**SUPERVISED MACHINE LEARNING: CLASSIFICATION**

**By: Botwe Dennis Kweku**

## About Data

The Iris flower data set or Fisher's Iris data set is one of the most famous multivariate data set used for testing various Machine Learning Algorithms.

The Iris flower data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper The use of multiple measurements in taxonomic problems.

The data set consists of 50 samples from each of three species of Iris (Iris-Setosa, Iris-virginica, and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.

The dataset was obtained from Kaggle, (*https://www.kaggle.com/datasets/arshid/iris-flower-dataset*)

## Initial Action Plan

- First load in the data.
- Analytical Approach
- Data Overview
- Feature Engineering and Cleaning
- Data Visualizations
- Evaluation on the models
- Insights
- Conclusion and Recommendations

## Analytical Approach

This task requires classification models because the target variable, Species, is categorical. First, the data is going to prepared, cleaned and explored to gain key findings and insights. The outcome will be used to develop classification models. The goal is to create Classification models that will predict which class the flower is, based on petal and sepal sizes and the models are going to be evaluated based on their **Predictability**.

## Data Overview

### Data and Statistical Insights

| | 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 |

```
print(f'Number of rows in the data: {data.shape[0]}')
print(f'Number of columns in the data: {data.shape[1]}')
```

```
Number of rows in the data: 150
Number of columns in the data: 6
```

The dataset is a CSV file which contains a set of 150 records under 6 attributes

- Id
- Petal Length
- Petal Width
- Sepal Length
- Sepal width
- Species (Target Variable)

```
<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
```

We can see that only one column has categorical data and all the other columns are of the numeric type with non-Null entries.

Getting a quick statistical summary of the dataset using the describe(). The count of each column along with their mean value, standard deviation, minimum and maximum values can be seen in the figure below and this gives a good picture of the distribution of data.

| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm |
|---|---|---|---|---|---|
| **count** | 150.000000 | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| **mean** | 75.500000 | 5.843333 | 3.054000 | 3.758667 | 1.198667 |
| **std** | 43.445368 | 0.828066 | 0.433594 | 1.764420 | 0.763161 |
| **min** | 1.000000 | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| **25%** | 38.250000 | 5.100000 | 2.800000 | 1.600000 | 0.300000 |
| **50%** | 75.500000 | 5.800000 | 3.000000 | 4.350000 | 1.300000 |
| **75%** | 112.750000 | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| **max** | 150.000000 | 7.900000 | 4.400000 | 6.900000 | 2.500000 |

## Data Preparation and Feature Engineering

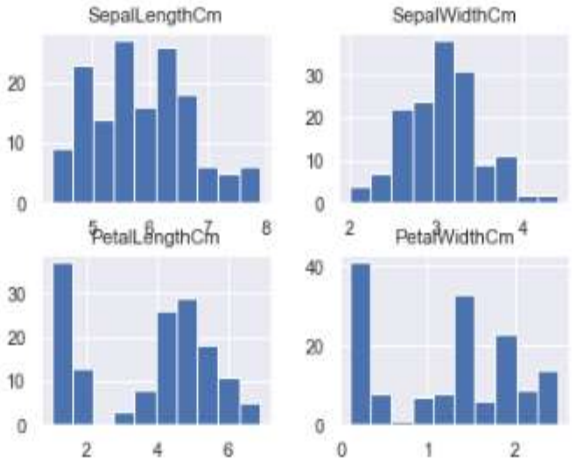
- Removed ‘Iris-’ from the Species column since it will affect the model negatively

| | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | 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 |

- There were no missing values in the data as below

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

- By inspection each column is below the skew limit. The data contained no skewed values.




```
SepalLengthCm    0.314911
SepalWidthCm     0.334053
PetalLengthCm   -0.274464
PetalWidthCm    -0.104997
dtype: float64
```

- The data was label encoded and 3 arrays was created

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

data['Species'] = le.fit_transform(data['Species'])

le.classes_
```

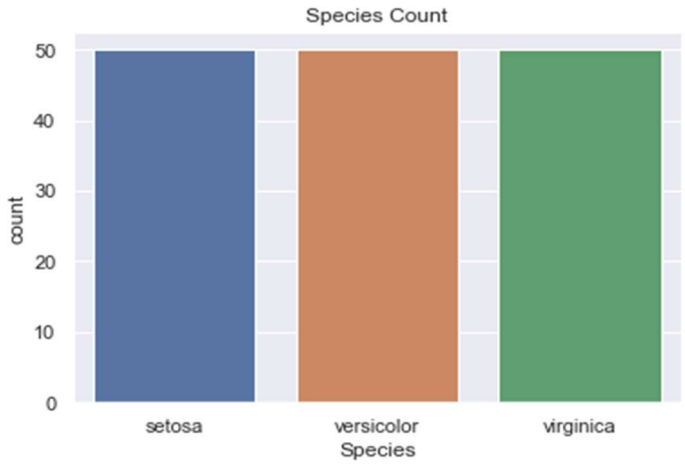
```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

```
data.Species.unique()
```

```
array([0, 1, 2])
```

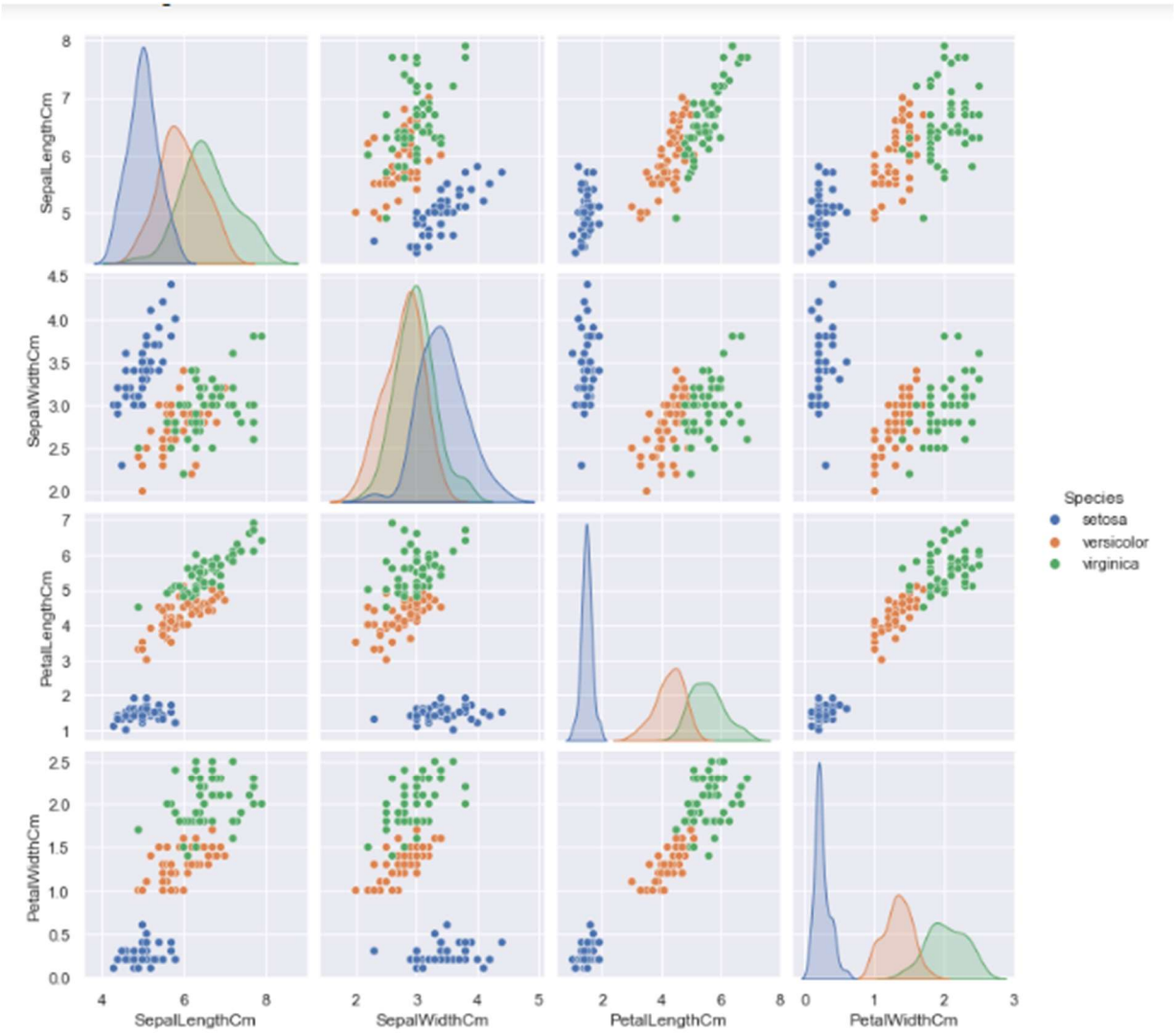
- Dropped the Id column since it is a unique value and is useless to the model

## Data Visualization




- This visualizes that species are well balanced
- Each species (Virginica, Setosa, Versicolor) has 50 as it's count

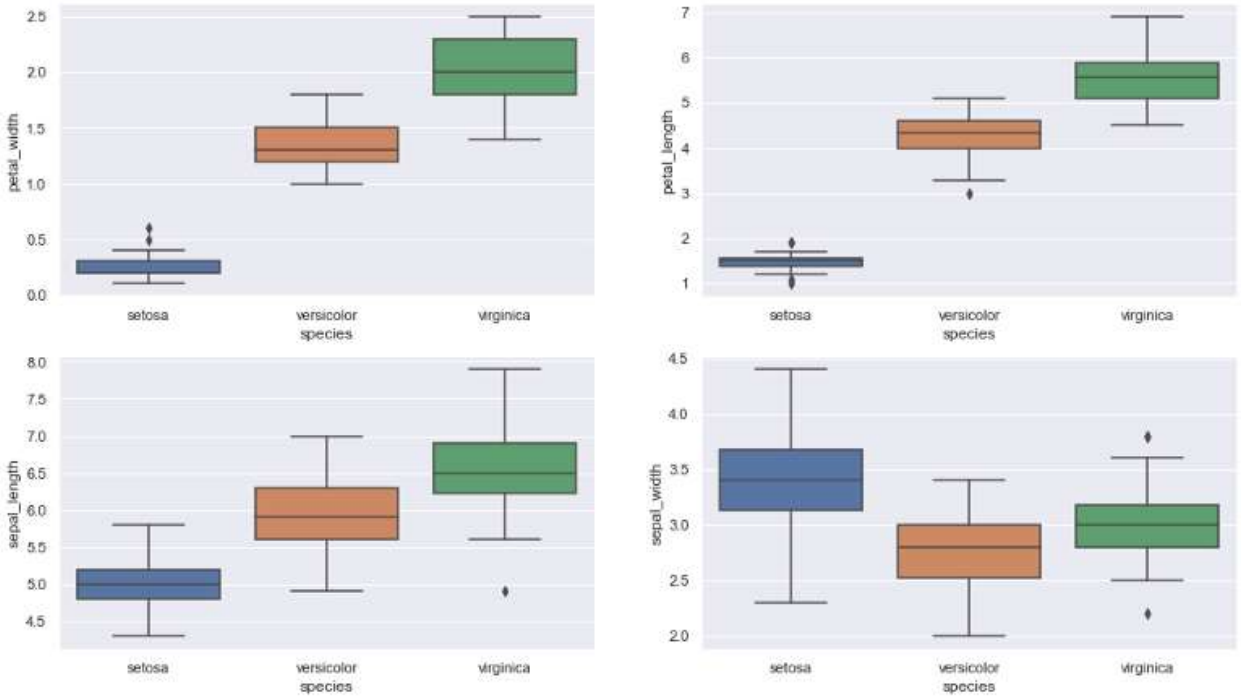
## Pairplot



It can be seen in the pairplot() above that petal length and width columns have a high correlation. Setosa has both a low petal length and a low petal width. Versicolor has both an average petal length and a high width. Virginica has both a high petal length and width. Sepal width for Setosa is high and length is low.

