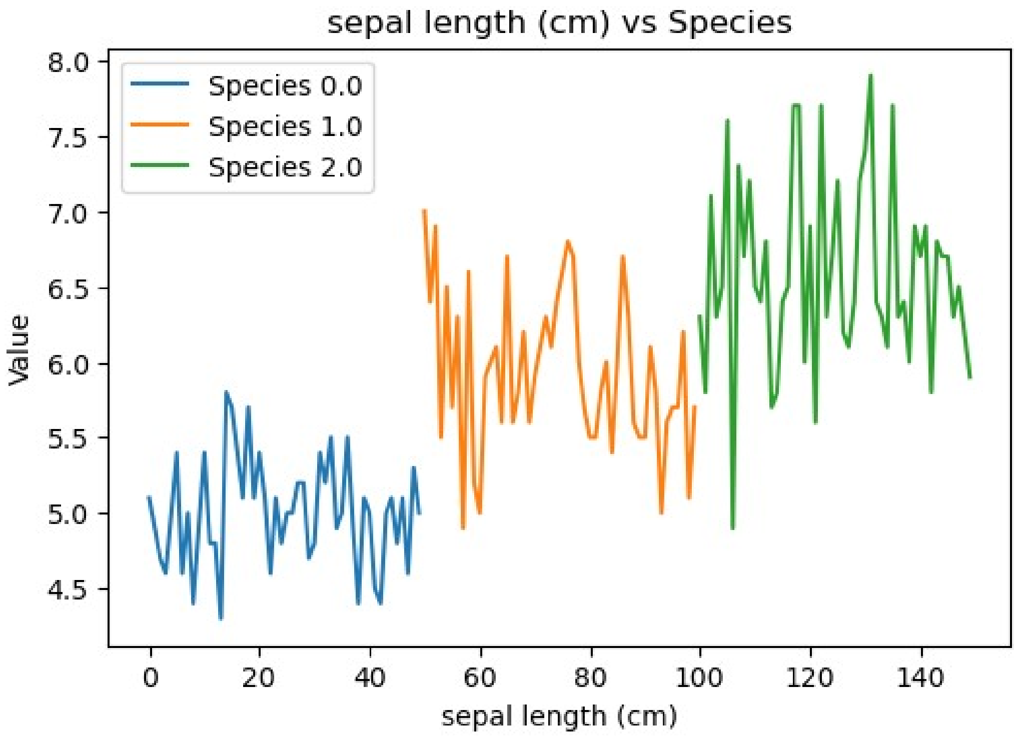
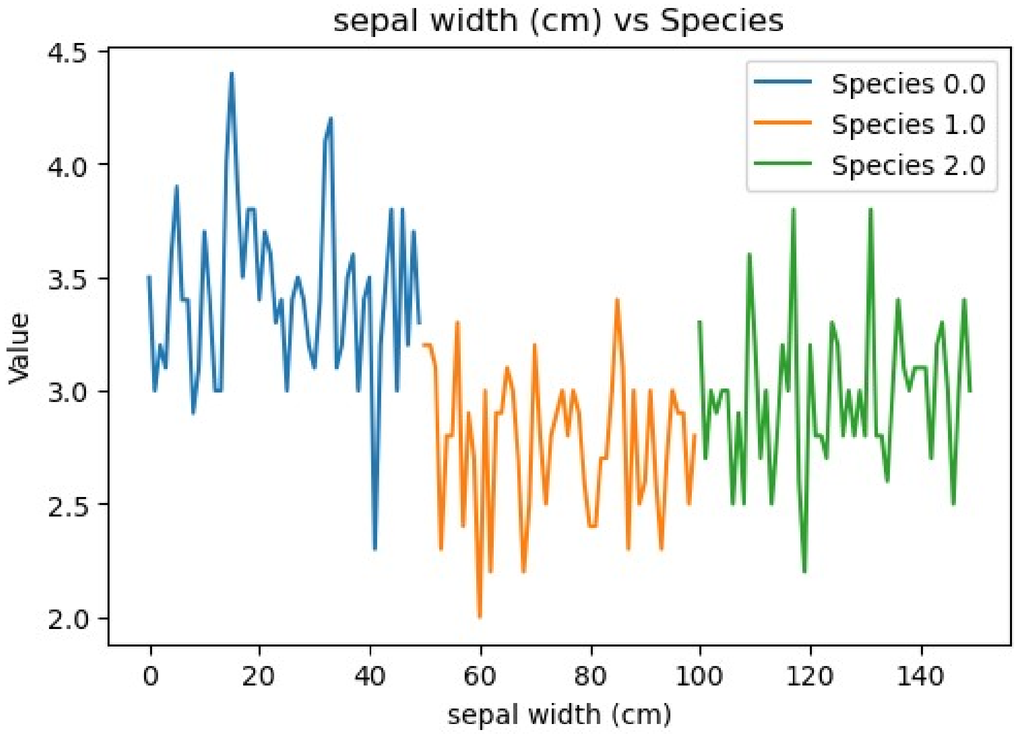
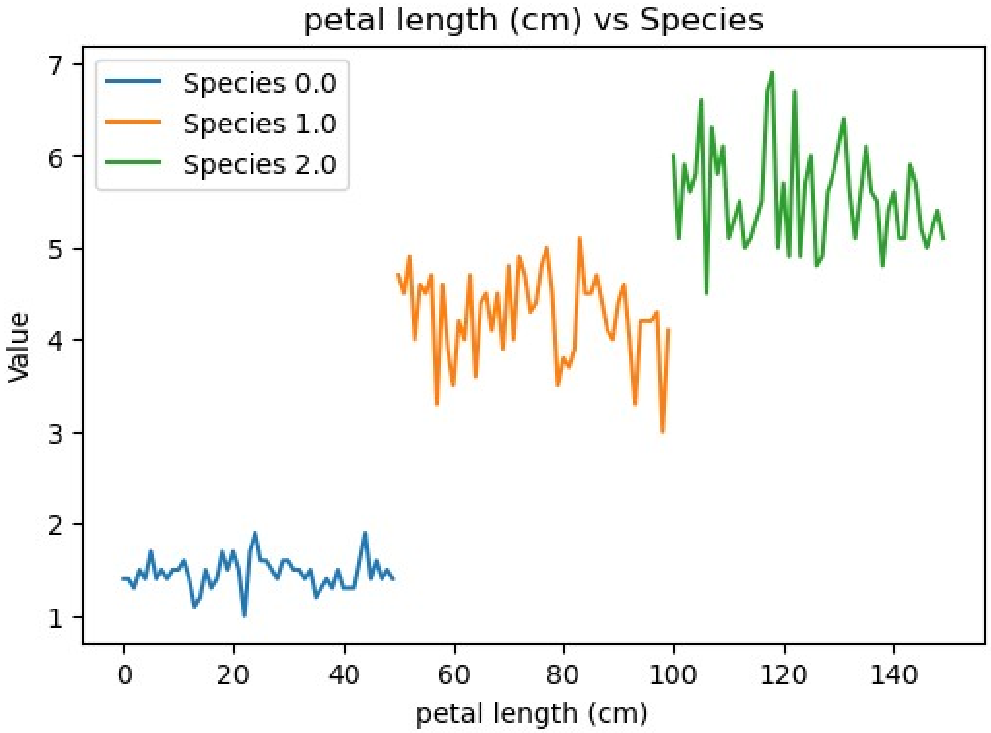
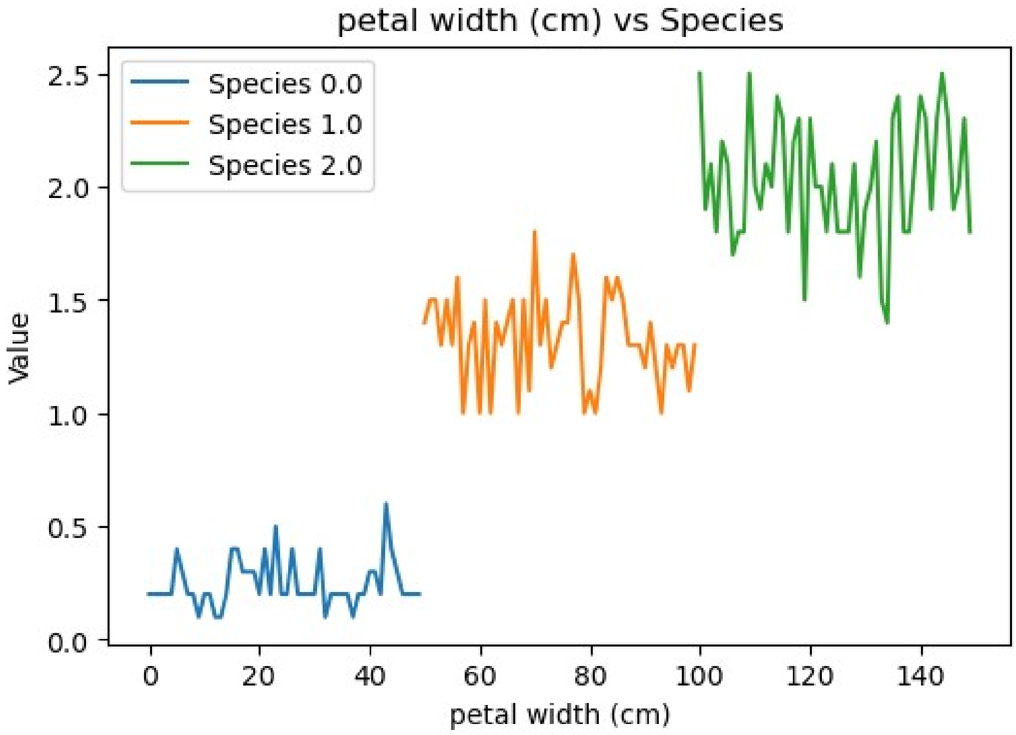
PART 1

```
# Task 1: Import necessary libraries
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris

# Task 2: Load the Iris dataset and convert it into a pandas DataFrame
iris = load_iris()
iris_df = pd.DataFrame(data=np.c_[iris['data'], iris['target']],
columns=iris['feature_names'] + ['species'])

# Task 3: Line Plot for each feature against the target variable
(species)
for feature in iris['feature_names']:
    plt.figure(figsize=(6, 4))
    for species in iris_df['species'].unique():
        plt.plot(iris_df[iris_df['species'] == species][feature],
label=f"Species {species}")
    plt.title(f"{feature} vs Species")
    plt.xlabel(feature)
    plt.ylabel("Value")
    plt.legend()
    plt.show()
```

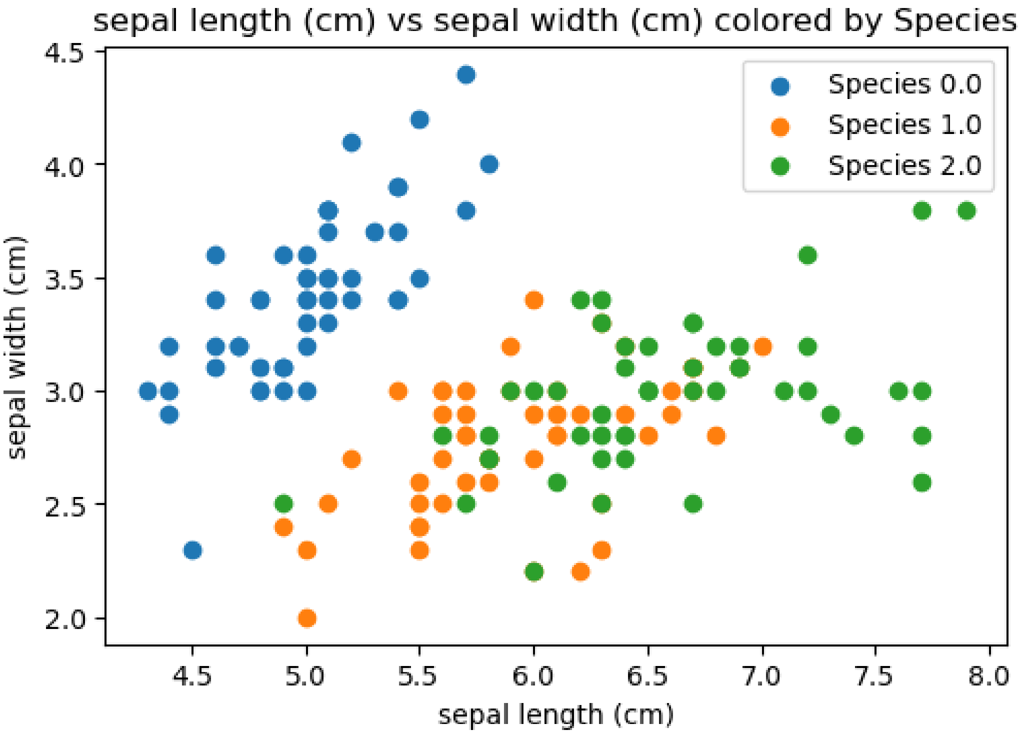


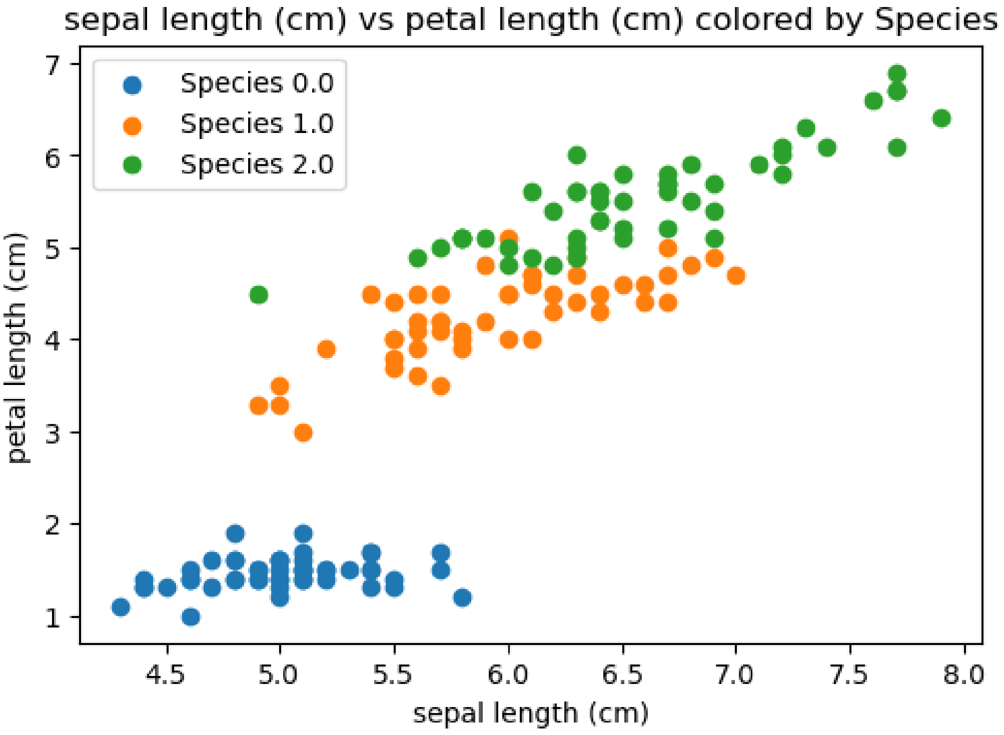

sepal width (cm) vs Species
Value
4.5
4.0
3.5
3.0
2.5
2.0
0
20
40
60
80
100
120
140
sepal width (cm)
Species 0.0
Species 1.0
Species 2.0

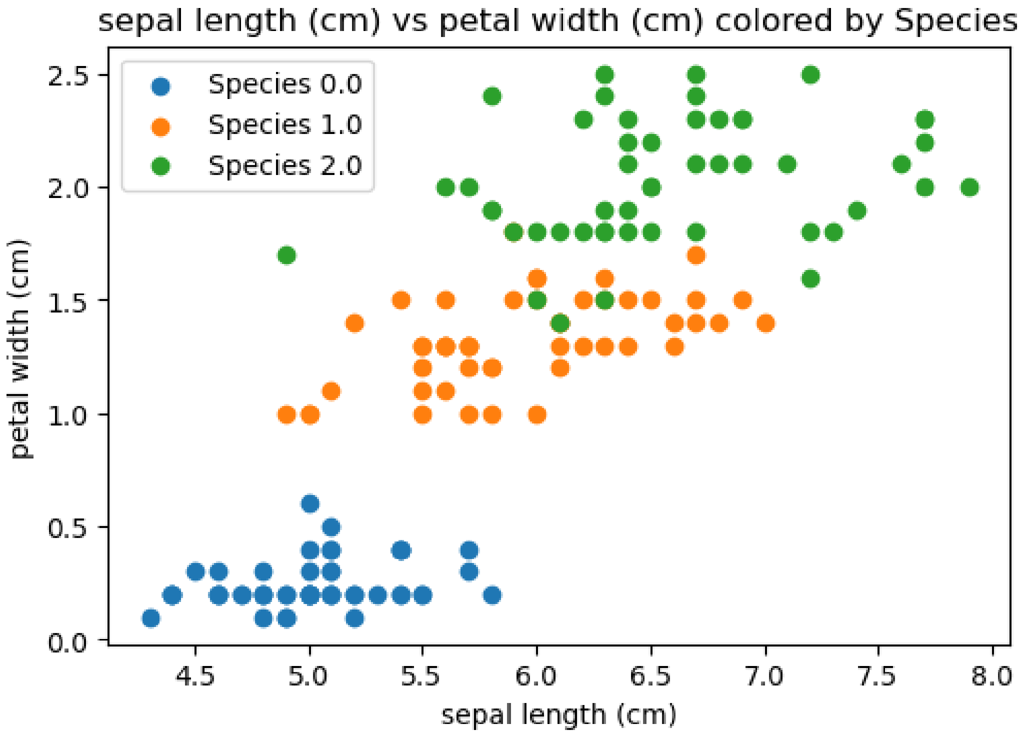

petal length (cm) vs Species
Value
7
6
5
4
3
2
1
0
20
40
60
80
100
120
140
petal length (cm)
Species 0.0
Species 1.0
Species 2.0



