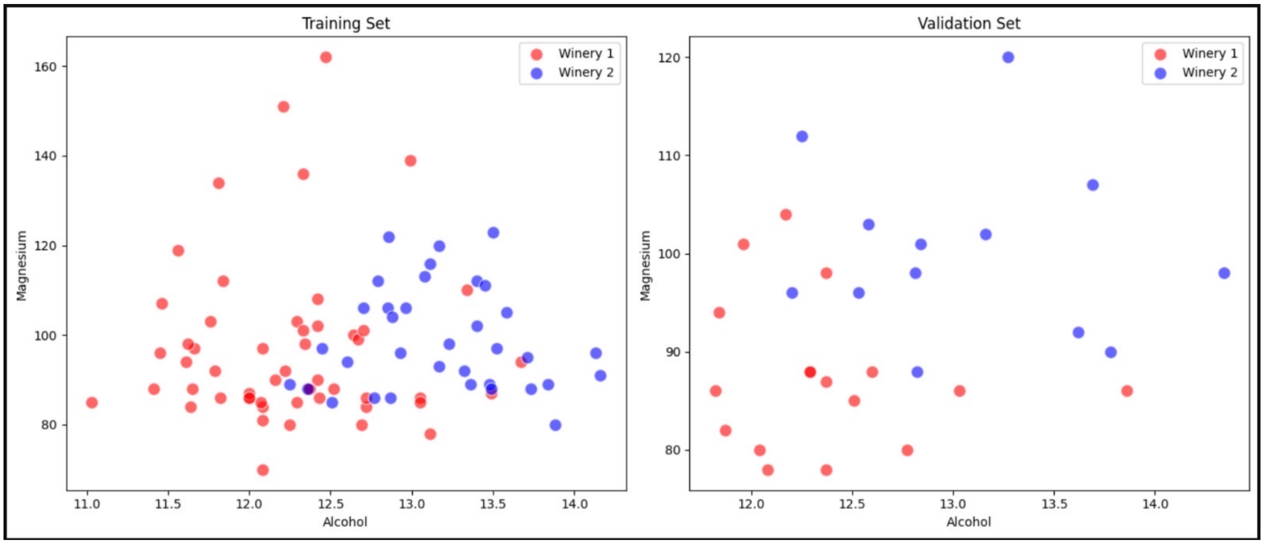
שאלה 1

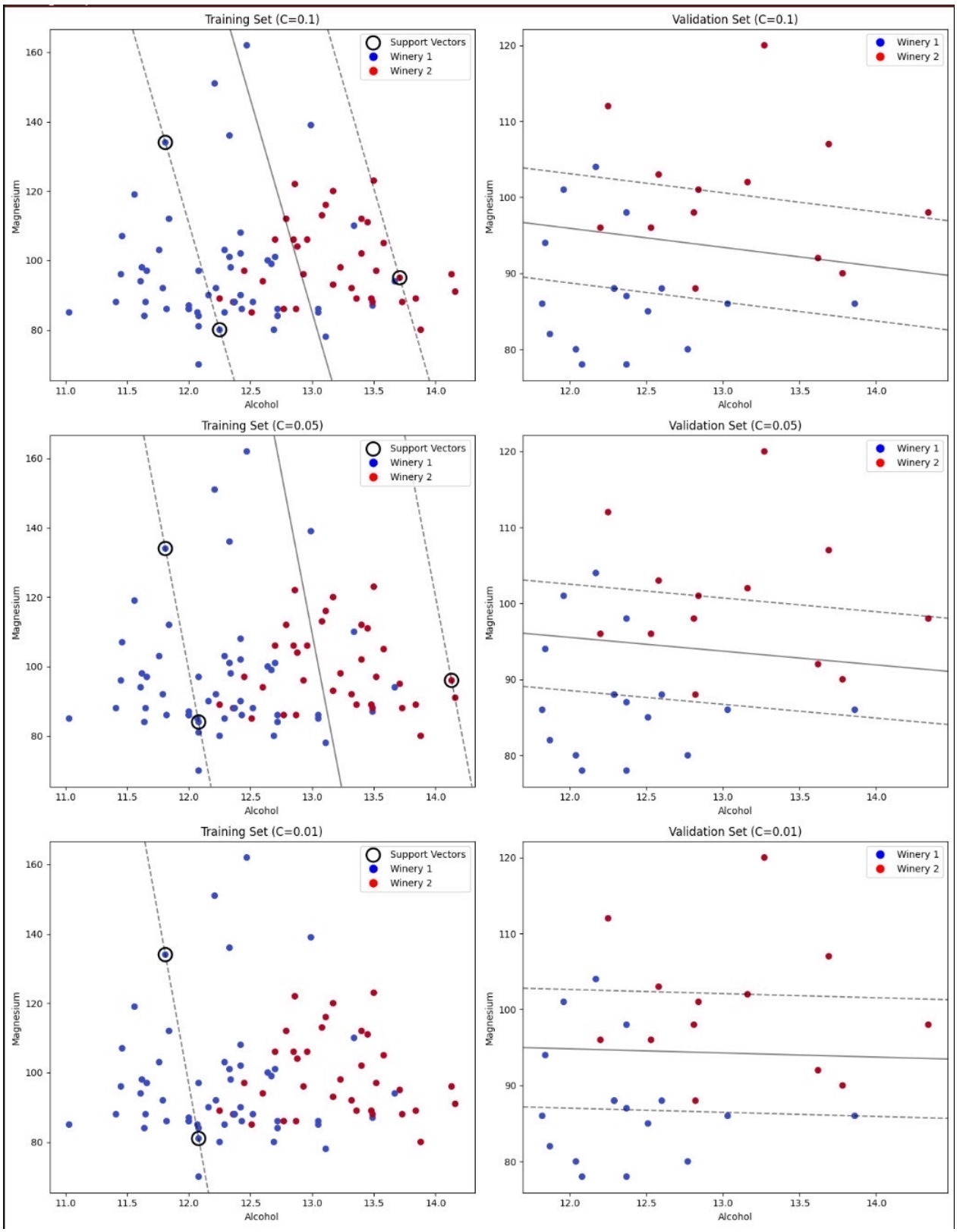
הקוד מצורף בסוף הקובץ.

