NATIONAL UNIVERSITY OF MODERN LANGUAGES
ISLAMABAD

Machine Learning(Lab Task 03)

Lab Task: part 2

Submitted to
Miss Qurat u lin Raja
Submitted By
Adeel Naeem
(BSAI-146)

Submission Date: octuber 28th,2024

Histogram of Sepal Length:

Purpose: This code creates a histogram to visualize the distribution of the `sepal_length` variable in the Iris dataset.

Plotting:

`plt.figure(figsize=(10, 5))` sets the size of the figure to 10 inches wide and 5 inches tall for better visibility.

`sns.histplot(df['sepal_length'], kde=True)` generates the histogram, with a kernel density estimate (KDE) overlay for a smoother representation of the distribution.

Labels:

`plt.title('Histogram of Sepal Length')` adds a title to the histogram.

`plt.xlabel('Sepal Length')` labels the x-axis.

`plt.ylabel('Frequency')` labels the y-axis, indicating the count of occurrences for each sepal length range.

Display: `plt.show()` renders the plot.

Summary:

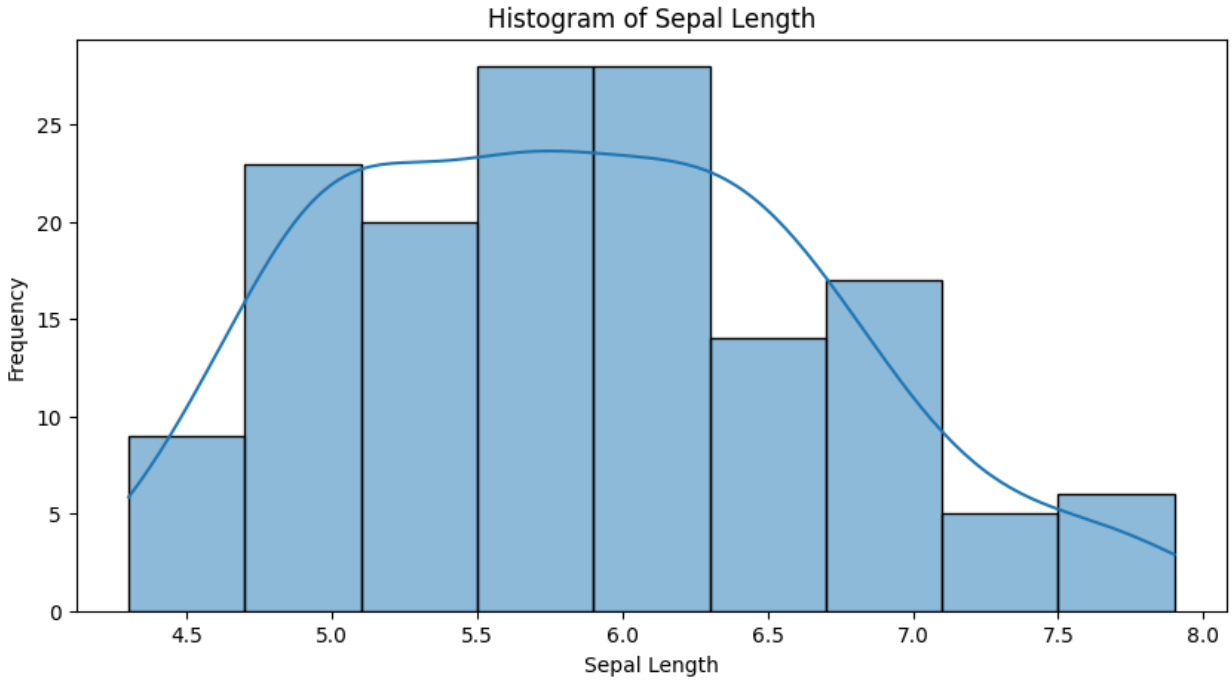
This histogram provides insights into the distribution and frequency of sepal lengths in the Iris dataset, helping to identify patterns such as skewness or modality, enhanced by the KDE overlay.

