HW5: Error Based Model Evaluation

Chapter 7

2. You have been hired by the European Space Agency to build a model that predicts the amount of oxygen that an astronaut consumes when performing five minutes of intense physical work. The descriptive features for the model will be the age of the astronaut and their average heart rate throughout the work. The regression model is
OXYCON = w[0] + w[1] × AGE + w[2] × HEARTRATE
The table below shows a historical dataset that has been collected for this task.

ID	OXYCON	AGE	HEART RATE
1	37.99	41	138
2	47.34	42	153
3	44.38	37	151
4	28.17	46	133
5	27.07	48	126
6	37.85	44	145
7	44.72	43	158
8	36.42	46	143
9	31.21	37	138
10	54.85	38	158
11	39.84	43	143
12	30.83	43	138

a. Assuming that the current weights in a multivariate linear regression model are w[0] = −59.50, w[1] = −0.15, and w[2] = 0.60, make a prediction for each training instance using this model.

Ans: with reference to weight given above, the equation will be:
Y = W0+ W1*Age + W2*HeartRate

The table below shows the prediction made using the above equation

ID	OXYCON	AGE	HEART RATE	PREDICTION
1	37.99	41	138	17.15
2	47.34	42	153	26
3	44.38	37	151	25.55
4	28.17	46	133	13.4
5	27.07	48	126	8.9
6	37.85	44	145	20.9
7	44.72	43	158	28.85
8	36.42	46	143	19.4
9	31.21	37	138	17.75
10	54.85	38	158	29.6
11	39.84	43	143	19.85
12	30.83	43	138	16.85

b. Calculate the sum of squared errors for the set of predictions generated in part (a).

Ans: Formula to calculate sum of squared errors is given below:
SSE= SUM((actual value y - predicted value yhat)^2)/2

ID	OXYCON	Prediction	Error	Squared Error	Error Delta W0= Error*weight 0	Error Delta W1= Error*weight 1	Error Delta W2 = Error*weight 3
1	37.99	17.15	20.84	434.3056	20.84	854.44	2875.92
2	47.34	26	21.34	455.3956	21.34	896.28	3265.02
3	44.38	25.55	18.83	354.5689	18.83	696.71	2843.33
4	28.17	13.4	14.77	218.1529	14.77	679.42	1964.41
5	27.07	8.9	18.17	330.1489	18.17	872.16	2289.42
6	37.85	20.9	16.95	287.3025	16.95	745.8	2457.75
7	44.72	28.85	15.87	251.8569	15.87	682.41	2507.46
8	36.42	19.4	17.02	289.6804	17.02	782.92	2433.86
9	31.21	17.75	13.46	181.1716	13.46	498.02	1857.48
10	54.85	29.6	25.25	637.5625	25.25	959.5	3989.5

11	39.84	19.85	19.99	399.6001	19.99	859.57	2858.57
12	30.83	16.85	13.98	195.4404	13.98	601.14	1929.24
				Sum= 4035.1863	**Sum= 216.47**	**Sum= 9128.37**	**Sum= 31271.96**

Sum of Squared Error = **4035.1863/2 = 2017.593**

c. Assuming a learning rate of 0.000002, calculate the weights at the next iteration of the gradient descent algorithm.

Ans: weights are updated as given below:
New weight = old weight + (learning rate * error delta)

So, for W0:
W[0]= W[0] + alpha * error delta (D,W[0])
W[0]= - 59.50 + (0. 000002 *216.47) **= -59.4996**

So, for W1:
W[1]= W[1] + alpha * error delta (D,W[1])
W[1]= - 0.15 + (0. 000002 *9128.37) = **-0.1317**

So, for W2:
W[2]= W[2] + alpha * error delta (D,W[2])
W[2]= 0.60 + (0. 000002 *31273.96) **= 0.6625**

d. Calculate the sum of squared errors for a set of predictions generated using the new set of weights calculated in part (c).

Ans: By using the new weight, the equation is :
Y = -59.4996+ (-0.1317 *Age) + (0.6625 *HeartRate)

Formula to calculate sum of squared errors is given below:
SSE= SUM((actual value y - predicted value yhat)^2)/2

Table below shows the prediction with new weights

ID	OXYCON	AGE	HEART RATE	PREDICTION	Error	Squared Error
1	37.99	41	138	26.5257	11.4643	131.430174

2	47.34	42	153	36.3315	11.0085	121.187072
3	44.38	37	151	35.665	8.715	75.951225
4	28.17	46	133	22.5547	5.6153	31.5315941
5	27.07	48	126	17.6538	9.4162	88.6648224
6	37.85	44	145	30.7681	7.0819	50.1533076
7	44.72	43	158	39.5123	5.2077	27.1201393
8	36.42	46	143	29.1797	7.2403	52.4219441
9	31.21	37	138	27.0525	4.1575	17.2848063
10	54.85	38	158	40.1708	14.6792	215.478913
11	39.84	43	143	29.5748	10.2652	105.374331
12	30.83	43	138	26.2623	4.5677	20.8638833
						SUM= 937.462

Sum of squared error = sum/2 = **468.731**

3. A multivariate logistic regression model has been built to predict the propensity of shoppers to perform a repeat purchase of a free gift that they are given. The descriptive features used by the model are the age of the customer, the socio-economic band to which the customer belongs (a, b, or c), the average amount of money the customer spends on each visit to the shop, and the average number of visits the customer makes to the shop per week. This model is being used by the marketing department to determine who should be given the free gift. The weights in the trained model are shown in the table below.

Feature	Weight
Intercept (w[0])	-3.82398
AGE	-0.02990
SOCIO ECONOMIC BAND B	-0.09089
SOCIO ECONOMIC BAND C	-0.19558
SHOP VALUE	0.02999
SHOP FREQUENCY	0.74572

Use this model to make predictions for each of the following query instances.

