assignment-3

September 14, 2023

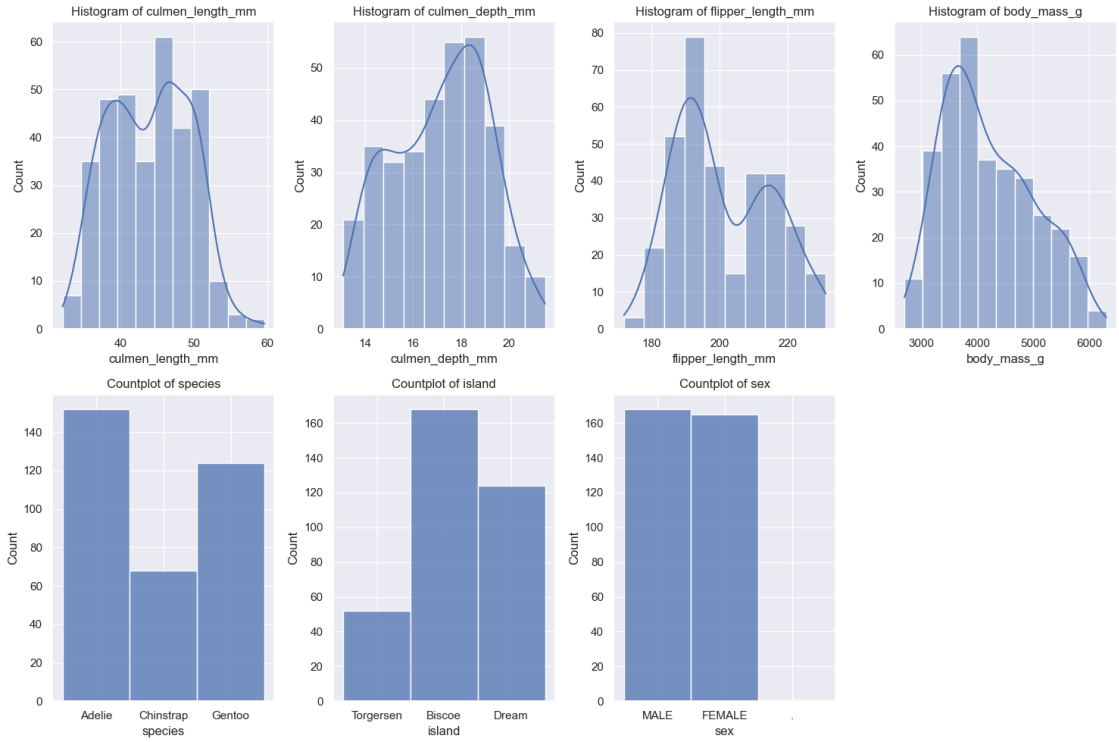
Name:- Anmol Vipin Devansh
Registration No.:- 21BCE10314
Assignment Number: 3

[1]:
```
# import necessary libraries
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.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import classification_report
from sklearn.cluster import KMeans
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
```

[2]:
```
# Task 1: Load the dataset
print("Task 1: Load the dataset")
missing_values = ["", "NA", "N/A", "NaN"]
penguins_data = pd.read_csv('penguinsize.csv', na_values=missing_values)
print(penguins_data)

# Task 2: Univariate Analysis
print("\n\n\n\nTask 2: Univariate Analysis")
sns.set(style="darkgrid")
plt.figure(figsize=(15, 10))

# Histograms for numeric attributes
numeric_attributes = ["culmen_length_mm", "culmen_depth_mm",␣
 ↪"flipper_length_mm", "body_mass_g"]
for i, col in enumerate(numeric_attributes, 1):
    plt.subplot(2, 4, i)
    sns.histplot(data=penguins_data, x=col, kde=True)
    plt.title(f'Histogram of {col}')

# Countplots for categorical attributes
categorical_attributes = ["species", "island", "sex"]
for i, col in enumerate(categorical_attributes, 1):
```

```
    plt.subplot(2, 4, i + 4)
    sns.histplot(data=penguins_data, x=col)
    plt.title(f'Countplot of {col}')

plt.tight_layout()
plt.show()
```

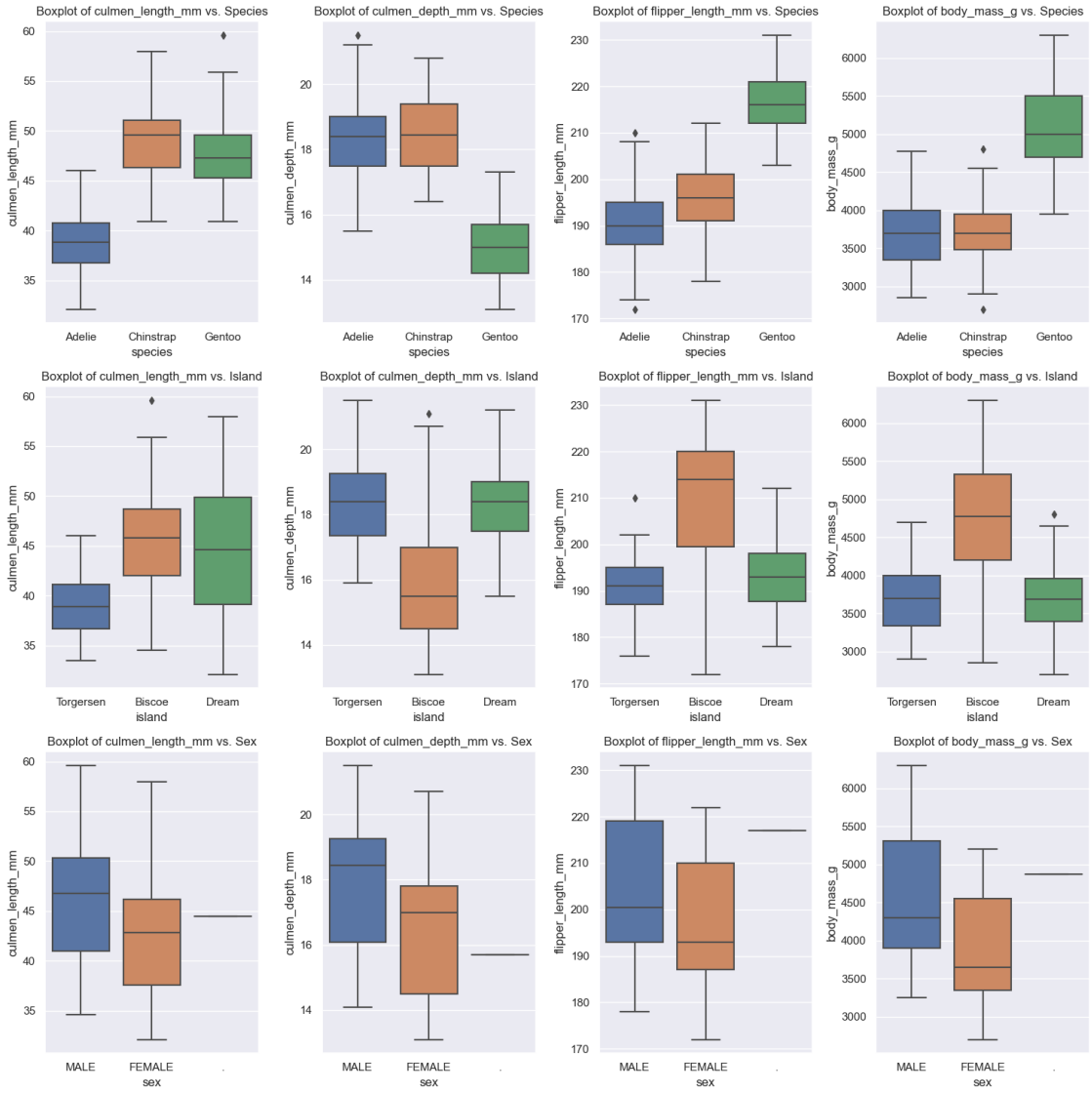
```
Task 1: Load the dataset
    species     island  culmen_length_mm  culmen_depth_mm  flipper_length_mm  \
0    Adelie  Torgersen              39.1             18.7              181.0
1    Adelie  Torgersen              39.5             17.4              186.0
2    Adelie  Torgersen              40.3             18.0              195.0
3    Adelie  Torgersen               NaN              NaN                NaN
4    Adelie  Torgersen              36.7             19.3              193.0
..      …        …                 …                …                 …
339  Gentoo     Biscoe               NaN              NaN                NaN
340  Gentoo     Biscoe              46.8             14.3              215.0
341  Gentoo     Biscoe              50.4             15.7              222.0
342  Gentoo     Biscoe              45.2             14.8              212.0
343  Gentoo     Biscoe              49.9             16.1              213.0

     body_mass_g     sex
0         3750.0    MALE
1         3800.0  FEMALE
2         3250.0  FEMALE
3            NaN     NaN
4         3450.0  FEMALE
..           …     …
339          NaN     NaN
340       4850.0  FEMALE
341       5750.0    MALE
342       5200.0  FEMALE
343       5400.0    MALE

[344 rows x 7 columns]



Task 2: Univariate Analysis
```



