תרגיל בית 3 – למידה חישובית 1
096411

<u>שמות המגישים</u>:
רוני פרידמן, ת.ז: 205517097
מור לוינבוק, ת.ז: 205451123

<u>שאלה 1:</u>

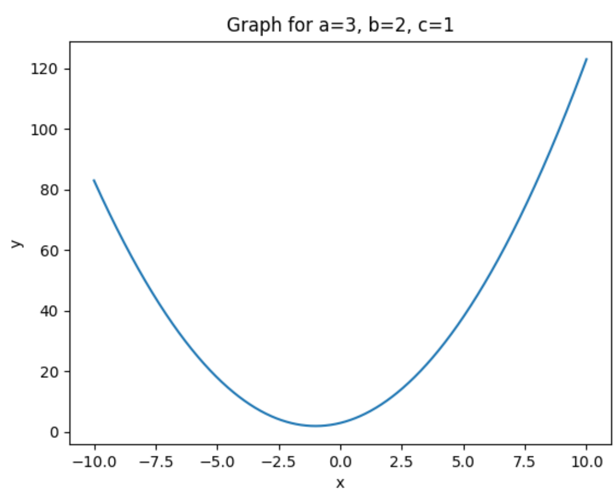
א. בחרנו את הפונקציה כך ש $a = 3, b = 2, c = 1$

$$f(x) = x^2 + 2x + 3$$

הפונקציה בעזרתה נדפיס את הגרף:

```
x = np.linspace(-10, 10, 1000)
y = 1*(x ** 2) + 2 * x + 3
fig, ax = plt.subplots()
plt.title("Graph for a=3, b=2, c=1")
plt.xlabel('x')
plt.ylabel('y')
ax.plot(x, y)
plt.show()
```

והגרף המתקבל הינו:



ב.נוכל לגזור, ונקבל:

$$f'(x) = 2x + 2$$

הפונקציה בפייתון שנגדיר:

```
def grad_f(x):
    return 2 * x + 2
```

ג. נקודת הקיצון תהייה:

$$f'(x) = 0 \rightarrow 2x = -2 \rightarrow x = -1$$

ד. הפונקצייה שמימשנו הינה:

```
def grad_update(x, z):
    return x - z * grad_f(x)
```

ה. הפונקציה שבעזרתה נקבל את המינימום תהייה:

```
z = 0.1
y = 0
x = 10
epsilon = 0.000001
x_t = []
while abs(y-x) > epsilon:
    x_t.append(x)
    y = x
    x = grad_update(x, z)
print(x)
```

- נא לשים לב, כי נשתמש באותה הפונקציה בסעיף ו'.

הפלט שהתקבל הינו `-0.9999964647363027` והוא בקירוב טוב מאוד (לפי האפסילון שבחרנו) נקודת המינימום של הפונקציה.
הסיבה כי הערך המתקבל לא זהה בידיוק ל-x שקיבלנו בסעיף ג', היא כי היינו חייבים להחליט על פרמטר עצירה שבו האלגוריתם מפסיק להתקרב לנק' המינימום שמצאנו בסעיף ג'.
בחרנו באפסילון מסויים (שגיאה שאנו מרשים) ולכן כשהאלגוריתם נכנס לתחום זה, הוא עצר והחזיר לנו את הקירוב בהנתן השגיאה שהצבנו לו.

קצב הלמידה אותו בחרנו הינו 0.1 ובחרנו אותו לשם הדיוק של נקודת המינימום.
ערך האיתחול שבחרנו הינו 10, אך לכל ערך אחר, התוצאה הייתה זהה בקירוב, אם כי האלגוריתם היה מתכנס במספר גדול או קטן יותר של איטרציות.

ו. הקוד מוצג כבר בסעיף ה'.
כאמור, נק' האיתחול והאפסילון שאותו בחרנו, צריכים להשאר קבועים – כמובן שאם נקרב את נק' האיתחול למינימום התאורטי או נגדיל את האפסילון, האלגוריתם יתכנס בפחות איטרציות.
אך, אם נקבע אותם (היות והאפסילון מורה על השגיאה אותה אנחנו מוכנים לקבל, ונק' האיתחול תשאר זהה על מנת שנוכל להשוות בין הפמטרים באופן הוגן) הפרמטר היחידי שנבחן הינו Z – גודל הצעד.
היות ו-Z מציין את גודל הצעד, ככל שהוא יהיה גדול הצעדים בו נתקרב לנק' המינימום יהיו גדולים יותר,

וכך מספר האיטרציות יקטן.
אך אם Z גדול מידי, אנחנו נפספס כל פעם את נק' המינימום, ואולי אף לא נתכנס אלייה אף פעם, לכן נרצה כי הצעד יהיה קטן מהערך $|x_{Min} - s_{start}|$.

ז. הקוד בו נשתמש בשאלה:
חישוב ערך הz האופטימלי

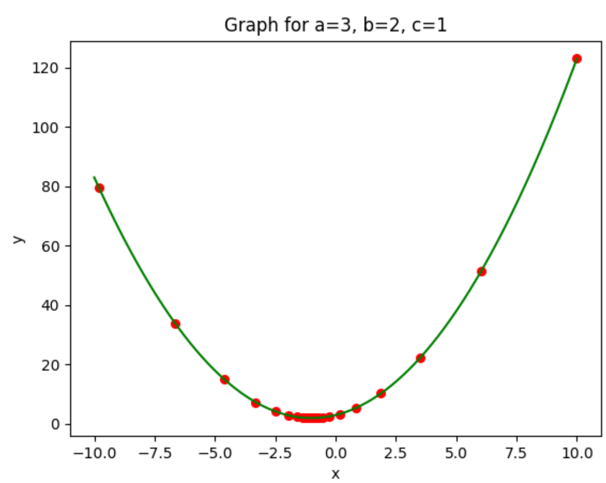
```
z_winner = [0, (100*22 ** 2) / epsilon ** 2]
for z1 in np.arange(0.0, 1.0, 0.1):
    x = 10
    y = 0
    x_t = []
    while abs(y - x) > epsilon:
        x_t.append(x)
        y = x
        x = grad_update(x, z)
    if len(x_t) <= z_winner[1]:
        z_winner = [z1, len(x_t)]
```

הפלט שהתקבל: *z_winner = 9.9*

שרטוט הגרף לאחר שמצאנו Z אופטימלי.

```
y = 0
x = 10
x_t = []
while abs(y-x) > epsilon:
    x_t.append(x)
    y = x
    x = grad_update(x, z_winner[0])
f = [i**2 + 2*i + 3 for i in x_t]
x = np.linspace(-10, 10, 1000)
y = 1*(x ** 2) + 2 * x + 3
plt.title("Graph for a=3, b=2, c=1")
plt.xlabel('x')
plt.ylabel('y')
plt.plot(x, y, c='g')
plt.scatter(x_t, f, c='r', s=30)
plt.show()
```

הגרף שהתקבל:



שאלה 4:

יהיה w מסויים. אנו יודעים כי הפונק' g_j הינה גזירה וקמורה. לכן, היא מקיימת:

$$g_j(u) \geq g_j(w) + \langle u - w, \nabla g_j(w) \rangle \ \ \forall u \in \mathbb{R}^d$$

אנו יודעים כי קיבלנו את g_j מכך שהיא הייתה הפונק' שעבורה קיבלנו את הערך הכי גבוהה בנק' w מכל קבוצת הפונקציות $g_i, i \in [r]$. ולכן, מתקיים שעבור הנק' w:

$$g_j(w) = \max_{i \in [r]} g_i(w) = g(w)$$

בנוסף לכל נק' u שנבחר, קיימת פונ' שמחזירה לה את הערך הכי גדול (לא בהכרח הפונק' g_j)
ולכן, הגדרת פונק' g:

$$g(u) \geq g_j(u) \ \ \forall u \in \mathbb{R}^d$$

אם כך, מתקיים לכל $u \in \mathbb{R}^d$:

$$g(u) \geq g_j(u) \geq g_j(w) + \langle u - w, \nabla g_j(w) \rangle = g(w) + \langle u - w, \nabla g_j(w) \rangle$$