סעיף 1 :



Hard-SVM יחפש על-מישור ובמקרה הנ"ל ישר המפריד לינארית את הנתונים ל-2 קבוצות: יקב 1, יקב 2 כך שכל קבוצה תהיה מצד אחר של הישר.
ניתן לראות כי הנתונים אינם פרידים לינארית ולכן Hard-SVM לא יניב תוצאה עבור נתונים אלו בצורתם הנוכחית (אין פתרון ל-Hard-SVM ולכן לא יתכנס).

שאלה 2:


Training Set (C=0.1)
Validation Set (C=0.1)
Training Set (C=0.05)
Validation Set (C=0.05)
Training Set (C=0.01)
Validation Set (C=0.01)
Support Vectors
Winery 1
Winery 2
Magnesium
Alcohol

סעיף ג:

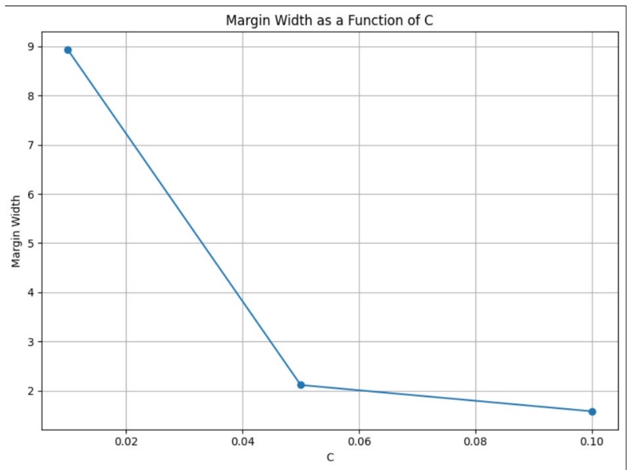
בבעיית האופטימיזציה הריבועית מקבעים את ה- margin להיות 1, כלומר $\min_i |\langle w_0, x_i \rangle| = 1$ נקבל:

$$\min_i |\langle w_0,x_i\rangle| = 1 \overset{\frac{1}{\|w_0\|}}{\Longrightarrow} \frac{1}{\|w_0\|}\min_i |\langle w_0,x_i\rangle| = \frac{1}{\|w_0\|} \Longrightarrow$$

$$\Longrightarrow \min_i \left|\left\langle \frac{w_0}{\|w_0\|}, x_i \right\rangle\right| = \frac{1}{\|w_0\|} \overset{\text{הגדרה}}{\Longrightarrow} \min_i |\langle \hat{w}, x_i\rangle| = \frac{1}{\|w_0\|}$$

$$\overset{\text{הגדרה}}{\Longrightarrow} \boxed{\text{margin} = \frac{1}{\|w_0\|}} \quad \blacksquare$$

סעיף ד:



ניתן לראות כי ככול ש- C גדל, ה- margin קטן. משום ש- C הינו גורם רגולריזציה הגורם ל- trade-off:

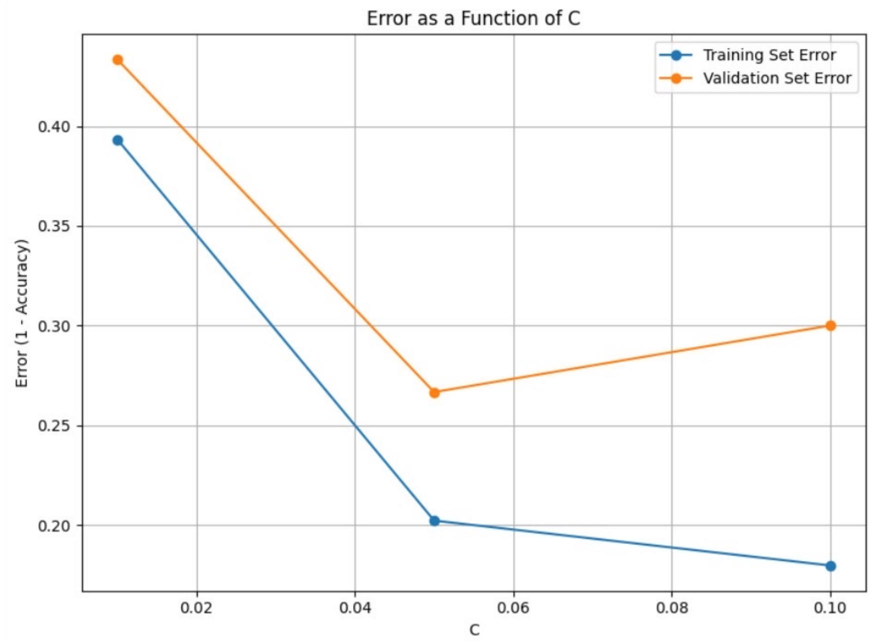
בבעיית ההגדרה של בעיית ה- Soft-SVM ישנם 2 רכיבים:

$$\min_w \|w\|^2 + C \cdot \frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y_i \langle w, x_i\rangle\}$$

לכן, C קטן משמעותו כי מתרכזים במקסימום ה margin ולכן בעדיפות משנית מפריע עם שוליים רחבים יותר, גם במחיר של הגדלת שגיאות האימון.

ו- C גדל משמעותו כי מתרכזים במזעור שגיאת סיווג ולקבל הפרדה לנקודות חזקה יותר, ולכן בעדיפות למזער את שגיאות האימון גם במחיר של הקטנת ה- margin.

