CSE 6363 MACHINE LEARNING
SINDOORA RAVIKUMAR MURTHY
1001862126

Project#	: 01
Problem statement	: Multivariate regression to predict flower species
Programming language	: Python3
Filename	: sindoora_MultiRegression.py
Libraries	: numpy, seaborn, pandas, matplotlib
Compilation	: python3 sindoora_MultiRegression.py

Data:

The IRIS dataset consists of 4 feature vectors: Sepal_length, Sepal_width, Petal_length and Petal_width that decide the species class that each particular flower belongs to.

Sepal_length	Sepal_width	Petal_length	Petal_width	Species
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
5.0	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5.0	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
5.4	3.7	1.5	0.2	Iris-setosa
4.8	3.4	1.6	0.2	Iris-setosa
4.8	3.0	1.4	0.1	Iris-setosa
4.3	3.0	1.1	0.1	Iris-setosa
5.8	4.0	1.2	0.2	Iris-setosa
5.7	4.4	1.5	0.4	Iris-setosa
5.4	3.9	1.3	0.4	Iris-setosa
5.1	3.5	1.4	0.3	Iris-setosa
5.7	3.8	1.7	0.3	Iris-setosa
5.1	3.8	1.5	0.3	Iris-setosa
5.4	3.4	1.7	0.2	Iris-setosa

Fig: IRIS data (original data)

Process:

1. *Training the model using linear regression:*
   Data set: IRIS.csv
   Approach: Multivariate regression.

Method: The nominal values 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica' are replaced with the values 1 ,2, 3 respectively. This lets the regression model interpret them. After replacing, the nominal values, a pairplot is plotted to show the datasets relationship with each independent value.

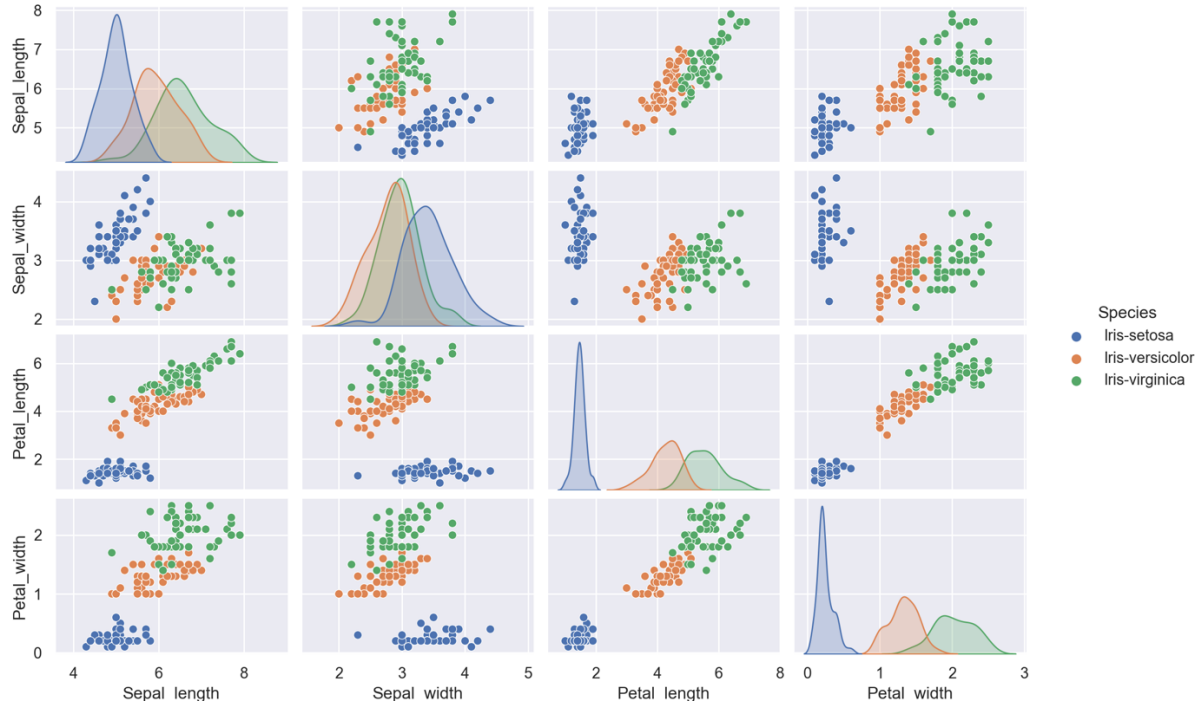

Fig: Relationship between the features.

Then using the least square estimation of multiple regression, the coefficients (β) are derived. The following equation is used to calculate co-efficients:

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

where, A -> data matrix, Y -> species values (1,2,3)

2. *Using the trained model to do the classification:*

As multivariate regression models are not designed to classify (as they return continuous values), the values returned are rounded up or down. First 4 columns of the data are multiplied with $\hat{\beta}$ to get the "predicted" value for each row.
E.g.: If the model returns 1.49, it's rounded down and classified to be of class 1. If the model returns 1.55, it's rounded up and classified as 2.

	Sepal_length	Sepal_width	Petal_length	Petal_width	Species	predicted
3	5.4	3.4	1.5	0.4	1	1.0
7	6.3	2.3	4.4	1.3	2	2.0
11	6.0	3.4	4.5	1.6	2	2.0
15	6.1	2.8	4.7	1.2	2	2.0
19	6.2	2.8	4.8	1.8	3	3.0
23	7.3	2.9	6.3	1.8	3	3.0
27	6.3	3.3	4.7	1.6	2	2.0
31	6.9	3.1	4.9	1.5	2	2.0
35	7.7	2.8	6.7	2.0	3	3.0
39	6.9	3.1	5.4	2.1	3	3.0
43	6.7	3.0	5.0	1.7	2	3.0
47	5.4	3.9	1.3	0.4	1	1.0
51	5.4	3.4	1.7	0.2	1	1.0
55	6.2	3.4	5.4	2.3	3	3.0
59	5.4	3.9	1.7	0.4	1	1.0
63	6.1	3.0	4.9	1.8	3	3.0
67	6.7	3.1	4.4	1.4	2	2.0
71	6.3	2.5	4.9	1.5	2	2.0
75	7.2	3.0	5.8	1.6	3	3.0
79	5.0	3.4	1.6	0.4	1	1.0

Fig: "predicted" column added and calculated using $\hat{\beta}$

3. *k-fold cross-validation:*

   For cross validation, the original dataset is split into k sets. The testing dataset is sequentially selected, 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.

	Sepal_length	Sepal_width	Petal_length	Petal_width	Species	predicted	accurate
3	4.8	3.0	1.4	0.1	1	1.0	1
7	6.3	2.8	5.1	1.5	3	2.0	0
11	6.4	2.8	5.6	2.1	3	3.0	1
15	4.4	3.0	1.3	0.2	1	1.0	1
19	6.3	2.7	4.9	1.8	3	3.0	1
23	6.2	2.2	4.5	1.5	2	2.0	1
27	5.5	2.4	3.7	1.0	2	2.0	1
31	6.1	3.0	4.6	1.4	2	2.0	1
35	5.0	3.2	1.2	0.2	1	1.0	1
39	6.6	3.0	4.4	1.4	2	2.0	1
43	6.9	3.1	5.4	2.1	3	3.0	1
47	7.7	3.0	6.1	2.3	3	3.0	1
51	6.7	3.3	5.7	2.1	3	3.0	1
55	5.0	3.0	1.6	0.2	1	1.0	1
59	6.1	2.9	4.7	1.4	2	2.0	1
63	5.5	4.2	1.4	0.2	1	1.0	1
67	5.7	4.4	1.5	0.4	1	1.0	1
71	7.6	3.0	6.6	2.1	3	3.0	1
75	5.4	3.0	4.5	1.5	2	2.0	1
79	5.0	3.3	1.4	0.2	1	1.0	1

Fig: "accurate" column added and entries are 0 or 1 based on whether the prediction is correct

4. *Results:*

   The model returns 3 accuracies at k =3,5,14, because the dataset is randomized at the beginning. Though they are different accuracies, they are close.
   At k =15, the accuracy steadies at 96.00000000000001%.

```
sindooramurthy@MacBook-Pro Assignment 1 % python3 sindoora_MultiRegression.py
Performance with k=3 folds = 95.33333333333333%

Performance with k=4 folds = 95.99928876244667%

Performance with k=5 folds = 96.0%

Performance with k=6 folds = 96.0%

Performance with k=7 folds = 96.04205318491033%

Performance with k=8 folds = 95.94298245614034%

Performance with k=9 folds = 96.03758169934639%

Performance with k=10 folds = 96.0%

Performance with k=11 folds = 95.90409590409593%

Performance with k=12 folds = 95.94017094017094%

Performance with k=13 folds = 95.9207459207459%

Performance with k=14 folds = 96.03896103896105%

Performance with k=15 folds = 96.00000000000001%

Performance with k=16 folds = 96.04166666666667%
```

Fig: Performance measure with different k values for cross validation

Functions used:

- read_data(): function is used to read the dataset from the folder and replace the Nominal values with some quantitative values. It also randomizes the dataset.
  Returns: data (processed, randomized, IRIS data set)

- train_model(train_data): function is used to train the model. It returns the coefficient $\hat{\beta}$ of the trained multivariate regression model. It uses the least squares estimator method to calculate the coefficients Returns: betacap (the co-effecients).

- test_model(test_data): function is used to check the accuracy of the trained model.
  Returns: accuracy (the percentage of accuracy achieved)

- split_data(lst,k): function returns the dataset after dividing it into the number of sets that the user specifies i.e., based on the value of 'k'.
  Returns: list of 'k' datasets

- kfold_CrossValidation(data,k): function the train_model and test_model functions, finds the average of the accuracies and returns the mean to the user.