TASK 1:-

<u>SIMPLE LINEAR REGRESSION</u>

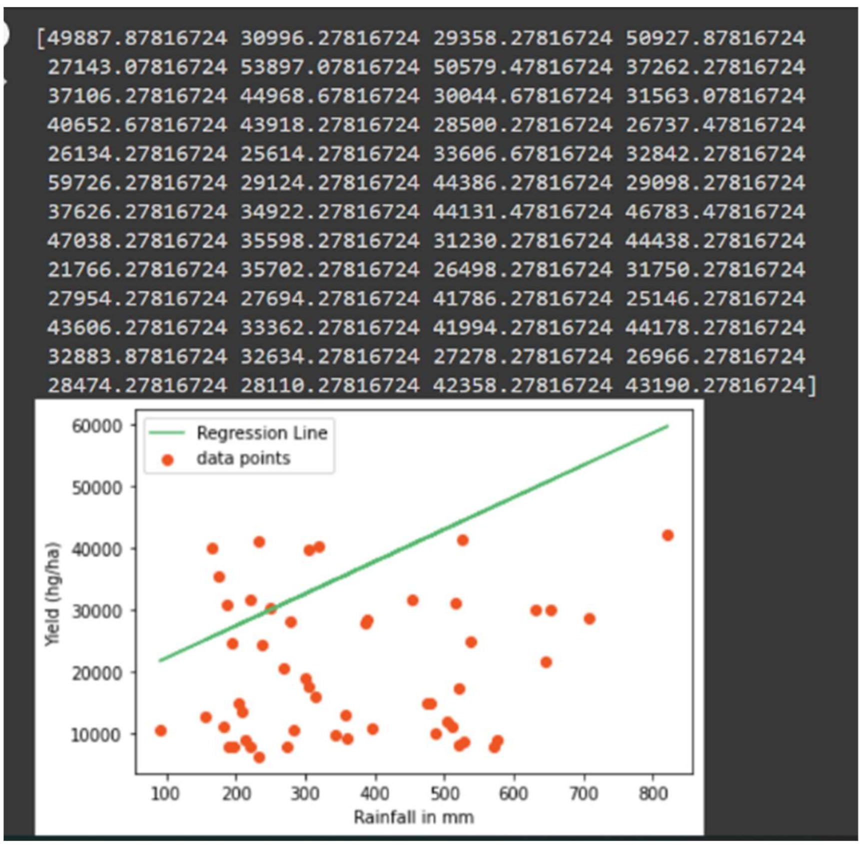
Steps / Procedures

- IMPORT LIBRARIES
- UPLOAD DATA FILES(CSV) FROM THE LOCAL DRIVE IN GOOGLE COLLAB
- READ AND PRINT CSV FILE
- FIND MEAN OF THE RAINFALL AND YIELD FROM THE GIVEN DATASETS
- USE THE FORMULA OF LEAST SQUARE METHOD TO FIND M AND C

$$m = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$c = \bar{y} - m\bar{x}$$

- PLOT THE VALUES AND REGRESSION LINE
- THE GIVEN FIGURE Fig1 IS OBTAINED FROM THE COLLAB PROGRAMMING CODE
-



- FIG1:- DATA AND LINE OF REGRESSION

- USE THE EQN Y=MX + C TO FIND THE PREDICTED CROP YIELD IN YEAR 2022 IN JODHPUR BY THE GIVEN DATA SET RAINFALL 560 MM

```
crop yield in year 2022 is  46154.27816724092 hg/ha
```

- USE THE FORMULA TO FIND MSE AND MAE

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right|$$

- ```
  MSE= 424444227.7396561
  MAE= 35770.69739801019
  ```

- REFERENCES:-

https://dphi.tech/blog/tutorial-on-linear-regression-using-least-squares/

https://www.youtube.com/watch?v=lzGKRSvs5HM

TASK 2:-

MULTI VARIANT REGRESSION

STEPS / PROCEDURES

● IMPORT LIBRARIES

● READ THE DATA SET WITH THE HELP OF A PANDA OBJECT

● IDENTIFY DEPENDENT VARIABLES AND INDEPENDENT VARIABLES (THE PRICE OF THE CAR)

● DATA CLEANING INCLUDES REPLACING '?' MISSING DATA AND OUTLIERS

index	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	height	curb-weight	eng
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	doh
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	2548	doh
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	52.4	2823	ohc
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	54.3	2337	ohc
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	54.3	2824	ohc