סעיף 5:

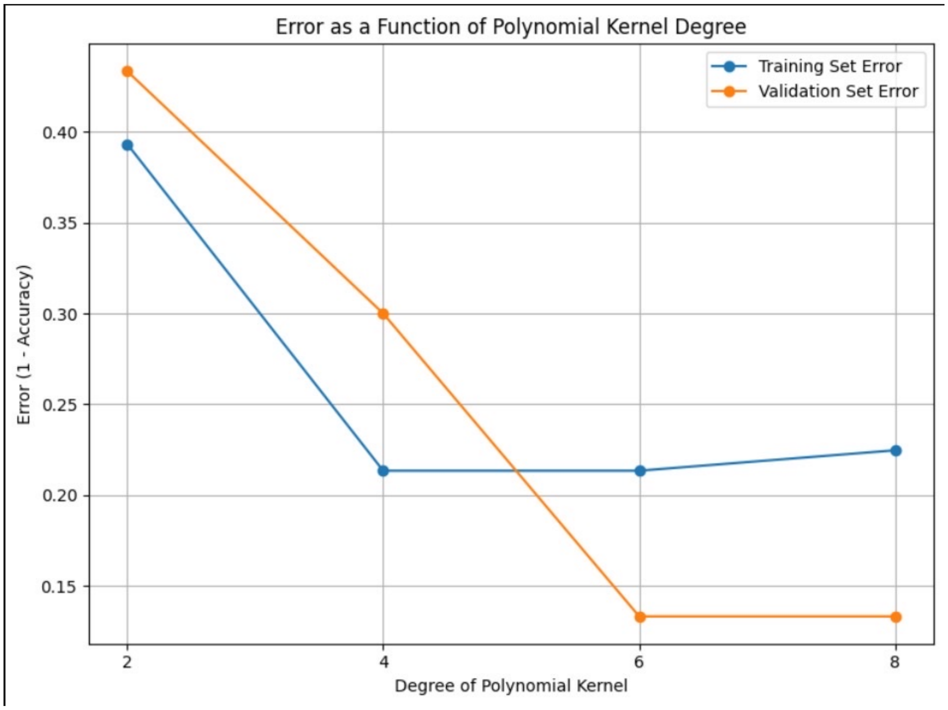


ניתן לראות כי עד C=0.05 שגיאות האימון והוולידציה יורדות יחד, החל מנקודה זו מתקיים כי ככל ש- C גדל, שגיאות האימון קטנה ושגיאות הולידציה גדלה.

הדבר קורה מכיוון שהחל מהנקודה הנ"ל C מתפקד כגורם רגולציה המממש את ה trade-off בין מיקסום ה- margin (שגיאת ולידציה קטנה) לבין מזעור שגיאת בסיווג (שגיאת אימון קטנה) כפי שהוסבר בסעיף הקודם.

לכן עבור אלגוריתם אופטימלי, נבחר ב- C=0.05 ונקבל את האיזון האופטימלי ב- trade-off שבפונקציית המטרה בבעית ה- soft-SVM.

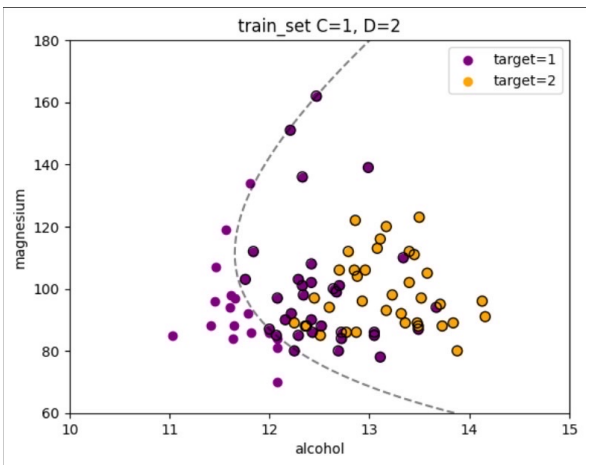
סעיף 6:

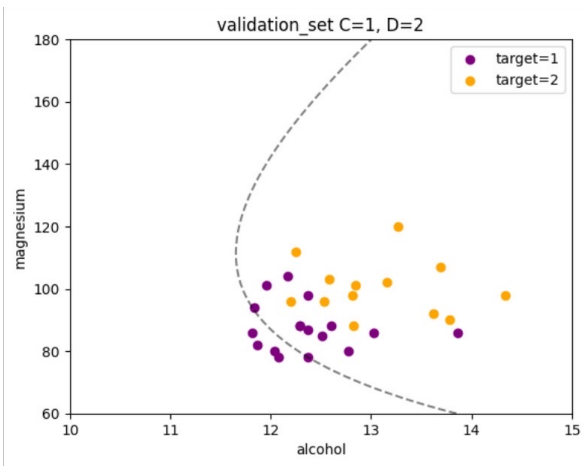


נשים לב כי ייצוג נתונים במימד גבוה יותר עזר בשיפור הדיוק על סט הולידציה.
לדעתי, דבר זה קורה מכיוון שהדאטה לא פריד לינארית, ובכל הגבהה כל שנגדיל את המימד - נקבל מישור החלטה הקרוב יותר להפרדה לינארית.
בגרף שקיבלנו ניתן לראות כי החל ממימד 6 - מישור ההחלטה מסווג בצורה הפולינומית המיטבית הנתונים כך שהשגיאה על סט הולידציה קטנה מאוד.
אם נגיע למימד המפריד את הנתונים בצורה מושלמת - כל מימד הגבוה ממנו הנתונים יהיו פרידים גם כן.
אחרת - ייתכן והמיפוי הפולינומי אינו מביא את הנתונים להפרדה לינארית מושלמת ויש לחפש מיפוי מתאים יותר.

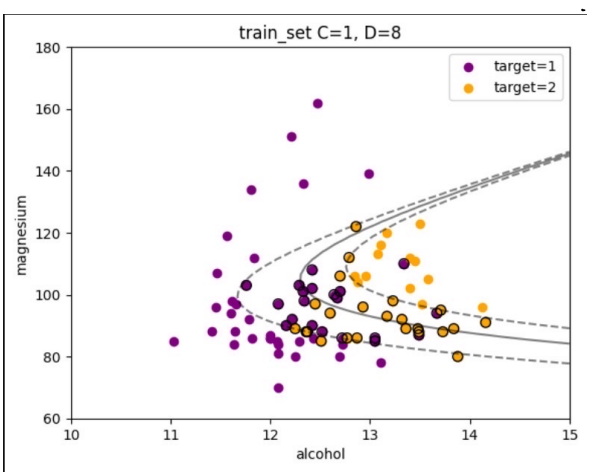
סעיף א':

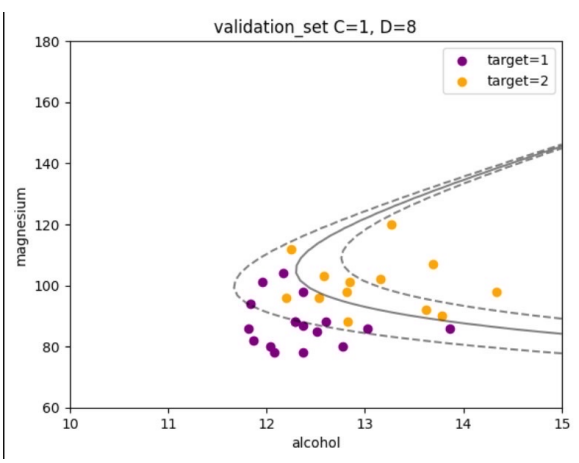
נקבע כי עומק העץ $d=2$ השגיאה הינה הגבוהה ביותר:





נקבע כי עומק העץ $d=5,6,7,8$ השגיאה היא הנמוכה ביותר, השגיאה מגיע $d=8$





جواب

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

# Check the sizes of the splits
print(f'Training set size: {train_df.shape[0]}')
print(f'Validation set size: {val_df.shape[0]}')
```

```
Training set size: 89
Validation set size: 30
```

سؤال ۴ :

```
# Map target values to colors
color_map = {1: 'red', 2: 'blue'}
train_colors = train_df['target'].map(color_map)
val_colors = val_df['target'].map(color_map)

# Create scatter plots
plt.figure(figsize=(14, 6))

# Scatter plot for training set
plt.subplot(1, 2, 1)
for target, color in color_map.items():
    subset = train_df[train_df['target'] == target]
    plt.scatter(subset['alcohol'], subset['magnesium'], c=color, alpha=0.6, edgecolors='w', s=100, label=f'Winery {target}')
plt.title('Training Set')
plt.xlabel('Alcohol')
plt.ylabel('Magnesium')
plt.legend()

# Scatter plot for validation set
plt.subplot(1, 2, 2)
for target, color in color_map.items():
    subset = val_df[val_df['target'] == target]
    plt.scatter(subset['alcohol'], subset['magnesium'], c=color, alpha=0.6, edgecolors='w', s=100, label=f'Winery {target}')
plt.title('Validation Set')
plt.xlabel('Alcohol')
plt.ylabel('Magnesium')
plt.legend()

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

: 2 ১.২০

```python
# Prepare data for SVM
X_train = train_df[['alcohol', 'magnesium']]
y_train = train_df['target']
X_val = val_df[['alcohol', 'magnesium']]
y_val = val_df['target']

# List of C values
C_values = [0.1, 0.05, 0.01]

# Function to plot decision boundary and support vectors
def plot_svm(C_value, X_train, y_train, X_val, y_val, ax, title, plot_support_vectors):
    model = SVC(C=C_value, kernel='linear')
    model.fit(X_train, y_train)

    ax.set_title(title)
    ax.scatter(X_train['alcohol'], X_train['magnesium'], c=y_train, cmap='coolwarm')

    if plot_support_vectors:
        # Get support vectors
        support_indices = model.support_
        support_vectors = model.support_vectors_
        dual_coefs = np.abs(model.dual_coef_.ravel())

        # Plot only support vectors on the margin
        plotted_sv = False
        for i, index in enumerate(support_indices):
            if 0 < dual_coefs[i] < C_value:
                if not plotted_sv:
                    ax.scatter(support_vectors[i, 0], support_vectors[i, 1], facecolors='none', edgecolors='k', s=200, linewidths=2, label='Support Vectors')
                    plotted_sv = True
                else:
                    ax.scatter(support_vectors[i, 0], support_vectors[i, 1], facecolors='none', edgecolors='k', s=200, linewidths=2)

    # Plot decision boundary and margins
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()
    xx, yy = np.meshgrid(np.linspace(xlim[0], xlim[1], 500), np.linspace(ylim[0], ylim[1], 500))
    Z = model.decision_function(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    ax.contour(xx, yy, Z, colors='k', levels=[-1, 0, 1], alpha=0.5, linestyles=['--', '-', '--'])
    ax.set_xlabel('Alcohol')
    ax.set_ylabel('Magnesium')

    # Adding legend with labels for Winery 1 and Winery 2
    handles, labels = ax.get_legend_handles_labels()
    ax.legend(handles + [plt.Line2D([0], [0], marker='o', color='w', markerfacecolor='b', markersize=10),
                         plt.Line2D([0], [0], marker='o', color='w', markerfacecolor='r', markersize=10)],
              labels + ['Winery 1', 'Winery 2'], loc='upper right')

# Plot SVM for each C value
fig, axes = plt.subplots(len(C_values), 2, figsize=(14, len(C_values) * 6))

for i, C in enumerate(C_values):
    # Plot for training set
    plot_svm(C, X_train, y_train, X_val, y_val, axes[i, 0], f'Training Set (C={C})', plot_support_vectors=True)

    # Plot for validation set
    plot_svm(C, X_val, y_val, X_train, y_train, axes[i, 1], f'Validation Set (C={C})', plot_support_vectors=False)

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

```python
# Function to calculate margin width
def calculate_margin_width(C_value, X_train, y_train):
    model = SVC(C=C_value, kernel='linear')
    model.fit(X_train, y_train)

    # Get support vectors
    support_vectors = model.support_vectors_
    dual_coefs = np.abs(model.dual_coef_.ravel())

    # Calculate margin width
    margin = 2 / np.linalg.norm(model.coef_)

    return margin

# Calculate margin width for each C value
margin_widths = []
for C in C_values:
    margin_width = calculate_margin_width(C, X_train, y_train)
    margin_widths.append(margin_width)

# Plotting the margin width as a function of C
plt.figure(figsize=(8, 6))
plt.plot(C_values, margin_widths, marker='o')
plt.xlabel('C')
plt.ylabel('Margin Width')
plt.title('Margin Width as a Function of C')
plt.grid(True)
plt.tight_layout()
plt.show()
```

سؤال ٥:

```python
# Calculate accuracy for each C value
train_accuracies = []
val_accuracies = []

for C in C_values:
    # Train SVM model
    model = SVC(C=C, kernel='linear')
    model.fit(X_train, y_train)