ID	AGE	SOCIO ECONOMIC BAND	SHOP FREQUENCY	SHOP VALUE
1	56	b	1.60	109.32
2	21	c	4.92	11.28
3	48	b	1.21	161.19
4	37	c	0.72	170.65
5	32	a	1.08	165.39

Ans: For this multivariate logistic regression model, threshold is set at 0.5. if the threshold is greater than equal to 0.5 then shoppers will be given free gift. However, in this model, socio economic band is categorical features. Hence this feature is one hot encoded and category level 'a' is dropped. So whenever, socio economic band B and socio economic band C is 0 then it refers to band A.
The equation to determine likelihood to receive free gift is given below:

$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = logistic(\mathbf{w} \cdot \mathbf{d}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{d}}}$$

For ID = 1:
W.D= (-3.82398 + (-0.02990*56) + (-0.09089 *1) + (-0.19558 *0) + (0.02999*109.32) + (0.74572*1.6) = -1.12

Likelihood= 1/ $(1+e^{-(-1.12)})$= 0.25
Hence, the likelihood to give free gift is less than 0.5 so prediction = No

For ID = 2:
W.D= (-3.82398 + (-0.02990*21) + (-0.09089 *0) + (-0.19558 *1) + (0.02999*11.28) + (0.74572*4.92) = -0.64

Likelihood= 1/ $(1+e^{-(-0.64)})$= 0.35
Hence, the likelihood to give free gift is less than 0.5 so prediction = No

For ID = 3:
W.D= (-3.82398 + (-0.02990*48) + (-0.09089 *1) + (-0.19558 *0) + (0.02999*161.19) + (0.74572*1.21) = 0.39

Likelihood= 1/ $(1+e^{-0.39})$= 0.60
Hence, the likelihood to give free gift is greater than 0.5 so prediction = Yes

For ID = 4:
W.D= (-3.82398 + (-0.02990*37) + (-0.09089 *0) + (-0.19558 *1) + (0.02999*170.65) + (0.74572*0.72) = 0.53

Likelihood= 1/ $(1+e^{-0.53})$= 0.63
Hence, the likelihood to give free gift is greater than 0.5 so prediction = Yes

For ID = 5:
W.D= (-3.82398 + (-0.02990*32) + (-0.09089 *0) + (-0.19558 *0) + (0.02999*165.39) + (0.74572*1.08) = 0.98

Likelihood= 1/ $(1+e^{-0.98})$= 0.73
Hence, the likelihood to give free gift is greater than 0.5 so prediction = Yes

Table below shows the prediction for all instances

ID	Age	Socio economic band B	Socio economic band C	Shop Frequency	Shop Value	Likelihood	Prediction
1	56	1	0	1.6	109.32	0.25	No
2	21	0	1	4.92	11.28	0.35	No
3	48	1	0	1.21	161.19	0.60	Yes
4	37	0	1	0.72	170.65	0.63	Yes
5	32	0	0	1.08	165.39	0.73	Yes

Chapter 8

2. The table below shows the predictions made for a continuous target feature by two different prediction models for a test dataset.

ID	Target	Model 1 Prediction	Model 2 Prediction
1	2,623	2,664	2,691
2	2,423	2,436	2,367

3	2,423	2,399	2,412
4	2,448	2,447	2,440
5	2,762	2,847	2,693
6	2,435	2,411	2,493
7	2,519	2,516	2,598
8	2,772	2,870	2,814
9	2,601	2,586	2,583
10	2,422	2,414	2,485
11	2,349	2,407	2,472
12	2,515	2,505	2,584
13	2,548	2,581	2,604
14	2,281	2,277	2,309
15	2,295	2,280	2,296
16	2,570	2,577	2,612
17	2,528	2,510	2,557
18	2,342	2,381	2,421
19	2,456	2,452	2,393
20	2,451	2,437	2,479
21	2,296	2,307	2,290
22	2,405	2,355	2,490
23	2,389	2,418	2,346
24	2,629	2,582	2,647
25	2,584	2,564	2,546
26	2,658	2,662	2,759
27	2,482	2,492	2,463
28	2,471	2,478	2,403
29	2,605	2,620	2,645
30	2,442	2,445	2,478

a. Based on these predictions, calculate the evaluation measures listed below for each model.

i. The sum of squared errors

Ans: Formula to calculate sum of squared errors is given below:

SSE= SUM((actual value y - predicted value yhat)^2)/2

Table below shows the SSE for Model 1

ID	Target(y)	Model 1 Prediction: (yhat)	Error: (y- yhat)	Error^2

1	2623	2664	-41	1681
2	2423	2436	-13	169
3	2423	2399	24	576
4	2448	2447	1	1
5	2762	2847	-85	7225
6	2435	2411	24	576
7	2519	2516	3	9
8	2772	2870	-98	9604
9	2601	2586	15	225
10	2422	2414	8	64
11	2349	2407	-58	3364
12	2515	2505	10	100
13	2548	2581	-33	1089
14	2281	2277	4	16
15	2295	2280	15	225
16	2570	2577	-7	49
17	2528	2510	18	324
18	2342	2381	-39	1521
19	2456	2452	4	16
20	2451	2437	14	196
21	2296	2307	-11	121
22	2405	2355	50	2500
23	2389	2418	-29	841
24	2629	2582	47	2209
25	2584	2564	20	400
26	2658	2662	-4	16
27	2482	2492	-10	100
28	2471	2478	-7	49
29	2605	2620	-15	225
30	2442	2445	-3	9
				SUM= 33500

Hence for Model 1 SSE= error^2/2 = 33500/2= **16750**

Table below shows the SSE for Model 2

ID	**Target(y)**	**Model 2 Prediction : (yhat)**	**Error: (y- yhat)**	**Error^2**
1	2623	2691	-68	4624
2	2423	2367	56	3136
3	2423	2412	11	121

4	2448	2440	8	64
5	2762	2693	69	4761
6	2435	2493	-58	3364
7	2519	2598	-79	6241
8	2772	2814	-42	1764
9	2601	2583	18	324
10	2422	2485	-63	3969
11	2349	2472	-123	15129
12	2515	2584	-69	4761
13	2548	2604	-56	3136
14	2281	2309	-28	784
15	2295	2296	-1	1
16	2570	2612	-42	1764
17	2528	2557	-29	841
18	2342	2421	-79	6241
19	2456	2393	63	3969
20	2451	2479	-28	784
21	2296	2290	6	36
22	2405	2490	-85	7225
23	2389	2346	43	1849
24	2629	2647	-18	324
25	2584	2546	38	1444
26	2658	2759	-101	10201
27	2482	2463	19	361
28	2471	2403	68	4624
29	2605	2645	-40	1600
30	2442	2478	-36	1296
				SUM= 94738

Hence for Model 2 SSE= error^2/2 = 94738/2= **47369**

ii. The R_2 measure

Ans: Formula to calculate R2 is given below:

$$R^2 = 1 - \frac{sum\ of\ squared\ errors}{total\ sum\ of\ squares}$$

Hence to calculate R2, we need to calculate SSE(sum of squared errors) and SST(total sum of squares) first.

