***BAHIR DAR UNIVERSITY***


***INSTITUTE OF TECHNOLOGY FUCALITY OF COMPUTINNG***

***DEPARTMENT OF SOFTWARE ENGINEERING***

***MACHINE LEARNING INDIVIDUAL ASSIGNMENT(I)***

**Title:** **Data Preprocessing and Descriptive Statistics.**

**Submitted to: Assefa M.(Phd)**

**Submision date: 9/7/2022**

## Acknowledgment

I would like to sincerely express my gratitude to Assefa M.(Phd). for your suggestions and cooperation to complete my assignment. This assignment helps me to dive into machine learning and made me interested on this field of study. I would like to say thank you again getting teachers like you is a golden opportunity. The assignment has taken so much effort and time to complete without your support . The assignment wouldn't have been completed.

Abebe wolie

## Introduction

Data mining and machine learning are both useful tools in the field of data analysis. Classification algorithm is one of the most important techniques in data mining, therefore, it is of great significance to select suitable classification models with high efficiency to show superiority when solving classification problems with the use of Iris data.

## Data collection

A number of data is required for the model to be trained and tested. This Iris dataset is collected from Kaggle Machine Learning Repository. There are five characteristics(Feature) in the data collection, each of which corresponds to a different Iris flower species. Class (Species), Petal Length ,Petal Width , Sepal Width and, Sepal Length are the characteristics and, 1 target label, namely Species, from 150 observations. There are 50 Instances of each genus, totaling 150 examples. This dataset form is broken down numerically in (cm) volume.

In this example we can do some exploratory data analysis on the Iris data set. This Dataset contains five features (Id, length and width of sepals and petals) of 150 samples of three species of Iris (Iris setosa, Iris virginica and Iris versicolor).These measures were used to create a linear discriminant model to classify the species. This dataset is often used in data mining, classification and clustering examples and to test algorithms. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other.

The features and Their description with possible value

| No | Feature | Description | Possible value | Measuring unit |
|---|---|---|---|---|
| 1 | id | Identification number<br>Each instance of data | Numeric | Unit less |
| 2 | Petal length | The length of petal of flower | Numeric | In cm |
| 3 | Petal width | The width of petal of the flower | Numeric | In cm |
| 4 | Sepal length | The length sepal of the flower | Numeric | In cm |
| 5 | Sepal width | The width of sepal of the flower | Numeric | In cm |

***Note*** There are 150 number of instances but, each of 50 instances represent one type of class(Iris setosa,Iris virginica or Iris versicolor).

Attribute Information**:**

There are 5 attributes in the dataset:

1 Sepal length in cm
2.Sepal width in cm
3. Petal length in cm
4. Petalwidth in cm
5. Class:
   Iris Setosa
   Iris Versicolour
   Iris Virginica

.
The use of this data set is cluster analysis however is not common, since the data set only contains two clusters with rather obvious separation. One of the clusters contains Iris setosa, while the other cluster contains both Iris virginica and Iris versicolor and is not separable without the species information used. This makes the data set a good example to explain the difference between supervised and unsupervised techniques in data mining: linear discriminant model can only be obtained when the object species are known: class labels and clusters are not necessarily the same.

## Data describing

**Measures of Central Tendency**

Measures of central tendency describe the center of the data, and are often represented by the mean, the median, and the mode.

**Mean**

Mean represents the arithmetic average of the data. It is possible to calculate the mean of a particular variable in a dataset, by indexing the data frame. I try to show how it work in the coding folder.

```
import numpy as np
import pandas as pd

import matplotlib
from matplotlib import pyplot as pt
import pylab

print('Hello World')

x = '../csv/Iris.csv'  # loading CSV file,
dataframe = pd.read_csv(x)  # READING DATA USING read_csv() methode

# MEAN FOR FEATURES
sepal_length_mean = round(dataframe['SepalLengthCm'].mean(), 2)
sepal_width_mean = round(dataframe['SepalWidthCm'].mean(), 2)
petal_length_mean = round(dataframe['PetalLengthCm'].mean(), 2)
petal_width_mean = round(dataframe['PetalWidthCm'].mean(), 2)
```

**Median**

In simple terms, median represents the 50th percentile, or the middle value of the data, that separates the distribution into two halves. There is a difference between the mean and the median values of these variables. It is also possible to calculate the median of a particular variable in a data. I try to show it in the coding folder.

```
import numpy as np
import pandas as pd

import matplotlib
from matplotlib import pyplot as pt
import pylab

print('Hello World')

x = '../csv/Iris.csv'   # loading CSV file,
dataframe = pd.read_csv(x)   # READING DATA USING read_csv() methode

# MEAN FOR FEATURES
# MEDIAN FOR FEATURES
sepal_length_median = dataframe['SepalLengthCm'].median()
sepal_width_median = dataframe['SepalWidthCm'].median()
petal_length_median = dataframe['PetalLengthCm'].median()
petal_width_median = dataframe['PetalWidthCm'].median()
```

**Mode**

Mode represents the most frequent value of a variable in the data. This is the only central tendency measure that can be used with categorical variables, unlike the mean and the median which can be used only with quantitative data. The interpretation of the mode is simple.

For numerical variables, the mode value represents the value that occurs most frequently. For example, the mode value of 2.3 for the variable 'Sepal length' means that the highest number (or frequency) of applicants are 2.3cm Sepal length.

```
import numpy as np
import pandas as pd

import matplotlib
from matplotlib import pyplot as pt
import pylab

print('Hello World')

x = '../csv/Iris.csv'   # loading CSV file,
dataframe = pd.read_csv(x)   # READING DATA USING read_csv() methode

# MODE FOR FEATURES
sepal_length_mode = dataframe['SepalLengthCm'].mode()
sepal_width_mode = dataframe['SepalWidthCm'].mode()
petal_length_mode = dataframe['PetalLengthCm'].mode()
petal_width_mode = dataframe['PetalWidthCm'].mode()
```

**Measures of Dispersion**

There are various measures of central tendency. However, the values of these measures differ for many variables. This is because of the extent to which a distribution is stretched or squeezed. In statistics, this is measured by dispersion which is also referred to as variability, scatter, or spread. The most popular measures of dispersion are standard deviation, variance, and the interquartile range.

**Standard Deviation**

Standard deviation is a measure that is used to quantify the amount of variation of a set of data values from its mean. A low standard deviation for a variable indicates that the data points tend to be close to its mean, and vice versa. While interpreting standard deviation values, it is important to understand them in conjunction with the mean.

```
import numpy as np
import pandas as pd

import matplotlib
from matplotlib import pyplot as pt
import pylab

print('Hello World')

x = '../csv/Iris.csv'  # loading CSV file,
dataframe = pd.read_csv(x)  # READING DATA USING read_csv() methode

# MEASURE OF DISPERSION FOR GIVEN NUMERIC FEATURES

print("\nSepal length variance:\t ", round(np.var(dataframe["SepalLengthCm"]), 2))
print("Sepal width variance: \t", round(np.var(dataframe["SepalWidthCm"]), 2))
print("Petal length variance:\t", round(np.var(dataframe["PetalLengthCm"]), 2))
print("Petal width variance: \t", round(np.var(dataframe["PetalWidthCm"]), 2))
```

**Variance**

Variance is one of the measure of dispersion. It is the square of the standard deviation and the covariance of the random variable with itself. It is calculated using python as shown below.

```
import numpy as np
import pandas as pd

import matplotlib
from matplotlib import pyplot as pt
import pylab

print('Hello World')

x = '../csv/Iris.csv'  # loading CSV file,
dataframe = pd.read_csv(x)  # READING DATA USING read_csv() methode

# MEASURE OF DISPERSION FOR GIVEN NUMERIC FEATURES

print("\nSepal length variance:\t ", round(np.var(dataframe["SepalLengthCm"]), 2))
print("Sepal width variance: \t", round(np.var(dataframe["SepalWidthCm"]), 2))
print("Petal length variance:\t", round(np.var(dataframe["PetalLengthCm"]), 2))
print("Petal width variance: \t", round(np.var(dataframe["PetalWidthCm"]), 2))
```

**Range**

The range is the difference between the highest and lowest values in a set of given dataset feature values. It can be calculated using python as the code demonstrated below.

.

```
import numpy as np
import pandas as pd

import matplotlib
from matplotlib import pyplot as pt
import pylab

print('Hello World')

x = '../csv/Iris.csv'  # loading CSV file,
dataframe = pd.read_csv(x)  # READING DATA USING read_csv() methode

# MEASURE OF DISPERSION FOR GIVEN NUMERIC FEATURES

print('\nMEASURE OF DISPERSION')
print("\nSepal length range: [%s, %s]" % (min(dataframe["SepalLengthCm"]), max(dataframe["SepalLengthCm"])))
print("Sepal width range:  [%s, %s]" % (min(dataframe["SepalWidthCm"]), max(dataframe["SepalLengthCm"])))
print("Petal length range: [%s, %s]" % (min(dataframe["PetalLengthCm"]), max(dataframe["PetalLengthCm"])))
print("Petal width range:  [%s, %s]" % (min(dataframe["PetalWidthCm"]), max(dataframe["PetalWidthCm"])))
```

## Statistical Summery for the features in the dataset.

```
description=round(dataframe[dataframe.columns[1:]].describe(), 2)
print(description)
```

This code gives us the following out put

| | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm |
|---|---|---|---|---|
| count | 150.00 | 150.00 | 150.00 | 150.00 |
| mean | 5.84 | 3.05 | 3.76 | 1.20 |
| std | 0.83 | 0.43 | 1.76 | 0.76 |
| min | 4.30 | 2.00 | 1.00 | 0.10 |
| 25% | 5.10 | 2.80 | 1.60 | 0.30 |
| 50% | 5.80 | 3.00 | 4.35 | 1.30 |
| 75% | 6.40 | 3.30 | 5.10 | 1.80 |
| max | 7.90 | 4.40 | 6.90 | 2.50 |

**Interpretation for the above data:**

1 Sepal Length:

- The average Sepal Length of the given instance of flower is 5.84, this means half of the sample have Sepal Length of greater than or equal to 5.84cm
- maximum and the minimum Sepal Length values are 7.9cm and 4.30cm respectively.
- The first Quartile(25%) for Sepal Length is 5.10cm. This means among the given instance of dataset 25% the total sample have 5.10 or less. In other word 75% of the data have Sepal Length of 5.10 or greater than 5.10cm.
- The 75%(75$^{th}$ percentile) of Sepal Length is 6.40cm.This means 75% of the record have the Petal Length of less than or equal to 6.40cm . In other word 25% of the sample have Sepal Length of 6.40cm or above.

2 Sepal Width:

- The average Sepal Width of the given instance of flower is 3.05cm, this means half of the sample have Sepal Width of greater than or equal to 3.05cm
- maximum and the minimum Sepal Width values are 4.40cm and 2.00 cm respectively. This means the Sepal Length of the records are between 4.40cm and 2.00cm.
- The first Quartile(25%) for Sepal Length is 2.80cm. This means among the given instance of dataset 25% the total sample have 2.80cm or less. in other word 75% of the data have Sepal Length of 2.80 or greater than 2.80cm
- the 75%(75$^{th}$ percentile) Sepal Width is 3.30cm.This means 75% of the record have the Sepal Length of less than or equal to 3.30cm . In other word 25% of the sample have sepal Length of 3.30orabove

3 PetalLenth :

- The average Petal Length of the given instance of flower is 3.76cm, this means half of the sample have Petal Length of greater than or equal to 3.76cm.
- maximum and the minimum Petal Length values are 6.90cm and 1.00 cm respectively. This means the Petal Length of the records are between 6.90cm and 1.00cm.
- The first Quartile(25%) for Petal Length is 1.60cm. This means among the given instance of dataset 25% the total sample have 1.60 or less. in other word 75% of the data have Petal Length of 1.60cm or greater than 1.60cm
- The 75%(75$^{th}$ percentile) Petal Length is 5.10cm.This means 75% of the record have the Petal Length of less than or equal to 5.10cm . In other word 25% of the sample have Petal Length of 5.10 or above.

## Data Exploration, Visualization and Preprocessing.

## Data exploration

Data exploration is a technique for analyzing data using several visual techniques. This technique allows you to get detailed information about a statistical summary of your data. You can also handle duplicate and outliers and see some trends and patterns in your dataset. Visualizations are used in data exploration to create a simpler view of a dataset than simply looking up thousands of individual numbers or names. during data exploration we assess many things such as remove unnecessary feature, checking missing value, checking Duplicate values, check wheather the data is balanced etc.

**Remove unnecessary features(attributes)**

In this dataset the feature which Is called name is Id is not significant for develope the model. There for remove this column is necessary.

```
dataframe = dataframe.drop(['Id'], axis=1)
print(dataframe)
```

This prints the data Frame without the Id column because the Id column is droped out.

