Assignment 1

Almansur Kakimov
Department of Intelligent Systems and Cybersecurity, Astana IT University
221181@astanait.edu.kz
Machine Learning, Prof. Mimenbayeva A.B
17.12.2024

1. Introduction

The goal of this analysis was to explore and clean a dataset to prepare it for future tasks like machine learning modeling. The steps included understanding the data through visualizations, handling missing values, encoding categorical columns, and analyzing relationships between numeric features.

2. Exploring the Data

We started by loading the dataset and looking at its structure. Histograms were used to show how the numeric features are distributed, and scatter plots helped us see how different features relate to each other. For example, the petal_length and petal_width features showed clear patterns when grouped by the target column (species). Violin plots were also created to compare the sepal_length feature for each class of the target variable, which showed noticeable differences between the species.

3. Handling Missing Values

Next, we checked for missing values in the dataset. Any missing values in numeric columns were filled with the mean of the respective columns. This ensured the dataset had no gaps and was ready for further analysis

4. Identifying Numeric and Categorical Columns

After this, we identified two types of columns: numeric columns (sepal_length, sepal_width, petal_length, and petal_width) and the categorical column (species).

5. Encoding Categorical Columns

To prepare the data for machine learning, we encoded the species column using two methods. First, one-hot encoding created new binary columns for each class of the target variable. Second, label encoding converted the classes into numbers, where Setosa was 0, Versicolor was 1, and Virginica was 2. Both methods successfully transformed the species column into a format that machines can understand.

6. Analyzing Correlations

Finally, we looked at the relationships between numeric features using a correlation matrix and a heatmap. This showed that petal_length and petal_width had a strong positive relationship, meaning they increase together. There was also a moderate positive relationship between sepal_length and petal_length. These relationships can help us understand which features are important for predicting the target variable.

7. **Conclusion**

In conclusion, the dataset was explored and cleaned effectively. We used visualizations to find patterns, filled missing values, and prepared the data using encoding methods. Correlation analysis helped us identify important relationships between features. This cleaned and structured dataset is now ready for further tasks like building machine learning models. The next steps could involve selecting the most important features and training models to make predictions.

Code Explanation: Data Analysis Process

1. **Importing Required Libraries**

The first step in the code is to import necessary libraries. These libraries are essential for data manipulation, visualization, and preprocessing tasks. We use pandas for handling the dataset, numpy for numerical operations, matplotlib and seaborn for creating visualizations, and plotly.express for advanced interactive plotting. Additionally, LabelEncoder from sklearn is imported to handle the encoding of categorical variables.

2. **Setting Plot Style**

To improve the appearance of the plots, the code uses the sns.set_theme() function to set the plot style to "darkgrid." This ensures that the visualizations will have a consistent, aesthetically pleasing background.

3. **Loading the Dataset**

The dataset is loaded using the pd.read_csv() function from a URL that points to the Iris dataset. The dataset consists of four numeric features (sepal_length, sepal_width, petal_length, and petal_width) and one categorical feature (species). The column names are explicitly set using the names argument in pd.read_csv(). After loading the data, the first few rows are displayed with df.head() to give an overview of the dataset.

4. **Checking Dataset Information**

Next, the code uses the df.info() function to display information about the dataset, such as the number of non-null values and the data types of each column. This helps to understand the structure and types of data present in the dataset.

5. **Logical Ideas for Approach**

The code outlines several logical approaches for analyzing the data. These include performing Exploratory Data Analysis (EDA) to understand the dataset through visualizations and statistics, handling missing values, encoding categorical features, and analyzing correlations between features to identify key factors that influence the target variable.

Data Visualization

1. Histograms for Numeric Features

The next step visualizes the distribution of numeric features using histograms. The df.hist() function is used to create histograms for each numeric column. The bins parameter is set to 20 for better granularity, and the figsize parameter ensures the plots are large enough to see details. The histograms help to identify the distribution of each feature, such as whether they follow a normal or skewed distribution.

