Name : Bhavani Sai Shriya Anumala

Mav ID: 1001870184

Project 1 - Report

Problem Statement:

In this project I have imported the necessary libraries and have split the data into two different datasets one is the training set and the other is the test in the ratio 80:20 where the training data is 80% and the test data is 20% of the entire iris.data dataset. Later, I trained the model using linear regression.

Further, I did the classification using the formula of **Least Squares Estimator** given below and have generated the Beta matrix to find the best fit over the generated trained linear regression model.

$$\hat{\beta} = (\mathrm{A}^T\mathrm{A})^{-1}\mathrm{A}^T\mathrm{Y}$$

The performance and the accuracy of the model is calculated by using the cross validation.

By using the stratified fold validation it is optimal and also generates multiple and a variety of combinations of the test data and training data from the given iris.data.

Coding:

Here, I have used the pandas pd.read_csv to read the iris.data and since the data did not have any header and had comma separated values unlike a table I have used the below code.

```
# Read iris data
data = pd.read_csv("iris.data", header = None , sep=',')
data.head()
```

	0	1	2	3	4
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Later, I have divided the data into training data and test data where the training data is 80% and the test data is 20% of the given dataset.
The classes have been assigned value since the class value is categorical.
The categories present in the data are Iris-setosa, Iris-virginica, Iris-versicolor, where Iris-setosa is given the value 0, Iris-virginica is given 1 and Iris-versicolor is given 2.
This was meant to be done as we cannot do matrix calculations using string data.

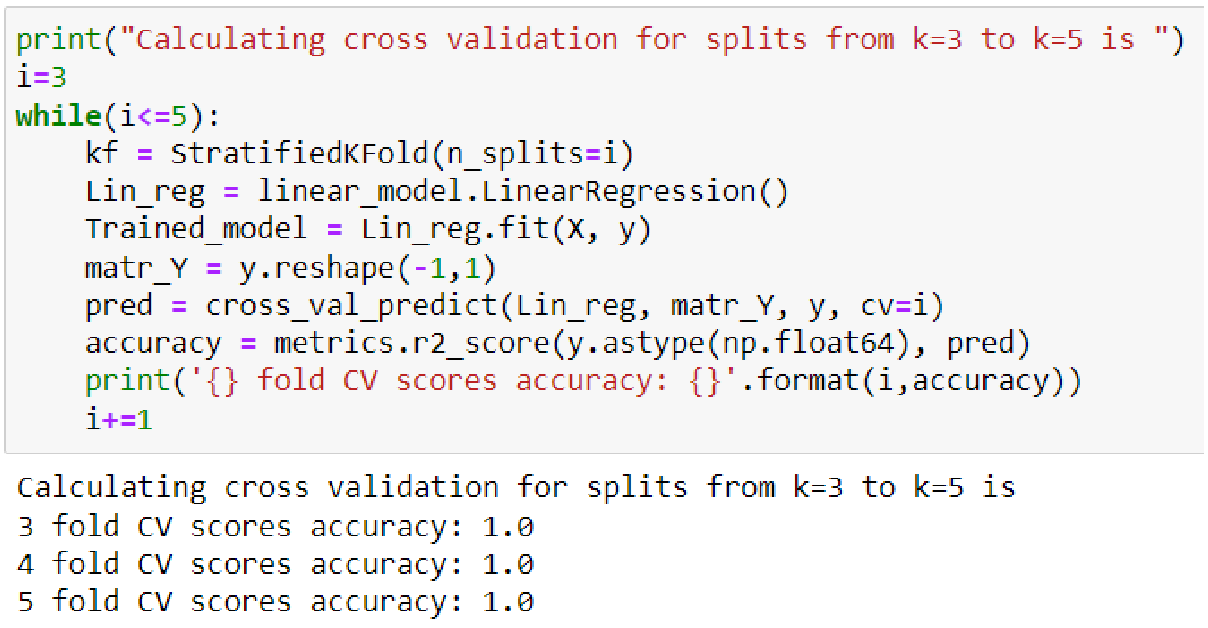
The best first matrix i.e. the beta cap matrix is found using the formula mentioned above.

```
matrA = X
Tmatr_A = np.transpose(matrA)
part1 = np.dot(Tmatr_A,matrA)
Inv_part1 = np.linalg.inv(part1)
part2 = np.dot(Inv_part1,Tmatr_A)
LSE = np.dot(part2,y)
```

```
print("Beta Cap martix for trained model")
print(LSE)
```

```
Beta Cap martix for trained model
[0.2713189453696171 -0.46653097481492944 0.3934858164439222
 -0.5491398267216205]
```

The ultimate task was to do the cross validation and I chose stratified K fold validation. I found the accuracy scores for three different K values starting from 3 to 5 and found the accuracy score to be 1.0 in all the cases.



```python
print("Calculating cross validation for splits from k=3 to k=5 is ")
i=3
while(i<=5):
    kf = StratifiedKFold(n_splits=i)
    Lin_reg = linear_model.LinearRegression()
    Trained_model = Lin_reg.fit(X, y)
    matr_Y = y.reshape(-1,1)
    pred = cross_val_predict(Lin_reg, matr_Y, y, cv=i)
    accuracy = metrics.r2_score(y.astype(np.float64), pred)
    print('{} fold CV scores accuracy: {}'.format(i,accuracy))
    i+=1
```

```
Calculating cross validation for splits from k=3 to k=5 is
3 fold CV scores accuracy: 1.0
4 fold CV scores accuracy: 1.0
5 fold CV scores accuracy: 1.0
```

Conclusion:

In conclusion, I found that the accuracy was 1.0 for all the k values. This means that the regression model is more accurate.