MACHINE LEARNING ENVIRONMENT SETUP & DATA ANALYSIS USING IRIS DATASET

A step-by-step guide

Aagam B.Shah

24202701

NTRODUCTION

Objective: Setting up a Python environment for ML and performing basic data analysis.

Expected Learning Outcomes:

- Understanding Python ML libraries.
- Hands-on dataset loading and preprocessing.
- Learning basic data visualization techniques.
- Understanding data distribution, correlation, and relationships between features.
- Preparing data for machine learning applications.

TASK 1 - ENVIRONMENT SETUP

nstall required libraries:

- NumPy: Provides support for large, multi-dimensional arrays and matrices, along with mathematical functions for numerical operations.
- Pandas: A library for data manipulation and analysis, particularly useful for handling structured data in tabular format.
- Matplotlib & Seaborn: Used for data visualization, where Seaborn provides enhanced statistical plotting capabilities.
- Scikit-learn: A machine learning library that includes dataset loading, preprocessing, model selection, and evaluation tools.

ython Code:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import datasets
```

VERIFYING INSTALLATIONS

Checking Versions:

```
print(pd.__version__)
print(np.__version__)
```

Why Check Versions?

- Ensures compatibility and stability of libraries.
- Avoids errors due to deprecated functions in older versions.

Output:

- Displays installed versions, confirming successful setup.

TASK 2 - LOADING THE IRIS DATASET

he **Iris dataset** is a well-known dataset containing **150 samples** from three different species of Iris owers:

Setosa, Versicolor and Virginica

ach sample includes **four features:**

Sepal Length, Sepal Width, Petal Length and Petal Width

ython Code:

```
rom sklearn.datasets import load_iris
ris = load_iris()
f = pd.DataFrame(data=iris.data, columns=iris.feature_names)
f['species'] = iris.target
```

Vhy Load with Pandas?

Allows easy data manipulation and visualization.

EXPLORING THE DATASET

Dataset Information:

ython code :

```
df.shape
df.dtypes
df.info()
df.describe()
```

Key Insights:

- Number of rows and columns.
- Data types of features.
- Summary statistics (mean, standard deviation, min, max, etc.).Helps identify any missing or inconsisten
  data.

Task 3 - Data Visualization

Histograms: Understanding Feature Distributions

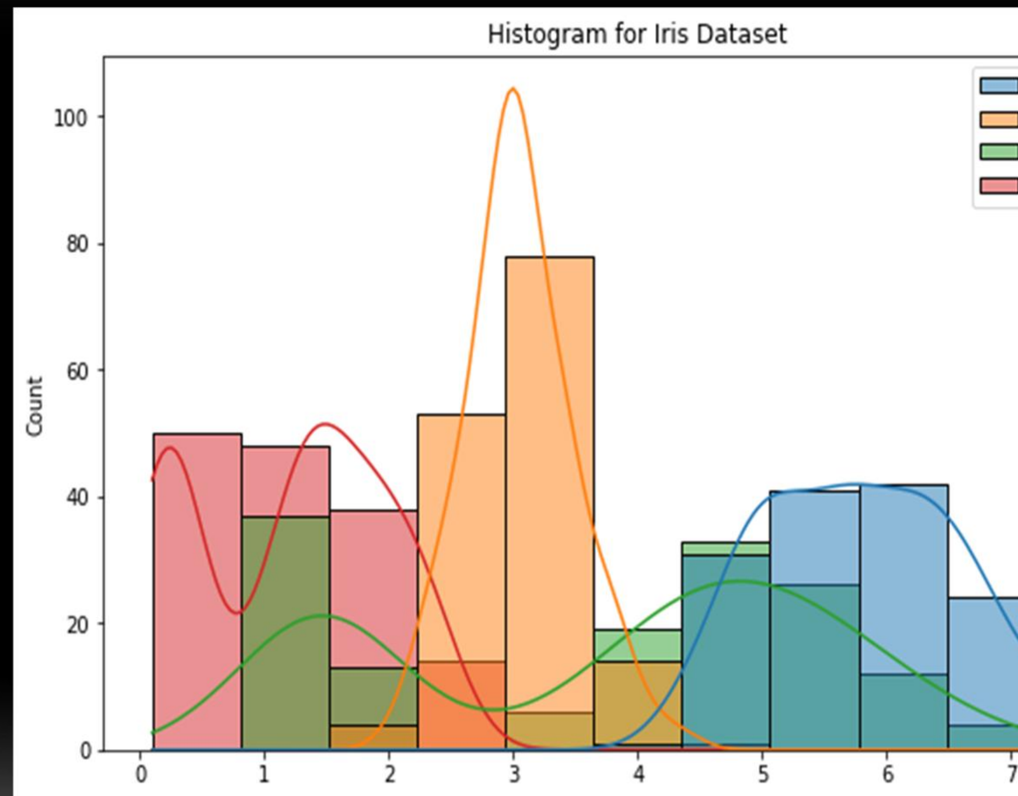
Purpose:

Histograms display the frequency distribution of numerical data.

Helps understand feature spread and identify patterns

Python Code :

```
plt.figure(figsize=(10,6))
sns.histplot(df,kde = True)
plt.title('Histogram for Iris Dataset')
plt.show()
```



Interpretation:

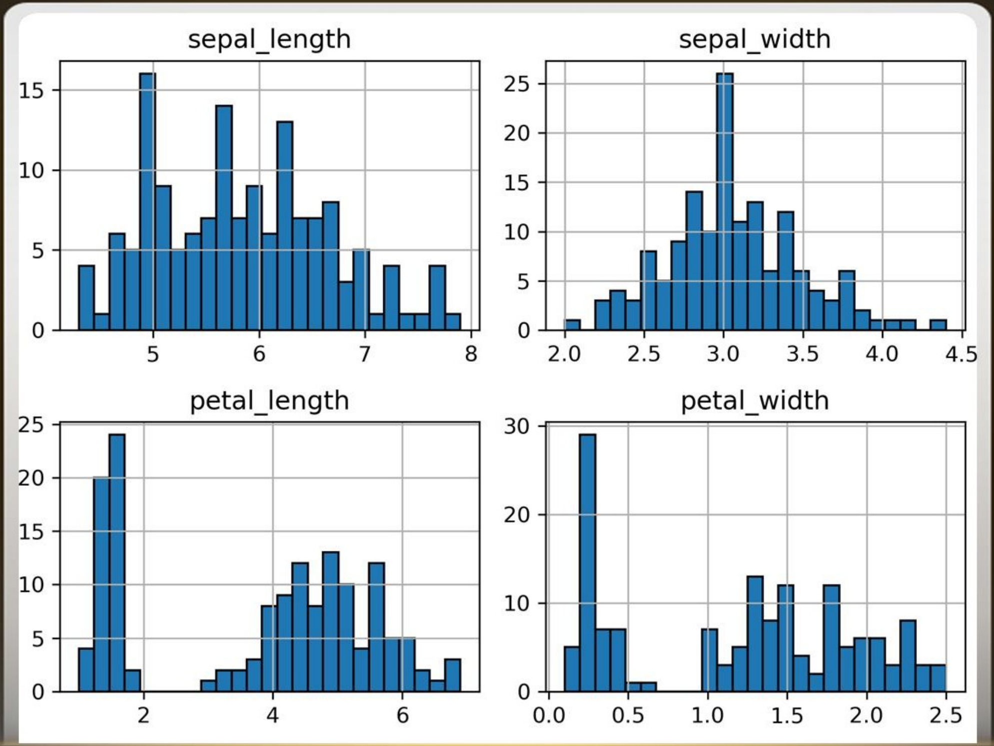
Peaks indicate frequent values.

Can detect skewness or uniformity in the data.

Helps in identifying whether features require normalization.

Python code :

```
plt.figure(figsize = (10,6))
df.hist(bins = 25,edgecolor = 'black')
plt.tight_layout()
plt.show()
```



Box Plots for Outliers

urpose:

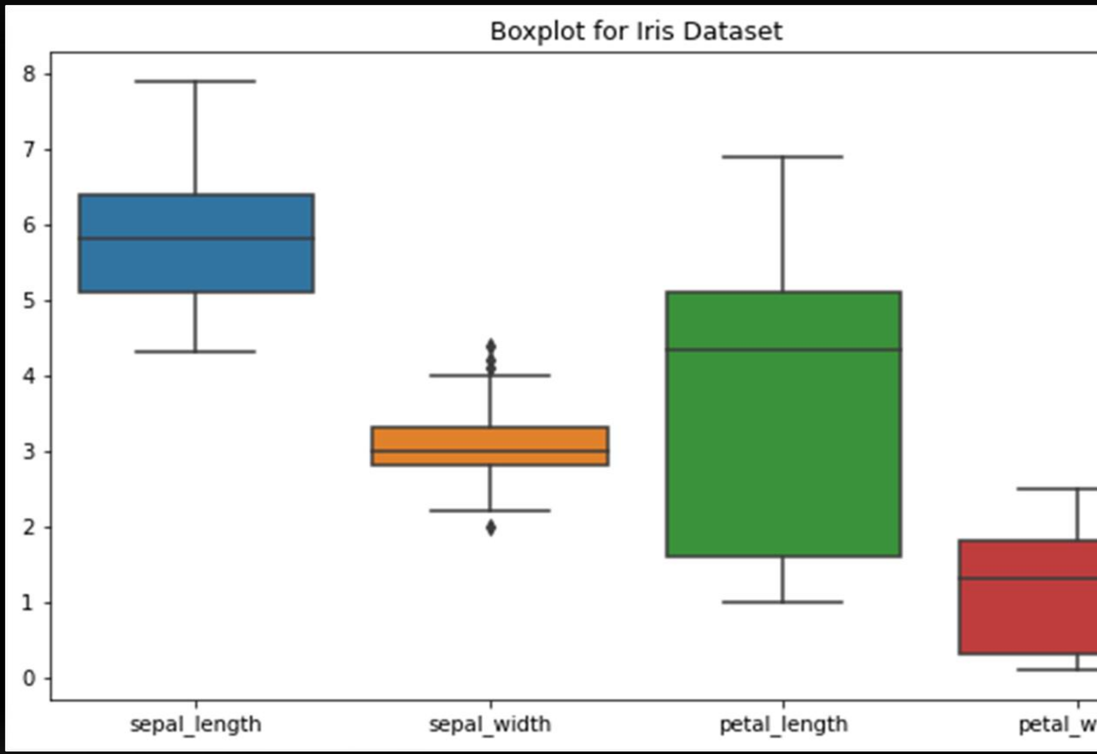
- Box plots help detect outliers by displaying data spread using quartiles.
- Shows median, interquartile range (IQR), and potential outliers.

ython Code :

