Question 1

In [59]:

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

In [60]:

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

In [61]:

```
df = df[['alcohol', 'magnesium', 'target']]
df = df[df.target != 0]

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

In [62]:

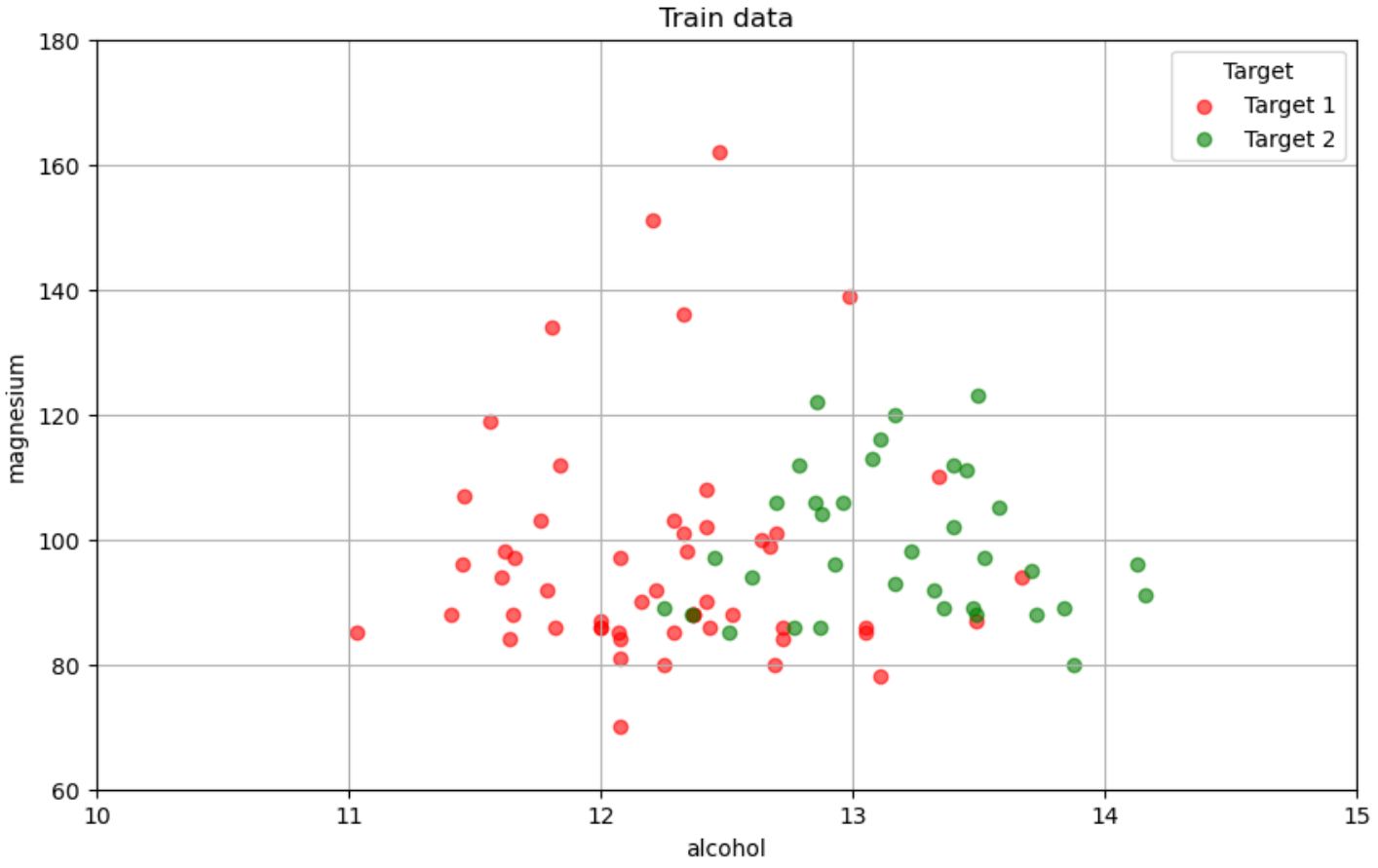
```
def scatter_targets(data, colors, title, xcolumn, ycolumn, size=(10, 6), xlim=None, ylim
=None):
    """
    Scatter data points by labels with diff colors
    :param data: data where first two columns are x and y, and third column is target
    :param colors: colors for each label
    :param title: title of the scatter plot
    :param xcolumn: x-axis column
    :param ycolumn: y-axis column
    :param xlim: Tuple of x limits
    :param ylim: Tuplem of y limits
    :return:
    """
    plt.figure(figsize=size)
    plt.title(title)
    plt.grid(True)

    plt.xlabel(xcolumn)
    plt.ylabel(ycolumn)

    if xlim:
        plt.xlim(xlim)
    if ylim:
        plt.ylim(ylim)

    for target, color in zip(data['target'].unique(), colors):
        subset = data[data['target'] == target]
        plt.scatter(subset[xcolumn], subset[ycolumn], c=color, label=f'Target {target}',
alpha=0.6)

    plt.legend(title='Target')
```

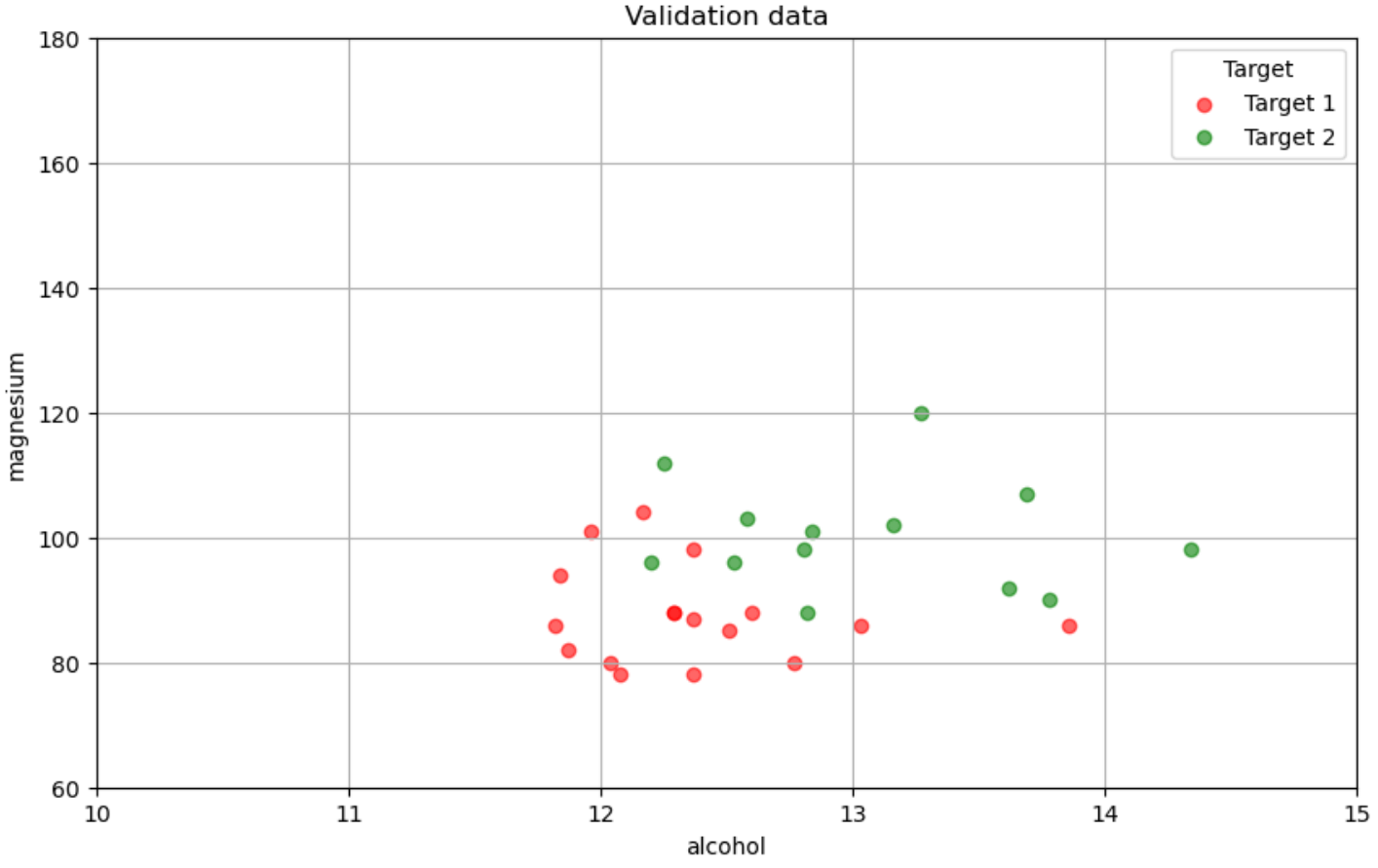
#1

In [63]:

