Problem Set 02

CSC 5601 - Theory of Machine Learning

Hudson Arney

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

Experiment 1

a.

In [2]:
```
num_points = 20

x1_min, x1_max = 0, 10
x2_min, x2_max = 0, 10
```

In [3]:
```
x1_values = np.linspace(x1_min, x1_max, num_points)
x2_values = np.linspace(x2_min, x2_max, num_points)

x1_grid, x2_grid = np.meshgrid(x1_values, x2_values)

x1_flat = x1_grid.flatten()
x2_flat = x2_grid.flatten()

points_2d = np.column_stack((x1_flat, x2_flat))
```

In [4]:
```
print("Shape of points 2-D array:", points_2d.shape)
```

```
Shape of points 2-D array: (400, 2)
```

b.

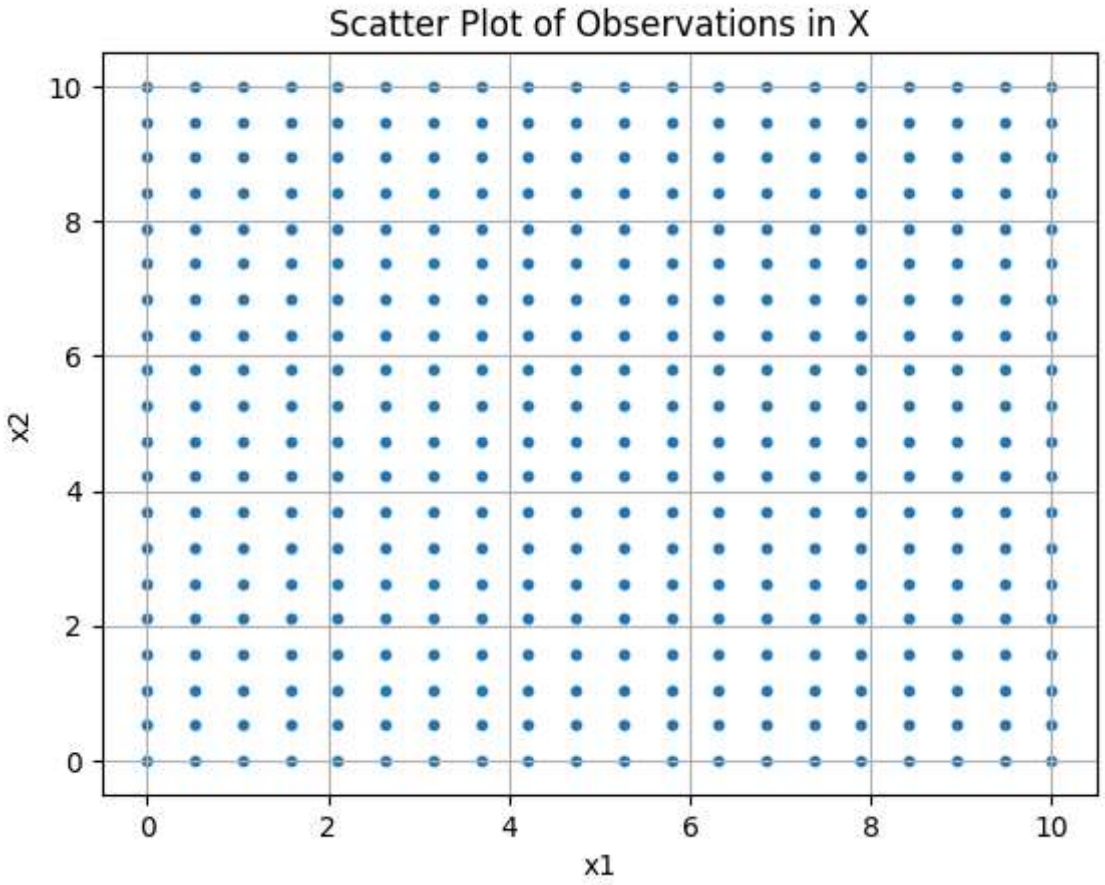
In [5]:
```
x1_vector = np.reshape(x1_grid, (num_points**2, 1))
x2_vector = np.reshape(x2_grid, (num_points**2, 1))
X = np.hstack((x1_vector, x2_vector))

print("Shape of X matrix:", X.shape)

plt.scatter(X[:, 0], X[:, 1], marker='o', s=10)
plt.title('Scatter Plot of Observations in X')
plt.xlabel('x1')
plt.ylabel('x2')
```

```
plt.grid(True)
plt.show()
```

```
Shape of X matrix: (400, 2)
```



C.