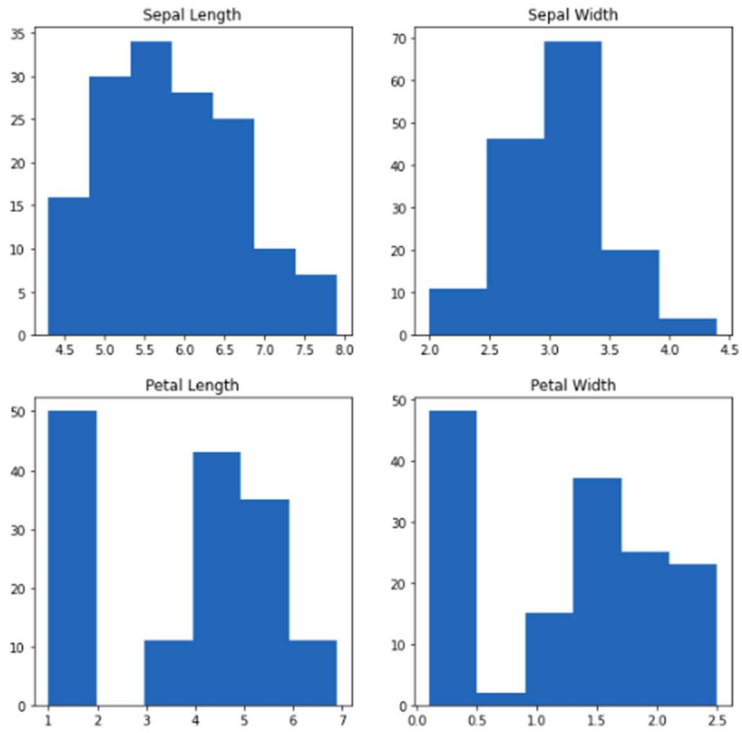
## Boxplot

A boxplot to see how the categorical feature “Species” is distributed with all other four input variables



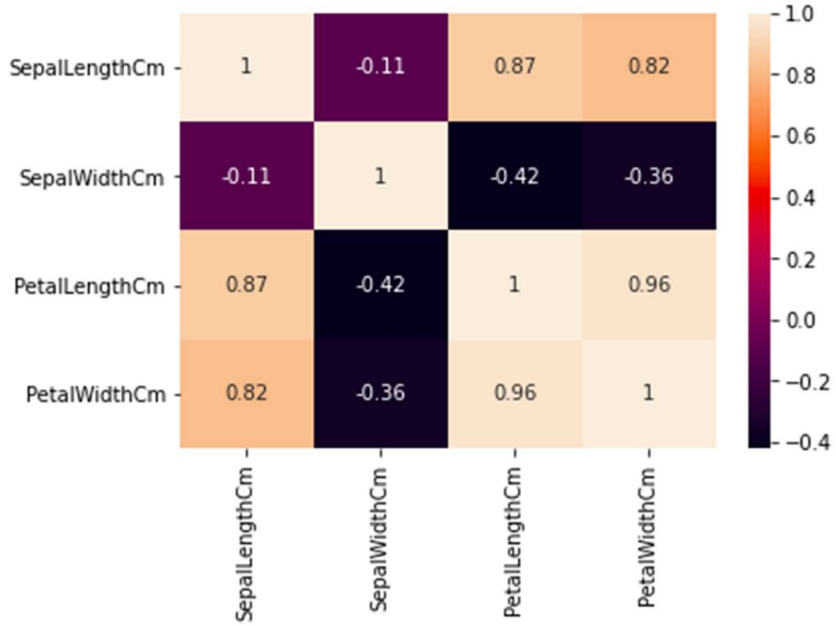

## Histogram

Histograms to display the distribution of data for various columns.




## Correlation

Sepal Length and Petal Width features are strongly correlated with each other.



## Classification Models

*Models trained*

- Logistic Regression
- Support Vector Classifier
- k-Nearest Neighbors

## Train Test Split

Once we separate the features from the target, we can create a train and test class. As the names suggest, we will train our model on the train set, and test the model on the test set. We will randomly select 70% of the data to be in our training, and 30% as test.

## Classifier Model 1: Logistic Regression

Now that the dataset has been cleaned and explored, the development of a model can proceed. The goal is to create a Logistic Regression classification model that will predict which class the flower is based on petal and sepal sizes. The model will train on the train set, and test the model on the test set. 70% of the data was randomly selected for training, and 30% for test.

Without any adjustments or tuning, this model is already performing very well with a test score of .9667 and a cross validation score of .9499. This means that the model is predicting the correct class for the flower about 97% of time.

| | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm |
|---|---|---|---|---|
| 0 | 0.413873 | 1.364041 | -2.159912 | -0.952707 |
| 1 | 0.225168 | -1.318099 | 0.640244 | -1.283935 |
| 2 | -1.504495 | -1.438993 | 2.232484 | 2.289684 |

Sepal Length was barely a deciding factor if a flower was in class 1, but petal width was a strong predictor for class 2.

