NGF COLLEGE OF ENGINEERING AND TECHNOLOGY, PALWAL (HARYANA)

IRIS FLOWER CLASSIFICATION IN DATA SCIENCE

PROJECT REPORT

Submitted By
Ishwa
B. tech (CSE 4th Year)

7th Sem

S19CSE022

Submitted To
Ms. Bhawna
Assistant Professor

LIST OF CONTENT

Introduction

The Iris Classification in data science is end to end machine learning project or data science project for iris flower classification. The aim of the iris flower classification is to predict flowers based on their specific features and separate different species and to identify them. It contains five columns namely – Petal Length, Petal Width, Sepal Length, Sepal Width and Species Type. All these lengths were in centimetres. And dependent feature, which will be the output for the model, is species. It contains the name of the species to which that particular flower with those measurements belongs. Iris is a flowering plant, the researchers have measured various features of the different iris flowers and recorded them digitally.

Machine learning?

Machine learning is about learning to predict something or extracting knowledge from data. ML is a part of artificial intelligence. ML algorithms build a model based on sample data or known as training data and based upon the training data the algorithm can predict something on new data.

Categories of Machine Learning:

- **Supervised machine learning:** Supervised machine learning are types of machine learning that are trained on well-labelled training data. Labelled data means the training data is already tagged with the correct output.
- **Unsupervised machine learning:** Unlike supervised learning, unsupervised learning doesn’t have any tagged data. It learned patterns from untagged data. Basically, it creates a group of objects based on the input data/features.
- **Semi-supervised machine learning:** Semi-supervised learning falls between supervised and unsupervised learning. It has a small amount of tagged data and a large amount of untagged data

Classification:

Is one of the major data mining processes which maps data into predefined groups. It comes under supervised learning method as the classes are determined before examining the data. For applying all approaches to performing classification it is required to have some knowledge of the data. Usually, the knowledge of the data helps to find some unknown patterns. The aim of pattern classification is to building a function that provides output of two or more than two classes from the input feature.

The Iris flower data set introduced by the British statistician and biologist Ronald Fisher that’s why it is also known by Fisher's Iris data set and it is a multivariate data set. The use of multiple measurements in taxonomic problems as an example of linear discriminant analysis. The expectation from mining iris data set would be discovering patterns from examining sepal and petal size of the iris plant and how the prediction was made from analyzing the pattern to predict the class of iris plant. In upcoming years, using the classification and pattern recognition other flowers can be individually distinguish to each other. It is unmistakably expressed that the sort of relationship that being mined utilizing iris dataset would be a classification model.

Objective

The objective of iris classification in data science is to classify the flowers according to their traits. The central goal is to design a model that makes proper classification for new flowers. The iris data set contains fifty instances of each of three species. Iris Flower Classification is a Machine Learning Project. The iris dataset contains three classes of flowers, Versicolor, Setosa, Virginica, and each class contains 4 features, ‘Sepal Length’, ‘Sepal width’, ‘Petal length’, ‘Petal width’. The aim of the iris flower classification is to predict flowers based on their specific features and separate different species and to identify them.

.

System Requirement

Hardware Requirement:

Processor Brand: Intel

Processor Type: Core i5

Processor Speed: 2.40GHz

Ram Size: 8.00 GB

Software Requirement

Operating system: Windows 10

Application Server: Anaconda

Language used: Python

System Analysis and design

Iris Data:

The dataset for this project originates from the UCI Machine Learning Repository. The Iris flower data set or Fisher's Iris data set is a multivariate data set. The data set consists of 50 samples from each of three species of Iris (Iris virginica, Iris versicolor and Iris setosa).

Four features were measured from each sample (in centimetres):

- Length of the petals
- Width of the petals
- Length of the sepals
- Width of the sepals

Load the data:

- Numpy will be used for any computational operations.
- We'll use Matplotlib and seaborn for data visualization.
- Pandas help to load data from various sources like local storage, database, excel file, CSV file, etc.
- Next, we load the data using pd.read_csv() and set the column name as per the iris data information.
- Pd.read_csv reads CSV files. CSV stands for comma separated value.
- df.head() only shows the first 5 rows from the data set table.

Analyse and visualize the dataset:

- We can see all the descriptions about the data, like average length and width, minimum value, maximum value, etc.
- To visualize the whole dataset we used the seaborn pair plot method. It plots the whole dataset's information.

Model training:

- We split the whole data into training and testing datasets. Later we'll use the testing dataset to check the accuracy of the model.

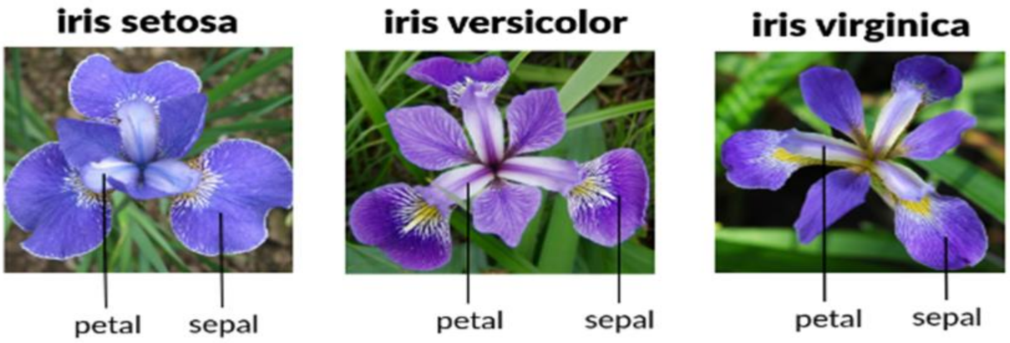
Model Evaluation:

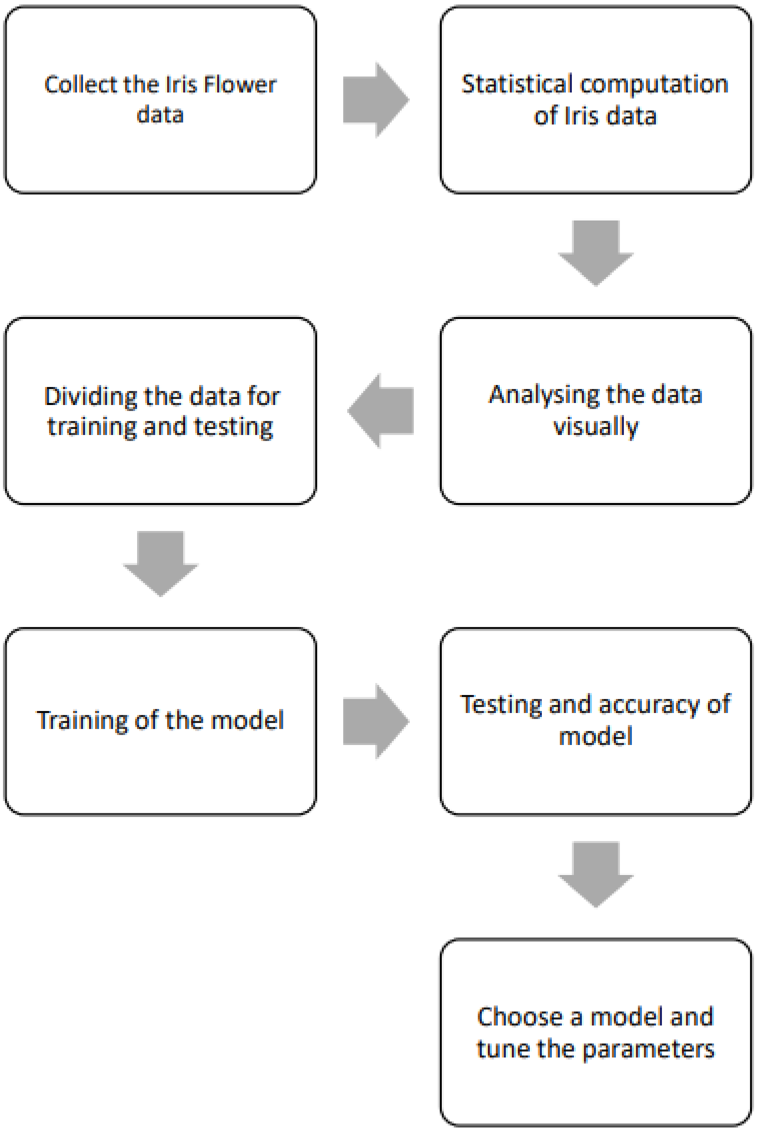
- Now we predict the classes from the test dataset using our trained model.
- Then we check the accuracy score of the predicted classes.

Testing the model:

- Here we take some random values based on the average plot to see if the model can predict accurately.

Data Flow Diagram



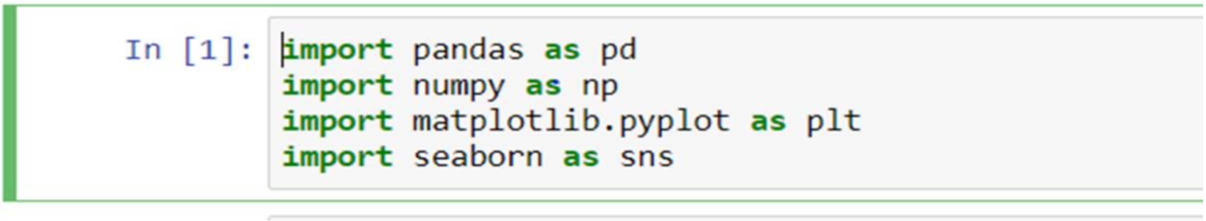

Collect the Iris Flower data
Statistical computation of Iris data
Analysing the data visually
Dividing the data for training and testing
Training of the model
Testing and accuracy of model
Choose a model and tune the parameters

Above block diagram is showing the step-by-step process to predict iris flower species.

Output

Step 1 - IMPORTING LIBRARIES AND DOWNLOAD THE DATA

The following libraries are required for this project:



```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
```

Fig. Imported Libraries

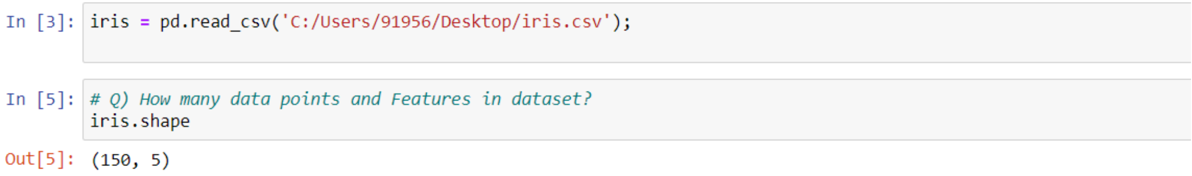
Imported Libraries

First, we've imported some necessary packages for the project

- Numpy will be used for any computational operations.
- We'll use Matplotlib and seaborn for data visualization.
- Pandas help to load data from various sources like local storage, database, excel file, CSV file, etc.