**Checking Missing values**

Check if the data contains missing values. Values may be missing if information for one or more items or the entire unit is not provided. use the isnull () method.

```
b = dataframe.isnull().sum()
print(b)
```

This provides the output like below

```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

**Checking Duplicate values**
Let's see if the dataset contains duplicates. Panda's drop_duplicates () method helps remove duplicates from the data frame.

```
data = dataframe.drop_duplicates(subset ="Species",)
```

**Checking wheather the data is balanced**

Make sure your dataset is balanced, that is, all species contain the same number of rows. Use the Series. value_counts () function. This function returns a series containing a number of unique values.

```
dataframe.value_counts("Species")
```

The output look like as the following

```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

We can see that all the species contain an equal amount of rows, so we should not delete any entries. because it is balanced data.

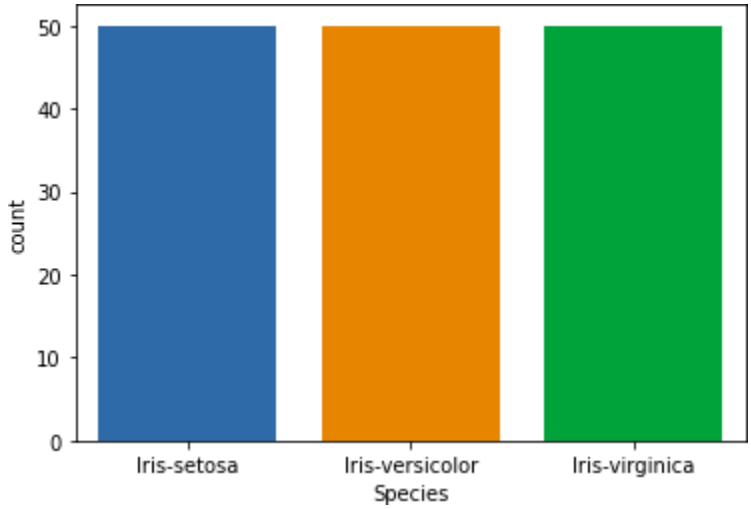
## Data visualization

### Visualizing the target column

The target column will be a type column because you will only need the results for each type in the end. Let's take a look at the countplot species.
Use matplotlib and the seaborn library for visualization our data.

```
import seaborn as sns
import matplotlib.pyplot as plt


sns.countplot(x='Species', data=dataframe, )
plt.show()
```

The above code gives us the following out put



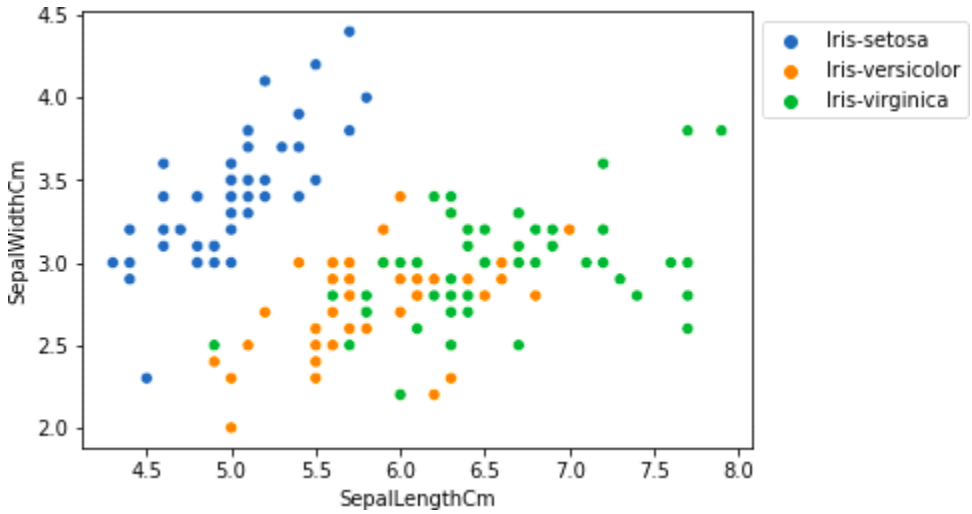

## Relation between variables

We will see the relationship between the sepal length and sepal width and also between petal length and petal width. for example let's Petal Length and Petal Width.

**1 Comparing Sepal Length and Sepal Width**

```
import seaborn as sns
import matplotlib.pyplot as plt


sns.scatterplot(x='SepalLengthCm', y='SepalWidthCm',
                hue='Species', data=dataframe, )

plt.legend(bbox_to_anchor=(1, 1), loc=2)
plt.show()
```

The above code outputs

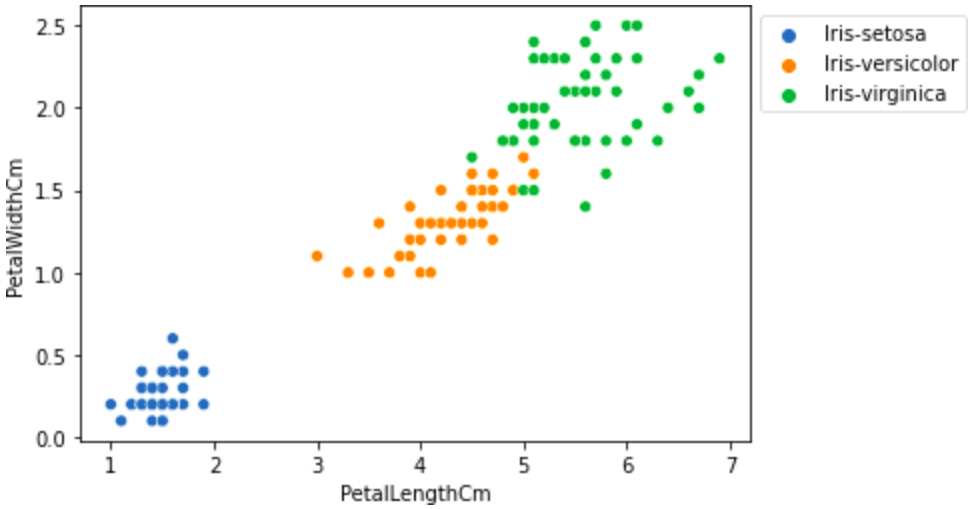



From the figure above, we can conclude that the length of the sepals of –
species is small, but the width of the sepals is large.
The Versicolor species is between the other two species in terms of sepal length and width.
The species Virginia has long sepals but small sepals.