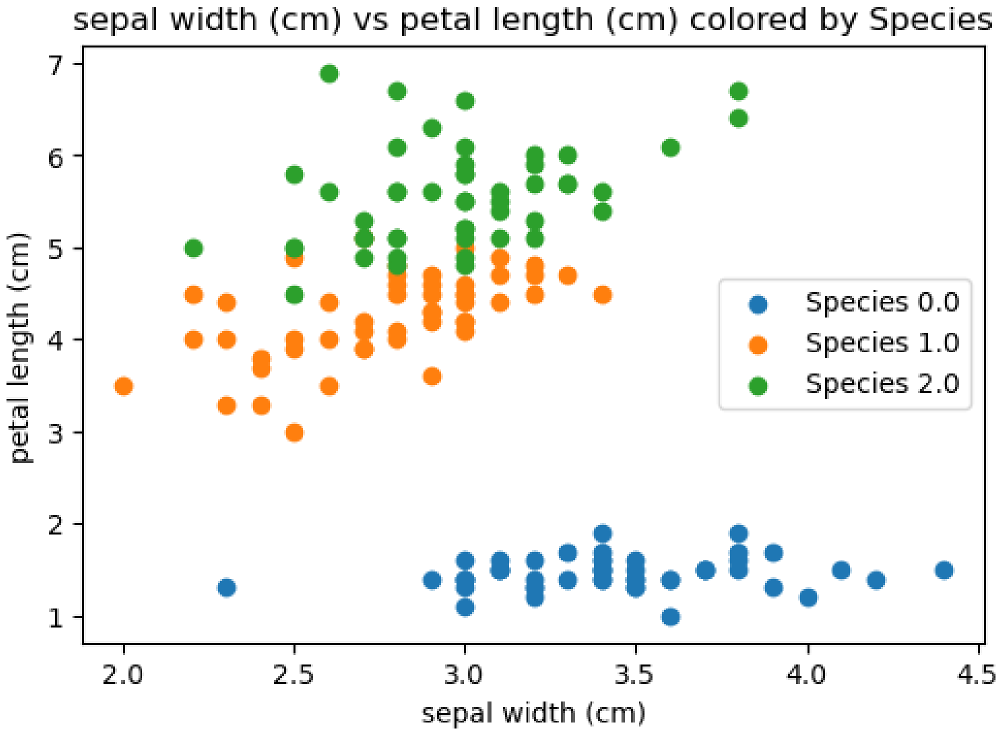
```
# Task 4: Scatter Plot for each pair of features
for i in range(4):
    for j in range(i + 1, 4):
        plt.figure(figsize=(6, 4))
        for species in iris_df['species'].unique():
            plt.scatter(iris_df[iris_df['species'] == species]
[iris['feature_names'][i]],
                        iris_df[iris_df['species'] == species]
[iris['feature_names'][j]], label=f"Species {species}")
        plt.title(f"{iris['feature_names'][i]} vs
{iris['feature_names'][j]} colored by Species")
        plt.xlabel(iris['feature_names'][i])
        plt.ylabel(iris['feature_names'][j])
        plt.legend()
        plt.show()
```


sepal length (cm) vs sepal width (cm) colored by Species
Species 0.0
Species 1.0
Species 2.0
sepal width (cm)
2.0
2.5
3.0
3.5
4.0
4.5
sepal length (cm)
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

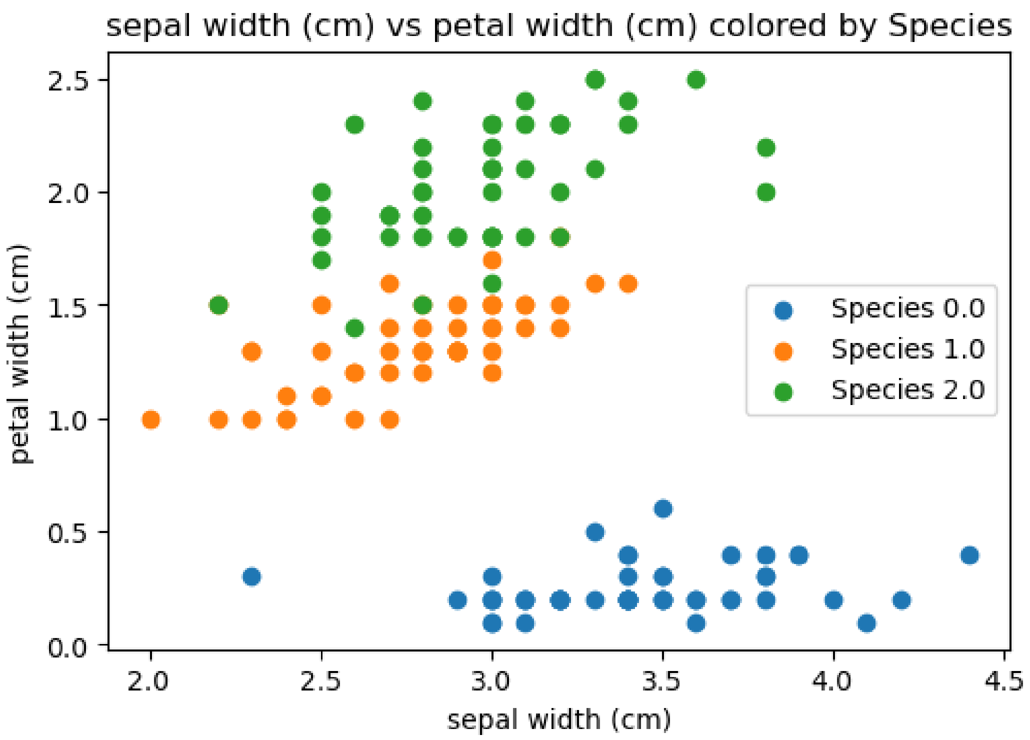

sepal length (cm) vs petal length (cm) colored by Species
Species 0.0
Species 1.0
Species 2.0
petal length (cm)
1
2
3
4
5
6
7
sepal length (cm)
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

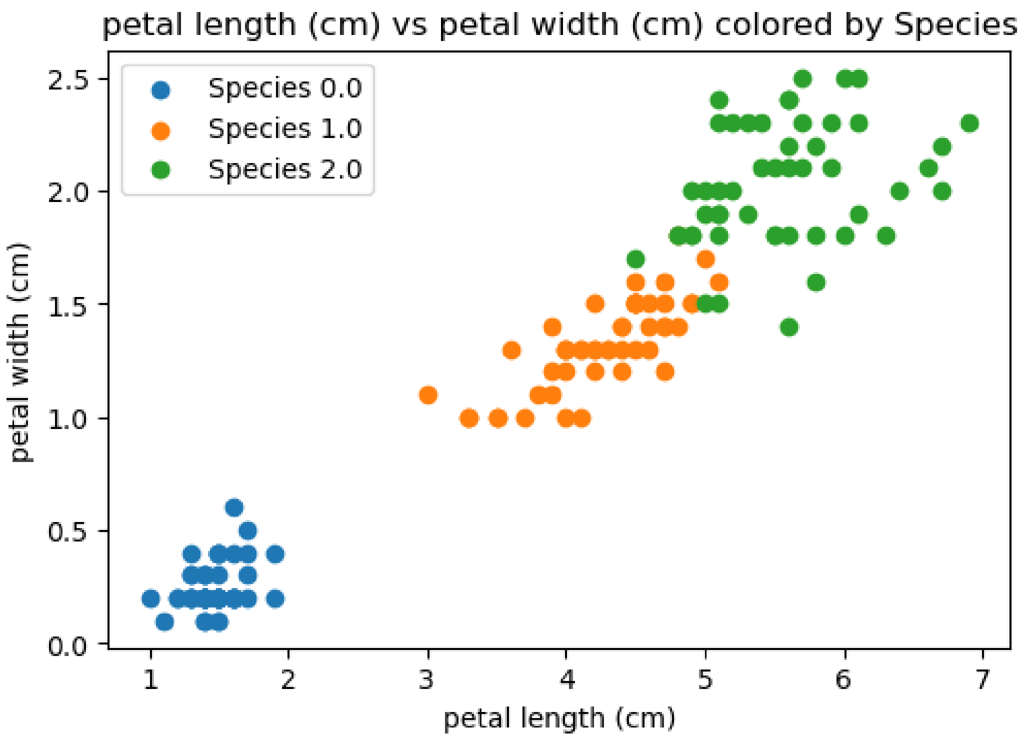


sepal length (cm) vs petal width (cm) colored by Species

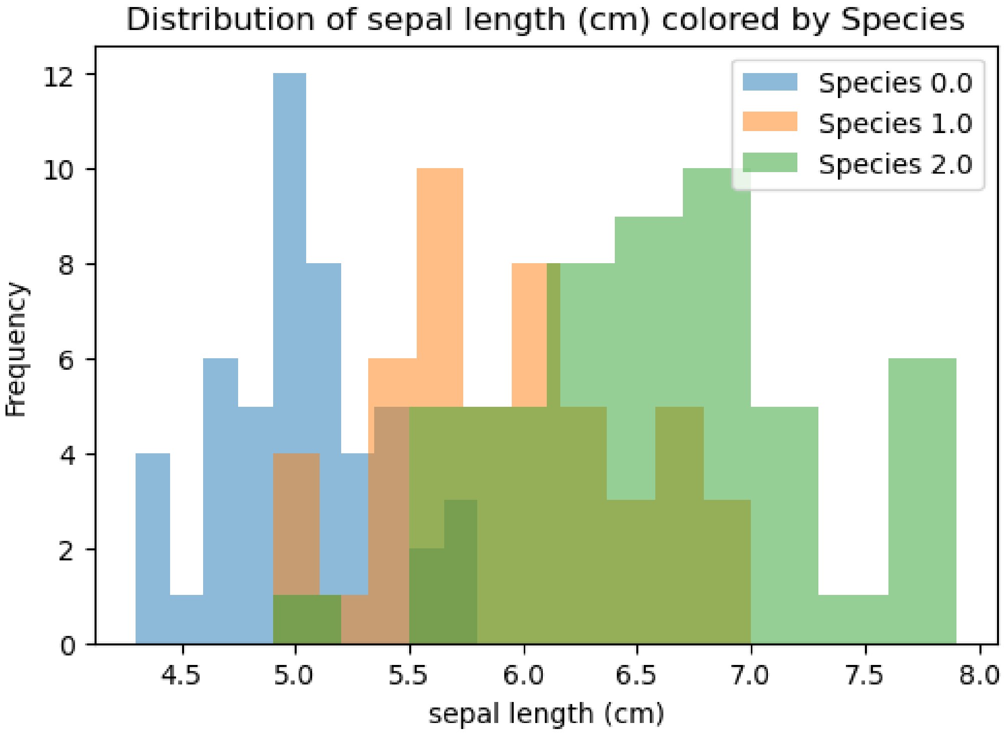


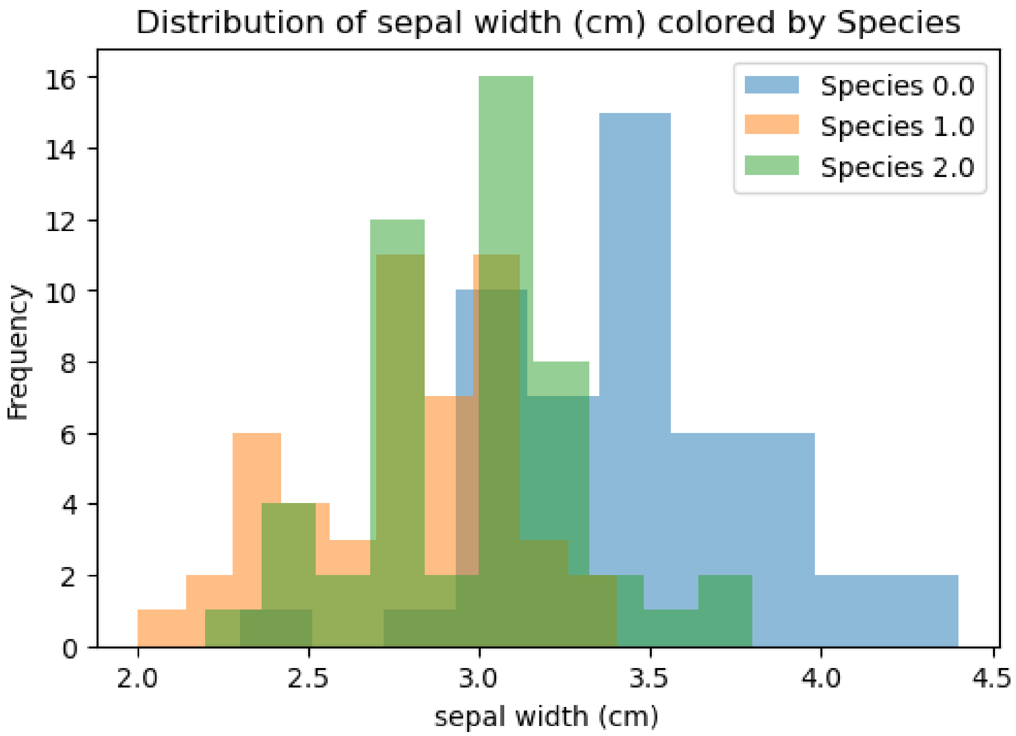
sepal width (cm) vs petal length (cm) colored by Species



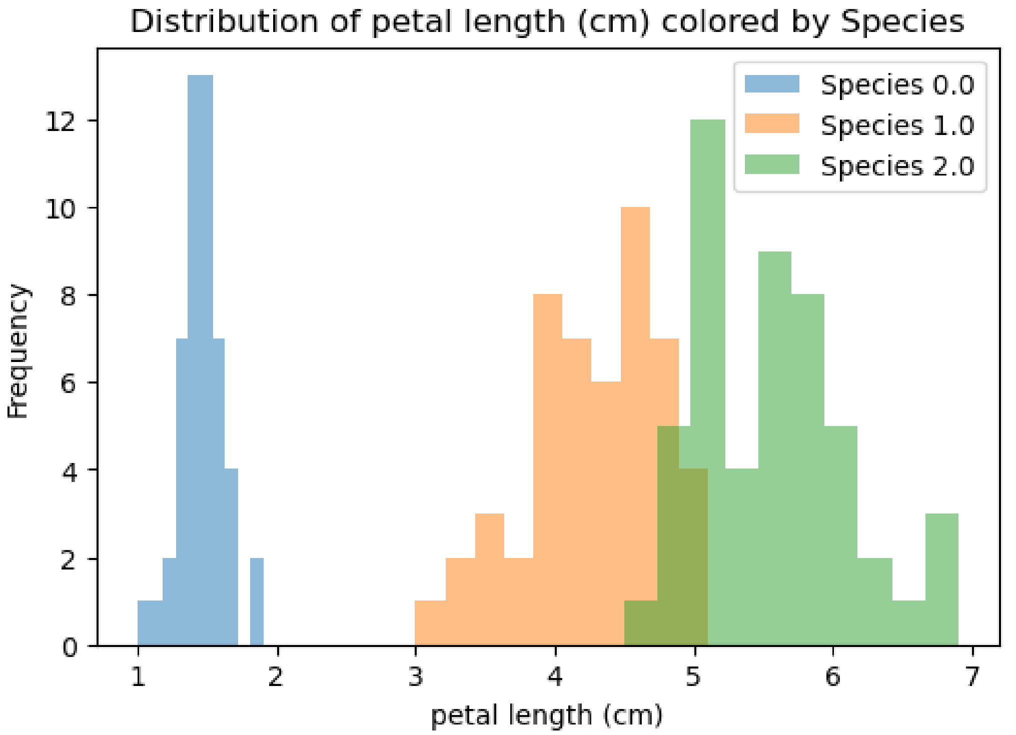


```
# Task 5: Histogram for each feature colored by species
for feature in iris['feature_names']:
    plt.figure(figsize=(6, 4))
    for species in iris_df['species'].unique():
        plt.hist(iris_df[iris_df['species'] == species][feature],
bins=10, alpha=0.5, label=f"Species {species}")
    plt.title(f"Distribution of {feature} colored by Species")
    plt.xlabel(feature)
    plt.ylabel("Frequency")
    plt.legend()
    plt.show()
```

