1. Problem Statement

I found a data of student test which shows their mid test result and final test result then do they fail the subject or pass. With all the information provided I cannot do anything to predict does the student fail or pass the subject, because knowing mid result and final result we can easily know do they pass the subject or not. Based on the subject criteria it will be very easy to find if the student fails or pass the subject after knowing mid result and final result.

After seeing past in my middle and high school. I realize that if on that time I was able to predict my subject status (fail/pass) based on my middle result it could help some students who are unable to split their effort evenly in all subjects. Then, I realize, I could solve this problem using this AI learning technique to predict student subject status.

Therefore, I decided to modify the CSV data a bit following my past problem scenario. I decided to remove the final result while keeping the rest of the data. All the data Including the raw data from internet (“full-mark.csv”) are place in “aima-data” folder. The main disadvantage of this problem is the CSV provided is lack of data which only consist around 100 student marks of a specific subject.

2. Selection of AI technique

To know exactly which algorithm can be used for this problem I decided to test all algorithms 5 times. This will provide me average accuracy percentage which can be used to decide which problem can be used for solving this problem. These below are the accuracy results from test using same CSV:

	DTL	KNN	LR	SVM
First Test	67.86 %	85.71 %	20.38 %	71.43 %
Second Test	64.28 %	75 %	42.87 %	71.43 %
Third Test	75 %	71.43 %	15.1 %	85.71 %
Fourth Test	64.28 %	82.14 %	30 %	78.57 %
Fifth Test	75 %	75 %	35.32 %	57.14 %

Based on the result we can conclude that LR has the worst accuracy. We can say worst accuracy the average is lower than 50%. This mean the algorithm is failed to solve this problem. Imagine having AI prediction that are lower than flipping a coin. LR returning low accuracy which mean the data we are given to solve this problem cannot be divided linearly.

The rest of accuracy are similar to each other, so I decided to see which algorithm has Maximum, Mean, Median. This result will give better answer regarding choosing best algorithm for this problem.

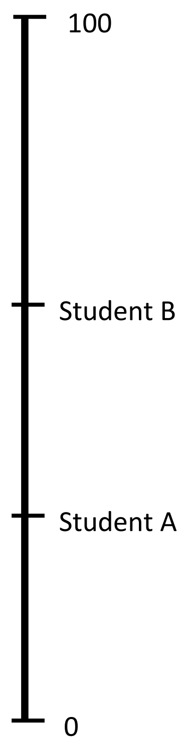
	Maximum	Mean	Median
DTL	75 %	69.284 %	67.86
KNN	85.71 %	77.856 %	75
SVM	85.71 %	72.856 %	71.43

After seeing the result, we can conclude that the best algorithm to be used on solving this problem is k-nearest neighbors (KNN). KNN win over other algorithm in all aspect. The second choices will be SVM with same amount of maximum accuracy however SVM accuracy can go to 57.14 %. On the other hand, DTL has pretty stable accuracy within range of 64.28 % to 75 %. However, LR are not recommended and suit to solve this problem with accuracy lower than 50 %.

3. Explanation of KNN technique

KNN algorithm or k-nearest neighbors is supervised machine learning algorithm which can solve classification and regression problem. The agent will classify similarity of the independent variable to predict the dependent variable. The main concept of KNN is similar things/data/variable are near to each other.

KNN advantages are simple to implement, and versatile to solve problem. However, the disadvantages of KNN are, get slower as the number of independent variable increase used in the dataset. Fortunately, this disadvantage does not matter on this problem. Because, we only have one independent variable based on the used CSV. However, if we have more independent variable, it might have significant impact to the AI.



On this problem, KNN solve this problem by classify all student score. Since this problem using single independent variable, the graph will not be two-dimension graph instead it will be one-dimension graph or similar to a single line. The visualization can be seen on Graph 1.1.

The Agent will decide where the standard is for a student to fail by learning the CSV provided. In assumption of, the subject has different difficulty regarding mid-exam and final exam. This makes the agent can decide student that scores around 60 marks for mid semester still fail the subject. Despite having lack of sample data, the agent can have accuracy more than 75%.

