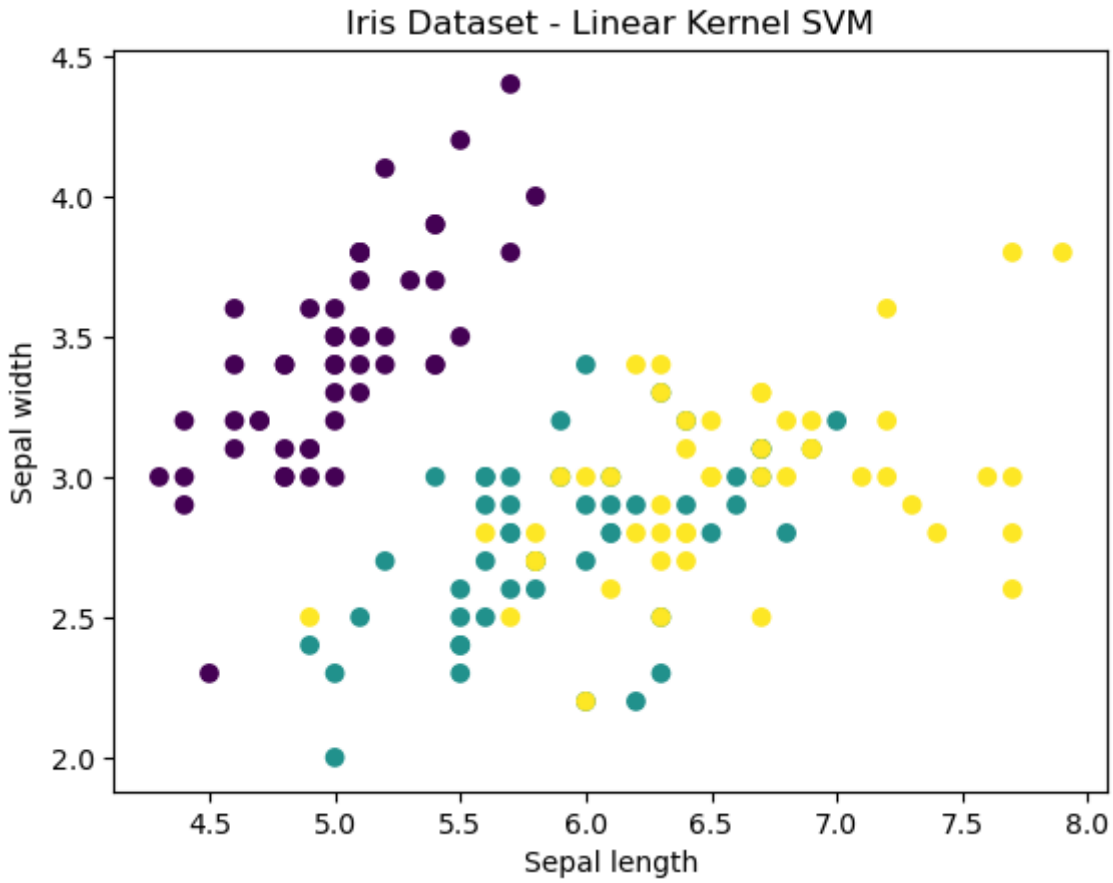
In [12]:
```
import sys

R = 3
C = 3

def minCost(cost, m, n):
    if (n < 0 or m < 0):
        return sys.maxsize
    elif (m == 0 and n == 0):
        return cost[m][n]
    else:
        return cost[m][n] + min(minCost(cost, m-1, n-1),
                                minCost(cost, m-1, n),
                                minCost(cost, m, n-1))
def min(x, y, z):
    if (x < y):
        return x if (x < z) else z
    else:
        return y if (y < z) else z

cost = [[1, 2, 3],
        [4, 8, 2],
        [1, 5, 3]]

sorted_cost = [sorted(row) for row in cost]
for row in sorted_cost:
    print(row)

print(minCost(cost, 2, 2))
```

```
[1, 2, 3]
[2, 4, 8]
[1, 3, 5]
8
```

In [13]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                    random_state=0)
svm = SVC(kernel='linear')
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap='viridis')
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.title('Iris Dataset - Linear Kernel SVM')
plt.show()
```

```
Accuracy: 0.9777777777777777
```


Iris Dataset - Linear Kernel SVM
Sepal width
Sepal length
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

In [ ]: