

Amartejas Manjunath

1001742606

axm2606

Machine Learning

Project 1

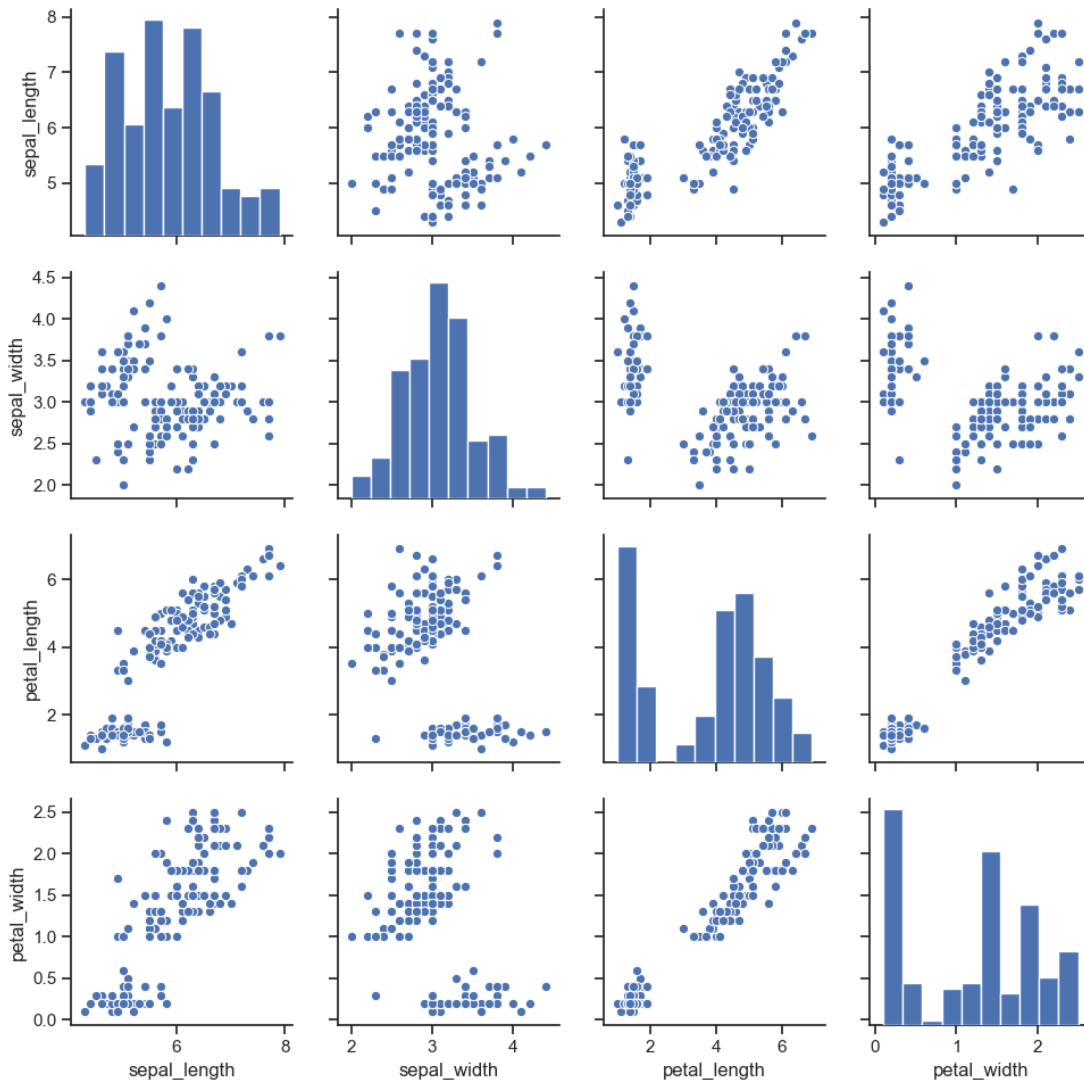
1.Training a model using linear regression

The dataset(iris), has four feature vectors. So, a multivariate regression approach was chosen.

Firstly the nominal values “setosa”, ‘versicolor’, ‘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.

A multi collinearity test is not conducted, as this might lead to dropping of attributes.



Then using the least square estimation of multiple regression the coefficients are derive

$$(X^T X)^{-1} X^T Y = B$$

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.