Graph 1.1

Sample Image of KNN works

4. Solution Result

This problem is a classification problem. KNN classify all the mid-exam score of all students and see whether they pass of fail on the subject. The KNN algorithm has around 77.8 % average accuracy which is pretty high including we only receive 100 data. If we could get more data, I believe this accuracy might went up.

This AI could be used in any school to provide student realistic feedback regarding their performance from start to the middle of semester on the specific subject. With this feedback student can know does the effort that they pour in is enough or they will need to put more effort in the mid to end semester. After using this AI, teacher of a specific subject may see the effectiveness of this AI to increase success/pass rate of a student in a specific subject.

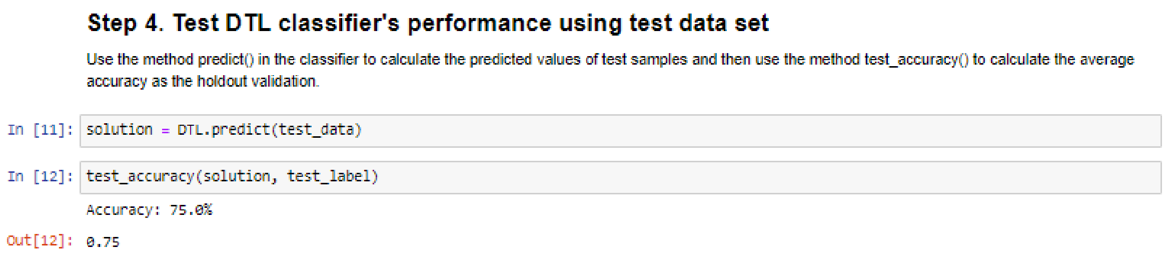
5. References

Onel Harrison 2018, *Machine Learning Basics with the K-Nearest Neighbors Algorithm*, viewed 31 August 2021, <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>.

Shubham Singh 2021, *Student Marks*, viewed 24 August 2021, <https://www.kaggle.com/shub99/student-marks>.

6. Screenshot

a. DTL



Step 4. Test DTL classifier's performance using test data set

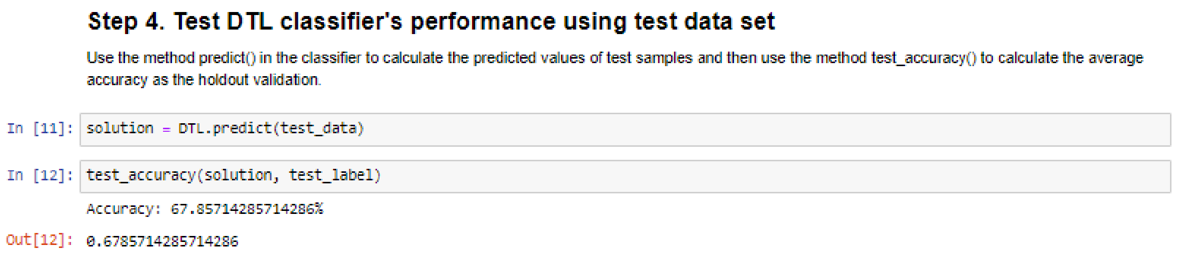
Use the method predict() in the classifier to calculate the predicted values of test samples and then use the method test_accuracy() to calculate the average accuracy as the holdout validation.

```
In [11]: solution = DTL.predict(test_data)

In [12]: test_accuracy(solution, test_label)

         Accuracy: 75.0%

Out[12]: 0.75
```



