rnmrtl4hy

February 9, 2025

```
[ ]: import pandas as pd
     import numpy as np
     import seaborn as sns
     import matplotlib.pyplot as plt
     from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LogisticRegression
     from sklearn.preprocessing import LabelEncoder
     from sklearn.metrics import accuracy_score, classification_report,␣
      ↪confusion_matrix
```

```
[ ]: data=pd.read_csv("/content/IRIS.csv")
```

```
[ ]: data
```

```
[ ]:      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]
```

```
[ ]: data["species"].unique()
```

```
[ ]: array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

```
[ ]: data.describe()
```

```
[ ]:        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
```

[ ]:

```
data.info()
```

```
<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
```

[ ]:

```
missing_values = data.isnull().sum()
print("\nMissing Values:")
missing_values
```

```

Missing Values:
```

[ ]:

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

[ ]:

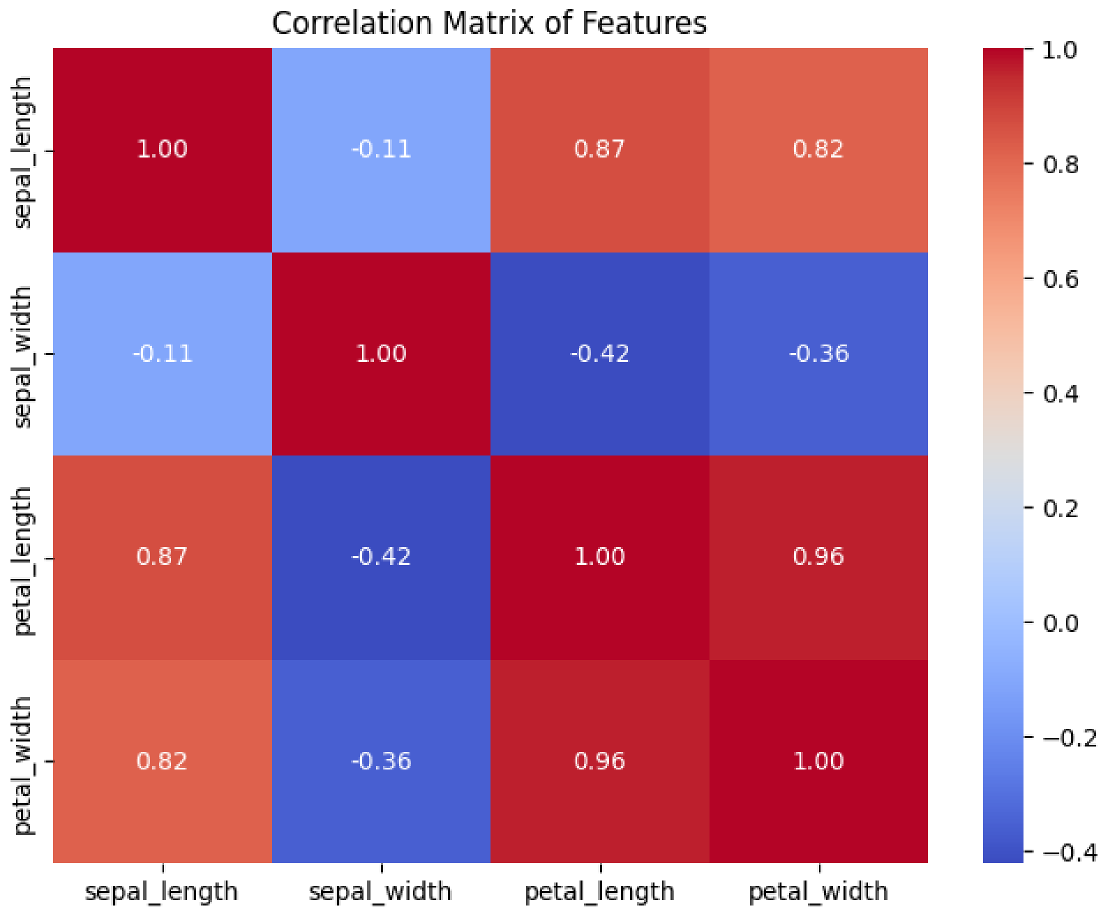
```
class_distribution = data['species'].value_counts()
print("Class Distribution:")
class_distribution
```

