SVM-Lineare Trennbarkeit (Beispiel Irisblumen)

Beispiel: Klassifikation von zwei Iris-Blumenarten mit SVM. Wir verwenden den bekannten Iris-Datensatz, der Informationen über drei Arten von Iris-Blumen enthält: Iris Setosa, Iris Versicolor und Iris Virginica. Jede Blume wird durch vier Merkmale beschrieben:

Sepallänge (in cm) Sepalbreite (in cm) Petallänge (in cm) Petalbreite (in cm)

Zur Vereinfachung beschränken wir uns auf zwei Arten von Blumen (Iris Setosa und Iris Versicolor) und zwei Merkmale (Petallänge und Petalbreite).

Datensatz: Iris-Blumen

Autor: Dr.H.Völlinger, Oktober, 2024

In [22]:
```
# 1. Bibliotheken importieren
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
import matplotlib.pyplot as plt
import numpy as np

# import der Library time zum Prüfen der Ausführungszeit
import time

# check version of numpy
print("numpy {}".format(np.__version__))

# print the date & time of the notebook
print('**********************************************************************
**')
print("Actual date & time of the notebook:",time.strftime("%d.%m.%Y  %H:%M:%
S"))
print('**********************************************************************
***')
```

```
numpy 1.18.1
**********************************************************************
Actual date & time of the notebook: 16.10.2024  07:43:26
**********************************************************************
```

Iris-Datensatz Laden und Beschränkung auf zwei Klassen

Wir extrahieren die Daten für zwei Klassen von Blumen:

Iris Setosa: Hat kurze Petalen ("Blütenblätter") Iris Versicolor: Hat längere Petalen.

Beispielhafte Daten aus dem Datensatz:

Iris Setosa: Petallänge: 1.4 cm, Petalbreite: 0.2 cm Petallänge: 1.5 cm, Petalbreite: 0.4 cm

Iris Versicolor: Petallänge: 4.7 cm, Petalbreite: 1.4 cm Petallänge: 4.6 cm, Petalbreite: 1.3 cm

Diese beiden Arten von Blumen haben deutliche Unterschiede in der Petallänge und Petalbreite, was uns hilft, eine Trennlinie mit SVM zu finden.

In [23]:
```
# 2. Iris-Datensatz laden, Beschränkung auf zwei Klassen und Aufteilung in Tr
aining- und Test-Daten
iris = datasets.load_iris()
X = iris.data[:, [2, 3]]  # Petallänge und Petalbreite verwenden
y = iris.target

# Daten auf zwei Klassen beschränken: Iris Setosa und Iris Versicolor
X = X[y != 2]  # Nur Klassen 0 und 1 (Setosa und Versicolor)
y = y[y != 2]

# Daten in Trainings- und Testmenge aufteilen
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rand
om_state=42)
```

SVM Modell erstellen und trainieren

Anwendung des SVM-Algorithmus Das Ziel ist es, eine Trennlinie zu finden, die die beiden Klassen basierend auf den beiden Merkmalen Petallänge und Petalbreite trennt.

SVM versucht, eine Linie zu finden, die die beiden Klassen mit einem maximalen Margin (Abstand zwischen den nächstgelegenen Punkten beider Klassen zur Trennlinie) trennt. Diese nächstgelegenen Punkte nennt man Support-Vektoren.

Das SVM-Modell: Nach dem Training des SVM-Modells (z. B. mit einem linearen Kernel) könnte die SVM eine Trennlinie erstellen, die die beiden Arten basierend auf den Petal-Dimensionen trennt. Eine mögliche Gleichung für die Trennlinie könnte sein: " Petalbreite = -1.5 × Petalllänge + 4"

- Alle Punkte unterhalb dieser Linie gehören zur Klasse Iris Setosa.
- Alle Punkte oberhalb dieser Linie gehören zur Klasse Iris Versicolor.

In [24]:
```
# 3. SVM-Modell erstellen und trainieren
model = SVC(kernel='linear')
model.fit(X_train, y_train)
```

Out[24]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Plotten und Support-Vektoren

Ergebnisinterpretation: Die SVM hat gelernt, dass:

- Iris Setosa typischerweise kürzere und schmalere Petalen hat.
- Iris Versicolor hat längere und breitere Petalen.

Das SVM-Modell trennt die beiden Klassen, indem es den Abstand zwischen der Trennlinie und den nächstgelegenen Punkten maximiert. Sobald das Modell trainiert ist, kann es verwendet werden, um neue Datenpunkte (mit Petallänge und Petalbreite) in eine der beiden Klassen einzuteilen.

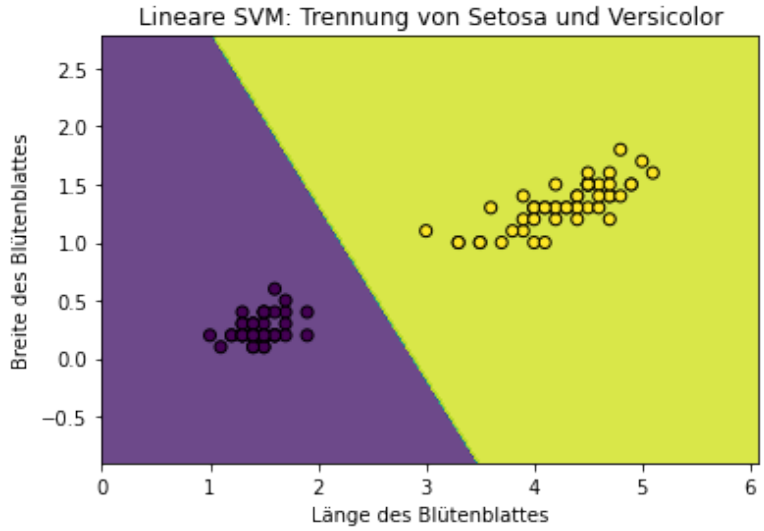
In [25]:
```
# 4. Trennlinie plotten
def plot_decision_boundary(X, y, model):
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
                         np.arange(y_min, y_max, 0.02))

    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.contourf(xx, yy, Z, alpha=0.8)
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', marker='o')
    plt.xlabel('Länge des Blütenblattes')
    plt.ylabel('Breite des Blütenblattes')
    plt.title('Lineare SVM: Trennung von Setosa und Versicolor')
    plt.show()

plot_decision_boundary(X, y, model)

# 5. Ausgabe der Support-Vektoren
print('Dies sind die Support-Vektoren:')
print(model.support_vectors_)
```




```
Dies sind die Support-Vektoren:
[[1.9 0.2]
 [1.7 0.5]
 [3.  1.1]]
```

Fazit und Programmende

Fazit: Dieses einfache Beispiel zeigt, wie SVM verwendet werden kann, um zwei Arten von Iris-Blumen anhand von zwei Merkmalen (Petallänge und Petalbreite) zu trennen. Mit SVM kann man nicht nur lineare, sondern auch nicht-lineare Klassifikationsprobleme lösen, indem man z. B. den Kernel-Trick verwendet.

In [27]:
```
# 6. Print current date and time

print('******* Aktuelles Datum und Zeit*********')
print("Date & Time:",time.strftime("%d.%m.%Y  %H:%M:%S"))
# end of import test
print ("Ende Python-Programm ***SVM_Irisblume ***")
```

```
******* Aktuelles Datum und Zeit*********
Date & Time: 16.10.2024  07:44:46
Ende Python-Programm ***SVM_Irisblume ***
```