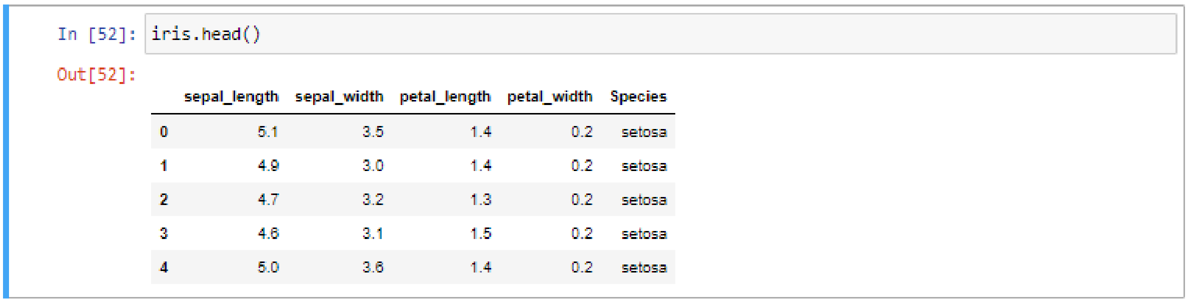
DATA EXPLORATION:

Now we are going to move into data exploration as well as analysis using the iris data. Let's import our data set using 'pandas' library, which will convert our data into the tabular format from the CSV format. The beauty of using pandas library is just that we can read the csv files.



```
In [3]: iris = pd.read_csv('C:/Users/91956/Desktop/iris.csv');

In [5]: # Q) How many data points and Features in dataset?
        iris.shape

Out[5]: (150, 5)
```

- Next, we load the data using pd.read_csv() and set the column name as per the iris data information.
- Pd.read_csv reads CSV files. CSV stands for comma separated value.
- iris.shape shows how many data points and Features in dataset



```
In [52]: iris.head()

Out[52]:
```

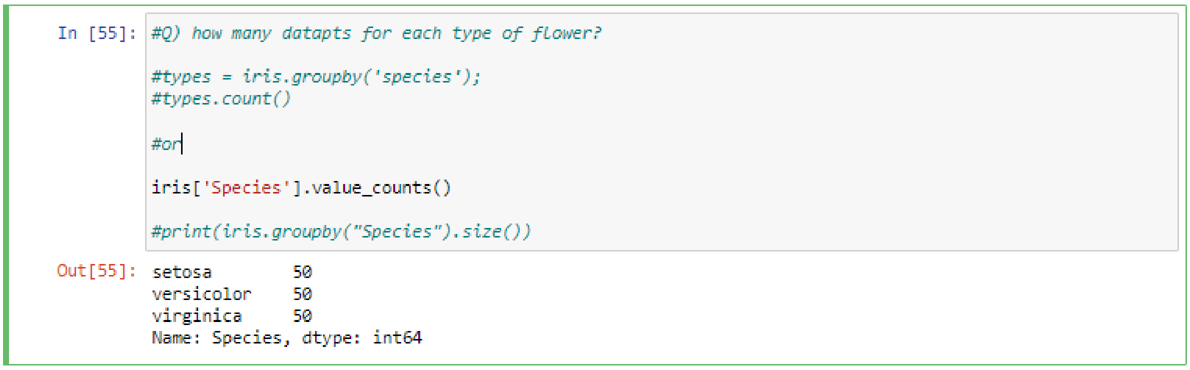
	sepal_length	sepal_width	petal_length	petal_width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Table: Showing Iris Dataset using pandas Library

Step 2 – Analyze and visualize the dataset:

Data analyses:

This dataset contains 150 samples Since the dataframe has four features (Petal length, petal width, sepal length and sepal width) with 150 samples belonging to either of the three target classes, and each class has distributed equally.

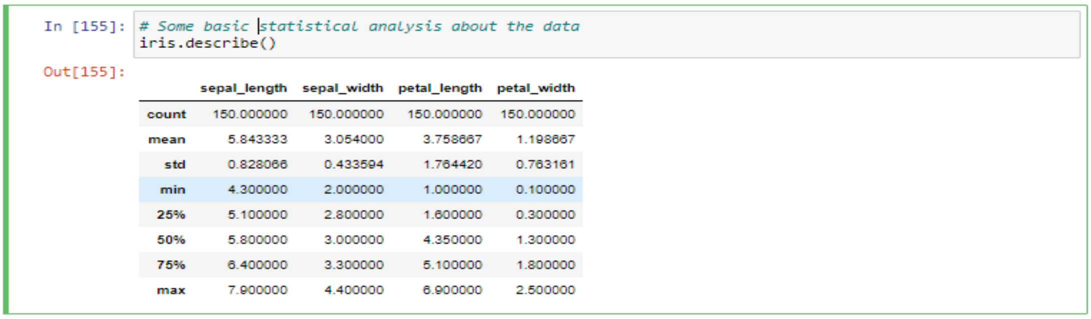


```
In [55]: #Q) how many datapts for each type of flower?

         #types = iris.groupby('species');
         #types.count()

         #or

         iris['Species'].value_counts()

         #print(iris.groupby("Species").size())

Out[55]: setosa        50
         versicolor    50
         virginica     50
         Name: Species, dtype: int64
```

Table: Species in Iris Dataset

Let's see some information about the dataset.

By using 'df.describe()' we can see the mathematics of the dataset, which helps to find out the standard deviation, mean, minimum value and the four quartile percentile of the data.



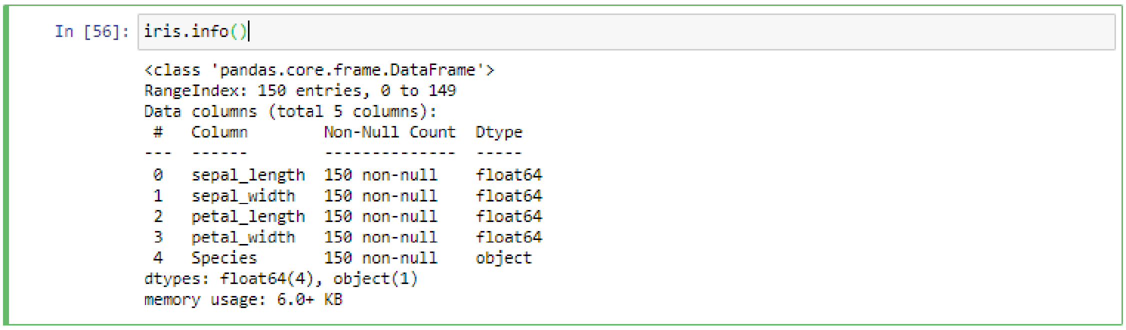
```
In [155]: # Some basic statistical analysis about the data
          iris.describe()

Out[155]:
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Table: Statistical description of iris dataset

- From this description, we can see all the descriptions about the data, like average length and width, minimum value, maximum value, the 25%, 50%, and 75% distribution value, etc.

- We can analyze some more information about our dataset, that it contains four non-null columns and one object-based column. We can also see memory usage by the iris dataset.



