IRIS FLOWERS CLASSIFICATION

TEAM MEMBERS

Lama Youssef

Sarah Ali

Kholoud Khalid

Joud Abdullah

TABLE OF CONTENTS

SENARIO

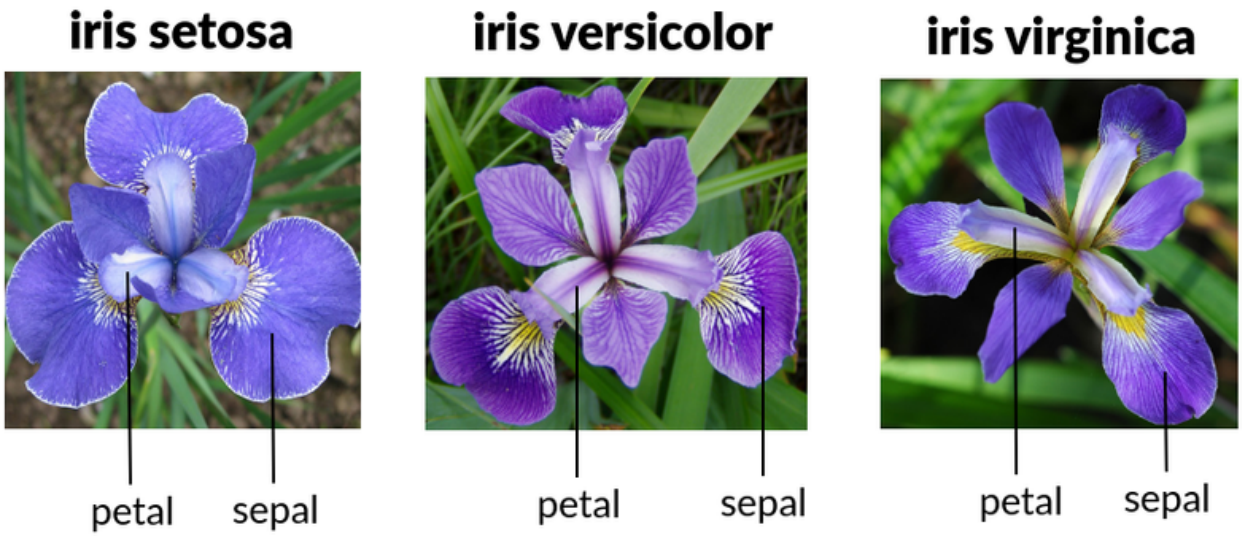
Develop a machine learning model to classify iris flowers based on their sepal and petal measurements.

MACHINE LEARNING GOAL

The goal of this project is to train a machine learning model to accurately classify iris flowers into one of three species: Iris setosa, Iris versicolor, or Iris virginica. The model will be trained on a dataset of iris flower measurements and then used to predict the species of new iris flowers.

SOURCES OF DATA

The Iris (or Iris) is a dataset that 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) in total 150 records. These measures were used to create a linear discriminant model to classify the species. The dataset is often used in data mining, classification and clustering examples and to test algorithms.



QUESTIONS/HYPOTHESES

- Can a machine learning model achieve an accuracy of at least 95% in classifying Iris flowers?
- Hypothesis: A support vector machine (SVM) model will achieve an accuracy of at least 95% in classifying Iris flowers.

