COSC 6337 Data Mining : Individual Project

Classification Of Iris Species Using Data Mining Algorithms

Instructor: Dr.Mamta Yadav

Yachamaneni Yashaswini – A04257757

Objective: The main goal of this project is to implement and evaluate classification/clustering mechanisms by using machine learning.

Contents:

Abstract:

Iris is famous species which is a flower. The thought that very flower differs from one another I thought of applying classification algorithm to this species by considering its petal and sepal dimensions. Also, I took three different species in Iris and compared the petal and sepal features.

Material and Methodologies Used:

I took references from kaggel.com and Scikit Learn.org. Also, I used jupyter Nootebook to run the code.

Formula to calculate Accuracy is,

Accuracy = no. of correct prediction / total number of predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

How I measured the classification/clusters quality?

We know data in the real-world is dirty. It may be inconsistent, noisy, or incomplete. Also, low-quality data leads to low-quality mining results. Measuring data quality is a multidimensional vision with dimensions such as accuracy, consistency, completeness, timeliness, validity, and interpretability.

Steps to be followed for measuring the data quality.

1. Loaded the dataset:
    - iris = pd.read_csv("../input/Iris.csv")
2. Checked for any inconsistency:
    - iris.info()
    - the dataset contains no null values, the data can be processed.

3. Removed the unneeded columns:
    - iris.drop('Id',axis=1,inplace=True)
    - dropping the Id column as it is unnecessary, axis=1 will specify that it is column wise, inplace =1 means the changes that will be reflected into the data-frame.

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

Fig 1.

Experiments:

SUPERVISED LEARNING:

It is a part of Machine learning and artificial Intelligence. It is used to label the training dataset and predict the outcomes accurately. For example, spam folder in your Gmail account.

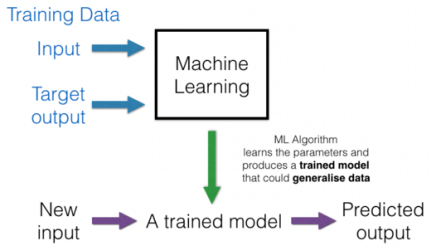


Fig 2.

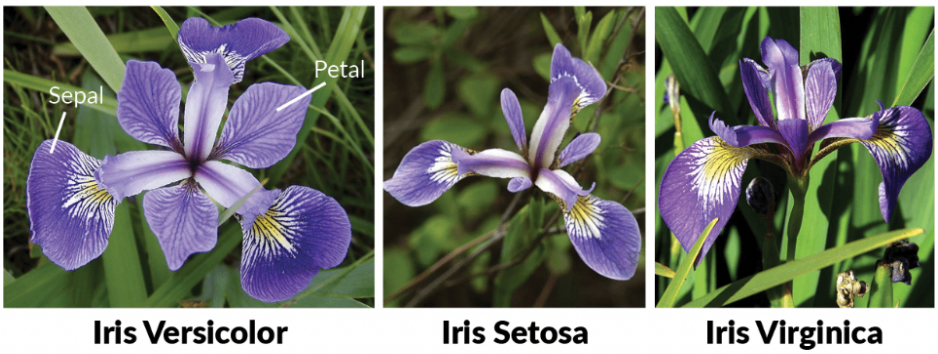
CLASSIFICATION:

- It is a supervised learning. In Classification, the samples belong to two or more classes, and we will learn from already labeled data how to predict the class of unlabeled data. It is a data analysis task.
- Classification is a method where the feature which need to be predicted contains the categories of values. Each category is considered as class in which the predicted value will fall.
- It is a two-step process:

A) Learning step
B) Classification step

Attributes: An instance's attribute is a characteristic that can be used to categorize it. The length and width of the petal and sepal are the attributes in the following dataset. It is also known as Features.

Target Variables: The attribute species is my target variable. For example, three different flower species.



Classifying three types of Iris.

Fig 3.

Steps to be followed:

- Import all necessary packages to use in various classification algorithm.
- Get the shape of dataset. (Features and their correlation)
- Draw the heatmap with input as correlation matrix calculated.

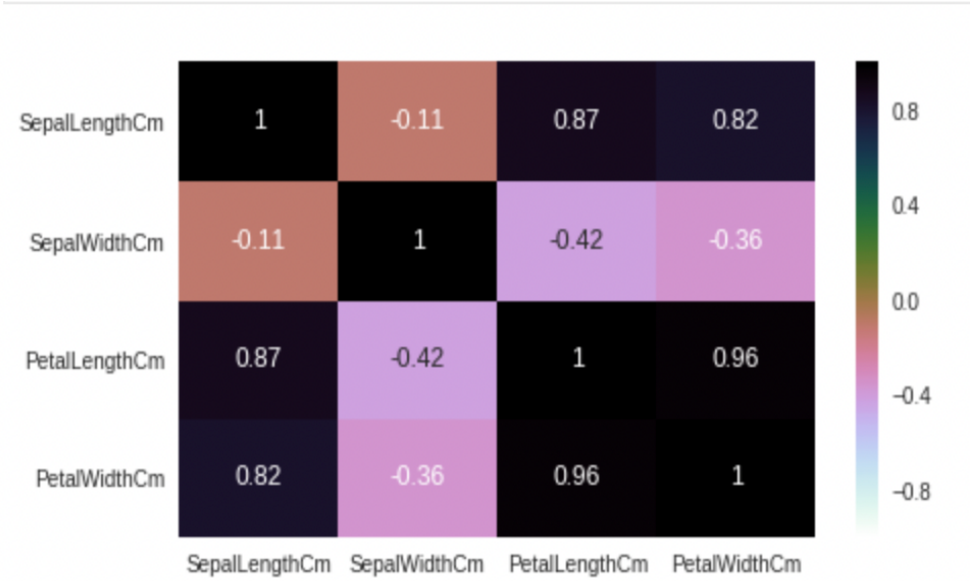


Fig 4.