```
# Defining limits and colors of plots
COLORS = ['red', 'green']
XLIMS = (10, 15)
```

```
YLIMS = (60, 180)

scatter_targets(train_df, COLORS, 'Train data', 'alcohol', 'magnesium', xlim=XLIMS, ylim
=YLIMS)
plt.show()

scatter_targets(val_df, COLORS, 'Validation data', 'alcohol', 'magnesium', xlim=XLIMS, y
lim=YLIMS)
plt.show()
```





Answer to #1:

Answer to #1:

hard-SVM needs data to be linearly separable, and as we can see from scatter plot, it is not the case. Thus, hard-SVM won't return us a solution.

In [64]:

```
from sklearn.svm import SVC
```

In [65]:

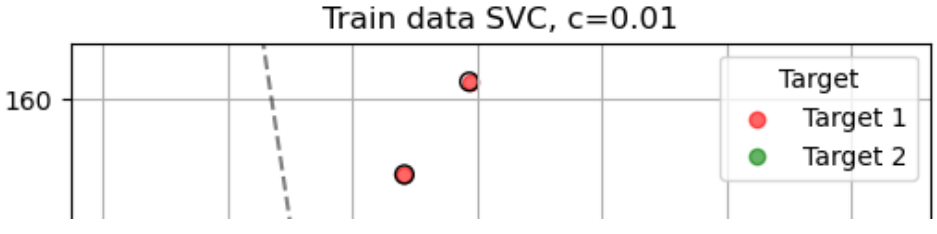
```
def plot_svc(model, show_support=False):
    """
    Plotting svm decision boundary, margins and support vectors
    """
    ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # Creating grid and plotting decision boundary and margins,
    # The same way we have seen in tutorial
    x = np.linspace(xlim[0], xlim[1], 30)
    y = np.linspace(ylim[0], ylim[1], 30)
    Y, X = np.meshgrid(y, x)
    xy = np.vstack([X.ravel(), Y.ravel()]).T
    P = model.decision_function(xy).reshape(X.shape)

    ax.contour(X, Y, P, colors='k',
           levels=[-1, 0, 1], alpha=0.5,
           linestyles=['--', '-', '--'])

    # plotting supports
    if show_support:
        ax.scatter(model.support_vectors_[:, 0], model.support_vectors_[:, 1], s=50, lin
ewidth=1, facecolors='none', edgecolor='black')

    ax.set_xlim(xlim)
    ax.set_ylim(ylim)
```

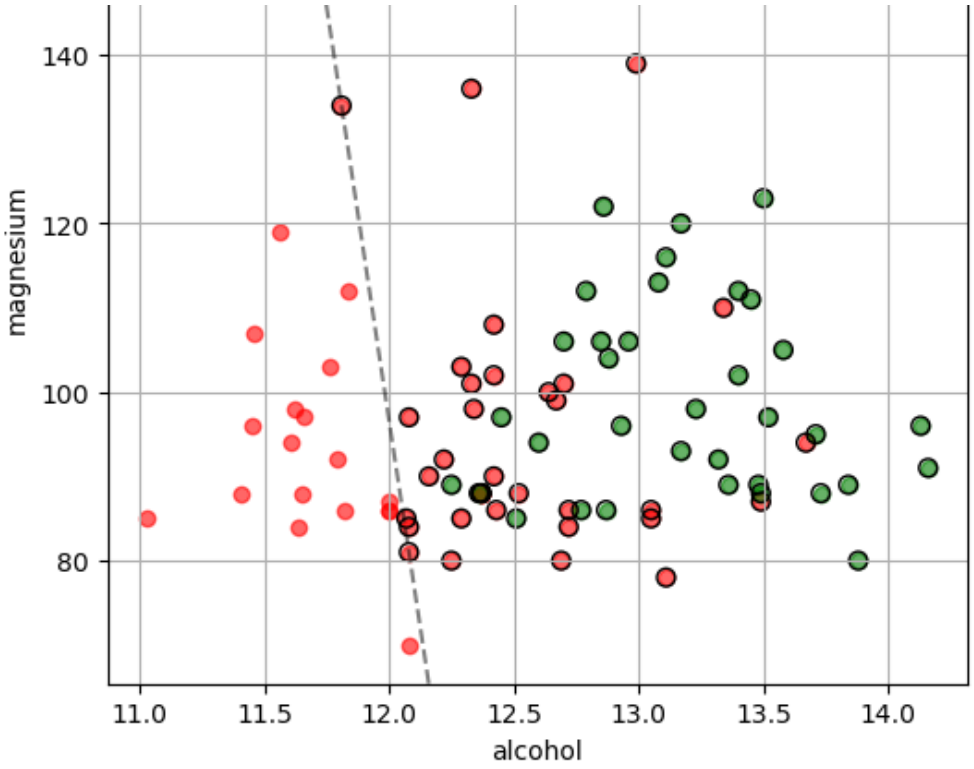
#2

In [66]:

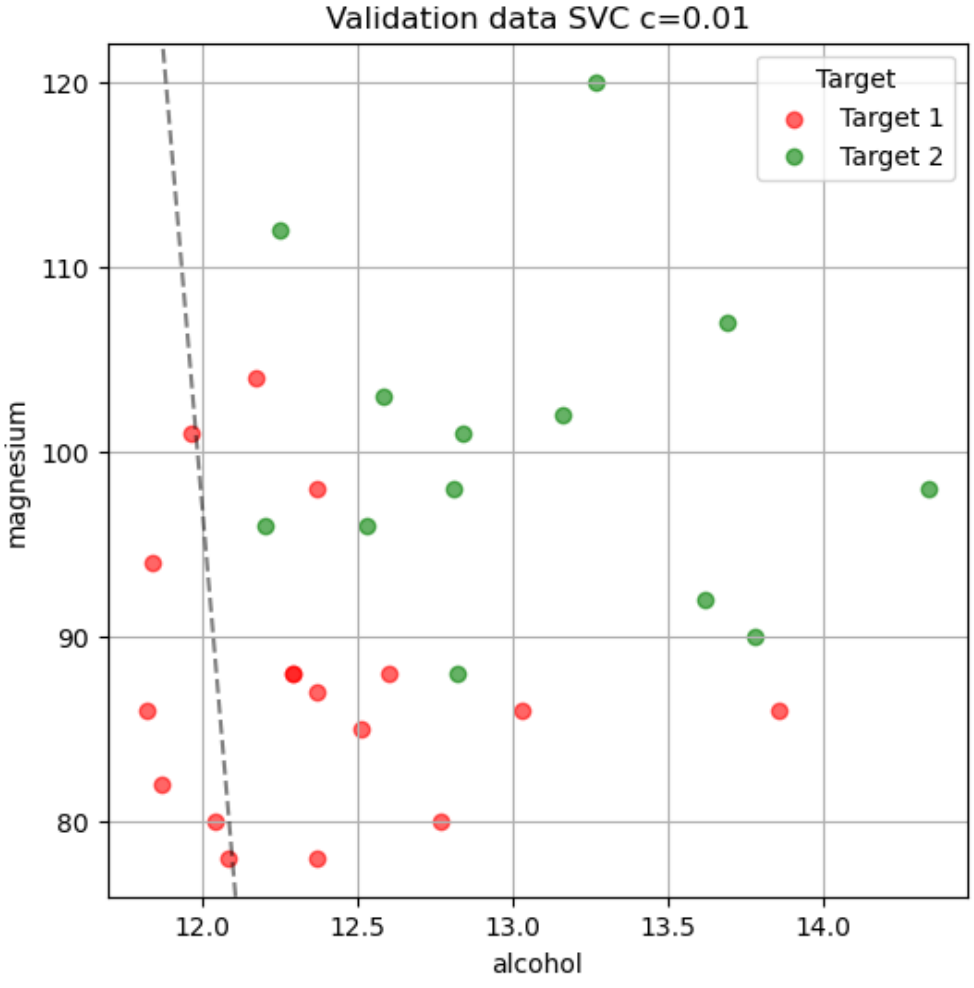
```
C = [0.01, 0.05, 0.1]

for c in C:
    model = SVC(kernel='linear', C=c)
    model.fit(train_df.drop('target', axis=1), train_df['target'])

    scatter_targets(train_df, ['red', 'green'], f'Train data SVC, c={c}', 'alcohol', 'ma
gnesium', (6, 6))
    plot_svc(model, show_support=True)
    plt.show()

    scatter_targets(val_df, ['red', 'green'], f'Validation data SVC c={c}', 'alcohol', '
magnesium', (6, 6))
    plot_svc(model)
    plt.show()
```

