Iris Flower Classification

Overview

Machine learning plays a crucial role in various domains, from healthcare to finance and beyond. In this article, we'll delve into a classic example of a machine-learning application: the Iris Flower Classification. We will explore the dataset, employ different machine-learning models, and discuss the insights gained from this fascinating project. This article contains code and resources for the Iris Flower Classification project. The objective of this project is to classify iris flowers into distinct species based on their sepal and petal measurements. The dataset used for training and evaluation is the well-known Iris dataset, consisting of samples from three iris species: Setosa, Versicolor, and Virginia.

Data Exploration

Before diving into machine learning models, let's explore the dataset. We load the data into a pandas data frame, perform basic statistical analysis, and visualize the distribution of features. The Seaborn library helps us create informative visualizations, such as count plots and pair plots, which offer insights into the relationships between different features. The Iris dataset includes features such as sepal length, sepal width, petal length, and petal width. The data is split into training and testing sets for model evaluation.

Importing Necessary Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import datasets
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
```

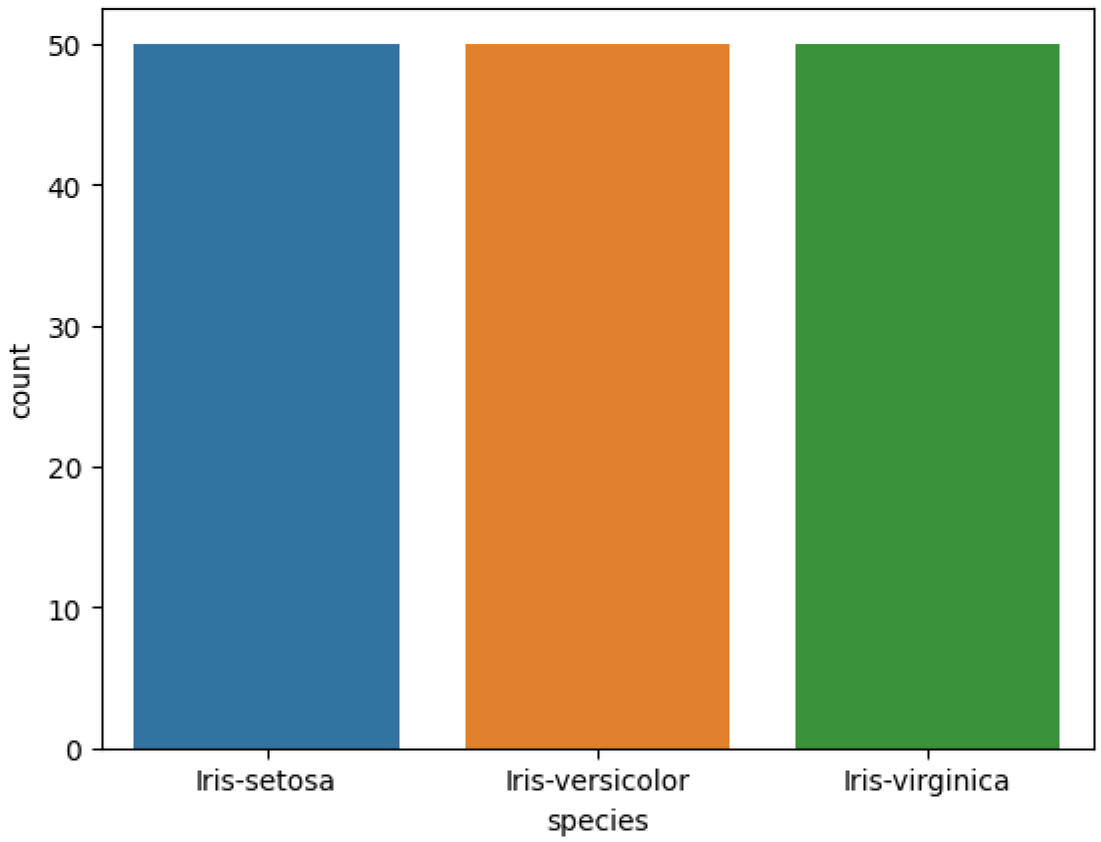
Load the dataset