In [6]:
```
w = np.array([5, 3])
w_0 = -10
```

In [7]:
```
v = np.dot(X, w) + w_0
```

In [8]:
```
print("Weighted Sum Vector (v):", v)
```

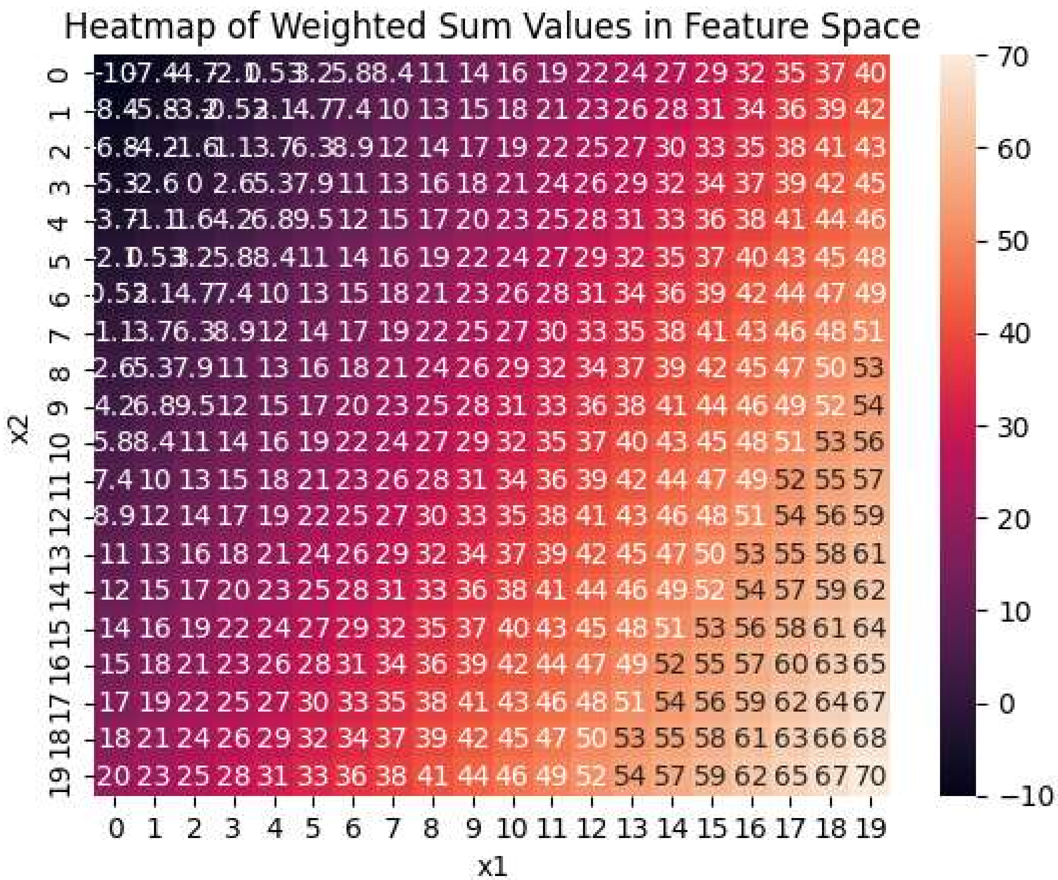
```
Weighted Sum Vector (v): [-10.          -7.36842105  -4.73684211  -2.10526316   0.52
631579
   3.15789474   5.78947368   8.42105263  11.05263158  13.68421053
  16.31578947  18.94736842  21.57894737  24.21052632  26.84210526
  29.47368421  32.10526316  34.73684211  37.36842105  40.
  -8.42105263  -5.78947368  -3.15789474  -0.52631579   2.10526316
   4.73684211   7.36842105  10.          12.63157895  15.26315789
  17.89473684  20.52631579  23.15789474  25.78947368  28.42105263
  31.05263158  33.68421053  36.31578947  38.94736842  41.57894737
  -6.84210526  -4.21052632  -1.57894737   1.05263158   3.68421053
   6.31578947   8.94736842  11.57894737  14.21052632  16.84210526
  19.47368421  22.10526316  24.73684211  27.36842105  30.
  32.63157895  35.26315789  37.89473684  40.52631579  43.15789474
  -5.26315789  -2.63157895   0.           2.63157895   5.26315789
   7.89473684  10.52631579  13.15789474  15.78947368  18.42105263
  21.05263158  23.68421053  26.31578947  28.94736842  31.57894737
  34.21052632  36.84210526  39.47368421  42.10526316  44.73684211
  -3.68421053  -1.05263158   1.57894737   4.21052632   6.84210526
   9.47368421  12.10526316  14.73684211  17.36842105  20.
  22.63157895  25.26315789  27.89473684  30.52631579  33.15789474
  35.78947368  38.42105263  41.05263158  43.68421053  46.31578947
  -2.10526316   0.52631579   3.15789474   5.78947368   8.42105263
  11.05263158  13.68421053  16.31578947  18.94736842  21.57894737
  24.21052632  26.84210526  29.47368421  32.10526316  34.73684211
  37.36842105  40.          42.63157895  45.26315789  47.89473684
  -0.52631579   2.10526316   4.73684211   7.36842105  10.
  12.63157895  15.26315789  17.89473684  20.52631579  23.15789474
  25.78947368  28.42105263  31.05263158  33.68421053  36.31578947
  38.94736842  41.57894737  44.21052632  46.84210526  49.47368421
   1.05263158   3.68421053   6.31578947   8.94736842  11.57894737
  14.21052632  16.84210526  19.47368421  22.10526316  24.73684211
  27.36842105  30.          32.63157895  35.26315789  37.89473684
  40.52631579  43.15789474  45.78947368  48.42105263  51.05263158
   2.63157895   5.26315789   7.89473684  10.52631579  13.15789474
  15.78947368  18.42105263  21.05263158  23.68421053  26.31578947
  28.94736842  31.57894737  34.21052632  36.84210526  39.47368421
  42.10526316  44.73684211  47.36842105  50.          52.63157895
   4.21052632   6.84210526   9.47368421  12.10526316  14.73684211
  17.36842105  20.          22.63157895  25.26315789  27.89473684
  30.52631579  33.15789474  35.78947368  38.42105263  41.05263158
  43.68421053  46.31578947  48.94736842  51.57894737  54.21052632
   5.78947368   8.42105263  11.05263158  13.68421053  16.31578947
  18.94736842  21.57894737  24.21052632  26.84210526  29.47368421
  32.10526316  34.73684211  37.36842105  40.          42.63157895
  45.26315789  47.89473684  50.52631579  53.15789474  55.78947368
   7.36842105  10.          12.63157895  15.26315789  17.89473684
  20.52631579  23.15789474  25.78947368  28.42105263  31.05263158
  33.68421053  36.31578947  38.94736842  41.57894737  44.21052632
  46.84210526  49.47368421  52.10526316  54.73684211  57.36842105
   8.94736842  11.57894737  14.21052632  16.84210526  19.47368421
  22.10526316  24.73684211  27.36842105  30.          32.63157895
  35.26315789  37.89473684  40.52631579  43.15789474  45.78947368
  48.42105263  51.05263158  53.68421053  56.31578947  58.94736842
  10.52631579  13.15789474  15.78947368  18.42105263  21.05263158
  23.68421053  26.31578947  28.94736842  31.57894737  34.21052632
  36.84210526  39.47368421  42.10526316  44.73684211  47.36842105
```

```
50.          52.63157895  55.26315789  57.89473684  60.52631579
12.10526316  14.73684211  17.36842105  20.          22.63157895
25.26315789  27.89473684  30.52631579  33.15789474  35.78947368
38.42105263  41.05263158  43.68421053  46.31578947  48.94736842
51.57894737  54.21052632  56.84210526  59.47368421  62.10526316
13.68421053  16.31578947  18.94736842  21.57894737  24.21052632
26.84210526  29.47368421  32.10526316  34.73684211  37.36842105
40.          42.63157895  45.26315789  47.89473684  50.52631579
53.15789474  55.78947368  58.42105263  61.05263158  63.68421053
15.26315789  17.89473684  20.52631579  23.15789474  25.78947368
28.42105263  31.05263158  33.68421053  36.31578947  38.94736842
41.57894737  44.21052632  46.84210526  49.47368421  52.10526316
54.73684211  57.36842105  60.          62.63157895  65.26315789
16.84210526  19.47368421  22.10526316  24.73684211  27.36842105
30.          32.63157895  35.26315789  37.89473684  40.52631579
43.15789474  45.78947368  48.42105263  51.05263158  53.68421053
56.31578947  58.94736842  61.57894737  64.21052632  66.84210526
18.42105263  21.05263158  23.68421053  26.31578947  28.94736842
31.57894737  34.21052632  36.84210526  39.47368421  42.10526316
44.73684211  47.36842105  50.          52.63157895  55.26315789
57.89473684  60.52631579  63.15789474  65.78947368  68.42105263
20.          22.63157895  25.26315789  27.89473684  30.52631579
33.15789474  35.78947368  38.42105263  41.05263158  43.68421053
46.31578947  48.94736842  51.57894737  54.21052632  56.84210526
59.47368421  62.10526316  64.73684211  67.36842105  70.        ]
```

d.

In [9]:
```
v_matrix = np.reshape(v, (num_points, num_points))

sns.heatmap(v_matrix, annot=True)
plt.title('Heatmap of Weighted Sum Values in Feature Space')
plt.xlabel('x1')
plt.ylabel('x2')
plt.show()
```



e.

In [10]:
```python
y_hat = (v > 0).astype(int)

print("Predicted Labels (y_hat):", y_hat)
```

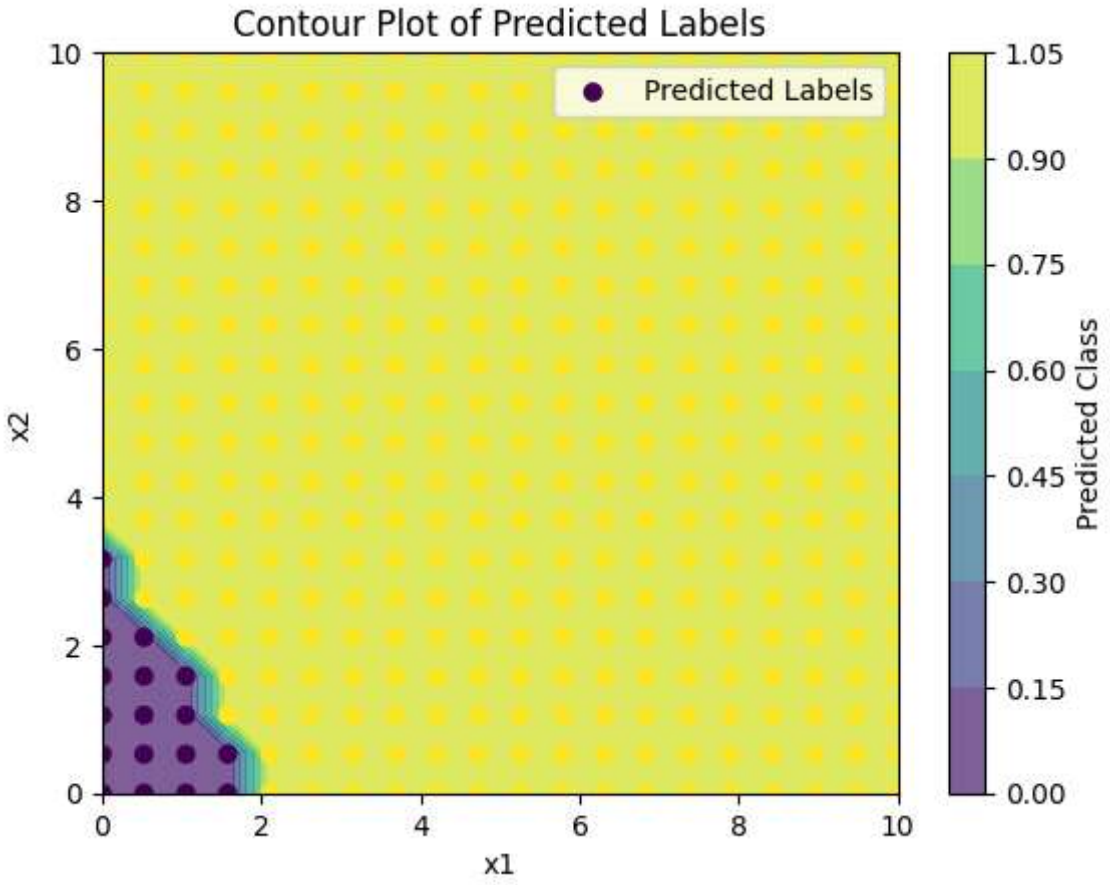
```
Predicted Labels (y_hat): [0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1
1 1 1 1 1 1 1 1
 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

f.

In [11]:
```python
y_hat_matrix = np.reshape(y_hat, (num_points, num_points))

plt.contourf(x1_grid, x2_grid, y_hat_matrix, alpha=0.7)
plt.colorbar(label='Predicted Class')
plt.scatter(X[:, 0], X[:, 1], c=y_hat, label='Predicted Labels')
plt.title('Contour Plot of Predicted Labels')
```

```
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend()
plt.show()
```



Experiment 2 - More Vectors

b) i)

In [12]:
```
n = np.array([2, 2])
points = np.array([[2, 15],
                   [3, 17],
                   [4, 13],
                   [4, 1],
                   [5, 3],
                   [6, 2]])
```

In [13]:
```
differences = points - np.array([5, 5])

dot_products = np.dot(differences, n)

signs = np.sign(dot_products)
```

In [14]:
```
for i, point in enumerate(points):
```