Comparing the Predicated Values, the predictions line up almost perfectly, and only once the model incorrectly predicted that a flower belonged to class 2 when it really belonged to class 1.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **actual** | 2 | 0 | 1 | 0 | 0 | 0 | 2 | 2 | 2 | 1 | ... | 1 | 1 | 0 | 0 | 2 | 2 | 0 | 0 | 1 | 1 |
| **predicted** | 2 | 0 | 1 | 0 | 0 | 0 | 2 | 2 | 2 | 1 | ... | 1 | 2 | 0 | 0 | 2 | 2 | 0 | 0 | 1 | 1 |

Checking model's performance by looking at the classification report. It shows the precision, recall, f1 scores, and accuracy scores, and below is a very brief summary of these features.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 10 |
| 1 | 1.00 | 0.90 | 0.95 | 10 |
| 2 | 0.91 | 1.00 | 0.95 | 10 |
| accuracy | | | 0.97 | 30 |
| macro avg | 0.97 | 0.97 | 0.97 | 30 |
| weighted avg | 0.97 | 0.97 | 0.97 | 30 |

**Confusion Matrix**


0- Setosa
1- Versicolor
2- Virginica

## Classifier Model 2: Support Vector Classifier

The goal is to create a Support Vector Classifier model that will predict which class the flower is based on petal and sepal sizes. The model will train on the train set, and test the model on the test set. 70% of the data was randomly selected for training, and 30% for test.

The data was fit into the SVC model and now it predicted the classes from the test dataset using our trained model. The Support Vector Classifier model yielded an accuracy score of 0.966667. This means that the model is predicting the correct class for the flower about 97% of time.

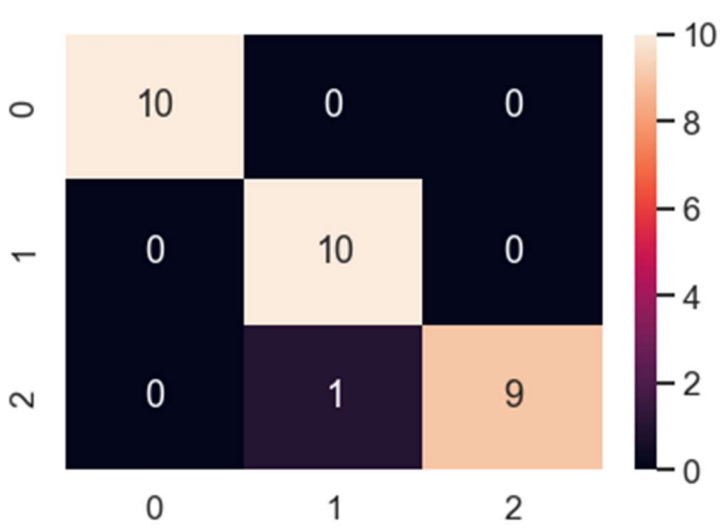
Comparing the Predicated Values, the predictions line up almost perfectly, and only once the model incorrectly predicted that a flower belonged to class 1 when it really belonged to class 2.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| actual | 2 | 0 | 1 | 0 | 0 | 0 | 2 | 2 | 2 | 1 | ... | 1 | 1 | 0 | 0 | 2 | 2 | 0 | 0 | 1 | 1 |
| predicted | 2 | 0 | 1 | 0 | 0 | 0 | 2 | 2 | 2 | 1 | ... | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 0 | 1 | 1 |

The classification report gives a detailed report of the prediction. It looks like the model is predicting correctly and it doing well on all three classes.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 10 |
| 1 | 0.91 | 1.00 | 0.95 | 10 |
| 2 | 1.00 | 0.90 | 0.95 | 10 |
| accuracy | | | 0.97 | 30 |
| macro avg | 0.97 | 0.97 | 0.97 | 30 |
| weighted avg | 0.97 | 0.97 | 0.97 | 30 |