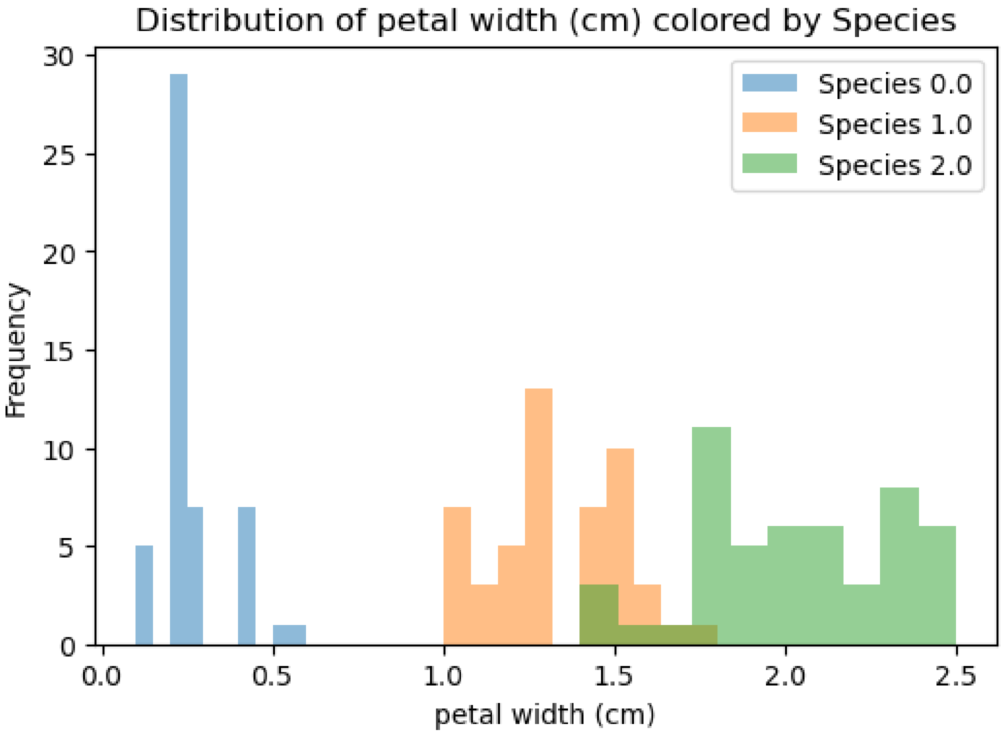




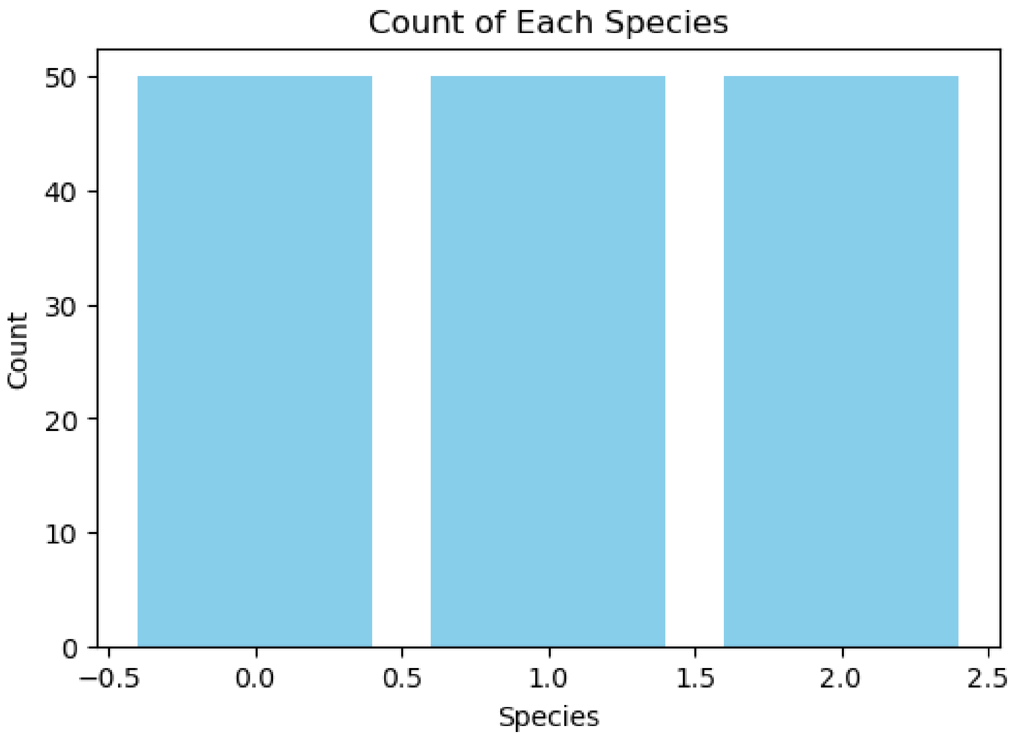
Distribution of sepal width (cm) colored by Species



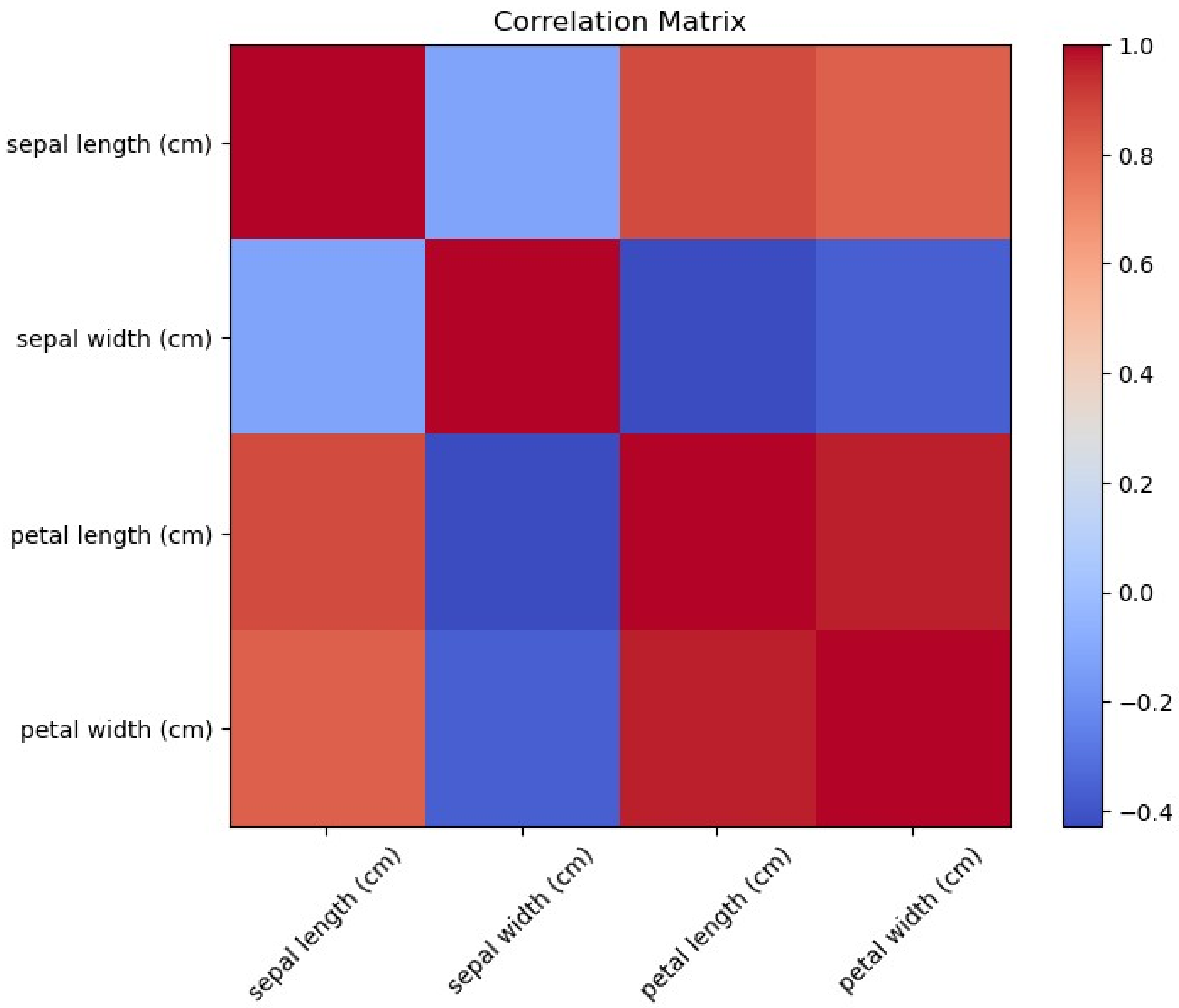
Distribution of petal length (cm) colored by Species



```
# Task 6: Bar Chart for target variable (species)
species_count = iris_df['species'].value_counts()
plt.figure(figsize=(6, 4))
plt.bar(species_count.index, species_count.values, color='skyblue')
plt.title("Count of Each Species")
plt.xlabel("Species")
plt.ylabel("Count")
plt.show()
```



```
# Task 7: Heatmap for correlation between features
plt.figure(figsize=(8, 6))
correlation_matrix = iris_df[iris['feature_names']].corr()
plt.imshow(correlation_matrix, cmap='coolwarm', interpolation='none')
plt.colorbar()
plt.title("Correlation Matrix")
plt.xticks(ticks=np.arange(4), labels=iris['feature_names'],
rotation=45)
plt.yticks(ticks=np.arange(4), labels=iris['feature_names'])
plt.show()
```