Formula to calculate sum of squared errors is given below:

SSE= SUM((actual value y - predicted value yhat)^2)/2

Formula to calculate sum of squared total is given below:
SST= SUM((mean of y - predicted value yhat)^2)/2

Table below shows the calculation for SSE and SST for Model 1

ID	Target(y)	Model 1 Prediction : (yhat)	Error: (y- yhat)	Error^2	SST Mean(y)-yhat	SST^2
1	2623	2664	-41	1681	-173.2	29998.24
2	2423	2436	-13	169	54.8	3003.04
3	2423	2399	24	576	91.8	8427.24
4	2448	2447	1	1	43.8	1918.44
5	2762	2847	-85	7225	-356.2	126878.44
6	2435	2411	24	576	79.8	6368.04
7	2519	2516	3	9	-25.2	635.04
8	2772	2870	-98	9604	-379.2	143792.64
9	2601	2586	15	225	-95.2	9063.04
10	2422	2414	8	64	76.8	5898.24
11	2349	2407	-58	3364	83.8	7022.44
12	2515	2505	10	100	-14.2	201.64
13	2548	2581	-33	1089	-90.2	8136.04
14	2281	2277	4	16	213.8	45710.44
15	2295	2280	15	225	210.8	44436.64
16	2570	2577	-7	49	-86.2	7430.44
17	2528	2510	18	324	-19.2	368.64
18	2342	2381	-39	1521	109.8	12056.04
19	2456	2452	4	16	38.8	1505.44
20	2451	2437	14	196	53.8	2894.44
21	2296	2307	-11	121	183.8	33782.44
22	2405	2355	50	2500	135.8	18441.64
23	2389	2418	-29	841	72.8	5299.84
24	2629	2582	47	2209	-91.2	8317.44
25	2584	2564	20	400	-73.2	5358.24
26	2658	2662	-4	16	-171.2	29309.44
27	2482	2492	-10	100	-1.2	1.44
28	2471	2478	-7	49	12.8	163.84
29	2605	2620	-15	225	-129.2	16692.64
30	2442	2445	-3	9	45.8	2097.64
				SUM/2= 16750		**SUM/2= 292604.6**

Hence, R2 for Model 1 = 1- (SSE/SST) = 1- (16750/292604.6) = 0.9428

Table below shows the calculation for SSE and SST for Model 2

ID	Target(y)	Model 2 Prediction : (yhat)	Error: (y- yhat)	Error^2	SST Mean(y)- yhat	SST^2
1	2623	2691	-68	4624	-200.2	40080.04
2	2423	2367	56	3136	123.8	15326.44
3	2423	2412	11	121	78.8	6209.44
4	2448	2440	8	64	50.8	2580.64
5	2762	2693	69	4761	-202.2	40884.84
6	2435	2493	-58	3364	-2.2	4.84
7	2519	2598	-79	6241	-107.2	11491.84
8	2772	2814	-42	1764	-323.2	104458.24
9	2601	2583	18	324	-92.2	8500.84
10	2422	2485	-63	3969	5.8	33.64
11	2349	2472	-123	15129	18.8	353.44
12	2515	2584	-69	4761	-93.2	8686.24
13	2548	2604	-56	3136	-113.2	12814.24
14	2281	2309	-28	784	181.8	33051.24
15	2295	2296	-1	1	194.8	37947.04
16	2570	2612	-42	1764	-121.2	14689.44
17	2528	2557	-29	841	-66.2	4382.44
18	2342	2421	-79	6241	69.8	4872.04
19	2456	2393	63	3969	97.8	9564.84
20	2451	2479	-28	784	11.8	139.24
21	2296	2290	6	36	200.8	40320.64
22	2405	2490	-85	7225	0.8	0.64
23	2389	2346	43	1849	144.8	20967.04
24	2629	2647	-18	324	-156.2	24398.44
25	2584	2546	38	1444	-55.2	3047.04
26	2658	2759	-101	10201	-268.2	71931.24
27	2482	2463	19	361	27.8	772.84
28	2471	2403	68	4624	87.8	7708.84
29	2605	2645	-40	1600	-154.2	23777.64
30	2442	2478	-36	1296	12.8	163.84
				SUM/2= 47369		**SUM/2= 274579.6**

Hence, R2 for Model 2 = 1- (SSE/SST) = 1- (47369/274579.6) = 0.8275

b. Based on the evaluation measures calculated, which model do you think is performing better for this dataset?

Ans: Based on the value of R2 i.e. coefficient of determination, the value of R2 is higher for Model 1(which is 0.9428) as compared to Model 2 (which is 0.8275). R2 value tells the proportion of variance in the dependent variable that can be explained by the independent variable. Hence, Model 1 has higher capacity to explain the change in dependent variable caused by independent variable.

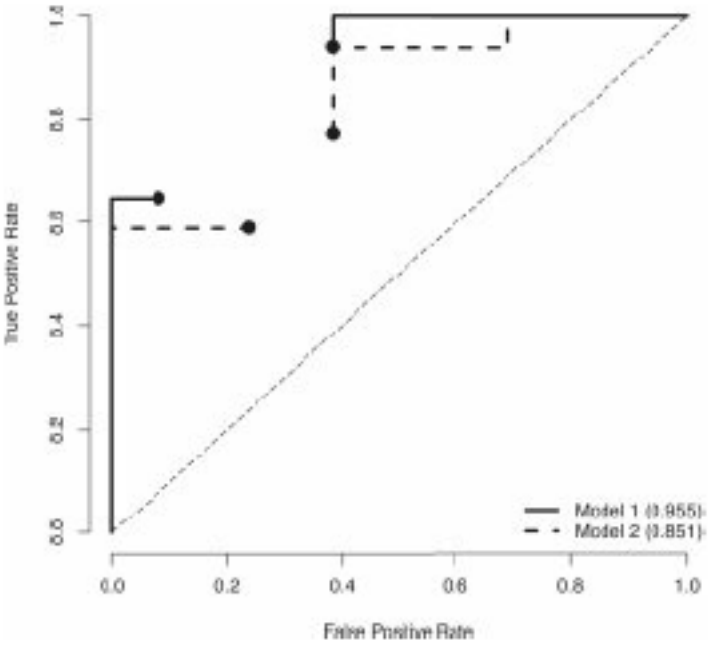
3. A credit card issuer has built two different credit scoring models that predict the propensity of customers to default on their loans. The outputs of the first model for a test dataset are shown in the table below.

ID	Target	Score	Prediction
1	bad	0.634	bad
2	bad	0.782	bad
3	good	0.464	good
4	bad	0.593	bad
5	bad	0.827	bad
6	bad	0.815	bad
7	bad	0.855	bad
8	good	0.500	good
9	bad	0.600	bad
10	bad	0.803	bad
11	bad	0.976	bad
12	good	0.504	bad
13	good	0.303	good
14	good	0.391	good
15	good	0.238	good
16	good	0.072	good
17	bad	0.567	bad
18	bad	0.738	bad
19	bad	0.325	good
20	bad	0.863	bad
21	bad	0.625	bad
22	good	0.119	good
23	bad	0.995	bad
24	bad	0.958	bad
25	bad	0.726	bad
26	good	0.117	good
27	good	0.295	good
28	good	0.064	good
29	good	0.141	good
30	good	0.670	bad

The outputs of the second model for the same test dataset are shown in the table below.

ID	Target	Score	Prediction
1	bad	0.230	bad
2	bad	0.859	good
3	good	0.154	bad
4	bad	0.325	bad
5	bad	0.952	good
6	bad	0.900	good
7	bad	0.501	good
8	good	0.650	good
9	bad	0.940	good
10	bad	0.806	good
11	bad	0.507	good
12	good	0.251	bad
13	good	0.597	good
14	good	0.376	bad
15	good	0.285	bad
16	good	0.421	bad
17	bad	0.842	good
18	bad	0.891	good
19	bad	0.480	bad
20	bad	0.340	bad
21	bad	0.962	good
22	good	0.238	bad
23	bad	0.362	bad
24	bad	0.848	good
25	bad	0.915	good
26	good	0.096	bad
27	good	0.319	bad
28	good	0.740	good
29	good	0.211	bad
30	good	0.152	bad

a. The image below shows an ROC curve for each model. Each curve has a point missing. Calculate the missing point in the ROC curves for Model 1 and Model 2. To generate the point for Model 1, use a threshold value of 0.51. To generate the point for Model 2, use a threshold value of 0.43.



Ans: As per threshold for each model true positive rate and false positive rate will be calculated. Formula for true positive rate and false positive rate is given below:

TPR (Sensitivity): True Positive/(True Positive +False Negative)

FPR(1- specificity) : False Positive/(False Positive +True Negative)

Then ROC Curve will be completed

For Model 1 at threshold = 0.51:

ID	Target	Score	Prediction	Prediction at threshold= 0.51	outcome
1	bad	0.634	Bad	bad	TP
2	bad	0.782	bad	bad	TP
3	good	0.464	good	good	TN
4	bad	0.593	bad	bad	TP
5	bad	0.827	bad	bad	TP
6	bad	0.815	bad	bad	TP
7	bad	0.855	bad	bad	TP
8	good	0.5	good	good	TN
9	bad	0.6	bad	bad	TP
10	bad	0.803	bad	bad	TP

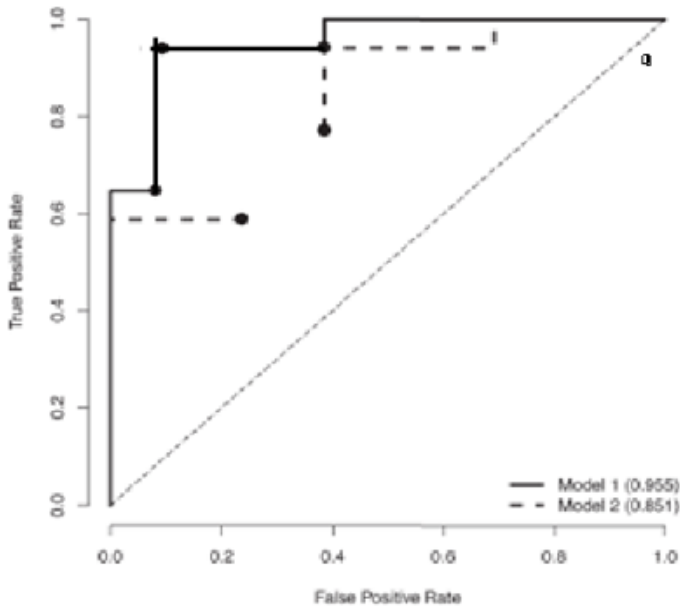
11	bad	0.976	bad	bad	TP
12	good	0.504	bad	good	TN
13	good	0.303	good	good	TN
14	good	0.391	good	good	TN
15	good	0.238	good	good	TN
16	good	0.072	good	good	TN
17	bad	0.567	bad	bad	TP
18	bad	0.738	bad	bad	TP
19	bad	0.325	good	good	FN
20	bad	0.863	bad	bad	TP
21	bad	0.625	bad	bad	TP
22	good	0.119	good	good	TN
23	bad	0.995	bad	bad	TP
24	bad	0.958	bad	bad	TP
25	bad	0.726	bad	bad	TP
26	good	0.117	good	good	TN
27	good	0.295	good	good	TN
28	good	0.064	good	good	TN
29	good	0.141	good	good	TN
30	good	0.67	bad	bad	FP

Confusion matrix is given below

	bad	good
bad	16	1
good	1	12

TPR= TP/ (TP+FN) = 16 / (16 +1) = 0.9412
FPR = FP/(FP+TN) = 1/(12+1) = 0.0769

Using these points, we can plot an extra point with (x,y) coordinate as (0.0769, 0.9412) on the ROC curve and this point will complete the curve as shown below.



For Model 2 at threshold = 0.43:

ID	Target	Score	Prediction	Prediction at threshold= 0.43	outcome
1	bad	0.230	bad	good	FN
2	bad	0.859	good	bad	TP
3	good	0.154	bad	good	TN
4	bad	0.325	bad	good	FN
5	bad	0.952	good	bad	TP
6	bad	0.900	good	bad	TP
7	bad	0.501	good	bad	TP
8	good	0.650	good	bad	FP
9	bad	0.940	good	bad	TP
10	bad	0.806	good	bad	TP
11	bad	0.507	good	bad	TP
12	good	0.251	bad	good	TN
13	good	0.597	good	bad	FP
14	good	0.376	bad	good	TN
15	good	0.285	bad	good	TN
16	good	0.421	bad	good	TN
17	bad	0.842	good	bad	TP
18	bad	0.891	good	bad	TP
19	bad	0.480	bad	bad	TP
20	bad	0.340	bad	good	FN
21	bad	0.962	good	bad	TP
22	good	0.238	bad	good	TN
23	bad	0.362	bad	good	FN

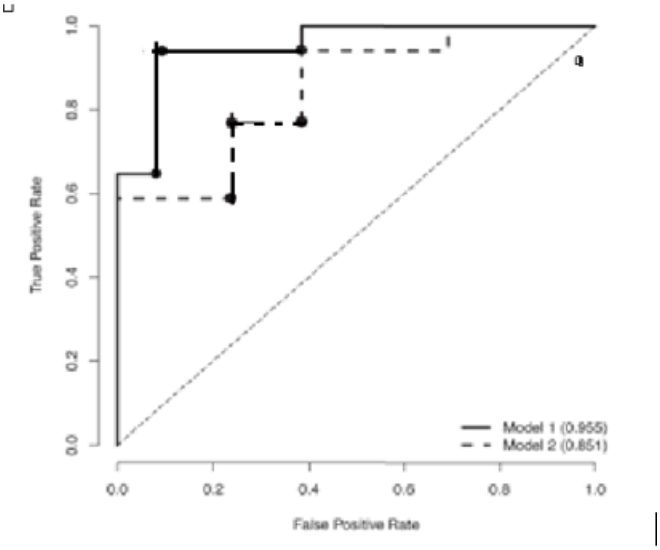
24	bad	0.848	good	bad	TP
25	bad	0.915	good	bad	TP
26	good	0.096	bad	good	TN
27	good	0.319	bad	good	TN
28	good	0.740	good	bad	FP
29	good	0.211	bad	good	TN
30	good	0.152	bad	good	TN

Confusion matrix is given below

	bad	good
bad	13	4
good	3	10

TPR= TP/ (TP+FN) = 13 / (13 +4) = 0.7647
FPR = FP/(FP+TN) = 3/(3+10) = 0.2308

Using these points, we can plot an extra point with (x,y) coordinate as (0.2308, 0.7647) on the ROC curve and this point will complete the curve as shown in figure below



b. The area under the ROC curve (AUC) for Model 1 is 0.955 and for Model 2 is 0.851. Which model is performing best?