Step 4. Test DTL classifier's performance using test data set

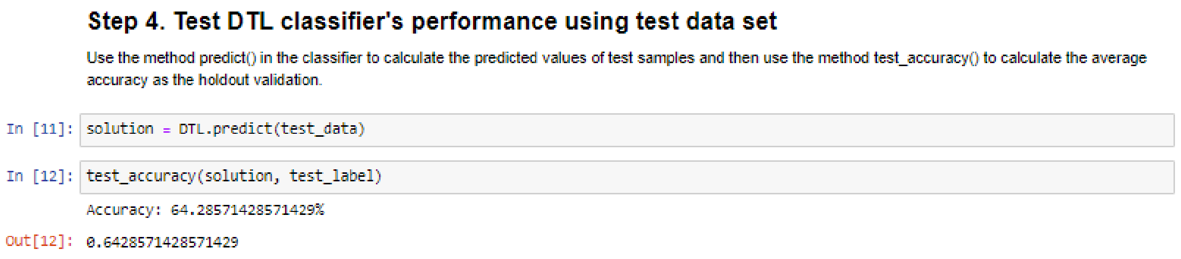
Use the method predict() in the classifier to calculate the predicted values of test samples and then use the method test_accuracy() to calculate the average accuracy as the holdout validation.

```
In [11]: solution = DTL.predict(test_data)

In [12]: test_accuracy(solution, test_label)

         Accuracy: 67.85714285714286%

Out[12]: 0.6785714285714286
```



Step 4. Test DTL classifier's performance using test data set

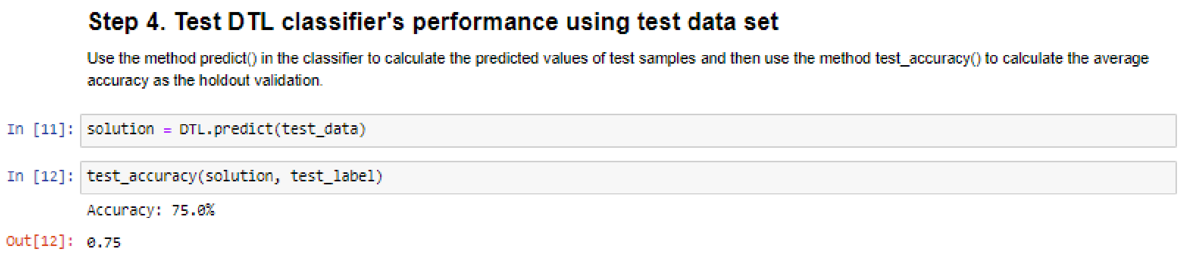
Use the method predict() in the classifier to calculate the predicted values of test samples and then use the method test_accuracy() to calculate the average accuracy as the holdout validation.

```
In [11]: solution = DTL.predict(test_data)

In [12]: test_accuracy(solution, test_label)

         Accuracy: 64.28571428571429%

Out[12]: 0.6428571428571429
```



Step 4. Test DTL classifier's performance using test data set

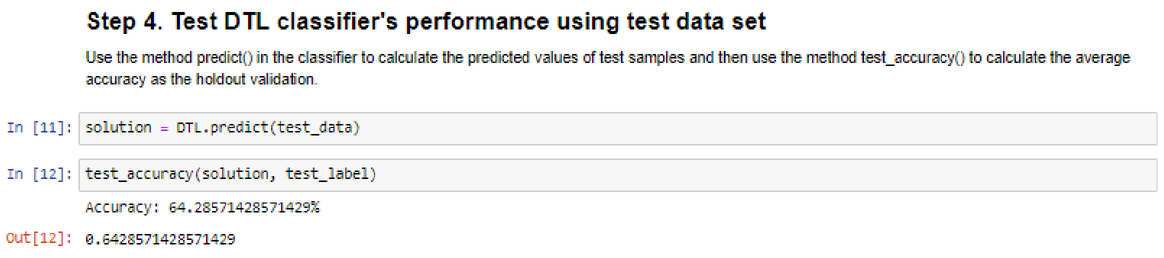
Use the method predict() in the classifier to calculate the predicted values of test samples and then use the method test_accuracy() to calculate the average accuracy as the holdout validation.

```
In [11]: solution = DTL.predict(test_data)

In [12]: test_accuracy(solution, test_label)

         Accuracy: 75.0%

Out[12]: 0.75
```



Step 4. Test DTL classifier's performance using test data set

Use the method predict() in the classifier to calculate the predicted values of test samples and then use the method test_accuracy() to calculate the average accuracy as the holdout validation.

