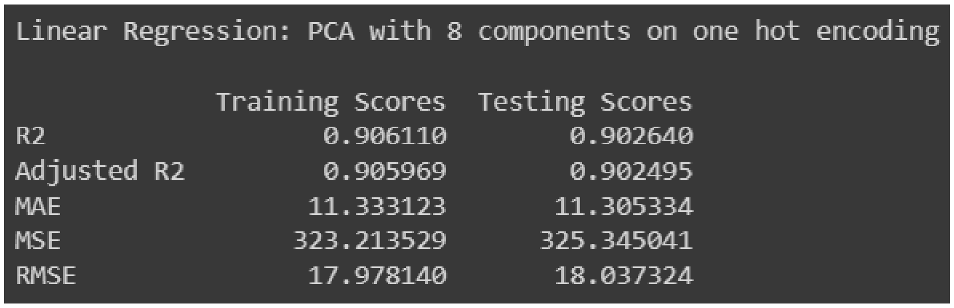
```
Linear Regression: PCA with 4 components on one hot encoding

             Training Scores  Testing Scores
R2                  0.903233        0.899983
Adjusted R2         0.903089        0.899833
MAE                11.545269       11.458312
MSE               333.115447      334.226384
RMSE               18.251451       18.281859
```

2) n = 6



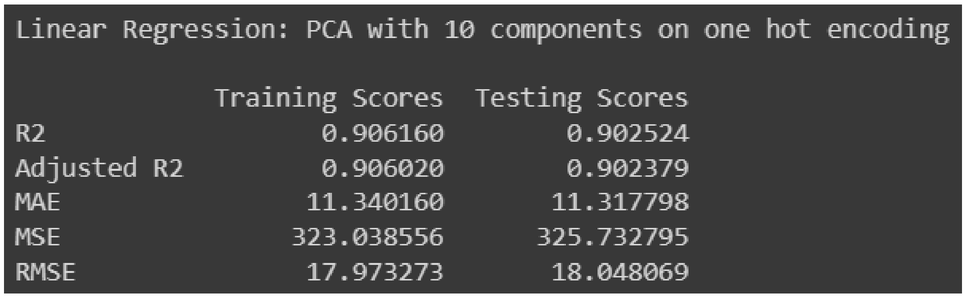
```
Linear Regression: PCA with 6 components on one hot encoding

             Training Scores  Testing Scores
R2                  0.906047        0.902382
Adjusted R2         0.905907        0.902237
MAE                11.357120       11.345445
MSE               323.429548      326.207842
RMSE               17.984147       18.061225
```

3) n = 8



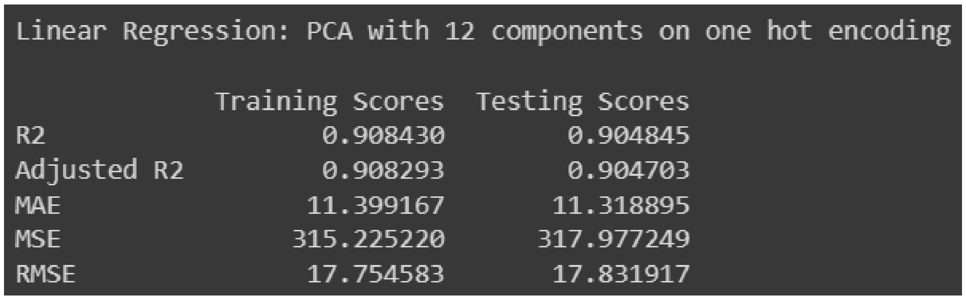
```
Linear Regression: PCA with 8 components on one hot encoding

             Training Scores  Testing Scores
R2                  0.906110        0.902640
Adjusted R2         0.905969        0.902495
MAE                11.333123       11.305334
MSE               323.213529      325.345041
RMSE               17.978140       18.037324
```

4) n = 10



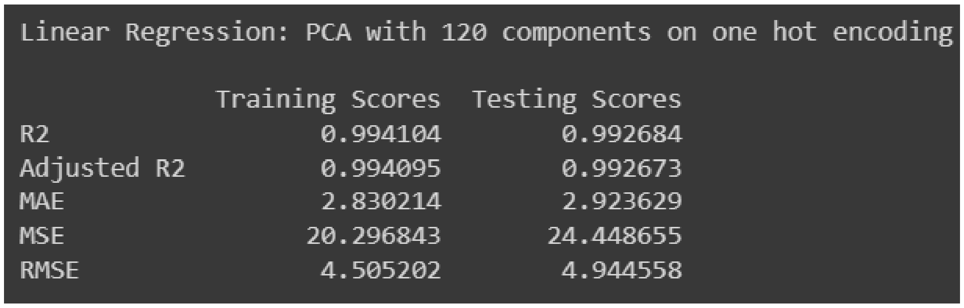
```
Linear Regression: PCA with 10 components on one hot encoding

             Training Scores  Testing Scores
R2                  0.906160        0.902524
Adjusted R2         0.906020        0.902379
MAE                11.340160       11.317798
MSE               323.038556      325.732795
RMSE               17.973273       18.048069
```

