Linear Regression for Classification

Created By:
Arijit Ganguly
1001871460

1 Problem Statement

The use of iris data set for the prediction of species is a classic example for classification problem. This classification problem needs to be solved by the Linear Regression which is a supervised learning problem. A linear regression algorithm needs to be developed that can predict the species of input provided to the algorithm with almost certainty (close to 100% accuracy).

2 Data

The Iris Data set contains 5 columns – Sepal Length, Sepal Width, Petal Length, Petal Width and Species.

We use the first 4 columns as our features – Sepal length, Sepal width, Petal length and Petal width. All the columns for features are float values. The values in these columns make up the A matrix.

Last column is going to be used as the labels. It states the species of the plant which have the corresponding features. The data set contains 3 species – Iris setosa, Iris virginica and Iris versicolor.

Data Distribution based on species:

Species	Number of Records
Iris Setosa	50
Iris Virginica	50
Iris Versicolor	50

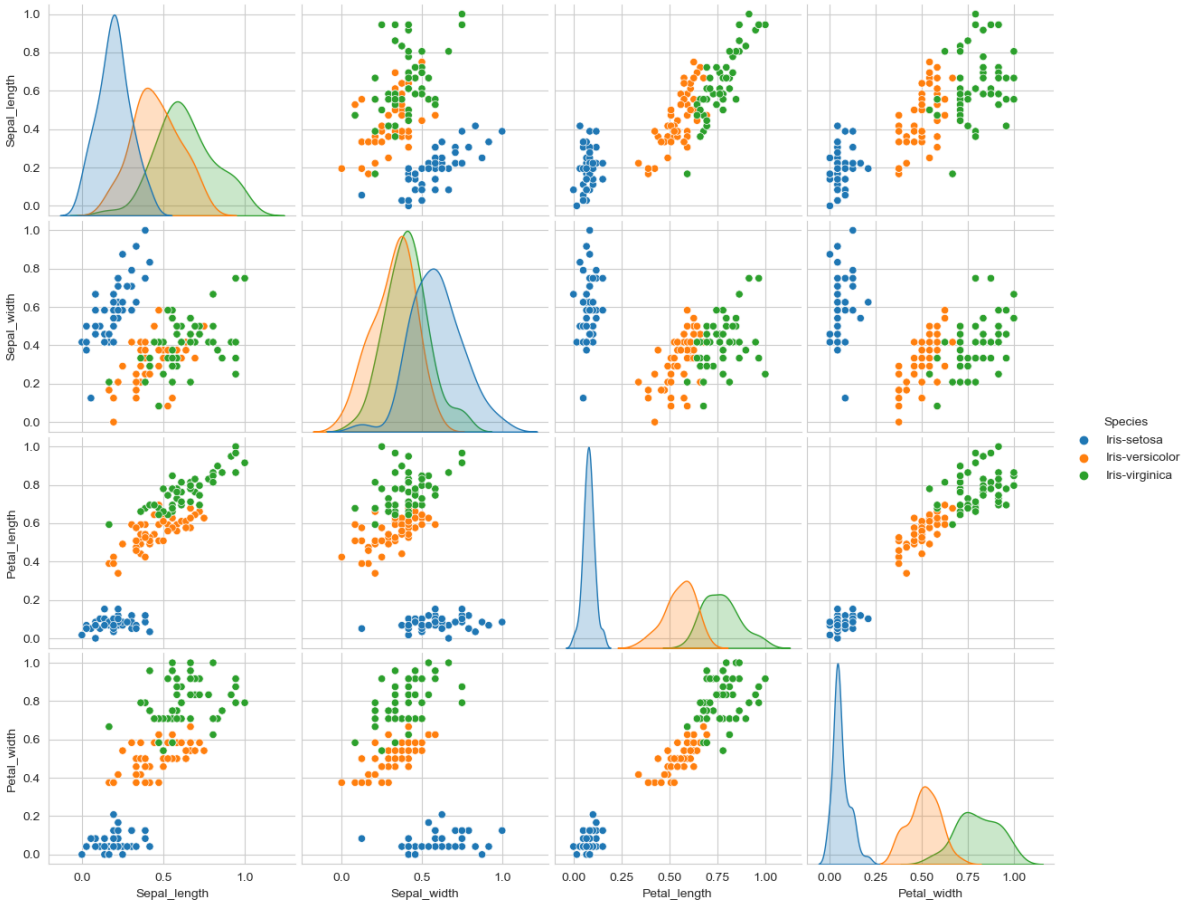
Information on the data received from method pandas.info()

Index number	Column	Non-Null Count	Data Type
0	Sepal Length	150	Float64
1	Sepal Width	150	Float64
2	Petal Length	150	Float64
3	Petal Width	150	Float64
4	Species	150	category
5	Species_cat	150	Int8

Data Description (Received from method pandas.describe())

	Sepal Length	Sepal Width	Petal Length	Petal Width
count	150.000000	150.000000	150.000000	150.000000
mean	0.428704	0.439167	0.467571	0.457778
std	0.230018	0.180664	0.299054	0.317984
min	0.000000	0.000000	0.000000	0.000000
25%	0.222222	0.333333	0.101695	0.083333
50%	0.416667	0.416667	0.567797	0.500000
75%	0.583333	0.541667	0.694915	0.708333
max	1.000000	1.000000	1.000000	1.000000

Data visualization:



3 Strategy

The strategy for implementing a solution is to use k-folds cross validation to create k number of bins in the data and then train and test data on each of these bins which will give **Beta** for each the bins. To reduce the overfitting, we will find the mean of each Beta value generates for each feature as **Beta Mean**.