```
Class Distribution:
```

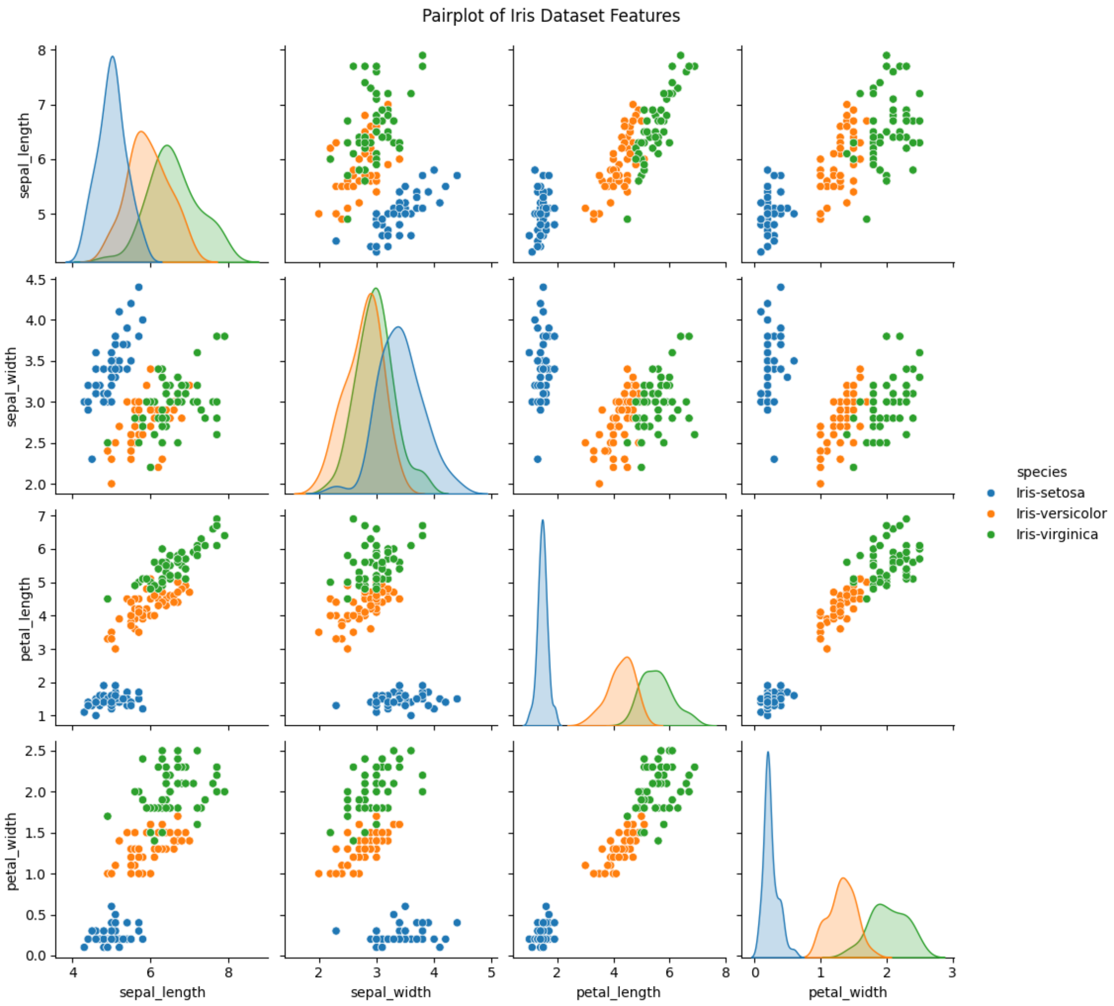
[ ]:

```
species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

```
[ ]: correlation_matrix = data[['sepal_length', 'sepal_width', 'petal_length',␣
      ↪'petal_width']].corr()
     plt.figure(figsize=(8, 6))
     sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
     plt.title("Correlation Matrix of Features")
     plt.show()
```