    # Predictions
    train_pred = model.predict(X_train)
    val_pred = model.predict(X_val)

    # Calculate accuracy
    train_accuracy = np.mean(train_pred == y_train)
    val_accuracy = np.mean(val_pred == y_val)

    # Calculate error
    train_error = 1 - train_accuracy
    val_error = 1 - val_accuracy

    # Append to lists
    train_accuracies.append(train_error)
    val_accuracies.append(val_error)

# Plotting the errors as a function of C
plt.figure(figsize=(8, 6))
plt.plot(C_values, train_accuracies, marker='o', label='Training Set Error')
plt.plot(C_values, val_accuracies, marker='o', label='Validation Set Error')
plt.xlabel('C')
plt.ylabel('Error (1 - Accuracy)')
plt.title('Error as a Function of C')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

: 6 ข้อ

```python
# Degrees to test
degrees = [2, 4, 6, 8]
C = 1

# Lists to store errors
train_errors = []
val_errors = []

# Function to calculate error (1 - accuracy)
def calculate_error(y_true, y_pred):
    accuracy = np.mean(y_true == y_pred)
    return 1 - accuracy

# Loop through each degree
for degree in degrees:
    # Train SVM model with polynomial kernel
    model = SVC(C=C, kernel='poly', degree=degree)
    model.fit(X_train, y_train)

    # Predictions
    train_pred = model.predict(X_train)
    val_pred = model.predict(X_val)

    # Calculate errors
    train_error = calculate_error(y_train, train_pred)
    val_error = calculate_error(y_val, val_pred)

    # Append to lists
    train_errors.append(train_error)
    val_errors.append(val_error)

# Plotting errors as a function of degree
plt.figure(figsize=(8, 6))
plt.plot(degrees, train_errors, marker='o', label='Training Set Error')
plt.plot(degrees, val_errors, marker='o', label='Validation Set Error')
plt.xlabel('Degree of Polynomial Kernel')
plt.ylabel('Error (1 - Accuracy)')
plt.title('Error as a Function of Polynomial Kernel Degree')
plt.xticks(degrees)
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

סעיף ד:

פונקציות עזר

```python
def plot_svc_decision_function_train(model, ax=None, plot_support=True):
    """Plot the decision function boundary for a SVC model."""
    if ax is None:
        ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # Create grid to evaluate model
    x = np.linspace(10, 15)
    y = np.linspace(60, 180)
    Y, X = np.meshgrid(y, x)
    xy = np.vstack([X.ravel(), Y.ravel()]).T
    df_xy = pd.DataFrame(xy, columns=['alcohol', 'magnesium'])
    P = model.decision_function(xy).reshape(X.shape)

    # Plot the decision contour
    ax.contour(X, 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=50, linewidth=1, facecolors='none', edgecolors='black')
    ax.set_xlim(xlim)
    ax.set_ylim(ylim)


def plot_svc_decision_function_val(model, 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(10, 15)
    y = np.linspace(60, 180)
    Y, X = np.meshgrid(y, x)
    xy = np.vstack([X.ravel(), Y.ravel()]).T
    df_xy = pd.DataFrame(xy, columns=['alcohol', 'magnesium'])
    P = model.decision_function(xy).reshape(X.shape)

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

```python
# Define the data
train_df, val_df = train_test_split(df, test_size=30, random_state=3)
train_df_1 = train_df[['alcohol', 'magnesium']][train_df.target == 1]
train_df_2 = train_df[['alcohol', 'magnesium']][train_df.target == 2]
val_df_1 = val_df[['alcohol', 'magnesium']][val_df.target == 1]
val_df_2 = val_df[['alcohol', 'magnesium']][val_df.target == 2]

# Define the model
model = SVC(kernel='poly', C=1, degree=2)

# Train the model
model.fit(train_df[['alcohol', 'magnesium']], train_df['target'])

# Create the scatter plot for the training data
plt.scatter(train_df_1['alcohol'], train_df_1['magnesium'], c='purple', label='target=1')
plt.scatter(train_df_2['alcohol'], train_df_2['magnesium'], c='orange', label='target=2')

# Plot the decision function boundary for the training data
plot_svc_decision_function_train(model)

plt.xlabel("alcohol")
plt.ylabel("magnesium")
plt.title("train_set C=1, D=2")

plt.legend(loc='upper right')
plt.xlim([10, 15])
plt.ylim([60, 180])
plt.show()

# Create the scatter plot for the validation data
plt.scatter(val_df_1['alcohol'], val_df_1['magnesium'], c='purple', label='target=1')
plt.scatter(val_df_2['alcohol'], val_df_2['magnesium'], c='orange', label='target=2')

# Plot the decision function boundary for the validation data
plot_svc_decision_function_val(model)

plt.xlabel("alcohol")
plt.ylabel("magnesium")
plt.title("validation_set C=1, D=2")

plt.legend(loc='upper right')
plt.xlim([10, 15])
plt.ylim([60, 180])
plt.show()
```

סעיף ה:

```python
# Define the model
model = SVC(kernel='poly', C=1, degree=8)

# Train the model
model.fit(train_df[['alcohol', 'magnesium']], train_df['target'])

# Create the scatter plot for the training data
plt.scatter(train_df_1['alcohol'], train_df_1['magnesium'], c='purple', label='target=1')
plt.scatter(train_df_2['alcohol'], train_df_2['magnesium'], c='orange', label='target=2')

# Plot the decision function boundary for the training data
plot_svc_decision_function_train(model)

plt.xlabel("alcohol")
plt.ylabel("magnesium")
plt.title("train_set C=1, D=8")

plt.legend(loc='upper right')
plt.xlim([10, 15])
plt.ylim([60, 180])
plt.show()

# Create the scatter plot for the validation data
plt.scatter(val_df_1['alcohol'], val_df_1['magnesium'], c='purple', label='target=1')
plt.scatter(val_df_2['alcohol'], val_df_2['magnesium'], c='orange', label='target=2')

# Plot the decision function boundary for the validation data
plot_svc_decision_function_val(model)

plt.xlabel("alcohol")
plt.ylabel("magnesium")
plt.title("validation_set C=1, D=8")

plt.legend(loc='upper right')
plt.xlim([10, 15])
plt.ylim([60, 180])
plt.show()
```