Ans: Based on AUC score, Model 1 performed better than model 2. Also the ROC curve for model 1 is higher than the ROC curve for model i.e. there is no data point for which model 2 has higher x,y coordinate than model 1 as shown in figure above.

c. Based on the AUC values for Model 1 and Model 2, calculate the Gini coefficient for each model.

Ans: Formula to calculate Gini coefficient is given below:

$$Gini\ coefficient = (2 \times ROC\ index) - 1$$

As per question b, roc index is AUC score. Hence, Roc index for model 1 = 0.955 and for Model 2 = 0.851.

Gini Coefficient for Model 1 = (2*0.955) -1 = 0.91

Gini Coefficient for Model 2 = (2*0.851) -1 = 0.702

6. (25%, coding assignment) What is ordinary least square (OLS) method to obtain linear regression model weights? See https://en.wikipedia.org/wiki/Ordinary_least_squares. Can you implement this OLS method by Python to predict following dataset (https://drive.google.com/open?id=1oakZCv7g3mlmCSdv9J8kdSaqO5_6dIOw)?

importing the data

```
1 scores_df = pd.read_csv(r"/Users/iqrabismi/Desktop/student_scores.csv")
```

```
1 scores_df.head()
```

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

```
1 len(scores_df)
```

```
25
```

```
1 len(scores_df[scores_df.duplicated()==False])
```

```
25
```

```
1 scores_df.isnull().sum()
```

```
Hours     0
Scores    0
dtype: int64
```

```
1 scores_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Hours   25 non-null     float64
 1   Scores  25 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 528.0 bytes
```

```
1 scores_df.describe()
```

	Hours	Scores
count	25.000000	25.000000
mean	5.012000	51.480000
std	2.525094	25.286887
min	1.100000	17.000000
25%	2.700000	30.000000
50%	4.800000	47.000000
75%	7.400000	75.000000
max	9.200000	95.000000

```
1 scores_df.corr()
```

	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

Running the Model

```
1 model = ols("Scores ~ Hours", data= scores_df)
2 model = model.fit()
3 print(model.params)
```

```
Intercept    2.483673
Hours        9.775803
dtype: float64
```

Hence equation is :

Y= 2.483673 + Hours*9.775803

Residual attribute

```
1 print(model.resid)
```

```
0     -5.923182
1     -5.340271
2     -6.766244
3    -10.578002
4     -6.698985
5      2.852622
6     -4.421065
7      3.749408
8     -2.622842
9     -3.878343
10     7.242640
11     1.839087
12    -5.474789
13     7.256175
14     3.762943
15     5.511676
16     3.076818
17     2.942300
18     4.883926
19    -5.824618
20     1.121657
21     4.592470
22    -4.631726
23     6.063283
24     7.265060
dtype: float64
```

```
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 Scores   R-squared:                       0.953
Model:                            OLS   Adj. R-squared:                  0.951
Method:                 Least Squares   F-statistic:                     465.8
Date:                Fri, 02 Dec 2022   Prob (F-statistic):           9.13e-17
Time:                        01:33:07   Log-Likelihood:                -77.514
No. Observations:                  25   AIC:                             159.0
Df Residuals:                      23   BIC:                             161.5
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      2.4837      2.532      0.981      0.337      -2.753       7.721
Hours          9.7758      0.453     21.583      0.000       8.839      10.713
==============================================================================
Omnibus:                        7.616   Durbin-Watson:                   1.460
Prob(Omnibus):                  0.022   Jarque-Bera (JB):                2.137
Skew:                          -0.216   Prob(JB):                        0.343
Kurtosis:                       1.634   Cond. No.                         13.0
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

R-squared - statistical measure of how well the model will detect variation in dependent variable caused by independent variable.

Adj. R-squared - It adjusts the statistics based on the number of independent variables present.

F-statistic - It the ratio of mean squared error of the model to the mean squared error of residuals.

AIC - estimates the relative quality of statistical models for a given dataset.

BIC - used as a criterion for model selection among a finite set of models.

coef - the coefficients of the independent variables and the constant term in the equation.

std err - the basic standard error of the estimate of the coefficient.

t - a measure of how statistically significant the coefficient is.

P > |t| - the null-hypothesis that the coefficient = 0 is true.

RSquared

```
print(model.rsquared)
```

```
0.9529481969048356
```

Rsquared is correlation squared

```
coeff_determination = scores_df["Scores"].corr(scores_df["Hours"]) ** 2
print(coeff_determination)
```

```
0.9529481969048358
```

Mean Squared Error and Root Mean Squared Error

```
1 mse = model.mse_resid
2 print('mse: ', mse)
```

```
mse:  31.394272292658112
```

```
1 rse = np.sqrt(mse)
2 print("rse: ", rse)
```

```
rse:  5.6030591905367295
```

Residual Squared

```
1 residuals_sq = model.resid ** 2
2 print("residuals sq: \n", residuals_sq)
```

```
residuals sq:
 0      35.084084
1       28.518491
2       45.782061
3      111.894131
4       44.876404
5        8.137449
6       19.545812
7       14.058060
8        6.879298
9       15.041541
10      52.455841
11       3.382239
12      29.973311
13      52.652082
14      14.159739
15      30.378577
16       9.466810
17       8.657130
18      23.852732
19      33.926181
20       1.258115
21      21.090784
22      21.452888
23      36.763403
24      52.781099
dtype: float64
```

RSE (Residual Standard Error)

```
residuals_sq = model.resid **2
resid_sum_of_sq =sum(residuals_sq)
deg_freedom =len(scores_df.index) -2
print
("deg freedom: ", deg_freedom)
```

```
('deg freedom: ', 23)
```

```
residuals_sq = model.resid **2
resid_sum_of_sq =sum(residuals_sq)
deg_freedom =len(scores_df.index) -2
rse = np.sqrt(resid_sum_of_sq/deg_freedom)
print("rse :", rse)
```