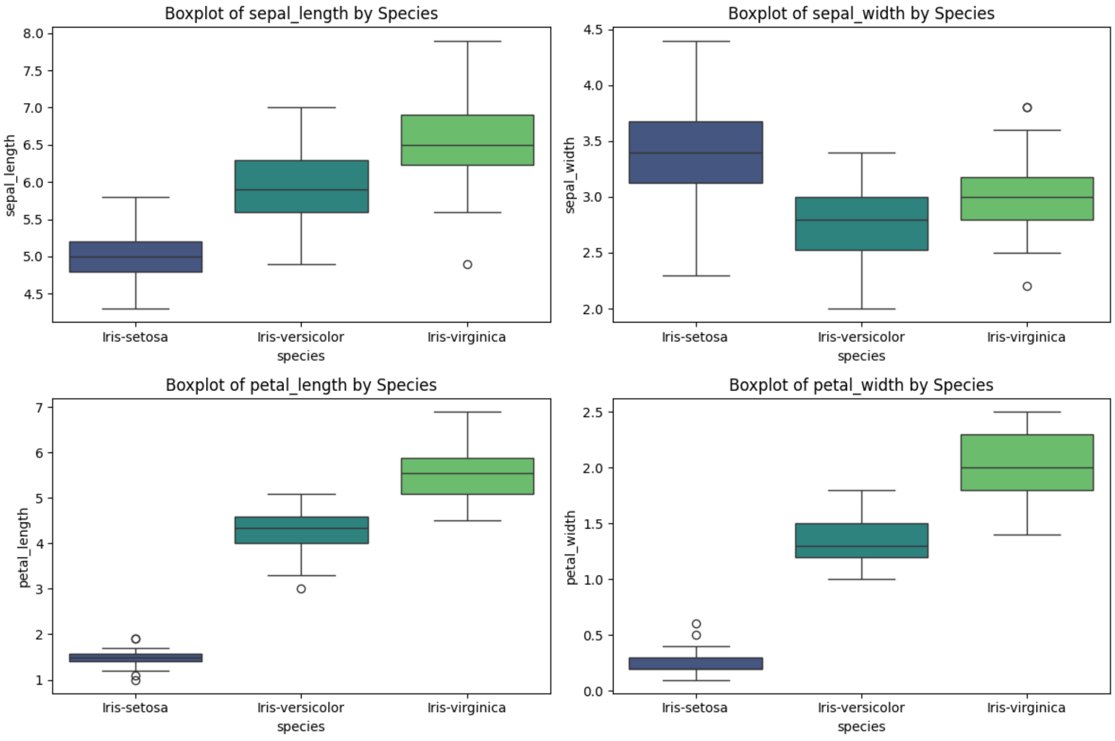
```
[ ]: try:
         sns.pairplot(data, vars=['sepal_length', 'sepal_width', 'petal_length',␣
      ↪'petal_width'], hue='species', diag_kind='kde')
         plt.suptitle("Pairplot of Iris Dataset Features", y=1.02)
         plt.show()
     except Exception as e:
         print(f"Error in pairplot: {e}")
```



```
[ ]: plt.figure(figsize=(12, 8))
     for i, feature in enumerate(['sepal_length', 'sepal_width', 'petal_length',␣
      ↪'petal_width'], 1):
         plt.subplot(2, 2, i)
         sns.boxplot(data=data, x='species', y=feature, palette='viridis')
         plt.title(f"Boxplot of {feature} by Species")
         plt.tight_layout()
     plt.show()
```

```
<ipython-input-32-4c22da8f037d>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(data=data, x='species', y=feature, palette='viridis')
```

```
<ipython-input-32-4c22da8f037d>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(data=data, x='species', y=feature, palette='viridis')
<ipython-input-32-4c22da8f037d>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(data=data, x='species', y=feature, palette='viridis')
<ipython-input-32-4c22da8f037d>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(data=data, x='species', y=feature, palette='viridis')
```



[ ]:

```
non_numeric_rows = data[
    ~data[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']].
 ↪applymap(lambda x: isinstance(x, (int, float))).all(axis=1)
]
print("\nNon-Numeric Rows (if any):")
print(non_numeric_rows)

unique_species = data['species'].unique()
species_types = data['species'].apply(type).unique()
print("\nUnique Species Values:")
print(unique_species)
print("Species Data Types:")
print(species_types)
```

```
Non-Numeric Rows (if any):
Empty DataFrame
Columns: [sepal_length, sepal_width, petal_length, petal_width, species]
Index: []

Unique Species Values:
['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