3.1 Method Used:

3.1.1 Fit Method

The fit method is the method use for obtaining the **Beta** values from the matrix **A**, which is the matrix created from the records in the features for each bin and **Y**, which is the label encoded values for the species.

Using the formula below, we get **Beta** values:

$$\mathbf{B = (A^T A)^{-1} A^T Y}$$

3.1.2 Predict Method

The predict method uses the B values and the input matrix **A** values to be used to get predicted **Y** values. As the values returned are in float values, we are rounding them up to the nearest integers and converting negatives to positives.

Using the formula below, we get **Y** values:

$$\mathbf{Y = AB}$$

3.1.3 Label Encoding Method

Label Encoding used to change the **Y** values to convert to a categorical numerical value so we can predict them. As we are using mathematical functions to predict new values, we need the species categories represented by a numerical value. For my code, I convert the **Species** column in my data as the category and use it to create a new column **Species_cat** to store the label encoded values.

3.2 Process

The process that the code follows is mentioned below:

1) The data is input from the file “iris.data” into a Pandas dataframe.
2) As the column names, have not been mentioned in the dataset, I have added the respective column names to make sense of the data.
3) We normalize the data to get all numerical values in the range of 0 to 1 using the function **normalize_data()**.
4) After normalizing the data, we need to label encode the species column to convert the Y values to the numerical values.
5) Initialize a **B_list** variable to store all the Beta values from the bins which concludes the initialization of our LinearRegression object.
6) Program will ask input for the Training and Test split. Please provide a float value representing the percentage. (Example: 0.3 = 30% which means 30% of data will be used for testing whereas 70% data will be used for training; **Recommended value is 0.2**)
7) Program will ask for the number of bins to be created for cross validation. This will split the data and, training and testing will run for the data that many times generating that number Beta value sets. (**Recommemded value is 5 bins**)
8) After that we describe the data by calling **describe_data()** for the user which is basic information about the iris data set.

9) Then we try to visually understand the data by calling the function **visualize_data()**
10) After this the cross validation function (**cross_validation()**) is called. This function splits the data into the 3 categories available in the data, then initializes the bins required.
11) Then from each of the categories, it starts storing equal number of records in each bin. For each bin, we split the data in train and test.
12) Once all the data has been initialized in bins, we take each bin and run the **fit** method and using the Beta values generated for that bin, we call the **predict** method. Each set of Beta values generated are stored in the variable **B_list**.
13) We store the Y values and the accuracy values in separate variables.
14) After all the beta values are generated, we take the mean of the beta values to reduce overfitting.
15) After receiving the mean Beta values we check the accuracy against the full dataset, which comes to be 98%.

4 Results

For input values:

Train Test Split = 20% or 0.2 and bins = 5, we get

```
Beta values generated for each bin are:
[[-1.268592834736018 -0.11101505113072652 2.751126240937218
  0.5732323348553314]
 [0.24686503448363528 -0.3538527412909931 0.42015013202833384
  1.9003837359961322]
 [-0.5799706579494273 -0.24490419102748395 0.8395735448138235
  2.081646986604492]
 [0.005588756183921406 -0.2668003714291395 2.056161265159612
  0.32995345219778427]
 [-0.38218153312033376 -0.3505588323233444 0.4953873302159345
  2.2353960564459237]]
Mean Beta values generated:
[-0.3956582470276445 -0.2654262374403375 1.3124797026309842
 1.4241225132199324]
Accuracy of the algorithm on the existing dataset is:
98.0%
```

Train Test Split = 30% or 0.3 and bins = 5, we get

```
Beta values generated for each bin are:
[[-1.2059126661707427 -0.1100846681405141 2.084551770122379
  1.169787423936563]
 [0.2831200957404447 -0.3604116022181369 0.3405435163816404
  1.951764317249471]
 [-0.1767882252854256 -0.3609039023841927 0.4303478444341014
  2.212740415138629]
 [0.247907736035398 -0.35813679676865207 1.9697921779310301
  0.25021271299334991]
 [-0.33768251079004297 -0.3725169002014973 0.5195499170713832
  2.1688971560697876]]
Mean Beta values generated:
[-0.2378711140940737 -0.31241077394259864 1.0689570451881067
 1.55068040507756021]
Accuracy of the algorithm on the existing dataset is:
96.66666666666667%
```

Train Test Split = 30% or 0.3 and bins =6, we get

```
Beta values generated for each bin are:
[[-1.634065781811415 0.024553360465966068 3.218137706635633
  0.3348527962806588]
 [0.2773200420537831 -0.3801015783592244 0.8067198735286343
  1.5078581739620995]
 [0.0523324887101797 -0.313828243816373 2.0234549302697005
  0.2461084363121263]
 [-1.493178699951947 0.040581726147498304 1.6169021139368551
  1.988275718722873]
 [0.20176394058298974 -0.32085662062636655 1.9959150942727093
  0.24299860607901863]
 [-0.9105823078014621 -0.33476765542354325 0.5375201353309949
  2.5096753041002815]]
Mean Beta values generated:
[-0.5844017197029786 -0.21406983526867382 1.6997749756624214
 1.1382948392428431]
Accuracy of the algorithm on the existing dataset is:
98.0%
```

5 References

5.1 My github repository for the code and project report

Github

5.2 For Iris dataset visualization and K-means reference materials

https://www.kaggle.com/khotijahs1/k-means-clustering-of-iris-dataset/notebook