3) א. שגיאת סיווג 0 משמעותה כי

$$\sum_{i=1}^{m} I\{sign(\langle w, x_i \rangle + b) \neq y_i\} = 0$$

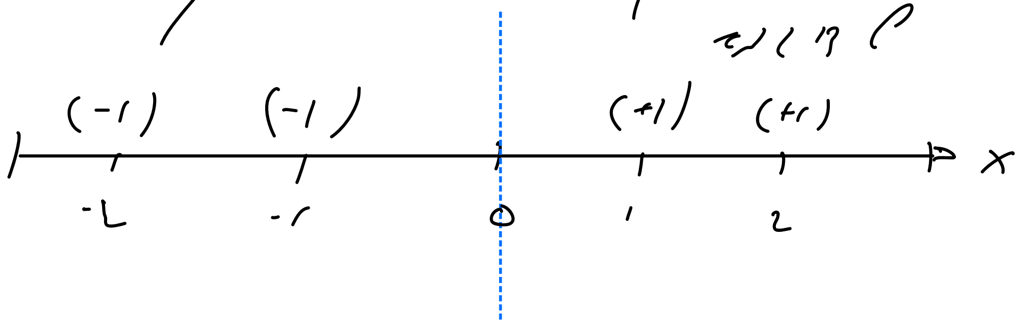
כלומר, שסיווגי S הינם נכון

* עבור d=1

לא נוכל לקבל שגיאת סיווג כאשר קיים

נק' a על הישר עבור $\bar{a} \geq a$

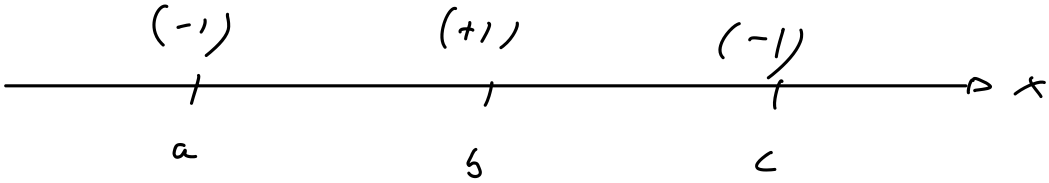
[illegible] רק נק' מסוים [illegible]



נקבל שגיאת סיווג אם עבור נק' קונקרטי

b קיימת 2 נק' a<b<c כך

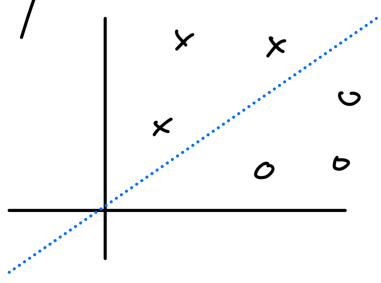
$y_a = y_c \neq y_b$

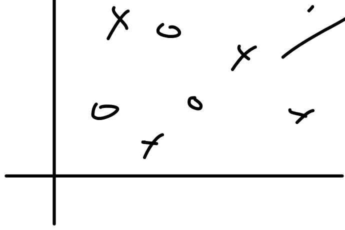


* עבור d=2

3 נקודות על [illegible] עבור [illegible]

2 סוגים הדוגמאות:





ב. נשים לב כי $\forall n \in \mathbb{N}$ מתקיים $n \in \mathbb{Z}$, לכן

$$\mathbb{N} \subseteq \mathbb{Z}$$

מכיוון ש $0 \in \mathbb{Z}$ אך $0 \notin \mathbb{N}$ נובע כי

$$\mathbb{N} \subsetneq \mathbb{Z}$$

ג. נשים לב כי $\forall n \in \mathbb{Z}$ מתקיים $n \in \mathbb{Q}$, לכן

$$\mathbb{Z} \subseteq \mathbb{Q}$$

מכיוון ש $\frac{1}{2} \in \mathbb{Q}$ אך $\frac{1}{2} \notin \mathbb{Z}$ נובע כי

$$\mathbb{Z} \subsetneq \mathbb{Q}$$

ד. נשים לב כי $\forall n \in \mathbb{N}$ מתקיים $n \in \mathbb{R}$, לכן

$$\mathbb{N} \subseteq \mathbb{R}$$

מכיוון ש $\sqrt{2} \in \mathbb{R}$ אך $\sqrt{2} \notin \mathbb{N}$ נובע כי

$$\mathbb{N} \subsetneq \mathbb{R}$$

ה. נשים לב כי $\forall n \in \mathbb{Z}$ מתקיים $n \in \mathbb{R}$, לכן

$$\mathbb{Z} \subseteq \mathbb{R}$$

מכיוון ש $\sqrt{2} \in \mathbb{R}$ אך $\sqrt{2} \notin \mathbb{Z}$ נובע כי

$$\mathbb{Z} \subsetneq \mathbb{R}$$

ו. נשים לב כי $\forall q \in \mathbb{Q}$ מתקיים $q \in \mathbb{R}$, לכן

$$\mathbb{Q} \subseteq \mathbb{R}$$

מכיוון ש $\sqrt{2} \in \mathbb{R}$ אך $\sqrt{2} \notin \mathbb{Q}$ נובע כי

$$\mathbb{Q} \subsetneq \mathbb{R}$$

ז. נשים לב כי $\forall n \in \mathbb{N}$ מתקיים $n \in \mathbb{Q}$, לכן

$$\mathbb{N} \subseteq \mathbb{Q}$$

מכיוון ש $\frac{1}{2} \in \mathbb{Q}$ אך $\frac{1}{2} \notin \mathbb{N}$ נובע כי

$$\mathbb{N} \subsetneq \mathbb{Q}$$

ח. נשים לב כי $\forall n \in \mathbb{N}$ מתקיים $n \in \mathbb{Z}$, לכן

$$\mathbb{N} \subseteq \mathbb{Z}$$

מכיוון ש $-1 \in \mathbb{Z}$ אך $-1 \notin \mathbb{N}$ נובע כי

$$\mathbb{N} \subsetneq \mathbb{Z}$$

ט. נשים לב כי $\forall n \in \mathbb{Z}$ מתקיים $n \in \mathbb{Q}$, לכן

$$\mathbb{Z} \subseteq \mathbb{Q}$$

מכיוון ש $\frac{1}{2} \in \mathbb{Q}$ אך $\frac{1}{2} \notin \mathbb{Z}$ נובע כי

$$\mathbb{Z} \subsetneq \mathbb{Q}$$

י. נשים לב כי $\forall n \in \mathbb{N}$ מתקיים $n \in \mathbb{R}$, לכן

$$\mathbb{N} \subseteq \mathbb{R}$$

מכיוון ש $\sqrt{2} \in \mathbb{R}$ אך $\sqrt{2} \notin \mathbb{N}$ נובע כי

$$\mathbb{N} \subsetneq \mathbb{R}$$

יא. נשים לב כי $\forall n \in \mathbb{Z}$ מתקיים $n \in \mathbb{R}$, לכן

$$\mathbb{Z} \subseteq \mathbb{R}$$

מכיוון ש $\sqrt{2} \in \mathbb{R}$ אך $\sqrt{2} \notin \mathbb{Z}$ נובע כי

$$\mathbb{Z} \subsetneq \mathbb{R}$$

יב. נשים לב כי $\forall q \in \mathbb{Q}$ מתקיים $q \in \mathbb{R}$, לכן

$$\mathbb{Q} \subseteq \mathbb{R}$$

מכיוון ש $\sqrt{2} \in \mathbb{R}$ אך $\sqrt{2} \notin \mathbb{Q}$ נובע כי

$$\mathbb{Q} \subsetneq \mathbb{R}$$

יג. נשים לב כי $\forall n \in \mathbb{N}$ מתקיים $n \in \mathbb{Q}$, לכן

$$\mathbb{N} \subseteq \mathbb{Q}$$

מכיוון ש $\frac{1}{2} \in \mathbb{Q}$ אך $\frac{1}{2} \notin \mathbb{N}$ נובע כי

$$\mathbb{N} \subsetneq \mathbb{Q}$$

יד. נשים לב כי $\forall n \in \mathbb{N}$ מתקיים $n \in \mathbb{Z}$, לכן

$$\mathbb{N} \subseteq \mathbb{Z}$$

מכיוון ש $-1 \in \mathbb{Z}$ אך $-1 \notin \mathbb{N}$ נובע כי

$$\mathbb{N} \subsetneq \mathbb{Z}$$

טו. נשים לב כי $\forall n \in \mathbb{Z}$ מתקיים $n \in \mathbb{Q}$, לכן

$$\mathbb{Z} \subseteq \mathbb{Q}$$

מכיוון ש $\frac{1}{2} \in \mathbb{Q}$ אך $\frac{1}{2} \notin \mathbb{Z}$ נובע כי

$$\mathbb{Z} \subsetneq \mathbb{Q}$$

טז. נשים לב כי $\forall n \in \mathbb{N}$ מתקיים $n \in \mathbb{R}$, לכן

$$\mathbb{N} \subseteq \mathbb{R}$$

מכיוון ש $\sqrt{2} \in \mathbb{R}$ אך $\sqrt{2} \notin \mathbb{N}$ נובע כי

$$\mathbb{N} \subsetneq \mathbb{R}$$

יז. נשים לב כי $\forall n \in \mathbb{Z}$ מתקיים $n \in \mathbb{R}$, לכן

$$\mathbb{Z} \subseteq \mathbb{R}$$

מכיוון ש $\sqrt{2} \in \mathbb{R}$ אך $\sqrt{2} \notin \mathbb{Z}$ נובע כי

$$\mathbb{Z} \subsetneq \mathbb{R}$$

יח. נשים לב כי $\forall q \in \mathbb{Q}$ מתקיים $q \in \mathbb{R}$, לכן

$$\mathbb{Q} \subseteq \mathbb{R}$$

מכיוון ש $\sqrt{2} \in \mathbb{R}$ אך $\sqrt{2} \notin \mathbb{Q}$ נובע כי

$$\mathbb{Q} \subsetneq \mathbb{R}$$

יט. נשים לב כי $\forall n \in \mathbb{N}$ מתקיים $n \in \mathbb{Q}$, לכן

$$\mathbb{N} \subseteq \mathbb{Q}$$

מכיוון ש $\frac{1}{2} \in \mathbb{Q}$ אך $\frac{1}{2} \notin \mathbb{N}$ נובע כי

$$\mathbb{N} \subsetneq \mathbb{Q}$$

כ. נשים לב כי $\forall n \in \mathbb{N}$ מתקיים $n \in \mathbb{Z}$, לכן

$$\mathbb{N} \subseteq \mathbb{Z}$$

מכיוון ש $-1 \in \mathbb{Z}$ אך $-1 \notin \mathbb{N}$ נובע כי