5) n = 12



```
Linear Regression: PCA with 12 components on one hot encoding

             Training Scores  Testing Scores
R2                  0.908430        0.904845
Adjusted R2         0.908293        0.904703
MAE                11.399167       11.318895
MSE               315.225220      317.977249
RMSE               17.754583       17.831917
```

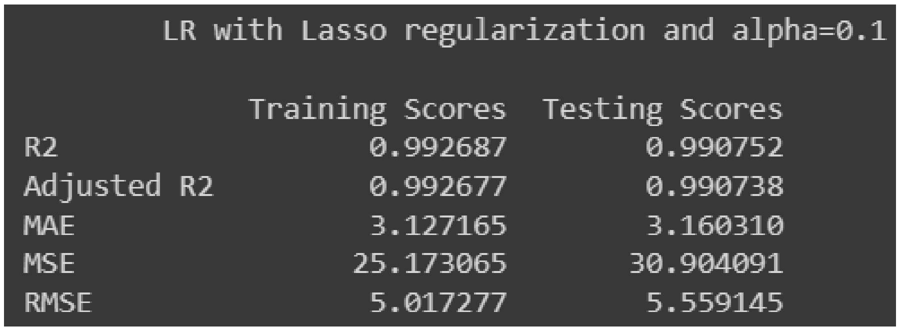
6) n = 120



```
Linear Regression: PCA with 120 components on one hot encoding

             Training Scores  Testing Scores
R2                  0.994104        0.992684
Adjusted R2         0.994095        0.992673
MAE                 2.830214        2.923629
MSE                20.296843       24.448655
RMSE                4.505202        4.944558
```

We can clearly notice that when we vary n by a small amount (4-12), then there is not much change in the performance of Linear Regression model. However, when we increase the number of components drastically (n=120) the model becomes better (MSE: 20). This is probably due to the fact that when n is between 4 to 12 PCA is selecting the features that were originated from one hot encoding too. However, when n is large (n=120) features selected consist of many actual features that PCA was selecting in part (d) plus some extra features which were originated from one hot encoding too. Thus the loss dramatically decreases. However, this can also be attributed to the shuffling of the data before train-test-split too to create an unfavorable/favorable split. Also, it can be that the original dataset might have complex relationships that require a higher-dimensional representation to capture effectively, and thus the Linear Regression model benefits from this increased dimensionality.

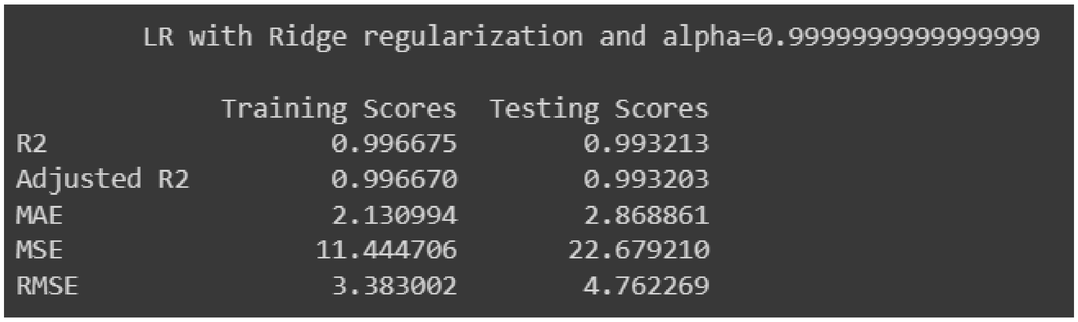
(g)
I performed **Lasso Regularization** on multiple values of lambda ranging from 0.1 to 2 and found out that the LR model performed best with lambda = 0.1.



```
        LR with Lasso regularization and alpha=0.1

             Training Scores  Testing Scores
R2                 0.992687        0.990752
Adjusted R2        0.992677        0.990738
MAE                3.127165        3.160310
MSE               25.173065       30.904091
RMSE               5.017277        5.559145
```

As I increased the values of lambda the MSE, RMSE, etc started increasing thus indicating worser performance. When compared to part (c) the model performs much better with regularization parameters enabled than without it. Regularization helps to distribute the weights of the parameters in a better way by adding a penalty term to the loss function.