```
In [56]: iris.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 150 entries, 0 to 149
         Data columns (total 5 columns):
          #   Column        Non-Null Count  Dtype
         ---  ------        --------------  -----
          0   sepal_length  150 non-null    float64
          1   sepal_width   150 non-null    float64
          2   petal_length  150 non-null    float64
          3   petal_width   150 non-null    float64
          4   Species       150 non-null    object
         dtypes: float64(4), object(1)
         memory usage: 6.0+ KB
```

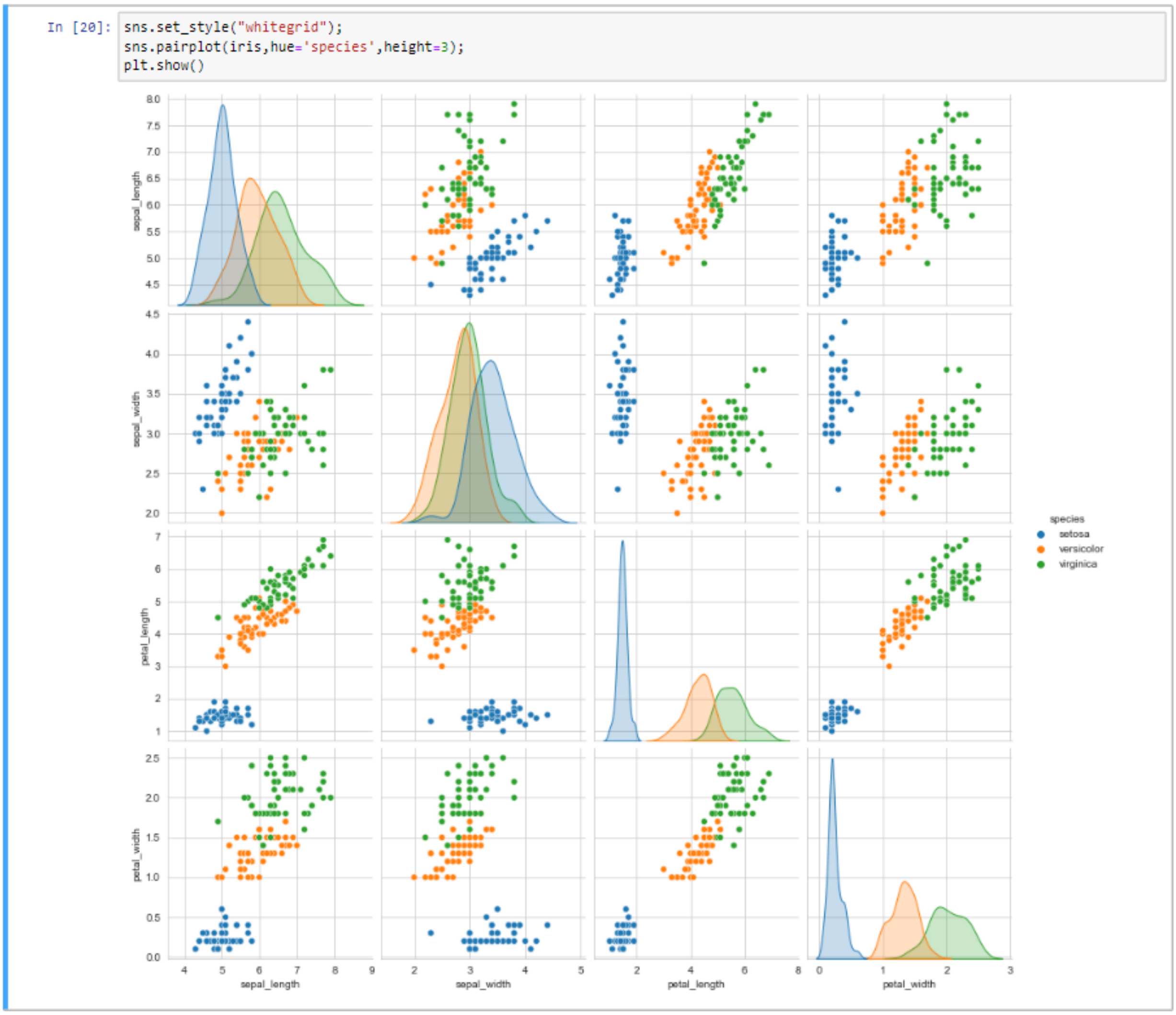
Fig. Information of iris dataset

DATA VISUALIZATION:

In the previous section what we gone through is the exploration of all the data where we did some preliminary analysis of the data and get a few of it, but to progress further and to dive into the data a little bit more we are going to do some visualization. Visualization is a great way to develop a better understanding of your data and python and has a lot of great tools for specifically that purpose.

i. Pair-plot:

To visualize the whole dataset we used the seaborn pair plot method. It plots the whole dataset's information.



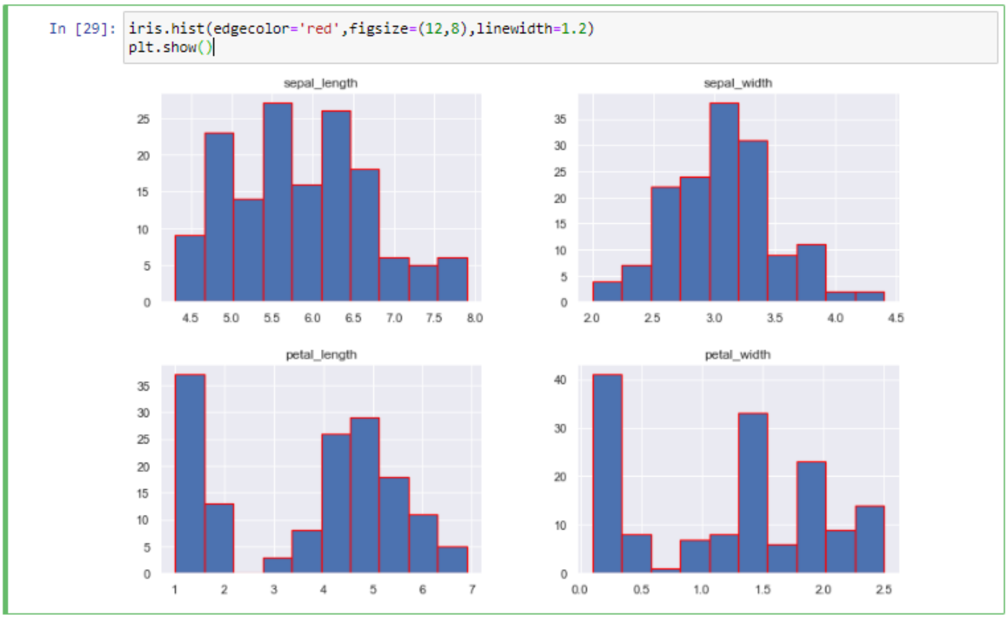
```
In [20]: sns.set_style("whitegrid");
         sns.pairplot(iris,hue='species',height=3);
         plt.show()
```

Fig. Pair-plot

- To visualize the whole dataset we used the seaborn pair plot method. It plots the whole dataset's information.
- From this visualization, we can tell that iris-setosa is well separated from the other two flowers.
- And iris virginica is the longest flower and iris setosa is the shortest.

ii. Histogram:

Historical representation is basically the pure distribution off all three combined species and from this it's not really all that informative because it just tells us overall distribution.

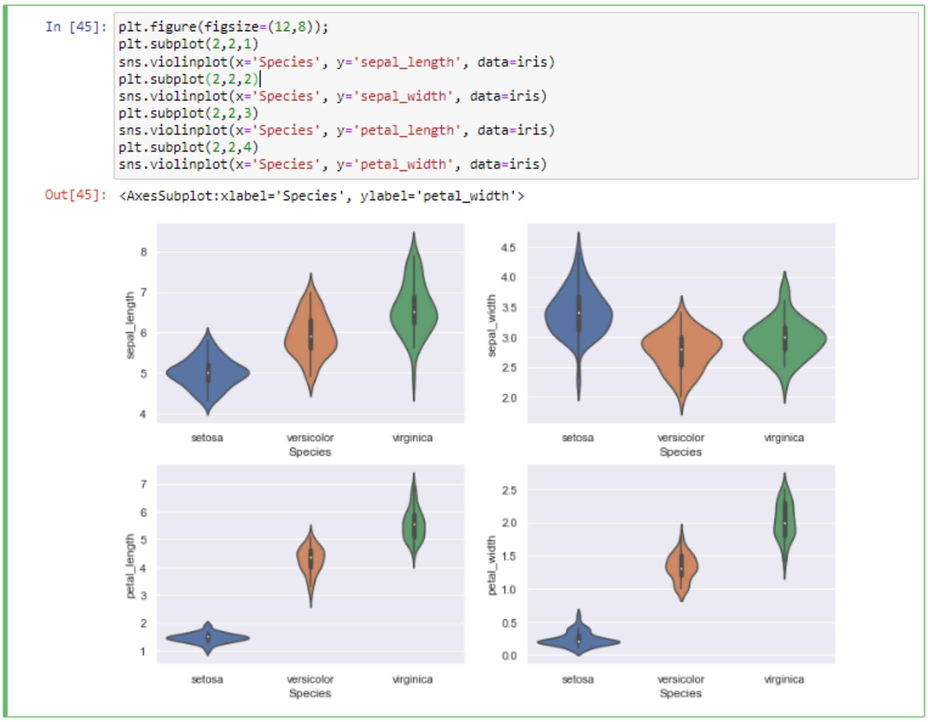