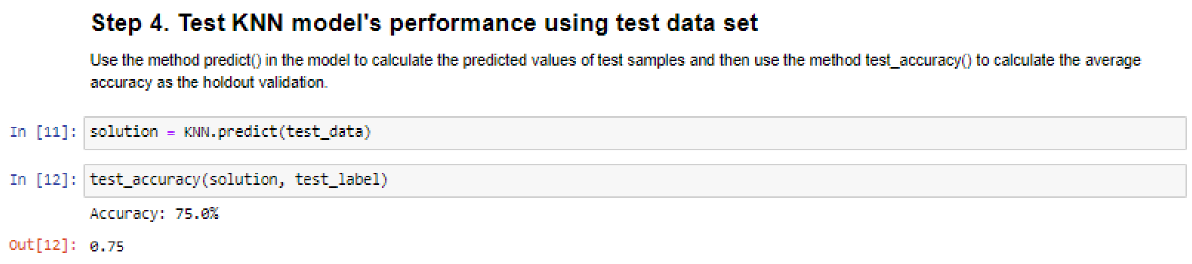
```
In [11]: solution = DTL.predict(test_data)

In [12]: test_accuracy(solution, test_label)

         Accuracy: 64.28571428571429%

Out[12]: 0.6428571428571429
```

b. KNN



Step 4. Test KNN model's performance using test data set

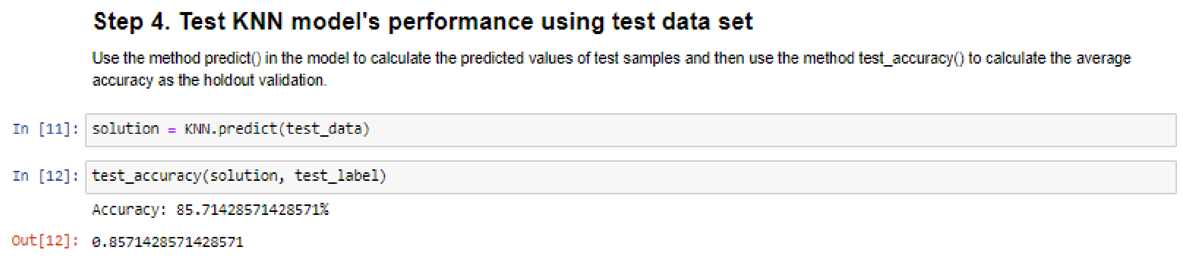
Use the method predict() in the model to calculate the predicted values of test samples and then use the method test_accuracy() to calculate the average accuracy as the holdout validation.

```
In [11]: solution = KNN.predict(test_data)

In [12]: test_accuracy(solution, test_label)

         Accuracy: 75.0%

Out[12]: 0.75
```



Step 4. Test KNN model's performance using test data set

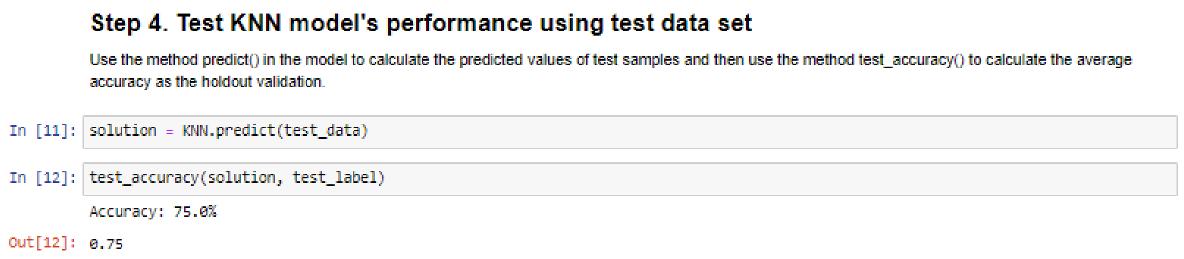
Use the method predict() in the model to calculate the predicted values of test samples and then use the method test_accuracy() to calculate the average accuracy as the holdout validation.

```
In [11]: solution = KNN.predict(test_data)

In [12]: test_accuracy(solution, test_label)

         Accuracy: 85.71428571428571%

Out[12]: 0.8571428571428571
```



