Name: Anil Bhavanam Honnappa Gowda

Linear Regression:

The dataset given was contained 4 independent variable and 1 dependent variable, since we should not able to use regression function from inbuilt library, only choice was to build a regression function , to build a regression function I used matrix operation to find B values , the formula to find regression for the given dataset is

$$Y=B_0+B_1*X_1+B_2*X_2+B_3*X_3+B_4*X_4$$

Here Y is the target value, to get Y we need x values and B values, since X values are input values, now remaining B values should be find first, to get values the formula is

$$B=(X^iX)^{-i} X^{-i}y$$

So to get B values first we need to use above formula and then after finding the B values, target values(Y) are calculated then they are test with the data and the performance score is calculated

The result is taken a picture and shown below

Untitled1.ipynb

File Edit View Insert Runtime Tools Help All chan

+ Code + Text

```
        y_predict.append(sum)

        return r2_score(y_test,y_pred)
    a=linear_regression(independent_value,dependent_value)
    print('performance score is',a)
```

```
beta variables are [ 0.19208399 -0.10974146 -0.04424045  0.22700138  0.60989412]
performance score is 0.9304223675331595
```

Classification

Untitled1.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

To find beta values

```python
def linear_regression(x,y):
    #X_values=x.drop(labels='flower',axis=1)
    #X_values.insert(loc=0,column='B',value=1) # adding the Y intersept (Bo)
    X_values.to_numpy()
    data_transpose=np.transpose(X_values)
    Xx=data_transpose @ X_values
    Xy=np.dot(data_transpose,y)
    inverse_Xx=np.linalg.inv(Xx)
    beta_values=inverse_Xx@Xy
    print('beta variables are',beta_values) # betas values are found

    x_test=x
    y_test=y
    y_predict=[]
    for i , r in x.iterrows():
        sum=beta_values[0]+(beta_values[1]*r['sepal_length'])+(beta_values[2]*r['sepal_width'])+(beta_values[3]*r['petal_lenth'])+(beta_values[4]*r['petal_width'])
        y_predict.append(sum)

    return r2_score(y_test,y_pred)
a=linear_regression(independent_value,dependent_value)
print(a)
```

```
beta variables are [ 0.19208399 -0.10974146 -0.04424045  0.22700138  0.60989412]
0.9304223675331595
```
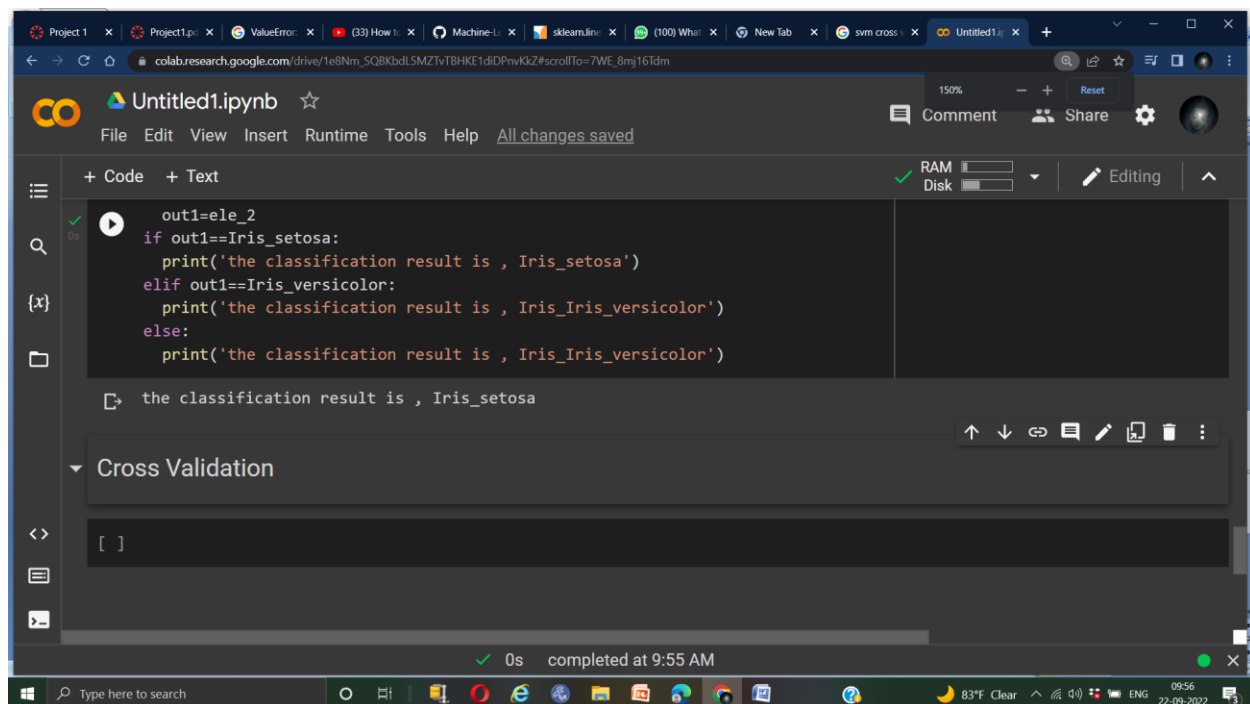
Classification:

For classification I used KNN algorithm, here I took k value as 3 and to classify we need a distance of the values for this and to find this I used Euclidean distance , the formula for 4 independent value is

Euclidean distance = $Sqrt(X^2+Y^2+Z^2+A^2)$

After finding the distance we should sort the distance and we need to check the first 3 of the sorted value and if any of the 2 k values having same dependent value group , then instance also classified as that instance.

The result is shown below: