שאלה 1:

(1) כפי שרואים בשרטוט ה Data אינו פריד לינארית ולכן לא ניתן להפעיל hard-SVM כדי לקבל פתרון

(3) נניח כי w_0 פתרון בעיית ה hard-SVM:

$\forall i,\ y_i\langle w, x_i\rangle > 0$

$$w_0 = \arg\max_{w:\|w\|=1}\left\{\min_{i\in[m]}|\langle w, x_i\rangle|\right\}$$

$$margin = \min_{i\in[m]}|\langle \hat{w}, x_i\rangle| \underset{\hat{w}=\frac{w_0}{\|w_0\|}}{=} \min_{i\in[m]}\left|\left\langle \frac{w_0}{\|w_0\|}, x_i\right\rangle\right|$$

$$= \frac{1}{\|w_0\|}\underbrace{\min_{i\in[m]}|\langle w_0, x_i\rangle|}_{1}\overset{(*)}{=}\frac{1}{\|w_0\|}$$

(*) מכיוון ש w_0 פתרון ולכן מקיים את האילוצים

$|y_i\langle w_0, x_i\rangle| \ge 1$ ומכיוון ש $y_i = \mp 1$

ולכן מתקיים $|\langle w_0, x_i\rangle| \ge 1 \underset{\text{המינימום (פתרון)}}{\Leftarrow} |\langle w_0, x_i\rangle| = 1$

(4) אני רואה לפי השרטוטים שככל ש C קטן ה margin גדול ופחות הסיווג נעשה טוב, זה חלק מהרגולריזציה שעשה ה... לפי מה שלמדנו בהרצאה:

$\uparrow C$ $\uparrow$ סיווג $\downarrow margin$

5. נשים לב שכל ש-C גדל גם val/train error קטן
ומה נובע מזה? הסבר בסעיף הקודם, ככל ש-C גדל המargin
קטן ואנו מדגישים על הסיווג שיהיה יותר מדויק ולכן
השגיאות הולכות וקטנות

6. מצאנו נתונים הנוכחית גבוה יותר לא בהכרח
עוזר לשיפור הדיוק על הולידציה כמו שראינו!
בהרצאה ויש מקרים שבהם הכמות מספר
המימדים הופכת ה-Data לפריד ליניארי.
ולכן לא לכל 60 הנתונים המתאימים מתכנס
ולכן כשמכניסים נתונים עלינו לאפיין הנתונים ולהחליט
אם הגדלת מספר ה נתונים עוזרת בדיוק הולידציה

HW2_096411

May 15, 2022

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
import pandas as pd

from sklearn.model_selection import train_test_split
# use seaborn plotting defaults
import seaborn as sns; sns.set()
```

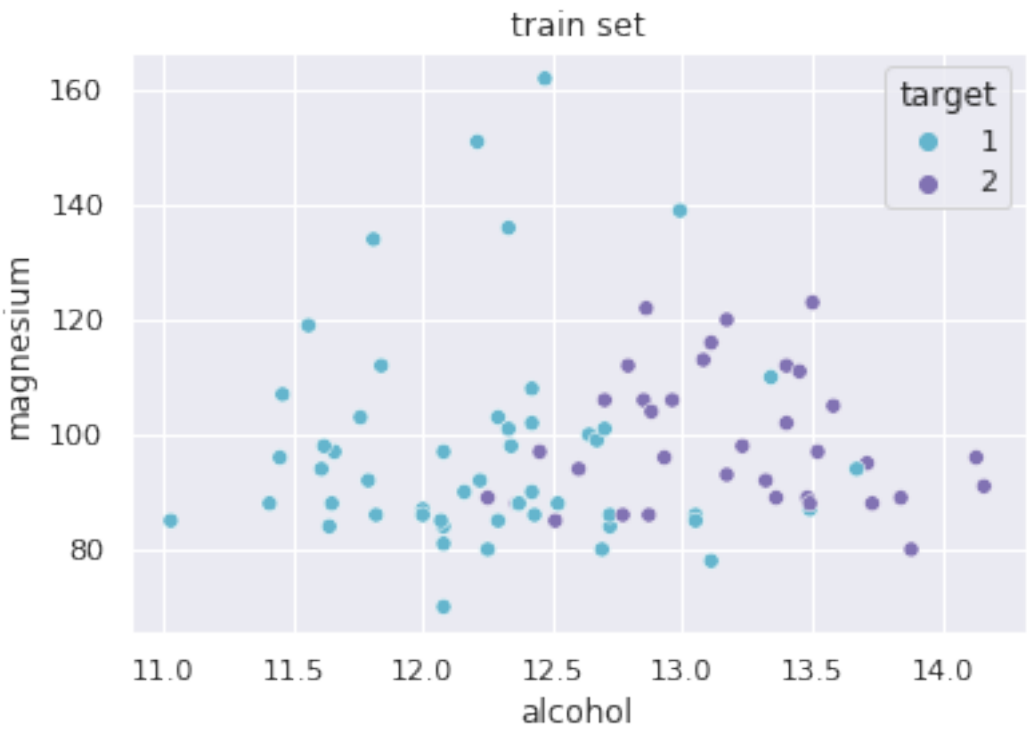
[ ]:
```
from sklearn.datasets import load_wine
# Read the wine dataset
dataset = load_wine()
df = pd.DataFrame(data=dataset['data'], columns=dataset['feature_names'])
df = df.assign(target=pd.Series(dataset['target']).values)
```

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

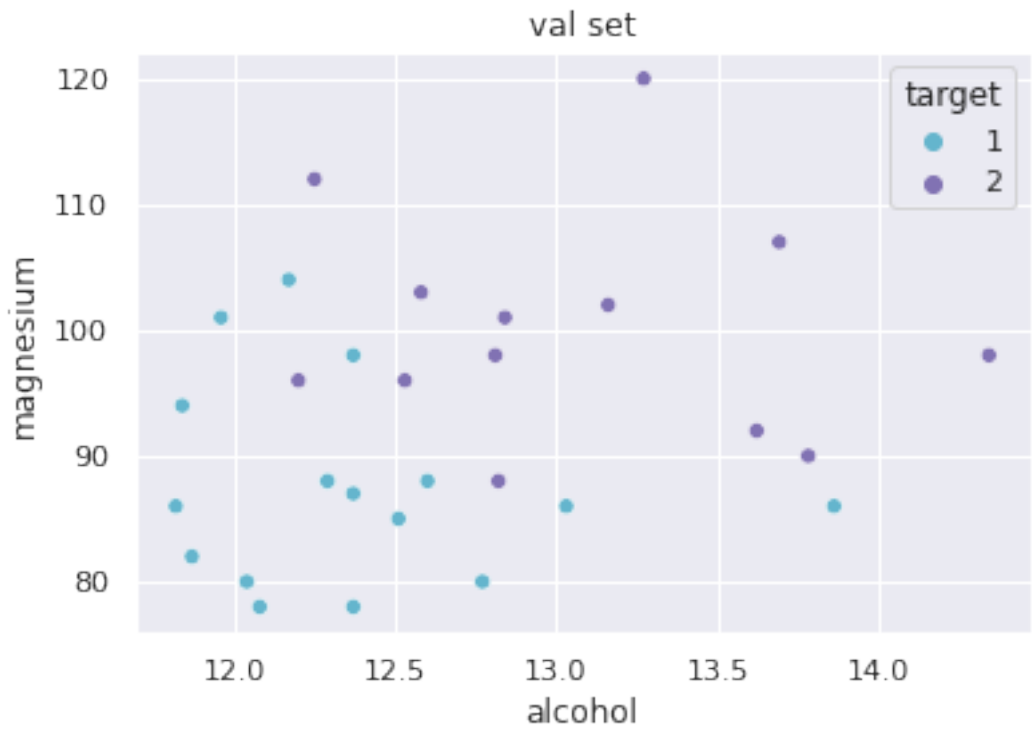
[ ]:
```
from sklearn.svm import SVC
```

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

[ ]:
```
#1.1
sns.scatterplot(data=train_df, x='alcohol', y='magnesium',␣
 ↪hue='target',palette=['c','m'])
plt.title("train set")
plt.show()
```



[ ]:
```
#1.2
sns.scatterplot(data=val_df, x='alcohol', y='magnesium',␣
 ↪hue='target',palette=['c','m'])
plt.title("val set")
plt.show()
```

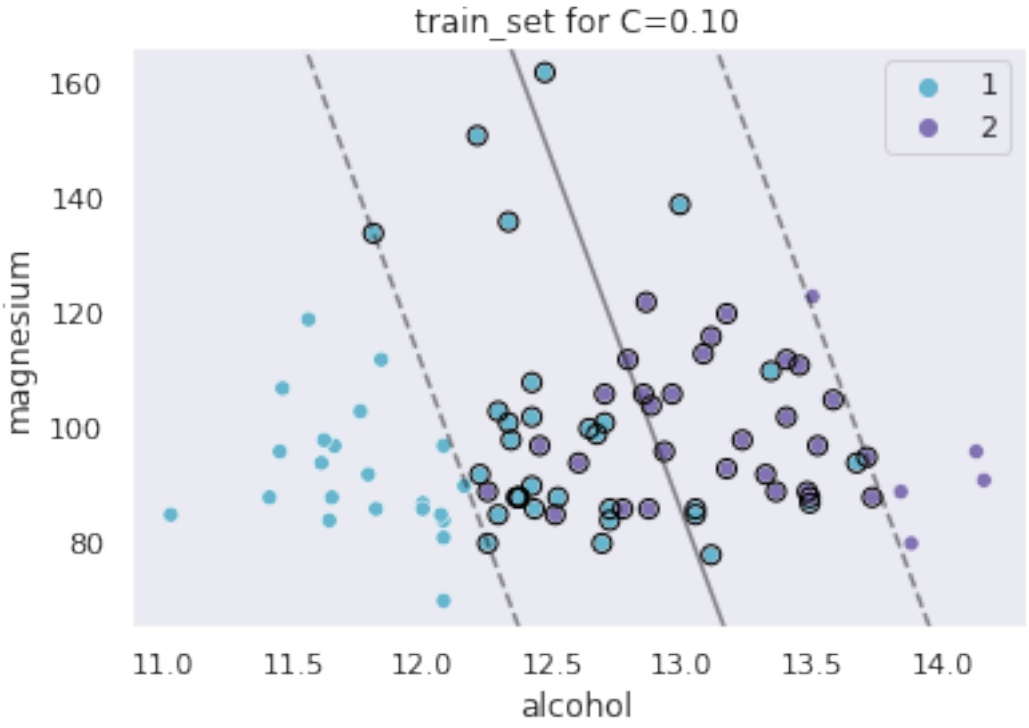


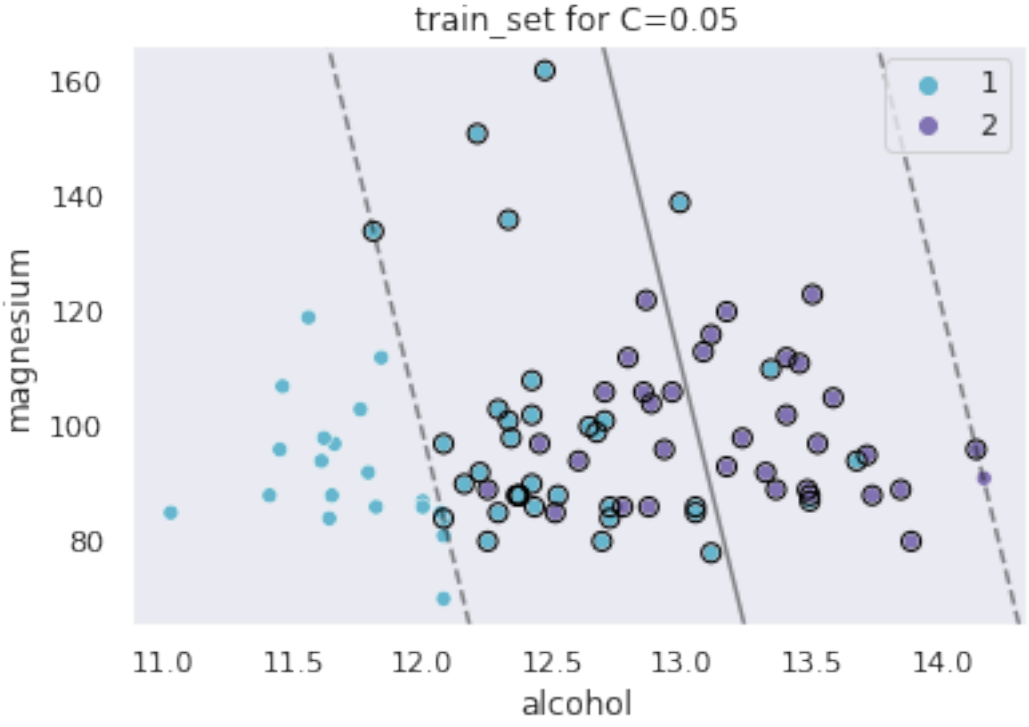
#1- hard-SVM

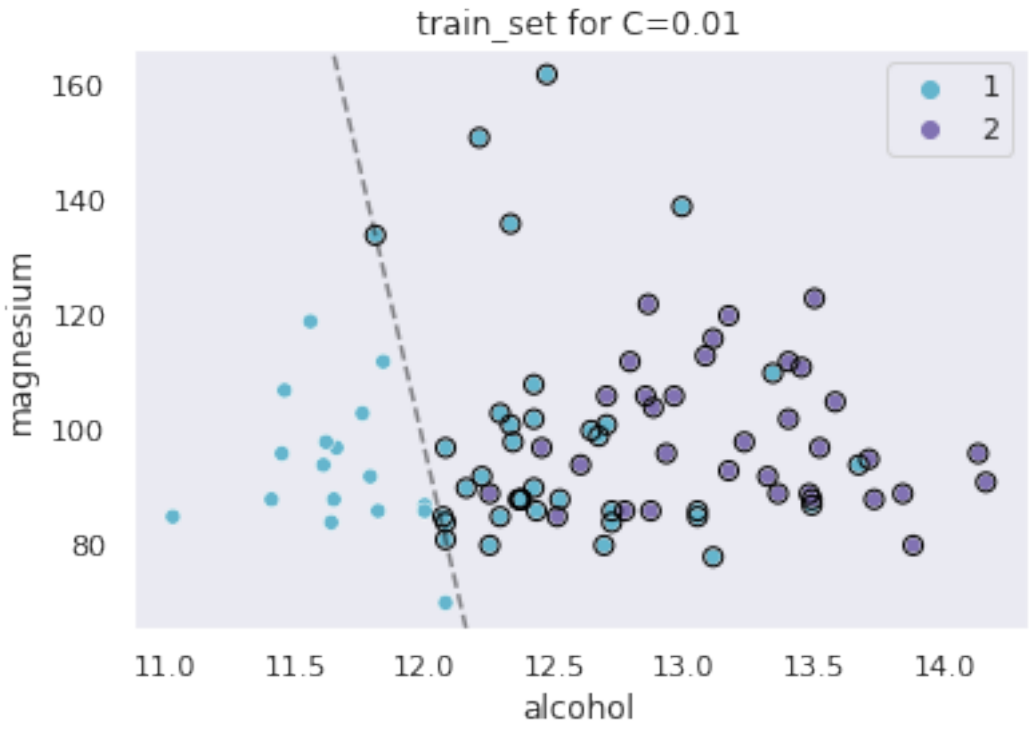
[ ]:
```
#2
from sklearn.svm import SVC

def plot_svc_decision_function(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(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)

    # plot decision boundary and margins
    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', edgecolor='black');
    ax.set_xlim(xlim)
    ax.set_ylim(ylim)
```

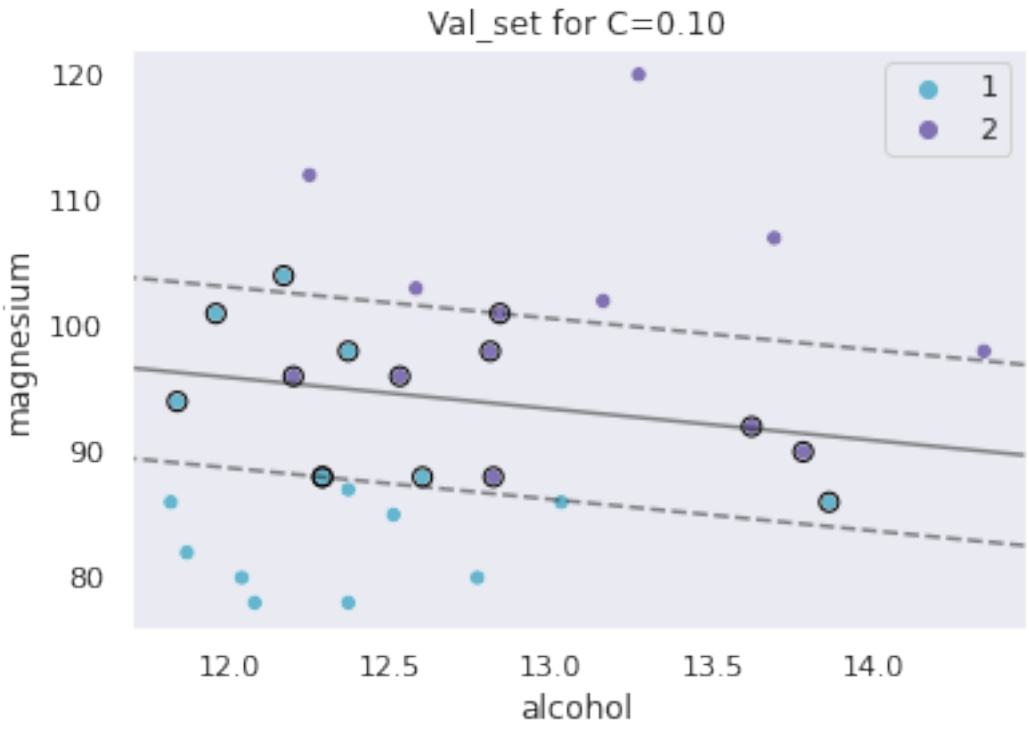
[ ]:
```
#2.1
C=[0.1, 0.05, 0.01]
for c in C:
  model = SVC(kernel='linear',C=c)

  sns.scatterplot(data=train_df, x="alcohol", y="magnesium",␣
 ↪hue="target",palette=['c','m'])
  X_t=train_df.to_numpy()
  y_t=X_t[:,-1]
  X_t=np.delete(X_t,2,1)
  model.fit(X_t,y_t)