```
    print(f"Point {i + 1}: Dot product = {dot_products[i]}, Sign = {signs[i]}")
```

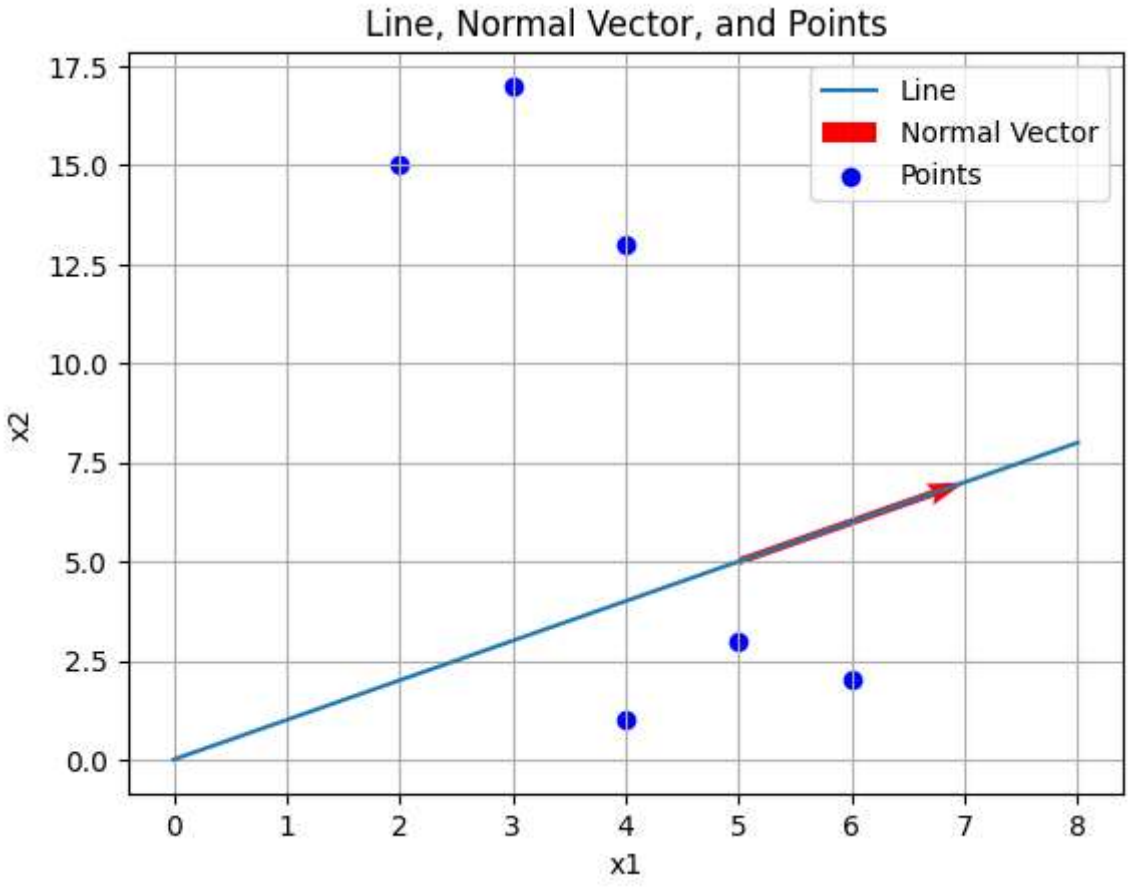
```
Point 1: Dot product = 14, Sign = 1
Point 2: Dot product = 20, Sign = 1
Point 3: Dot product = 14, Sign = 1
Point 4: Dot product = -10, Sign = -1
Point 5: Dot product = -4, Sign = -1
Point 6: Dot product = -4, Sign = -1
```

ii)

In [26]:

```
x_line = np.linspace(min(points[:, 0]) - 2, max(points[:, 0]) + 2, 100)
y_line = (n[1]/n[0]) * (x_line - 5) + 5

plt.plot(x_line, y_line, label='Line')
plt.quiver(5, 5, n[0], n[1], angles='xy', scale_units='xy', scale=1, color='red', l
plt.scatter(points[:, 0], points[:, 1], color='blue', label='Points')
plt.xlabel('x1')
plt.ylabel('x2')
plt.title('Line, Normal Vector, and Points')
plt.legend()
plt.grid(True)
plt.show()
```



iii)

In [27]:
```
all_positive = all(dot_products > 0)

if all_positive:
    print("All points lie on the same side of the normal vector, and the dot produc
else:
    print("Not all points lie on the same side of the normal vector, or the dot pro
```

