##### TASK 1:IRIS FLOWER CLASSIFICATION

# Import necessary libraries

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

import joblib

# Load the dataset

iris = load\_iris()

df = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

df['species'] = iris.target

df['species'] = df['species'].map({0: 'setosa', 1: 'versicolor', 2: 'virginica'})

# Display first few rows of the dataset

print(df.head())

# Exploratory Data Analysis (EDA)

sns.pairplot(df, hue='species')

plt.show()

# Check for class distribution

print(df['species'].value\_counts())

# Describe the dataset

print(df.describe())

# Data Preprocessing

X = df.drop(columns=['species'])

y = df['species']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize and train models

models = {

'Decision Tree': DecisionTreeClassifier(),

'K-Nearest Neighbors': KNeighborsClassifier(),

'Support Vector Machine': SVC(),

'Random Forest': RandomForestClassifier()

}

# Evaluate each model

for name, model in models.items():

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"{name} Accuracy: {accuracy:.2f}")

if name == 'Decision Tree': # Displaying detailed report for one model as an example

print(classification\_report(y\_test, y\_pred))

cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=iris.target\_names, yticklabels=iris.target\_names)

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()

# Example of hyperparameter tuning for RandomForestClassifier

from sklearn.model\_selection import GridSearchCV

param\_grid = {

'n\_estimators': [50, 100, 200],

'max\_depth': [None, 10, 20, 30]

}

grid\_search = GridSearchCV(RandomForestClassifier(), param\_grid, cv=5, scoring='accuracy')

grid\_search.fit(X\_train, y\_train)

print(f"Best parameters: {grid\_search.best\_params\_}")

print(f"Best score: {grid\_search.best\_score\_:.2f}")

# Save the best model

best\_model = grid\_search.best\_estimator\_

joblib.dump(best\_model, 'iris\_classifier.pkl')

# Load the model and make predictions (optional)

loaded\_model = joblib.load('iris\_classifier.pkl')

y\_loaded\_pred = loaded\_model.predict(X\_test)

print(f"Loaded model accuracy: {accuracy\_score(y\_test, y\_loaded\_pred):.2f}")

#### Project Overview

The goal of this project is to classify iris flowers into three species: Setosa, Versicolor, and Virginica based on their measurements (sepal length, sepal width, petal length, and petal width). The project uses machine learning models to perform this classification task. The dataset used is the well-known Iris dataset, which is available in the scikit-learn library.

#### Steps in the Implementation

1. **Importing Libraries**
2.  **pandas**: For data manipulation and analysis.
3.  **seaborn and matplotlib**: For data visualization.
4.  **scikit-learn**: For machine learning models and utilities.
5.  **joblib**: For saving and loading the trained model.

**2.Loading the Dataset**

**Explanation**:

* The dataset is loaded using load\_iris() from scikit-learn.
* A pandas DataFrame is created for ease of data manipulation.
* Species are mapped from numerical values to their corresponding names for better interpretability.

3.Exploratory Data Analysis (EDA)

**Explanation**:

* The first few rows of the dataset are displayed to understand its structure.
* A pairplot is created to visualize the relationships between features.
* The distribution of species in the dataset is printed.
* Descriptive statistics are computed to understand the data distribution.

**4.** Data Preprocessing

 **Explanation**:

* Features (X) and target variable (y) are separated.
* The dataset is split into training and testing sets using an 80-20 split ratio to evaluate the models.

**5Model Training and Evaluation**

**Explanation**:

* Four models are initialized: Decision Tree, K-Nearest Neighbors, Support Vector Machine, and Random Forest.
* Each model is trained on the training data and evaluated on the test data.
* Accuracy of each model is printed. For the Decision Tree model, a detailed classification report and confusion matrix are displayed.

6.**Hyperparameter Tuning for RandomForestClassifier**

**Explanation**:

* Hyperparameter tuning is performed using GridSearchCV to find the best parameters for the RandomForestClassifier.
* A grid of hyperparameters is defined, and the model is trained and evaluated using cross-validation.

**7.**

**Saving the Best Model**

**Explanation**:

* The best model found by GridSearchCV is saved to a file using joblib for future use.

**8.** Loading and Testing the Saved Model (Optional)

1. **Explanation**:
   * The saved model is loaded from the file.
   * The loaded model is used to make predictions on the test set, and its accuracy is printed.

#### Design Choices

* **Choice of Models**: A variety of models (Decision Tree, K-Nearest Neighbors, Support Vector Machine, Random Forest) were chosen to compare their performance on the dataset.
* **Grid Search for Hyperparameter Tuning**: Grid search was chosen for its thoroughness in exploring the hyperparameter space, ensuring the best model is found.
* **Visualization**: Pairplot and confusion matrix heatmaps were used for their effectiveness in visualizing data relationships and model performance.

#### Challenges Faced

* **Data Visualization**: Ensuring that the visualizations effectively communicated the necessary information required careful selection of plots and their customization.
* **Model Selection and Evaluation**: Balancing between different models and ensuring fair evaluation required a structured approach to splitting the data and assessing performance metrics.
* **Hyperparameter Tuning**: Hyperparameter tuning can be computationally expensive, but it was necessary to find the optimal model settings.

This implementation provides a robust and comprehensive approach to classifying iris flowers using machine learning, ensuring that the best model is chosen through systematic evaluation and tuning.