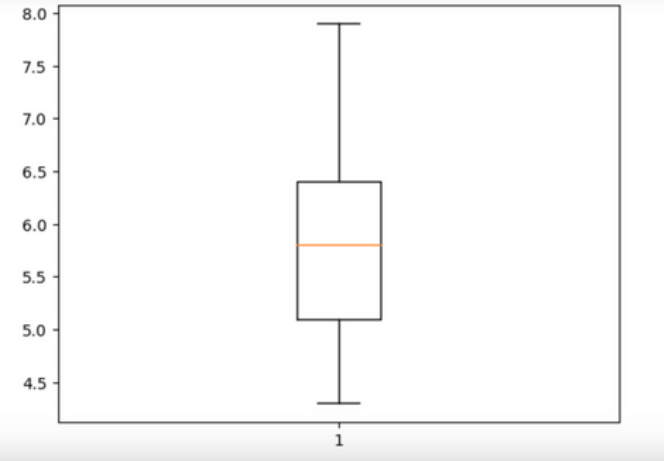
TOOLS/LIBRARIES

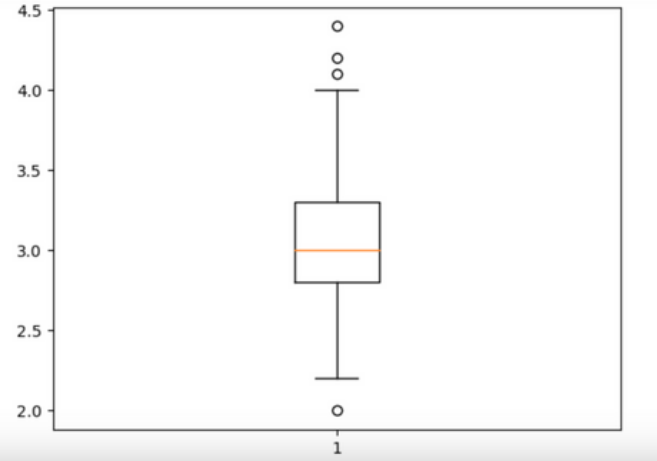
NumPy: Numerical operations on arrays, vectors, and matrices
Matplotlib.pyplot: Data visualization (plots, charts)
Seaborn: High-level statistical data visualization
Pandas: Data analysis and manipulation (DataFrames, Series)
sklearn.linear_model: Linear regression models (LogisticRegression)
sklearn.model_selection: Training and testing data splits
sklearn.neighbors: K-nearest neighbors classifiers (KNeighborsClassifier)
sklearn.svm: Support vector machine (SVM) algorithms
sklearn.metrics: Model performance evaluation (accuracy, precision, recall, F1-score)

In [12]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
get_ipython().run_line_magic('matplotlib', 'inline')
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
from sklearn import metrics
```

DATA ANALYSIS

Here we draw a boxplot to visually inspect the presence of outliers in the dataset.





	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

DATA PREPROCESSING

In the initial exploration of the dataset, we conducted a thorough check for missing values. Fortunately, the dataset is clean, and there are no instances of missing values in any of the features or the target variable. As we can see in this code:

Checking for null values

```
]: #Checking for the null values
   data.isnull().sum()
```

```
]: sepal_length    0
   sepal_width     0
   petal_length    0
   petal_width     0
   species         0
   dtype: int64
```

This absence of missing data simplifies the preprocessing steps, allowing us to proceed with the analysis without the need for imputation or handling missing values.

PREPARE DATA FOR MODELING

To train the model and next test the model we have to split the entire dataset into train and test sets using the train_test_split function. The dataset was divided such that 70% of the instances were allocated to the training set and 30% to the testing set.

Train test split

```
train, test = train_test_split(data, test_size = 0.3)
print(train.shape)
print(test.shape)
```

```
(105, 5)
(45, 5)
```

 # Prepare data for modeling

```
train_X = train[['sepal_length', 'sepal_width', 'petal_length',
                 'petal_width']]
train_y = train.species

test_X = test[['sepal_length', 'sepal_width', 'petal_length',
               'petal_width']]
test_y = test.species
```

To assess the performance of machine learning models, the dataset was split into training and testing sets. The features (sepal_length, sepal_width, petal_length, and petal_width) were extracted for both the training and testing sets, and the target labels (species) were isolated accordingly.

THE MODEL IMPLEMENTATION

Using some of the commonly used algorithms, we will be training our model to check how accurate every algorithm is. We will be implementing these algorithms to compare:

1] Logistic Regression

2] K – Nearest Neighbour (KNN)

3] Support Vector Machine (SVM)

Logistic Regression

We can start with the first algorithm Logistic Regression, a commonly used algorithm for binary and multi-class classification tasks, was employed in this project for its simplicity and interpretability. The scikit-learn library was utilized to implement the Logistic Regression model.