```
Not all points lie on the same side of the normal vector, or the dot product is not
always positive.
```

Experiment 3 - Standardization

a.

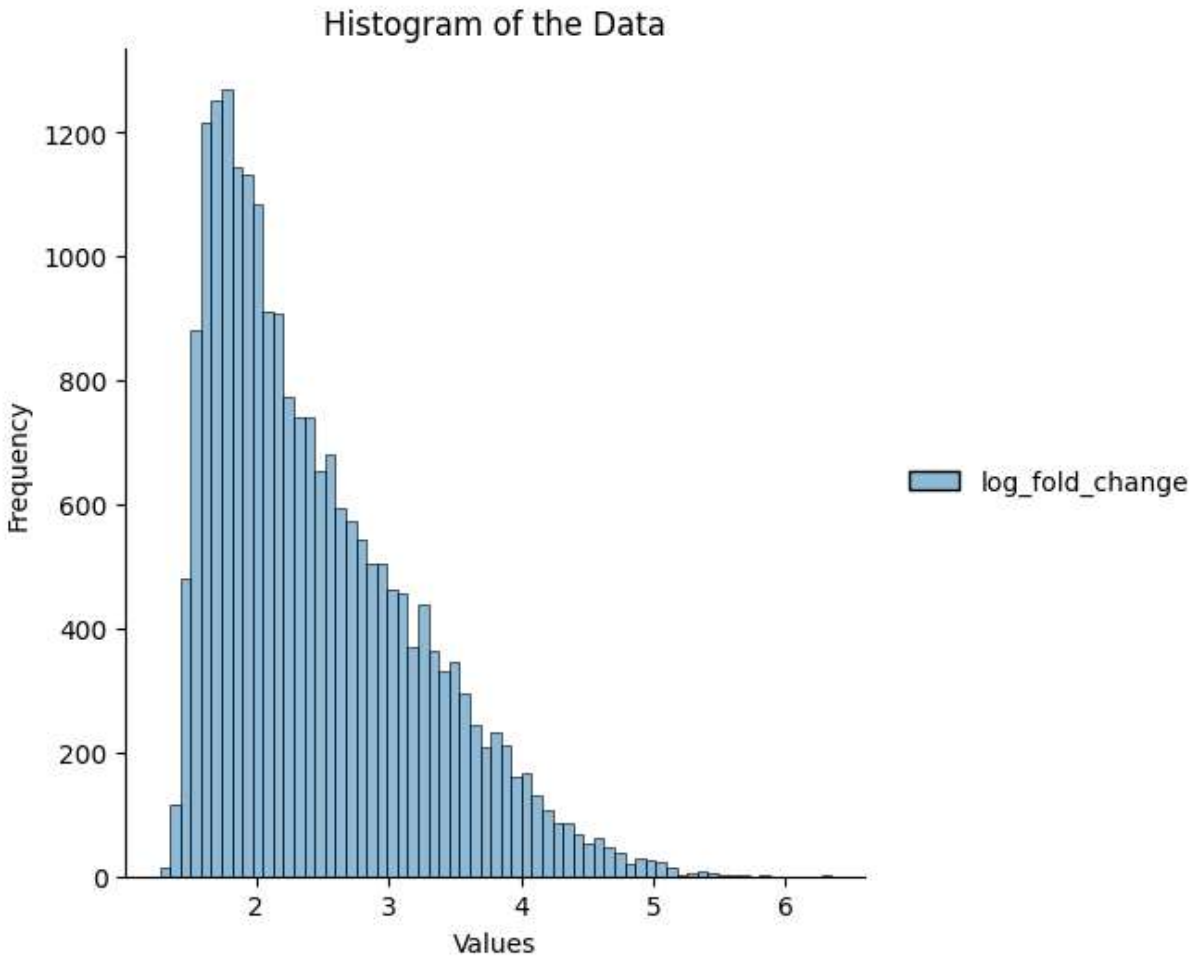
In [15]:
```
data = pd.read_csv("peak_lfc.csv")
data.head()
```

Out[15]:

	log_fold_change
0	4.28351
1	2.79465
2	1.51552
3	1.51892
4	2.05290

b.

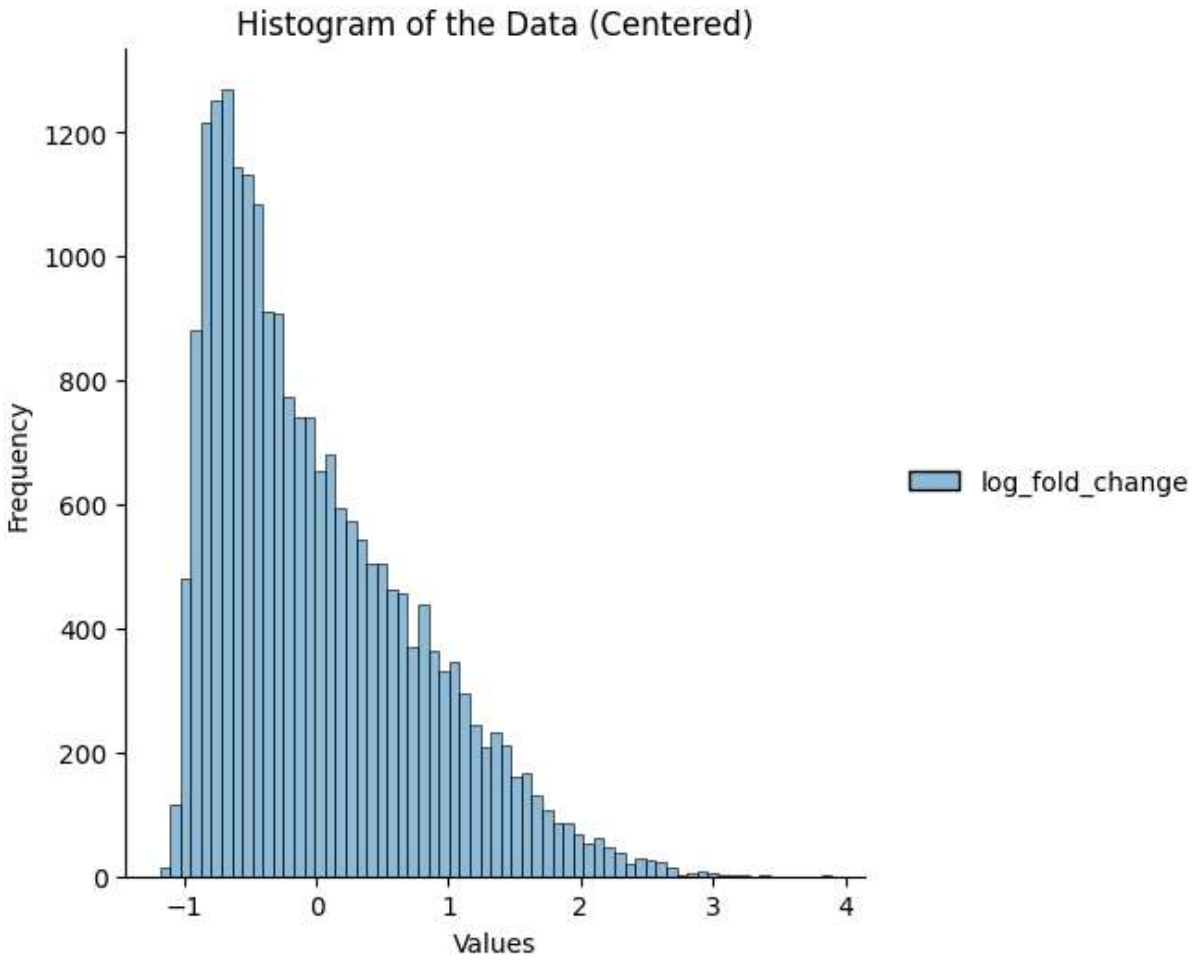
In [16]:
```
sns.displot(data)
plt.title('Histogram of the Data')
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.show()
```



C.

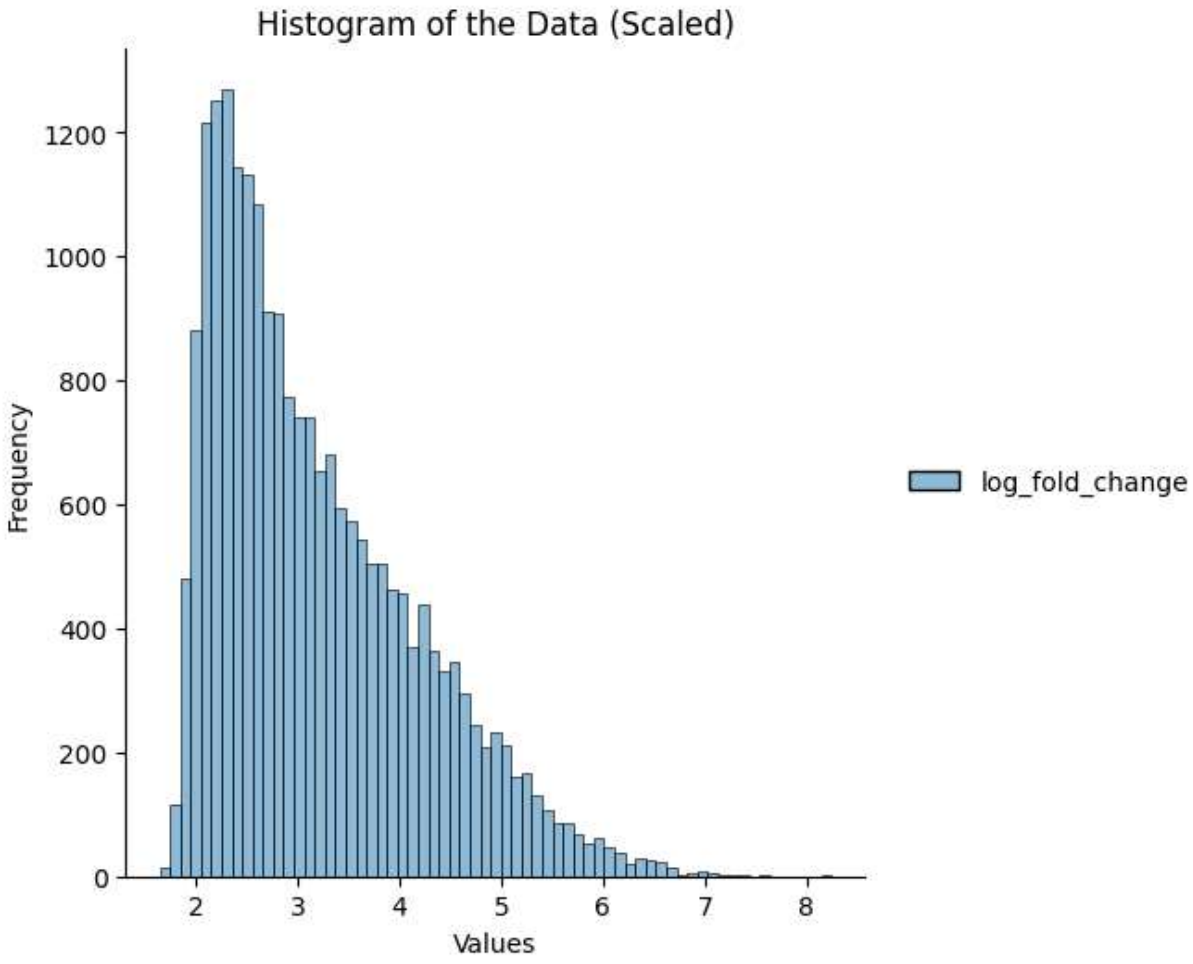
In [17]:
```
centered_data = data - data.mean(axis=0)
```

In [18]:
```
sns.displot(centered_data)
plt.title('Histogram of the Data (Centered)')
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.show()
```



d.

