A Report

on

IRIS FLOWER CLASSIFICATION

carried out as part of the course CSE CS3203 Submitted by

MANAS KUMAR JHA

209301209

VI-CSE

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering

MANIPAL UNIVERSITY JAIPUR

Department of Computer Science & Engineering,
School of Computing and IT,
Manipal University Jaipur,

March 2023

Introduction

The iris flower classification problem is a well-known and frequently studied problem in machine learning. The problem involves predicting the species of iris flowers based on various measurements of their flowers, including sepal length, sepal width, petal length, and petal width. The classification problem is a multiclass problem with three classes: setosa, versicolor, and Virginia.

The iris flower classification problem is an important problem in the field of botany and plant science, as it provides a way to classify and identify different species of iris based on their physical characteristics. This information can be used for a variety of purposes, such as identifying rare or endangered species, monitoring changes in plant populations over time, and understanding the evolutionary relationships between different species of iris.

From a machine learning perspective, the iris flower classification problem is a classic example of a supervised learning problem. The problem involves using a labelled dataset of iris flower measurements to train a machine learning model to predict the species of new, unlabelled iris flowers. The problem is relatively simple, with a small number of input features and a limited number of output classes, making it an ideal problem for beginners in machine learning.

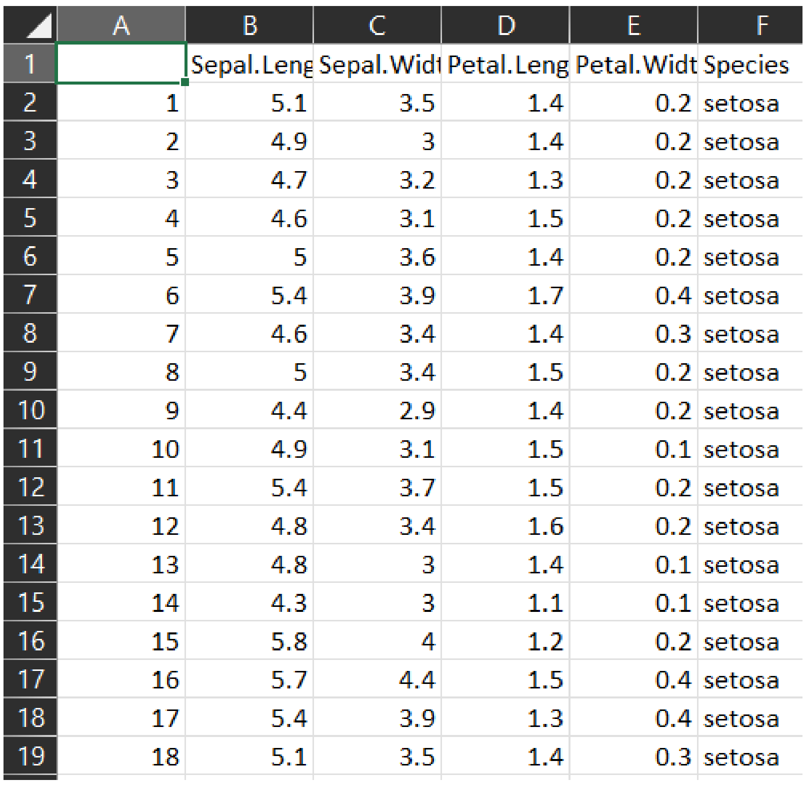
In this report, we will explore various machine-learning techniques for solving the iris flower classification problem. We will start by exploring the dataset and performing some basic exploratory data analysis to gain a better understanding of the data. We will then implement various machine learning algorithms, including logistic regression, decision trees, and support vector machines, and evaluate their performance using various performance metrics such as accuracy, precision, and recall. Finally, we will discuss the strengths and weaknesses of each algorithm and make recommendations for future research in this area.

Dataset description

The dataset contains measurements of four features of three different species of Iris flowers: Iris setosa, Iris versicolor, and Iris virginica. The four features measured for each flower are sepal length, sepal width, petal length, and petal width, all measured in centimetres.

The dataset contains a total of 150 observations, with 50 observations for each species of Iris. The dataset is balanced, with an equal number of observations for each class. The dataset is often used as a benchmark dataset in machine learning, as it is a relatively simple classification problem with a small number of features and classes, making it an ideal dataset for beginners in machine learning.

The Iris flower dataset is widely used for testing and comparing different classification algorithms, including logistic regression, decision trees, support vector machines, and neural networks. The dataset has also been used for other purposes, such as feature selection and dimensionality reduction. The dataset is freely available and can be downloaded from various sources, including the UCI Machine Learning Repository and the scikit-learn Python library.



	A	B	C	D	E	F
1		Sepal.Leng	Sepal.Widt	Petal.Leng	Petal.Widt	Species
2	1	5.1	3.5	1.4	0.2	setosa
3	2	4.9	3	1.4	0.2	setosa
4	3	4.7	3.2	1.3	0.2	setosa
5	4	4.6	3.1	1.5	0.2	setosa
6	5	5	3.6	1.4	0.2	setosa
7	6	5.4	3.9	1.7	0.4	setosa
8	7	4.6	3.4	1.4	0.3	setosa
9	8	5	3.4	1.5	0.2	setosa
10	9	4.4	2.9	1.4	0.2	setosa
11	10	4.9	3.1	1.5	0.1	setosa
12	11	5.4	3.7	1.5	0.2	setosa
13	12	4.8	3.4	1.6	0.2	setosa
14	13	4.8	3	1.4	0.1	setosa
15	14	4.3	3	1.1	0.1	setosa
16	15	5.8	4	1.2	0.2	setosa
17	16	5.7	4.4	1.5	0.4	setosa
18	17	5.4	3.9	1.3	0.4	setosa
19	18	5.1	3.5	1.4	0.3	setosa

Algorithms

- **Decision tree classifier:**

The decision tree classifier is a supervised learning algorithm that builds a tree-like model to make predictions based on a set of features. The algorithm works by recursively splitting the data into subsets based on the value of a single feature until a stopping criterion is met, such as reaching a maximum depth or a minimum number of samples in a leaf node. The decision tree algorithm uses a measure of impurity, such as Gini impurity or entropy, to determine the best feature to split the data at each node.

- **K nearest neighbor classifier:**

The k-nearest neighbor classifier is a simple supervised learning algorithm that works by finding the k-nearest neighbors in the training set to a given query point and classifying the query point based on the majority class of its k-nearest neighbors. The algorithm uses a distance metric, such as Euclidean or Manhattan distance, to measure the similarity between the query point and the training data.

- **Support vector machine (SVM):**

SVM is a supervised learning algorithm that can be used for classification or regression tasks. SVM finds the optimal hyperplane that separates two classes by maximizing the margin, which is the distance between the hyperplane and the closest points from each class. SVM can also handle non-linearly separable data by using kernel functions to map the data to a higher-dimensional space where the data can be separated by a linear hyperplane.

- **Logistic Regression:**

Logistic regression is a classification algorithm that predicts the probability of a binary or multi-class outcome based on a set of features. The algorithm models the relationship between the features and the probability of a given outcome using a logistic function. The logistic regression algorithm uses maximum likelihood estimation to estimate the model parameters.

- **Metrics:**

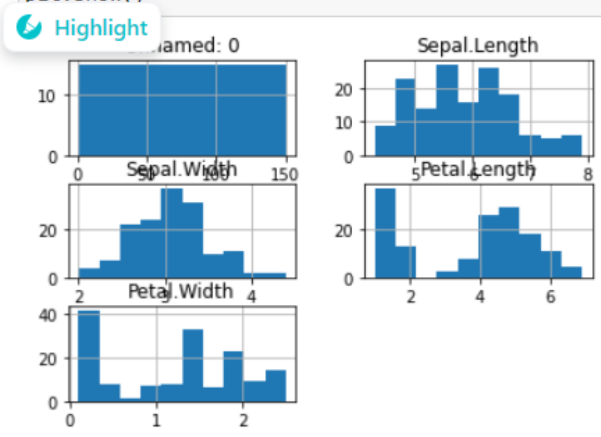
Metrics are used to evaluate the performance of a machine learning algorithm. Common classification metrics include accuracy, precision, recall, F1 score, and ROC-AUC score. These metrics can be used to compare the performance of different algorithms or to optimize the hyperparameters of a given algorithm.

- **Train_test_split:**

Train_test_split is a function in Python's sci-kit-learn library that splits a dataset into training and testing sets. The function randomly selects a subset of the data for training and the remaining data for testing. The purpose of the function is to evaluate the performance of a machine learning algorithm on unseen data and to prevent overfitting.