**2 Comparing Petal Length and Petal Width.**

```
sns.scatterplot(x='PetalLengthCm', y='PetalWidthCm',
                hue='Species', data=dataframe, )
plt.legend(bbox_to_anchor=(1, 1), loc=2)

plt.show()
```

The above code output looks like:




From the chart above, we can conclude that setosa has a small petal length and width, and the

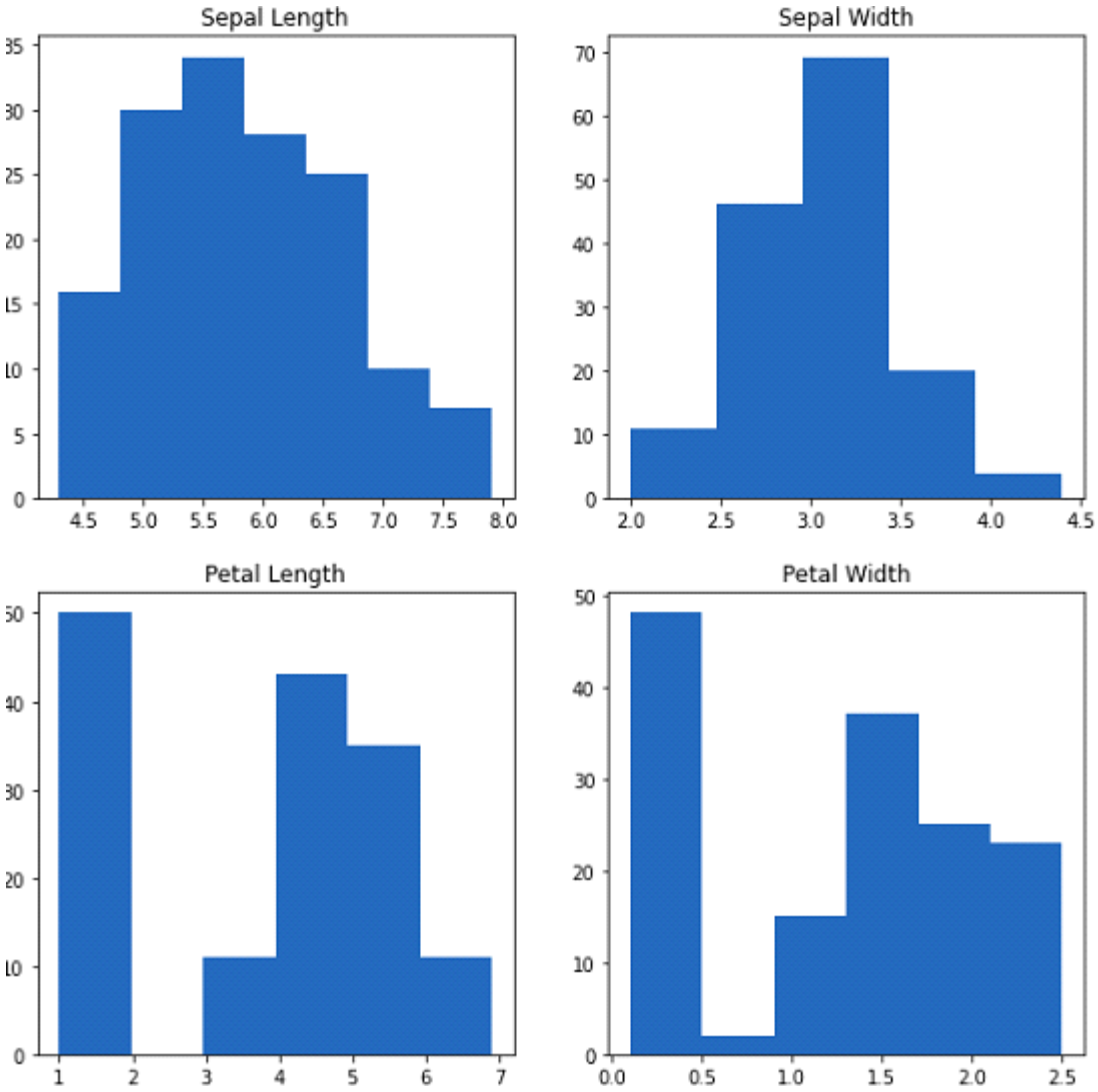
Versicolor species is in the middle of the other two species in terms of petal length and width. Virginica species have the largest petals in length and width.

## Histograms

A histogram is a display of statistics that use rectangles to show the frequency of data items at consecutive numerical intervals of the same size. In the most common form of histogram, independent variables are plotted along the horizontal axis and dependent variables are plotted along the vertical axis. Histograms allow seeing the distribution of data for various columns. It can be used for uni as well as bi-variate analysis.

```
fig, axes = plt.subplots(2, 2, figsize=(10,10))

axes[0,0].set_title("Sepal Length")
axes[0,0].hist(dataframe['SepalLengthCm'], bins=7)

axes[0,1].set_title("Sepal Width")
axes[0,1].hist(dataframe['SepalWidthCm'], bins=5);

axes[1,0].set_title("Petal Length")
axes[1,0].hist(dataframe['PetalLengthCm'], bins=6);

axes[1,1].set_title("Petal Width")
axes[1,1].hist(dataframe['PetalWidthCm'], bins=6);

plt.show()
```

The above code output looks like:




From the above plot, we can see that

- The highest frequency of the sepal length is between 30 and 35 which is between 5.5 and 6
- The highest frequency of the sepal Width is around 70 which is between 3.0 and 3.5
- The highest frequency of the petal length is around 50 which is between 1 and 2
- The highest frequency of the petal width is between 40 and 50 which is between 0.0 and 0.5

## Handling Correlation

Correlation is a statistical measure of how two variables are linearly related (that is, change together at a constant rate). This is a common tool for describing simple relationships without mentioning the cause and effect.

