Name: Arlene Antony D'costa
Seating Pin: 79
SUID: 594303899

CIS 662 HW5 Report: Regression (Both Linear and Logistic

PART 1: Linear Regression and Comparison with NN:

Dataset selected is **71-80.csv**

Data has also been split in the same way as the previous homework 80:20 to compare the results.

Linear regression model has certain **assumptions such as:**

- Linear relationship
- Multivariate normality
- No or little multicollinearity
- No auto-correlation
- Homoscedasticity

The assumption and requirement of the model is that the data should be normally distributed. Hence, I have applied **MinMaxScaler** as it normalised the data to a common range. Also, since the unit of all the values is common, I wanted to preserve the original range of the values.

Linear Regression uses MSE for loss computation by default.

I have used **MSE**, **MAE** and **MAPE** for analysis and better understanding of the model fit to the data.

Also, the corresponding outputs for both Linear Regression and Neural Network are attached for better analysis.

Formulas for the above can be found in the code file.

OUTPUT:

Model:

```
model = LinearRegression()
model.fit(X_train_scaled, y_train_scaled)
```

```
✓ 0.0s
```

```
▾ LinearRegression
LinearRegression()
```

Name: Arlene Antony D'costa
Seating Pin: 79
SUID: 594303899

MSE for Linear Regression:

Calculate MSE

+ Code + Markdown

```
# Calculate squared differences
squared_errors = (y_actual - y_pred) ** 2

    # Calculate the mean of squared errors
mse = np.mean(squared_errors)
# Print the MSE
print(f"Mean Squared Error (MSE): {squared_errors}")
```

✓ 0.0s

```
Mean Squared Error (MSE): [7.30643031e+02 1.56619585e+01 1.08785459e+04 5.18030883e+02
 1.14454176e+03 1.78096118e+01 5.27974648e+03 5.89735145e+00
 2.01051712e+03 1.45040148e+02 8.84814609e+03 2.85402932e+02
 6.04755662e+03 1.45774442e+02 1.20345361e+01 3.02496567e+03
 6.24869138e+02 1.99230315e+02 8.43175719e+02 3.61256219e+03]
```

```
mse
```

✓ 0.0s

```
2219.507594917693
```

MAE and MAPE for Linear Regression:

```
print(f"MAE for the above dataset using Linear Regression Model is {mae}")
```

✓ 0.0s

```
MAE for the above dataset using Linear Regression Model is 35.779670348067704
```

Calculate Mean Absolute Percentage Error

```
mape = np.mean(np.abs((y_pred.flatten() - y_actual.flatten())/y_actual.flatten())) * 100
```

✓ 0.0s

```
print(f"MAPE for the above dataset using Linear Regression Model is {mape}")
```

✓ 0.0s

```
MAPE for the above dataset using Linear Regression Model is 27.72027163267112
```

Name: Arlene Antony D'costa
Seating Pin: 79
SUID: 594303899

Screenshot for Linear Regression output:

	cit_2017	cit_2018	cit_2019	cit_2020	cit_2021	cit_2022_actual	cit_2022_pred	absolute_error	abs_by_act
0	77.0	71.0	91.0	96.0	139.0	163.0	135.969591	27.030409	0.165831
1	4.0	2.0	3.0	7.0	4.0	3.0	6.957519	3.957519	1.319173
2	840.0	699.0	668.0	576.0	631.0	613.0	508.699732	104.300268	0.170147
3	76.0	77.0	96.0	161.0	138.0	164.0	141.239708	22.760292	0.138782
4	36.0	42.0	67.0	34.0	45.0	62.0	28.168923	33.831077	0.545663
5	6.0	5.0	37.0	41.0	39.0	39.0	34.779856	4.220144	0.108209
6	62.0	93.0	129.0	135.0	159.0	221.0	148.338136	72.661864	0.328787
7	156.0	183.0	191.0	166.0	267.0	256.0	253.571554	2.428446	0.009486
8	99.0	83.0	79.0	78.0	57.0	88.0	43.161210	44.838790	0.509532
9	442.0	443.0	404.0	355.0	385.0	333.0	320.956738	12.043262	0.036166
10	67.0	163.0	255.0	384.0	553.0	683.0	588.935415	94.064585	0.137723
11	246.0	212.0	201.0	146.0	134.0	103.0	86.106127	16.893873	0.164018
12	38.0	110.0	184.0	334.0	532.0	668.0	590.233962	77.766038	0.116416
13	4.0	2.0	10.0	6.0	14.0	27.0	14.926291	12.073709	0.447174
14	276.0	281.0	321.0	312.0	371.0	328.0	331.469083	3.469083	0.010576
15	44.0	56.0	53.0	74.0	60.0	115.0	60.000312	54.999688	0.478258
16	19.0	26.0	28.0	22.0	21.0	43.0	18.002617	24.997383	0.581334
17	229.0	235.0	262.0	273.0	296.0	280.0	265.885103	14.114897	0.050410
18	741.0	885.0	1030.0	986.0	996.0	861.0	831.962512	29.037488	0.033725
19	427.0	469.0	435.0	344.0	338.0	312.0	251.895406	60.104594	0.192643

