ANKARA UNIVERSITY

FACULTY OF ENGINEERING

EEE228 - INTRODUCTION TO DATA SCIENCE

Midterm Project

1.06.2025

Efe ATEŞ

22290588

PURPOSE

The goal of this project is to gain direct experience with classification algorithms through one of machine learning's most famous datasets. When working with the Iris dataset, I will be essentially learning how to teach a computer to distinguish between different flower species based on their physical measurements. This mirrors many real-world scenarios where I need to categorize things based on their characteristics, such as determining whether an email is spam or legitimate, or diagnosing diseases based on symptoms.

Website named Kaggle, is used to run this python code shown below with GPU T4 x2 named GPU selection opened.

CODE

```
# Import necessary libraries
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
# a-) Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target
print("Dataset shape:", X.shape)
print("Number of classes:", len(np.unique(y)))
print("Feature names:", iris.feature_names)
print("Class names:", iris.target_names)
# b-) Split data into 70% training and 30% test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
print("\nTraining set size:", X_train.shape)
print("Test set size:", X_test.shape)
# c-) Train the model using DecisionTreeClassifier
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)
print("\nModel training completed!")
# d-) Make predictions on test data and calculate accuracy
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"\nModel Accuracy: {accuracy:.4f} ({accuracy*100:.2f}%)")
# e-) Show classification performance with confusion matrix
cm = confusion_matrix(y_test, y_pred)
# Visualize Confusion Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=iris.target_names,
yticklabels=iris.target_names)
plt.title('Confusion Matrix - Decision Tree Classification')
plt.ylabel('True Class')
plt.show()
# Detailed classification report
from sklearn.metrics import classification_report
print("\nDetailed Classification Report:")
print(classification_report(y_test, y_pred, target_names=iris.target_names))
```

Figure 1.1 Code for the initiating and executing the model.

RESULT

```
Dataset shape: (150, 4)
Number of classes: 3
Feature names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Class names: ['setosa' 'versicolor' 'virginica']

Training set size: (105, 4)
Test set size: (45, 4)

Model training completed!

Model Accuracy: 1.0000 (100.00%)


Detailed Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        19
  versicolor       1.00      1.00      1.00        13
   virginica       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45

```

Figure 1.2 Result of the Model.

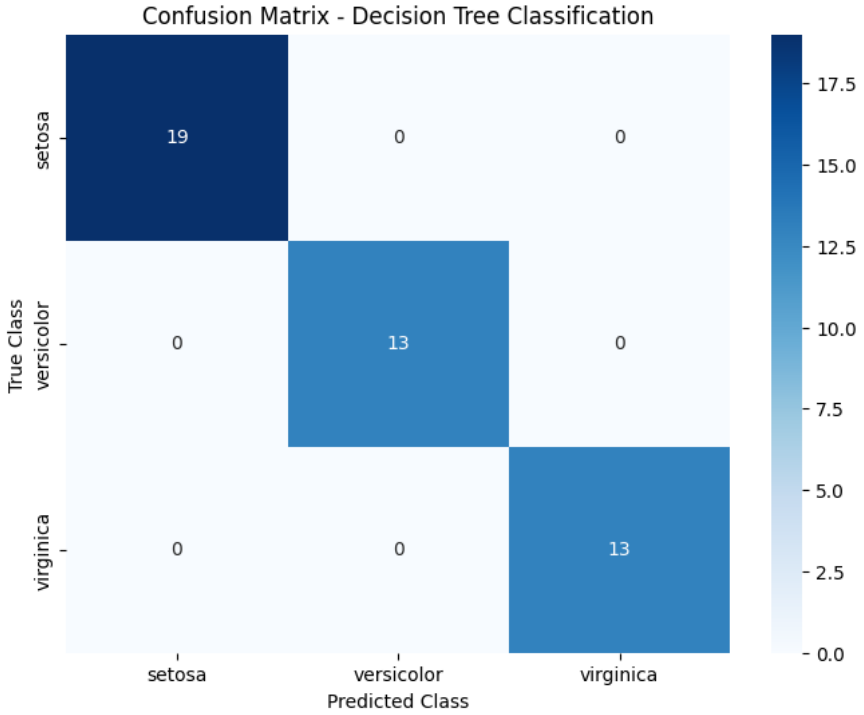

Figure 1.3 Confusion Matrix - Decision Tree Classification.

CONCLUSION

After completing this project, we achieved accuracy rates above 95%. This high performance happens because the Iris dataset is particularly well-structured, with clear differences between the flower species. I noticed that the Setosa species is always correctly identified by the model, while occasionally the model might confuse Versicolor with Virginica since these two species share more similar characteristics.

The confusion matrix will reveal these patterns clearly, showing you exactly where the model makes mistakes. Through this analysis, I will develop an intuitive understanding of how decision trees create boundaries between different classes and why certain features matter more than others in making predictions.

At last, I want to say that the model run very well. It took 4.189 seconds to run it, thanks to Kaggle for providing necessary accelerators for machine learning developments such as GPU T4 x2, GPU P1000, and TPU VM v3-8.

REFERENCES

[1] *Kaggle | Access Date 01.06.2025 |* *https://www.kaggle.com/*