Step 4. Test KNN model's performance using test data set

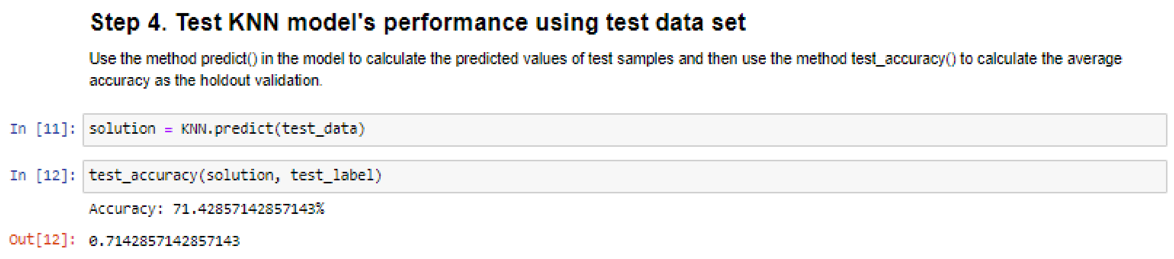
Use the method predict() in the model to calculate the predicted values of test samples and then use the method test_accuracy() to calculate the average accuracy as the holdout validation.

```
In [11]: solution = KNN.predict(test_data)

In [12]: test_accuracy(solution, test_label)

         Accuracy: 75.0%

Out[12]: 0.75
```



Step 4. Test KNN model's performance using test data set

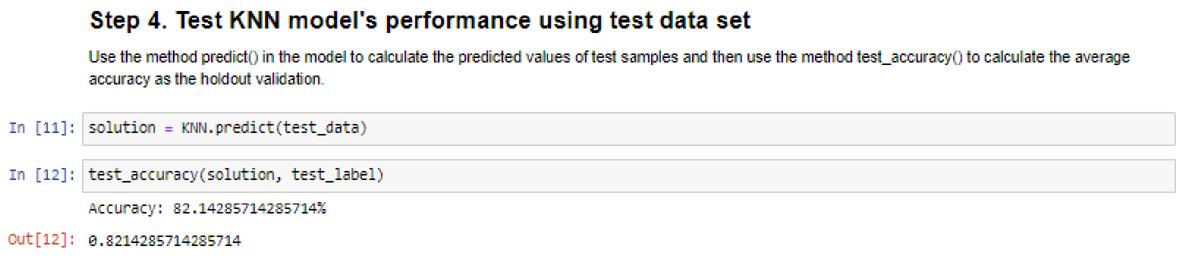
Use the method predict() in the model to calculate the predicted values of test samples and then use the method test_accuracy() to calculate the average accuracy as the holdout validation.

```
In [11]: solution = KNN.predict(test_data)

In [12]: test_accuracy(solution, test_label)

         Accuracy: 71.42857142857143%

Out[12]: 0.7142857142857143
```



Step 4. Test KNN model's performance using test data set

Use the method predict() in the model to calculate the predicted values of test samples and then use the method test_accuracy() to calculate the average accuracy as the holdout validation.

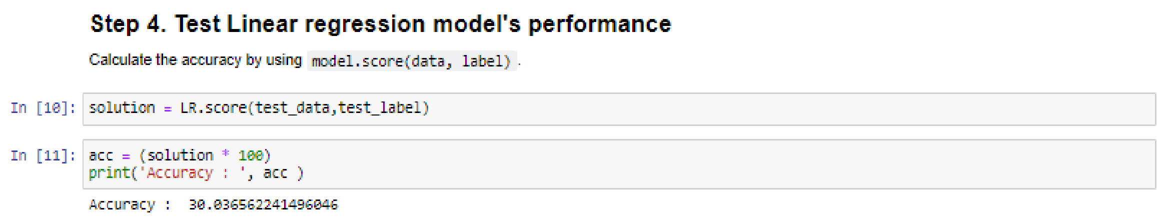
```
In [11]: solution = KNN.predict(test_data)

In [12]: test_accuracy(solution, test_label)

         Accuracy: 82.14285714285714%

Out[12]: 0.8214285714285714
```