```
In [29]: iris.hist(edgecolor='red',figsize=(12,8),linewidth=1.2)
         plt.show()
```

Fig. Histogram

iii. Violin-plot:

Let us look at the violin-plot which is as similar to the box-plot, Violin plot shows us the visual representation of how our data is scattered over the plane. Here we can conclude from the below picture that in sepal length we can see this the distribution in setosa is much smaller than the versicolor and verginica. In sepal width we can examine that the distribution of setosa is 13 widest and also the longest sepal width and longest petal length in comparisons to the other attributes.

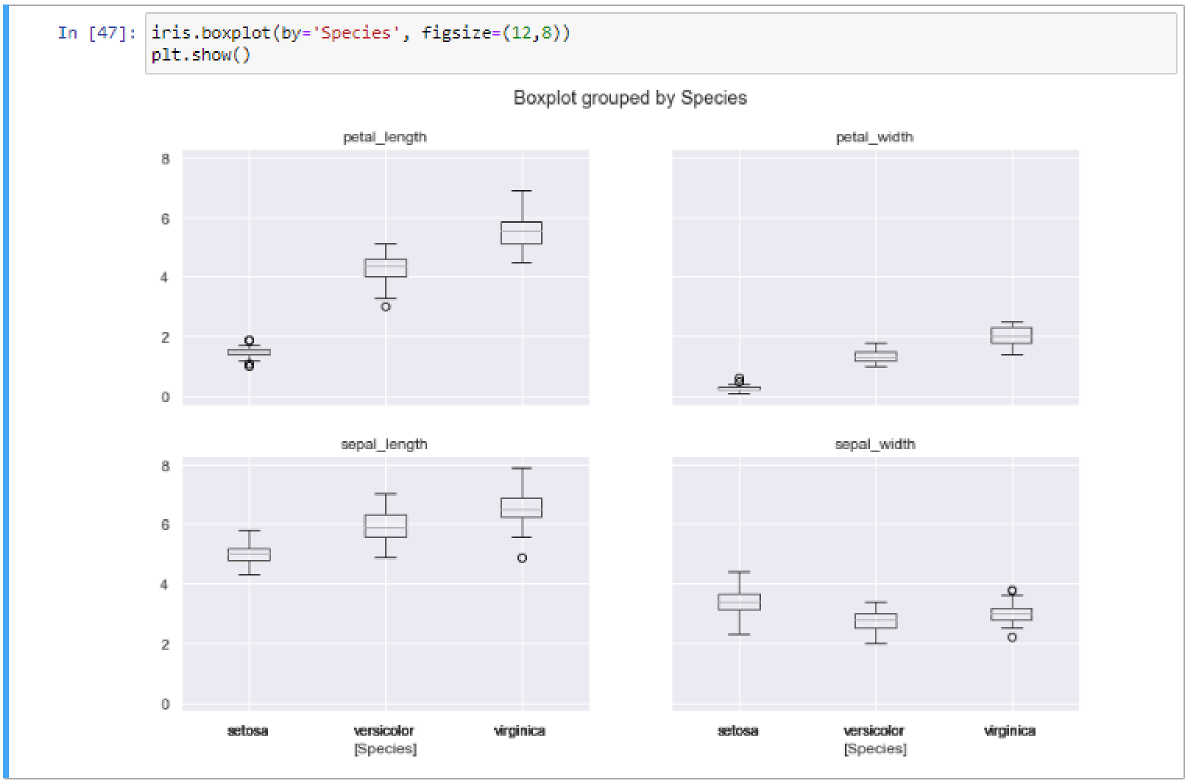