```
# You can also directly load the dataset.
# iris = datasets.load_iris()
# X = iris.data
# y = iris.target

df = pd.read_csv("/content/IRIS.csv")
X = df[['sepal_length','sepal_width','petal_length','petal_width']]
y = df['species']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Standardize the features by removing the mean and scaling to unit
variance
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Exploratory Data Analysis (EDA)

- The dataset is explored using descriptive statistics and visualizations to understand the distribution and characteristics of the features.
- NULL values are checked to ensure data integrity.

```
• df.isnull().sum()

  # This will give you the count of each species
  y.value_counts()
  sns.countplot(x=df['species'],hue=df['species'])
```

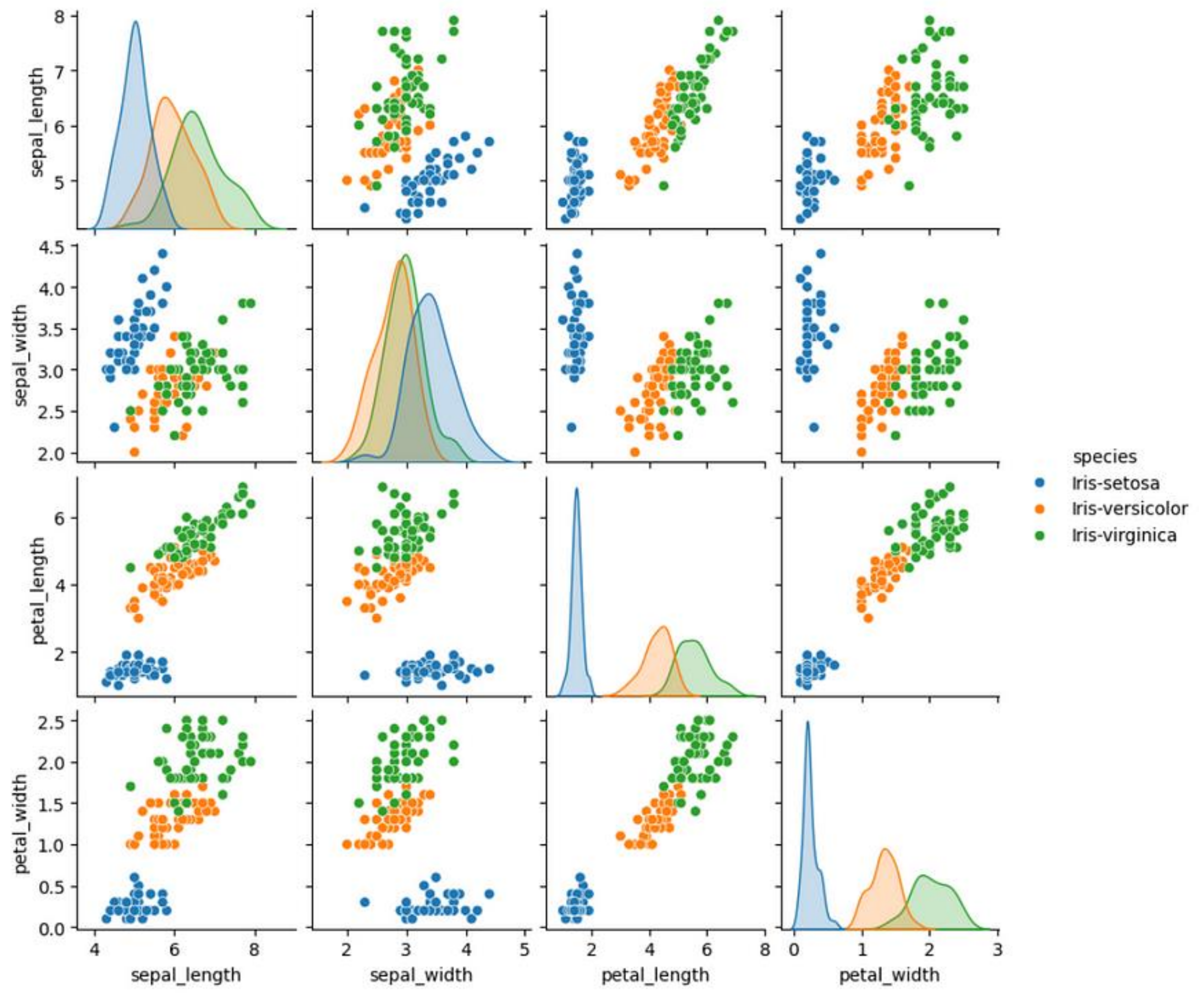


Feature Visualization

- A pair plot is generated to visualize the relationships between features for each iris species.

```
# Visualize the whole dataset
plt.figure(figsize=(8, 6))
```

```
sns.pairplot(df, hue='species', height=2)
plt.show()
```



Visualize the whole dataset

Machine Learning Models:

We employ various machine learning models to classify the iris flowers based on their features. The models used include:

1. Support Vector Machine (SVM): Achieved an accuracy of 69.67%.