```
model1 = LogisticRegression()
model1.fit(train_X, train_y)
prediction = model1.predict(test_X)
prediction
```

The logistic regression model demonstrated a commendable accuracy of [0.97], indicating its capability to effectively classify Iris flower species based on the provided features.

```
print('Accuracy:',metrics.accuracy_score(prediction,test_y))
```

```
Accuracy: 0.9777777777777777
```

K–Nearest Neighbour (KNN)

The K-Nearest Neighbors algorithm is a non-parametric and instance-based method for classification. It classifies instances based on the majority class of their k-nearest neighbors. The choice of 'k' was optimized through cross-validation.

Here is our second model code:

```
from sklearn.neighbors import KNeighborsClassifier
model2 = KNeighborsClassifier(n_neighbors=5)
model2.fit(train_X,train_y)
y_pred2 = model2.predict(test_X)
```

The KNN model demonstrated a commendable accuracy of [0.95], indicating its capability to effectively classify Iris flower species based on the provided features.

```
from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(test_y,y_pred2)
      )
```

```
Accuracy Score: 0.9555555555555556
```

Support Vector Machine (SVM)

Support Vector Machines are powerful classifiers known for their effectiveness in handling both linear and non-linear decision boundaries. In our implementation, we employed the scikit-learn library to build an SVM model. Here is the code:

```
from sklearn.svm import SVC
model1 = SVC()
model1.fit(train_X,train_y)

pred_y = model1.predict(test_X)

from sklearn.metrics import accuracy_score
```

The SVM model demonstrated a commendable accuracy of [0.97], indicating its capability to effectively classify Iris flower species based on the provided features.

```
print("accuracy=",accuracy_score(test_y,pred_y))
```

```
accuracy= 0.9777777777777777
```

RESULTS

In this section, we present a comparison of the performance of three distinct classification models: Support Vector Machine (SVM), k-Nearest Neighbors (KNN), and Logistic Regression. Each model underwent training on the provided dataset, and their accuracies were evaluated on the test dataset.

```
:
results = pd.DataFrame({
    'Model': ['Logistic Regression','Support Vector Machines','KNN'],
    'Score': [0.9777,0.9777,0.9555]})

result_df = results.sort_values(by='Score', ascending=False) $
result_df = result_df.set_index('Score')
result_df.head(9)
```

The resulting summary table provides a clear comparison of the models based on their accuracy scores:

Score	Model
0.9777	Logistic Regression
0.9777	Support Vector Machines
0.9555	KNN

The high accuracies across all models underscore their suitability for the Iris flower classification task. Logistic Regression and SVM both demonstrated strong performances with an accuracy of 97.0%, slightly outperforming KNN.

CONCLUSION

In conclusion, this study embarked on the task of Iris flower classification using three distinct machine learning models: Logistic Regression, Support Vector Machines (SVM), and k-Nearest Neighbors (KNN). The primary objective was to explore the effectiveness of these models in accurately categorizing Iris flower species based on their sepal and petal dimensions. The results obtained from the evaluation of each model on the test dataset yielded valuable insights into their respective strengths and limitations. Logistic Regression and Support Vector Machines emerged as the top-performing models, both achieving an impressive accuracy of 97.77%. These models demonstrated robust capabilities in handling the complexities of the Iris dataset, showcasing their efficacy in linear and non-linear decision boundary scenarios. While k-Nearest Neighbors (KNN) delivered a commendable accuracy of 95.55%, it was slightly outperformed by the other models.