```
<ipython-input-34-963c0e6615d2>:2: FutureWarning: DataFrame.applymap has been
deprecated. Use DataFrame.map instead.
  ~data[['sepal_length', 'sepal_width', 'petal_length',
'petal_width']].applymap(lambda x: isinstance(x, (int, float))).all(axis=1)
```

[ ]:

```
label_encoder = LabelEncoder()
```

[ ]:

```
data['species_encoded'] = label_encoder.fit_transform(data['species'])
# Split the data into training and testing sets
X = data[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
y = data['species_encoded']
```

[ ]:

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.
 ↪2,random_state=42)
model=LogisticRegression()
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
y_pred
```

[ ]:

```
array([1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 2,
       0, 2, 2, 2, 2, 2, 0, 0])
```

[ ]:

```
y_pred_labels = label_encoder.inverse_transform(y_pred)
y_pred_labels
```

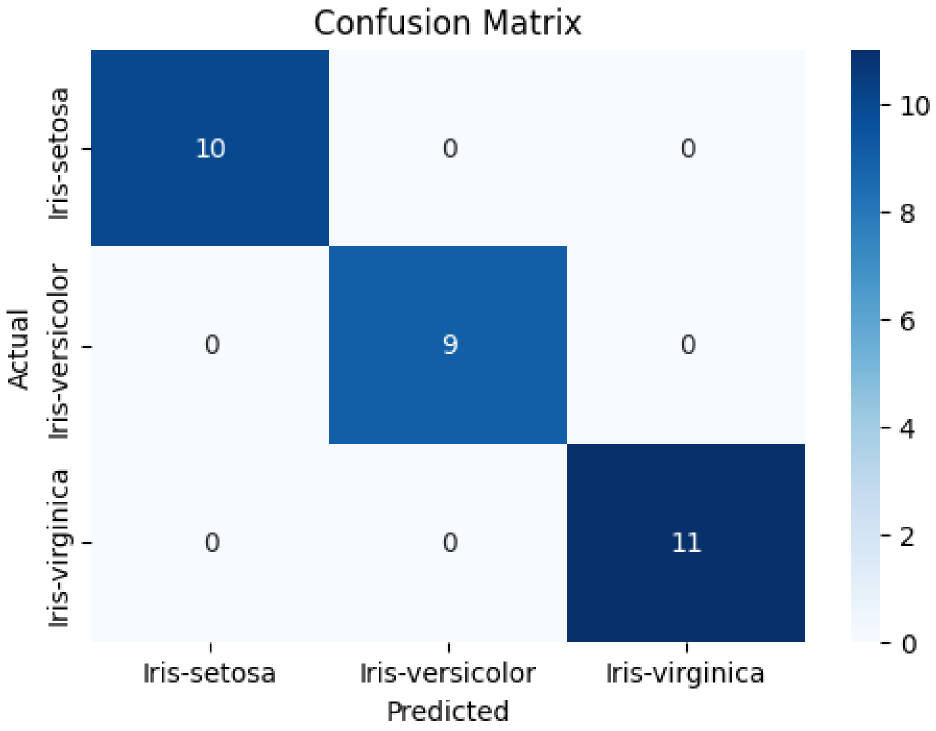
```
[ ]: array(['Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
            'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
            'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
            'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
            'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
            'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
            'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
            'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
            'Iris-virginica', 'Iris-setosa', 'Iris-setosa'], dtype=object)
```

```
[ ]: accuracy = accuracy_score(y_test, y_pred)
     conf_matrix = confusion_matrix(y_test, y_pred)
     report = classification_report(y_test, y_pred)
```

```
[ ]: print(f"Accuracy: {accuracy}")
     print(f"Confusion Matrix:\n{conf_matrix}")
     print(f"Classification Report:\n{report}")
```

```
Accuracy: 1.0
Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       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
```

```
[ ]: plt.figure(figsize=(6, 4))
     sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',␣
      ↪xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
     plt.xlabel('Predicted')
     plt.ylabel('Actual')
     plt.title('Confusion Matrix')
     plt.show()
```


Confusion Matrix
Actual
Iris-setosa
Iris-versicolor
Iris-virginica
10
0
0
0
9
0
0
0
11
Predicted
Iris-setosa
Iris-versicolor
Iris-virginica
10
8
6
4
2
0