```
rse : 5.6030591905367295
```

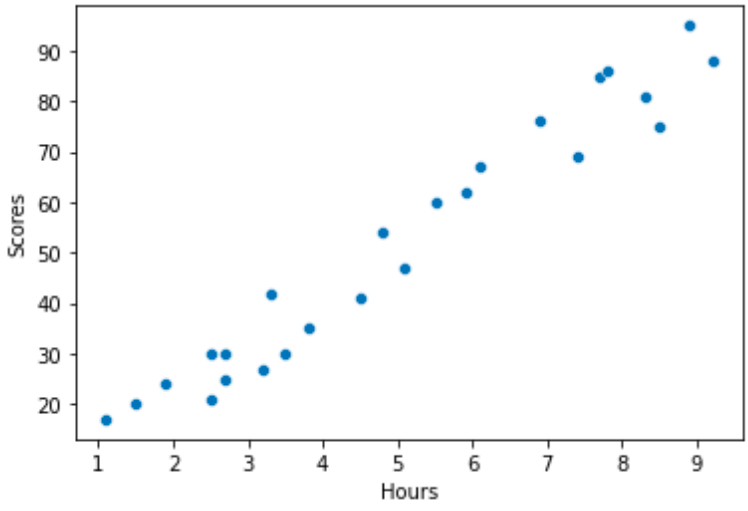
Intrepreting RSE: The difference between predicted scores and observed scores is typically around 5.

Analysing the Model fit based on Linear Regression conditions

1. Linear Relation Between Dependent and independent variable

```
sns.scatterplot(data= scores_df, x='Hours', y='Scores')
```

```
<AxesSubplot:xlabel='Hours', ylabel='Scores'>
```

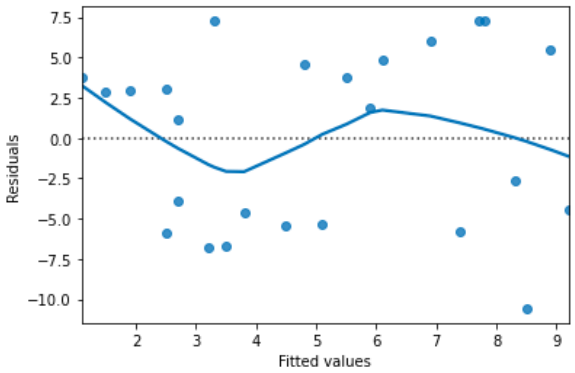


Relation between independent and dependent variable is linear

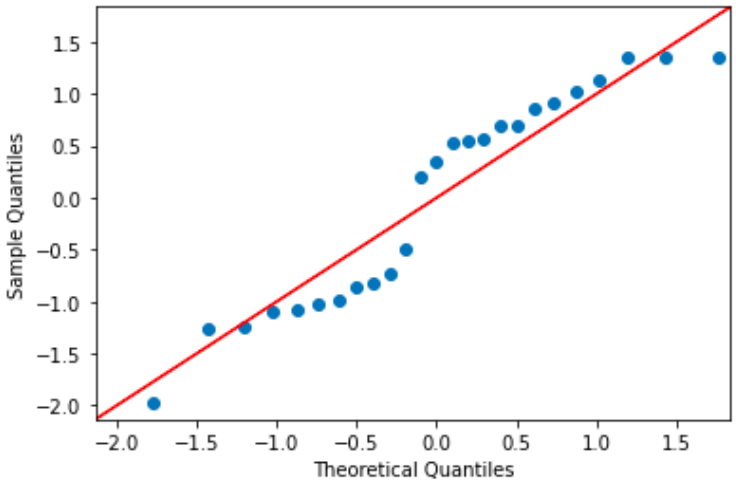
2. Residual are to be normally distributed. This can be checked by quantile plot and distplot

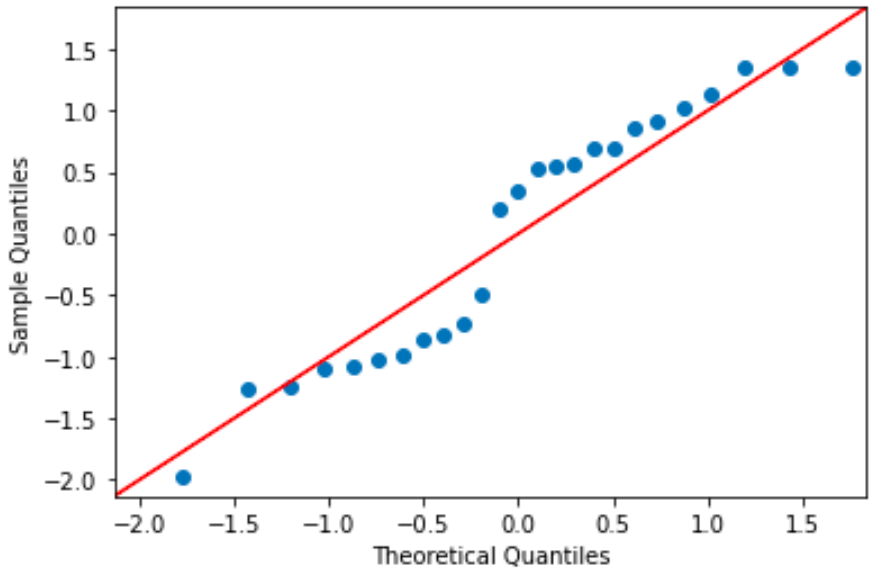
```
sns.residplot(x="Hours", y="Scores", data=scores_df, lowess=True)
plt.xlabel("Fitted values")
plt.ylabel("Residuals")
```

```
Text(0, 0.5, 'Residuals')
```




```
from statsmodels.api import qqplot
qqplot(data=model.resid, fit=True, line="45")
```

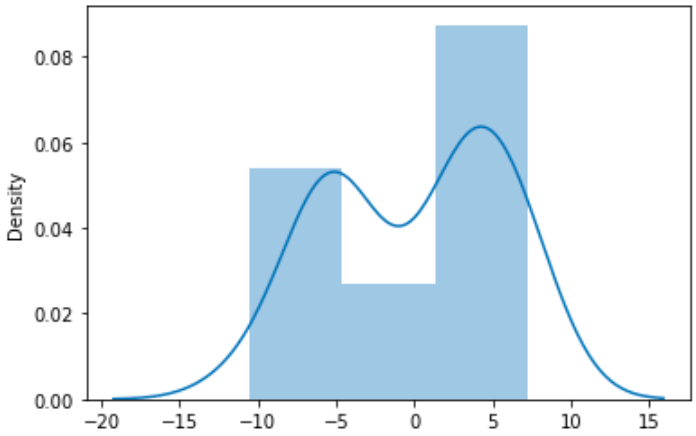





```
1 sns.distplot(model.resid)
```

```
/opt/anaconda3/lib/python3.9/site-packages/seaborn/distributions.py:2619: Futu
unction and will be removed in a future version. Please adapt your code to use
ion with similar flexibility) or `histplot` (an axes-level function for histog
  warnings.warn(msg, FutureWarning)
```