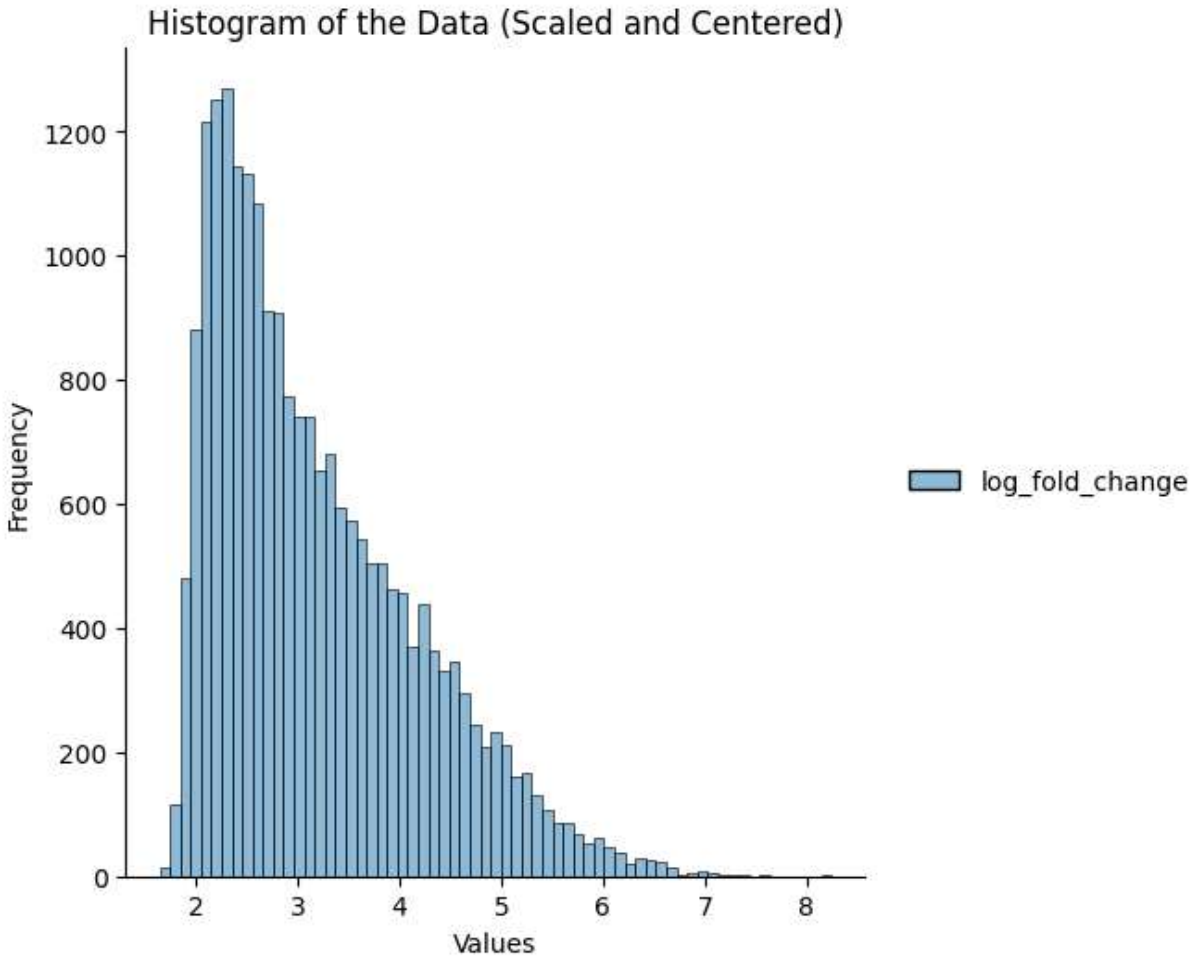
In [19]:
```
scaled_data = data / data.std(axis=0)
sns.displot(scaled_data)
plt.title('Histogram of the Data (Scaled)')
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.show()
```



e.

In [20]:
```
center_scaled_data = centered_data / np.std(centered_data)
```

In [21]:
```
sns.displot(scaled_data)
plt.title('Histogram of the Data (Scaled and Centered)')
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.show()
```



Experiment 4 - Observation Distances

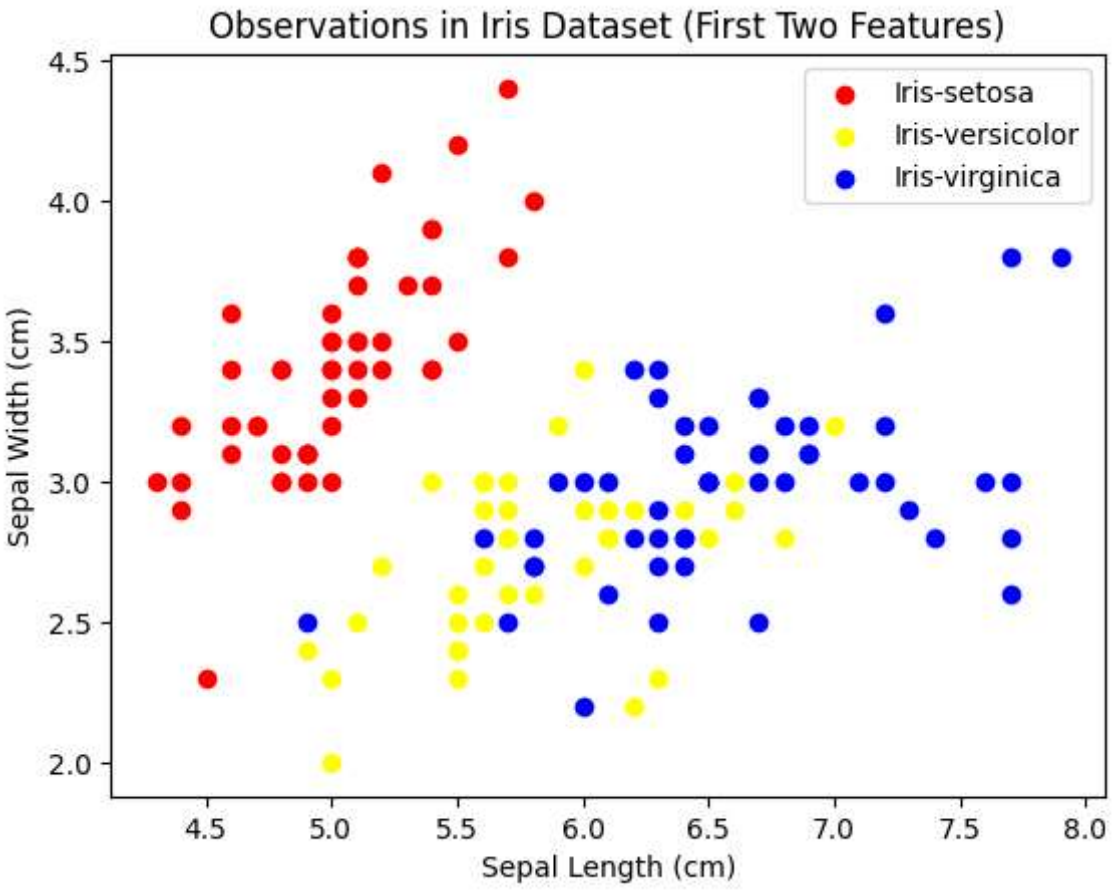
Before considering the following indexing procedures, think about the following question. Can I index a vector (nx1) using a matrix (nxm)? What would happen if I try?

In [44]:
```
iris_df = pd.read_csv("IRIS.csv")
X = iris_df.iloc[:, :2]