[3]:
```
# Task 3: Bivariate Analysis
print("Task 3: Bivariate Analysis\n\n\n\n")
# numeric_attributes are "culmen_length_mm", "culmen_depth_mm",␣
  ↪"flipper_length_mm", "body_mass_g"
# (species vs. numeric_attributes)
plt.figure(figsize=(15, 15))
for i, col in enumerate(numeric_attributes, 1):
    plt.subplot(3, 4, i)
    sns.boxplot(data=penguins_data, x="species", y=col)
    plt.title(f'Boxplot of {col} vs. Species')

# (island vs. numeric_attributes)
for i, col in enumerate(numeric_attributes, 1):
    plt.subplot(3, 4, i + 4)
    sns.boxplot(data=penguins_data, x="island", y=col)
    plt.title(f'Boxplot of {col} vs. Island')

# (sex vs. numeric_attributes)
for i, col in enumerate(numeric_attributes, 1):
    plt.subplot(3, 4, i + 8)
    sns.boxplot(data=penguins_data, x="sex", y=col)
    plt.title(f'Boxplot of {col} vs. Sex')
```

```
plt.tight_layout()
plt.show()
```

```
Task 3: Bivariate Analysis
```

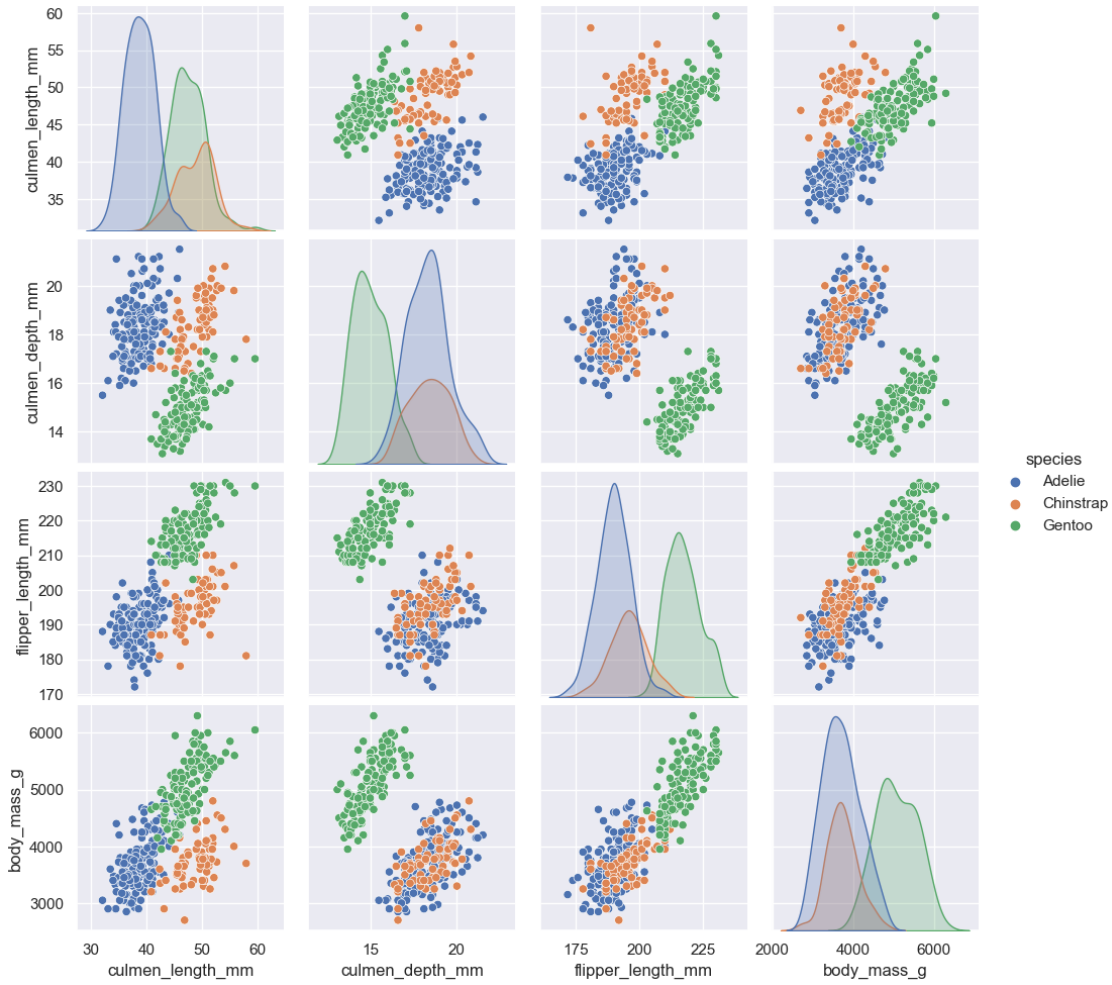


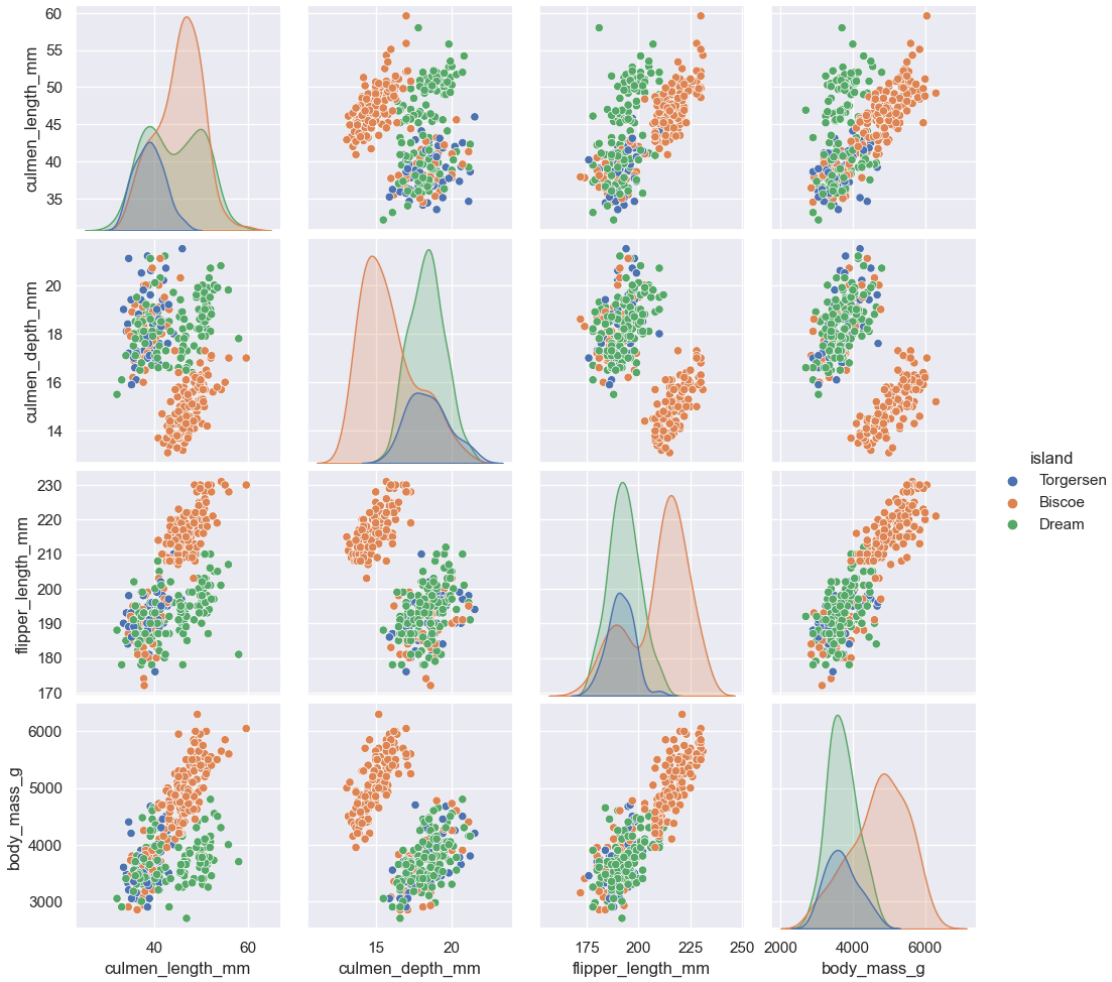
[4]:
```
# Task 4: Multivariate Analysis
print("Task 4: Multivariate Analysis\n\n\n\n")

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

```
sns.pairplot(data=penguins_data, hue="species", diag_kind="kde")
plt.show()

sns.pairplot(data=penguins_data, hue="island", diag_kind="kde")
plt.show()

sns.pairplot(data=penguins_data, hue="sex", diag_kind="kde")
plt.show()
```

Task 4: Multivariate Analysis

<Figure size 500x500 with 0 Axes>




culmen_length_mm
culmen_depth_mm
flipper_length_mm
body_mass_g
island
Torgersen
Biscoe
Dream

[5]:
```
# Task 5: Descriptive Statistics
print("Task 5: Descriptive statistics of dataset:\n\n\n\n")
descriptive_stats = penguins_data.describe()
print(descriptive_stats)
```

```
Task 5: Descriptive statistics of dataset:
```

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
count	342.000000	342.000000	342.000000	342.000000
mean	43.921930	17.151170	200.915205	4201.754386
std	5.459584	1.974793	14.061714	801.954536
min	32.100000	13.100000	172.000000	2700.000000
25%	39.225000	15.600000	190.000000	3550.000000
50%	44.450000	17.300000	197.000000	4050.000000

```
75%          48.500000        18.700000         213.000000  4750.000000
max          59.600000        21.500000         231.000000  6300.000000
```

[6]:
```
# Task 6: Handle Missing Values
print("Task 6: Handle Missing Values\n\n\n\n")
missing_values = penguins_data.isnull().sum()
# drop rows with missing value
print("Dataset after dropping rows with missing values")
penguins_data.dropna()
# Load the dataset (replace 'your_dataset.csv' with the actual file path)
```

```
Task 6: Handle Missing Values
```

```
Dataset after dropping rows with missing values
```

[6]:
```
     species     island  culmen_length_mm  culmen_depth_mm  flipper_length_mm  \
0     Adelie  Torgersen              39.1             18.7              181.0   
1     Adelie  Torgersen              39.5             17.4              186.0   
2     Adelie  Torgersen              40.3             18.0              195.0   
4     Adelie  Torgersen              36.7             19.3              193.0   
5     Adelie  Torgersen              39.3             20.6              190.0   
..       ...        ...               ...              ...                ...   
338   Gentoo     Biscoe              47.2             13.7              214.0   
340   Gentoo     Biscoe              46.8             14.3              215.0   
341   Gentoo     Biscoe              50.4             15.7              222.0   
342   Gentoo     Biscoe              45.2             14.8              212.0   
343   Gentoo     Biscoe              49.9             16.1              213.0   

     body_mass_g     sex  
0         3750.0    MALE  
1         3800.0  FEMALE  
2         3250.0  FEMALE  
4         3450.0  FEMALE  
5         3650.0    MALE  
..           ...     ...  
338       4925.0  FEMALE  
340       4850.0  FEMALE  
341       5750.0    MALE  
342       5200.0  FEMALE  
343       5400.0    MALE  