  plot_svc_decision_function(model)
  plt.legend()
  s1= "train_set for C=%.2f"
  plt.title(s1%c)
  plt.grid()
  plt.show()
```

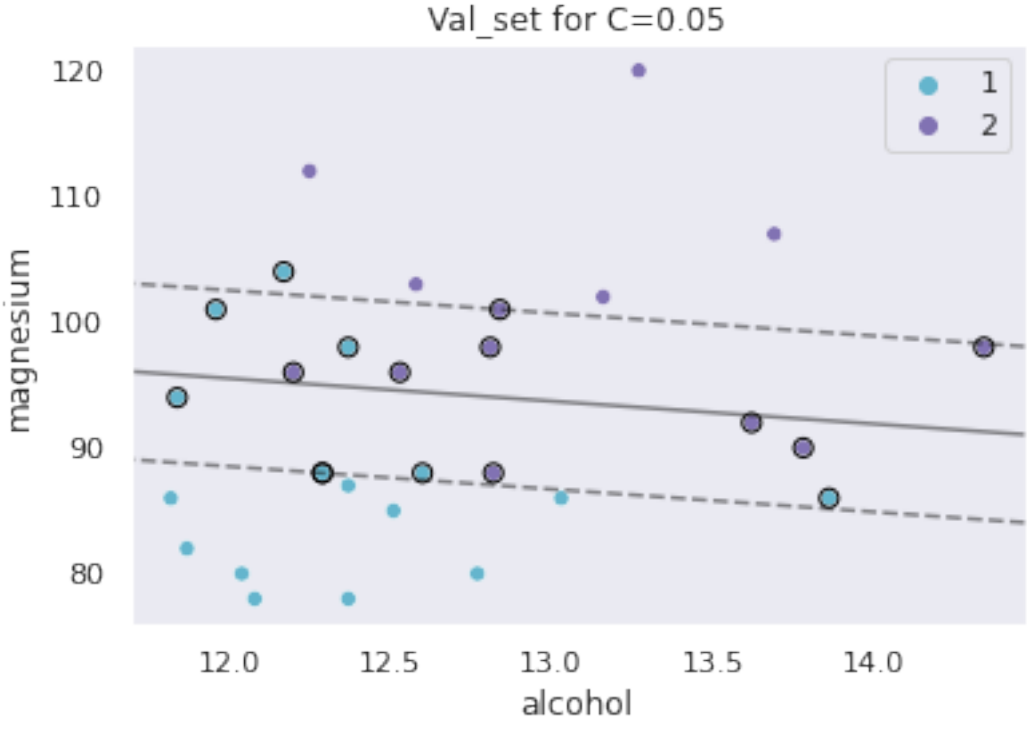

train_set for C=0.10
160
140
120
100
80
magnesium
11.0
11.5
12.0
12.5
13.0
13.5
14.0
alcohol
1
2

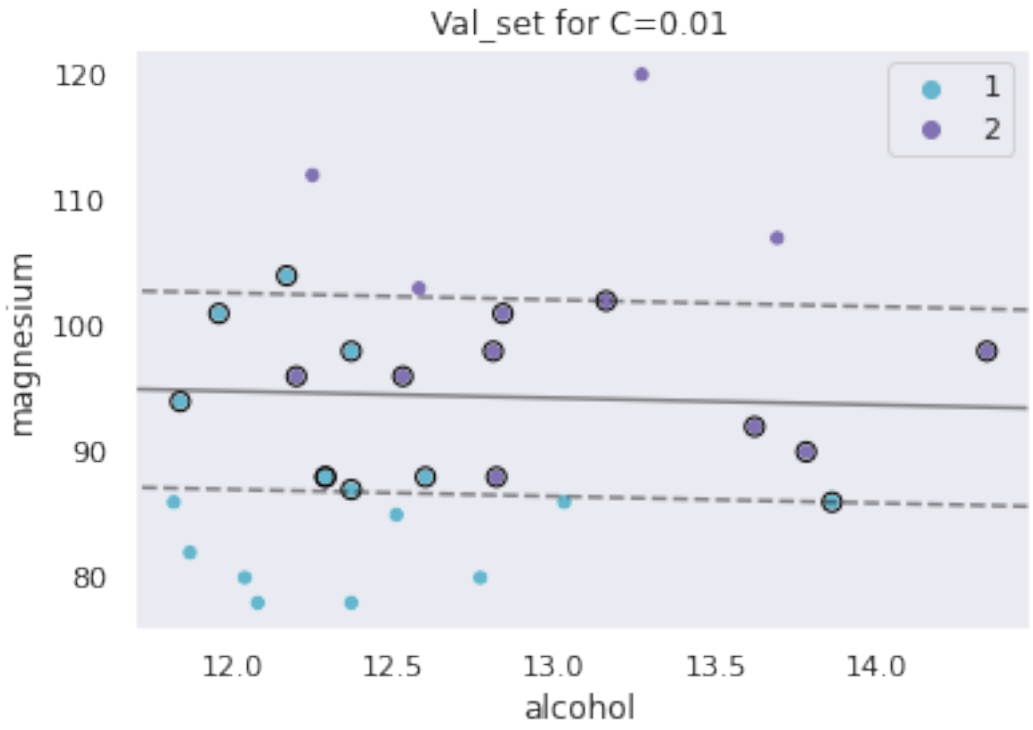

train_set for C=0.05
160
140
120
100
80
magnesium
11.0
11.5
12.0
12.5
13.0
13.5
14.0
alcohol
1
2



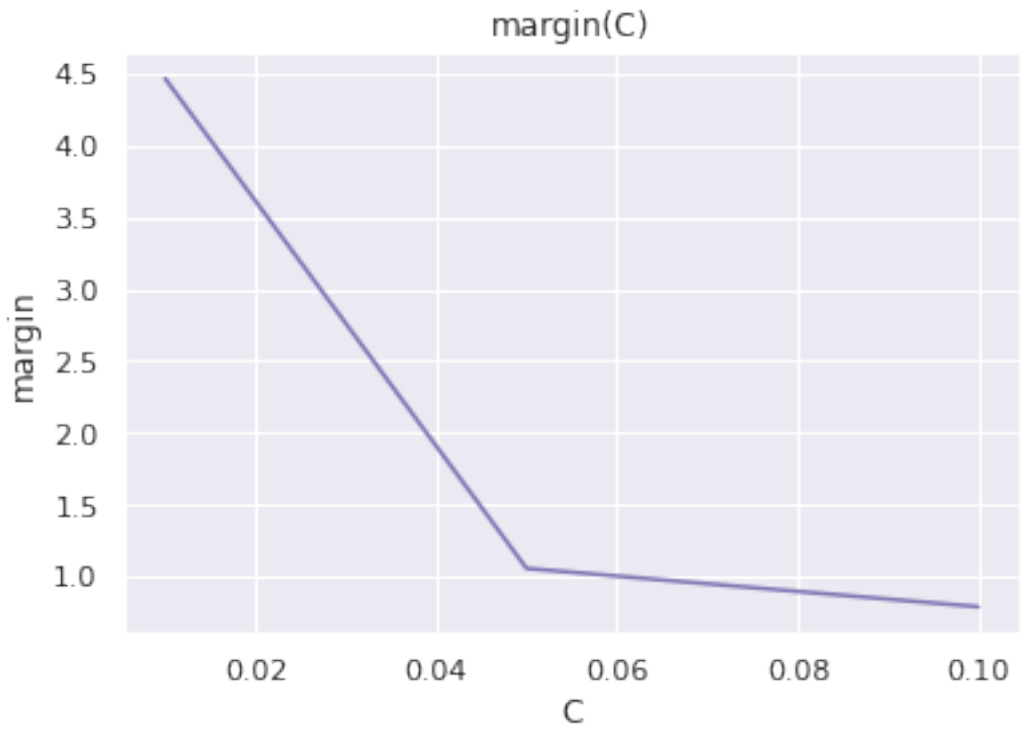
[ ]:
```
#2.2
C=[0.1, 0.05, 0.01]
for c in C:
  model = SVC(kernel='linear',C=c)

  sns.scatterplot(data=val_df, x="alcohol", y="magnesium",␣
 ↪hue="target",palette=['c','m'])
  X_V=val_df.to_numpy()
  y_V=X_V[:,-1]
  X_V=np.delete(X_V,2,1)
  model.fit(X_V,y_V)

  plot_svc_decision_function(model)
  plt.legend()
  s1= "Val_set for C=%.2f"
  plt.title(s1%c)
  plt.grid()
  plt.show()
```


Val_set for C=0.10
120
110
100
90
80
magnesium
12.0
12.5
13.0
13.5
14.0
alcohol
1
2


Val_set for C=0.05
120
110
100
90
80
magnesium
12.0
12.5
13.0
13.5
14.0
alcohol
1
2

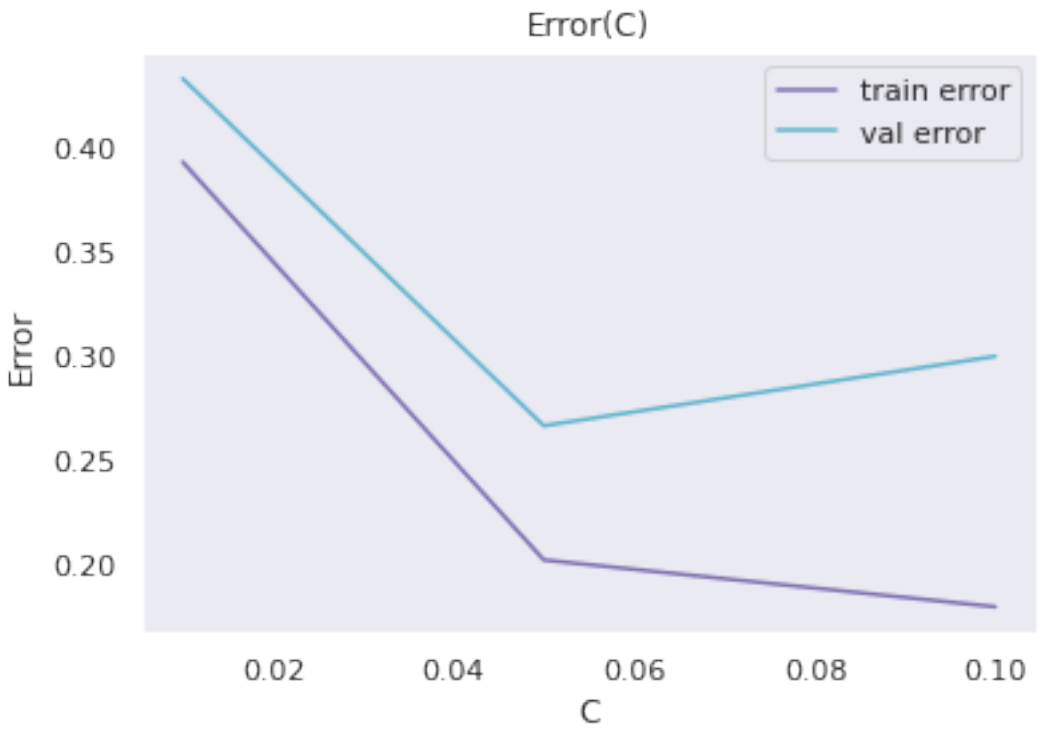


[ ]:
```
#4
C=[0.1, 0.05, 0.01]
train=[]
val=[]
margin=[]
for c in C:
    model=SVC(kernel='linear',C=c)
    model.fit(train_df[['alcohol','magnesium']],train_df['target'])
    train.append(model.
 ↪score(train_df[['alcohol','magnesium']],train_df['target']))
    margin.append(1/(np.linalg.norm(model.coef_)))
fig,ax=plt.subplots()