2. Pairplot to Visualize Relationships

A pairplot is created using sns.pairplot() to visualize the relationships between all pairs of features, colored by the species column. This plot helps to identify patterns and relationships between features, and it shows how the target variable (species) is distributed across the feature space.

3. Violin Plot for Target vs Numeric Features

A violin plot is generated using plotly.express to show the distribution of the sepal_length feature for each species. This plot combines aspects of box plots and density plots, offering a clearer view of the distribution and allowing for comparison between the different species.

Handling Missing Values

1. **Checking for Missing Values**

The code checks for missing values using the df.isnull().sum() function, which provides the count of missing values for each column. This is an important step to ensure that the dataset is complete before performing further analysis.

2. **Filling Missing Values**

If any missing values are found, they are filled using the mean of each respective numeric column. This is done with a loop that iterates over the numeric columns, replacing missing values with the mean using df[col].fillna(df[col].mean(), inplace=True). This imputation ensures that the dataset is complete and ready for analysis.

3. **Verifying Missing Values After Imputation**

After filling missing values, the code verifies that there are no remaining missing values by printing the result of df.isnull().sum(). This ensures that the imputation process was successful.

4. **Identifying Numeric and Categorical Columns**

The code separates the dataset's columns into numeric and categorical ones. The select_dtypes() function is used to identify columns with numeric data types (int64, float64) and categorical data types (object). The names of these columns are stored in separate lists, numeric_cols and categorical_cols, to aid in further processing.

5. **Encoding Categorical Features**

One-Hot Encoding The categorical columns are encoded using one-hot encoding with pd.get_dummies(). This creates new binary columns for each category in the species column, turning categorical variables into a format suitable for machine learning models.

6. **Label Encoding**

Label encoding is performed using LabelEncoder() from sklearn. This method converts the categories of the species column into numerical labels (0, 1, 2). The label encoding step makes the data easier for machine learning algorithms to process by converting categorical labels into numeric ones.

7. **Correlation Analysis**

Computing the Correlation Matrix The correlation matrix for numeric columns is computed using the .corr() function. This matrix shows the strength of the relationships between numeric features. The values range from -1 (strong negative correlation) to 1 (strong positive correlation), and a value of 0 indicates no correlation.

8. **Visualizing the Correlation**

Matrix A heatmap of the correlation matrix is created using sns.heatmap(). This visualization helps to quickly identify which features are strongly correlated. The annot=True option displays the correlation values on the heatmap, and cmap='coolwarm' ensures that high correlations are shown in warm colors, making the correlations easier to interpret.

9. **Conclusion**

The analysis concludes with a statement about the findings. The heatmap of the correlation matrix is used to identify features that are strongly correlated, which can help in feature selection and understanding the relationships in the dataset. Highly correlated features can be more influential in predicting the target variable.

```python
# Import Required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.preprocessing import LabelEncoder

# Optional: Set styles for plots
sns.set_theme(style="darkgrid")
```

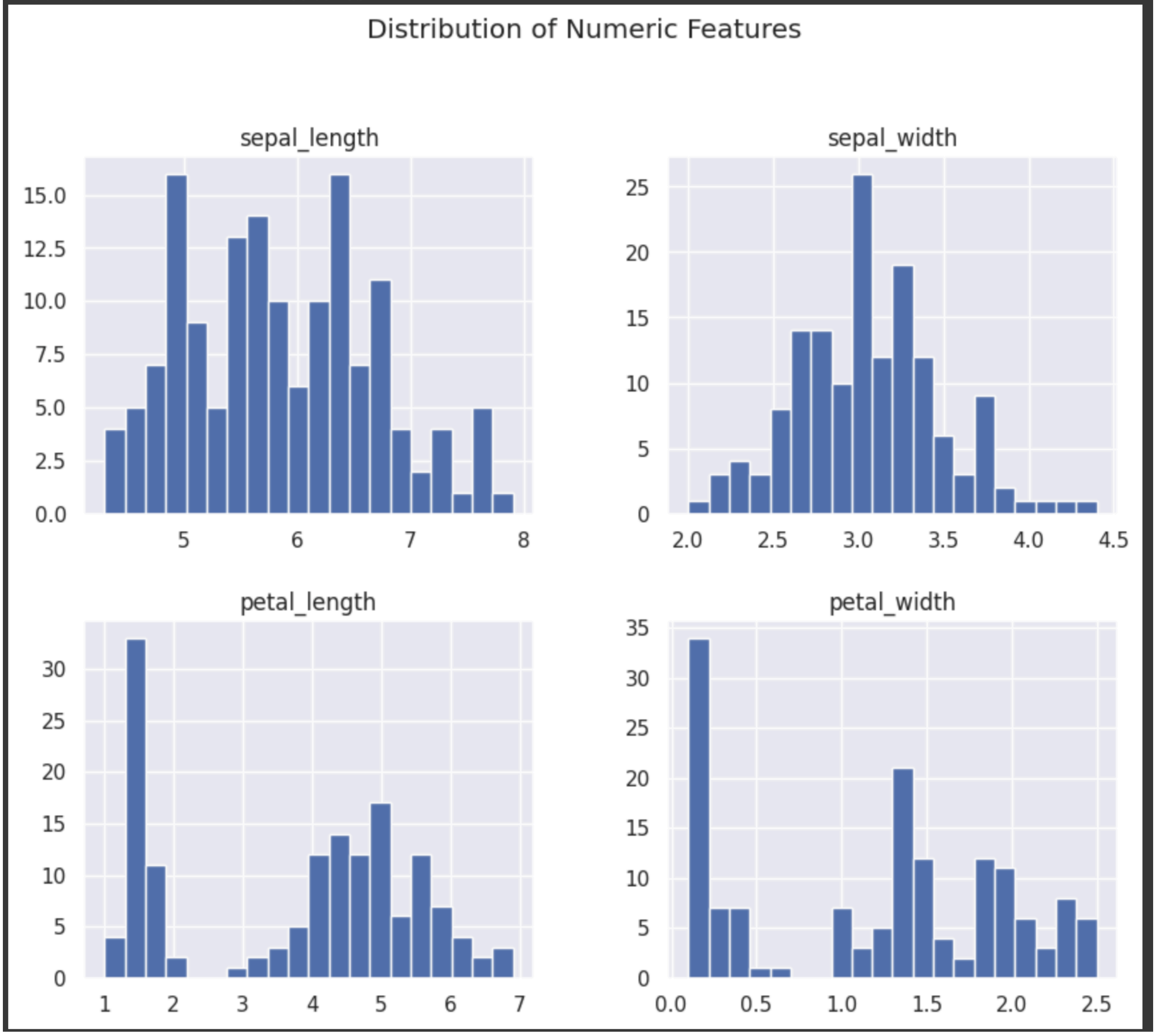
```python
# Load Dataset
# Replace the URL with your desired dataset
dataset_url =
'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.da
ta'
column_names = ['sepal_length', 'sepal_width', 'petal_length',
'petal_width', 'species']
df = pd.read_csv(dataset_url, header=None, names=column_names)

# Display First Few Rows
print("First 5 Rows of the Dataset:")
print(df.head())

# Check Dataset Information
print("\nDataset Information:")
print(df.info())

# Task 1: Logical Approaches
print("\nTask 1: Logical Ideas for Approaching the Problem")
print("1. Perform Exploratory Data Analysis (EDA) using graphs and
statistics.")
print("2. Handle missing values, encode categorical features, and
analyze correlations.")
print("3. Identify key features that influence the target variable.")

# Task 2: Data Visualization
print("\nTask 2: Data Visualization")

# Histograms for Numeric Features
print("\nDistribution of Numeric Features:")
df.hist(figsize=(10, 8), bins=20)
plt.suptitle("Distribution of Numeric Features", y=1.01)
plt.show()

# Pairplot to Visualize Relationships Between Features
print("\nPairplot to Visualize Relationships:")
sns.pairplot(df, hue='species')
plt.show()

# Violin Plot for Target vs Features
print("\nViolin Plot for Target (species) vs Numeric Features:")
```

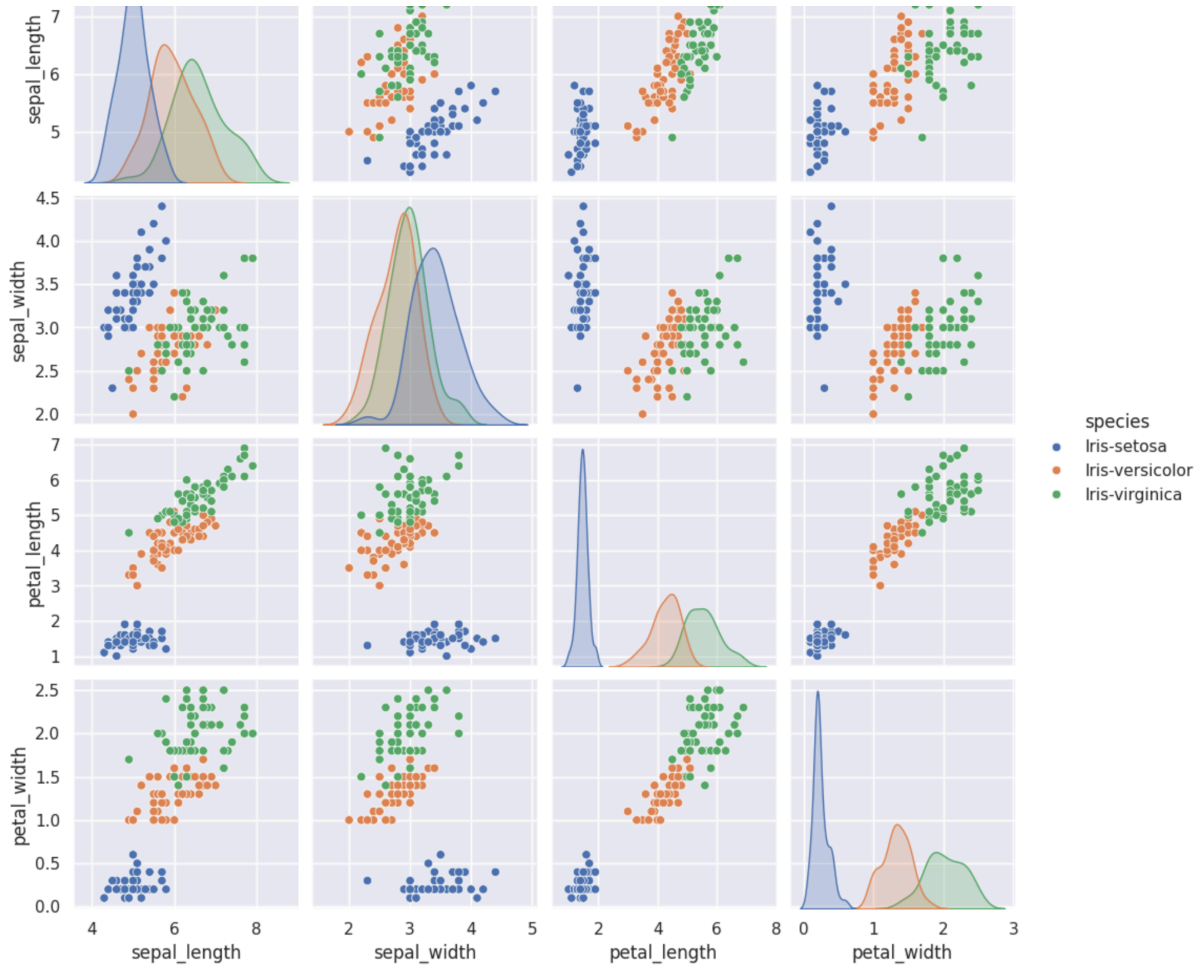
```python
fig = px.violin(df, x='species', y='sepal_length', box=True,
points="all")
fig.show()

# Task 3: Handling Missing Values
print("\nTask 3: Handling Missing Values")

# Check for Missing Values
print("Missing Values in Each Column:")
print(df.isnull().sum())

# Count Total Missing and Non-Missing Values
total_missing = df.isnull().sum().sum()
print(f"\nTotal Missing Values: {total_missing}")
print(f"Total Non-Missing Values: {df.notnull().sum().sum()}")

# Fill Missing Numeric Columns with Mean
for col in df.select_dtypes(include=['float64', 'int64']):
   df[col].fillna(df[col].mean(), inplace=True)

# Verify Missing Values After Imputation
print("\nMissing Values After Imputation:")
print(df.isnull().sum())

# Task 4: Separate Numeric and Categorical Columns
print("\nTask 4: Identify Numeric and Categorical Columns")

# Separate Numeric and Categorical Columns
numeric_cols = df.select_dtypes(include=['int64',
'float64']).columns.tolist()
categorical_cols =
df.select_dtypes(include=['object']).columns.tolist()

print("Numeric Columns:", numeric_cols)
print("Categorical Columns:", categorical_cols)

# Check Data Types
print("\nData Types of Columns:")
print(df.dtypes)

# Task 5: Encoding Categorical Features
print("\nTask 5: Encoding Categorical Features")
```

```python
# One-Hot Encoding
print("\nOne-Hot Encoding of Categorical Columns:")
df_one_hot = pd.get_dummies(df, columns=categorical_cols)
print(df_one_hot.head())

# Label Encoding
print("\nLabel Encoding of Categorical Columns:")
le = LabelEncoder()
for col in categorical_cols:
   df[col] = le.fit_transform(df[col])

print("After Label Encoding:")
print(df.head())

# Task 6: Correlation Analysis
print("\nTask 6: Correlation Analysis")

# Compute Correlation Matrix
correlation_matrix = df[numeric_cols].corr()

# Plot Heatmap for Correlation Matrix
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Correlation Heatmap")
plt.show()

# Conclusion
print("\nTask 6 Conclusion:")
print("The heatmap shows relationships between numeric features.
Features with high correlation can influence the target variable.")
```