$$\mathbb{N} \subsetneq \mathbb{Z}$$

כא. נשים לב כי $\forall n \in \mathbb{Z}$ מתקיים $n \in \mathbb{Q}$, לכן

$$\mathbb{Z} \subseteq \mathbb{Q}$$

מכיוון ש $\frac{1}{2} \in \mathbb{Q}$ אך $\frac{1}{2} \notin \mathbb{Z}$ נובע כי

$$\mathbb{Z} \subsetneq \mathbb{Q}$$

כב. נשים לב כי $\forall n \in \mathbb{N}$ מתקיים $n \in \mathbb{R}$, לכן

$$\mathbb{N} \subseteq \mathbb{R}$$

מכיוון ש $\sqrt{2} \in \mathbb{R}$ אך $\sqrt{2} \notin \mathbb{N}$ נובע כי

$$\mathbb{N} \subsetneq \mathbb{R}$$

כג. נשים לב כי $\forall n \in \mathbb{Z}$ מתקיים $n \in \mathbb{R}$, לכן

$$\mathbb{Z} \subseteq \mathbb{R}$$

מכיוון ש $\sqrt{2} \in \mathbb{R}$ אך $\sqrt{2} \notin \mathbb{Z}$ נובע כי

$$\mathbb{Z} \subsetneq \mathbb{R}$$

כד. נשים לב כי $\forall q \in \mathbb{Q}$ מתקיים $q \in \mathbb{R}$, לכן

$$\mathbb{Q} \subseteq \mathbb{R}$$

מכיוון ש $\sqrt{2} \in \mathbb{R}$ אך $\sqrt{2} \notin \mathbb{Q}$ נובע כי

$$\mathbb{Q} \subsetneq \mathbb{R}$$

כה. נשים לב כי $\forall n \in \mathbb{N}$ מתקיים $n \in \mathbb{Q}$, לכן

$$\mathbb{N} \subseteq \mathbb{Q}$$

מכיוון ש $\frac{1}{2} \in \mathbb{Q}$ אך $\frac{1}{2} \notin \mathbb{N}$ נובע כי

$$\mathbb{N} \subsetneq \mathbb{Q}$$

כו. נשים לב כי $\forall n \in \mathbb{N}$ מתקיים $n \in \mathbb{Z}$, לכן

$$\mathbb{N} \subseteq \mathbb{Z}$$

מכיוון ש $-1 \in \mathbb{Z}$ אך $-1 \notin \mathbb{N}$ נובע כי

$$\mathbb{N} \subsetneq \mathbb{Z}$$

כז. נשים לב כי $\forall n \in \mathbb{Z}$ מתקיים $n \in \mathbb{Q}$, לכן

$$\mathbb{Z} \subseteq \mathbb{Q}$$

מכיוון ש $\frac{1}{2} \in \mathbb{Q}$ אך $\frac{1}{2} \notin \mathbb{Z}$ נובע כי

$$\mathbb{Z} \subsetneq \mathbb{Q}$$

כח. נשים לב כי $\forall n \in \mathbb{N}$ מתקיים $n \in \mathbb{R}$, לכן

$$\mathbb{N} \subseteq \mathbb{R}$$

מכיוון ש $\sqrt{2} \in \mathbb{R}$ אך $\sqrt{2} \notin \mathbb{N}$ נובע כי

$$\mathbb{N} \subsetneq \mathbb{R}$$

כט. נשים לב כי $\forall n \in \mathbb{Z}$ מתקיים $n \in \mathbb{R}$, לכן

$$\mathbb{Z} \subseteq \mathbb{R}$$

מכיוון ש $\sqrt{2} \in \mathbb{R}$ אך $\sqrt{2} \notin \mathbb{Z}$ נובע כי

$$\mathbb{Z} \subsetneq \mathbb{R}$$

ל. נשים לב כי $\forall q \in \mathbb{Q}$ מתקיים $q \in \mathbb{R}$, לכן

$$\mathbb{Q} \subseteq \mathbb{R}$$

מכיוון ש $\sqrt{2} \in \mathbb{R}$ אך $\sqrt{2} \notin \mathbb{Q}$ נובע כי

$$\mathbb{Q} \subsetneq \mathbb{R}$$

$$d_2 = \left| \frac{0 + 0 \cdot \frac{d}{b} - 0}{\sqrt{1 + \left(\frac{d}{b}\right)^2}} \right| = \left| \frac{0}{\sqrt{1 + \frac{d^2}{b^2}}} \right|$$

$$d_1 = \left| \frac{0 + 0 \cdot \frac{d}{b} - 0}{\sqrt{1 + \left(\frac{d}{b}\right)^2}} \right| = \left| \frac{0}{\sqrt{1 + \left(\frac{d}{b}\right)^2}} \right|$$

$$\boxed{w = (w_1, w_2) = (-b, d)}$$

ב. במקרה והדאטה מאומן על המסווג שלנו
עם הנקודה שנשמטה לא היינו מקבלים
מסווג שונה:
כי - Hard-SVM, כל מה שמעניין אותנו
זה ה-support vectors (תרגול 3 עמוד 55)
וכיוון שהנקודה המושמטת לא היא support vectors
אז התוצאה לא תשתנה

3. המשתנה λ מייצג את ה-tradeoff
בין הרצון שלנו שהמסווג יסווג טוב את האימון
נכון (וכמה שפחות נקודות לא מסווגות נכון)
לבין הרצון למצוא מישור מפריד עם
margin גדול יותר
כאשר λ קטן יש דגש רב יותר על האימון
כאשר λ גדול, יש דגש רב יותר על margin גדול
לכן,

figure 1 → $\lambda = 2$
figure 2 → $\lambda = 200$

4) א. נכתוב קודם שאפשר לכתוב את הווקטור המשקולות w בצורה קומבינציה לינארית:

$w = \sum \alpha_i y_i x_i$

כי w מאותחל לוקטור 0,
ולכן רק את x_i מוסיפים כפול וקטור
$y_i x_i$ ל-w α_i פעמים

לכן, כל מה שאנחנו צריכים שהאלגוריתם ימצא
הוא את α_i לכל i ב- $S = \{(x_i, y_i)\}_{i=1}^m$

באלגוריתם המקורי, החיזוי הוא:

$\hat{y} = sign(w^T x) = sign[(\sum \alpha_i y_i x_i)^T x] =$

מכפלה פנימית

$sign[\sum \alpha_i y_i \langle x_i, x \rangle]$

כעת, נחליף את המכפלה הפנימית ב-
בגרעין: kernel $k(x, x')$
ונקבל את החיזוי:

$\hat{y} = sign[\sum \alpha_i y_i k(x_i, x)]$

כעת נכתוב את האלגוריתם -

```
init α = (0, 0 .. 0) , len(α) = m
for t = 1, 2 ...
    if (∃i s.t y_i ≠ ŷ(x_i)) : α[i] += 1
    else : return α
```

ב. נוכיח נכונות

בדרך דומה הוכחנו את נכונות (א) האלגוריתם

במקרה:

בגלל שגם באלגוריתם המקורי

$w = \sum \alpha_i \psi(x_i)$

נוכל את האלגוריתם שלנו

נשים לב כי $\psi(x)$ לא נדרש לנו

מידע רק על מכפלות פנימיות

כיוון שהאלגוריתם משתמש רק במכפלות

פנימיות

מכאן שהאלגוריתם מתכנס כשמשתמשים ב ψ

פריד לינארית ב F