[334 rows x 7 columns]
```

[15]:
```
#Task 7: Find the outliers and replace them outliers
import pandas as pd
import numpy as np

# Load your dataset into a DataFrame
penguins_data = pd.read_csv('penguinsize.csv')

# Define a function to replace outliers using the IQR method
def replace_outliers_iqr(df, column_name, multiplier=1.5):
    Q1 = df[column_name].quantile(0.25)
    Q3 = df[column_name].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - multiplier * IQR
    upper_bound = Q3 + multiplier * IQR

    df[column_name] = np.where(df[column_name] < lower_bound, lower_bound,␣
 ↪df[column_name])
    df[column_name] = np.where(df[column_name] > upper_bound, upper_bound,␣
 ↪df[column_name])

# Specify the columns to check for outliers
columns_to_check = ['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm',␣
 ↪'body_mass_g']

# Replace outliers in the specified columns
for column in columns_to_check:
    replace_outliers_iqr(penguins_data, column)

# Display the data after replacing outliers
print("Data After Replacing Outliers:")
print(penguins_data.head())
```

```
Data After Replacing Outliers:
  species     island  culmen_length_mm  culmen_depth_mm  flipper_length_mm
0  Adelie  Torgersen              39.1             18.7              181.0  \
1  Adelie  Torgersen              39.5             17.4              186.0
2  Adelie  Torgersen              40.3             18.0              195.0
3  Adelie  Torgersen               NaN              NaN                NaN
4  Adelie  Torgersen              36.7             19.3              193.0

   body_mass_g     sex
0       3750.0    MALE
1       3800.0  FEMALE
2       3250.0  FEMALE
3          NaN     NaN
4       3450.0  FEMALE
```

[8]:
```
#Task 8: Check the correlation of independent variables with the target
import pandas as pd

# Load your dataset into a DataFrame
df = pd.read_csv('penguinsize.csv')

# Define the target variable and numerical features of interest
target_variable = 'body_mass_g'  # Replace with your actual target variable
numerical_features = ['culmen_length_mm', 'culmen_depth_mm',␣
 ↪'flipper_length_mm']  # Replace with your actual numerical features

# Calculate Pearson correlation coefficients
correlations = penguins_data[numerical_features].corrwith(df[target_variable])

# Print the correlations
print("Correlation with the target variable:")
print(correlations)
```

```
Correlation with the target variable:
culmen_length_mm     0.595110
culmen_depth_mm     -0.471916
flipper_length_mm    0.871202
dtype: float64
```

[9]:
```
#Task 9: Check for Categorical columns and perform encoding

import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Load your dataset into a DataFrame
penguins_data = pd.read_csv('penguinsize.csv')

# Identify categorical columns
categorical_columns = penguins_data.select_dtypes(include=['object',␣
 ↪'category']).columns.tolist()

# Perform encoding based on the type of categorical variable
for column in categorical_columns:
    unique_values = penguins_data[column].nunique()

    # If the number of unique values is low (indicating ordinal categorical),␣
 ↪use label encoding
    if unique_values <= 10:
        label_encoder = LabelEncoder()
        penguins_data[column] = label_encoder.
 ↪fit_transform(penguins_data[column])
    else:
```

```
        # Use one-hot encoding for nominal categorical variables
        penguins_data = pd.get_dummies(penguins_data, columns=[column],␣
↪drop_first=True)

# Display the data after encoding
print(penguins_data)
```

```
     species  island  culmen_length_mm  culmen_depth_mm  flipper_length_mm
0          0       2               39.1             18.7              181.0  \
1          0       2               39.5             17.4              186.0
2          0       2               40.3             18.0              195.0
3          0       2                NaN              NaN                NaN
4          0       2               36.7             19.3              193.0
..       …     …                 …                …                 …
339        2       0                NaN              NaN                NaN
340        2       0               46.8             14.3              215.0
341        2       0               50.4             15.7              222.0
342        2       0               45.2             14.8              212.0
343        2       0               49.9             16.1              213.0

     body_mass_g  sex
0         3750.0    2
1         3800.0    1
2         3250.0    1
3            NaN    3
4         3450.0    1
..           …  …
339          NaN    3
340       4850.0    1
341       5750.0    2
342       5200.0    1
343       5400.0    2

