Part 2: Data Science and Machine Learning

• Exploratory Data Analysis (EDA):

o Use libraries like pandas and matplotlib/ seaborn:

▪ Load a dataset of your choice (find open-source datasets on Kaggle or UCI Machine Learning repository).

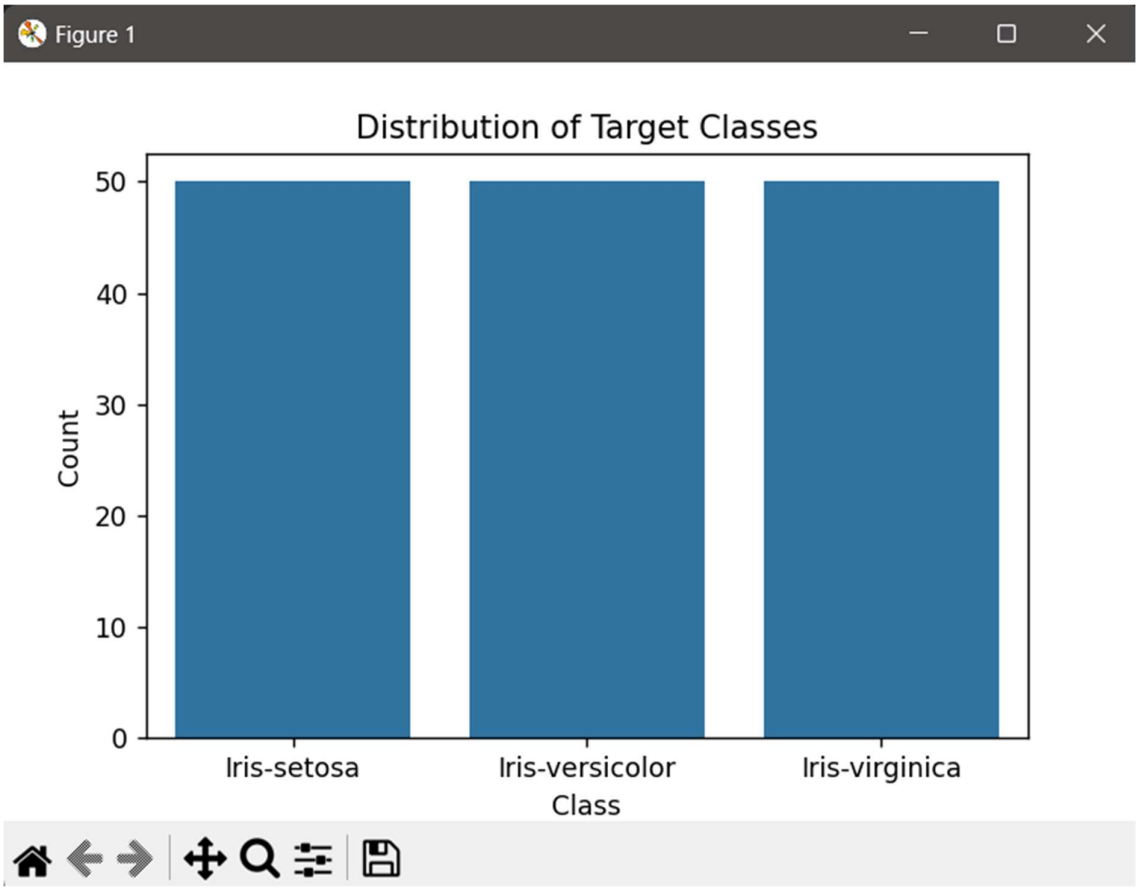
▪ Perform EDA, including summary statistics, distribution analysis, and correlation analysis.

▪ Create informative visualizations.

C: > Users > 91996 > Desktop > Python > 1.py > ...

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the Iris dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
column_names = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']
iris_df = pd.read_csv(url, names=column_names)

# Display the first few rows of the dataset
print("First few rows of the Iris dataset:")
print(iris_df.head())

# Summary statistics
print("\nSummary statistics of the Iris dataset:")
print(iris_df.describe())

# Distribution of target classes
plt.figure(figsize=(6, 4))
sns.countplot(x='class', data=iris_df)
plt.title("Distribution of Target Classes")
plt.xlabel("Class")
plt.ylabel("Count")
plt.show()

# Pairplot for visualizing relationships between variables
sns.pairplot(iris_df, hue='class', diag_kind='hist')
plt.title("Pairplot of the Iris dataset")
plt.show()