c. LR



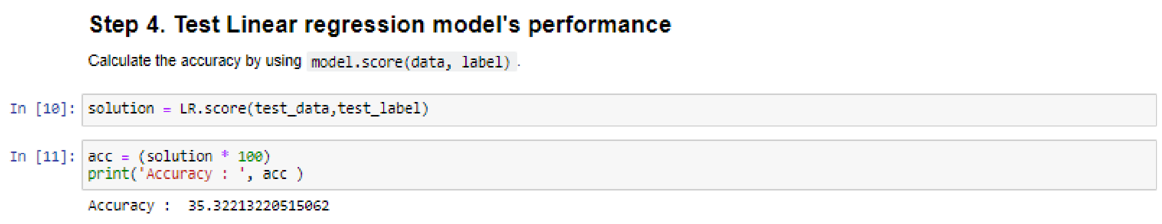
Step 4. Test Linear regression model's performance

Calculate the accuracy by using `model.score(data, label)` .

```
In [10]: solution = LR.score(test_data,test_label)
```

```
In [11]: acc = (solution * 100)
         print('Accuracy : ', acc )
```

```
Accuracy :  30.036562241496046
```



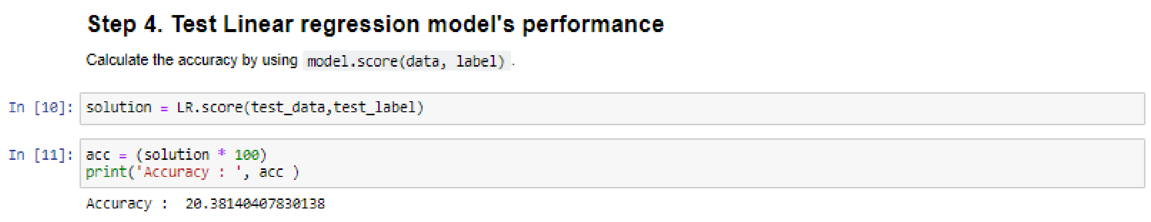
Step 4. Test Linear regression model's performance

Calculate the accuracy by using `model.score(data, label)` .

```
In [10]: solution = LR.score(test_data,test_label)
```

```
In [11]: acc = (solution * 100)
         print('Accuracy : ', acc )
```

```
Accuracy :  35.32213220515062
```



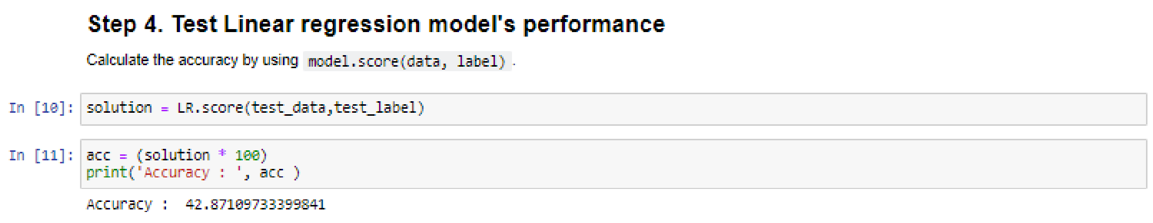
Step 4. Test Linear regression model's performance

Calculate the accuracy by using `model.score(data, label)` .

```
In [10]: solution = LR.score(test_data,test_label)
```

```
In [11]: acc = (solution * 100)
         print('Accuracy : ', acc )
```

```
Accuracy :  20.38140407830138
```



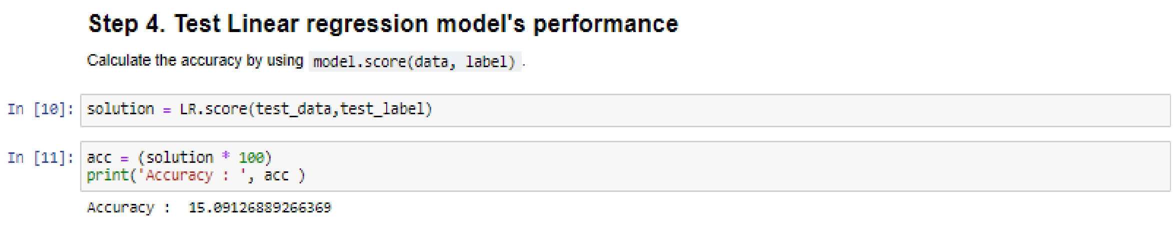
Step 4. Test Linear regression model's performance