Screenshot for NN output:

	y_pred	y_act	absolute_error	sbs_by_act
0	144.929120	163.0	18.070880	0.110864
1	21.023752	3.0	18.023752	6.007917
2	597.744570	613.0	15.255430	0.024887
3	176.276280	164.0	12.276280	0.074855
4	54.530724	62.0	7.469276	0.120472
5	59.818886	39.0	20.818886	0.533818
6	181.366090	221.0	39.633910	0.179339
7	289.453160	256.0	33.453160	0.130676
8	72.339610	88.0	15.660390	0.177959
9	375.314060	333.0	42.314060	0.127069
10	672.731600	683.0	10.268400	0.015034
11	127.656044	103.0	24.656044	0.239379
12	686.652400	668.0	18.652400	0.027923
13	27.225061	27.0	0.225061	0.008336
14	384.565280	328.0	56.565280	0.172455
15	86.264440	115.0	28.735560	0.249874
16	37.402290	43.0	5.597710	0.130179
17	310.873140	280.0	30.873140	0.110261
18	878.681800	861.0	17.681800	0.020536
19	284.623440	312.0	27.376560	0.087745

Comparison with NN:

MSE:

Neural Network Model	Linear Regression
669.15	2219.51

MAE:

Neural Network Model	**Linear Regression**
22.18	35.78

MAPE:

Neural Network Model	**Linear Regression**
42.74	27.72

CONCLUSION:

As attached in the screenshots above, NN outperforms Linear Regression as linear Regression is a simple model and can draw a simple best fit line between the data points. However, it cannot capture complex patterns within data as a neural network that contains hidden neurons.

MSE and MAE is lower in NN, but MAPE is higher in neural networks as they capture high **variance** in data and are susceptible to capturing outliers.

1	21.023752	3.0	18.023752	6.007917
2	597.744570	613.0	15.255430	0.024887

Here, we can see the predicted value is 21 but the actual value is 3, and the abs value is 18. Linear Regression doesn’t capture such variance and patterns as they can capture high bias and low variance in data, hence they do not capture outlier patterns and thus, MAE and MSE are higher in Linear Regression models.

PART 2: Logistic Regression and its comparison with NN:

Like Linear Regression, I have used normalization to preserve the original characteristics and values as all the feature values are of the same scale.

As the data doesn’t contain categorical values, as done in HW5, I have used binning to categorize the ratio between columns 2022 and 2021 into 3 distinct categories such as {Low, Medium, High}.

Citations columns cit_2017 to cit_2022 are considered as feature columns (X features) and column **category** is considered as Y column. (Column to be classified)

Since we are doing multi-label classification, I enabled the multi-class feature available for Logistic Regression.

Also, I have used a classification report and AUC-ROC diagram to compare the results of both logistic regression and neural networks.

Name: Arlene Antony D’costa
Seating Pin: 79
SUID: 594303899

OUTPUT:

Logistic Regression Model:

```
model = LogisticRegression(multi_class='multinomial', solver='lbfgs')
model.fit(X_train_resampled, y_train_resampled)
```

✓ 0.0s

```
LogisticRegression
LogisticRegression(multi_class='multinomial')
```

Screenshot for Logistic Regression classification results:

	cit_2017	cit_2018	cit_2019	cit_2020	cit_2021	cit_2022	ratio_21_22	category	category_pred
70	0.104938	0.104498	0.158426	0.186693	0.251553	0.308532	1.24	High	Low
74	0.292665	0.222185	0.188022	0.140504	0.138026	0.132765	0.97	Low	Low
2	0.305011	0.236388	0.231546	0.184109	0.216356	0.208191	0.97	Low	Low
44	0.228758	0.221170	0.274373	0.267765	0.265355	0.240956	0.92	Low	Low
56	0.029775	0.036185	0.019150	0.024871	0.027260	0.031058	1.13	Medium	Medium
48	0.049746	0.063578	0.067549	0.072351	0.106280	0.151195	1.43	High	High
12	0.062092	0.066622	0.091226	0.094315	0.105590	0.097611	0.93	Low	Low
36	0.057008	0.049713	0.052228	0.032300	0.048654	0.041297	0.86	Low	Medium
0	0.037763	0.033480	0.035167	0.027132	0.022774	0.021160	0.93	Low	Medium
49	0.963689	0.918160	0.933844	0.902455	0.924086	0.809215	0.89	Low	Low
76	0.165214	0.158945	0.155641	0.132429	0.105935	0.116724	1.11	Medium	Low
93	0.002179	0.001691	0.011838	0.011305	0.012077	0.012287	1.00	Low	Medium
88	0.004720	0.001691	0.001741	0.000646	0.002761	0.002048	0.75	Low	Medium
92	0.089325	0.071694	0.068942	0.045220	0.044859	0.034130	0.77	Low	Medium
5	0.006173	0.008793	0.003482	0.000000	0.005176	0.013311	2.21	High	Medium
77	0.169572	0.144403	0.135794	0.133721	0.114562	0.090444	0.80	Low	Low
23	0.028322	0.036524	0.037256	0.042313	0.079710	0.090785	1.14	Medium	High
63	0.337691	0.376733	0.429318	0.373708	0.407177	0.359386	0.89	Low	Low
87	0.832607	0.760568	0.762535	0.714470	0.741201	0.713652	0.97	Low	Low
38	0.024328	0.029084	0.034819	0.032623	0.051415	0.034130	0.67	Low	Medium

Name: Arlene Antony D'costa
Seating Pin: 79
SUID: 594303899

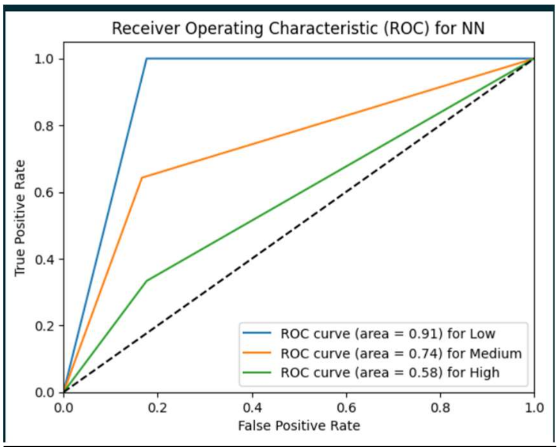
NN Classification Report Output:

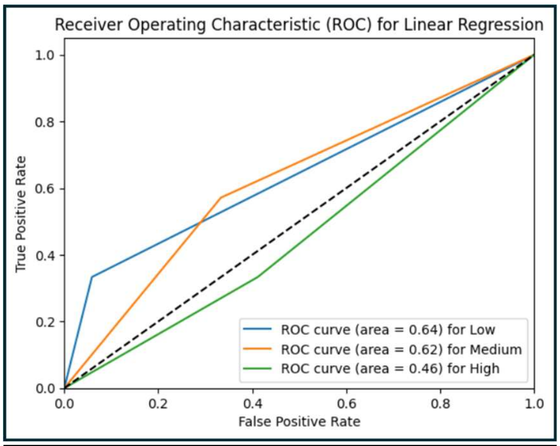
```
              precision    recall  f1-score   support

           0       0.50      1.00      0.67         3
           1       0.90      0.64      0.75        14
           2       0.25      0.33      0.29         3

   micro avg       0.65      0.65      0.65        20
   macro avg       0.55      0.66      0.57        20
weighted avg       0.74      0.65      0.67        20
 samples avg       0.65      0.65      0.65        20
```

Logistic Regression Classification Report Output:

```
classification_rep = classification_report(y_test_scalec
print(classification_rep)
✓ 0.0s

              precision    recall  f1-score   support

        High       0.50      0.33      0.40         3
         Low       0.80      0.57      0.67        14
      Medium       0.12      0.33      0.18         3

    accuracy                           0.50        20
   macro avg       0.48      0.41      0.42        20
weighted avg       0.65      0.50      0.55        20
```

ROC-AUC of NN verses Logistic Regression





CONCLUSION:

As attached in the above screenshots, we can observe that Neural Networks perform better and provide better classification results (F1 score) than Logistic Regression.

Categorizing continuous numerical data using logic of obtaining ratio between 2022/2021 can lead to add a bit of **non-linearity** within the data which **is captured better by neural networks**. Also, neural networks use softmax as an activation function, however for **logistic Regression sigmoid activation function** is used where each output can be interpreted as the probability of the input belonging to a particular class and the most probable value is considered.

As you can see in the screenshots for ROC-AUC for both NN and Logistic Regression, NN performs better with a **higher value AUC value** for all the 3 classes as opposed to that of Logistic Regression.