כלומר:

$$g(u) \geq g(w) + \langle u - w, \nabla g_j(w) \rangle$$

שאלה 2:

א. נשתמש בשלושה משפטי עזר מהתרגול ובמשפט המוצג בשאלה:

1. אם $f_i(x)$ קבוצת פונקציות קמורות לכל $i \in [r]$ אזי $g(x) = \max_{i \in [r]} f_i(x)$ גם היא פונ' קמורה.
2. אם $f_i(x)$ קבוצת פונקציות קמורות לכל $i \in [r]$ אזי $g(x) = \frac{1}{r}\sum_{i=1}^{r} f_i(x)$ גם היא פונק' קמורה.
3. נורמה הינה פונקציה קמורה.

ונסמן גם את המשפט מהשאלה:

4. אם f פונ' קמורה, אזי גם f^2 הינה פונק' קמורה.

$$f_1(w) = 0, \qquad f_2(w) = 1 - y_i\langle x_i, w \rangle + b$$

מתקיים שהפונקציות f_1, f_2 קמורות היות וf_1 קבועה, f_2 פונ' לינארית. לכן:
לפי (1) מתקיים $\max\{0, 1 - y_i\langle x_i, w \rangle\}$ גם היא פונק' קמורה. נסמנה $g(w)$.
מסיבה זו, $\frac{1}{m}\sum_{i=1}^{m} g(w)$ גם היא פונ' קמורה, וזאת לפי (2).
נסתכל על $h(w) = \lambda\|w\|^2$
הנגזרת תהייה:

$$h(w) = \lambda\|w\|^2 = \lambda\left(\left(\sum_{i=1}^{n} w_i^2\right)^{\frac{1}{2}}\right)^2 = \lambda\sum_{i=1}^{n} w_i^2 \to\to h'(w) = \lambda\sum_{i=1}^{n} 2w_i = 2\lambda w$$

אנו יודעים מטענה (3) כי הפונקציה $h'(w)$ קמורה.
אם כך מתקיים:

$$\left(h'(w)\right)^2 = h(w)$$

ולפי טענה (4) זה אומר שגם $h(w)$ קמורה.

פונקציית המטרה מוגדרת בתור $\frac{1}{m}\sum_{i=1}^{m} g(w) + h(w)$ וסכום של זוג פונקציות קמורות גם היא פונק' קמורה.

ב. יהיו w_1, w_2 נרצה להראות כי מתקיים:

$$|l(w_1, x_i, y_i) - l(w_2, x_i, y_i| \le \max_k \|x_k\| \, \|w_1 - w_2\|$$

עבור המקרה בו שני המשתנים מסווגים את הנקודה נכון, מתקיים כי אי השוויון טריוויאלי:

$$|l(w_1, x_i, y_i) - l(w_2, x_i, y_i| = |0 - 0| \le \max_k \|x_k\| \, \|w_1 - w_2\|$$

עבור המקרה בו שניהם מסווגים לא נכון את הנקודה (x_i, y_i):

$$|l(w_1, x_i, y_i) - l(w_2, x_i, y_i| = |y_i\langle w_2, x_i\rangle - y_i\langle w_1, x_i\rangle| =^1 |\langle w_2, x_i\rangle - \langle w_1, x_i\rangle|$$

(1) בלי הגבלת הכלליות, ש$y_i = 1$. במקרה שהוא שווה -1 אנחנו פשוט מחליפים מיקום, ואין שינויי בסופו של דבר.

$$|\langle w_2, x_i\rangle - \langle w_1, x_i\rangle| = |\langle x_i, w_2 - w_1\rangle| \le^{Cauchy-Shwartz} \|x_i\|\|w_2 - w_1\| \le$$
$$\le \max_k \|x_k\| \, \|w_2 - w_1\|$$

ג. לפי משפט, אם קיימת פונק' $g(x)$ שניתן לבטא אותה על ידי:

$$g(x) = \max_i g_i(x)$$

כאשר קבוצת הפונקציות $g_i(x)$ כולן גזירות וקמורות. בהנתן x מסויים יהיה $j \in argmax_i \; g_i(x)$ אזי נוכל להגיד כי:

$$\nabla \mathrm{g_j}(x) = \partial g(x)$$

במקרה שלנו $x = w$. נחשב את ה*Subgradient* של פונ' המטרה.

<u>Subgradient</u> <u>לפי w:</u>

פונ' המטרה היא סכום של שני ביטויים, כאשר $\lambda\|w\|^2$ מקיימת כי ה*gradient* שלה בזמן t הינו:

$$\nabla(\lambda\|w\|^2) = 2\lambda w_t$$

שזהו הכיוון של w בזמן הנתון.

הביטויי השני, מאותו הגיון, נוכל להשיגו על ידי סכימת על ה*Gradients* או *Subgradients* של כל הנקודות במדגם.

כלומר מתקיים:

$$\partial f(w,b) = 2\lambda w + \frac{1}{m}\sum_{i=1}^{m} \partial l(w, b, x_i, y_i) = \frac{1}{m}\sum_{i=1}^{m} 2\lambda w + \partial l(w, b, x_i, y_i)$$

כאשר לפי המשפט שציינו בתחילת הסעיף מתקיים(היות ופונ' ה*Hinge-Loss* היא מקסימום של פונק' גזירות וקמורות):

$$\partial l(w, b, x_i, y_i) = \begin{cases} 0, & if\ 1 - y_i(\langle x_i, w\rangle + b) \le 0 \\ -y_i x_i, & if\ 1 - (y_i\langle x_i, w\rangle + b) > 0 \end{cases}$$

ועבור הנקודה (x_i, y_i) שנבחרה:

$$\partial f(w, b, x_i, y_i) = \begin{cases} 2\lambda w, & if\ 1 - y_i(\langle x_i, w\rangle + b) \le 0 \\ 2\lambda w - y_i x_i\,, & if\ 1 - (y_i\langle x_i, w\rangle + b) > 0 \end{cases}$$

Subgradient לפי b:

הפעם מוגדר כי:

$$\partial f(w,b) = \frac{1}{m}\sum_{i=1}^{m} \partial l(w,b,x_i,y_i)$$

היות ו- $\frac{\mathrm{d}}{\mathrm{d(b)}}\lambda\|w\|^2 = 0$.

לכן הפעם מתקיים:

$$\partial l(w,b,x_i,y_i) = \begin{cases} 0, & if\ 1 - y_i(\langle x_i, w\rangle + b) \le 0 \\ -y_i, & if\ 1 - y_i(\langle x_i, w\rangle + b) > 0 \end{cases}$$

היות ונגזור את הפונק' $1 - y_i(\langle x_i, w\rangle + b)$ לפי b.

ועבור נק' ספציפית שנבחרה:

$$\partial f(w,b,x_i,y_i) = \partial l(w,b,x_i,y_i) = \begin{cases} 0, & if\ 1 - y_i(\langle x_i, w\rangle + b) \le 0 \\ -y_i, & if\ 1 - y_i(\langle x_i, w\rangle + b) > 0 \end{cases}$$

ד.הפונקציה:

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


def svm_with_sgd(X, y, lam=0, epochs=1000, l_rate=0.01, sgd_type='practical'):
    np.random.seed(2)
    m = np.shape(X)[0]
    d = np.shape(X)[1]
    w = np.random.rand(d,)
    b = np.random.rand()
    if sgd_type == 'practical':
        for j in range(epochs):
            perm = np.random.permutation(m)
            for i in perm:
                flag = 1-y[i]*(np.inner(X[i], w)+b)
                if flag <= 0:
                    dw = lam * w * 2
                    db = 0
                else:
                    dw = lam * w * 2 - X[i] * y[i]
                    db = -y[i]

                w = w - l_rate * dw
                b = b - l_rate * db
        return w, b

    if sgd_type == 'theory':
        w_avg = w / (epochs*m)
        b_avg = b / (epochs*m)
        for j in range(m*epochs):
            i = np.random.randint(0, m)
            flag = 1-y[i]*(np.inner(X[i], w)+b)
            if flag <= 0:
                dw = lam * w * 2
                db = 0
            else:
                dw = lam * w * 2 - X[i] * y[i]
                db = -y[i]

            w = w - l_rate * dw
            b = b - l_rate * db
            w_avg = w / (epochs * m)
            b_avg = b / (epochs * m)
        return w_avg, b_avg
```

ה. הפונקציה:

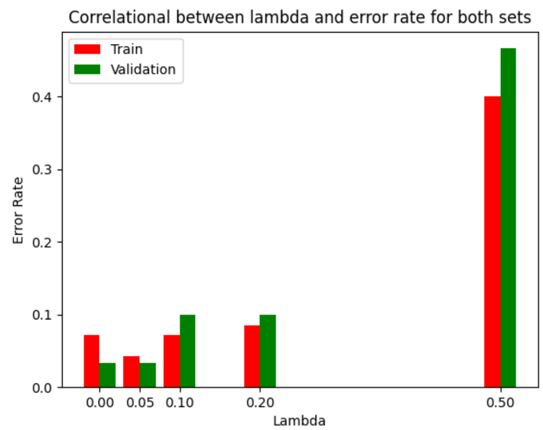
```python
def calculate_error(X, y, w, b):
    m = np.shape(X)[0]
    error_count = 0
    y_predicted = []
    for i in range(m):
        flag = np.inner(X[i, ], w) + b
        if flag > 0:
            y_predicted.append(1)
        else:
            y_predicted.append(-1)

    for i in range(m):
        if y_predicted[i] != y[i]:
            error_count += 1

    return error_count/m
```

ו. הקוד:

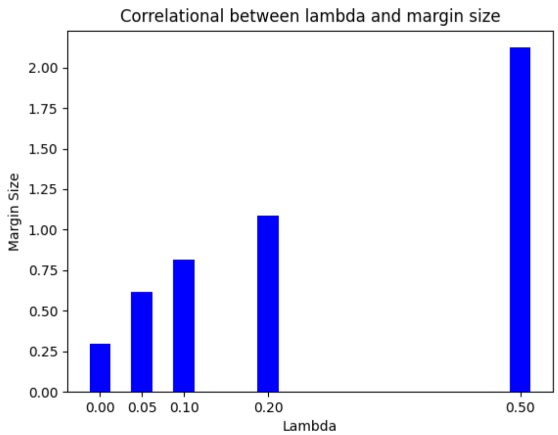
```python
    X, y = load_iris(return_X_y=True)
    X = X[y != 0]
    y = y[y != 0]
    y[y == 2] = -1
    X = X[:, 2:4]
    X_train, X_val, y_train, y_val = train_test_split(X, y, test_size
=0.3, random_state=0)
    lambdas = [0, 0.05, 0.1, 0.2, 0.5]
    models = []
    margins = []
    for lam in range(len(lambdas)):
        w, b = svm_with_sgd(X_train, y_train, lambdas[lam])
        margin = 1/np.linalg.norm(w)
        margins.append(margin)
        models.append([w, b])
    train_error = []
    val_error = []
    for i in range(len(models)):
        train_error.append(calculate_error(X_train,y_train, models[i
][0], models[i][1]))
        val_error.append(calculate_error(X_val, y_val, models[i][0],
models[i][1]))


    fig1, ax1 = plt.subplots()
    ax1.set_xlabel("Lambda")
    ax1.set_ylabel("Error Rate")
    ax1.bar([l-0.02/2 for l in lambdas], train_error, width=0.02,
color='r', label='Train')
    ax1.bar([l + 0.02 / 2 for l in lambdas], val_error, width=0.02,
color='g', label='Validation')
    ax1.set_xticks(lambdas)
    plt.title("Correlational between lambda and error rate for both
sets")
    plt.legend()
    plt.show()
```

הגרפים שהתקבלו:

גרף a:



גרף b:

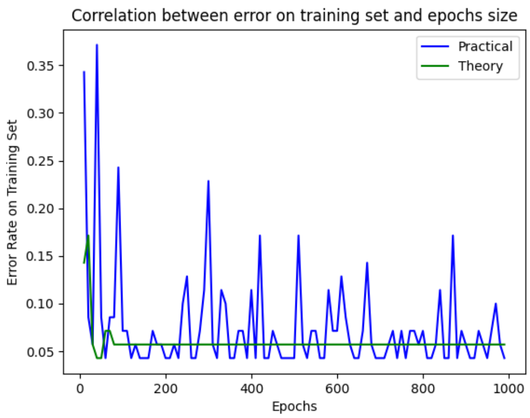


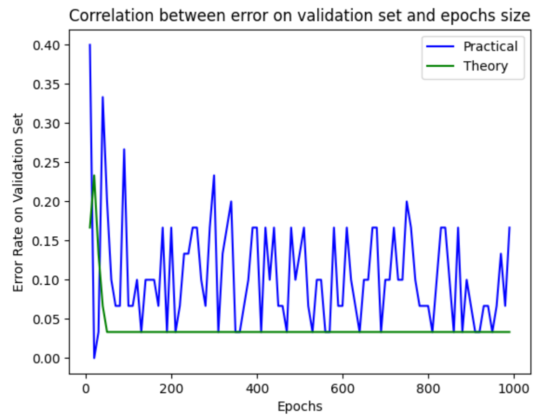
המודל שניראה לנו הכי מהימן הוא המודל המתקבל על ידי $\lambda = 0.05$
אנו יכולים לראות כי השגיאה בסט האימון נמוכה מאוד, אך גם בסט הולידציה (מה שמראה כי אין *Overfit*) והיות והוא בעל ערכי השגיאה (לשני הסטים) הכי נמוכים, נבחר בערך זה.

ז.הקוד בו השתמשנו:

```
epoch_list = range(10, 1000, 10)
lam1 = 0.05
train_error_practical = []
train_error_theory = []
val_error_practical = []
val_error_theory = []
for epoch in epoch_list:
    w_p, b_p = svm_with_sgd(X_train, y_train, epochs=epoch, lam=lam1)
    w_th, b_th = svm_with_sgd(X_train, y_train, epochs=epoch, lam=lam1, sgd_type='theory')
    train_error_practical.append(calculate_error(X_train, y_train, w_p, b_p))
    train_error_theory.append(calculate_error(X_train, y_train, w_th, b_th))

plt.xlabel("Epochs")
plt.ylabel("Error Rate on Training Set")
plt.plot(epoch_list, train_error_practical, color='b', label='Practical')
plt.plot(epoch_list, train_error_theory, color='g', label='Theory')
plt.title("Correlation between error on training set and epochs size")
plt.legend()
plt.show()
for epoch in epoch_list:
    w_p, b_p = svm_with_sgd(X_train, y_train, epochs=epoch, lam=lam1)
    w_th, b_th = svm_with_sgd(X_train, y_train, epochs=epoch, lam=lam1, sgd_type='theory')
    val_error_practical.append(calculate_error(X_val, y_val, w_p, b_p))
    val_error_theory.append(calculate_error(X_val, y_val, w_th, b_th))