plt.title('margin(C)')
plt.xlabel('C')
plt.ylabel('margin')
ax.plot(C,margin,c='m')
plt.grid
plt.show()
```



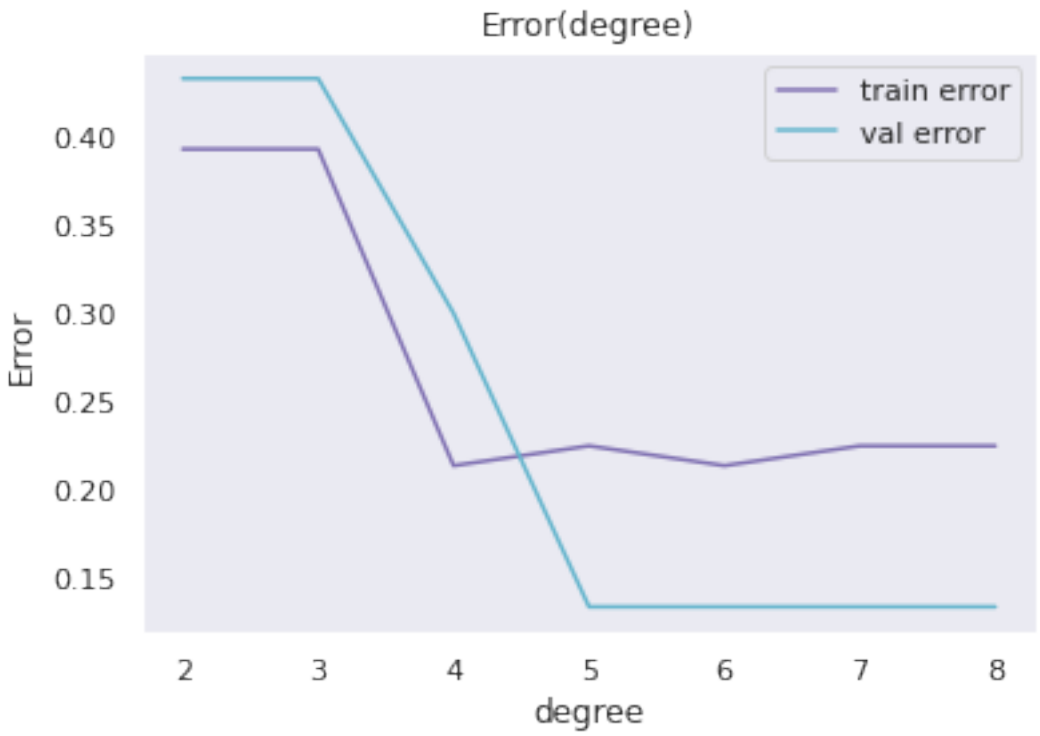
[ ]:
```
#5