● DIVIDING THE DATA INTO TRAIN AND TEST DATA

● MAKING FEATURE MATRIX AND THETA MATRIX

● MAKING COST FUNCTION (NOTE:- GOAL IS TO MAXIMIZATION OF THE COST FUNCTION)
  ```

- MAKING GRADIENT DESCENT FUNCTION
- MAXIMIZING THE COST FUNCTION

ing	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	ci
3	103	2	0	1	1	0	1	1	88.6	...	130	0	17	28	9.0	25	4	
3	103	2	0	1	1	0	1	1	88.6	...	130	0	17	28	9.0	25	4	
1	103	2	0	1	1	2	1	1	94.5	...	152	0	26	19	9.0	36	4	
2	164	9	0	1	2	4	0	1	99.8	...	109	0	20	3	10.0	11	10	
2	164	9	0	1	2	4	2	1	99.4	...	136	0	20	3	8.0	7	10	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
-1	95	13	0	1	2	4	1	1	109.1	...	141	0	29	20	9.5	4	20	
-1	95	13	0	0	2	4	1	1	109.1	...	141	0	29	20	8.7	56	2	
-1	95	13	0	1	2	4	1	1	109.1	...	173	0	30	27	8.8	43	10	
-1	95	13	1	0	2	4	1	1	109.1	...	145	1	38	3	23.0	58	12	
-1	95	13	0	0	2	4	1	1	109.1	...	141	0	29	20	9.5	4	20	

columns

FIG2. CLEANED DATA

```
[58]
     (164, 23)
     Final value of theta =
      [4.0521238589705455e+133 4.417103769436437e+132 3.210798282435659e+133
      6.310393508102201e+133 1.2047377266709173e+134 2.153540803111886e+133
      3.998608272259483e+133 4.0451080586482074e+135 7.142656173300054e+135
      2.6864452588086375e+135 2.1826405509252533e+135 1.0827495801242705e+137
      1.5838663854536586e+134 4.032918713291077e+133 5.395378261128599e+135
      7.105373811350715e+133 7.79044473005342e+134 6.748308251701769e+134
      4.163794747427944e+134 1.1393579903726368e+135 3.886208607845247e+134
      9.813257537061921e+134 1.2010808292857039e+135]
     First 5 values from cost_history = [5.32700289e+14 6.38364302e+25 7.64987350e+36 9.16726772e+47
      1.09856454e+59]
     Last 5 values from cost_history = [1.98699214e+236 2.38112290e+247 2.85343164e+258 3.41942540e+269
      4.09768711e+280]

[59] y_pred = X_test.dot(thetax[1:])
     y_pred += thetax[0]
     # sum((y_test-y_pred)**2)/(2*m)
     # sum((y_pred-np.mean(y_train))**2)/sum((y_train-np.mean(y_train))**2)
     sum((y_pred-np.mean(train_df['price']))**2)/sum((train_df['price']-np.mean(train_df['price']))**2)

     6.789112675931917e+276
```

FIG2. FINAL VALUE OF THETA.

REFERENCE:-

https://satishgunjal.com/machine_learning/

KRISH NAYAK NOTES

TASK3:-

POLYNOMIAL REGRESSION STEPS

/ PROCEDURES

- IMPORT LIBRARIES
- READ THE DATA SET WITH THE HELP OF A PANDA OBJECT
- DRAW THE LINEAR REGRESSION MODEL 1 FOR THE GIVEN DATASET
- LINEAR REGRESSION MODEL ALGO IS SAME AS IN TASK1