plt.xlabel("Epochs")
plt.ylabel("Error Rate on Validation Set")
plt.plot(epoch_list, val_error_practical, color='b', label='Practical')
plt.plot(epoch_list, val_error_theory, color='g', label='Theory')
plt.title("Correlation between error on validation set and epochs size")
plt.legend()
plt.show()
```

הגרפים שהתקבלו:





ניתן לראות שבשני הגרפים, השגיאה התאורתית מתכנסת לערך קבוע מנקודה מסויימת.
זאת כי לפי השיטה התאורתית הראנו בהרצאה כי:

$$\mathbb{E}(v_t|w_t) = \nabla f(w_t)$$

כלומר, היות ואנחנו בוחרים נק' רנדומלית בכל פעם, הווקטור $v_t = \nabla l(w, x_i, y_i)$ יתכנס לתוחלת שלו, שהיא הסאב-גרדיאנט של פונקציית המטרה בנק' w_t. דבר זה גורם לכך שעבור מספר חזרות גדול מספיק, נקבל בקירוב את אותם הערכים עבור w ו-b ולכן נוכל לראות כי השגיאה מתכנסת לערך קבוע בקירוב.
דבר זה אינו קורה בגרף השגיאה הפרקטית, וזאת כי הסאב-גרדיאנט שלפיו אנו מעדכנים את w לא מתכנס אף פעם ל$\nabla f(w_t)$ אלא הוא ממשיך להיות תמיד הגרדיאנט של פונק' ה$Loss$ לפי הנק' הבאה בפרמוטציה שהוגרלה.

שאלה 3:

א. הפונקצייה תהייה:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split


def cross_validation_error(X, y, model, folds):
    samples_sub_arrays = np.array_split(X, folds)
    labels_sub_arrays = np.array_split(y, folds)
    error_k_train = []
    error_k_test = []
    for k in range(folds):
        # Split the array into sub-arrays of approximatlly the same size.
        # Choose the k-th array as the test array, and combine all others as
the training set.
        X_test = samples_sub_arrays.pop(k)
        y_test = labels_sub_arrays.pop(k)
        X_train = np.vstack(samples_sub_arrays)
        y_train = np.concatenate(labels_sub_arrays)
        model.fit(X_train, y_train)
        # Calculate error on training set
        y_pred_train = model.predict(X_train)
        error_count = 0
        for i in range(len(y_train)):
            if y_pred_train[i] != y_train[i]:
                error_count += 1
        error_k_train.append(error_count / len(y_train))
        # Calculate error on test set
        y_pred_test = model.predict(X_test)
        error_count = 0
        for i in range(len(y_pred_test)):
            if y_pred_test[i] != y_test[i]:
                error_count += 1
        error_k_test.append(error_count / len(y_test))
        # Return the chosen sub-arrays back with all training points
        samples_sub_arrays.insert(k, X_test)
        labels_sub_arrays.insert(k, y_test)
    avg_error = (sum(error_k_train)/folds, sum(error_k_test)/folds)
    return avg_error
```