```
<AxesSubplot:ylabel='Density'>
```

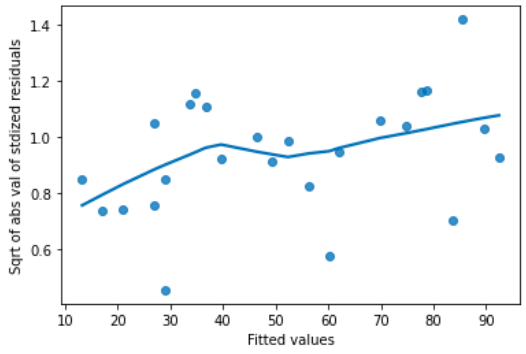


In quantile plot,the residuals roughly lie on the straight line. Also, in distplot, the distribution is bimodal. Hence, residual are normally distributed

3. Homodasticity i.e variance of the residual is constant. This can be checked by Scale-location plot

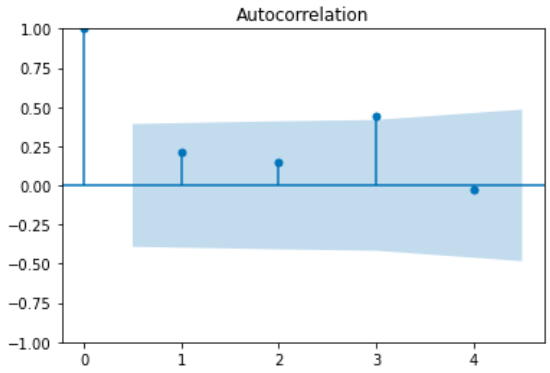
```
model_norm_residuals = model.get_influence().resid_studentized_internal
model_norm_residuals_abs_sqrt = np.sqrt(np.abs(model_norm_residuals))
sns.regplot(x=model.fittedvalues, y=model_norm_residuals_abs_sqrt, ci=None, lowess=True)
plt.xlabel("Fitted values")
plt.ylabel("Sqrt of abs val of stdized residuals")
```

```
Text(0, 0.5, 'Sqrt of abs val of stdized residuals')
```




```
import statsmodels.tsa.api as smt
acf = smt.graphics.plot_acf(model.resid, lags=4 , alpha=0.05)
acf.show()
```

```
/var/folders/fg/0cvyhxdj5v7503hv35z31yhr0000gn/T/ipykernel_43864/1753263740.py:3: UserWarning: Matp
y using module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot show the f
  acf.show()
```



The line is roughly horizontal and the residuals are randomly scattered. Hence, no correlation amongst the residuals.

4. No Multicollinearity

As there is only one independent feature, so multicollinearity doesn't exist. If the number of descriptive features are more than one. Then we can check multicollinearity by using statsmodel variance inflation factor.

from statsmodels.stats.outliers_influence import variance_inflation_factor

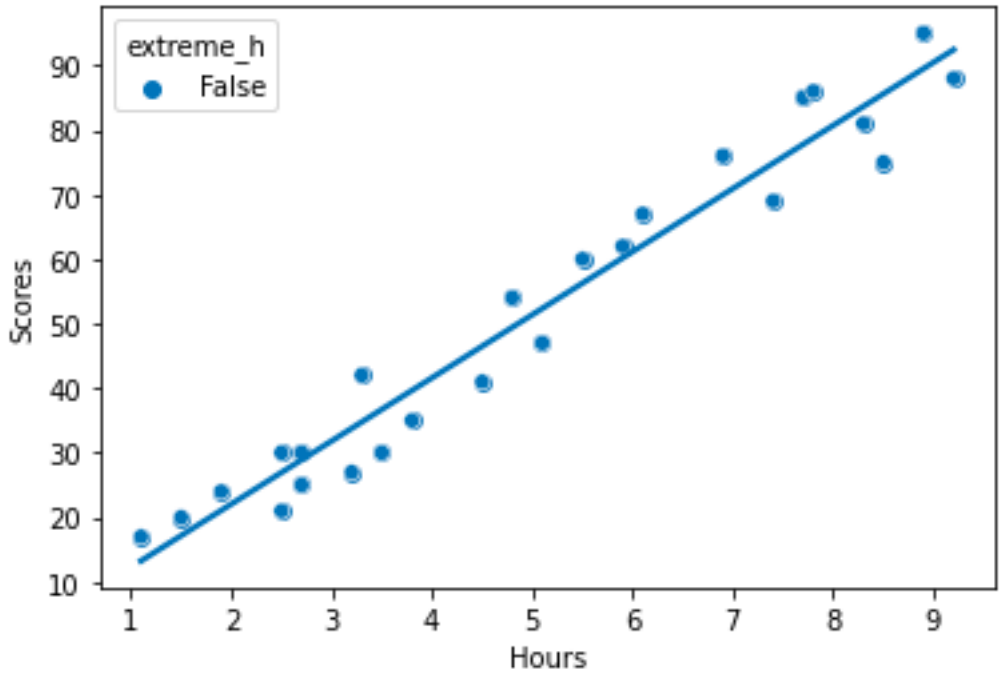
5. No outliers

Checking for extreme values by IQR ie. if the value is less than q1 - 1.5 iqr and if greater than q3+ 1.5 iqr. Then it is extreme.

```
iqr= np.quantile(scores_df['Hours'],0.75) - np.quantile(scores_df['Hours'],0.25)
```

```
iqr= np.quantile(scores_df['Hours'],0.75) - np.quantile(scores_df['Hours'],0.25)
```

```
scores_df["extreme_h"] = ((scores_df["Hours"] < (np.quantile(scores_df['Hours'],0.25) - (1.5*iqr))) |
(scores_df["Hours"] > (np.quantile(scores_df['Hours'],0.75) + (1.5*iqr))))
fig = plt.figure()
sns.regplot(x="Hours",
y="Scores",
data=scores_df,
ci=None)
sns.scatterplot(x="Hours",
y="Scores",
hue="extreme_h",
data=scores_df)
```




```
Calculating outliers by cook's distance. Cook's distance is the most common measure of infuence.

Leverage is a measure of how extreme the explanatory variable values are.