```
m = 23794.938432413885
c = -691984.4346015881
[ 124933.63573529  -50726.31055802 1960633.42275227  525840.75231874
 -502773.74676979 1701267.16614265 2210023.88937348 -162012.14426442
 2003693.45718772  905037.84297522  941347.2533774  1592428.40390464
  861023.39355925    5054.97386217 1491245.42515787 1536142.4288403
 1361681.50949461 -183320.51163065  590241.75318607  466245.85311599
 2420452.76175484 1200360.15515129 2301532.08410302 1734601.49539262
 1663373.25079027 1735491.90198876 2235556.33421023  -27645.22027857
 1887017.59402761 -433816.49109143 -581010.07663926 1768728.43404177]
```

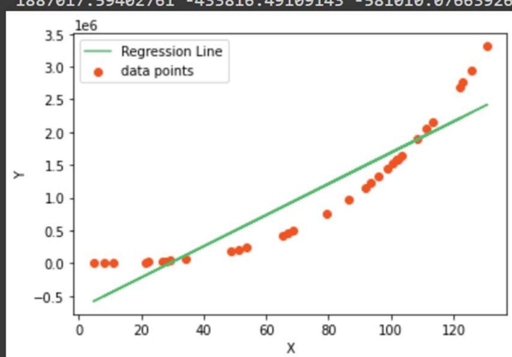


FIG1: Linear Regression Model

- MAKED POLYNOMIAL REGRESSION MODEL 2 OF DEGREE 2

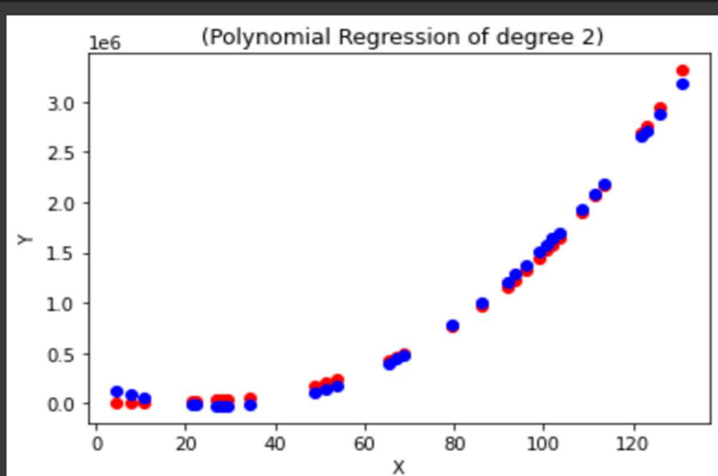


Fig2:- Polynomial Regression (Degree 2) []

● MAKED POLYNOMIAL REGRESSION MODEL 2 OF DEGREE 3

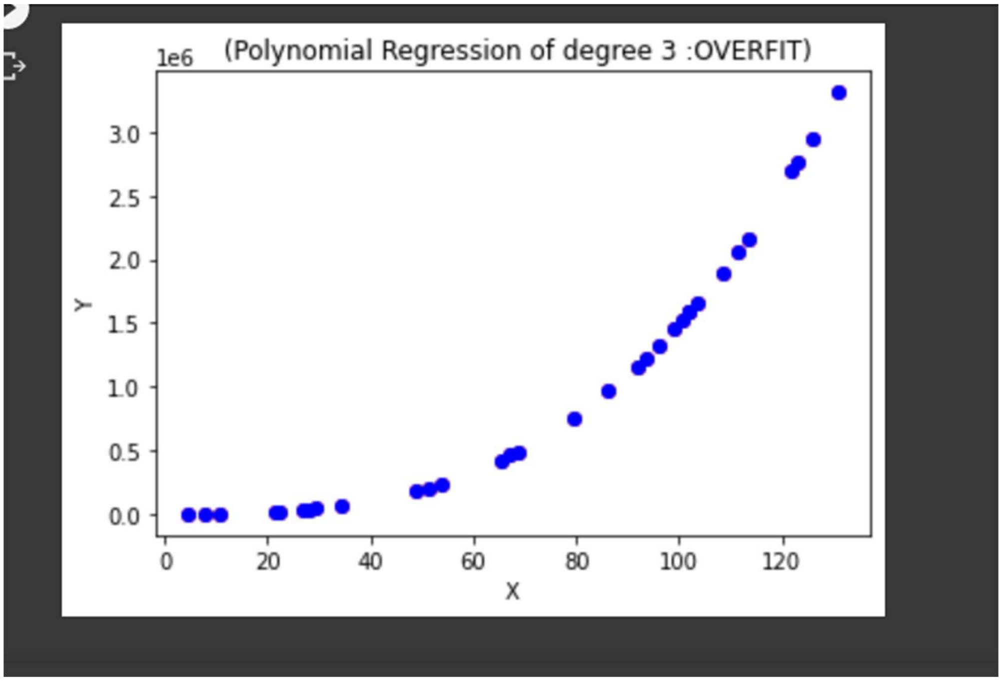


Fig3:-Polynomial Regression Model 3 [OVERFITTING]

HENCE, WE HAVE SEEN FROM THE GRAPH THAT: -

1)IN LINEAR REGRESSION ,DATASET IS CUTTING AT ONLY 2 POINTS, SO ERROR RATE IS VERY HIGH.SO THE MODEL CANT BE USED FOR PREDICTION

2)IN POLYNOMIAL REGRESSION OF DEGREE 2 , ERROR RATE IS NOMINAL, AND PREDICTION WILL BE GOOD AS MAX. POINTS ARE OVERLAPPING .AND THE MODEL LOOKS GOOD

3)IN POLYNOMIAL REGRESSION OF DEGREE=3 ,CONDITION OF OVERFITTING IS CLEARLY VISIBLE AS THERE IS NO RED DOTS VISIBLE WHICH IS ACTUAL DATA SET

HENCE POLYNOMIAL REGRESSION OF DEGREE 2 IS BEST FOR MODEL.

Reference:-

● Ref 1:- https://www.javatpoint.com/machine-learning-polynomial-regression

● Ref 2:- https://www.analyticsvidhya.com/blog/2021/07/all-you-need-to-know-about-polynomial-re gression/#:~:text=Polynomial%20Regression%20is%20a%20form%20of%20Linear%20r egression%20known%20as,also%20badly%20affect%20the%20performance.

Task5:-

POLYNOMIAL REGRESSION

STEPS / PROCEDURES

5.1.PLOTTING SIGMOID FUNCTIONS

● IMPORT LIBRARIES IN THE PYTHON

● PLOTTING SIGMOID FUNCTION IN RANGE(-10,+10)

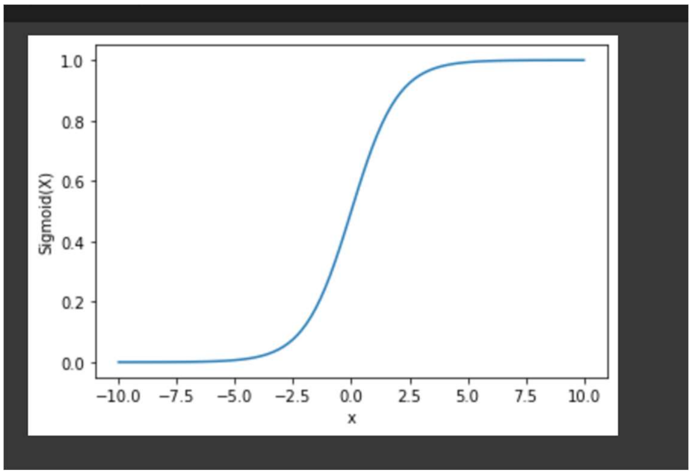


Fig1:- Sigmoid Function

5.2. EXPERIMENT 1:- TO CLASSIFY FROM THE GIVEN GLASS MATERIAL WHETHER IT'S A WINDOW SET OR A NON WINDOW SET USING MAX MIN NORMALISATION

- IMPORT LIBRARIES
- UPLOAD DATA FORMAT FROM LOCAL DRIVE
- CHANGE IT INTO CSV FILE USING PYTHON ALGO