Code :

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

# Load dataset
df = sns.load_dataset('iris')
```

```
plt.figure(figsize=(10, 5))
sns.histplot(df['sepal_length'], kde=True)
plt.title('Histogram of Sepal Length')
plt.xlabel('Sepal Length')
plt.ylabel('Frequency')
plt.show()
```



Bar Chart:

- **Purpose**: This code creates a bar chart to visualize the count of each species in the Iris dataset.

- **Data**: It uses the `species` column from the DataFrame `df`.

- **Plotting**:

  - `plt.figure(figsize=(10, 5))` sets the figure size for better visibility.

  - `sns.countplot(x='species', data=df)` generates the bar chart, automatically counting the occurrences of each species and displaying them on the x-axis.

- **Labels**:

- `plt.title('Count of Each Species')` adds a title to the chart.

- `plt.xlabel('Species')` and `plt.ylabel('Count')` label the x-axis and y-axis, respectively.

- **Display**: Finally, `plt.show()` renders the plot.

Summary:

This bar chart effectively visualizes the distribution of different species in the dataset, allowing for easy comparison of their counts.

Code:

#Bar chart

plt.figure(figsize=(10, 5))

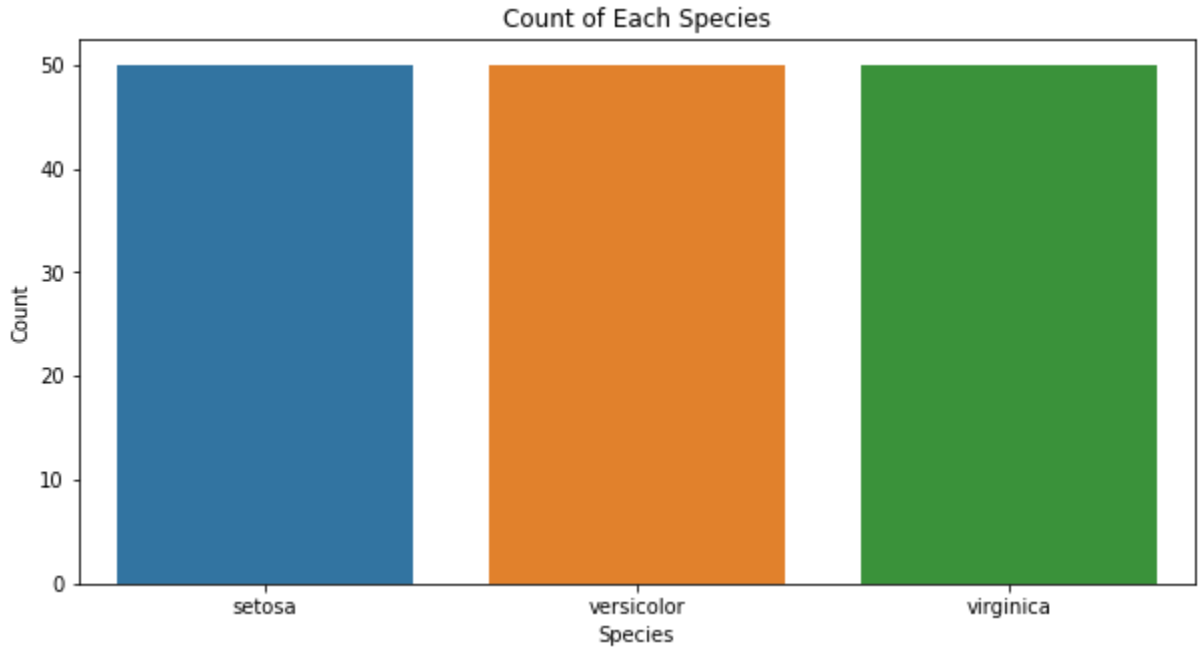
sns.countplot(x='species', data=df)

plt.title('Count of Each Species')

plt.xlabel('Species')

plt.ylabel('Count')

plt.show()



Line chart:

1. **Data Preparation**: A DataFrame named `time_data` is created, containing two columns: `day` (ranging from 1 to 10) and `sepal_length`, which takes the first 10 entries from another DataFrame `df`.

2. **Plotting**: A figure of size 10x5 inches is generated, and a line plot is created using Seaborn's `lineplot` function. The x-axis represents the day, while the y-axis shows the corresponding sepal lengths.

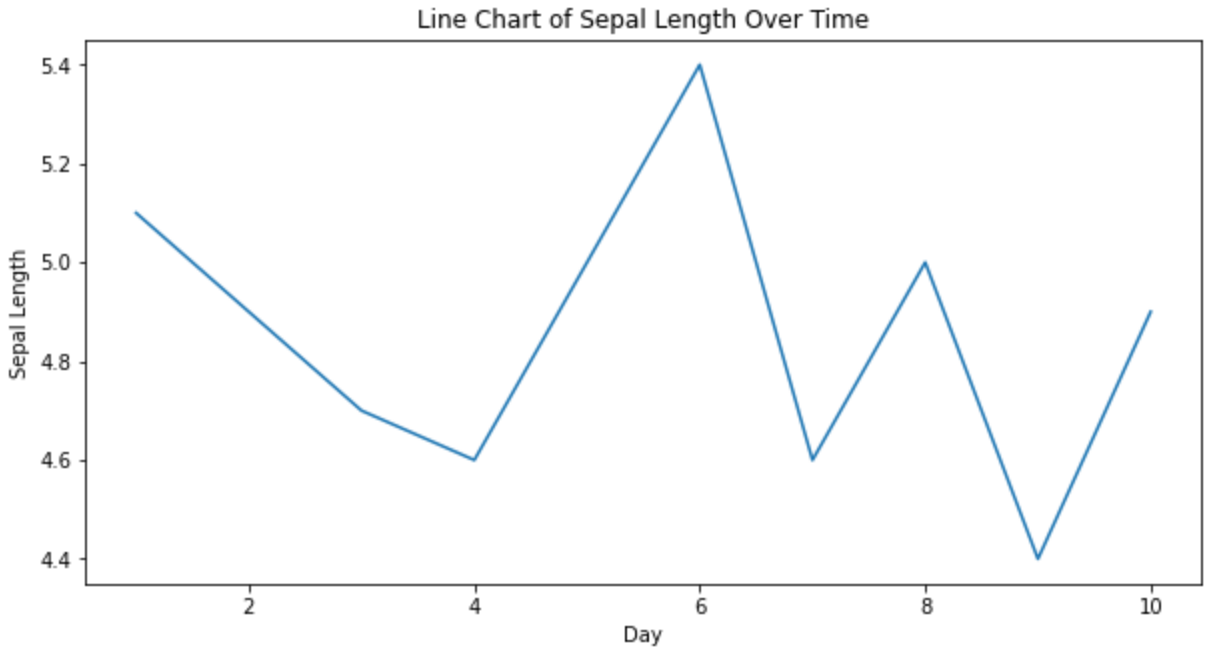
3. **Labels and Title**: The plot is titled "Line Chart of Sepal Length Over Time," with appropriate labels for both axes.

4. **Display**: Finally, the chart is displayed using `plt.show()`.

This visualization helps in understanding trends in sepal length over the specified days.

Code:

```
time_data = pd.DataFrame({
    'day': range(1, 11),
    'sepal_length': df['sepal_length'][:10]
})
plt.figure(figsize=(10, 5))
sns.lineplot(x='day', y='sepal_length', data=time_data)
plt.title('Line Chart of Sepal Length Over Time')
plt.xlabel('Day')
plt.ylabel('Sepal Length')
plt.show()
```



1. **Figure Setup**: A figure with specified size (10x5 inches) is created, and an axes object `ax1` is initialized.

2. **Bar Chart**: A bar chart using Seaborn's `countplot` displays the count of each species in the DataFrame `df`. The bars are blue (`color='b'`), and the y-axis is labeled "Count" in blue.

3. **Twin Axis for Line Chart**: A second y-axis (`ax2`) is created using `twinx()` to overlay a line chart. This line chart shows the average sepal length for each species using `pointplot`. The line is red (`color='r'`), and the y-axis is labeled "Average Sepal Length" in red.

4. **Title**: The plot is titled "Bar and Line Chart of Species Count and Sepal Length."

5. **Display**: Finally, `plt.show()` is called to render the combined visualization.

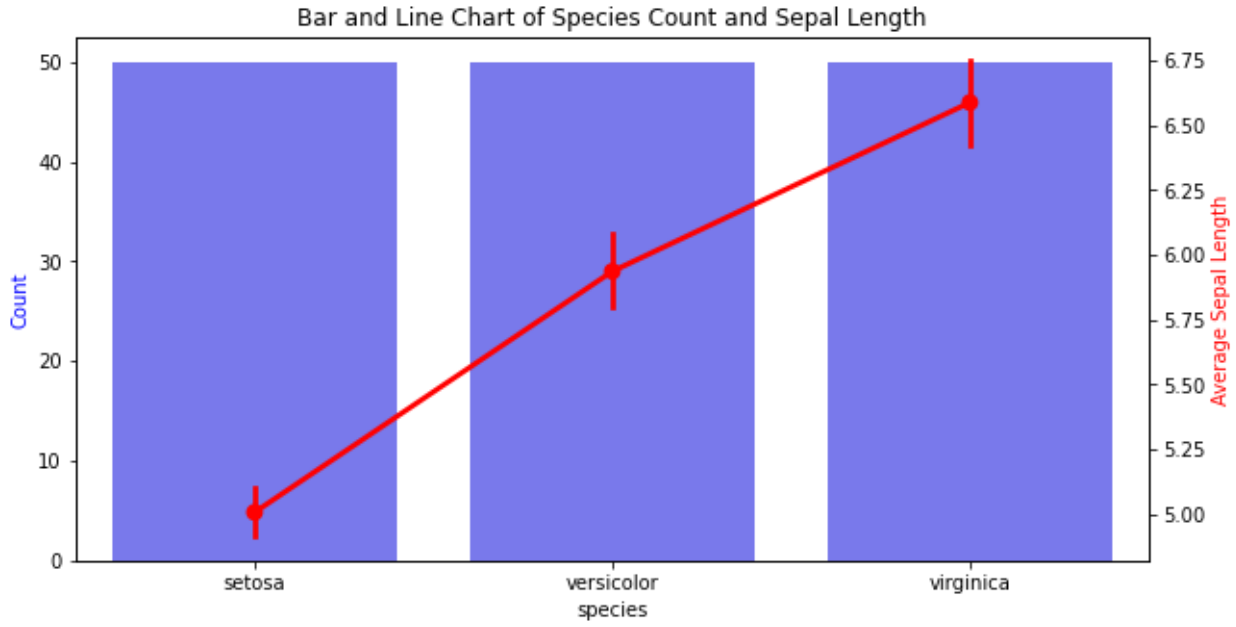
This approach allows for an easy comparison between the count of species and their average sepal lengths on the same graph.

Code:

```
fig, ax1 = plt.subplots(figsize=(10, 5))
```

```
# Bar chart for count of species
sns.countplot(x='species', data=df, ax=ax1, alpha=0.6, color='b')
ax1.set_ylabel('Count', color='b')

# Line chart for average sepal length of species
ax2 = ax1.twinx()
sns.pointplot(x='species', y='sepal_length', data=df, ax=ax2, color='r', markers='o', linestyles='-')
ax2.set_ylabel('Average Sepal Length', color='r')
plt.title('Bar and Line Chart of Species Count and Sepal Length')
plt.show()
```



Pie chart

1. **Species Count Calculation**: The `value_counts()` method is used to count the occurrences of each species in the `species` column of `df`. The result is stored in `species_count`.

2. **Printing Counts**: The counts for each species are printed to the console for reference.

3. **Figure Setup**: A figure with a size of 10x5 inches is created for the pie chart.

4. **Pie Chart Creation**: The `plt.pie()` function is used to generate the pie chart, where:
   - `species_count` provides the sizes of the slices.
   - `labels=species_count.index` adds the species names as labels.
   - `autopct='%1.1f%%'` formats the percentage display on each slice.
   - `startangle=140` rotates the start of the pie chart for better visual appeal.

5. **Title**: The pie chart is titled "Pie Chart of Species Distribution."

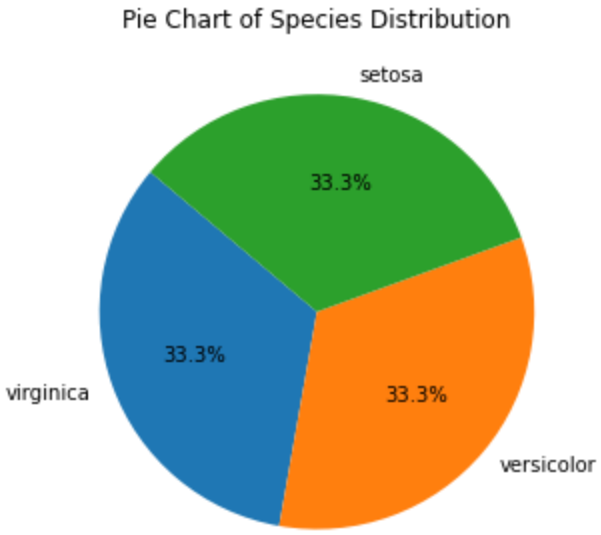
6. **Display**: Finally, `plt.show()` is called to render the pie chart.

This visualization helps to understand the proportion of each species in the dataset, making it easier to see which species are more prevalent.

Code:

```
species_count = df['species'].value_counts()
print(species_count)
plt.figure(figsize=(10, 5))
plt.pie(species_count, labels=species_count.index, autopct='%1.1f%%', startangle=140)
plt.title('Pie Chart of Species Distribution')
plt.show()
```

```
virginica     50
versicolor    50
setosa        50
Name: species, dtype: int64
```



Heat Map

1. **Figure Setup**: A figure is created with a size of 10x5 inches.

2. **Correlation Calculation**: The `corr()` method is called on `df` to compute the correlation coefficients between the numerical columns.

3. **Heatmap Creation**: The `sns.heatmap()` function from Seaborn is used to create the heatmap, with the following features:

   - `annot=True` displays the correlation coefficient values on the heatmap.

   - `cmap='coolwarm'` sets the color palette, with cool colors for negative correlations and warm colors for positive correlations.

   - `linewidths=0.5` adds lines between the cells for better readability.

4. **Title**: The heatmap is titled "Correlation Heatmap."

5. **Display**: Finally, `plt.show()` renders the heatmap.

This visualization helps to quickly identify relationships between variables, showing which features are positively or negatively correlated.

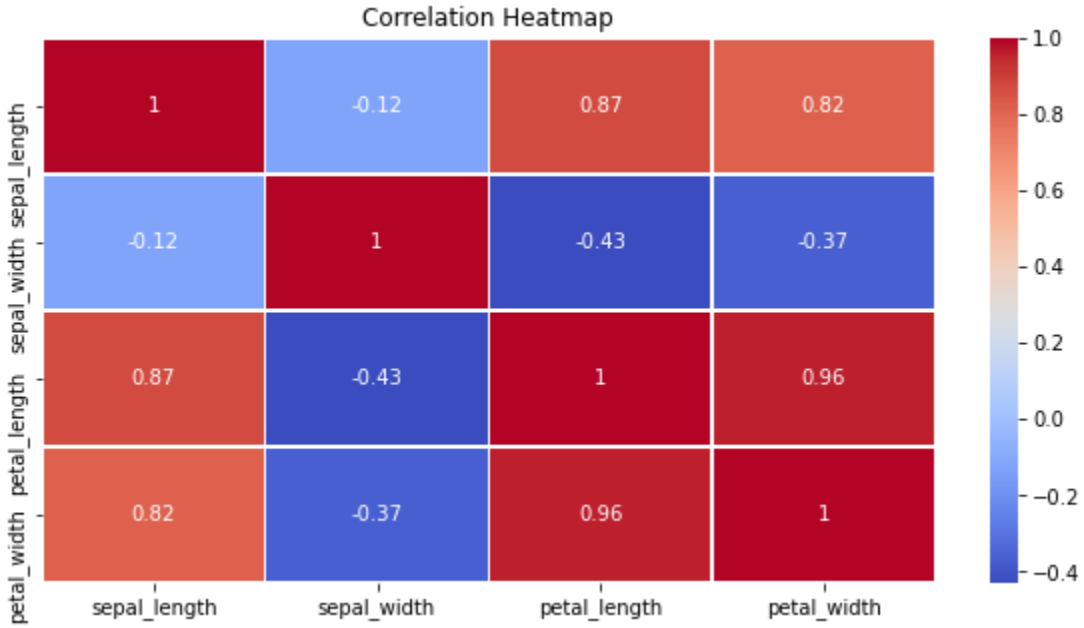
Code:

```
plt.figure(figsize=(10, 5))

sns.heatmap(df.corr(), annot=True, cmap='coolwarm', linewidths=0.5)

plt.title('Correlation Heatmap')

plt.show()
```



Pair plot.

1. **Pair Plot Creation**: The `sns.pairplot()` function from Seaborn generates a matrix of scatter plots for each pair of features in the DataFrame `df`. This includes diagonal plots showing the distribution of each individual feature.

2. **Title**: The title "Pair Plot of Iris Dataset" is set for the overall plot.

3. **Display**: Finally, `plt.show()` is called to render the pair plot.

The pair plot allows you to see how different features correlate with each other and can help identify patterns, clusters, or potential outliers within the dataset. It's particularly useful for exploring relationships in multi-dimensional data.

Code:

```
sns.pairplot(df)

plt.title('Pair Plot of Iris Dataset')

plt.show()
```