Panda's **dataframe.corr ()** is used to find pairwise correlations for all columns in a dataframe. All Numeric values are automatically excluded. Ignored for all non-numeric datatype columns in the data frame.

```
correlation = dataframe.corr(method='pearson')
print(correlation)
```

The above code output looks like:

```
>                SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
SepalLengthCm       1.000000     -0.109369       0.871754      0.817954
SepalWidthCm       -0.109369      1.000000      -0.420516     -0.356544
PetalLengthCm       0.871754     -0.420516       1.000000      0.962757
PetalWidthCm        0.817954     -0.356544       0.962757      1.000000
```

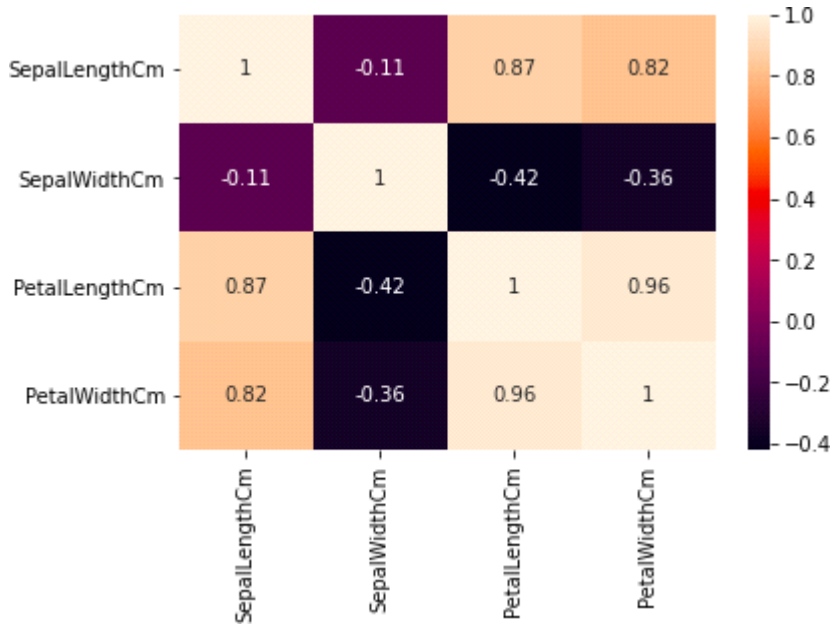
## Heatmaps

A heat map is a two-dimensional visualization of data in which colors stand in for values. A straightforward heat map offers a quick visual representation of the data. The user can comprehend complex data sets with the help of more intricate heat maps. can be presented in a variety of ways, but they all have one thing in common: they make use of color to convey correlations between data values that would be far more difficult to comprehend if provided numerically in a spreadsheet.

Heatmaps are a type of data visualization method for two-dimensional color analysis of datasets. In essence, it displays the relationship between each numerical variable in the dataset. Simply expressed, you can plot the aforementioned correlations using heatmaps.

```
sns.heatmap(dataframe.corr(method='pearson'), annot=True)
plt.show()
```

The above code output looks like:




The graph shown above demonstrates that

- Petal length and width exhibit strong connections.
- Good relationships exist between sepal breadth and petal length.
- There are strong relationships between petal width and sepal length.

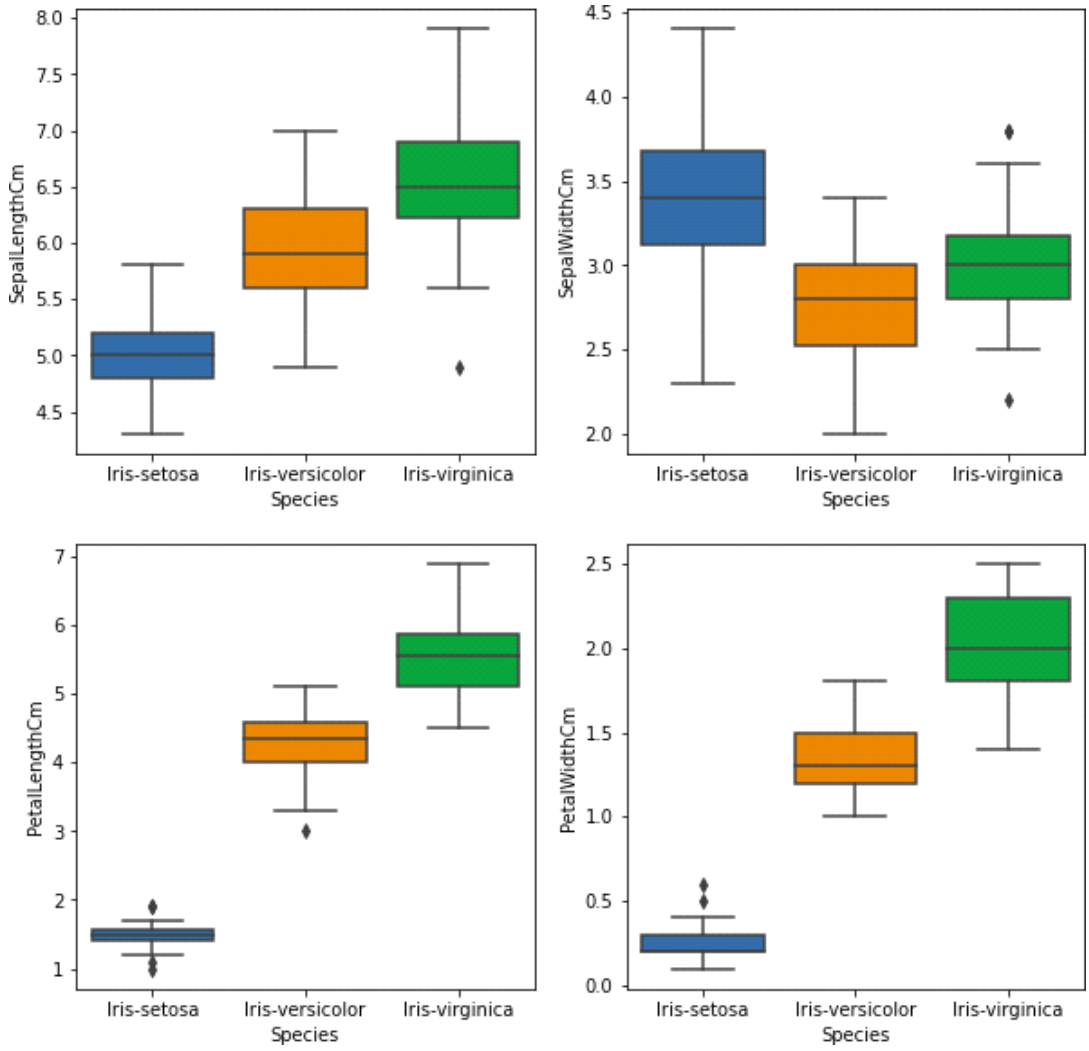
## Box Plots

A box plot is a chart that shows data from a five-number summary including one of the measures of central tendency. It does not show the distribution in particular as much as a stem and leaf plot or histogram does. But it is primarily used to indicate a distribution is skewed or not and if there are potential unusual observations (also called outliers) present in the data set. Boxplots are also very beneficial when large numbers of data sets are involved or compared.