colors = {'Iris-setosa': 'red', 'Iris-versicolor': 'yellow', 'Iris-virginica': 'blu
labels = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']

for species, color in colors.items():
    species_data = iris_df[iris_df['species'] == species]
    plt.scatter(species_data.iloc[:, 0], species_data.iloc[:, 1], c=color)

plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.title('Observations in Iris Dataset (First Two Features)')
plt.legend(labels=labels)
plt.show()
```



2)

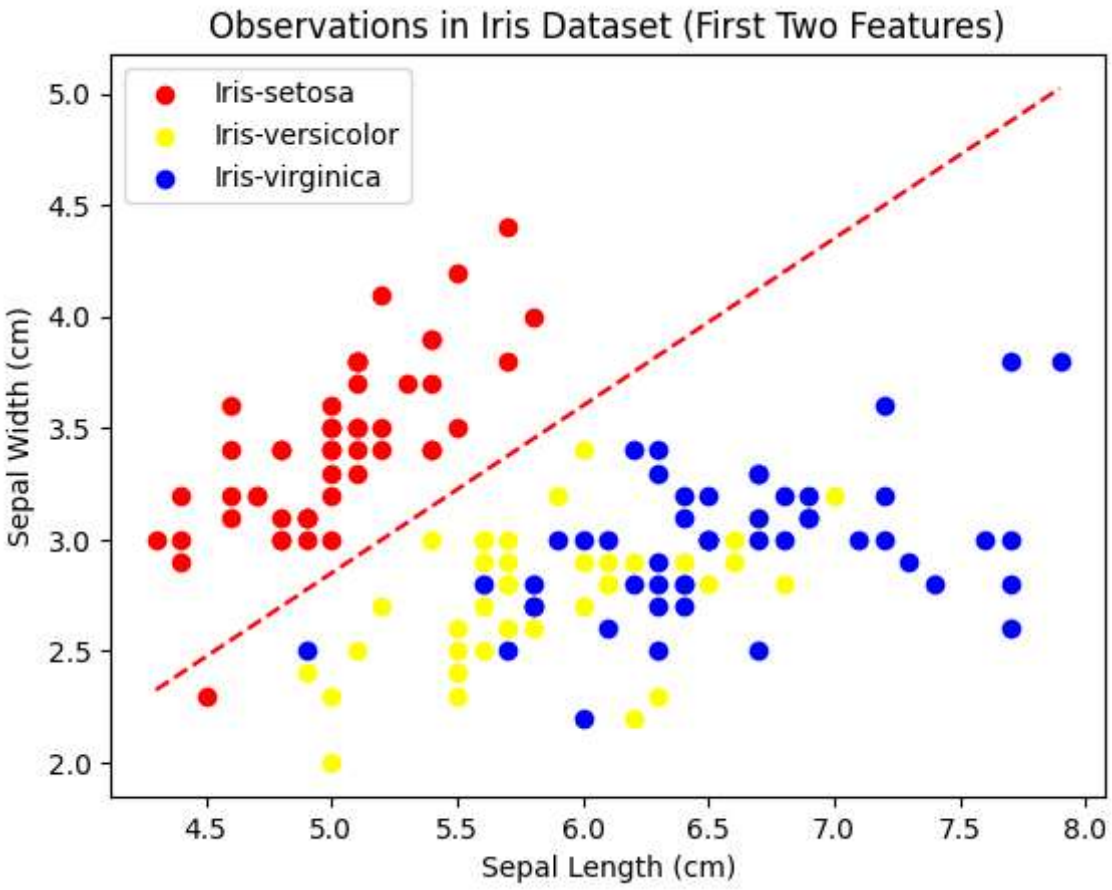
In [40]:
```
def decision_boundary(x1):
    return 0.75 * x1 - 0.9
```

In [43]:
```
colors = {'Iris-setosa': 'red', 'Iris-versicolor': 'yellow', 'Iris-virginica': 'blu
labels = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']