2. k-Nearest Neighbors (k-NN): Achieved an accuracy of 100%.
3. Decision Tree Classifier: Achieved an accuracy of 100%.

4. Random Forest Classifier: After hyperparameter tuning, achieved an accuracy of 100%.

These models are trained and evaluated on a testing set to measure their accuracy. The scikit-learn library simplifies the implementation of these models, making it accessible even for those new to machine learning.

Training and Testing

```python
# Support Vector Machine (SVM)
svm_classifier = SVC(kernel='linear', C=1.0, random_state=42)
svm_classifier.fit(X_train, y_train)
y_pred_svm = svm_classifier.predict(X_test)
accuracy_svm = accuracy_score(y_test, y_pred_svm)

# k-Nearest Neighbors (k-NN)
knn_classifier = KNeighborsClassifier(n_neighbors=3)
knn_classifier.fit(X_train, y_train)
y_pred_knn = knn_classifier.predict(X_test)
accuracy_knn = accuracy_score(y_test, y_pred_knn)

# Decision Tree
dt_classifier = DecisionTreeClassifier(random_state=42)
dt_classifier.fit(X_train, y_train)
y_pred_dt = dt_classifier.predict(X_test)
accuracy_dt = accuracy_score(y_test, y_pred_dt)

# Print accuracies
print(f"SVM Accuracy: {accuracy_svm * 100:.2f}%")
print(f"k-NN Accuracy: {accuracy_knn * 100:.2f}%")
print(f"Decision Tree Accuracy: {accuracy_dt * 100:.2f}%")
```