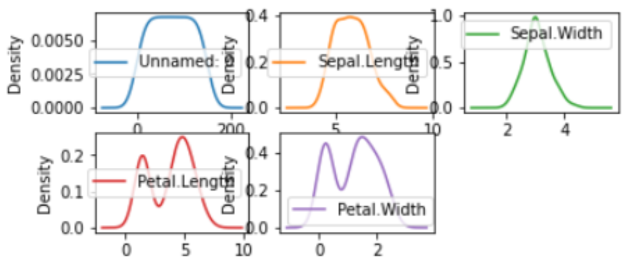
Data Visualization

```
In [17]: iris.hist()
         plt.show()
```



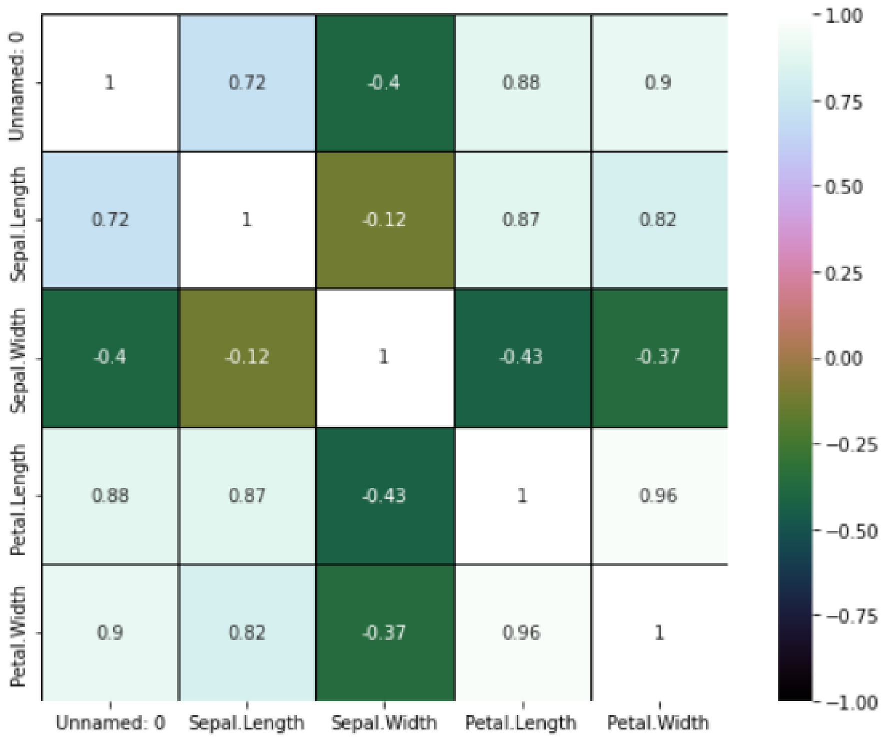
```
In [18]: iris.plot(kind ='density',subplots = True, layout =(3,3),sharex = False)

Out[18]: array([[<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
                 <Axes: ylabel='Density'>],
                [<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
                 <Axes: ylabel='Density'>],
                [<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
                 <Axes: ylabel='Density'>]], dtype=object)
```



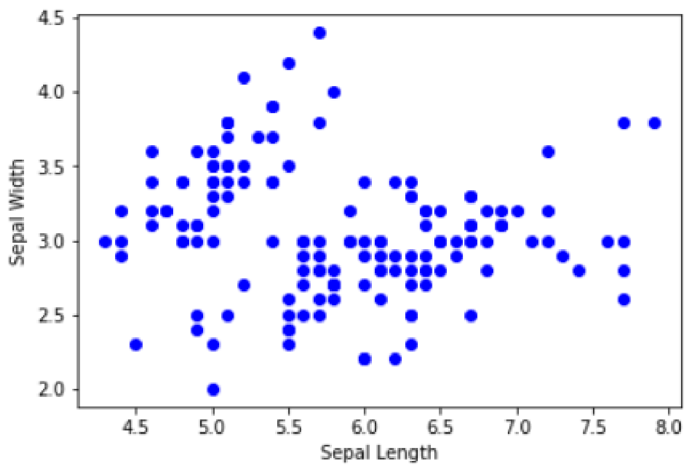
Result

Heat Map



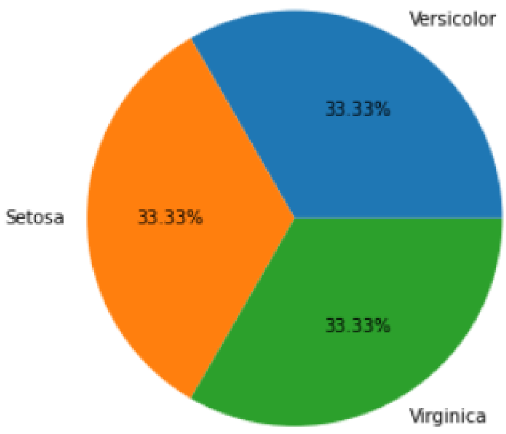
Scatter Plot

```
In [25]: plt.xlabel("Sepal Length")
         plt.ylabel("Sepal Width")
         plt.scatter(X,Y,color='b')
         plt.show()
```

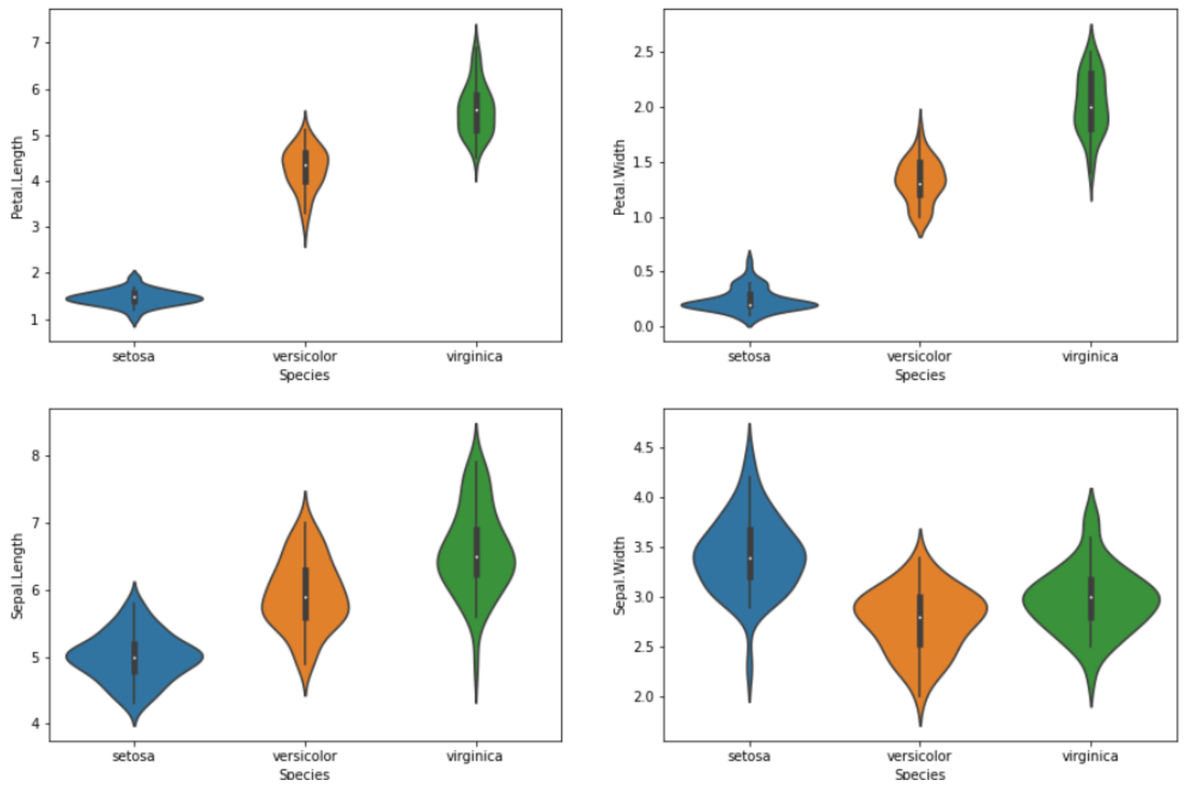


Pie Chart

```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.axis('equal')
l = ['Versicolor', 'Setosa', 'Virginica']
s = [50,50,50]
ax.pie(s, labels = l,autopct='%1.2f%%')
plt.show()
```



```
<Axes: xlabel='Species', ylabel='Sepal.Width'>
```



Violin Plot

Confusion Matrix

In [34]:
```
#Confusion matrix
from sklearn.metrics import confusion_matrix,classification_report
confusion_mat = confusion_matrix(test_y,prediction)
print("Confusion matrix: \n",confusion_mat)
print(classification_report(test_y,prediction))
```

```
Confusion matrix:
 [[16  0  0]
 [ 0 14  0]
 [ 0  0  8]]
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        16
  versicolor       1.00      1.00      1.00        14
   virginica       1.00      1.00      1.00         8

    accuracy                           1.00        38
   macro avg       1.00      1.00      1.00        38
weighted avg       1.00      1.00      1.00        38
```

Accuracy Comparison

In [39]:
```
results = pd.DataFrame({
    'Model': ['Logistic Regression','Support Vector Machines', 'Naive Bayes','KNN' ,'Decision Tree'],
    'Score': [0.947,0.947,0.947,0.947,0.921]})

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

Out[39]:

	Model
Score	
0.947	Logistic Regression
0.947	Support Vector Machines
0.947	Naive Bayes
0.947	KNN
0.921	Decision Tree