Calculate the accuracy by using `model.score(data, label)` .

```
In [10]: solution = LR.score(test_data,test_label)
```

```
In [11]: acc = (solution * 100)
         print('Accuracy : ', acc )
```

```
Accuracy :  42.87109733399841
```



Step 4. Test Linear regression model's performance

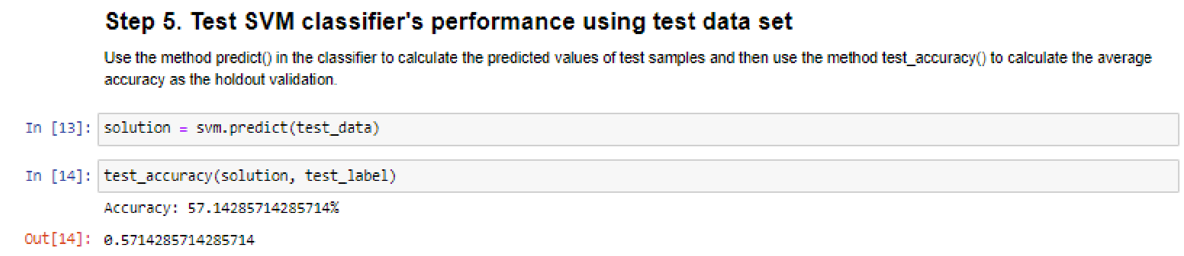
Calculate the accuracy by using `model.score(data, label)` .

```
In [10]: solution = LR.score(test_data,test_label)
```

```
In [11]: acc = (solution * 100)
         print('Accuracy : ', acc )
```

```
Accuracy :  15.09126889266369
```

d. SVM



Step 5. Test SVM classifier's performance using test data set

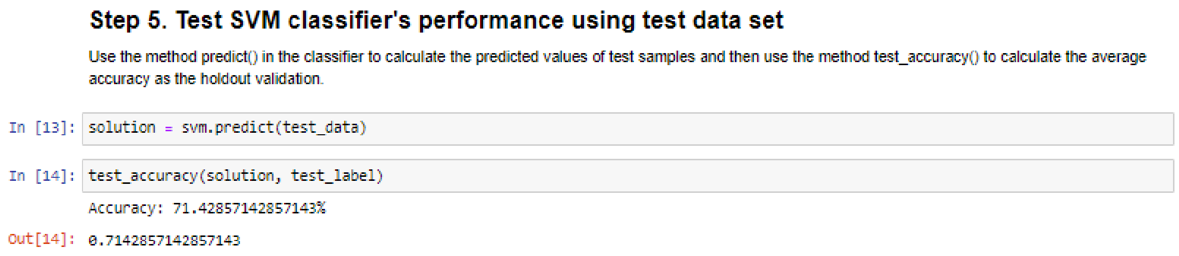
Use the method predict() in the classifier to calculate the predicted values of test samples and then use the method test_accuracy() to calculate the average accuracy as the holdout validation.

```
In [13]: solution = svm.predict(test_data)

In [14]: test_accuracy(solution, test_label)

         Accuracy: 57.14285714285714%

Out[14]: 0.5714285714285714
```