```
C:\Users\yumif\anaconda3\envs\mlcourse\lib\site-packages\sklearn\base.py:439: UserWarning
: X does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

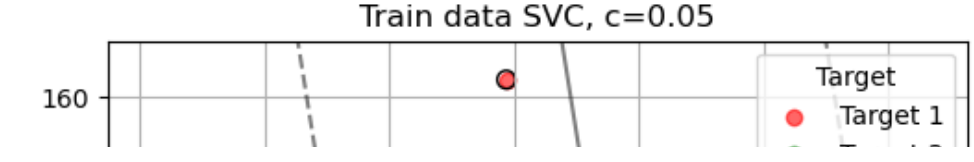


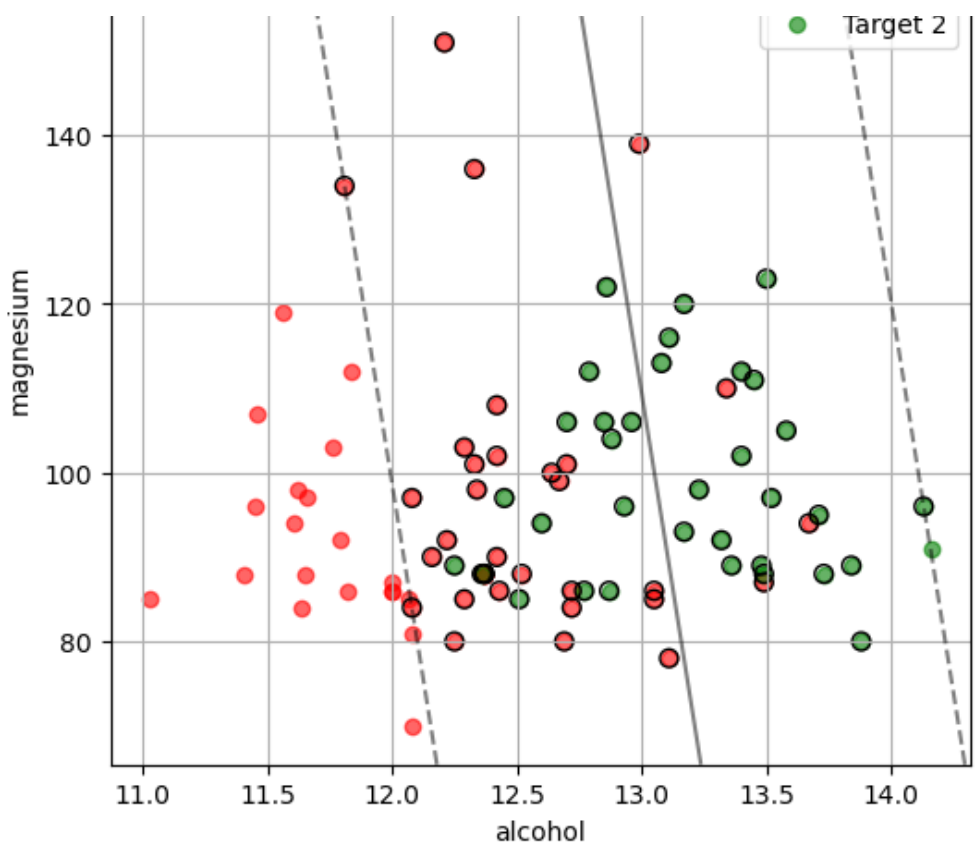



```
C:\Users\yumif\anaconda3\envs\mlcourse\lib\site-packages\sklearn\base.py:439: UserWarning
: X does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

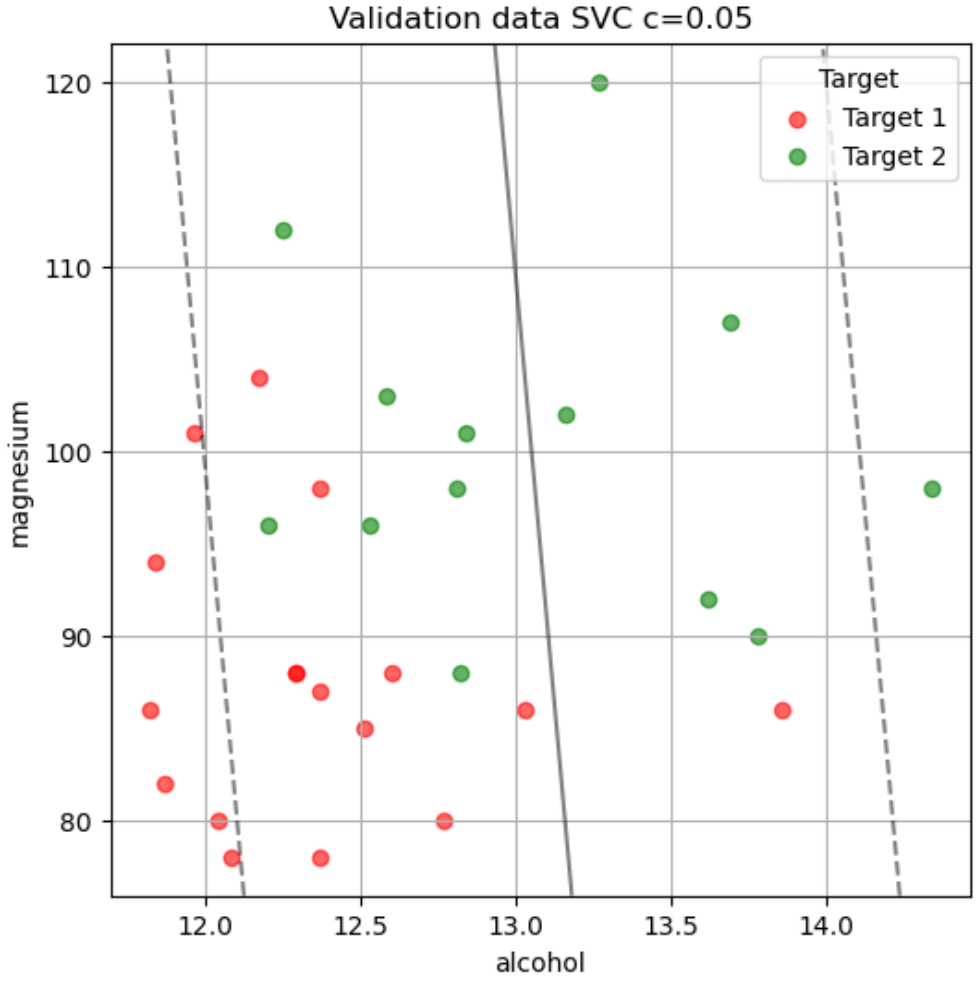



```
C:\Users\yumif\anaconda3\envs\mlcourse\lib\site-packages\sklearn\base.py:439: UserWarning
: X does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

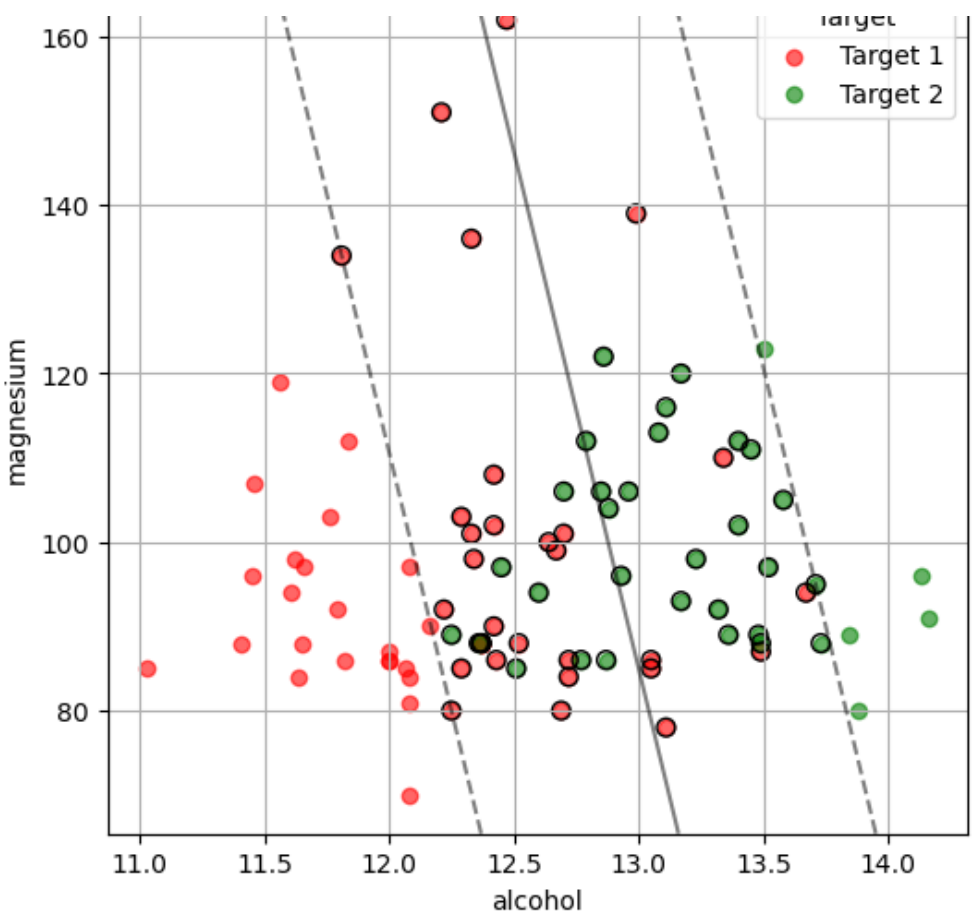




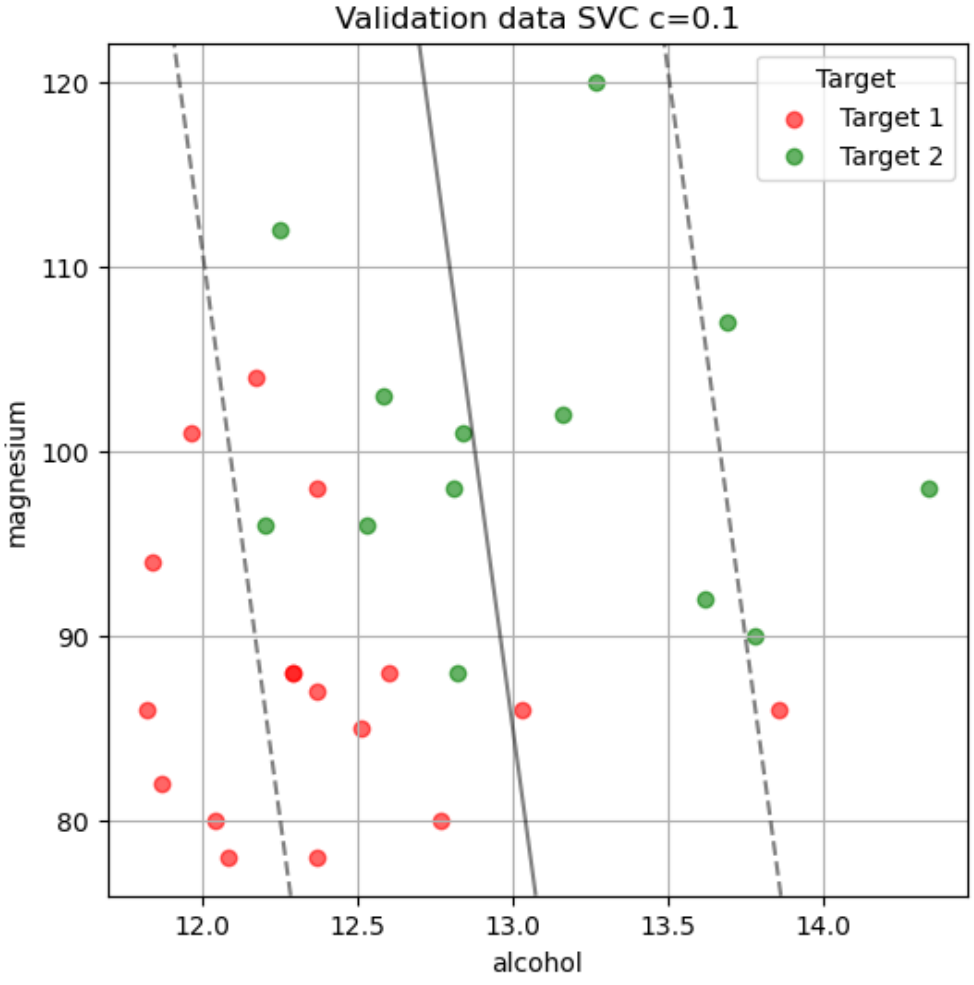

```
C:\Users\yumif\anaconda3\envs\mlcourse\lib\site-packages\sklearn\base.py:439: UserWarning
: X does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```




```
C:\Users\yumif\anaconda3\envs\mlcourse\lib\site-packages\sklearn\base.py:439: UserWarning
: X does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```




```
C:\Users\yumif\anaconda3\envs\mlcourse\lib\site-packages\sklearn\base.py:439: UserWarning
: X does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```



Let $w_0 = \arg\min_w \|w\|^2$ s.t. $\forall i, y_i \cdot \langle w, x_i \rangle \geq 1$

$$\widehat{w} = \frac{w_0}{\|w_0\|}$$

$$\min_i |\langle \widehat{w}, x_i \rangle| = \min_i \left| \left\langle \frac{w_0}{\|w_0\|}, x_i \right\rangle \right| = \frac{1}{\|w_0\|} \min_i |\langle w_0, x_i \rangle| = \frac{1}{\|w_0\|} \min_i |y_i \cdot \langle w_0, x_i \rangle| \geq \frac{1}{\|w_0\|}$$

Assume by contradiction

$$\min_i |\langle \widehat{w}, x_i \rangle| > \frac{1}{\|w_0\|}$$

Since for all i, $y_i \cdot \langle w_0, x_i \rangle \geq 1$,

$$\Rightarrow \min_i y_i \cdot \langle w_0, x_i \rangle = \min_i |y_i \cdot \langle w_0, x_i \rangle| = \min_i |\langle w_0, x_i \rangle| = \|w_0\| \min_i |\langle \widehat{w}, x_i \rangle| > 1$$

$$\Rightarrow \forall i \ \ y_i \cdot \langle w_0, x_i \rangle > 1$$

$$\Rightarrow \ \exists k > 1 \colon \forall i \ \ y_i \cdot \langle w_0, x_i \rangle \geq k > 1$$

Take $\overline{w} = \frac{w_0}{k}$, $\forall i \ \ y_i \cdot \langle \overline{w}, x_i \rangle = \ y_i \cdot \left\langle \frac{w_0}{k}, x_i \right\rangle = y_i \cdot \langle w_0, x_i \rangle \cdot \frac{1}{k} \geq \frac{k}{k} = 1$

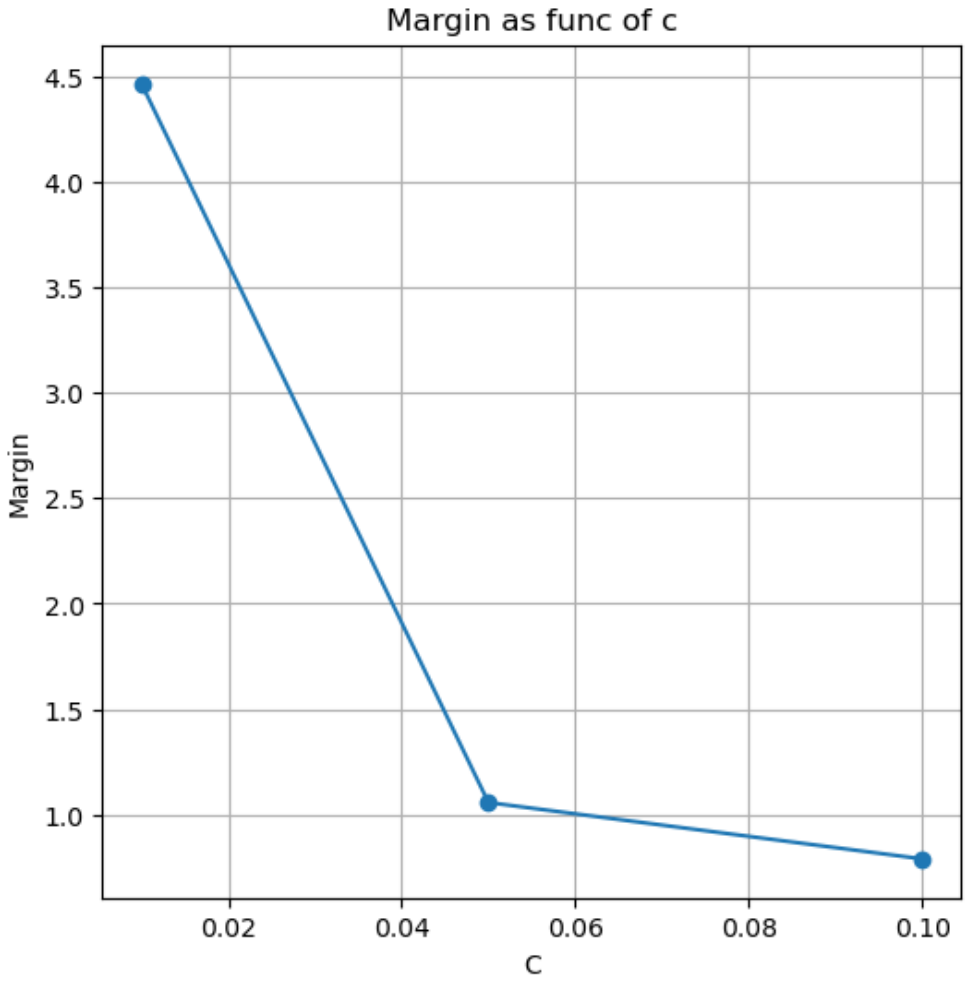
And $\|\overline{w}\|^2 = \left\| \frac{w_0}{k} \right\|^2 = \frac{\|w_0\|^2}{k^2} < \|w_0\|^2$, which is a contradiction since $w_0 = \arg\min_w \|w\|^2$ s.t. $\forall i, y_i \cdot \langle w, x_i \rangle \geq 1$

$$\Rightarrow margin = \min_i |\langle \widehat{w}, x_i \rangle| = \frac{1}{\|w_0\|}$$

#4

In [67]:

```
margins = []

for c in C:
    model = SVC(kernel='linear', C=c)
    model.fit(train_df.drop('target', axis=1), train_df['target'])

    # Calculating margin as we have seen in #3
    margins.append(1 / (np.linalg.norm(model.coef_)))

plt.figure(figsize=(6, 6))
plt.plot(C, margins, marker='o')
plt.xlabel('C')
plt.ylabel('Margin')
plt.title('Margin as func of c')
plt.grid(True)
plt.show()
```



Answer to #4:

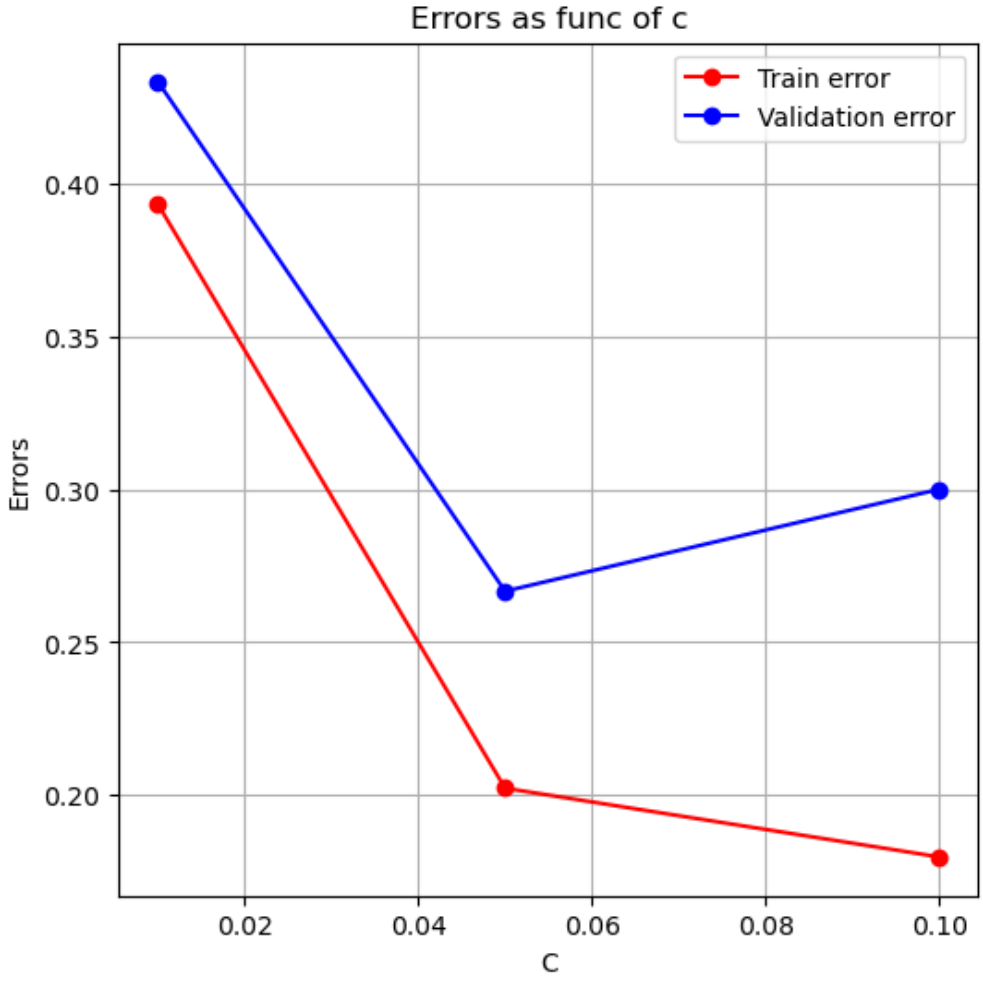
As C, the parameter of regularization grows, our margin decreases. In soft-SVM, we have a component of $||w||^2$, with which margin has an opposite relationship between (Smaller norm leads to bigger margin). The other component is our mean of hinge losses, which leads to smaller train error. For bigger values of C, we essentially make our train error (mean of hinge losses) more important than norm of w, which leads to smaller margin

#5

In [68]:

```
train_errors = []
val_errors = []
```

```
for c in C:
    model = SVC(kernel='linear', C=c)
    model.fit(train_df.drop('target', axis=1), train_df['target'])

    # Getting train/validation errors of each model by 1-score
    train_errors.append(1-model.score(train_df.drop('target', axis=1), train_df['target'
]))
    val_errors.append(1-model.score(val_df.drop('target', axis=1), val_df['target']))

plt.figure(figsize=(6, 6))
plt.plot(C, train_errors, marker='o', color='red', label='Train error')
plt.plot(C, val_errors, marker='o', color='blue', label='Validation error')
plt.xlabel('C')
plt.ylabel('Errors')
plt.title('Errors as func of c')
plt.legend()
plt.grid(True)
plt.show()
```



Answer to #5

As regularization parameter C increases, we get smaller error on train. For validation, we get smaller error and then increase of it on 0.1. From the previous section, we got that for bigger values of C, we get smaller margin, which leads to less errors on the train set.

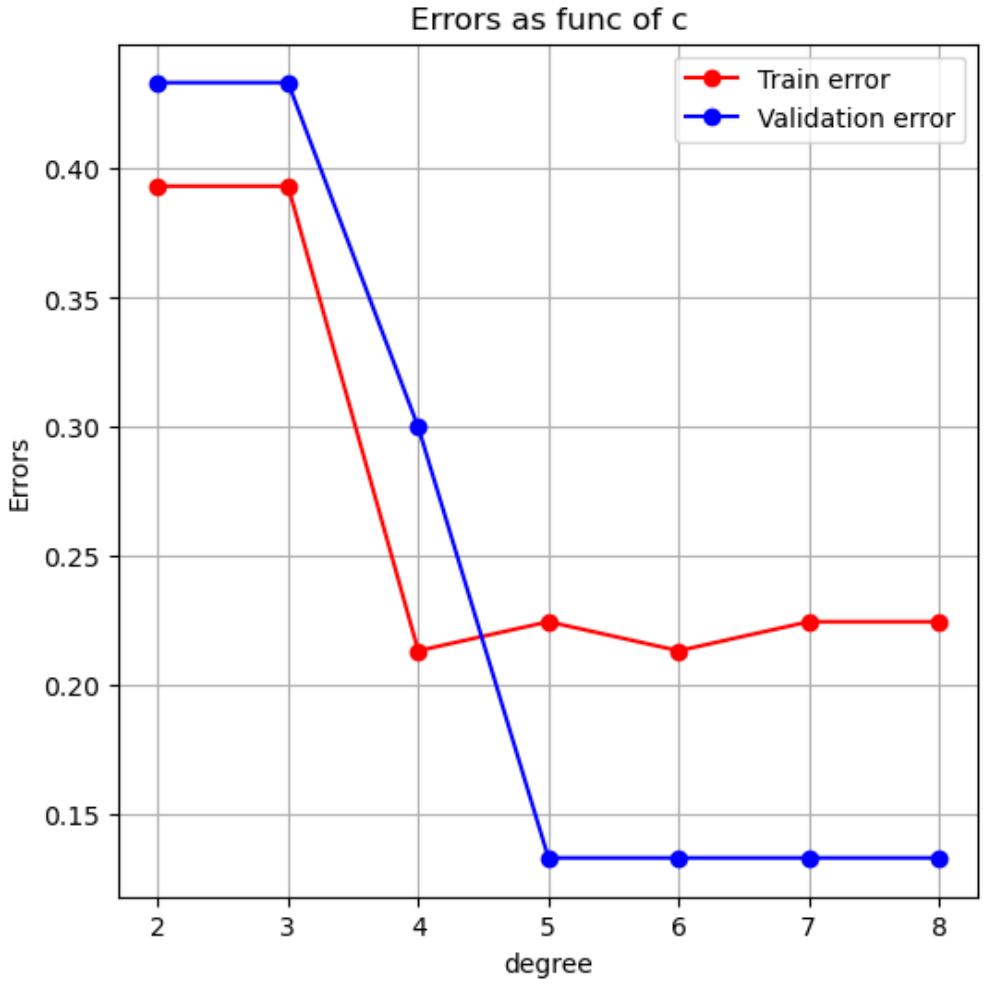
On the other hand, our validation error decreases with a slight increase from 0.05 onwards. It may be due to the fact that regularization parameter of 0.05 is better, although the margin became smaller. Usually, a larger margin (smaller c) gives a more generic model, which explains the slight increase in validation error for c = 0.1.

#6

In [69]:

```
degrees = [2, 3, 4, 5, 6, 7, 8]
```

```
train_errors = []
val_errors = []

for d in degrees:
    model = SVC(kernel='poly', degree=d, C=1.0)
    model.fit(train_df.drop('target', axis=1), train_df['target'])

    # Getting train/validation errors of each model by 1-score
    train_errors.append(1-model.score(train_df.drop('target', axis=1), train_df['target'
]))
    val_errors.append(1-model.score(val_df.drop('target', axis=1), val_df['target']))

plt.figure(figsize=(6, 6))
plt.plot(degrees, train_errors, marker='o', color='red', label='Train error')
plt.plot(degrees, val_errors, marker='o', color='blue', label='Validation error')
plt.xlabel('degree')
plt.ylabel('Errors')
plt.title('Errors as func of c')
plt.legend()
plt.grid(True)
plt.show()
```



Answer to #6

As we can see, using higher dimensions for representation of our data leads to better results. At some point, (degree=5) there is no improvement in terms of error. We believe it is due to the fact that data becomes linearly separable in higher dimensions, which leads to smaller error, and at some point we don't need to raise it to a higher dimension, since the data is already linearly separable as much as it can be.

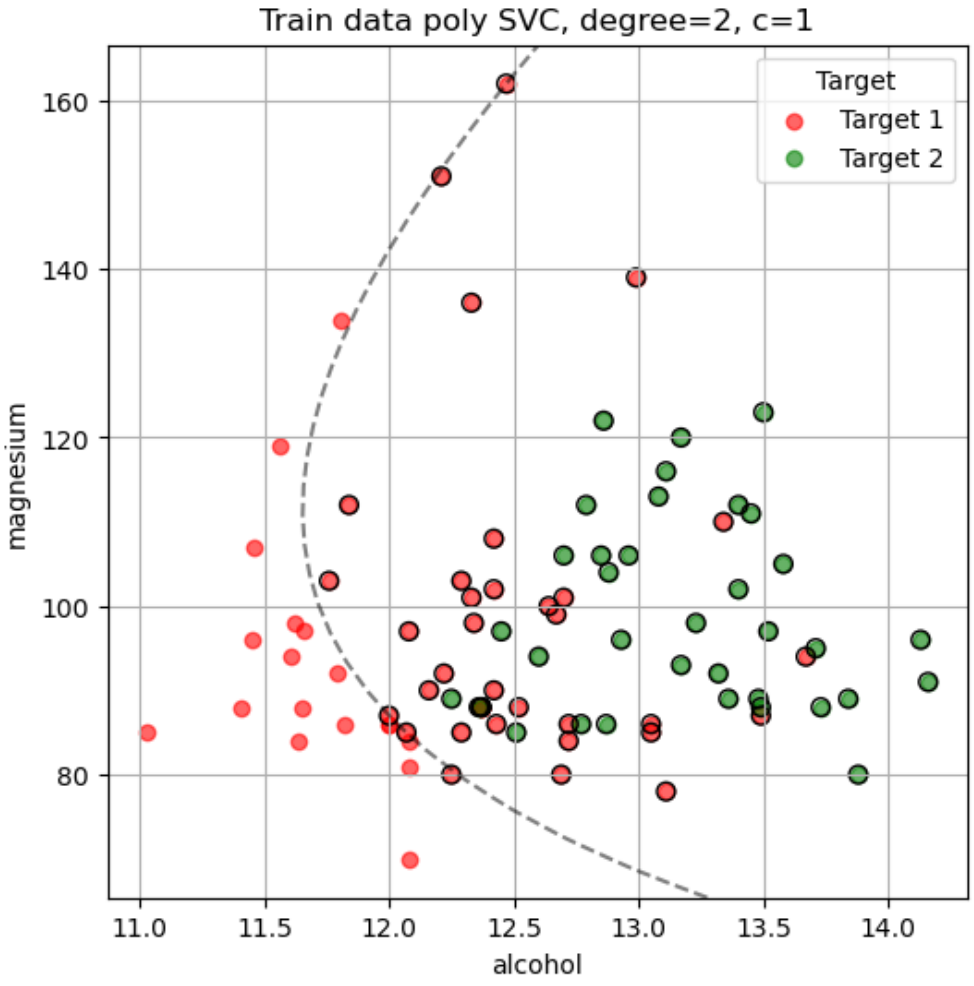
#7

In [70]:

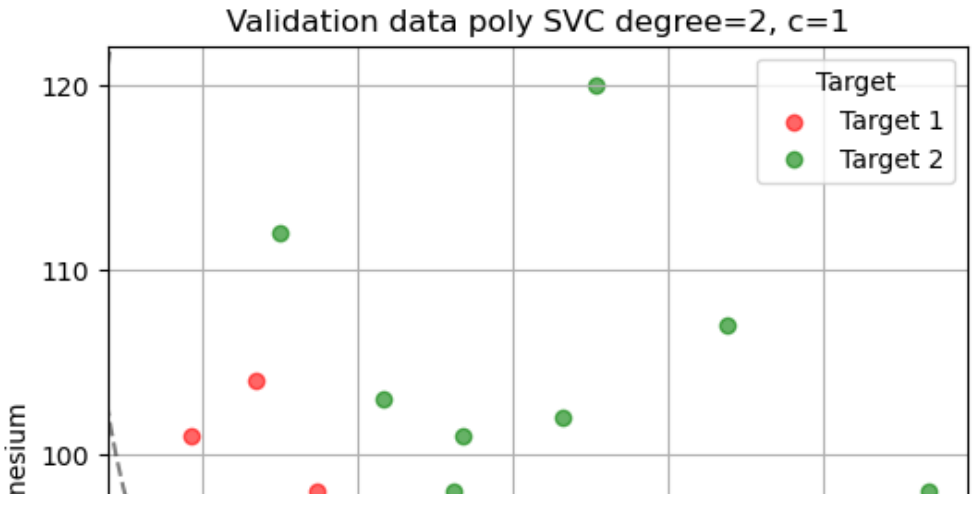
```
minmax_degrees = [2, 6]
```

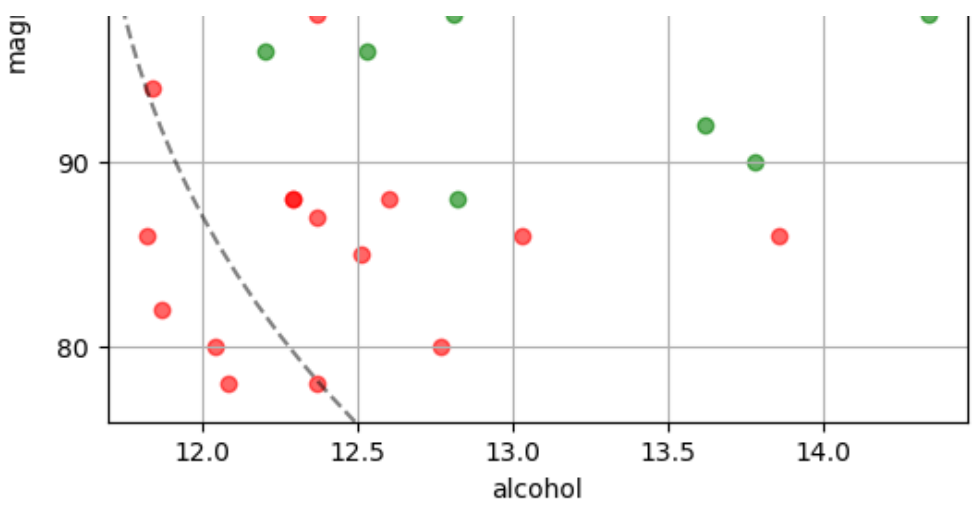
```
for d in minmax_degrees:
    model = SVC(kernel='poly', degree=d, C=1.0)
    model.fit(train_df.drop('target', axis=1), train_df['target'])

    scatter_targets(train_df, ['red', 'green'], f'Train data poly SVC, degree={d}, c=1',
'alcohol', 'magnesium', (6, 6))
    plot_svc(model, show_support=True)
    plt.show()

    scatter_targets(val_df, ['red', 'green'], f'Validation data poly SVC degree={d}, c=1
', 'alcohol', 'magnesium', (6, 6))
    plot_svc(model)
    plt.show()
```

```
C:\Users\yumif\anaconda3\envs\mlcourse\lib\site-packages\sklearn\base.py:439: UserWarning
: X does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

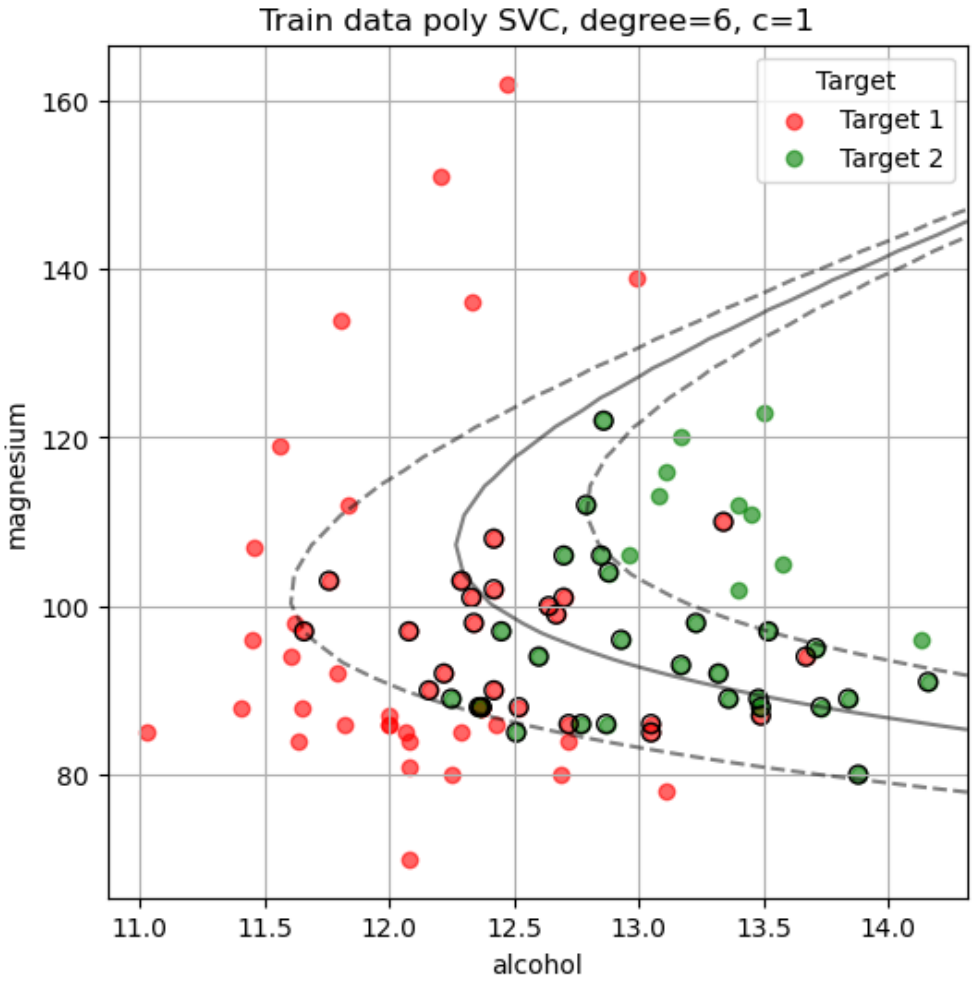



```
C:\Users\yumif\anaconda3\envs\mlcourse\lib\site-packages\sklearn\base.py:439: UserWarning
: X does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

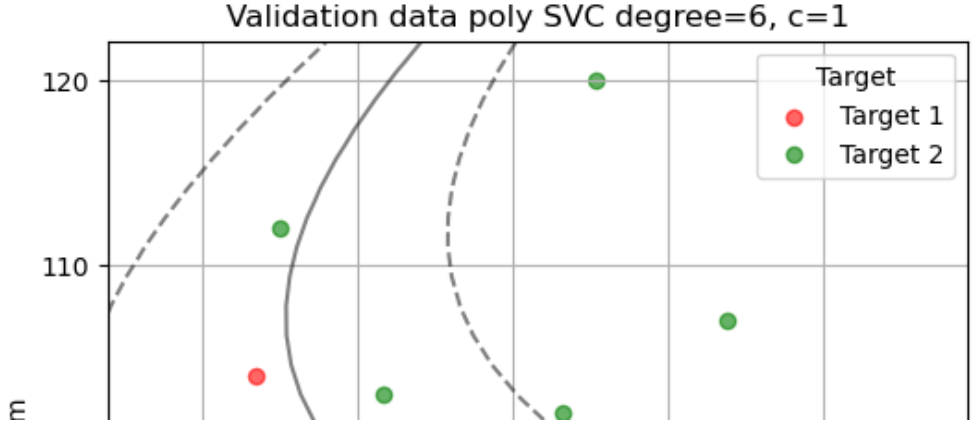


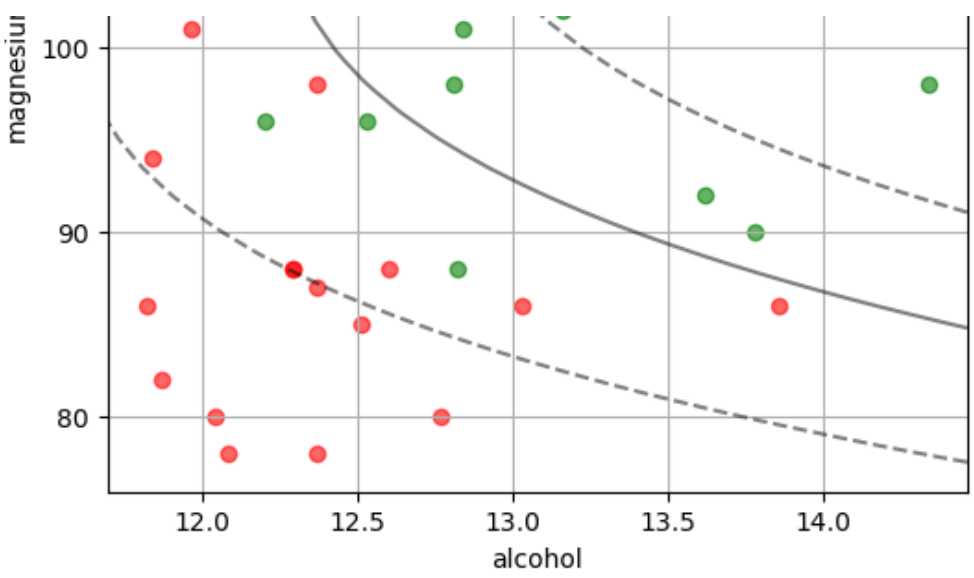



```
C:\Users\yumif\anaconda3\envs\mlcourse\lib\site-packages\sklearn\base.py:439: UserWarning
: X does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```




```
C:\Users\yumif\anaconda3\envs\mlcourse\lib\site-packages\sklearn\base.py:439: UserWarning
: X does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```





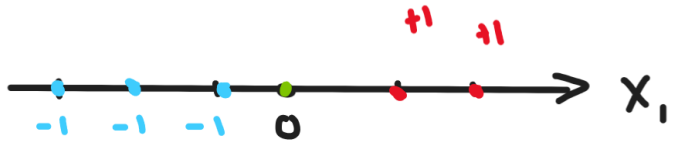
In [70]:

Question 3:

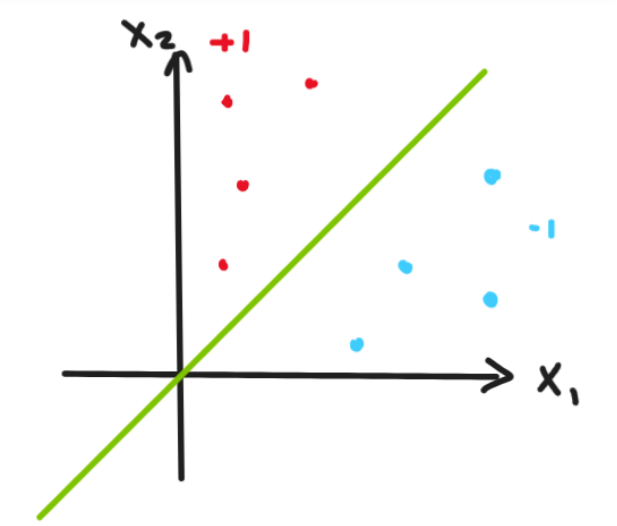
1.

The classifier will achieve error of zero if the data is linearly separable through the origin (since $w \in \mathbb{R}^d$), that is there is hyper plane going through the origin, separating points with label +1 on one side and -1 on the other side.

d = 1



d = 2



2.

Let $w = (w_1, w_2)$

We need to solve the following optimization problem (the quadratic form of hard-SVM):

$$w_0 = \arg\min_w \|w\|^2 \ \ s.t. \forall i \ \ y_i \cdot \langle w, x_i \rangle \geq 1$$

In our instance,

$$w_0 = \arg\min_w w_1^2 + w_2^2 \ \ s.t. -1 \cdot pw_1 \geq 1 \ \wedge 1 \cdot qw_2 \geq 1$$

Let us divide the possible cases of p, q:

Case $p = 0 \ \vee \ q = 0$:

Either $-1 \cdot pw_1 \geq 1 \Rightarrow 0 \geq 1$ or $1 \cdot qw_2 \geq 1 \Rightarrow 0 \geq 1$

And there is no solution to problem.

Case $p \neq 0 \ \wedge \ q \neq 0$:

$$pw_1 \leq -1 \Rightarrow w_1 \leq -\frac{1}{p}$$

$$qw_2 \geq 1 \Rightarrow w_2 \geq \frac{1}{q}$$

$$w_0 = \arg\min_w w_1^2 + w_2^2 \ \ s.t. \ \ w_1 \leq -\frac{1}{p} \ \wedge \ w_2 \geq \frac{1}{q}$$

$$w_1^2 + w_2^2 \geq \left(-\frac{1}{p}\right)^2 + \left(\frac{1}{q}\right)^2$$

And

$$w_1 = -\frac{1}{p}, w_2 = \frac{1}{q}$$

Gives the following lower bound. Thus, minimum exists, and there is a solution to hard-SVM.

3.

We will get the same result. Data is still linearly separable, and maximizing the margin is done only via the support vectors, vectors which are the closest to the separating hyper plane. None of the points which disappeared are support vectors, so then our solution to hard-SVM will remain the same.

4.

Figure 1 has $\lambda = 200$, Figure 2 has $\lambda = 2$.

For bigger regularization parameter, we put more weight on the $\|w\|^2$, which leads to bigger margin but also bigger train error. For smaller values of λ, we expect smaller margin, but better results on train set. As we can see on figure 1, the support vectors are much further away from the separating hyper plane than in figure 2, that is bigger margin. For $\lambda = 2$, we put much more importance on the mean of hinge losses, which gives us smaller train error. And as we can see on figure 2, it is exactly the case, we separated all the points perfectly, giving smaller train error unlike on figure 2.

תהי $\psi: X \to F$ פונקציה הממפה דוגמאות נתונות למרחב פיצ'רים כלשהוא F (כאשר נדרוש ש- F הוא תת קבוצה של מרחב הילברט). אזי הפונקציה $K: X \times X \to R$ המוגדרת על ידי

$$K(x, x') = \langle \psi(x), \psi(x') \rangle$$

היא פונקציית kernel.

בהינתן מדגם אימון $S = \{(x_i, y_i)\}_{i=1}^m$, כאשר $y_i \in \{-1, 1\}$, $x_i \in R^d =: X$, נמפה את הדוגמאות למדגם אימון ב- F,
$S^\psi = \{(\psi(x_i), y_i)\}_{i=1}^m$.

1. כתבו פסאודו-קוד של אלגוריתם הפרספטרון על מדגם האימון S^ψ. שימו לב ש- ψ לא ידועה ורק K נתון. בהדרכה לסעיף זה, ענו גם על השאלות הבאות:
   - *a.* מהו הפלט של האלגוריתם? (לא ניתן להחזיר מישור מפריד ב- F מכיוון ש- ψ אינה ידועה).
   - *b.* בהינתן הפלט של האלגוריתם, הביעו את כלל ההחלטה, $h: X \to \{-1,1\}$, באמצעות K (ללא ψ).
2. הוכיחו שהאלגוריתם שכתבתם מתכנס אם ורק אם S^ψ פריד לינארית ב- F.

Perceptron$(\{(x_i, y_i)\}_{i=1}^m)$:

$\alpha^{(0)} = (0, \dots, 0)$ // m length

for $t = 1, 2, \dots, T$ do

  if $\exists i \in [m]: y_i\left(\sum_{j=1}^{m} \alpha_j k(x_j, x_i)\right) \le 0$

  (*) $\alpha^{(t+1)}(i) = \alpha^{(t)}(i) + y_i$

  $\alpha^{(t+1)}(\forall j \ne i) = \alpha^{(t)}(j)$

  else:

   break

return $\alpha^{(t)}$

.1 אנחנו יודעים כי $\hat{w}$ הוא צ"ל של $\forall i: \psi(x_i)$

$\langle w, \psi(x_i) \rangle = \langle \sum_{j=1}^{m} \alpha_j \psi(x_j), \psi(x_i) \rangle$

$= \sum_{i=1}^{m} \alpha_i k(x_i, x_j)$

.a הפלט זה וקטור α שהם מקדמים לינאריים של המישור w עם $\forall i \in [m]: \psi(x_i)$

.b $h: X \to \{-1, 1\}$

$h(x_j) = \text{sign}\left(\sum_{i=1}^{m} \alpha_i k(x_j, x_i)\right)$

כאשר α המקדמים זה (*)

.2 $\Leftarrow$: האלגוריתם נעצר כלומר פריד לינארית כאשר

$\forall i \in m: y_i\left(\sum_{i=1}^{m} \alpha_i k(x_i, x_j)\right) > 0$

$\Rightarrow \text{sign}\left(\sum_{i=1}^{m} \alpha_i k(x_i, x_j)\right) = y_i \Rightarrow \text{sign}(\langle w, \psi(x_i) \rangle) = y_i$

כאשר $w = \sum_{i=1}^{m} \alpha_i \cdot \psi(x_i)$

$\Rightarrow$: פריד לינארית $\Leftarrow$ הפרספטרון. ניתן לראות כי בכל

$y_i \langle w^{(t)}, x_i \rangle \le 0$

והראינו כי מתקיים $y_i \cdot \langle w, \psi(x_i) \rangle = y_i \left(\sum_{j=1}^{m} \alpha_j k(x_i, x_j) \right)$

כאשר $w = \sum_{j=1}^{m} \alpha_j \cdot \psi(x_j)$

ובפרט $w^{(t+1)} = w^{(t)} + y_i \cdot \psi(x_i)$

$$\Updownarrow$$

$$\alpha_i^{(t+1)} = \alpha_i^{(t)} + y_i$$

ולכן עבור $i \neq j$ נקבל $\alpha_j^{(t+1)} = \alpha_j^{(t)}$

ובפרט האלגוריתם שיש לנו שקול להרצה של ה perceptron רגיל על S^ψ

$\Leftarrow$ perceptron מוצא אמ"מ S^ψ פריד לינארית ב- F
אזי האלגוריתם שלנו מוצא אמ"מ S^ψ פריד לינארית ב- F ∎