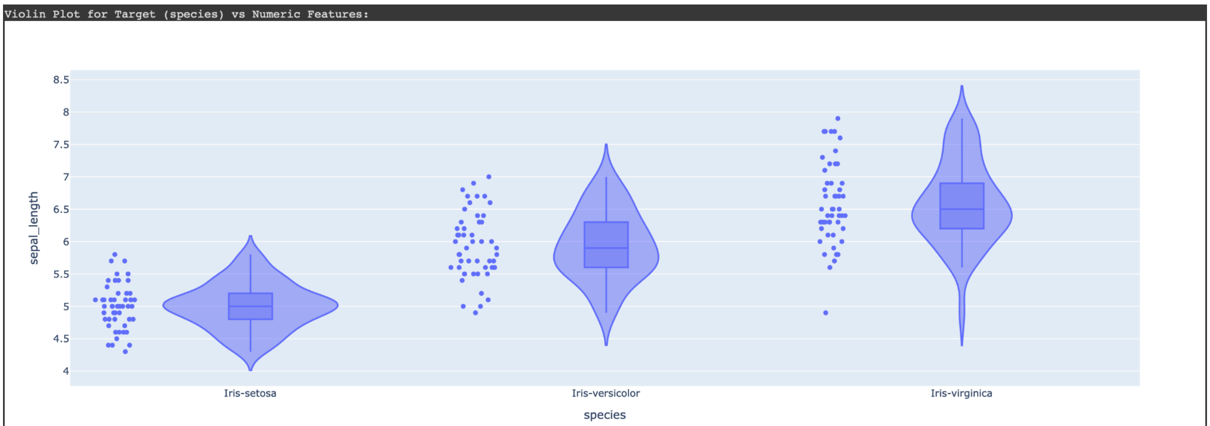
```
First 5 Rows of the Dataset:
   sepal_length  sepal_width  petal_length  petal_width      species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa

Dataset Information:
<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
None

Task 1: Logical Ideas for Approaching the Problem
1. Perform Exploratory Data Analysis (EDA) using graphs and statistics.
2. Handle missing values, encode categorical features, and analyze correlations.
3. Identify key features that influence the target variable.
```

```
Task 2: Data Visualization

Distribution of Numeric Features:
```



