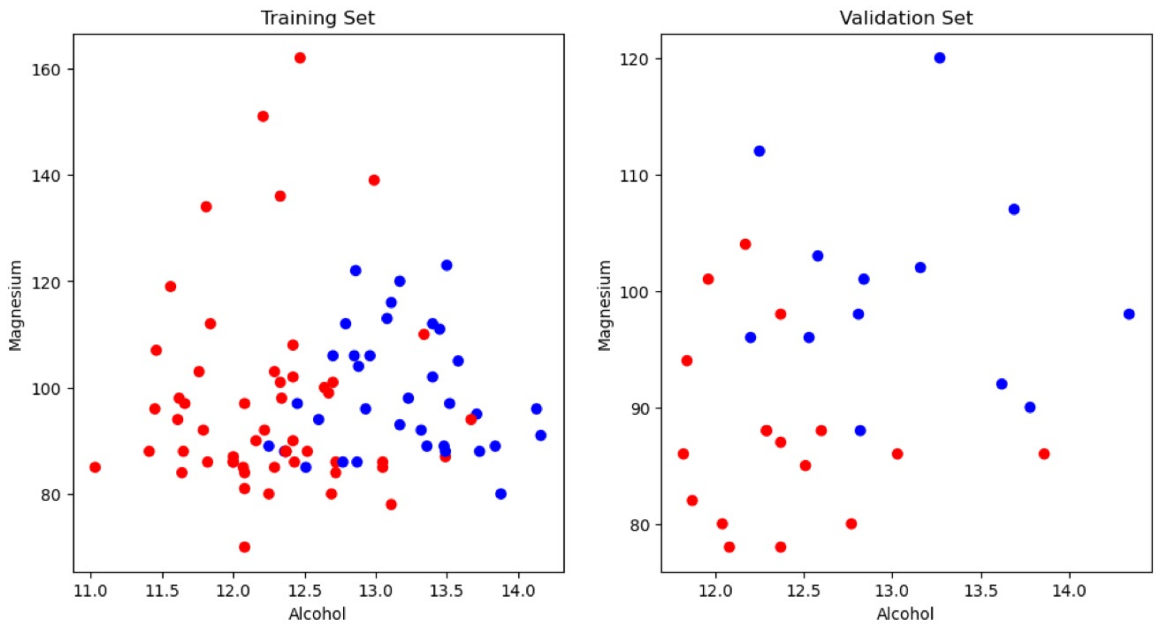
In [1]:

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split

import matplotlib.pyplot as plt
from sklearn.svm import SVC
import seaborn as sns
```

Question 1.1

In [2]:

```
# Read the wine dataset
dataset = load_wine()
df = pd.DataFrame(data=dataset['data'], columns=dataset['feature_names'])
df = df.assign(target=pd.Series(dataset['target']).values)

# Filter the irrelevant columns
df = df[['alcohol', 'magnesium', 'target']]
# Filter the irrelevant label
df = df[df.target != 0]

train_df, val_df = train_test_split(df, test_size=30, random_state=3)

# Define colors for wineries
colors = {1: 'red', 2: 'blue'}

# Create scatter plots
fig, axes = plt.subplots(1, 2, figsize=(12, 6))

# Training set scatter plot
axes[0].scatter(train_df['alcohol'], train_df['magnesium'],
                c=train_df['target'].map(colors))
axes[0].set_title('Training Set')
axes[0].set_xlabel('Alcohol')
axes[0].set_ylabel('Magnesium')

# Validation set scatter plot
axes[1].scatter(val_df['alcohol'], val_df['magnesium'],
                c=val_df['target'].map(colors))
axes[1].set_title('Validation Set')
axes[1].set_xlabel('Alcohol')
axes[1].set_ylabel('Magnesium')

plt.show()
```



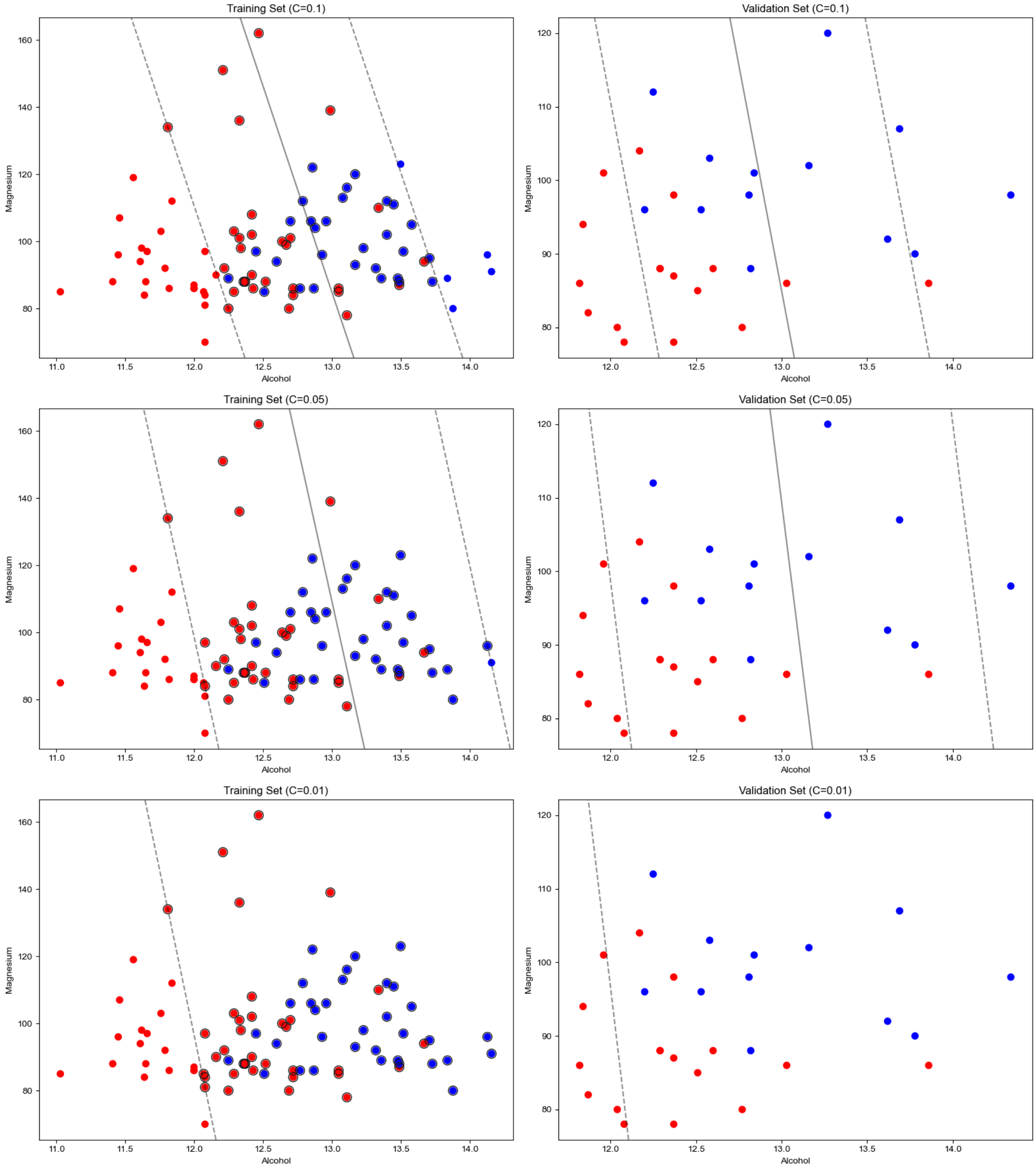
As we can see in the above plot, the data cannot be linearly separated in perfect manner. If the data is not linearly separable, running Hard-SVM will result in the model failing to find a suitable separating hyperplane. This is because Hard-SVM requires the data to be perfectly

Question 1.2

In [3]:
```
from sklearn.svm import SVC
```

In [4]:
```
# Function to plot the decision boundary
def plot_svc_decision_function(model, X, y, ax=None, plot_support=True):
    """Plot the decision function for a 2D SVC"""
    if ax is None:
        ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # Create grid to evaluate model
    x = np.linspace(xlim[0], xlim[1], 30)
    y = np.linspace(ylim[0], ylim[1], 30)
    Y, X_grid = np.meshgrid(y, x)
    xy = np.vstack([X_grid.ravel(), Y.ravel()]).T
    xy_df = pd.DataFrame(xy, columns=['alcohol', 'magnesium'])
    P = model.decision_function(xy_df).reshape(X_grid.shape)

    # Plot decision boundary and margins
    ax.contour(X_grid, Y, P, colors='k',
               levels=[-1, 0, 1], alpha=0.5,
               linestyles=['--', '-', '--'])

    # Plot support vectors
    if plot_support:
        ax.scatter(model.support_vectors_[:, 0],
                   model.support_vectors_[:, 1],
                   s=100, linewidth=1, facecolors='none', edgecolors='k')
    ax.set_xlim(xlim)
    ax.set_ylim(ylim)

# Train SVM models with different C values
C_values = [0.1, 0.05, 0.01]
models = [SVC(kernel='linear', C=C) for C in C_values]

# Fit the models
for model in models:
    model.fit(train_df[['alcohol', 'magnesium']], train_df['target'])

# Create plots
fig, axes = plt.subplots(3, 2, figsize=(16, 18))
sns.set()

for i, (model, C) in enumerate(zip(models, C_values)):
    # Training set scatter plot
    ax = axes[i, 0]
    ax.scatter(train_df['alcohol'], train_df['magnesium'],
               c=train_df['target'].map(colors), s=50)
    plot_svc_decision_function(model,
                               train_df[['alcohol', 'magnesium']], train_df
    ax.set_title(f'Training Set (C={C})')
    ax.set_xlabel('Alcohol')
    ax.set_ylabel('Magnesium')

    # Validation set scatter plot
    ax = axes[i, 1]
    ax.scatter(val_df['alcohol'], val_df['magnesium'],
               c=val_df['target'].map(colors), s=50)
    plot_svc_decision_function(model, val_df[['alcohol', 'magnesium']],
                               val_df['target'], ax=ax, plot_support=False)
    ax.set_title(f'Validation Set (C={C})')
    ax.set_xlabel('Alcohol')
    ax.set_ylabel('Magnesium')
```

```
plt.tight_layout()
plt.show()
```



Question 1.3

כפי שראינו בהרצאה, בעיית האופטימיזציה הינה:

$$\min_{w} \|w\|^2 \quad s.t. \ \forall i, \ y_i\langle w, x_i\rangle \geq 1$$

כאשר ה-margin מוגדר כ: $\min_{i\in[m]} |\langle \hat{w}, x_i\rangle|$

$$\Rightarrow \quad \min_{i\in[m]} |\langle \hat{w}, x_i\rangle| = \min_{i\in[m]} \left|\left\langle \frac{w_0}{\|w_0\|}, x_i\right\rangle\right| = \frac{1}{\|w_0\|} \min |\langle w_0, x_i\rangle|$$

מהנתון $\hat{w} = \frac{w_0}{\|w_0\|}$

סקלר יוצא החוצה

וקטור $(u,y)\in S$ המקיים $|\langle w_0, u\rangle| = \min_{i\in[m]} |\langle w_0, x_i\rangle|$ הינו support vector לפי ההגדרה.

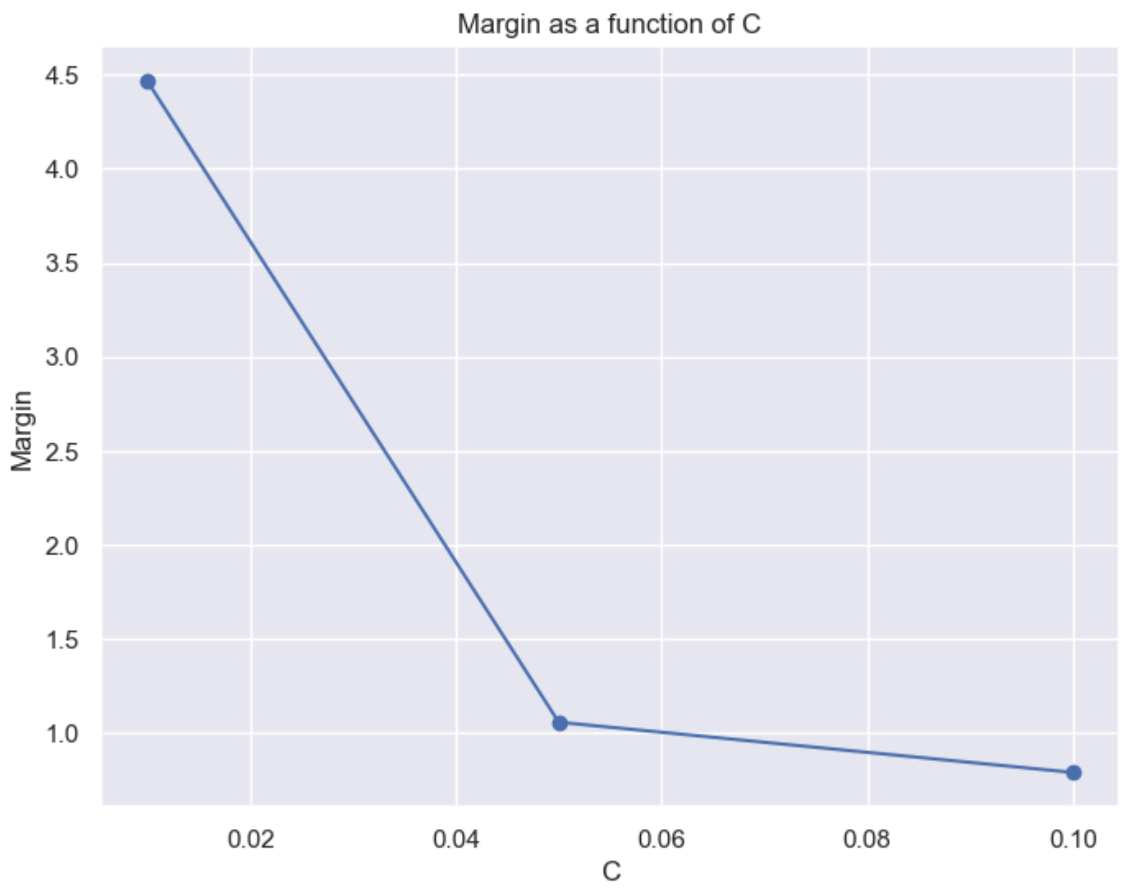
ראינו בהרצאה כי support vectors מקיימים $|\langle w_0, u\rangle|$ שכן הם נמצאים במרחק המינ' מ-w_0 אשר הינו 1.

מכאן שמתקיים:

$$\min_{i\in[m]} |\langle \hat{w}, x_i\rangle| = \frac{1}{\|w_0\|} \min_{i\in[m]} |\langle w_0, x_i\rangle| = \frac{1}{\|w_0\|} |\langle w_0, u\rangle| = \frac{1}{\|w_0\|}$$

Question 1.4

In [5]:
```
# Fit the models
for model in models:
    model.fit(train_df[['alcohol', 'magnesium']], train_df['target'])

# Function to calculate margin from SVC model
def calculate_margin(model):
    w_norm = np.linalg.norm(model.coef_)
    return 1 / w_norm

# Calculate margins for each model
margins = [calculate_margin(model) for model in models]

# Plot margin as a function of C
plt.figure(figsize=(8, 6))
plt.plot(C_values, margins, marker='o', linestyle='-', color='b')
plt.xlabel('C')
plt.ylabel('Margin')
plt.title('Margin as a function of C')
plt.grid(True)
plt.show()
```

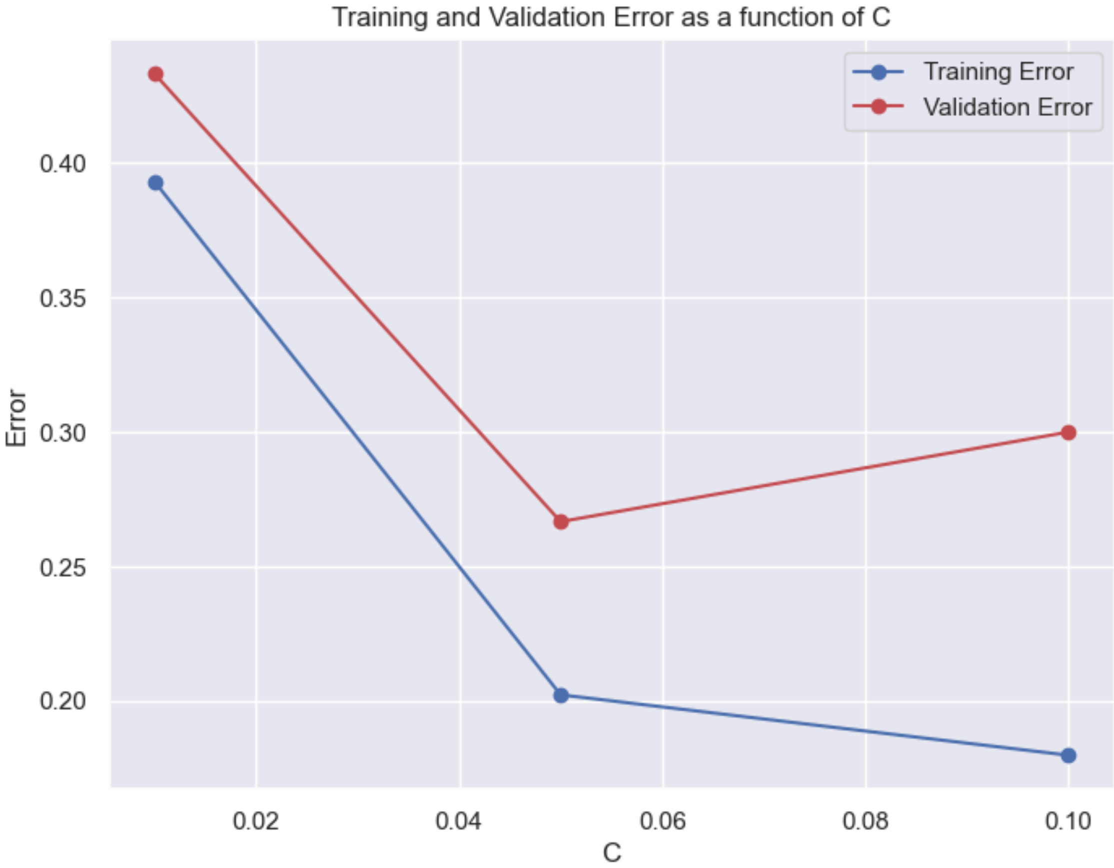


We can see from the plot above that while C grows the margin narrows. In the optimization problem we have two parameters:

1. ||W|| which affects the width of the margins.
2. The hinge loss * C. when C grows we would want to have a less training errors, since we give more weight to each mistake. Therefore the margin narrows and allow for less mistakes to be made. On the other hand, as C gets smaller - the margin widens and the model generalized better.

Question 1.5:

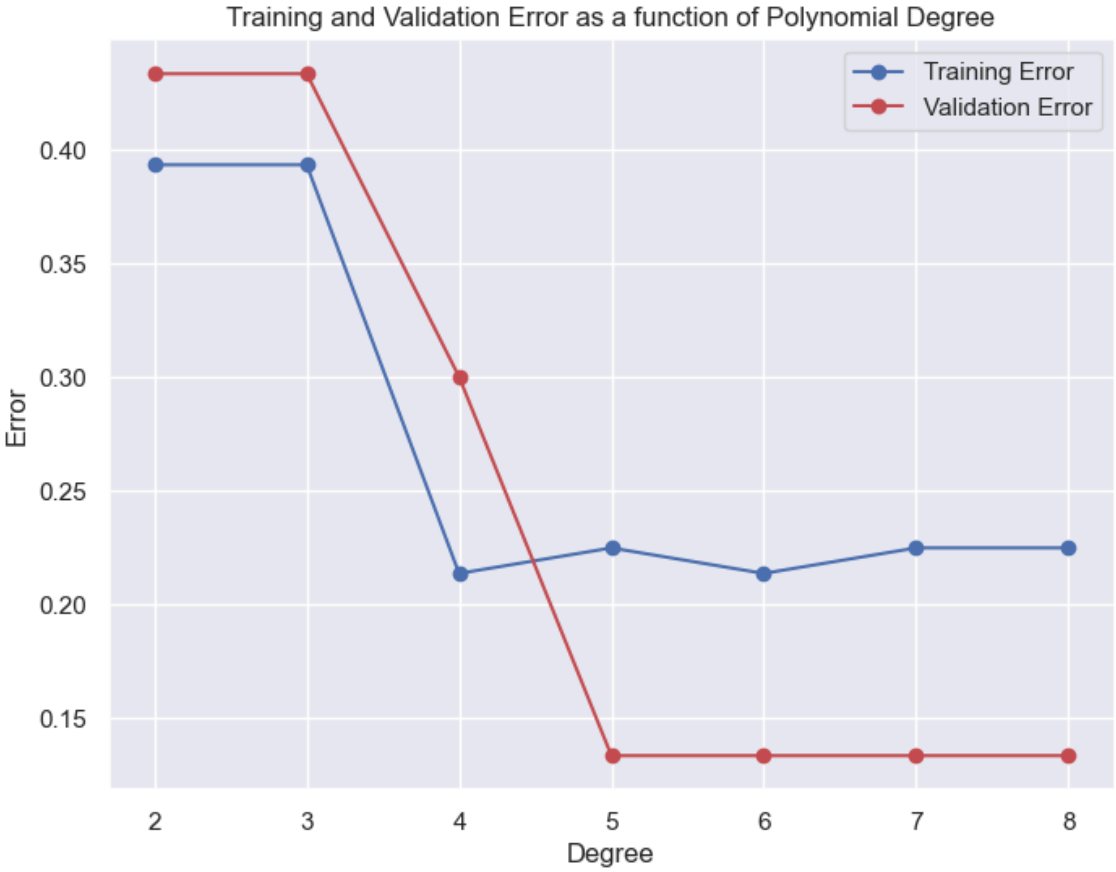
In [6]:
```
from sklearn.metrics import accuracy_score

# Train SVM models with different C values
C_values = [0.1, 0.05, 0.01]
models = [SVC(kernel='linear', C=C) for C in C_values]

# Initialize lists to store errors
train_errors = []
val_errors = []

# Fit the models and compute errors
for model in models:
    model.fit(train_df[['alcohol', 'magnesium']], train_df['target'])

    # Predictions on training set
    train_pred = model.predict(train_df[['alcohol', 'magnesium']])
    train_accuracy = accuracy_score(train_df['target'], train_pred)
    train_error = 1.0 - train_accuracy
    train_errors.append(train_error)

    # Predictions on validation set
    val_pred = model.predict(val_df[['alcohol', 'magnesium']])
    val_accuracy = accuracy_score(val_df['target'], val_pred)
    val_error = 1.0 - val_accuracy
    val_errors.append(val_error)

# Plot errors as a function of C
plt.figure(figsize=(8, 6))
plt.plot(C_values, train_errors, marker='o', linestyle='-', color='b',
         label='Training Error')
plt.plot(C_values, val_errors, marker='o', linestyle='-', color='r',
         label='Validation Error')
plt.xlabel('C')
plt.ylabel('Error')
plt.title('Training and Validation Error as a function of C')
plt.grid(True)
plt.legend()
plt.show()
```



As we can observe in the plot above, both training and validation error are decreasing until C=0.05, moreover - the error value of the validation set is higher than the training set. for C larger than 0.05 we can see that as we expected for the last question (1.4) the margin narrows which may result in overfitting of the training set which in turn causes a higher error values in the validation set.

Question 1.6

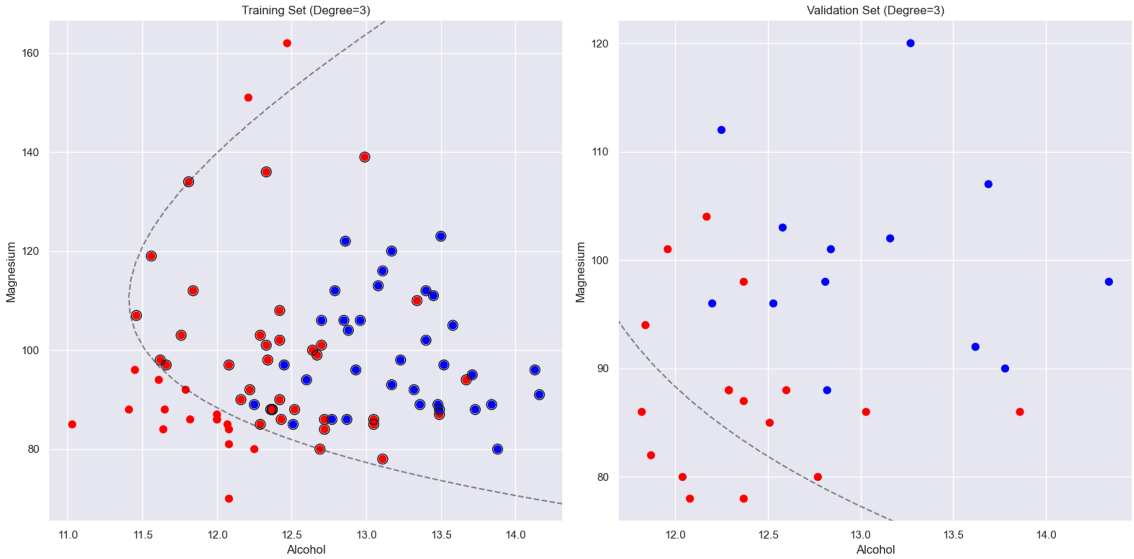
In [7]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Define the degree range and model parameter C
degrees = range(2, 9)
C = 1

# Initialize lists to store errors
train_errors = []
val_errors = []

# Fit the models and compute errors
for degree in degrees:
    model = SVC(kernel='poly', degree=degree, C=C)
    model.fit(train_df[['alcohol', 'magnesium']], train_df['target'])

    # Predictions on training set
    train_pred = model.predict(train_df[['alcohol', 'magnesium']])
    train_accuracy = accuracy_score(train_df['target'], train_pred)
    train_error = 1.0 - train_accuracy
    train_errors.append(train_error)

    # Predictions on validation set
    val_pred = model.predict(val_df[['alcohol', 'magnesium']])
    val_accuracy = accuracy_score(val_df['target'], val_pred)
    val_error = 1.0 - val_accuracy
    val_errors.append(val_error)

# Plot errors as a function of polynomial degree
plt.figure(figsize=(8, 6))
plt.plot(degrees, train_errors, marker='o', linestyle='-', color='b',
         label='Training Error')
plt.plot(degrees, val_errors, marker='o', linestyle='-', color='r',
         label='Validation Error')
plt.xlabel('Degree')
plt.ylabel('Error')
plt.title('Training and Validation Error as a function of Polynomial Degree
plt.legend()
plt.show()
```



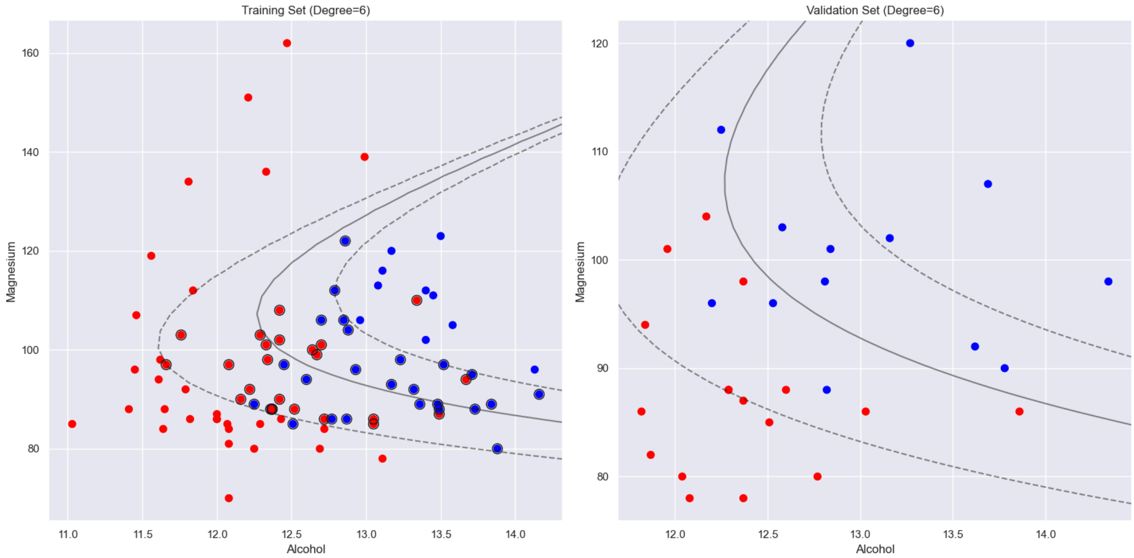
As observed in the plot above, when moving to a higher dimension the error value decreases for both training and validation sets, and specificaly for the validation set - starting from 5D we can assume that the data is almost linearly seperable as the error value is the lowest.

Question 1.7

In [12]:
```
# Function to plot the decision boundary
def plot_svc_decision_function(model, X, y, ax=None, plot_support=True):
    if ax is None:
        ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # Create grid to evaluate model
    x = np.linspace(xlim[0], xlim[1], 30)
    y = np.linspace(ylim[0], ylim[1], 30)
    Y, X_grid = np.meshgrid(y, x)
    xy = np.vstack([X_grid.ravel(), Y.ravel()]).T
    xy_df = pd.DataFrame(xy, columns=['alcohol', 'magnesium'])
    P = model.decision_function(xy_df).reshape(X_grid.shape)

    # Plot decision boundary and margins
    ax.contour(X_grid, Y, P, colors='k', levels=[-1, 0, 1], alpha=0.5, line

    # Plot support vectors
    if plot_support:
        ax.scatter(model.support_vectors_[:, 0],
                   model.support_vectors_[:, 1],
                   s=100, linewidth=1, facecolors='none', edgecolors='k')
    ax.set_xlim(xlim)
    ax.set_ylim(ylim)

# Train SVM model with polynomial kernel of degree 3
model = SVC(kernel='poly', degree=3, C=1)
model.fit(train_df[['alcohol', 'magnesium']], train_df['target'])

# Create plots
fig, axes = plt.subplots(1, 2, figsize=(16, 8))
sns.set()

# Training set scatter plot
ax = axes[0]
ax.scatter(train_df['alcohol'], train_df['magnesium'],
           c=train_df['target'].map(colors), s=50)
plot_svc_decision_function(model, train_df[['alcohol', 'magnesium']],
                           train_df['target'], ax=ax, plot_support=True)
ax.set_title('Training Set (Degree=3)')
ax.set_xlabel('Alcohol')
ax.set_ylabel('Magnesium')

# Validation set scatter plot
ax = axes[1]
ax.scatter(val_df['alcohol'], val_df['magnesium'],
           c=val_df['target'].map(colors), s=50)
plot_svc_decision_function(model, val_df[['alcohol', 'magnesium']],
                           val_df['target'], ax=ax, plot_support=False)
ax.set_title('Validation Set (Degree=3)')
ax.set_xlabel('Alcohol')
ax.set_ylabel('Magnesium')

plt.tight_layout()
plt.show()
```


Training Set (Degree=3)
Magnesium
Alcohol
160
140
120
100
80
11.0
11.5
12.0
12.5
13.0
13.5
14.0
Validation Set (Degree=3)
Magnesium
Alcohol
120
110
100
90
80
12.0
12.5
13.0
13.5
14.0

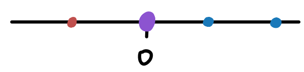
In [13]:
```
# Function to plot the decision boundary
def plot_svc_decision_function(model, X, y, ax=None, plot_support=True):
    if ax is None:
        ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # Create grid to evaluate model
    x = np.linspace(xlim[0], xlim[1], 30)
    y = np.linspace(ylim[0], ylim[1], 30)
    Y, X_grid = np.meshgrid(y, x)
    xy = np.vstack([X_grid.ravel(), Y.ravel()]).T
    xy_df = pd.DataFrame(xy, columns=['alcohol', 'magnesium'])
    P = model.decision_function(xy_df).reshape(X_grid.shape)

    # Plot decision boundary and margins
    ax.contour(X_grid, Y, P, colors='k', levels=[-1, 0, 1],
               alpha=0.5, linestyles=['--', '-', '--'])

    # Plot support vectors
    if plot_support:
        ax.scatter(model.support_vectors_[:, 0],
                   model.support_vectors_[:, 1],
                   s=100, linewidth=1, facecolors='none', edgecolors='k')
    ax.set_xlim(xlim)
    ax.set_ylim(ylim)

# Train SVM model with polynomial kernel of degree 6
model = SVC(kernel='poly', degree=6, C=1)
model.fit(train_df[['alcohol', 'magnesium']], train_df['target'])

# Create plots
fig, axes = plt.subplots(1, 2, figsize=(16, 8))
sns.set()

# Training set scatter plot
ax = axes[0]
ax.scatter(train_df['alcohol'], train_df['magnesium'],
           c=train_df['target'].map(colors), s=50)
plot_svc_decision_function(model, train_df[['alcohol', 'magnesium']],
                           train_df['target'], ax=ax, plot_support=True)
ax.set_title('Training Set (Degree=6)')
ax.set_xlabel('Alcohol')
ax.set_ylabel('Magnesium')

# Validation set scatter plot
ax = axes[1]
ax.scatter(val_df['alcohol'], val_df['magnesium'],
           c=val_df['target'].map(colors), s=50)
plot_svc_decision_function(model, val_df[['alcohol', 'magnesium']],
                           val_df['target'], ax=ax, plot_support=False)
ax.set_title('Validation Set (Degree=6)')
ax.set_xlabel('Alcohol')
ax.set_ylabel('Magnesium')

plt.tight_layout()
plt.show()
```


Training Set (Degree=6)
Magnesium
Alcohol
160
140
120
100
80
11.0
11.5
12.0
12.5
13.0
13.5
14.0
Validation Set (Degree=6)
Magnesium
Alcohol
120
110
100
90
80
12.0
12.5
13.0
13.5
14.0

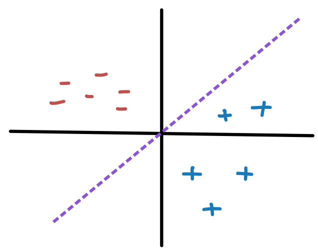
שאלה 3:

1. ניקח דוגמה כאשר $bias=0$. במקרה זה, המפריד הלינארי יעבור בראשית הצירים, כלומר הפרדה לינארית של הדאטה צריכה להתקיים ביחס לראשית הצירים.

<u>עבור $d=1$:</u>



<u>עבור $d=2$:</u>



2. נסמן: $x_2=(0,q)\ ,\ y_2=1$ $\qquad x_1=(p,0),\ y_1=-1$

מהאילוצים נקבל: $y_1\langle w,(p,0)\rangle = -w_1p \ge 1 \Longrightarrow w_1p \le -1$

$$y_2\langle w,(0,q)\rangle = w_2\cdot q \ge 1 \Longrightarrow w_2q \ge 1$$

נשים לב כי על מנת שהדאטה יהיה פריד לינארית ביחס לראשית הצירים, נדרש שאף אחת מהנקודות לא תמצא ב-$(0,0)$, ולכן לכל מקרה בו $p\cdot q=0$, הדאטה לא יהיה פריד לינארית, ואז לא יהיה פתרון ל-hard SVM.

נחלק למקרים:

$$\underline{p>0:}\ \begin{cases} w_1 \le -\frac{1}{p} \\ w_1 \ge -\frac{1}{p} \end{cases} \Longrightarrow |w_1| \ge \left|\frac{-1}{p}\right|$$

$$\begin{cases} w_2 \ge \frac{1}{q} & \underline{:q>0} \\ w_2 \le \frac{1}{q} & \underline{:q<0} \end{cases} \Longrightarrow |w_2| \ge \left|\frac{1}{q}\right|$$

$w_1 + w_2$

ראינו בהרצאה כי:

$$\|w\|^2 = w_1^2+w_2^2 = |w_1|^2+|w_2|^2 \ge \left|-\frac{1}{p}\right|^2 + \left|\frac{1}{q}\right|^2 = \frac{1}{p^2}+\frac{1}{q^2}$$

נבחר $\tilde{w} = (\frac{1}{p}, -\frac{1}{q})$. אילוצי הבעיה מתקיימים, שכן:

$y_1 \langle \tilde{w}, (p,0) \rangle = -(-\frac{1}{p}) \cdot p = 1 \geq 1$ ✓

$y_2 \langle \tilde{w}, (0,q) \rangle = \frac{1}{q} \cdot q = 1 \geq 1$ ✓

בנוסף, $\|\tilde{w}\|^2 = \tilde{w}_1 + \tilde{w}_2 = \frac{1}{p^2} + \frac{1}{q^2}$, ולכן $\tilde{w}$ מקיים את החסם התחתון ומקיים $\tilde{w} = \underset{w}{\arg\min} \|w\|^2$

ועל כן $\tilde{w}$ הוא פתרון של hard SVM.

3. המפרד אותו היינו מקבלים לא היה משתנה, היות שה-margin נקבע לפי ה-Support vectors, וכל הנקודות שנוספו לצורה נמצאות מעבר ל-margin.

4. נשים לב שככל שנגדיל את λ, ניתן חשיבות גבוהה יותר ל-margin ונחשב פחות בטעויות הסיווג. לפיכך, ככל ש-λ גדולה יותר, נשאף ל-margin ולהכללה טובה יותר של λ.

$\lambda = 200$ $\qquad\qquad$ $\lambda = 2$

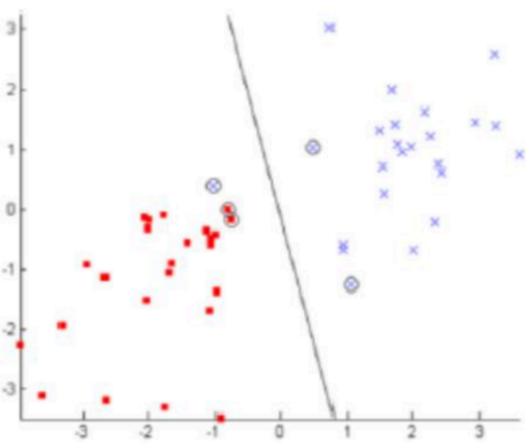


Figure 1

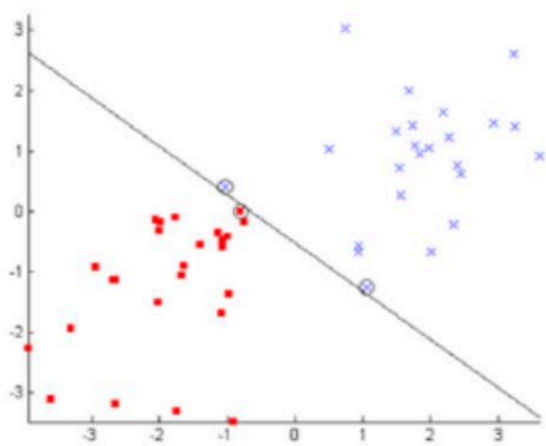


Figure 2

שאלה 4:

1. a) הפלט של האלגוריתם יהיה סקלרים $1 \le i \le M$, המייצגים את כמות העדכונים שעשינו על הדוגמה ה-i.

b) נגדיר את כלל ההחלטה, $\hat{y}(x) = \text{Sign}\left(\sum_{i=1}^{m} \alpha_i y_i K(x, x_i)\right)$

הפסאודו קוד:

$\forall i \quad \alpha_i = 0$

for $t = 1, 2, \ldots$

if $(\exists i \text{ s.t } y_i \neq \hat{y}(x_i))$

$\alpha_i = \alpha_i + 1$

else

return $[\alpha_i]_{i=1}^{m}$

2. נסמן: $w = \sum_{i=1}^{m} \alpha_i \Psi(x_i)$, וכאשר נשתמש ב-$w$ באלגוריתם הפרספטרון, נקבל את האלגוריתם החדש שבנינו.
Ψ לא ידועה, ולכן נוכל להשתמש ב-w רק כשהוא נמצא בתוך מכפלה פנימית עם דוגמה אחרת ו-K נתון.
לאורך הוכחה זו השתמשנו רק במכפלות פנימיות של w עם וקטורים אחרים, ולכן בהינתן K, נוכל לחשב גם את המכפלות הללו. ניתן לוודא שכל המעברים הנ"ל תקפים עבור הגדרה זו של w. לפיכך, נכונות אלגוריתם הפרספטרון מתקיים כשהאלגוריתם המוצע מתכנס. $\iff$ S^{Ψ} פריד לינארית ב-F.