```
lt.figure(figsize = (10,6))
ns.boxplot(data = df)
lt.title('Boxplot for Iris Dataset')
lt.show()
```

nterpretation:

- Features like petal length and petal width show clear eparation between species.
- Helps in feature selection for classification tasks.



Pair Plots: Exploring Feature Relationships

urpose:

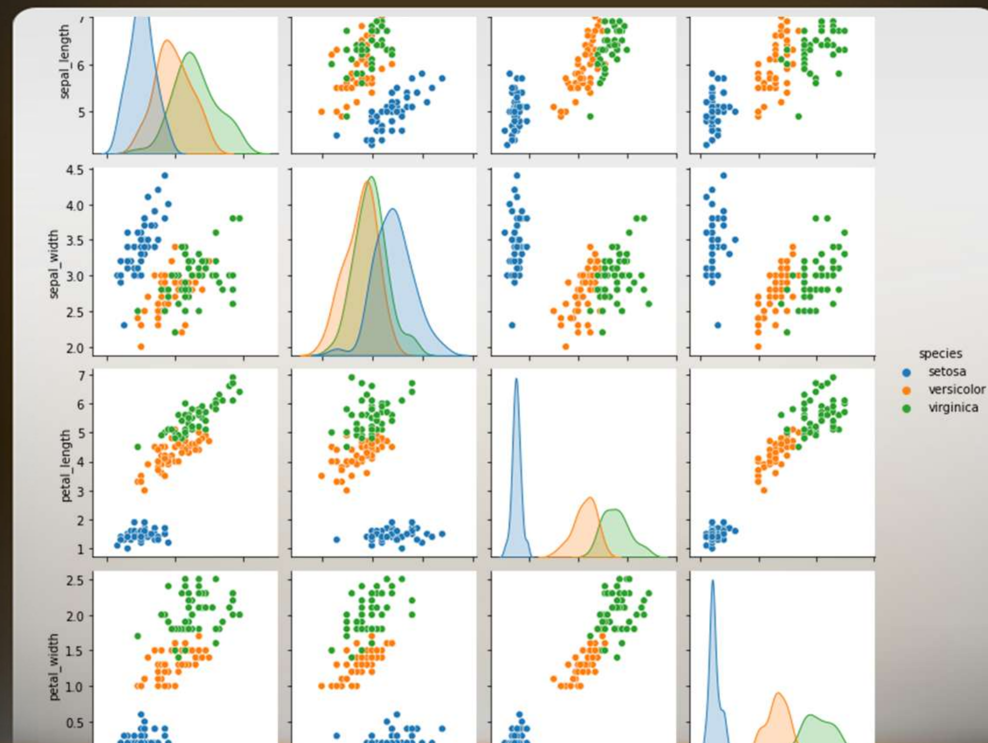
- Pair plots show the relationships between ifferent features.
- Helps visualize how well features separate ifferent species.

ython Code:

```
ns.pairplot(df,hue = 'species’)
lt.show()
```

nterpretation:

- Features like petal length and petal width show lear separation between species.
- Helps in feature selection for classification asks.



Correlation Heatmap: Feature Relationships

urpose:

A heatmap visually represents feature orrelations.

High correlation between features may indicate edundancy.

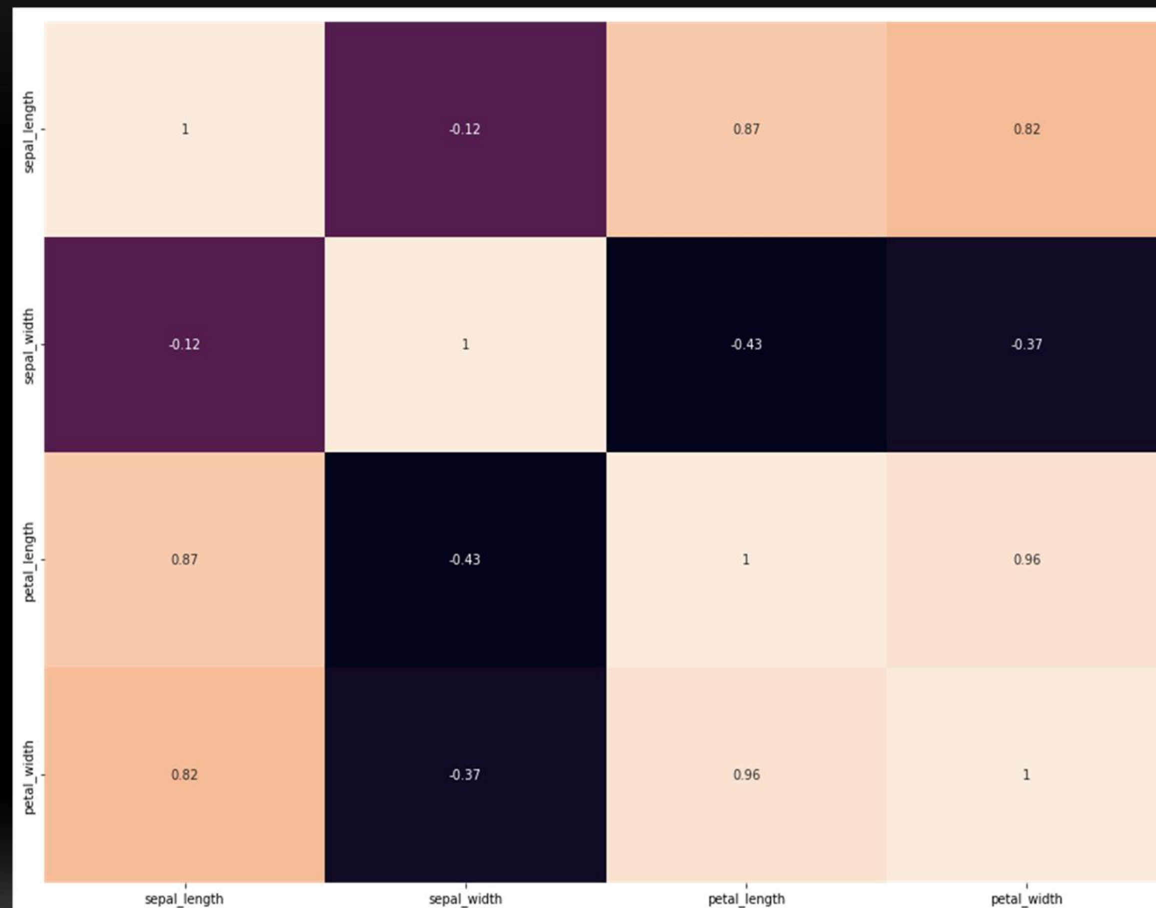
ython Code :

```
corr = df.corr(numeric_only = True)
plt.figure(figsize = (20,12))
sns.heatmap(corr,annot = True)
plt.show()
```

nterpretation:

Strong positive/negative correlations help in eature engineering.

Weak correlations suggest independent ariables.



TASK 4 - BASIC DATA PREPROCESSING

Handling Missing Values:

Python code :

```
df.isnull()
df.isnull().sum()
```

Why Check for Missing Values?

- Missing values can impact model accuracy.
- The Iris dataset does not contain missing values, but this step is crucial for real-world datasets.

STANDARDIZATION

Why Standardization?

- Ensures features have a mean of 0 and standard deviation of 1.
- Essential when using algorithms sensitive to feature scales.

Python Code :

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df.iloc[:, :-1])
```

SPLITTING DATA INTO TRAIN & TEST SETS

Why Split Data?

- To evaluate model performance on unseen data.
- To also prevent data from overfitting.

ython Code:

```
rom sklearn.model_selection import train_test_split
 = df.iloc[:,:-1].values
 = df.iloc[:,-1].values
_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
```

plit Ratio:

- 80% training data.
- 20% testing data.

CONCLUSION

Key Takeaways:

- Successfully set up Python ML environment.
- Explored and visualized the Iris dataset.
- Performed preprocessing for ML models.

Next Steps:

- Apply ML algorithms (e.g., kNN, Decision Trees) to classify Iris species.
- Perform model evaluation and optimization.

THANK YOU!