```
Saving glass.data to glass (2).data
       1  1.52101  13.64  4.49  1.10  71.78  0.06  8.75  0.00  0.00.1  1.1
0      2  1.51761  13.89  3.60  1.36  72.73  0.48  7.83  0.00    0.00    1
1      3  1.51618  13.53  3.55  1.54  72.99  0.39  7.78  0.00    0.00    1
2      4  1.51766  13.21  3.69  1.29  72.61  0.57  8.22  0.00    0.00    1
3      5  1.51742  13.27  3.62  1.24  73.08  0.55  8.07  0.00    0.00    1
4      6  1.51596  12.79  3.61  1.62  72.97  0.64  8.07  0.00    0.26    1
..   ...      ...    ...   ...   ...    ...   ...   ...   ...     ...  ...
208  210  1.51623  14.14  0.00  2.88  72.61  0.08  9.18  1.06    0.00    7
209  211  1.51685  14.92  0.00  1.99  73.06  0.00  8.40  1.59    0.00    7
210  212  1.52065  14.36  0.00  2.02  73.42  0.00  8.44  1.64    0.00    7
211  213  1.51651  14.38  0.00  1.94  73.61  0.00  8.48  1.57    0.00    7
212  214  1.51711  14.23  0.00  2.08  73.36  0.00  8.62  1.67    0.00    7