Simply we can define a boxplot in terms of descriptive statistics concepts. In other words, a box plot or whisker plot is a way to plot a group of numerical data in quartiles. These may have some lines extending from the boxplot. The terms boxplot and boxplot are used because they show variations outside the lower and upper quartiles. Outliers can be specified as a single point.

```
def graph(y):
    sns.boxplot(x="Species", y=y, data=dataframe)


plt.figure(figsize=(10, 10))

plt.subplot(221)
graph('SepalLengthCm')

plt.subplot(222)
graph('SepalWidthCm')

plt.subplot(223)
graph('PetalLengthCm')

plt.subplot(224)
graph('PetalWidthCm')

plt.show()
```

The above code output looks like:




From the above graph, we can conclude

- Species Setosa has the smallest features and less distributed with some outliers.
- Species Versicolor has the average features.
- Species Virginica has the highest features

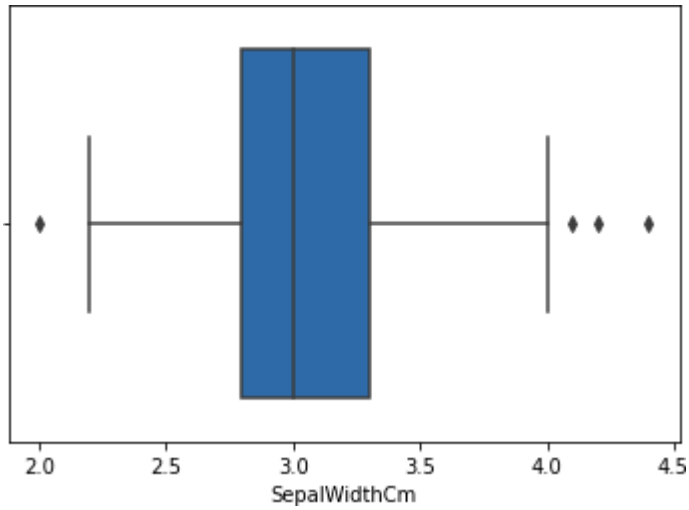
## Handling Outliers

Outliers are extreme values that stand out greatly from the pattern of values in a very dataset or graph. In simple terms, an outlier is a particularly high or extremely low information relative to the closest datum and also the remainder of the neighboring co-existing values in a very data graph or dataset you're working with.

An Outlier could be a data-item/object that deviates significantly from the remainder of the (so-called normal)objects. they'll be caused by measurement or execution errors. The analysis for outlier detection is observed as outlier mining. There are many ways to detect the outliers, and also the removal process is that the data frame same as removing an information item from the panda's dataframe.

```
sns.boxplot(x='SepalWidthCm', data=dataframe)
plt.show()
```

The above code output looks like:




- In the above graph, the values above 4 and below 2 are acting as outliers.

## *ASSIGNMENT II*

## Model development

The model-building process involves setting up ways to collect data, understanding and paying attention to what is important in the data to answer the questions to be, finding a statistical, mathematical or simulation model to gain understanding and make predictions.

Labeling categorical features, clearing data and normalization has been done in the previous data pre processing phase. In this phase I am going to develop a model for this iris dataset and test it with different algorithm.

### Splitting data into feature and targets

Before implementing any model it is neccessary to split the dataset to train and test sets. **train_test_split** class from **sklearn.model_selection** library is used for split our dataset..Then allocating 80% of data for training tasks and the remainder 20% for validation purposes.

```
#IMPORT NECESSARY LIBRARIES
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
# Load the iris data
dataFrame=pd.read_csv('Iris.csv')
dataFrame=dataFrame.drop('Id',axis=1)  # REMOVE THE ID COLUMN IT IS NOT NECESSARY FOR DEVELOPE MODEL

#SPLITTING  THE DATASET INTO FEATURES AND TARGET DATA
featurData=dataFrame.drop('Species',axis=1)  # THE TARGET DATA
targetData=dataFrame['Species']              #THE FEATURE DATA

# TEST_SIZE. SET THE SIZE OF THE TEST DATA TO BE 20% OF THE FULL DATASET.
feature_train, feature_test, targat_train, target_test = train_test_split(featurData, targetData,
    test_size=0.2)
```

## Train the model (Modeling)

Now, to determine the best suitable algorithm for getting effective accuracy. I going to evaluate Two famous frequently used algorithms such as.

- ⇒ K-nearest neighbors algorithm
- ⇒ Support Vector Machine

Let's get started on the model and estimate the accuracy of each algorithm. We can also see which method produces the best results. First, we can use the first algorithm, KNN. We can construct our model as follows:

```
#  TRAINING THE MODEL USING K-NEAREST NEIGHBOR
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

model = KNeighborsClassifier()
model.fit(feature_train, targat_train)
```

## Classify using the developed Model

```
# PREDICTION USING DEVELOPED MODEL
predictOnTraining=model.predict(feature_train)
predictOnTest=model.predict(feature_test)
```

### Performance(classification) Report on dataset

per-class representation of the key classification metrics may be seen in the classification report. This provides a better understanding of the classifier's behavior relative to overall accuracy.

True and false positives and false negatives are used to determine the metrics. In this instance, the terms positive and negative are simply names for the classes in a binary classification problem. We would regard true and false to be occupied in the aforementioned scenario, as well as true and false to be vacant. Therefore, a true positive occurs when both the actual class and the estimated class are positive. When the actual class is negative but the estimated class is positive, this is known as a false positive. These terms are used to define the meterics.

**precision**

Precision is the ability of a classifer not to label an instance positive that is actually negative. For each class it is defined as as the ratio of true positives to the sum of true and false positives. Said another way, “for all instances classified positive, what percent was correct?”

**recall**

Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives. Said another way, “for all instances that were actually positive, what percent was classified correctly?”

**f1 score**

The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. Generally speaking, F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy.

**support**

Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or re-balancing. Support doesn’t change between models but instead diagnoses the evaluation process.

