# CODE

```
# Import required libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
import seaborn as sns
import matplotlib.pyplot as plt

# Step 1: Load the Dataset from CSV
df = pd.read_csv('iris_is.csv', encoding='ISO-8859-1')  # Load the dataset
print(df)

# Step 2: Perform Basic EDA
def basic_eda(data):
    """
    Basic EDA functions to check the data info, head, description, and missing values
    """
    print("First 5 rows of the dataset:\n", data.head())
    print("\nInfo about the dataset:\n", data.info())
    print("\nStatistical description of the dataset:\n", data.describe())

    # Checking for missing values
    print("\nMissing values in each column:\n", data.isnull().sum())

# Step 3: Handle Missing Values (if any)
def handle_missing_values(data):
    """
```

```
    Handle missing values by filling them with the mean (for numeric columns)
    """
    # Select only numeric columns (excluding 'species' column)
    numeric_columns = data.select_dtypes(include=['float64', 'int64']).columns
    data[numeric_columns] = data[numeric_columns].fillna(data[numeric_columns].mean())

    print("\nMissing values after handling:\n", data.isnull().sum())
    return data

# Step 4: Data Visualization (Exploratory Data Analysis)
def plot_eda(data):
    """
    Visualize relationships between features using pairplot and heatmap
    """
    # Pairplot to see relationships between the features
    sns.pairplot(data, hue='species')
    plt.title("Pairplot of Iris Dataset")
    plt.show()

    # Correlation Heatmap (exclude the 'species' column)
    correlation_matrix = data.drop(columns='species').corr()
    sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
    plt.title("Correlation Heatmap")
    plt.show()

# Step 5: Data Preprocessing and Model Training
def train_model(data):
    """
    Function to train a Random Forest model for Iris classification
```

```
    """
    # Define features and target variable
    X = data.drop(columns=['species'])
    y = data['species']

    # Split the data into train and test sets (80% train, 20% test)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Create the RandomForestClassifier model
    model = RandomForestClassifier(random_state=42)

    # Train the model
    model.fit(X_train, y_train)

    # Make predictions on the test set
    y_pred = model.predict(X_test)

    # Evaluate the model
    print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
    print("\nClassification Report:\n", classification_report(y_test, y_pred))

    return model, X_train, X_test, y_train, y_test, y_pred

# Step 6: Predicting for New Data
def predict_new_data(model, new_data, feature_names):
    """
    Predict the species of a new Iris flower based on its measurements.
    Ensure new data has the same feature names as the trained model.
    """
```

```
    # Convert the new data to a DataFrame with the correct feature names
    new_data_df = pd.DataFrame(new_data, columns=feature_names)

    # Make the prediction
    prediction = model.predict(new_data_df)
    print("\nPredicted species for the new flower:", prediction[0])

# Main function to execute the steps
if __name__ == "__main__":
    # Load the data (already done at the top)
    data = df  # Using the DataFrame directly since it is already loaded

    # Perform basic EDA
    basic_eda(data)

    # Handle missing values (if any)
    data = handle_missing_values(data)

    # Visualize the data with pairplot and heatmap
    plot_eda(data)

    # Train the model and evaluate
    model, X_train, X_test, y_train, y_test, y_pred = train_model(data)

    # Get the feature names (columns) from the training data
    feature_names = X_train.columns

    # Example: Predicting a new flower's species
    new_flower = [[5.1, 3.5, 1.4, 0.2]]  # Sepal length, Sepal width, Petal length, Petal width
```

```
predict_new_data(model, new_flower, feature_names)
```

## OUTPUT

```
 sepal_length  sepal_width  petal_length  petal_width         species
0           5.1          3.5           1.4          0.2     Iris-setosa
1           4.9          3.0           1.4          0.2     Iris-setosa
2           4.7          3.2           1.3          0.2     Iris-setosa
3           4.6          3.1           1.5          0.2     Iris-setosa
4           5.0          3.6           1.4          0.2     Iris-setosa
..          ...          ...           ...          ...             ...
145          6.7          3.0           5.2          2.3  Iris-virginica
146          6.3          2.5           5.0          1.9  Iris-virginica
147          6.5          3.0           5.2          2.0  Iris-virginica
148          6.2          3.4           5.4          2.3  Iris-virginica
149          5.9          3.0           5.1          1.8  Iris-virginica

[150 rows x 5 columns]
First 5 rows of the dataset:
   sepal_length  sepal_width  petal_length  petal_width      species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
```

```
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB

Info about the dataset:
 None

Statistical description of the dataset:
        sepal_length  sepal_width  petal_length  petal_width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000

Missing values in each column:
 sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
```

dtype: int64

Missing values after handling:

sepal_length 0

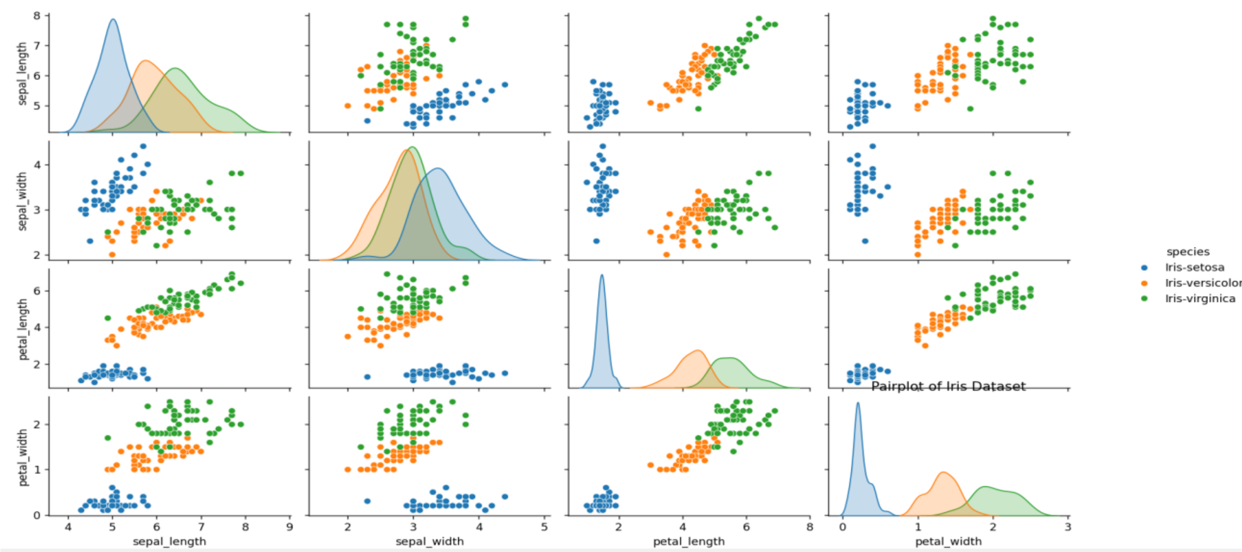
sepal_width 0

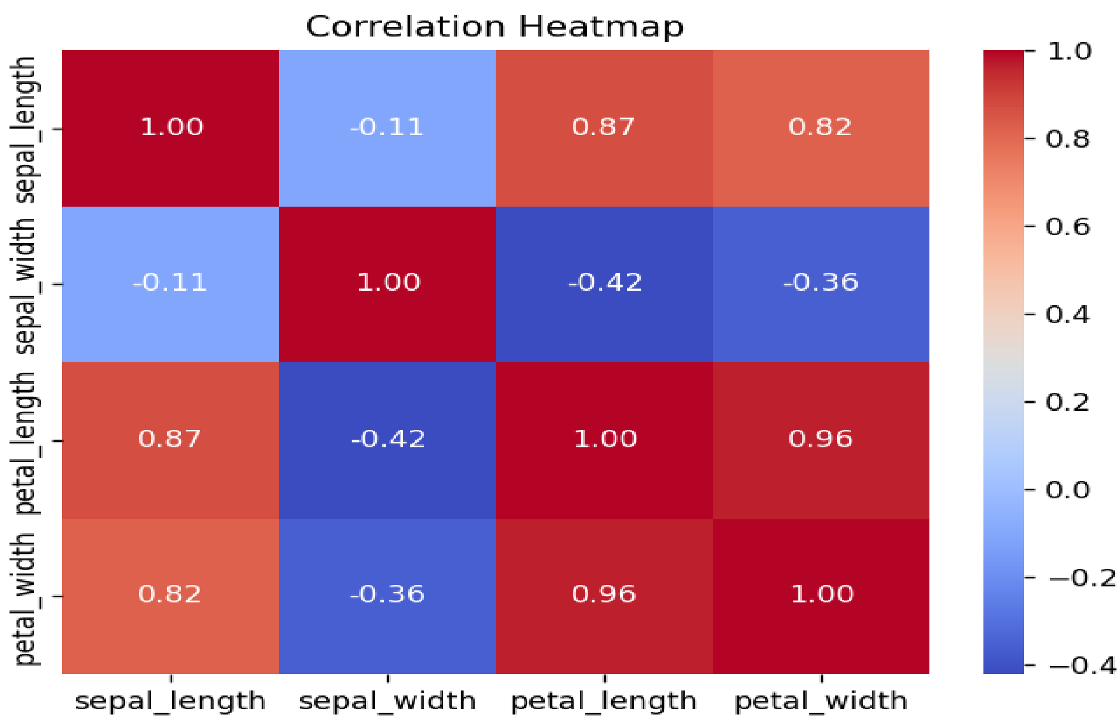
petal_length 0

petal_width 0

species 0

dtype: int64






Accuracy: 1.0

Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Iris-setosa | 1.00 | 1.00 | 1.00 | 10 |
| Iris-versicolor | 1.00 | 1.00 | 1.00 | 9 |
| Iris-virginica | 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 for the new flower: Iris-setosa