Step 5. Test SVM classifier's performance using test data set

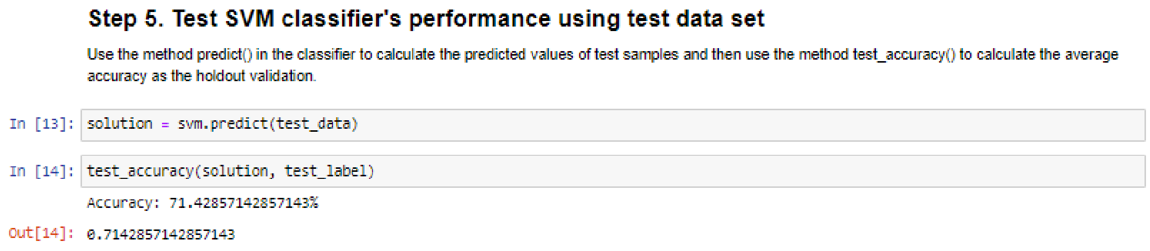
Use the method predict() in the classifier to calculate the predicted values of test samples and then use the method test_accuracy() to calculate the average accuracy as the holdout validation.

```
In [13]: solution = svm.predict(test_data)

In [14]: test_accuracy(solution, test_label)

         Accuracy: 71.42857142857143%

Out[14]: 0.7142857142857143
```



Step 5. Test SVM classifier's performance using test data set

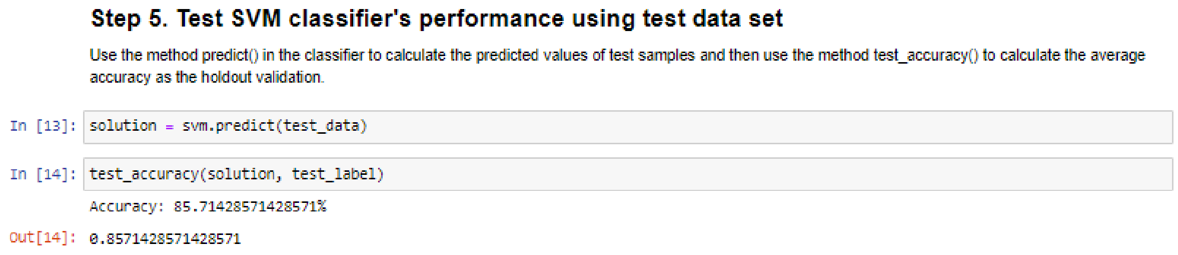
Use the method predict() in the classifier to calculate the predicted values of test samples and then use the method test_accuracy() to calculate the average accuracy as the holdout validation.

```
In [13]: solution = svm.predict(test_data)

In [14]: test_accuracy(solution, test_label)

         Accuracy: 71.42857142857143%

Out[14]: 0.7142857142857143
```



Step 5. Test SVM classifier's performance using test data set

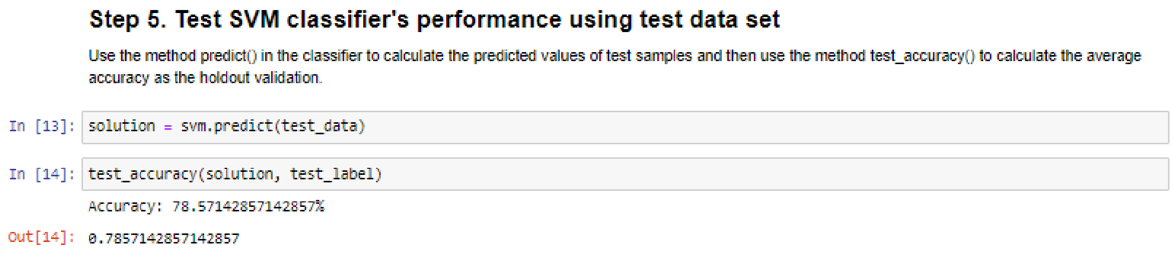
Use the method predict() in the classifier to calculate the predicted values of test samples and then use the method test_accuracy() to calculate the average accuracy as the holdout validation.

```
In [13]: solution = svm.predict(test_data)

In [14]: test_accuracy(solution, test_label)

         Accuracy: 85.71428571428571%

Out[14]: 0.8571428571428571
```



Step 5. Test SVM classifier's performance using test data set

Use the method predict() in the classifier to calculate the predicted values of test samples and then use the method test_accuracy() to calculate the average accuracy as the holdout validation.

```
In [13]: solution = svm.predict(test_data)

In [14]: test_accuracy(solution, test_label)

         Accuracy: 78.57142857142857%

Out[14]: 0.7857142857142857
```