**Confusion Matrix**

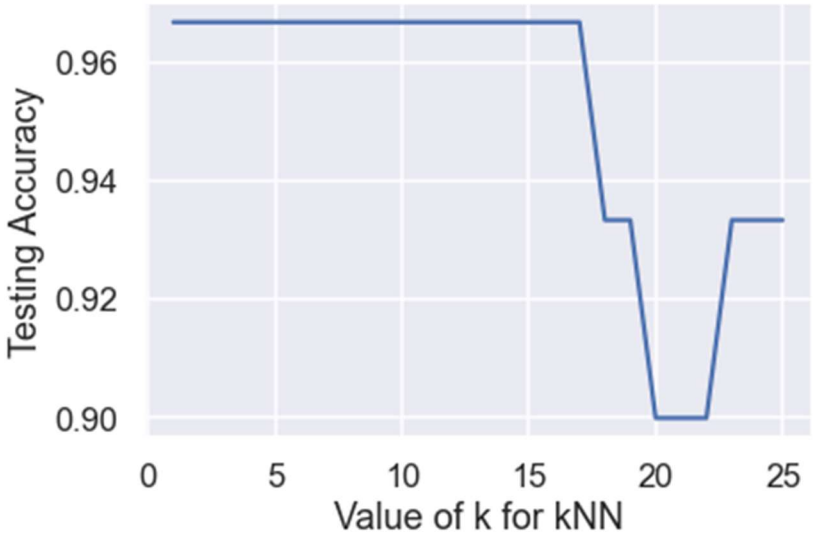



0- Setosa
1- Versicolor
2- Virginica

## Classifier Model 3: K-Nearest Neighbors

The knn algorithm is known by many names such as lazy learning, instance-based learning, case-based learning, or local-weighted regression, this is because it does not split the data while training. In other words, it uses all the data while training.

To check the accuracy of the model changes with changing values of k, a loop was created to store the accuracy score of the model for each value of k. From the plot it can be noted that the values of k between 1 and 16 have an accuracy of 0.9666667 as it plummets as k continues to increase.

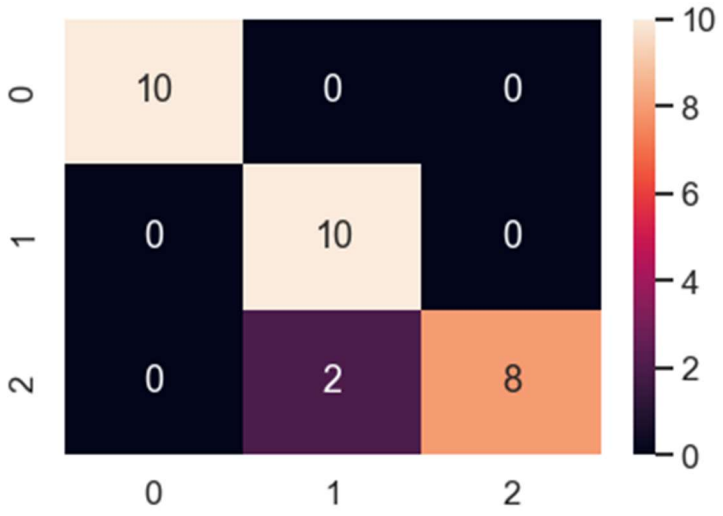



From the detailed Classification Report, it can be observed that the knn model performed very well on some classes and poorly on others compared to the models with an accuracy of 0.9333. This means that the model is predicting the correct class for the flower about 93% of time.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 10 |
| 1 | 0.83 | 1.00 | 0.91 | 10 |
| 2 | 1.00 | 0.80 | 0.89 | 10 |
| accuracy | | | 0.93 | 30 |
| macro avg | 0.94 | 0.93 | 0.93 | 30 |
| weighted avg | 0.94 | 0.93 | 0.93 | 30 |

**Confusion Matrix**

From the confusion matrix it can be observed that the model made perfect predictions and only twice the model confused class 2 with class 1.




0- Setosa
1- Versicolor
2- Virginica

## Insights and Key Insights

The Iris Dataset contains four features (length and width of sepals and petals) of 50 samples of three species of Iris (Iris setosa, Iris virginica, and Iris versicolor). From the Exploratory Data Analysis done, it can be said that the Petal length and width and also the Sepal length and width are very important since they have a major influence on the target variable.

Based on predictions, it was deduced that the model is predicting correctly because the Setosa is shortest and Virginica is the longest and Versicolor is between these two.

## Conclusion

From the evaluation and all the analyses done, it can be concluded that Logistic Regression model (One vs Rest Method) performed well with an accuracy 97% likewise Support Vector Classifier with K-Nearest Neighbors model gaining an accuracy of 93%. Therefore, I recommend using the simple Logistic Regression model for this task since it did very well without any adjustments and tuning.

## Recommendations

I recommend Ensemble Techniques such as VotingClassifier techniques since it often give the best results and it thrives in cases where some of the models work well on one part of the parameter space while others work better on other parts. Also, the use of hyperparameter tuning to help tune the models to improve its prediction.