assignment-3

September 14, 2023

[45]:
```
# Step 1: Load the dataset
df = pd.read_csv('penguins_size.csv')
df.head()
```

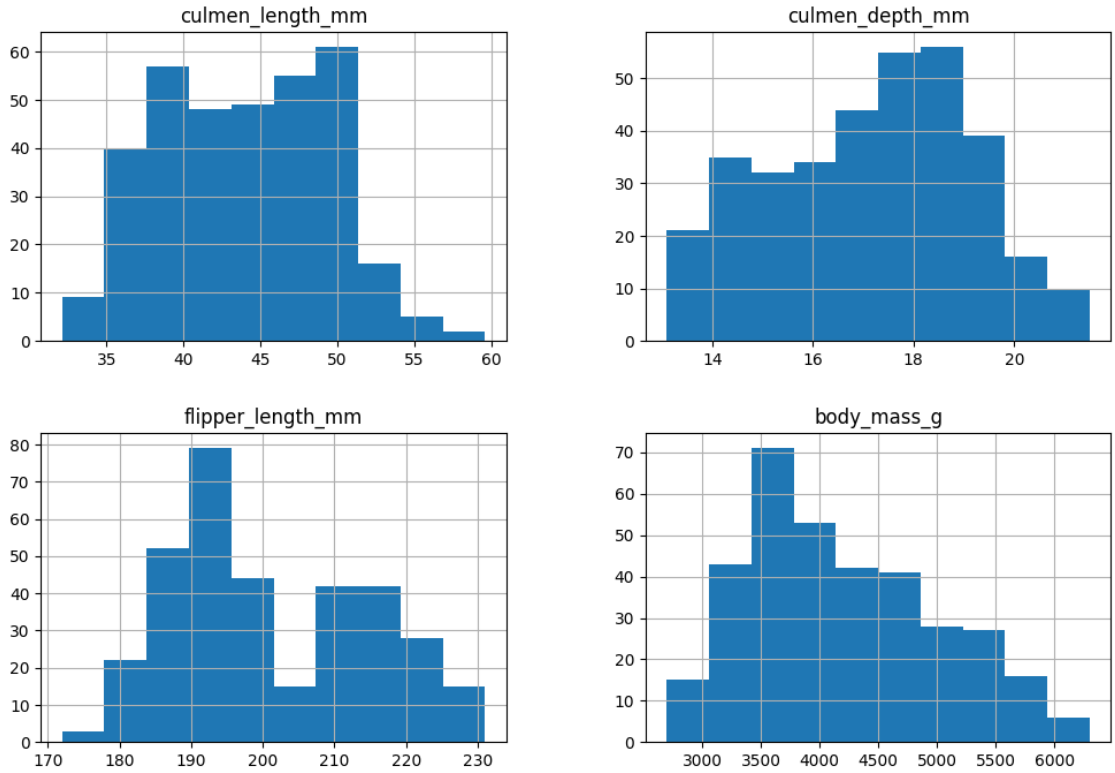
[45]:

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE

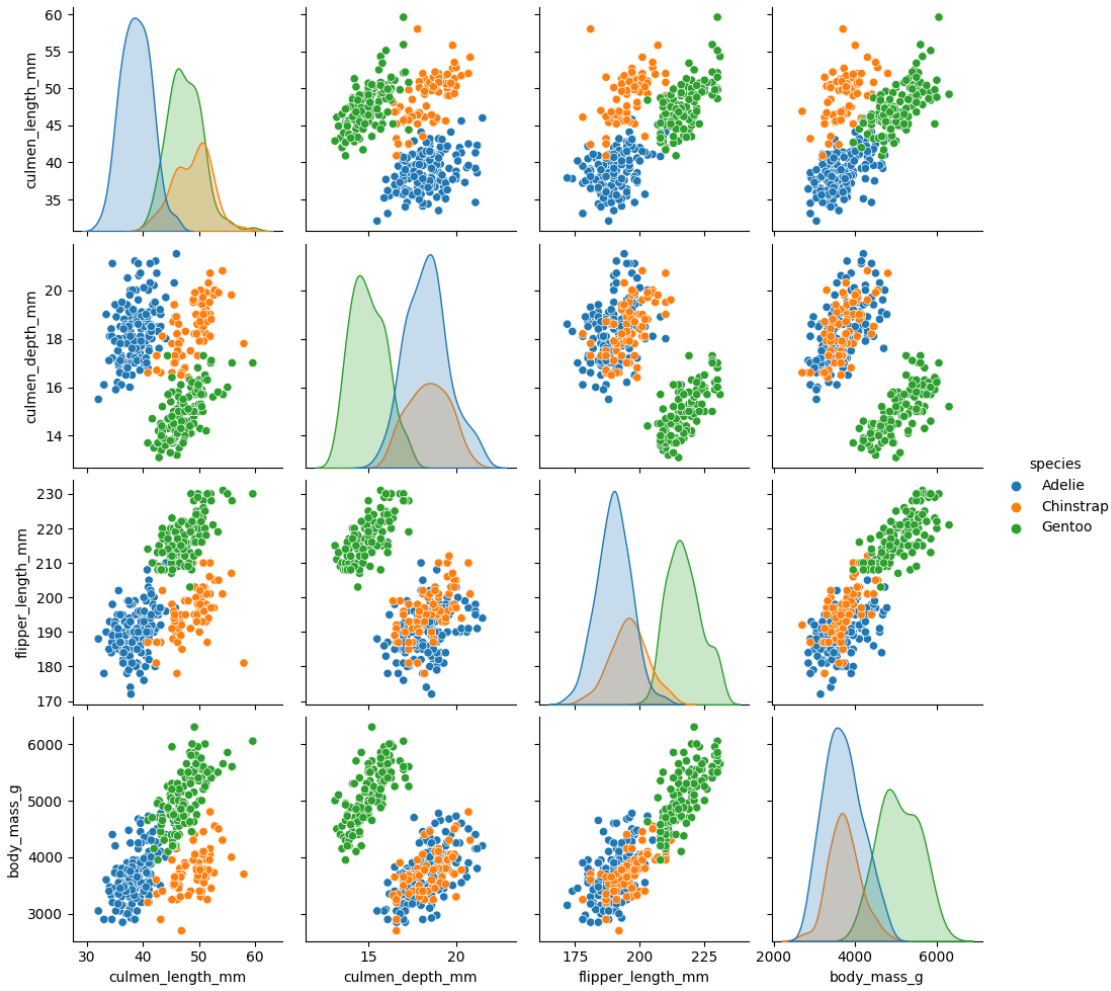
[46]:
```
import pandas as pd

# Load the dataset
data = pd.read_csv("penguins_size.csv")
```

[47]:
```
# Univariate Analysis
# Plot histograms for numeric features
data.hist(figsize=(12, 8))
plt.show()
```

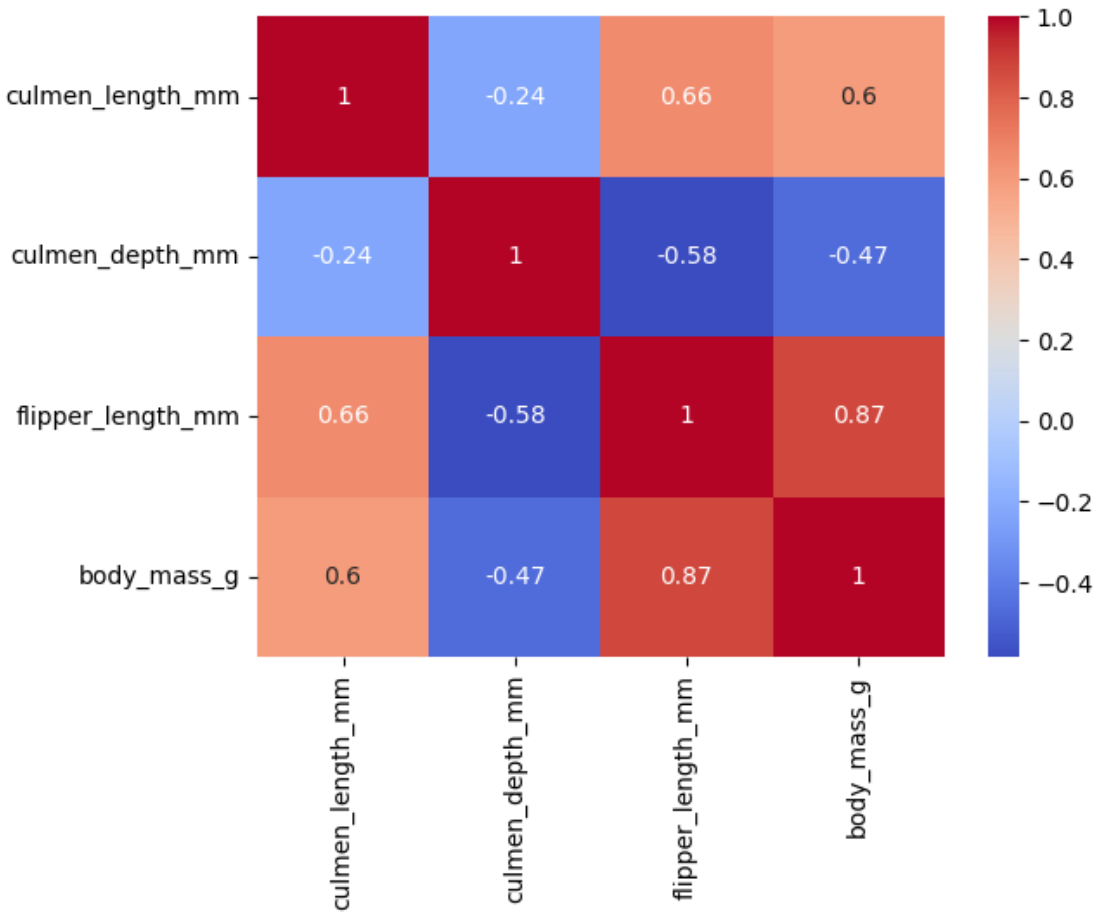


[49]:
```
# Bi-Variate Analysis
# Pairplot for numeric features colored by species
sns.pairplot(data, hue="species")
plt.show()
```



[50]:
```
# Multi-Variate Analysis
# Create a heatmap to visualize correlation between numeric features
correlation_matrix = data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.show()
```

```
<ipython-input-50-4b3654ef6637>:3: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  correlation_matrix = data.corr()
```



[51]:
```
# Descriptive statistics
descriptive_stats = data.describe()
print(descriptive_stats)
```

```
       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
```

[53]:
```
# Check for missing values
missing_values = data.isnull().sum()
print("Missing Values:\n", missing_values)
```

```
Missing Values:
```

```
 species               0
island                 0
culmen_length_mm       2
culmen_depth_mm        2
flipper_length_mm      2
body_mass_g            2
sex                   10
dtype: int64
```

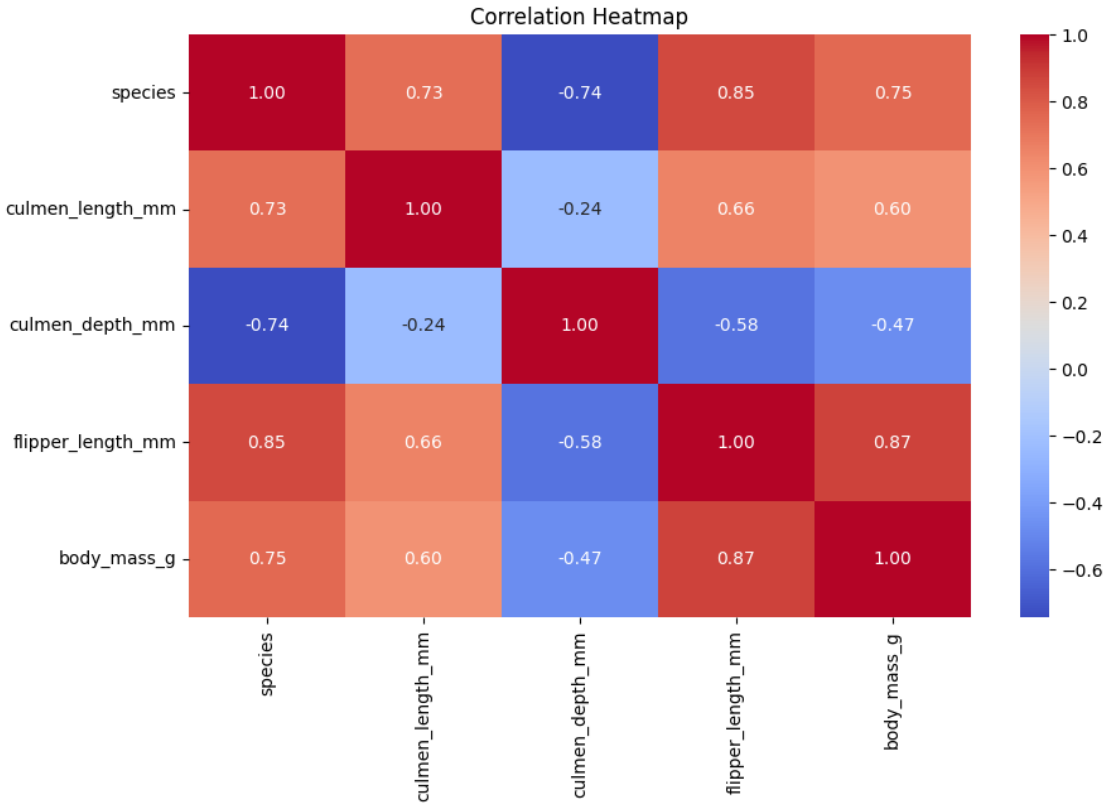
[54]:
```
# Outlier Detection and Replacement
numeric_columns = ["culmen_length_mm", "culmen_depth_mm", "flipper_length_mm",␣
 ↪"body_mass_g"]
z_scores = data[numeric_columns].apply(lambda x: (x - x.mean()) / x.std())
data = data[(z_scores.abs() < 3).all(axis=1)]
```

[55]:
```
# Checking Correlation with Target (assuming "species" is the target variable)
data["species"] = LabelEncoder().fit_transform(data["species"])
correlation_matrix = data.corr()

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

```
<ipython-input-55-6c36972c745f>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data["species"] = LabelEncoder().fit_transform(data["species"])
<ipython-input-55-6c36972c745f>:3: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  correlation_matrix = data.corr()
```



[58]:
```
# Task 8: Check for Categorical columns and perform encoding
# Encode categorical columns (e.g., 'Sex' and 'Island') using one-hot encoding
data_encoded = pd.get_dummies(data, columns=["sex", "island"], drop_first=True)
```

[59]:
```
# Task 9: Split the data into dependent and independent variables
X = data_encoded.drop(columns=["species"])
y = data_encoded["species"]
```

[60]:
```
# Task 10: Scaling the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

[61]:
```
#Task 11
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,␣
 ↪random_state=42)
```

[63]:
```
print("Training data shape:", X_train.shape, y_train.shape)
print("Testing data shape:", X_test.shape, y_test.shape)
```

```
Training data shape: (273, 8) (273,)
```

```
Testing data shape: (69, 8) (69,)
```