I then performed **Ridge Regularization** on multiple values of lambda ranging from 0.1 to 2 and found out that the LR model performed best with lambda = 1



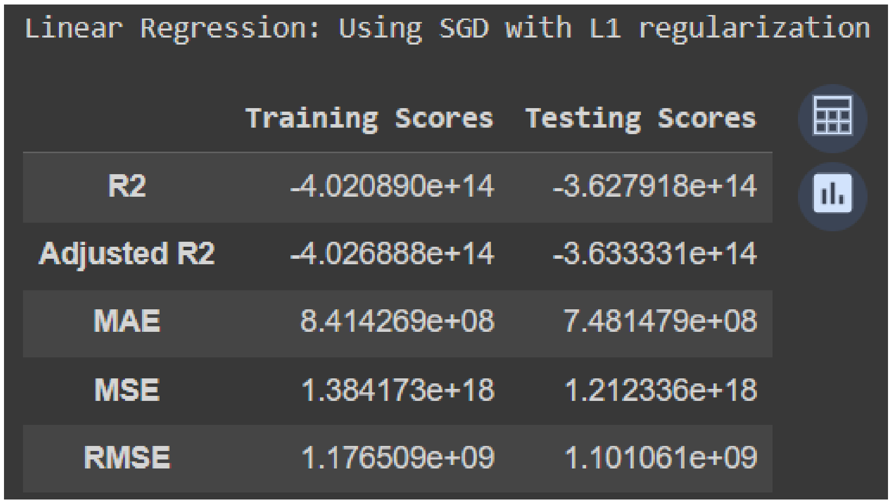
```
        LR with Ridge regularization and alpha=0.9999999999999999

             Training Scores  Testing Scores
R2                 0.996675        0.993213
Adjusted R2        0.996670        0.993203
MAE                2.130994        2.868861
MSE               11.444706       22.679210
RMSE               3.383002        4.762269
```

Here, we can see that on comparing with part (c) the linear regression model with regularization parameters enabled are performing much better than with the regularization parameters. (The dataset for both of them is unscaled). The errors MAE, MSE, RMSE obtained are way less than before. Also, on comparing L1 and L2 regularization, we can see that the L2 regularization performs much better than L1 regularization, and the error obtained is much smaller than L1.

(h)

On using SGDRegressor library we can see that the model is highly underfitting the data irrespective of whether L1 or L2 regularization applied or not.

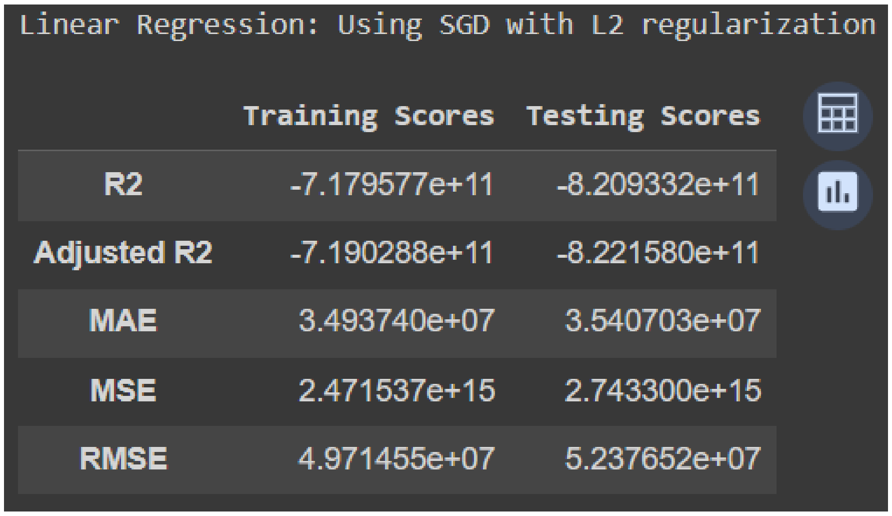
Using SGDRegressor Library with L1 regularization -



```
Linear Regression: Using SGD with L1 regularization
```

	Training Scores	Testing Scores
R2	-4.020890e+14	-3.627918e+14
Adjusted R2	-4.026888e+14	-3.633331e+14
MAE	8.414269e+08	7.481479e+08
MSE	1.384173e+18	1.212336e+18
RMSE	1.176509e+09	1.101061e+09

Using SGDRegressor Library with L2 regularization -



```
Linear Regression: Using SGD with L2 regularization
```

	Training Scores	Testing Scores
R2	-7.179577e+11	-8.209332e+11
Adjusted R2	-7.190288e+11	-8.221580e+11
MAE	3.493740e+07	3.540703e+07
MSE	2.471537e+15	2.743300e+15
RMSE	4.971455e+07	5.237652e+07

When compared with each other we can see that L2 regularization performs much better than L1 regularization but overall the training and testing scores of both the models are pretty low (very high errors) as compared to part (c).