train_error=[]
val_error=[]
for c in C:
    model=SVC(kernel='linear',C=c)
    model.fit(train_df[['alcohol','magnesium']],train_df['target'])
    train_error.append(1-model.
 ↪score(train_df[['alcohol','magnesium']],train_df['target']))
    val_error.append(1-model.
 ↪score(val_df[['alcohol','magnesium']],val_df['target']))
fig,ax=plt.subplots()
plt.title('Error(C)')
plt.xlabel('C')
plt.ylabel("Error")
ax.plot(C,train_error,c='m',label='train error')
ax.plot(C,val_error,c='c' ,label='val error')
plt.grid()
plt.legend()
plt.plot()
```

[ ]:
```
[]
```



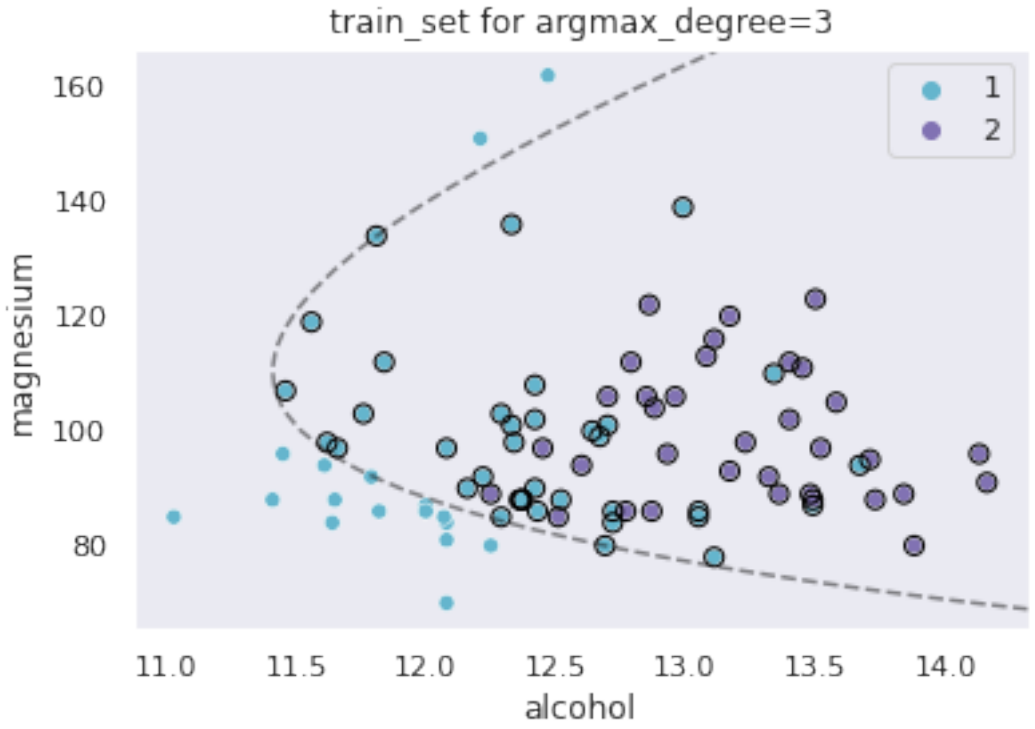
[ ]:
```
#6
degree=[2,3,4,5,6,7,8]
train_error=[]
val_error=[]
for d in degree:
    model=SVC(kernel='poly',C=1,degree=d)
    model.fit(train_df[['alcohol','magnesium']],train_df['target'])
    train_error.append(1-model.
 ↪score(train_df[['alcohol','magnesium']],train_df['target']))
    val_error.append(1-model.
 ↪score(val_df[['alcohol','magnesium']],val_df['target']))
fig,ax=plt.subplots()
plt.title('Error(degree)')
plt.xlabel('degree')
plt.ylabel("Error")
ax.plot(degree,train_error,c='m',label='train error')
ax.plot(degree,val_error,c='c' ,label='val error')
plt.grid()
plt.legend()
plt.plot()
```

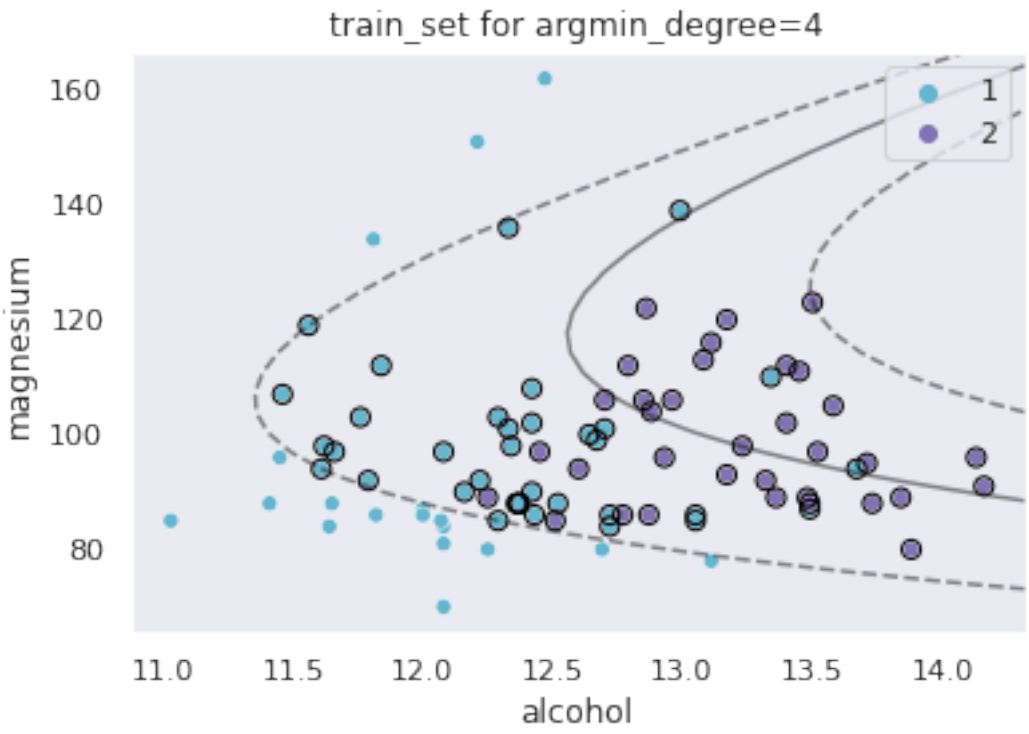
[ ]: []



[ ]:
```
#7.1
model = SVC(kernel='poly',degree=3,C=1)

sns.scatterplot(data=train_df, x="alcohol", y="magnesium",␣
 ↪hue="target",palette=['c','m'])
X_t=train_df.to_numpy()
y_t=X_t[:,-1]
X_t=np.delete(X_t,2,1)
model.fit(X_t,y_t)

plot_svc_decision_function(model)
plt.legend()

plt.title('train_set for argmax_degree=3')
plt.grid()
plt.show()

#7
model = SVC(kernel='poly',degree=4,C=1)

sns.scatterplot(data=train_df, x="alcohol", y="magnesium",␣
 ↪hue="target",palette=['c','m'])
X_t=train_df.to_numpy()
```

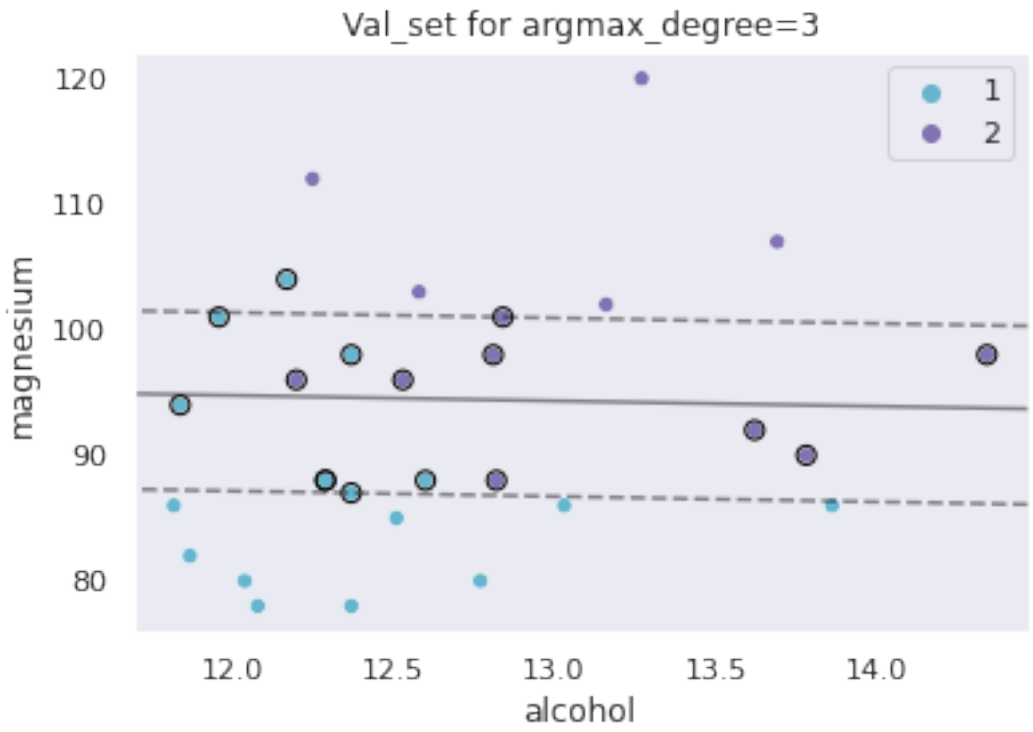
```
y_t=X_t[:,-1]
X_t=np.delete(X_t,2,1)
model.fit(X_t,y_t)

plot_svc_decision_function(model)
plt.legend()

plt.title('train_set for argmin_degree=4')
plt.grid()
plt.show()
```

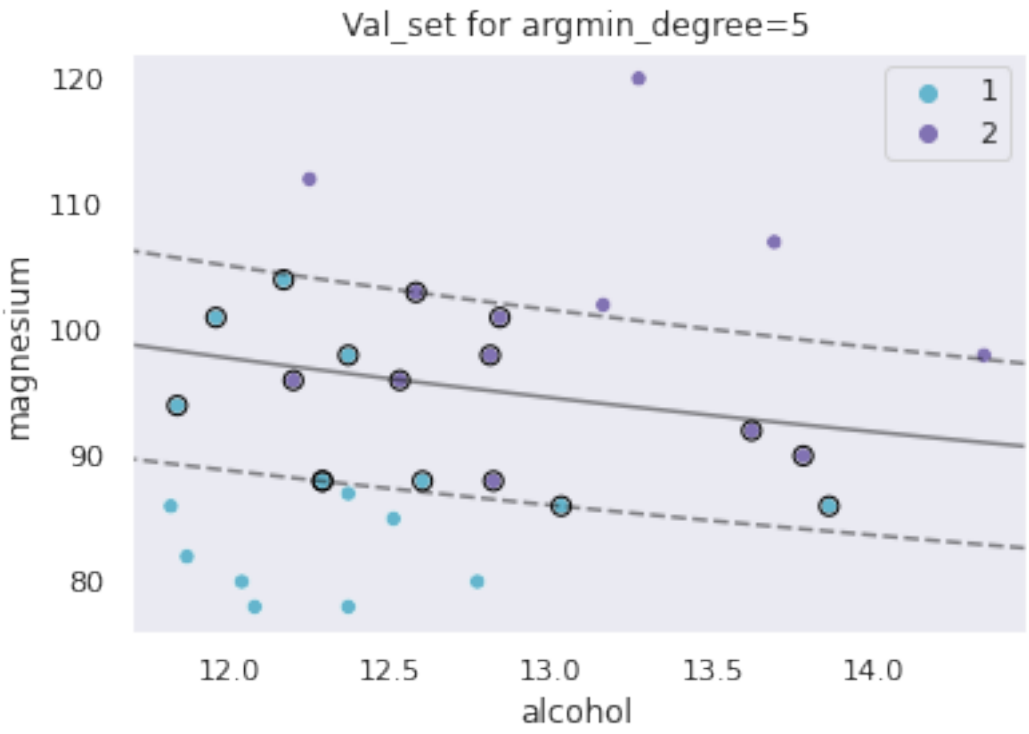




[ ]:
```
#7.2
#7
model = SVC(kernel='poly',degree=3,C=1)

sns.scatterplot(data=val_df, x="alcohol", y="magnesium",␣
 ↪hue="target",palette=['c','m'])
X_V=val_df.to_numpy()
y_V=X_V[:,-1]
X_V=np.delete(X_V,2,1)
model.fit(X_V,y_V)

plot_svc_decision_function(model)
plt.legend()

plt.title('Val_set for argmax_degree=3')
plt.grid()
plt.show()

model2 = SVC(kernel='poly',degree=5,C=1)

sns.scatterplot(data=val_df, x="alcohol", y="magnesium",␣
 ↪hue="target",palette=['c','m'])
X_V=val_df.to_numpy()
```

```
y_V=X_V[:,-1]
X_V=np.delete(X_V,2,1)
model2.fit(X_V,y_V)

plot_svc_decision_function(model2)
plt.legend()

plt.title('Val_set for argmin_degree=5')
plt.grid()
plt.show()
```




Val_set for argmin_degree=5
120
110
100
90
80
magnesium
12.0
12.5
13.0
13.5
14.0
alcohol
1
2

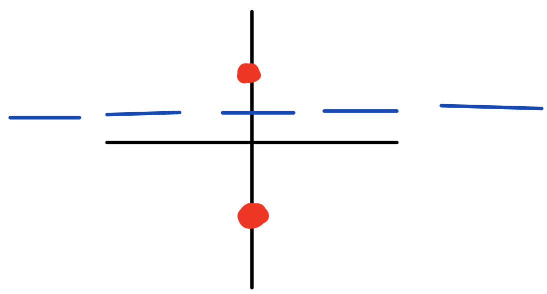
2 : file

```python
import os
import sys
import argparse
import time
import itertools
import numpy as np
import pandas as pd


class PerceptronClassifier:
    def __init__(self):
        """
        Constructor for the PerceptronClassifier.
        """
        # TODO - Place your student IDs here. Single submitters please use a tuple like so: self.ids = (123456789,)
        self.ids = (212699581, 211709597)
        self.W = None
        # W = [[ -w1- ],
        #      [ -w2- ],
        #      ...
        #      [ -wd- ]]
        # where d = X.shape[1]


    def fit(self, X: np.ndarray, y: np.ndarray):
        """
        This method trains a multiclass perceptron classifier on a given training set X with label set y.
        :param X: A 2-dimensional numpy array of m rows and d columns. It is guaranteed that m >= 1 and d >= 1.
        Array datatype is guaranteed to be np.float32.
        :param y: A 1-dimensional numpy array of m rows. it is guaranteed to match X's rows in length (|m_x| == |m_y|).
        Array datatype is guaranteed to be np.uint8.
        """

        K = len(set(y))
        self.W = np.zeros((K, X.shape[1]))
        flag = True

        while flag:
            flag = False
            for i, x in enumerate(X):
                y_hat = int(np.argmax([w.T @ x for w in self.W]))
                y_x = y[i]
                if y_hat != y_x:
                    flag = True
                    self.W[y_x] = self.W[y_x] + x
                    self.W[y_hat] = self.W[y_hat] - x

    def predict(self, X: np.ndarray) -> np.ndarray:
        """
        This method predicts the y labels of a given dataset X, based on a previous training of the model.
        It is mandatory to call PerceptronClassifier.fit before calling this method.
        :param X: A 2-dimensional numpy array of m rows and d columns. It is guaranteed that m >= 1 and d >= 1.
        Array datatype is guaranteed to be np.float32.
        :return: A 1-dimensional numpy array of m rows. Should be of datatype np.uint8.
        """
        ret = list()
        for x in X:
            ret.append(int(np.argmax([w.T @ x for w in self.W])))

        return np.array(ret)

        ### Example code - don't use this:
        # return np.random.randint(low=0, high=2, size=len(X), dtype=np.uint8)


if __name__ == "__main__":

    print("*" * 20)
    print("Started HW2_ID1_ID2.py")
    # Parsing script arguments
    parser = argparse.ArgumentParser()
    parser.add_argument('csv', type=str, help='Input csv file path')
    args = parser.parse_args()

    print("Processed input arguments:")
    print(f"csv = {args.csv}")

    print("Initiating PerceptronClassifier")
    model = PerceptronClassifier()
    print(f"Student IDs: {model.ids}")
    print(f"Loading data from {args.csv}...")
    data = pd.read_csv(args.csv, header=None)
    print(f"Loaded {data.shape[0]} rows and {data.shape[1]} columns")
    X = data[data.columns[:-1]].values.astype(np.float32)
    y = pd.factorize(data[data.columns[-1]])[0].astype(np.uint8)

    print("Fitting...")
    is_separable = model.fit(X, y)
    print("Done")
    y_pred = model.predict(X)
    print("Done")
    accuracy = np.sum(y_pred == y.ravel()) / y.shape[0]
    print(f"Train accuracy: {accuracy * 100 :.2f}%")

    print("*" * 20)

```

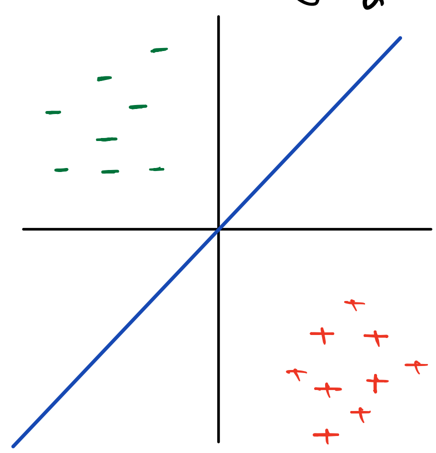
שאלה 3:

1 - הסיווג יכול להשיג שגיאת סיווג אפס
אם״ם ה- נתונים פרידים לינארי

לכן בדוגמאות עבור $d=1$:



עבור $d=2$:



2 ניזכר שלכל $i\in\{1,2,3\}$:

$$y_i\langle w,x_i\rangle \geq 1$$

לכן עבור $i=1$:

$$y_1\langle w,x_1\rangle = -1\langle w,x_1\rangle = -1\cdot w^T x_1 =$$

$$-1\begin{pmatrix}w_1\\ w_2\end{pmatrix}(p,0) = -1\cdot w_1 p \geq 1 \underset{p\neq 0}{\Rightarrow} \boxed{w_1 \leq -\frac{1}{p}}$$

עבור $i=2$:

$$y_2\langle w,x_2\rangle = 1\cdot\langle w,x_2\rangle = 1\cdot\begin{pmatrix}w_1\\ w_2\end{pmatrix}(0,q)$$

$$= w_2 q \geq 1 \underset{q\neq 0}{\Rightarrow} \boxed{w_2 \geq \frac{1}{q}}$$

⇐ נחפש את הוקטור w עם הנורמה הקטנה ביותר- כלומר למצוא את פונקציה המטרה (מינימום שאנו מחפשים או ($\arg\min_{w} \|w\|$

$$\min \|w\|_2^2 = \min\ w_1^2 + w^2 = \frac{1}{p^2} + \frac{1}{q^2}$$

ולכן ה w האופטימלי - פתרון : $w=\left(-\frac{1}{p}, \frac{1}{q}\right)$

$p\neq 0$! $q \neq 0$: אם $p=0$ או $q=0$ נקבל ש x_2 או x_1 שווה

ל $(0,0)$ ובכך לא יתאפשר לבצע SVM ואין פתרון

למציאת הפתרון ($|w_1p|=0<1$ או $|w_2q|=0<1$).

③ מכיוון שהנקודות החסרות בהרצת האלגוריתם הראשונה הן נקודות רחוקות מה margin - כלומר אין נקודות support-vectors אלו כך שהן לא משפיעות על תוצאות האלגוריתם ולכן אם נפעיל האלגוריתם שוב נקבל אותו מפריד

④ כפי שראינו בתרגול 4 הפרמטר λ יוצר tradeoff בין גודל המרווח ובין הסיווג/הפרדה. ככל ש λ קטן יותר אנו נדרשים שיהיה סיווג מדויק יותר ופחות אכפת לנו מה margin לכן בשרטוט 2 ה margin כל כך קטן אך הסיווג הינו יחסית פרפקט' $(2 - \lambda = 2)$ וכאשר אכפת לנו מרווח יותר מהסיווג אנו נבחר λ שהוא גדול כמו בשרטוט 1 $(1 - \lambda = 200)$.

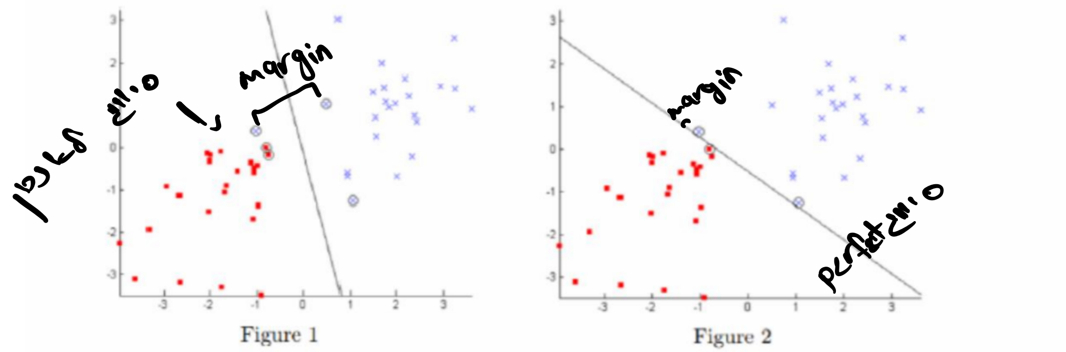


Figure 1 $\lambda = 200$

Figure 2 $\lambda = 2$

שאלה 4

1) ממה שלמדנו בהרצאות המטרה של הפונק' Ψ היא להפוך את המדגם שלנו S
מלא פריד לינארית ב $\mathbb{R}^d$ (המרחב של הנתונים) לפריד לינארית ב F
כלומר נקבל מדגם חדש S^Ψ כך שנקודות הדגימה הן $\{\Psi(x_i)\}_{i=1}^m$
ופונק' ה Kernel $K(x,x')=\langle\Psi(x),\Psi(x')\rangle$ היא המכפלה הפנימית של נק' הפיצ'רים ב F
ולכן אלגוריתם ה Perceptron הופך מ $y_i\langle w,x_i\rangle>0$ ל $y_i\langle\Psi(w),\Psi(x_i)\rangle>0$ $= y_iK(w,x_i)$: $\forall i$
כלומר אלגוריתם ה Perceptron עם פונק' kernel הינו

input: a training set $S=\{(x_i,y_i)\}_{i=1}^m$

initialize: $w^{(1)}=(0,\dots,0)$

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

  if $\exists i$ s.t. $y_i K(w,x_i)\le 0$ then:

    $w^{(t+1)}=w^{(t)}+x_iy_i$

  else

    return $w^{(t)}$

ובהינתן הפלט של האלגוריתם כלל ההחלטה:

$$h(x)=\text{Sign}(K(w,x))$$

2)

נניח שהאלגוריתם מתכנס כלומר החזיר $w^{(t)}$ עבור איזשהו t
עבור t זה מתקיים $\forall i\ \ y_iK(w,x_i)>0$ (כי אחרת התנאי ה if היה מתקיים ונכנס ועדכון
$w^{(t+1)}$ וביצע עוד איטרציה בסתירה להנחה שעצר ב t)
⇐ קיבלנו שעבור w שהוחזר מתקיים $\forall i: y_iK(w,x_i)>0 \equiv \forall i: y_i\langle\Psi(w),\Psi(x_i)\rangle>0$
כלומר אם $y_i=-1$ ⇐ $\langle\Psi(w),\Psi(x_i)\rangle<0$: $\forall i$
$y_i=1$ ⇐ $\langle\Psi(w),\Psi(x_i)\rangle>0$
כלומר $\Psi(w)$ מפריד הנתונים ב F ⇐ S^Ψ פריד לינארית ב F

נניח בשלילה שהאלגוריתם לא מתכנס

כלומר לכל t קיים i כך ש $y_i K(w^{(t)}, x_i) \le 0$ כלומר לכל w תמיד קיימת נק' x_i שמסווגת

לא נכון או במילים אחרות S^{Ψ} אינו פריד לינארית

ולכן אם S^{Ψ} פריד לינארית ב F אז האלגוריתם מתכנס