[213 rows x 11 columns]
```

Fig2: uploaded data of Glass

- USE ATTRIBUTES TO THE GIVEN DATA SET
- USE MAX MIN ALGORITHM

- NORMALIZE THE DATA

```
the data after applying  min-max normalization
[[0.         0.43283582 0.43759398 ... 0.         0.         1.        ]
 [0.00469484 0.28358209 0.47518797 ... 0.         0.         1.        ]
 [0.00938967 0.22080773 0.42105263 ... 0.         0.         1.        ]
 ...
 [0.99061033 0.41703248 0.54586466 ... 0.52063492 0.         0.        ]
 [0.99530516 0.23529412 0.54887218 ... 0.4984127  0.         0.        ]
 [1.         0.26163301 0.52631579 ... 0.53015873 0.         0.        ]]
```

Fig3:-Normalised Data

- CLASSIFIED WINDOW VALUE IN THE FORM OF 0 AND 1

```
[-0.19683356  0.2682528   0.09786406  0.82424539  0.01233495  0.26660797
  0.02776092  0.24758773 -0.15478134  0.13815018]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
logistic_meanerror= 0.09345794392523364
```

Fig4:- Window Value in 0,1

5.3. SPLITTING THE DATA INTO TEST AND TRAINING DATA . & APPLYING LOGISTIC REGRESSION USING GRAD ND DESC METHID

- SPLITTING THE DATA INTO THE RATIO 60:40
- FINDING COST FUNCTION
- USE GRADIENT DESCIENT METHOD
- TRAIN THE MODEL USING CALLING SIGMOID FUNCTION, PREDICT FUCTION, LOGLOSS FUNCTION
- THE TRAINED MODEL IS THEN RUN TO GIVE THE OUTPUT, AND THE ERROR AS BELOW: -

```
[5.54886290e-03 4.87269313e-03 5.47445272e-03 1.09659387e-02
 4.52141783e-03 7.37415287e-03 1.11275879e-03 4.83253806e-03
 1.13431866e-05 1.89195845e-03 1.48019265e-02]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
logrithimerrormean= 0.7674418604651171

[136] predict_y_train=[1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
```

Fig5:- Trained Model Data And Error value

5.4:-FIND THE CONFUSION MATRIX AND DISPLAY TEST DATA AND TRAINING DATA GRAPHICALLY AND ROC DIAGRAM

- `array([[10, 1], [ 0, 43]])`

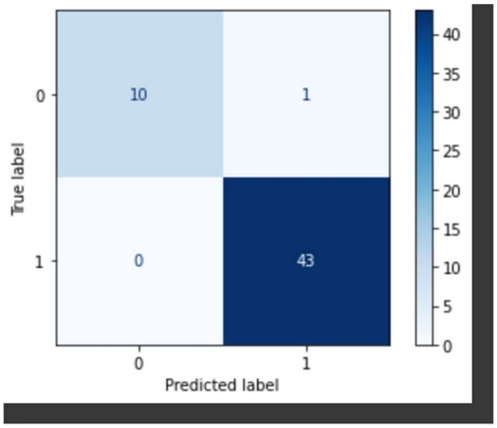
- 

Fig6:-Test Label Vs Predicted Label CONFUSE MATRIX

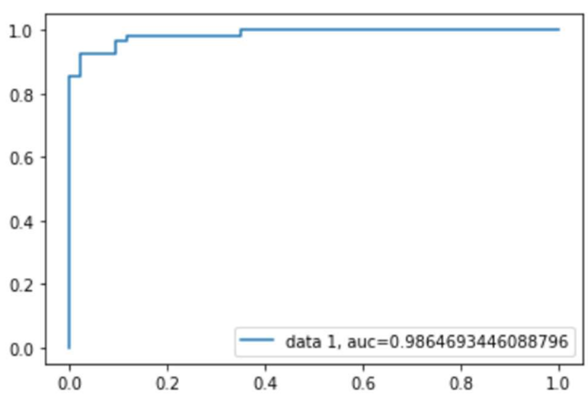


Fig:- ROC Curve with AUC

:-

Reference:-

https://towardsdatascience.com/logistic-regression-using-gradient-descent-optimizer-in-python-485148bd3ff2

https://towardsdatascience.com/machine-learning-polynomial-regression-with-python-5328e4e8a386

<u>COLLAB FOLDER LINK</u>

https://drive.google.com/drive/folders/1H5FW2TZtuhA-C9SQ90dPmrG6TUE6ZF9G?usp=sharing