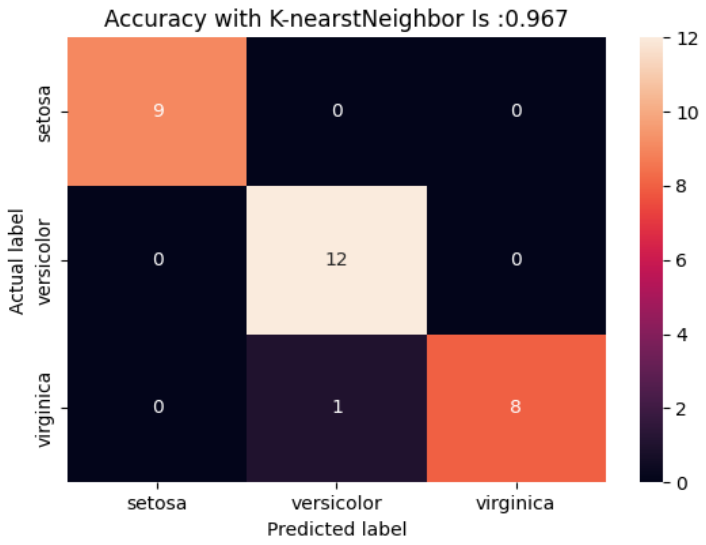
```
# CLASSIFICATION REPORT
from sklearn.metrics import classification_report

print(classification_report(target_test,predictOnTest))
```

Output looks like:

```
>                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         8
Iris-versicolor       1.00      0.92      0.96        13
 Iris-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
```

⇒ now let's draw a heat map for finding what percentage of flowers that are predicted from model are correct comparing with actual class labels.

```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

print("classification_report")
print(classification_report(target_test,predictOnTest))

# Creates a confusion matrix
cm = confusion_matrix(target_test, predictOnTest)
# Transform to df for easier plotting
cm_df = pd.DataFrame(cm,
                     index = ['setosa','versicolor','virginica'],
                     columns = ['setosa','versicolor','virginica'])
sns.heatmap(cm_df, annot=True)
plt.title('Accuracy Is :{0:.3f}'.format(accuracy_score(target_test, predictOnTest)))
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```



From the above confusion matrix we can conclude that:-

1. All setosa flowers are correctly classified by model.
2. 12 versicolor flowers are correctly classified and 1 versicolor is wrongly classified as viginica
3. 8 virginica are correctly classified by our model and 1versicolor is wrongly classified as versicolor.

**note**:values may change when we run the same program multiple times because train and test set in this case are changing.

## Developing model with support vector machine(SVM) algorithm

From the above we developed model using KNN Algorithm and we made some manipulation in the dataframe such as split data into Feature and targat as well as train and test. therefore it is not necessarily to repeat that process I just use that and try to develop only the model compare its accuracy.

```
# PREDICTION USING DEVELOPED MODEL
predictOnTraining=model.predict(feature_train)
predictOnTest=model.predict(feature_test)

model=SVC()
model.fit(feature_train, targat_train)
```

Let's predict using the trained model

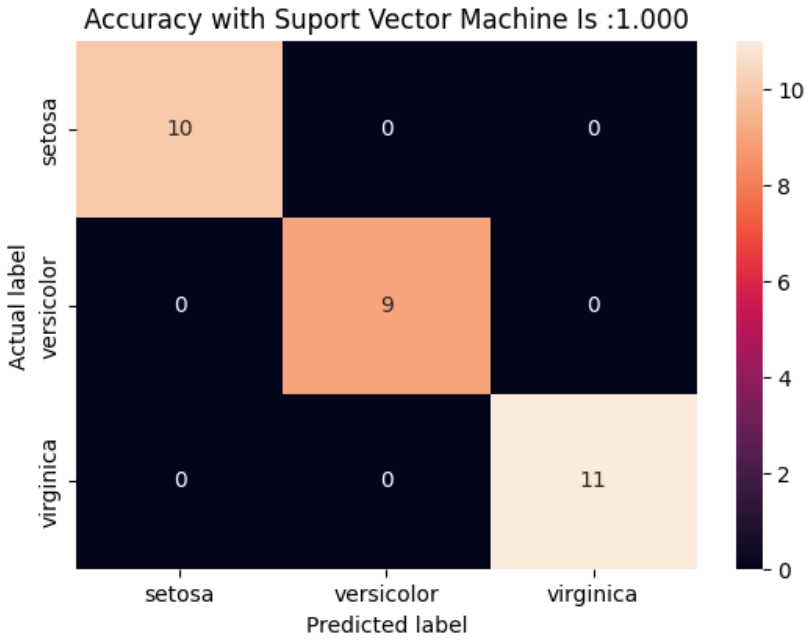
### *Performance Report on dataset*

I have already described about attribute of performance report in detail from the previous model now it is enough to show only the codes.

```
# PORFORMANCE(CLASSIFICATION) REPORT
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

print("classification_report")
print(classification_report(target_test,predictOnTest))

# Creates a confusion matrix
cm = confusion_matrix(target_test, predictOnTest)
# Transform to df for easier plotting
cm_df = pd.DataFrame(cm,
                     index = ['setosa','versicolor','virginica'],
                     columns = ['setosa','versicolor','virginica'])
sns.heatmap(cm_df, annot=True)
plt.title('Accuracy Is :{0:.3f}'.format(accuracy_score(target_test, predictOnTest)))
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Iris-setosa | 1.00 | 1.00 | 1.00 | 10 |
| Iris-versicolor | 1.00 | 1.00 | 1.00 | 9 |
| Iris-virginica | 1.00 | 1.00 | 1.00 | 11 |
| accuracy | | | 1.00 | 30 |
| macro avg | 1.00 | 1.00 | 1.00 | 30 |
| weighted avg | 1.00 | 1.00 | 1.00 | 30 |

## Confusion matrix




From the above observation we can conclude the following assumption

1. All setosa flowers are correctly classified by model.
2. All versicolor flowers are correctly classified by model.
3. All virginica flowers are correctly classified by model.

**Note :**values may change when we run the same program multiple times because train and test set in this case are changing. And we observed the accuracy of the model is 100% is looks like unreal, because in real world application no model can have 100% accuracy but since I used small amount of data and the algorithm which I used is support vector machine(it is much strong algorithm) cause them model sames like unreal(100%) Accuracy.

## Compare and contrast the performance of your models for the different algorithms

Machine learning algorithms employ a variety of statistical, probabilistic and optimization methods to learn from past experience and detect useful patterns from large, unstructured and complex datasets . classifier learns an input features from a dataset using specific approach and tuning parameters, develop a classification model, and use the model to predict the corresponding class of new input in an unseen dataset. different variants of machine learning algorithms have relative performance of for dataset prediction. This important information of relative performance can be used to aid researchers in the selection of an appropriate machine learning algorithm for their studies.

Now let's compare the performance of the two well known algorithms K-nearest neighbors(KNN) And Support vector machine(SVM).