# Correlation analysis
correlation_matrix = iris_df.corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```

```
PS C:\Users\91996\Desktop\WEB DEV> & C:/Python311/python.exe c:/Users/91996/Desktop/Python/1.py
First few rows of the Iris dataset:
   sepal_length  sepal_width  petal_length  petal_width        class
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

Summary statistics of the Iris 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
```



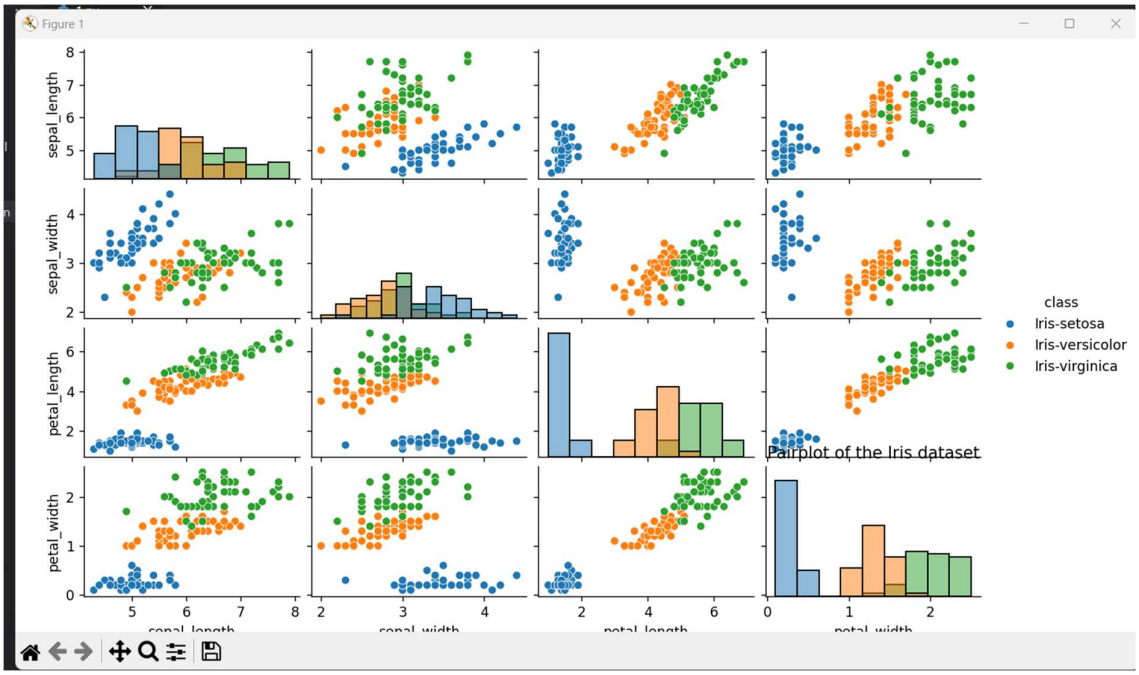


Figure 1
sepal_length
sepal_width
petal_length
petal_width
class
Iris-setosa
Iris-versicolor
Iris-virginica
Pairplot of the Iris dataset

Machine Learning Model:

o Choose a relevant classification or regression algorithm (e.g., Random Forest, Linear Regression) based on the dataset.

o Split the data, train the model, and evaluate its performance using appropriate metrics.

o Experiment with hyperparameter tuning to potentially improve the model.

C: > Users > 91996 > Desktop > Python > 1.py > ...

```python
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load the Iris dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
column_names = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']
iris_df = pd.read_csv(url, names=column_names)

# Split the data into features (X) and target variable (y)
X = iris_df.drop('class', axis=1)
y = iris_df['class']

# Split the data 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)

# Initialize the Random Forest Classifier
rf_classifier = RandomForestClassifier(random_state=42)

# Define hyperparameters for tuning
param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Perform hyperparameter tuning using GridSearchCV
grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)

```

```
32
33  # Print the best parameters found by GridSearchCV
34  print("Best parameters found by GridSearchCV:")
35  print(grid_search.best_params_)
36
37  # Evaluate the model on the test set
38  y_pred = grid_search.predict(X_test)
39  accuracy = accuracy_score(y_test, y_pred)
40  print("\nAccuracy:", accuracy)
41
42  # Print classification report
43  print("\nClassification Report:")
44  print(classification_report(y_test, y_pred))
45
```

```
PS C:\Users\91996\Desktop\WEB DEV> & C:/Python311/python.exe c:/Users/91996/Desktop/Python/1.py
Best parameters found by GridSearchCV:
{'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 150}

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
```