Infuence measures how much the model would change if you le/ the observation out of the
dataset when modeling.
```

```
model = ols("Scores ~ Hours", data=scores_df).fit()
summary = model.get_influence().summary_frame()
scores_df["leverage"] = summary_roach["hat_diag"]
print(scores_df.head())
```

```
   Hours  Scores  leverage
0    2.5      21  0.081236
1    5.1      47  0.040051
2    3.2      27  0.061456
3    8.5      75  0.119504
4    3.5      30  0.054940
```

```
scores_df["cooks_dist"] = summary_roach["cooks_d"]
print(scores_df.head())
```

```
   Hours  Scores  leverage  cooks_dist
0    2.5      21  0.081236    0.053773
1    5.1      47  0.040051    0.019741
2    3.2      27  0.061456    0.050871
3    8.5      75  0.119504    0.274696
4    3.5      30  0.054940    0.043965
```

```
print(scores_df.sort_values("cooks_dist", ascending = False))
```

```
    Hours  Scores  leverage  cooks_dist
3     8.5      75  0.119504    0.274696
24    7.8      86  0.090795    0.092328
15    8.9      95  0.138784    0.090532
10    7.7      85  0.087216    0.087453
6     9.2      88  0.154616    0.067347
13    3.3      42  0.059153    0.056037
0     2.5      21  0.081236    0.053773
2     3.2      27  0.061456    0.050871
19    7.4      69  0.077265    0.049032
4     3.5      30  0.054940    0.043965
14    1.1      17  0.140007    0.042691
23    6.9      76  0.063294    0.042237
12    4.5      41  0.041713    0.021684
9     2.7      25  0.074931    0.020976
5     1.5      20  0.120601    0.020211
18    6.1      67  0.047736    0.019998
1     5.1      47  0.040051    0.019741
22    3.8      35  0.049599    0.018761
17    1.9      24  0.103287    0.017711
8     8.3      81  0.110648    0.015327
21    4.8      54  0.040294    0.014695
16    2.5      30  0.081236    0.014510
7     5.5      60  0.041556    0.010129
11    5.9      62  0.045153    0.002668
20    2.7      30  0.074931    0.001754
```

Hence all the values are in range and no extreme values are observed by cook's distance as well as IQR method

From the above analysis,we checked that all the base conditions are satisfied

Splitting the data

```
X= scores_df.Hours.values
Y= scores_df.Scores.values
```

```
xtrain,xtest, ytrain,ytest= train_test_split(X,Y,test_size= 0.2, random_state= 10)
```

Running the Model on trained data

```
scores_df2= pd.DataFrame({'Hours':xtrain,'Scores':ytrain})
```

```
model = ols("Scores ~ Hours", data= scores_df2)
model = model.fit()
print(model.params)
```

```
Intercept    2.649965
Hours        9.814305
dtype: float64
```

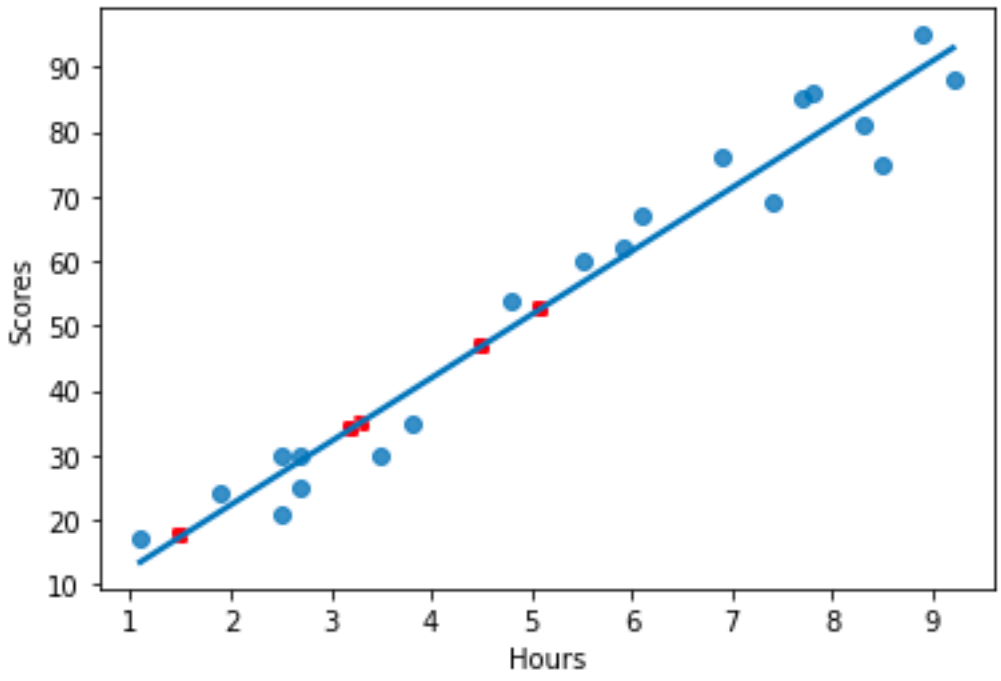
```
print(model.predict(pd.DataFrame({'Hours':xtest})))
```

```
0     17.371423
1     52.702923
2     35.037173
3     34.055743
4     46.814340
dtype: float64
```

```
explanatory_data = pd.DataFrame(
{'Hours':xtest}
)
prediction_data = explanatory_data.assign(
Scores=model.predict(explanatory_data)
)
```

Showing the Predictions

```
fig = plt.figure()
sns.regplot(x="Hours",
y="Scores",
ci=None,
data=scores_df2,)
sns.scatterplot(x="Hours",
y="Scores",
data=prediction_data,
color="red",
marker="s")
plt.show()
```

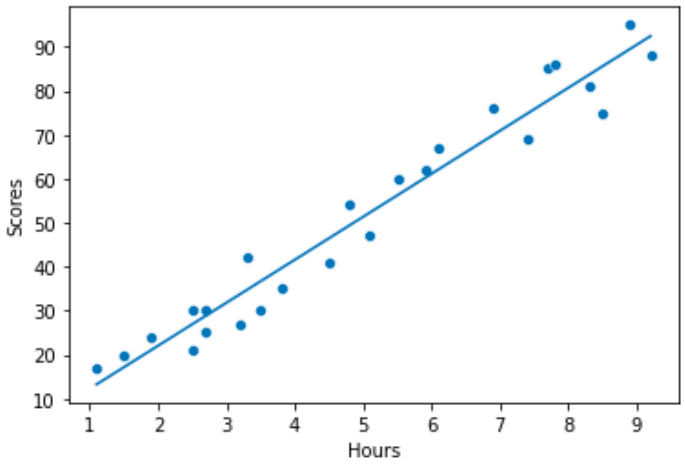


Fitting the OLS line

```
# Fitting the OLS regression line

sns.scatterplot(data=scores_df,x="Hours",y="Scores")
x=np.linspace(min(scores_df['Hours']),max(scores_df['Hours']),30)
y=model.params[0]+ (x * model.params[1])
plt.plot(x,y)
```

```
[<matplotlib.lines.Line2D at 0x7fd6c02dcdc0>]
```




```
sns.regplot(data= scores_df, x='Hours', y='Scores', ci= None)
```

```
<AxesSubplot:xlabel='Hours', ylabel='Scores'>
```