IRIS FLOWER CLASSIFICATION

Objective:

The objective is to elaborates on the main goal of the project, which is to develop a machine learning model capable of accurately classifying iris flowers into their respective species based on certain features. This page emphasizes the importance of classification accuracy and the practical applications of such a model.

Data Source:

It explains that the Iris dataset will be used for this project, mentioning its availability within the scikit-learn library. A brief overview of the dataset's features and target classes is provided, setting the stage for further exploration.

Import Library:

It lists the libraries required for data manipulation, visualization, model building, and evaluation, such as NumPy, pandas, seaborn, matplotlib, and scikit-learn.

Import Data:

The code imports the Iris dataset using scikit - learn' s **load _ iris** function. It loads the feature matrix (X) and the target vector (y) into memory, making them ready for further analysis and model building.

Describe Data:

Visualization plays a crucial role in understanding the data. This section showcases pair plots using seaborn' s “**pairplot”** function to visualize relationships between different features and their distributions across different target classes.

Data Pre-processing:

Data pre-processing is essential for preparing the data for modelling . This section covers standardization using scikit - learn' s “**StandardScaler”** to scale the features to have mean 0 and variance 1, ensuring that all features contribute equally to the model.

Define Target Variable (y) and Feature Variables (X):

Here, the data is split into feature variables (X) and the target variable (y). The **train_test_split** function from scikit-learn is used to split the data into training and testing sets, with 80% for training and 20% for testing.

Train Test Split:

This section explains the importance of splitting the data into training and testing sets. It highlights the need for evaluating the model's performance on unseen data and preventing overfitting.

Modeling:

Modeling involves selecting and training a machine learning algorithm on the training data. This section introduces the Random Forest classifier and fits it to the training data using scikit-learn's **RandomForestClassifier**.

Model Evaluation:

After training the model, it's crucial to evaluate its performance. This section covers model evaluation metrics such as accuracy, which measures the proportion of correctly classified instances, and provides insights into the model's effectiveness.

Prediction:

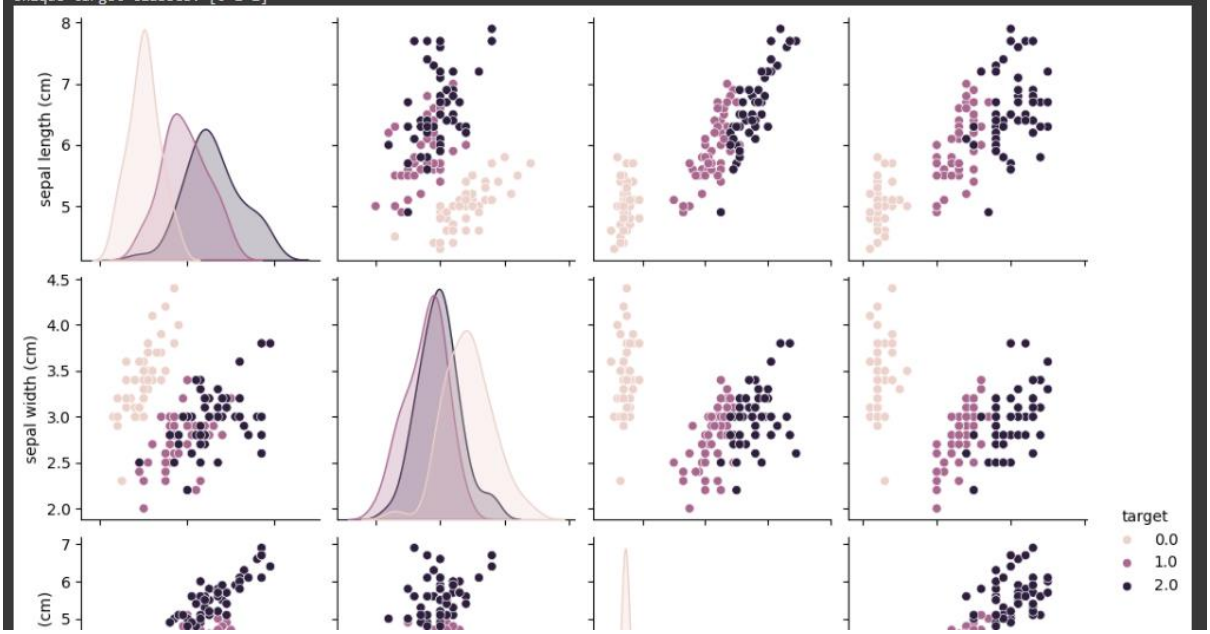
Prediction is the process of applying the trained model to new, unseen data. This section demonstrates how to make predictions using the trained Random Forest classifier, including an example prediction on a new sample.