for species, color in colors.items():
    species_data = iris_df[iris_df['species'] == species]
    plt.scatter(species_data.iloc[:, 0], species_data.iloc[:, 1], c=color)

plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.title('Observations in Iris Dataset (First Two Features)')
plt.legend(labels=labels)
x1_values = np.linspace(min(X.iloc[:, 0]), max(X.iloc[:, 0]), 100)
x2_values = decision_boundary(x1_values)
plt.plot(x1_values, x2_values, color='red', linestyle='dashed', label='Decision Bou
```

Out[43]:
```
[<matplotlib.lines.Line2D at 0x7f95a9e07af0>]
```



3)

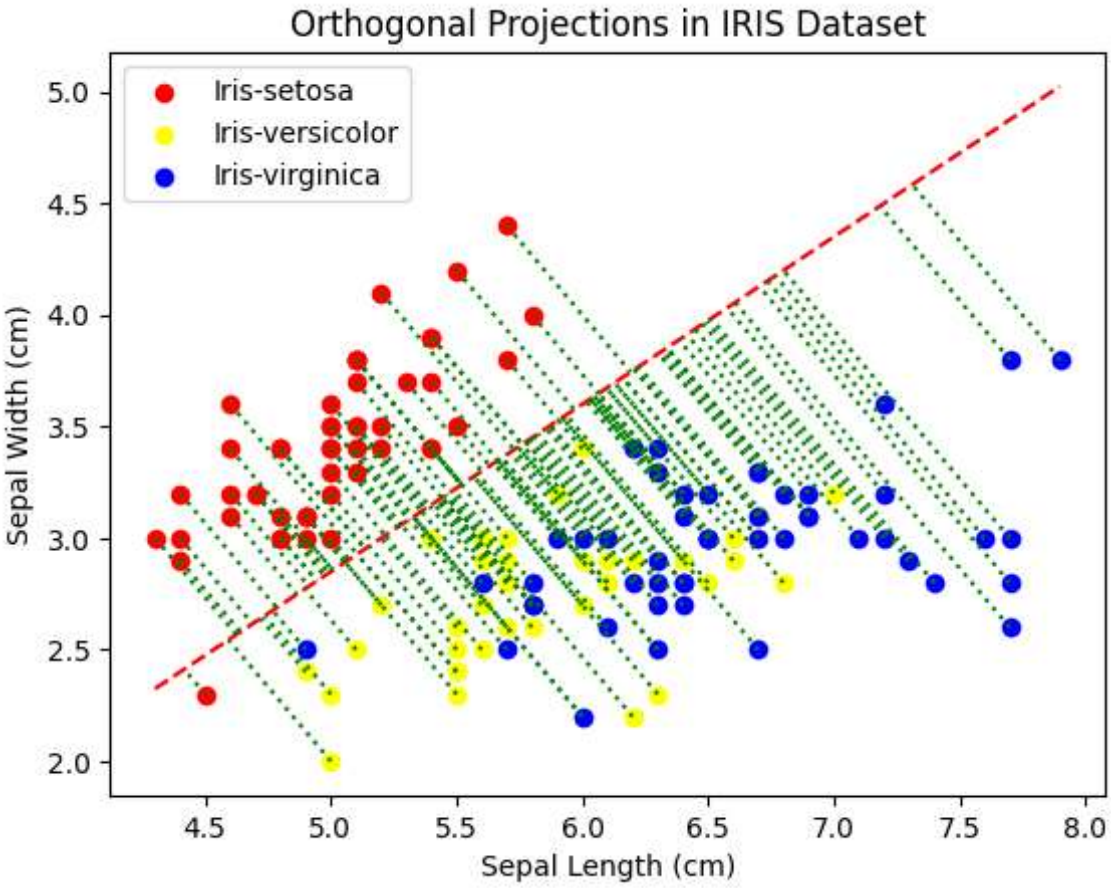
In [46]:
```
def find_projection(observation):
    x1_obs, x2_obs = observation
    x1_proj = (x1_obs + 0.75 * x2_obs - 0.75 * (-0.9)) / (1 + 0.75**2)
    x2_proj = 0.75 * x1_proj - 0.9
    return x1_proj, x2_proj
```

In [53]:
```
for species, color in colors.items():
    species_data = iris_df[iris_df['species'] == species]
    plt.scatter(species_data.iloc[:, 0], species_data.iloc[:, 1], c=color)

x1_values = np.linspace(min(X.iloc[:, 0]), max(X.iloc[:, 0]), 100)
x2_values = decision_boundary(x1_values)
plt.plot(x1_values, x2_values, color='red', linestyle='dashed', label='Decision Bou

for index, observation in X.iterrows():
    projection = find_projection(observation)
    plt.plot([observation[0], projection[0]], [observation[1], projection[1]], colo

plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.title('Orthogonal Projections in IRIS Dataset')
plt.legend(labels=labels)
plt.show()
```


Orthogonal Projections in IRIS Dataset
Iris-setosa
Iris-versicolor
Iris-virginica
Sepal Width (cm)
Sepal Length (cm)
5.0
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0