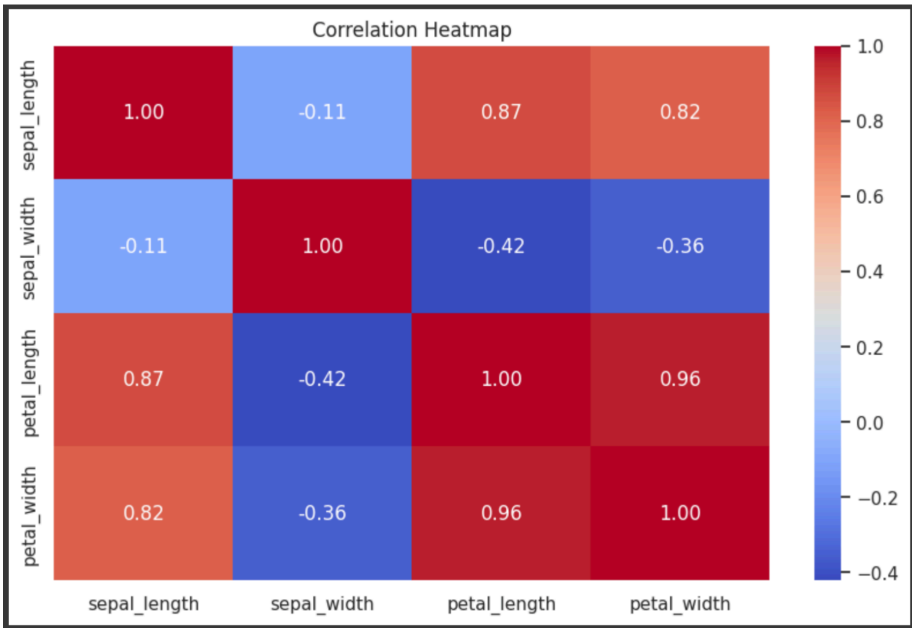


Violin Plot for Target (species) vs Numeric Features:



```
Task 3: Handling Missing Values
Missing Values in Each Column:
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64

Total Missing Values: 0
Total Non-Missing Values: 750

Missing Values After Imputation:
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64

Task 4: Identify Numeric and Categorical Columns
Numeric Columns: ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
Categorical Columns: ['species']

Data Types of Columns:
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

```
Task 5: Encoding Categorical Features

One-Hot Encoding of Categorical Columns:
   sepal_length  sepal_width  petal_length  petal_width  species_Iris-setosa  \
0           5.1          3.5           1.4          0.2                 True
1           4.9          3.0           1.4          0.2                 True
2           4.7          3.2           1.3          0.2                 True
3           4.6          3.1           1.5          0.2                 True
4           5.0          3.6           1.4          0.2                 True

   species_Iris-versicolor  species_Iris-virginica
0                    False                   False
1                    False                   False
2                    False                   False
3                    False                   False
4                    False                   False

Label Encoding of Categorical Columns:
After Label Encoding:
   sepal_length  sepal_width  petal_length  petal_width  species
0           5.1          3.5           1.4          0.2        0
1           4.9          3.0           1.4          0.2        0
2           4.7          3.2           1.3          0.2        0
3           4.6          3.1           1.5          0.2        0
4           5.0          3.6           1.4          0.2        0
```




```
Task 6 Conclusion:
The heatmap shows relationships between numeric features. Features with high correlation can influence the target variable.
```