Steps To Be followed When Applying an Algorithm:

1. Create training and testing datasets from the dataset. The testing dataset is typically smaller than the training dataset since it will aid in better model training
2. Choose whatever algorithm you think would be effective based on the problem (classification or regression).
3. After that, give the algorithm the training dataset so it can be trained. The .fit() technique is employed.
4. The trained algorithm will then be given the test data to predict the result. The .predict() technique is employed.
5. After that, we run the model with both the actual output and the anticipated result to check its correctness.

How I prepared the data?

1. First I divided the data into two halves which is training data and testing data.
2. Training data = 70% and testing data = 30%
3. Load the features of training and testing data.

```
In [15]: train_X.head(2)
```

Out[15]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
75	6.6	3.0	4.4	1.4
16	5.4	3.9	1.3	0.4

Fig 5.

```
In [16]: test_X.head(2)
```

Out[16]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
91	6.1	3.0	4.6	1.4
134	6.1	2.6	5.6	1.4

Fig 6.

Parameters chosen for the algorithm.

In my dataset I have 150 entities and five attributes. I have considered three different species and for each species I took different petal and sepal measurements (i.e. petal-length, petal-width, sepal-length, sepal-width).
There we no null values after removing the unneeded columns, hence the data is now ready to get processed.

```
In [12]: iris.shape
Out[12]: (150, 5)
```

```
In [4]: iris.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Id              150 non-null    int64
 1   SepalLengthCm   150 non-null    float64
 2   SepalWidthCm    150 non-null    float64
 3   PetalLengthCm   150 non-null    float64
 4   PetalWidthCm    150 non-null    float64
 5   Species         150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

Fig 7.

Algorithm 1: K-Nearest Neighbors:

It is a supervised learning. The K-NN algorithm will assume that both new case and existing case are comparable, and it will place new instance in the place of category that is more likely to be the existing category. It is a non-parametric algorithm data doesn't make any assumptions on data.

Applications using KNN algorithms:

- Text Mining
- Agriculture
- Finance
- Face Recognition

Pros and Cons:

- Non-parametric, No assumption on data
- High Memory requirement.

Algorithm 2: Support Vector Machine:

It is a supervised learning algorithm which is mainly used for classification as well as regression problems. The goal of this SVM algorithm is to create best line or a decision boundary that will segregate n-dimensional space into the classes, so that we can easily put the new data point in place of correct category in the future.

.

Applications using Support Vector Machine algorithm:

- Face detection
- Image classification
- Text categorization

Pros and cons:

- High dimensional space and more efficient.
- Long overtraining and does not work well if there are overlaps.

Algorithm 3: Decision Tree:

It is a strong and well-liked categorization and prediction tool. It is a supervised learning, and which is mainly used for both classification and regression problems.

A decision tree is a tree like structure that will resemble a flowchart, in which each internal node will represent a test on an attribute, where each branch is a test result, and each leaf node (terminal node) is a class label.

Applications using decision tree:

- Medical diagnosis
- Customer Retention

Pros and Cons:

- Simple and fast
- Overfitting and chances of information loss.

Experimental result summary:

The accuracy level of K-Nearest Neighbor algorithm: 0.955
The accuracy level of SVM algorithm: 0.955
The accuracy level of Decision Tree: 0.933

We have observed that, in K-Nearest Neighbor algorithm the accuracy level is same for three different values of k (neighbors). The SVM and KNN algorithm is giving 0.02% better accuracy level compared to Decision Tree.

Output:

K-Nearest Neighbors:

For k =3,

```
In [22]: model=KNeighborsClassifier(n_neighbors=3) #this examines 3 neighbours for putting the new data into a class
         model.fit(train_X,train_y)
         prediction=model.predict(test_X)
         print('The accuracy of the KNN is',metrics.accuracy_score(prediction,test_y))

         The accuracy of the KNN is 0.9555555555555556
```

Fig 8

For k=4,

```
model=KNeighborsClassifier(n_neighbors=4) #this examines 4 neighbours for putting the new data into a class
model.fit(train_X,train_y)
prediction=model.predict(test_X)
print('The accuracy of the KNN is',metrics.accuracy_score(prediction,test_y))
```

```
The accuracy of the KNN is 0.9555555555555556
```

Fig 9.

For k=5,

```
model=KNeighborsClassifier(n_neighbors=5) #this examines 5 neighbours for putting the new data into a class
model.fit(train_X,train_y)
prediction=model.predict(test_X)
print('The accuracy of the KNN is',metrics.accuracy_score(prediction,test_y))
```

```
The accuracy of the KNN is 0.9555555555555556
```

Fig 10.

Support Vector Machine:

```
In [21]: model = svm.SVC() #select the algorithm
         model.fit(train_X,train_y) # we train the algorithm with the training data and the training output
         prediction=model.predict(test_X) #now we pass the testing data to the trained algorithm
         print('The accuracy of the SVM is:',metrics.accuracy_score(prediction,test_y))#now we check the accuracy of the algorit
         #we pass the predicted output by the model and the actual output

         The accuracy of the SVM is: 0.9555555555555556
```

Fig 11.

Decision Tree:

```
In [24]: model=DecisionTreeClassifier()
         model.fit(train_X,train_y)
         prediction=model.predict(test_X)
         print('The accuracy of the Decision Tree is',metrics.accuracy_score(prediction,test_y))

         The accuracy of the Decision Tree is 0.9333333333333333
```

Fig 12.

References:

1. https://www.kaggle.com/code/ash316/ml-from-scratch-with-iris
2. https://www.kaggle.com/code/anniepyim/essential-classification-algorithms-explained/notebook