Eg: if the model returns 1.49, it's rounded down and classified to be of class 1. If the model returns 1.78, it's rounded up and classified as 2.

```

localhost:54005 /home/dapperpig/Documents/Studies/machine-learning
sepal length sepal width petal length petal width species
0 6.3 3.3 4.7 1.6 2
1 5.0 2.3 3.3 1.0 2
2 6.9 3.2 5.7 2.3 3
3 5.2 2.7 3.9 1.4 2
4 4.5 2.3 1.3 0.3 1
.. ..
145 6.0 2.2 5.0 1.5 3
146 6.1 2.8 4.7 1.2 2
147 5.9 3.2 4.8 1.8 2
148 6.2 2.9 4.3 1.3 2
149 5.0 3.0 1.6 0.2 1

[150 rows x 5 columns]

```

```

47 test_data.loc[:, 'correct'] = np.where(test_data['species'] == test_data['predicted'], 1, 0);
48 print(test_data)
You are a few seconds and a few uncommitted changes behind

```


	sepal length	sepal width	petal length	petal width	species	predicted	correct
146	6.9	3.1	4.9	1.5	2	2.0	1
2	5.4	3.0	4.5	1.5	2	2.0	1
7	5.0	3.4	1.5	0.2	1	1.0	1
12	5.6	3.0	4.1	1.3	2	2.0	1
17	6.3	3.3	4.7	1.6	2	2.0	1
22	6.2	2.9	4.3	1.3	2	2.0	1
27	6.4	2.9	4.3	1.3	2	2.0	1
32	6.4	3.1	5.5	1.8	3	3.0	1
37	5.8	2.7	5.1	1.9	3	3.0	1
42	4.4	3.0	1.3	0.2	1	1.0	1
47	6.3	2.5	4.9	1.5	2	2.0	1
52	4.8	3.0	1.4	0.1	1	1.0	1
57	4.6	3.2	1.4	0.2	1	1.0	1
62	6.8	3.2	5.9	2.3	3	3.0	1
67	4.8	3.4	1.6	0.2	1	1.0	1
72	6.8	2.8	4.8	1.4	2	2.0	1
77	5.6	2.5	3.9	1.1	2	2.0	1
82	6.5	3.2	5.1	2.0	3	3.0	1
87	7.3	2.9	6.3	1.8	3	3.0	1
92	6.0	2.2	5.0	1.5	3	2.0	0
97	7.6	3.0	6.6	2.1	3	3.0	1
102	7.7	3.8	6.7	2.2	3	3.0	1
107	6.3	2.3	4.4	1.3	2	2.0	1
112	5.0	3.6	1.4	0.2	1	1.0	1
117	6.0	3.4	4.5	1.6	2	2.0	1
122	6.7	3.3	5.7	2.1	3	3.0	1
127	5.8	2.8	5.1	2.4	3	3.0	1
132	5.4	3.4	1.7	0.2	1	1.0	1
137	5.9	3.2	4.8	1.8	2	3.0	0
142	4.9	2.4	3.3	1.0	2	2.0	1
147	6.4	3.2	5.3	2.3	3	3.0	1

3. Using cross-validation

For cross validation, I divided the original dataset into k chunks. The k is currently hardcoded but can be changed through the code. 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.

4. Results

My model returns 3 accuracies at $k=3,5,10$. This might be because I am randomizing the dataset in the very beginning. Though they are different accuracies, they are close. (95.33333333333334%).

 Anaconda Prompt

```
(base) C:\Users\Amar\Desktop>python pro.py
Accuracy for a multivariate regression with 3 folds is 95.33333333333333%
```

```
(base) C:\Users\Amar\Desktop>python pro.py
Accuracy for a multivariate regression with 15 folds is 96.00000000000001%
```

At $k=15$, the accuracy steadies at 96.00000000000001%.

Methods:

There are 6 methods in the python file.

1. **def read_replace():** The read_replace function is used to read the dataset from the folder and replace the Nominal values with some quantitative values. It also randomizes the dataset.

parameters returned: **data** (the processed, randomized, iris data set) # train_model function.
This function is used to train the model. It returns the coefficients of the trained multivariate regression model.

2. **def kfold(data,k):** The kfold function is used to implement the k-fold cross validation. It calls the train_model and test_model functions, finds the average of the accuracies and prints it to the user.

parameters accepted: data(the cleaned iris data), k (the number of folds for cross validation).

3. **def train_model(train_data):** This function is used to train the model. It returns the coefficients of the trained multivariate regression model. It uses the least squares estimator method to calculate the coefficients

parameters accepted: train_data(the data that is derived from kfold after being separated during cross validation).

parameters returned: betacap(the coefficients).

4. **def test_mode(test_data):** This function is used to check the accuracy of the trained model.

parameters accepted: test_data(the left over data that is derived from kfold after being separated during cross validation)

parameters returned: accuracy (the percentage of accuracy achieved).

5. **def chunkify(lst,n):** This function returns the dataset after dividing it into the number of chunks that the user needs.

parameters accepted: lst,n (lst is the dataframe to be divided and n is the number of ways it has to be divided).

parameters returned: list

6. **def main():** This function is the main method. Calls read_replace and kfold.