```
In [45]: plt.figure(figsize=(12,8));
         plt.subplot(2,2,1)
         sns.violinplot(x='Species', y='sepal_length', data=iris)
         plt.subplot(2,2,2)
         sns.violinplot(x='Species', y='sepal_width', data=iris)
         plt.subplot(2,2,3)
         sns.violinplot(x='Species', y='petal_length', data=iris)
         plt.subplot(2,2,4)
         sns.violinplot(x='Species', y='petal_width', data=iris)

Out[45]: <AxesSubplot:xlabel='Species', ylabel='petal_width'>
```

Fig. Violin-plot

iv. Box-plot:

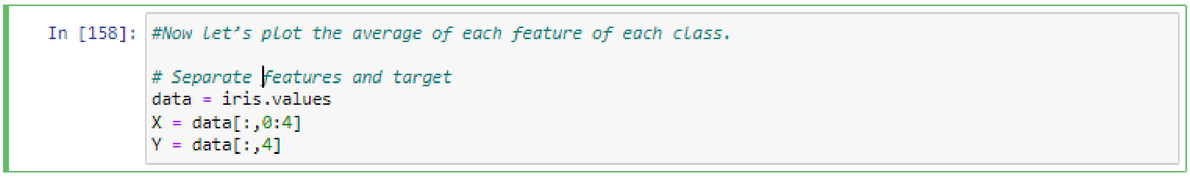
Box plot is a graph which is based on percentile, which divides the data into four quartiles of 25% each. This method is numerously used in statistical analysis to understand various measures such as max, min, mean, median and deviation.



```
In [47]: iris.boxplot(by='Species', figsize=(12,8))
         plt.show()
```

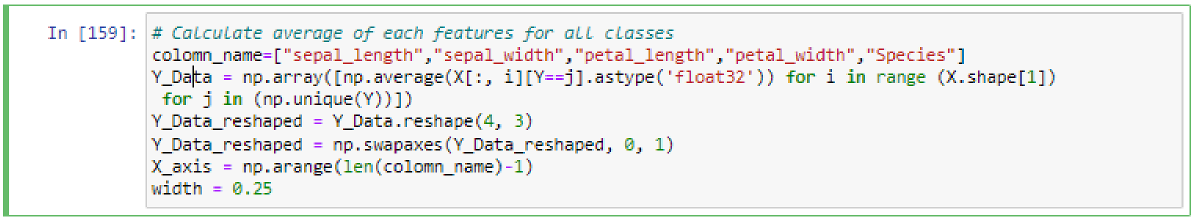
Fig. Box-plot

Now let's plot the average of each feature of each class.



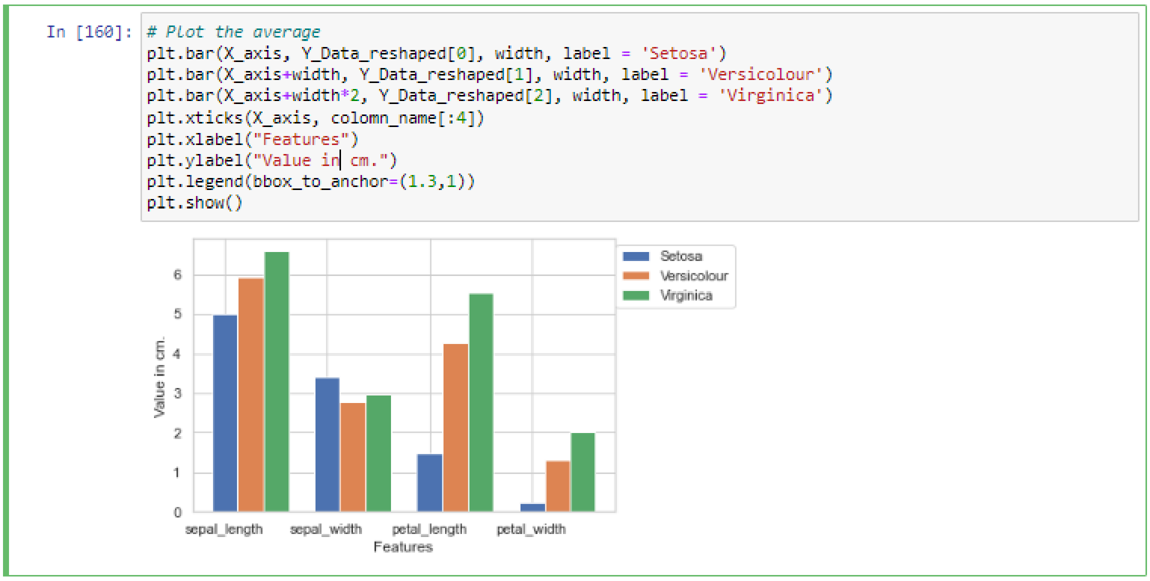
```
In [158]: #Now let's plot the average of each feature of each class.

