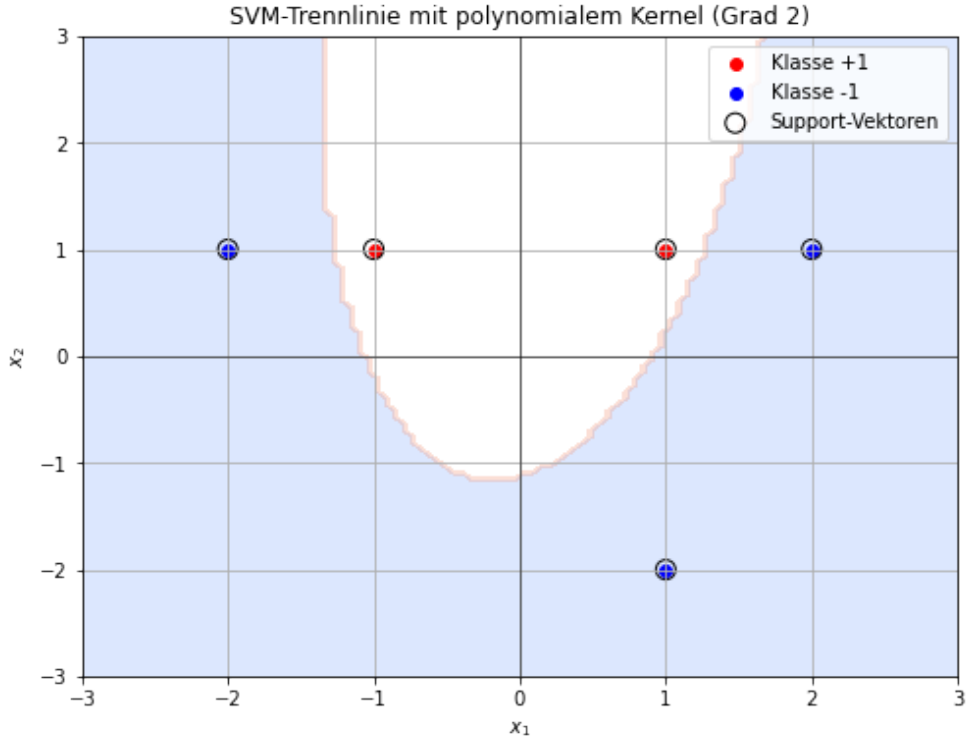
In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC

# Definierte Datenpunkte und Labels
X = np.array([[1, 1], [-1, 1], [2, 1], [1, -2], [-2, 1]])  # A, B, C, D, E
y = np.array([1, 1, -1, -1, -1])  # Labels: +1 für A und B und -1 für C, D un
d E

# Erstelle das SVM-Modell mit einem polynomialen Kernel (Grad 2)
svm_model = SVC(kernel='poly', degree=2, coef0=1)  # coef0=1 entspricht dem k
onstanten Term in K
svm_model.fit(X, y)

# Bereich für x1 und x2 (damit die Linie in einem sinnvollen Bereich dargeste
llt wird)
x1_range = np.linspace(-3, 3, 100)
x2_range = np.linspace(-3, 3, 100)
xx1, xx2 = np.meshgrid(x1_range, x2_range)
grid = np.c_[xx1.ravel(), xx2.ravel()]

# Vorhersagen für das Gitter
Z = svm_model.predict(grid)
Z = Z.reshape(xx1.shape)

# Plot der Datenpunkte und der Entscheidungsfläche
plt.figure(figsize=(8, 6))

# Plot der Entscheidungsebene
plt.contourf(xx1, xx2, Z, alpha=0.3, cmap='coolwarm', levels=np.arange(-1.5,
1.5, 1))

# Plot der Datenpunkte
plt.scatter(X[y == 1, 0], X[y == 1, 1], color='red', label='Klasse +1')
plt.scatter(X[y == -1, 0], X[y == -1, 1], color='blue', label='Klasse -1')

# Plot der Support-Vektoren
plt.scatter(svm_model.support_vectors_[:, 0], svm_model.support_vectors_[:,
1],
            s=100, facecolors='none', edgecolors='k', label='Support-Vektore
n')

# Achsenbeschriftungen und Titel
plt.xlim(-3, 3)
plt.ylim(-3, 3)
plt.axhline(0, color='black', linewidth=0.5)
plt.axvline(0, color='black', linewidth=0.5)
plt.xlabel('$x_1$')
plt.ylabel('$x_2$')
plt.title('SVM-Trennlinie mit polynomialem Kernel (Grad 2)')
plt.legend()
plt.grid(True)
plt.show()
```


SVM-Trennlinie mit polynomialem Kernel (Grad 2)
Klasse +1
Klasse -1
Support-Vektoren
x_1
x_2
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3