Megha Vijendra

Machine Learning

Language used: Python
[illegible]
Libraries used: NumPy, pandas, matplotlib

Overview

A Linear Regression model implemented from scratch using forward propagation and backward propagation to train the regression model.

1. Exploring the Dataset

The dataset is obtained from the UCI Machine Learning Repository and contains 3 classes of 50 instances each, where each class refers to a type of iris plant. The file is comma separated and the fields/attributes in each record are as follows:

Features

- Sepal Length in cm
- Sepal Width in cm
- Petal Length in cm
- Petal Width in cm

Label

The class of the plant:

- Iris Setosa
- Iris Versicolour
- Iris Virginica

Statistics of the dataset

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

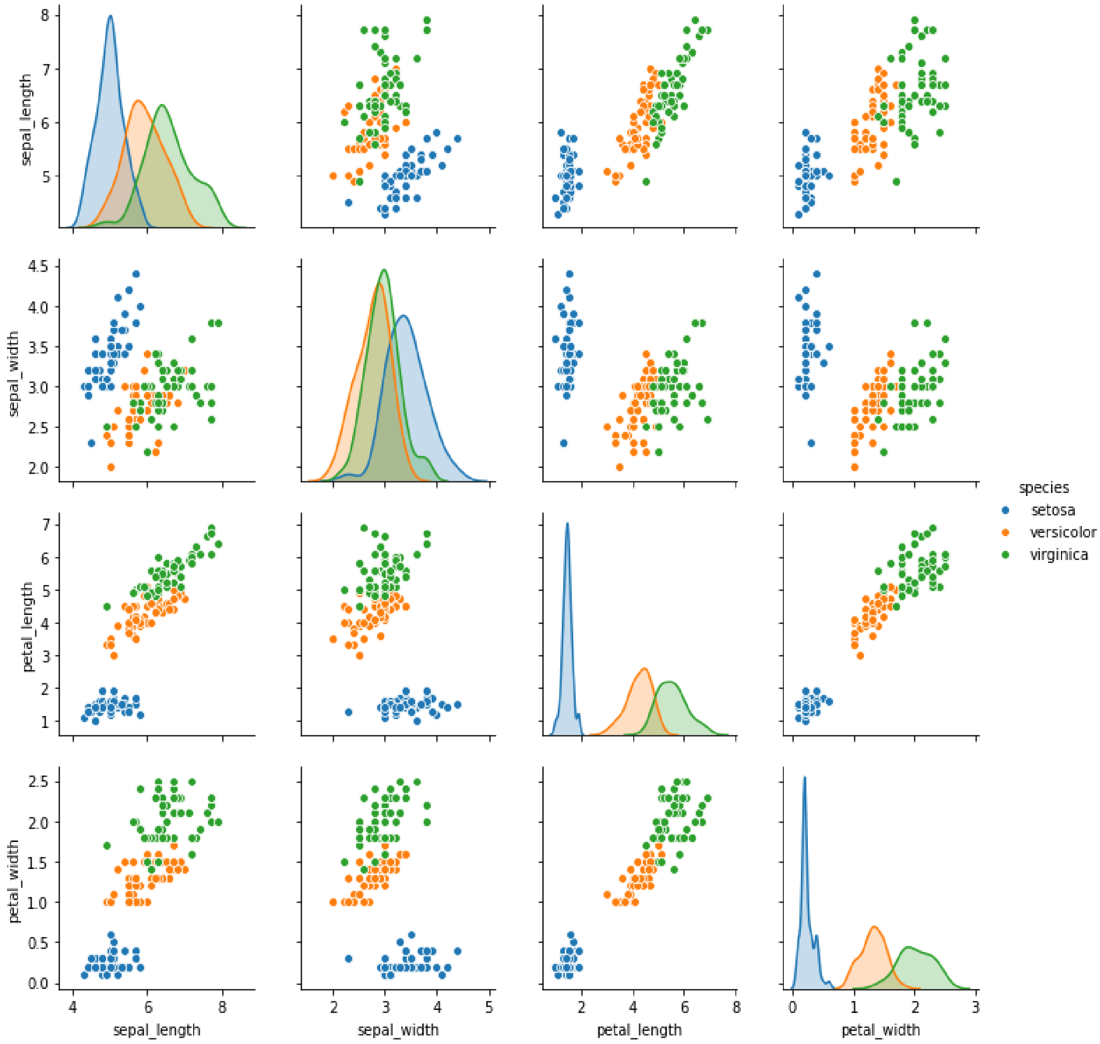
Sample Data

sepal length	sepal width	petal length	petal width	class
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa

Visualizing the dataset

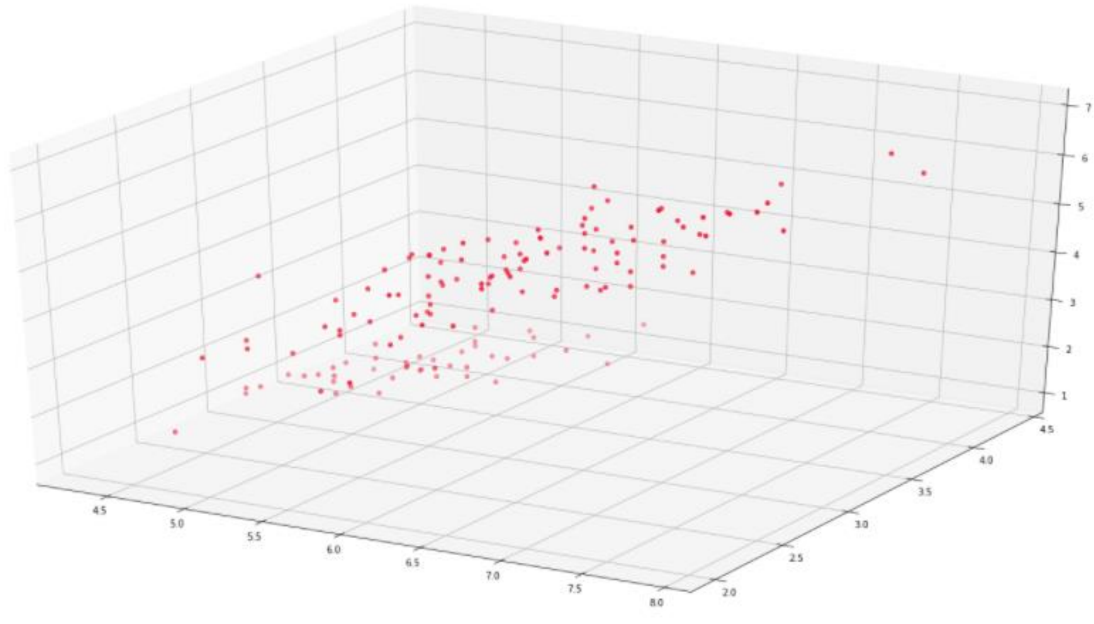
Using seaborn pair plot to see the bivariate relation between each pair of features:

```
Out[30]: <seaborn.axisgrid.PairGrid at 0x12cac12a788>
```



From the above, we can see that **Iris-Setosa** is separated from both other species in all the features.

Visualizing the scores of the dataset using the scatter plot:



2. Implementation of the Linear Regression Model

Here the problem statement is to predict the species of the iris flowers.

The steps and the methods used in the implementation are as follows:

STEP 1: Data Pre-processing:
Method used: def data_preprocessing ()
Here we are importing the iris data set and will be creating a new column called the target which is a dictionary which contains:

- 0 for Iris Setosa
- 1 for Iris Versicolour
- 2 for Iris Virginica

STEP 2: Shuffling the data:
Method used: def data_shuffling (df)
As we can see from the dataset that the data is grouped according to the species. Hence, we are shuffling the dataset to get an even split of the data.

STEP 3: Splitting the data:
Method used: def data_split (df, X, Y)
Here we are going to split the data for X_train, X_test, Y_train and Y_test in 80:20 ratio. 80 for the training set and 20 for the test set

STEP 4: Training the Linear Regression model:
Method used: def train (X_train, Y_train)
To train the model we use the slope equation y = mx + b or y = wx + band use the following methods in the given order:

1. **def init_parameters(lenw)**
   Here we initialize w and b to random values for the equation y = mx+b where m or w is weight or bias and it is going to be initialized only once for the model.

2. **def forward_prop(X, w, b)**
   Here we use mathematical y= wx+ b, to propagate the data to the next function. Also gives us the y_hat value.

3. **def cost_function(y_hat,y)**
   We find the Euclidean distance between expected from predicted value,what it means is that we find the difference between predicted values and the original y values and sum them up. Then we find the average and return it. The returned value is the cost. The cost will go down gradually, for each slope and data.

4. **def back_prop(X,y,y_hat)**
   Here we find the delta values for dw and db. y is actual value and y_hat predicted value based on these values and the X we calculate the differences and do delta changes for weight and bias.

5. **def gradient_desent(w,b,dw,db,lr)**
   It finds the optimum value for theta parameters so that the cost decreases. Based on learning rate (learning rate* delta change), multiply and subtract it and changes w and b**.**

The train method returns the constants w, b which are the trained constants. So that we can predict the test set using these constants.
The result of running the train method is as follows:

```
C:\Users\Megha\AppData\Local\Programs\Python\Python38-32\python.exe "C:/Users/Megha/Desktop/Notes/Machine Learning/Project1/Project1_mxv6938.py"
Training and predicting results for the test data using the trained model:
Epoch number - 0 : Loop no - 120 : Cost : 0.11359013615634302
Epoch number - 2 : Loop no - 120 : Cost : 0.1388704177647513
Epoch number - 4 : Loop no - 120 : Cost : 0.1553356193168382
Epoch number - 6 : Loop no - 120 : Cost : 0.16526851218766997
Epoch number - 8 : Loop no - 120 : Cost : 0.17113285224429498
Training constants
 w:[-0.37961826438646895 0.38701751621842384 0.3189195890181432
 0.7719148978547007] b:-0.08331921149574924
```

STEP 5: Predicting the values of the test set using the Linear Regression model:
Method used: def predict (X_test,Y_test, w, b)
Here we are going to use the trained model to predict the values for the X_test and the Y_test and for every predicted value == actual value the hit will become more. The hit is used to calculate the accuracy of the predict function.
The results of the predict function are shown below:

```
Predicted Value - 1.0 : Actual Value - 1
Predicted Value - 1.0 : Actual Value - 1
Predicted Value - 1.0 : Actual Value - 1
Predicted Value - 2.0 : Actual Value - 2
Predicted Value - 2.0 : Actual Value - 2
Predicted Value - 2.0 : Actual Value - 2
Predicted Value - 1.0 : Actual Value - 1
Predicted Value - 1.0 : Actual Value - 1
Predicted Value - -0.0 : Actual Value - 0
Predicted Value - 0.0 : Actual Value - 0
Predicted Value - -0.0 : Actual Value - 0
Predicted Value - -0.0 : Actual Value - 0
Predicted Value - 1.0 : Actual Value - 1
Predicted Value - 1.0 : Actual Value - 1
Predicted Value - -0.0 : Actual Value - 0
Predicted Value - 1.0 : Actual Value - 1
Predicted Value - 2.0 : Actual Value - 2
Predicted Value - 1.0 : Actual Value - 2
Predicted Value - -0.0 : Actual Value - 0
Predicted Value - 1.0 : Actual Value - 1
Predicted Value - 1.0 : Actual Value - 1
Predicted Value - 1.0 : Actual Value - 1
Predicted Value - 1.0 : Actual Value - 1
Predicted Value - -0.0 : Actual Value - 0
Predicted Value - -0.0 : Actual Value - 0
Predicted Value - 2.0 : Actual Value - 2
Predicted Value - 2.0 : Actual Value - 2
Predicted Value - 1.0 : Actual Value - 1
Predicted Value - 2.0 : Actual Value - 2
Predicted Value - 2.0 : Actual Value - 2
Accuracy = 96.66666666666667%
```

From the above image we can see the actual and predicted values for the given test set using the training constant and we can see that the accuracy using this model is 96.6 percentage.

STEP 6: Cross validation for trained Linear Regression model:
Method used: def cross_validation(X,Y)
Here we are going to test the trained model based by dividing the original dataset into k chunks. The testing dataset is selected using the method **data_sampler(X,Y, i,fold),** and the training set is the difference of the original iris dataset and the newly created testing set. This goes on, till the k loop ends and the mean of the accuracies is calculated.

The method **def data_sampler(X,Y, i,fold)** is used to sample A fold for dataset. So that we can train test and verify the accuracy of the model.
The accuracy of the model for the following folds is shown in the figure below:

0-fold is 0 to 30

1-fold is 30 to 60

4-fold is 120 to 150

```
Predicted Value - 2.0 : Actual Value - 2
Predicted Value - 2.0 : Actual Value - 2
Predicted Value - 1.0 : Actual Value - 1
Predicted Value - 2.0 : Actual Value - 2
Predicted Value - 2.0 : Actual Value - 2
{0: 96.66666666666667, 1: 100.0, 2: 93.33333333333333, 3: 90.0, 4: 96.66666666666667}

Process finished with exit code 0
```

REFERENCES:

[1]: https://machinelearningmastery.com/k-fold-cross-validation/

[2]: https://www.kaggle.com/arshid/iris-flower-dataset/notebooks

[3]: https://medium.com/we-are-orb/multivariate-linear-regression-in-python-without-scikit-learn-7091b1d45905