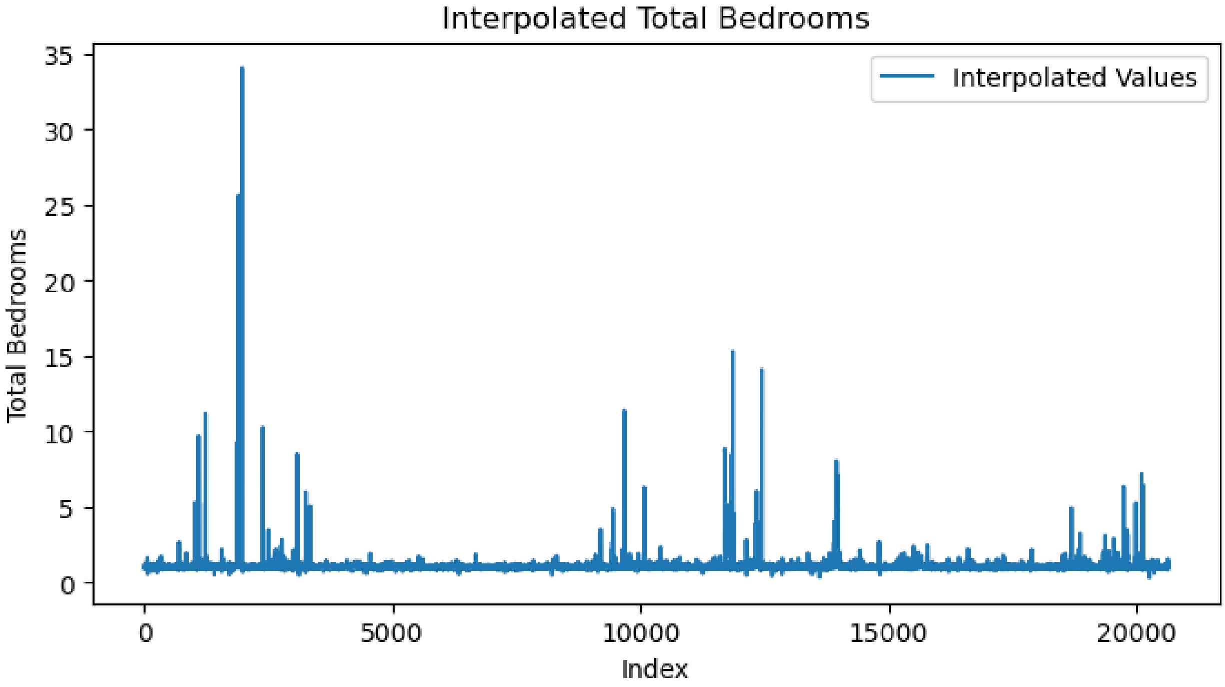


PART 2

```
import pandas as pd
import numpy as np
from sklearn.datasets import fetch_california_housing
import matplotlib.pyplot as plt

# Task 2: Load the California Housing Prices dataset and convert it
into a pandas DataFrame
california_housing = fetch_california_housing()
california_df = pd.DataFrame(data=california_housing.data,
columns=california_housing.feature_names)

# Add target column to the DataFrame
california_df['Target'] = california_housing.target
```

```
# Task 3: Identify Null Values
null_values = california_df.isnull().sum()
print("Missing Values in Each Column:\n", null_values)

Missing Values in Each Column:
 MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
Target        0
dtype: int64

# Task 4: Dropping Null Values
california_df_dropped = california_df.dropna()
print("Dataset after Dropping Null Values:\n",
california_df_dropped.head())

Dataset after Dropping Null Values:
    MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup
Latitude  \
0  8.3252      41.0  6.984127   1.023810       322.0  2.555556
37.88
1  8.3014      21.0  6.238137   0.971880      2401.0  2.109842
37.86
2  7.2574      52.0  8.288136   1.073446       496.0  2.802260
37.85
3  5.6431      52.0  5.817352   1.073059       558.0  2.547945
37.85
4  3.8462      52.0  6.281853   1.081081       565.0  2.181467
37.85

   Longitude  Target
0    -122.23   4.526
1    -122.22   3.585
2    -122.24   3.521
3    -122.25   3.413
4    -122.25   3.422

# Task 5: Filling Null Values with Median
median_total_bedrooms = california_df['AveBedrms'].median()
california_df_median_filled = california_df.fillna({'AveBedrms':
median_total_bedrooms})
print("Dataset after Filling with Median:\n",
california_df_median_filled.head())
```

```
Dataset after Filling with Median:
    MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup
Latitude  \
0  8.3252      41.0  6.984127   1.023810       322.0  2.555556
37.88
1  8.3014      21.0  6.238137   0.971880      2401.0  2.109842
37.86
2  7.2574      52.0  8.288136   1.073446       496.0  2.802260
37.85
3  5.6431      52.0  5.817352   1.073059       558.0  2.547945
37.85
4  3.8462      52.0  6.281853   1.081081       565.0  2.181467
37.85

   Longitude  Target
0    -122.23   4.526
1    -122.22   3.585
2    -122.24   3.521
3    -122.25   3.413
4    -122.25   3.422

# Task 6: Interpolation and Visualization
california_df_interpolated =
california_df['AveBedrms'].interpolate(method='linear')
plt.figure(figsize=(8, 4))
plt.plot(california_df_interpolated, label="Interpolated Values")
plt.title("Interpolated Total Bedrooms")
plt.xlabel("Index")
plt.ylabel("Total Bedrooms")
plt.legend()
plt.show()
```


Interpolated Total Bedrooms
Interpolated Values
Total Bedrooms
0
5
10
15
20
25
30
35
Index
0
5000
10000
15000
20000