          # Separate features and target
          data = iris.values
          X = data[:,0:4]
          Y = data[:,4]
```

- Here we separated the features from the target value.



```
In [159]: # Calculate average of each features for all classes
          colomn_name=["sepal_length","sepal_width","petal_length","petal_width","Species"]
          Y_Data = np.array([np.average(X[:, i][Y==j].astype('float32')) for i in range (X.shape[1])
           for j in (np.unique(Y))])
          Y_Data_reshaped = Y_Data.reshape(4, 3)
          Y_Data_reshaped = np.swapaxes(Y_Data_reshaped, 0, 1)
          X_axis = np.arange(len(colomn_name)-1)
          width = 0.25
```

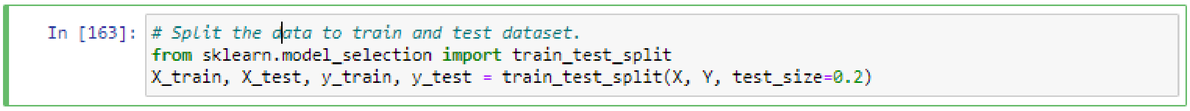
- Np.average calculates the average from an array.
- Here we used two for loops inside a list. This is known as list comprehension.
- List comprehension helps to reduce the number of lines of code.
- The Y_Data is a 1D array, but we have 4 features for every 3 classes. So we reshaped Y_Data to a (4, 3) shaped array.
- Then we change the axis of the reshaped matrix.



```
In [160]: # Plot the average
          plt.bar(X_axis, Y_Data_reshaped[0], width, label = 'Setosa')
          plt.bar(X_axis+width, Y_Data_reshaped[1], width, label = 'Versicolour')
          plt.bar(X_axis+width*2, Y_Data_reshaped[2], width, label = 'Virginica')
          plt.xticks(X_axis, colomn_name[:4])
          plt.xlabel("Features")
          plt.ylabel("Value in cm.")
          plt.legend(bbox_to_anchor=(1.3,1))
          plt.show()
```

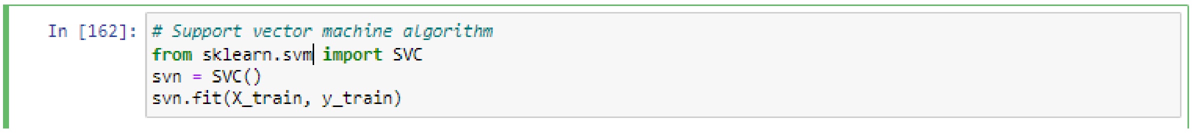
- We used matplotlib to show the averages in a bar plot.
- Here we can clearly see the verginica is the longest and setosa is the shortest flower.

Step 3 – Model training:



```
In [163]: # Split the data to train and test dataset.
          from sklearn.model_selection import train_test_split
          X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
```

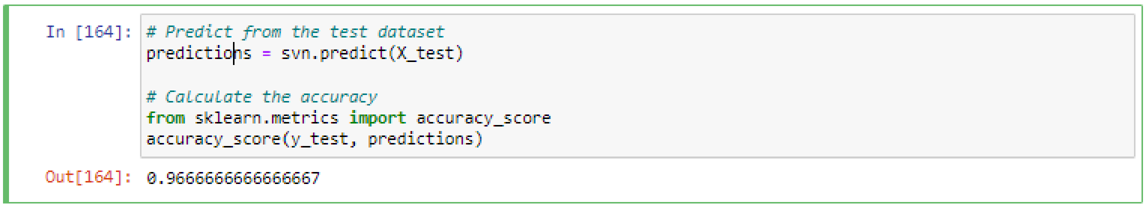
- Using train_test_split we split the whole data into training and testing datasets. Later we'll use the testing dataset to check the accuracy of the model.



```
In [162]: # Support vector machine algorithm
          from sklearn.svm import SVC
          svn = SVC()
          svn.fit(X_train, y_train)
```

- Here we imported a support vector classifier from the scikit-learn support vector machine.
- Then, we created an object and named it svn.
- After that, we feed the training dataset into the algorithm by using the svn.fit() method.

Step 4 – Model Evaluation:



```
In [164]: # Predict from the test dataset
          predictions = svn.predict(X_test)

          # Calculate the accuracy
          from sklearn.metrics import accuracy_score
          accuracy_score(y_test, predictions)