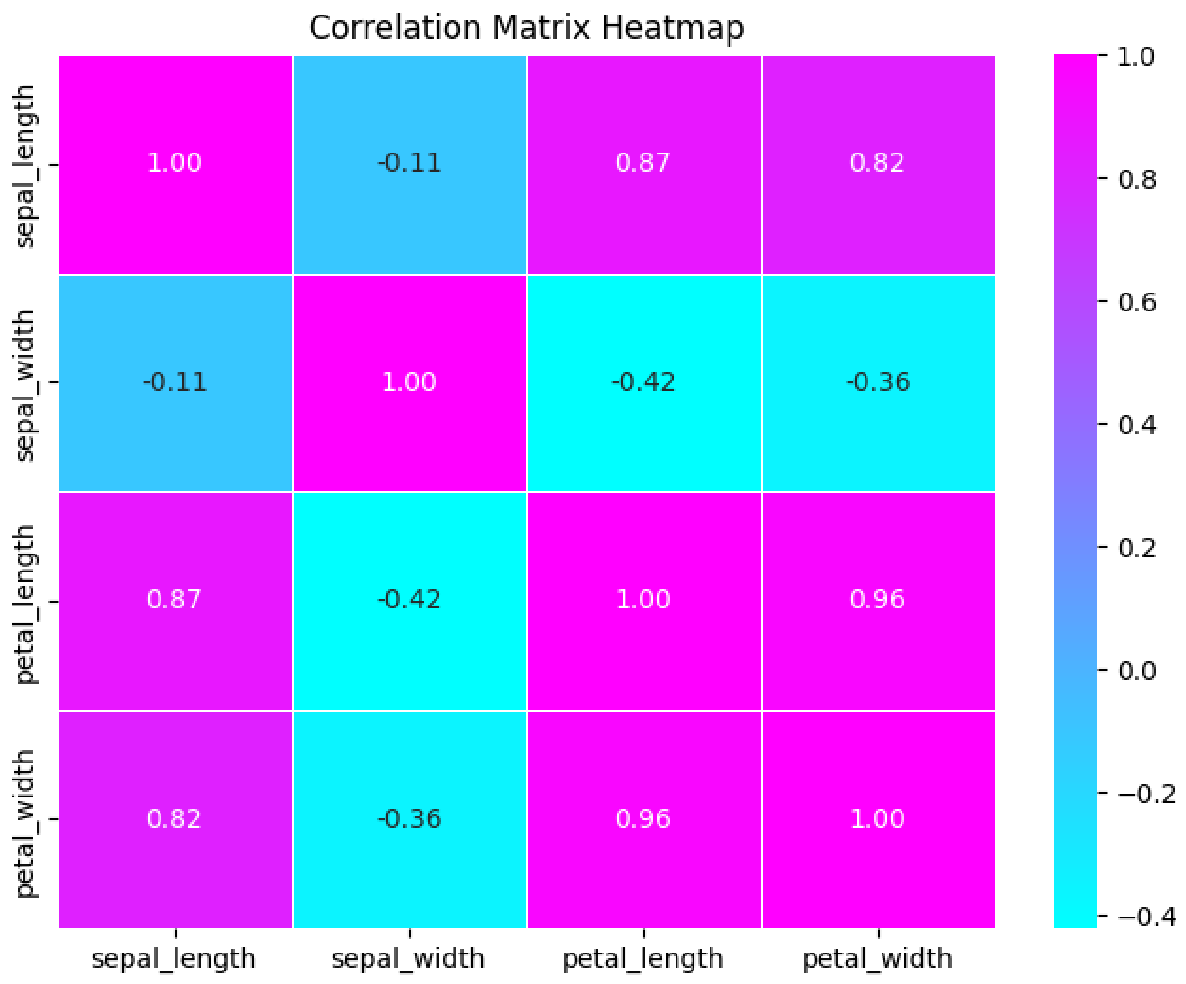
1. SVM Accuracy: 96.67%

2. k-NN Accuracy: 100.00%

3. Decision Tree Accuracy: 100.00%

Correlation Matrix

A heatmap displays the correlation matrix of the feature set.