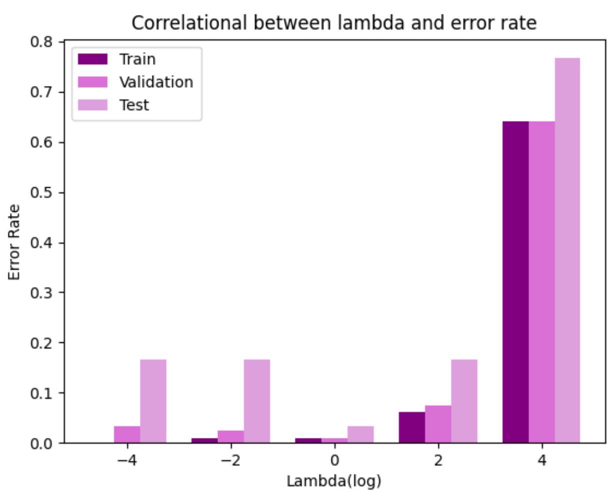
ב. הפונקצייה תהייה:

```
def svm_results(X_train, X_test, y_train, y_test):
    lambdas = [0.0001, 0.01, 1, 100, 10000]
    error_dict = {}
    for lam in lambdas:
        model = SVC(kernel='linear', C=1/lam)
        avg_error = cross_validation_error(X_train, y_train, model, 5)
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        error_count = 0
        for i in range(len(y_test)):
            if y_pred[i]!=y_test[i]:
                error_count += 1
        error_dict["SVM_lambda_" + str(lam)] = (avg_error[0], avg_error[1],
error_count/len(y_test))
    return error_dict
```

ג. הקוד בו השתמשנו:

```
    iris_data = load_iris()
    X, y = iris_data['data'], iris_data['target']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=7)
    lambdas = [0.0001, 0.01, 1, 100, 10000]
    error_dict = svm_results(X_train, X_test, y_train, y_test)
    train_error = []
    val_error = []
    test_error = []
    for key in error_dict.keys():
        train_error.append(error_dict[key][0])
        val_error.append(error_dict[key][1])
        test_error.append(error_dict[key][2])

    lam_rescale = [np.log10(l) for l in lambdas]
    fig1, ax1 = plt.subplots()
    ax1.set_xlabel("Lambda(log)")
    ax1.set_ylabel("Error Rate")
    width = 0.5
    ax1.bar([lam-width for lam in lam_rescale], train_error, width=width,
color='r', label='Train')
    ax1.bar([lam for lam in lam_rescale], val_error, width=width, color='g',
label='Validation')
    ax1.bar([lam+width for lam in lam_rescale], test_error, width=width, color
='b', label='Test')
    ax1.set_xticks(lam_rescale)
    plt.title("Correlational between lambda and error rate")
    plt.legend()
    plt.show()
```

הגרף שהתקבל הינו:



ניתן לראות, כי כאשר λ מאוד קטנה,
השגיאה על סט האימון, היא אפסית
(אפילו שווה ל-0 כאשר $\lambda = 10^{-4}$)
אך, השגיאה על סט הולידציה וסט המבחן
גדולות יותר מאשר $\lambda = 1$.
זאת כי הראנו שעבור ערך נמוך יותר של λ אנחנו
מקטינים את גודל ה*Margin* ובכך פוגעים
ביכולת ההכללה של המודל.
כלומר בעוד שהוא יפעל טוב על סט האימון שנבחר
בתהליך ה*CV*, הוא יבצע עבודה פחות טובה
על סט המבחן וסט הולידציה.
כאשר $\lambda = 10^4$ השגיאה על כל הסטים גדולה מאוד.
הסיבה לכך היא שבעוד מגדילים את ה*Margin* ויכולת ההכללה שלנו נהיית טובה יותר (ניתן לראות כי באופן יחסי, ההבדל בין השגיאה על ה*Train* וה-*Validation* קטנה מאשר ההבדל היחסי עבור ערכים נמוכים יותר) אך אנחנו מרשים למודל בעקבות כך, לבצע טעויות סיווג בבניית המודל.
זה גורם לכך שהמודל מסווג קבוצה גדולה מידי (יחסית לכמות הנתונים שברשותנו) בצורה לא נכונה.
ולמרות שיכולת ההכללה של המודל טובה יותר, ביצועיו באופן כללי גרועים הרבה יותר.
נשים לב, כי הערך שנתן לנו את האיזון הטוב ביותר – גם יכולת ההכללה גבוהה לפי ההפרש בין שגיאת סט המבחן, הולידציה והאימון, וגם רמת שגיאה נמוכה באופן כללי, יהיה בערך $\lambda = 1$.