Out[164]: 0.9666666666666667
```

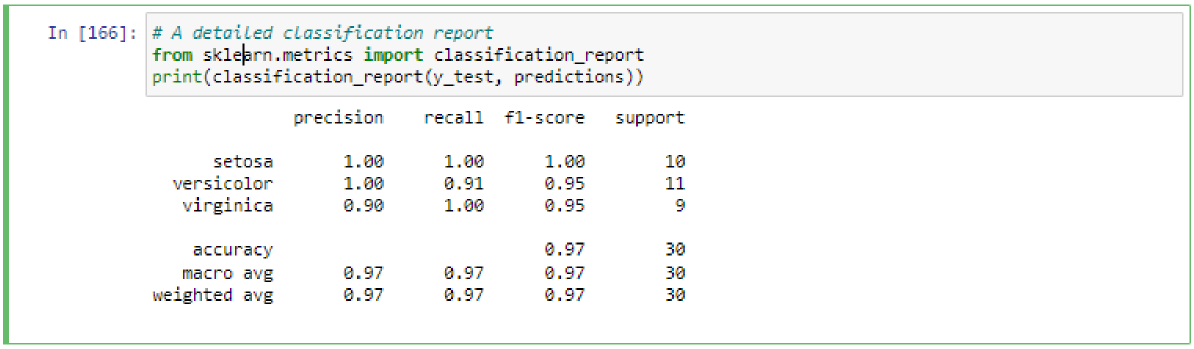
- Now we predict the classes from the test dataset using our trained model.
- Then we check the accuracy score of the predicted classes.
- accuracy_score() takes true values and predicted values and returns the percentage of accuracy.

Output:

0.9666666666666667

The accuracy is above 96%.

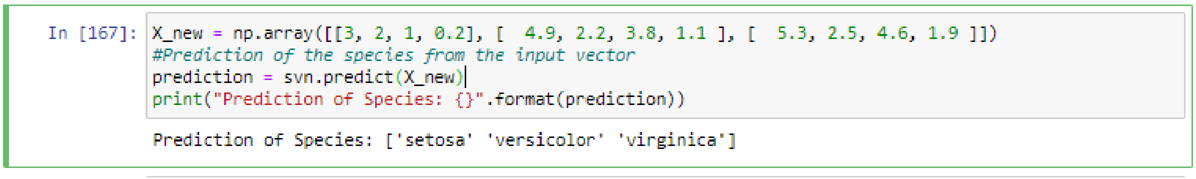
Now let’s see the detailed classification report based on the test dataset



```
In [166]: # A detailed classification report
          from sklearn.metrics import classification_report
          print(classification_report(y_test, predictions))

                        precision    recall  f1-score   support

                setosa       1.00      1.00      1.00        10
            versicolor       1.00      0.91      0.95        11
             virginica       0.90      1.00      0.95         9

              accuracy                           0.97        30
             macro avg       0.97      0.97      0.97        30
          weighted avg       0.97      0.97      0.97        30
```

- The classification report gives a detailed report of the prediction.
- Precision defines the ratio of true positives to the sum of true positive and false positives.
- Recall defines the ratio of true positive to the sum of true positive and false negative.
- F1-score is the mean of precision and recall value.
- Support is the number of actual occurrences of the class in the specified dataset.

Step 5 – Testing the model:



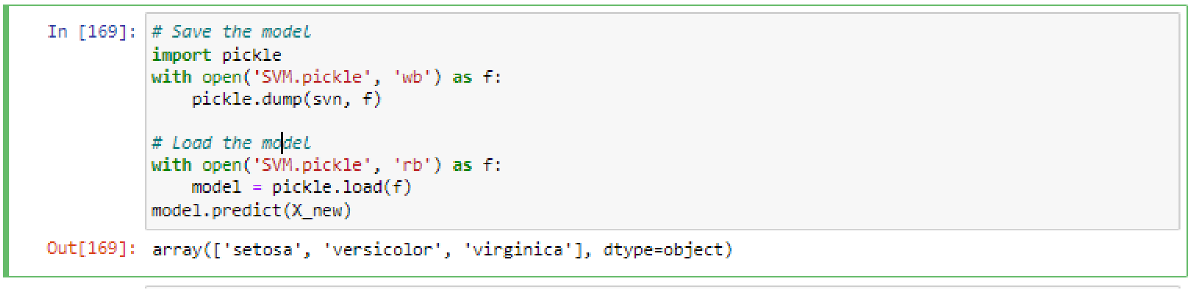
```
In [167]: X_new = np.array([[3, 2, 1, 0.2], [  4.9, 2.2, 3.8, 1.1 ], [  5.3, 2.5, 4.6, 1.9 ]])
          #Prediction of the species from the input vector
          prediction = svn.predict(X_new)
          print("Prediction of Species: {}".format(prediction))

          Prediction of Species: ['setosa' 'versicolor' 'virginica']
```

- Here we take some random values based on the average plot to see if the model can predict accurately.

Output:

Prediction of Species: [‘Iris-setosa’ ‘Iris-versicolor’ ‘Iris-virginica’]

It looks like the model is predicting correctly because the setosa is shortest and virginica is the longest and versicolor is in between these two.



```
In [169]: # Save the model
          import pickle
          with open('SVM.pickle', 'wb') as f:
              pickle.dump(svn, f)

          # Load the model
          with open('SVM.pickle', 'rb') as f:
              model = pickle.load(f)
          model.predict(X_new)

Out[169]: array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

- We can save the model using pickle format.
- And again we can load the model in any other program using pickle and use it using model.predict to predict the iris data.

Conclusion

In this project, we used the various powerful algorithms to train our data. Processing of data is also important to acquire the best result and as we can see the above results, they are very satisfactory. The accuracy score of above models are very good and they can be used to predict the species of iris flower. As we can conclude that in future with appropriate data of features of any flower it is possible to classify the species of any flower.

We can easily distinguish Setosa. Due to a lack of data, it is hard to distinguish between Virginica and Versicolor. In this project, we learned to train our own supervised machine learning model using Iris Flower Classification Project. Through this project, we learned about machine learning, data analysis, data visualization, model creation, etc.

Future Scope

After the project has been settled, the computer should have the ability to aggregate three different classifications of Iris flower to three categories. The whole workflow of machine learning should work smoothly. The users do not need to tell the computer which class the Iris belongs to, the computer can recognize them all by itself. The final purpose of this project is to let everyone who read this thesis have a basic understanding of machine learning. Even through someone never touched this field, they can realize that the machine learning algorithm will become more popular and useful in the future. Moreover, the case study of Iris recognition will show how to implement machine learning by. Become more popular and useful in the future.

REFERENCE

- https://seaborn.pydata.org/generated/seaborn.pairplot.html
- https://seaborn.pydata.org/generated/seaborn.FaceGrid.html
- https://www.kaggle.com/arshid/iris-flower-dataset
- https://www.datacamp.com/community/tutorial/machine-learning-in-r