[344 rows x 7 columns]
```

[10]:
```
#Task 10: Split the data into dependent and independent variables.

import pandas as pd

# Load your dataset into a DataFrame
penguins_data = pd.read_csv('penguinsize.csv')

# Define the dependent variable (target) and independent variables (features)
# Replace 'target_column' with the name of your target variable
target_column = 'species'  # Example target variable name

# Create a DataFrame for the dependent variable (target)
```

```
y = penguins_data[target_column]

# Create a DataFrame for the independent variables (features) by dropping the␣
 ↪target column
X = penguins_data.drop(columns=[target_column])

# Display the data after splitting
print("Dependent Variable (Target - y):")
print(y.head())

print("\nIndependent Variables (Features - X):")
print(X.head())
```

```
Dependent Variable (Target - y):
0    Adelie
1    Adelie
2    Adelie
3    Adelie
4    Adelie
Name: species, dtype: object

Independent Variables (Features - X):
      island  culmen_length_mm  culmen_depth_mm  flipper_length_mm
0  Torgersen              39.1             18.7              181.0  \
1  Torgersen              39.5             17.4              186.0
2  Torgersen              40.3             18.0              195.0
3  Torgersen               NaN              NaN                NaN
4  Torgersen              36.7             19.3              193.0

   body_mass_g     sex
0       3750.0    MALE
1       3800.0  FEMALE
2       3250.0  FEMALE
3          NaN     NaN
4       3450.0  FEMALE
```

[11]:
```
# Task 11: Scaling the Data
import pandas as pd
from sklearn.preprocessing import StandardScaler, LabelEncoder

# Load your dataset into a DataFrame
penguins_data = pd.read_csv('penguinsize.csv')

# Identify categorical columns
categorical_columns = penguins_data.select_dtypes(include=['object',␣
 ↪'category']).columns.tolist()
```

```
# Perform encoding for categorical columns
label_encoders = {}
for column in categorical_columns:
    le = LabelEncoder()
    penguins_data[column] = le.fit_transform(penguins_data[column])
    label_encoders[column] = le

# Define the dependent variable (target) and independent variables (features)
target_column = 'species'  # Example target variable name

# Create a DataFrame for the dependent variable (target)
y = penguins_data[target_column]

# Create a DataFrame for the independent variables (features) by dropping the␣
 ↪target column
X = penguins_data.drop(columns=[target_column])

# Scale only the numeric columns using StandardScaler
numeric_columns = X.select_dtypes(include=[np.number]).columns.tolist()
scaler = StandardScaler()
X[numeric_columns] = scaler.fit_transform(X[numeric_columns])

# Display the data after all tasks, including scaling
print("Data After All Tasks:")
print(penguins_data.head())
```

```
Data After All Tasks:
   species  island  culmen_length_mm  culmen_depth_mm  flipper_length_mm
0        0       2              39.1             18.7              181.0  \
1        0       2              39.5             17.4              186.0
2        0       2              40.3             18.0              195.0
3        0       2               NaN              NaN                NaN
4        0       2              36.7             19.3              193.0

   body_mass_g  sex
0       3750.0    2
1       3800.0    1
2       3250.0    1
3          NaN    3
4       3450.0    1
```

[12]:
```
#Task 12: Split the data into training and testing
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets (adjust the test_size as needed)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

```
# Display the shapes of the resulting sets
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (275, 6)
X_test shape: (69, 6)
y_train shape: (275,)
y_test shape: (69,)
```

[13]:
```
#Task 13: check the training and testing data shape.

# Check the shapes of the training and testing data
print("Training Data Shapes:")
print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)

print("\nTesting Data Shapes:")
print("X_test shape:", X_test.shape)
print("y_test shape:", y_test.shape)
```

```
Training Data Shapes:
X_train shape: (275, 6)
y_train shape: (275,)

Testing Data Shapes:
X_test shape: (69, 6)
y_test shape: (69,)
```