Heatmap for visualizing correlation

Hyperparameter Tuning

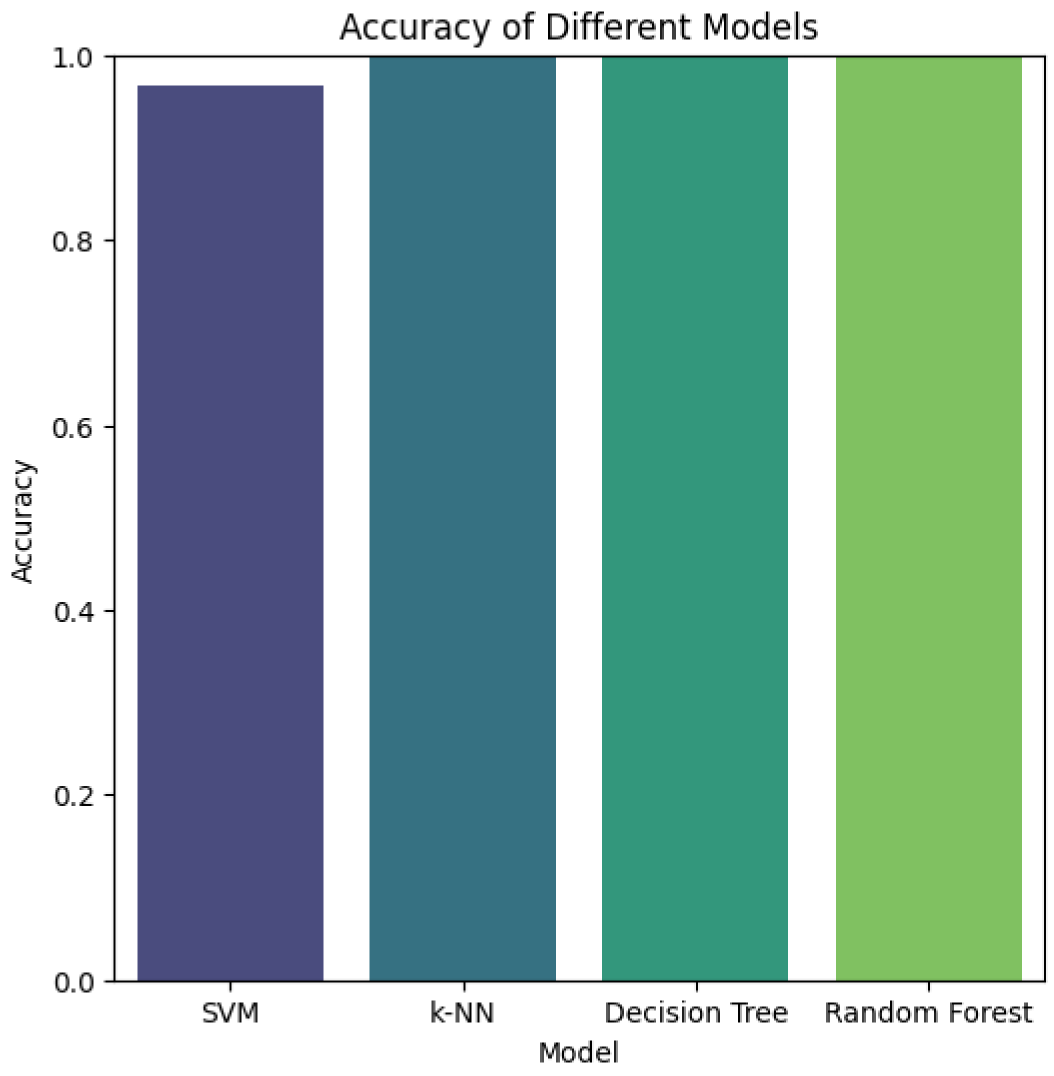
The Random Forest model undergoes hyperparameter tuning using grid search with cross-validation. The best parameters are identified, and the model is trained with those parameters.

The best-performing model is the Random Forest Classifier with an accuracy of 100%.

Accuracy of different Models

We can visualize the accuracy of models used using this code.

```
# Create a DataFrame with model names and accuracy values
results_df = pd.DataFrame({
    'Model': ['SVM', 'k-NN', 'Decision Tree', 'Random Forest'],
    'Accuracy': [accuracy_svm, accuracy_knn, accuracy_dt, accuracy_rf]
})

# Plot the bar graph
plt.figure(figsize=(6, 6))
sns.barplot(x='Model', y='Accuracy', data=results_df, palette='viridis')
plt.title('Accuracy of Different Models')
plt.ylim(0, 1)  # Set y-axis limit to the range of 0 to 1 (accuracy range)
plt.show()
```



Accuracy of different models

Conclusion

In conclusion, the Iris Flower Classification project provides a hands-on exploration of machine learning techniques. Through data exploration, model training, and visualization, we gain a deeper understanding of the dataset and the capabilities of various classification models.

This project serves as an excellent starting point for those new to machine learning, offering practical insights into data preprocessing, model training, and performance evaluation. Feel free to explore the code, experiment with different models, and contribute to the project's growth.

How to Use

How to Use the Code:

The code is available in the associated GitHub repository https://github.com/Modassirnazar/OIBSIP_1. You can clone the repository, install the necessary dependencies, and run the provided script to experience the project firsthand.

By engaging with this project, you'll not only enhance your understanding of machine learning but also contribute to a valuable resource for others exploring this fascinating field.

Happy coding!

Modassir

If you find the article helpful, please like and comment on this.