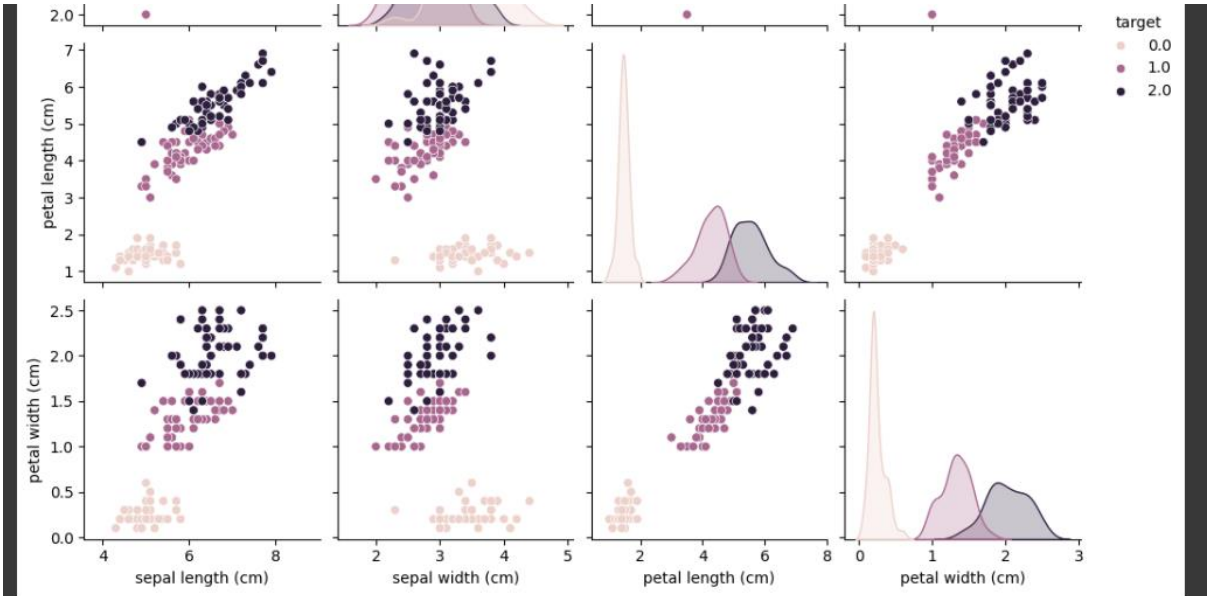
Python based model code / output :

```
import numpy as np
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.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
iris = load_iris()
X = iris.data
y = iris.target
print("Number of samples:", X.shape[0])
print("Number of features:", X.shape[1])
print("Unique target classes:", np.unique(y))
sns.pairplot(pd.DataFrame(np.c_[iris['data'], iris['target']],
columns=iris['feature_names'] + ['target']), hue='target')
plt.show()
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.2, random_state=42)
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
y_pred = rf_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:")
print(classification_report(y_test, y_pred))
new_sample = [[5.1, 3.5, 1.4, 0.2]]
new_sample_scaled = scaler.transform(new_sample)
predicted_class = rf_model.predict(new_sample_scaled)[0]
predicted_species = iris.target_names[predicted_class]
print("Predicted Species:", predicted_species)
```

OUTPUT:

```
Number of samples: 150
Number of features: 4
Unique target classes: [0 1 2]
```






```
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

Predicted Species: setosa
```

Explanation:

The final section serves as a conclusion to the project. It summarizes the key findings, including the model's accuracy, performance metrics, and practical implications. Additionally, it provides an explanation of the model's workings and its relevance to real-world applications.

Conclusion:

The machine learning project outlines the objective, which is to build a machine learning model to classify iris flowers into different species. Additionally, it mentions that the Iris dataset from scikit-learn will be used as the data source.

SHAIK NOORUDDIN

2200040145@kluniversity.in