Support Vector Machines (SVM) and k-Nearest Neighbor (kNN) are two common machine learning algorithms. They are used to learn patterns in data. The kNN and SVM are good for classifying data, but each has its own strengths and weaknesses. The SVM uses a hyperplane to divide the input space into different categories, and then uses the object's location on the hyperplane to classify it.. The kNN uses a voting system to determine which class an unclassified object belongs to, based on the classes of the nearest neighbors in the decision space. The SVM is very fast at classifying data, doing so in seconds compared to the kNN, which takes longer than the SVM. The k-nearest-neighbor classifier is usually accurate;; however, it sometimes makes a few small mistakes that interfere with the final classification. The SVM will occasionally incorrectly classify a large object. While both algorithms yield positive results regarding the accuracy in which they classify the images, the SVM provides significantly better classification accuracy and classification speed than the kNN.

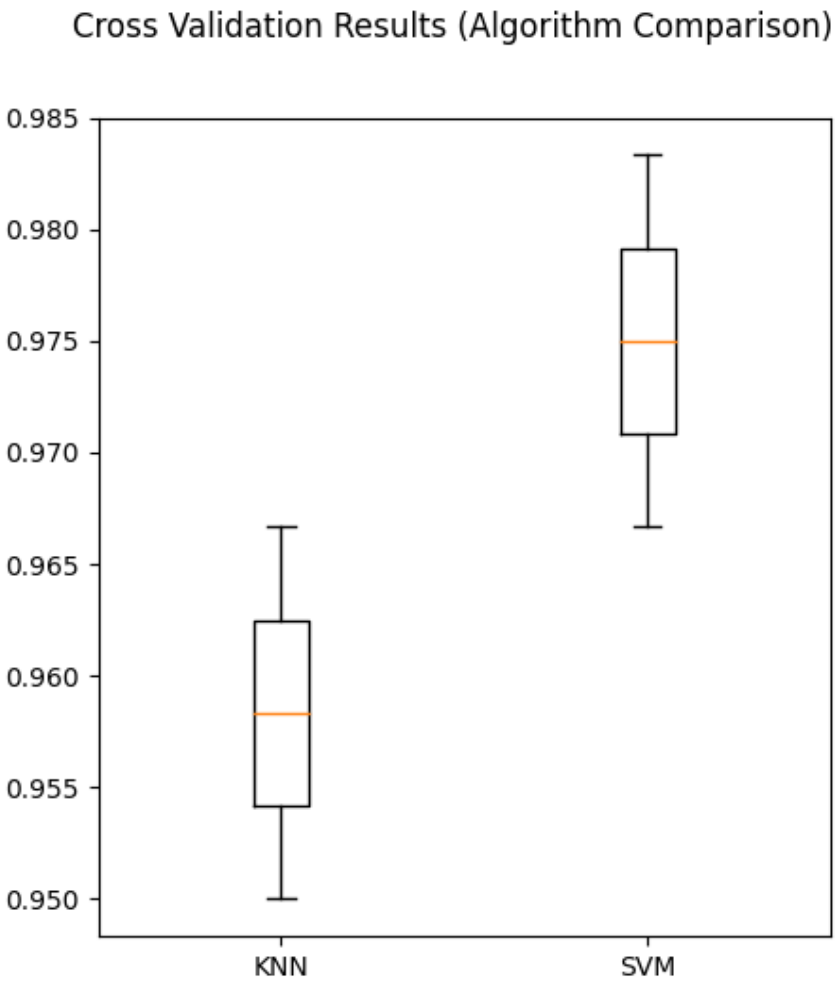
**Note :** In my demonstration SVM was little better than KNN but Since I used variable data the performance may vary . Even there might be the accuracy flactution between these two Algorithms demonstrated above

I try to demonstrate cross validation result and visualizing the comparison in the following code sample.

```
def evaluateAlgorithm(feature_train, feature_test, targat_train, target_test):
    ##Machine Lreaning Algorithm, Parameter setting

    ## Compare different ML Algorithms with default parameter setting
    model_List = []
    model_List.append(('KNN',   'K Neighbors Classifier',       KNeighborsClassifier()))
    model_List.append(('SVM',   'Support Vector Machine ',      SVC()))

    ##Cross Validation
    print("Cross Validation Results ")
    outcomes = []
    description = []
    shortDescription = []
    for shtDes, des, model in model_List:
        cv_results = cross_val_score(model, feature_train, targat_train, cv = 2,
                                     scoring='accuracy', n_jobs = -1, verbose = 0)
        outcomes.append(cv_results)
        description.append(des)
        shortDescription.append(shtDes)
        prt_string = "\n %s:\n \tMean Accuracy: %f (Std: %f)" % (des, cv_results.mean()
                                                                 , cv_results.std())
        print(prt_string)

    ##Visualise the outcomes / results from Cross Validation
    fig = plt.figure(figsize = (5,6))
    fig.suptitle('Cross Validation Results (Algorithm Comparison)')
    ax = fig.add_subplot(111)
    plt.boxplot(outcomes)
    ax.set_xticklabels(shortDescription)
    plt.show()

evaluateAlgorithm(feature_train,feature_test,targat_train,target_test)
```

The above code produces the following out put

```
Cross Validation Results

 K Neighbors Classifier:
    Mean Accuracy: 0.958333 (Std: 0.008333)

 Support Vector Machine :
    Mean Accuracy: 0.975000 (Std: 0.008333)
```




## conclusion

Use of machine learning (ML) in clinical research is growing steadily given the increasing availability of complex data sets. ML presents important advantages in terms of predictive performance and identifying undiscovered phenomena. Despite this popularity, many researchers are not yet familiar with the evaluation and interpretation of ML analysis. Consequently, readers and peer-reviewers alike may either overestimate or underestimate the validity and credibility of an ML-based model. Currently machine learning is  become the huge research area with continuously evoliving algorithms . In this assignment  I try to demonstrate the K nearest neighbor(KNN) and Support Vector Machine(SVM)  from date description upto developing the model and even comparing these two well known algorithms based on iris data set. From my observation both of these two Algorithms have almost the same accuracy. But I recommended to use  SVM because of its speed.

**Softwares and Packeges used**

- PyCharm Community Edition
- **Python 3.10.5**
- Google Docs

**References**


https://www.kaggle.com/
**https://developers.google.com/**
**https://www.simplilearn.com/tutorials/data-analytics-tutorial/**
**https://www.datacamp.com/tracks/career**
**https://opendatascience.com/12-excellent-datasets-for-data-visualization-in-2022/**
**https://www.techtarget.com/**
**https://www.ibm.com/cloud/learn/machine-learning**