שאלה 1

סעיף א':

- $S^{(i)}$ - מדגם האימון כאשר הוצאנו ממנו את התצפית ה-i-ית (x_i, y_i) והכנסנו במקומה את התצפית הנוספת (x', y').
- $A(S)$ - אלגוריתם A על מדגם אימון S, כאשר האלגוריתם מחזיר w אשר ממזער את הערך של $f_S(w)$.

סעיף ב':

$$f_S(v)-f_S(u) = L_S(v)+\lambda\|v\|^2-\left(L_S(u)+\lambda\|u\|^2\right) \overset{\text{לפי הגדרה}}{=}$$

$$= \frac{1}{m}\sum_{j=1}^{m} \ell(v,x_i,y_i)+\lambda\|v\|^2-\left(\frac{1}{m}\sum_{j=1}^{m} \ell(u,x_i,y_i)+\lambda\|u\|^2\right)=$$

$$= \frac{1}{m}\left(\sum_{\substack{j=1\\ j\neq i}}^{m} \ell(v,x_i,y_i)+\ell(u,x_i,y_i)\right)+\lambda\|v\|^2-\left(\frac{1}{m}\sum_{\substack{j=1\\ i\neq 1}}^{m} \ell(u,x_j,y_j)+\ell(u,x_i,y_i)+\lambda\|u\|^2\right)=$$

$$= \frac{1}{m}\left(\sum_{\substack{j=1\\ j\neq i}}^{m} \ell(v,x_i,y_i)+\ell(u,x_i,y_i)\right)+\lambda\|v\|^2-\left(\frac{1}{m}\sum_{\substack{j=1\\ i\neq 1}}^{m} \ell(u,x_j,y_j)+\ell(u,x_i,y_i)+\lambda\|u\|^2\right)$$

$$+\frac{1}{m}\ell(v,x',y')-\frac{1}{m}\ell(v,x',y')-\left(\frac{1}{m}\ell(u,x',y')-\frac{1}{m}\ell(u,x',y')\right)=$$

$$= \overbrace{\frac{1}{m}\left(\sum_{\substack{j=1\\ j\neq i}}^{m} \ell(v,x_i,y_i)+\ell(u,x_i,y_i)\right)}^{L_{S^{(i)}}(v)}+\lambda\|v\|^2-\left(\overbrace{\frac{1}{m}\sum_{\substack{j=1\\ i\neq 1}}^{m} \ell(u,x_j,y_j)+\ell(u,x_i,y_i)}^{L_{S^{(i)}}(u)}+\lambda\|u\|^2\right)+$$

$$+\frac{1}{m}\left(\ell(v,x',y')-\ell(v,x',y')\right)-\left(\frac{1}{m}\left(\ell(u,x',y')-\ell(u,x',y')\right)\right) \overset{\text{לפי הגדרה}}{=}$$

$$= L_{S^{(i)}}(v)+\lambda\|v\|^2-\left(L_{S^{(i)}}(u)+\lambda\|u\|^2\right)+\frac{\ell(v,x_i,y_i)-\ell(u,x_i,y_i)}{m}+\frac{\ell(v,x',y')-\ell(u,x',y')}{m}$$

■

סעיף ג':

$$f_S(A(S^{(i)})) - f_S(A(S)) =$$

$$= \overbrace{L_{S^{(i)}}(A(S^{(i)})) + \lambda\|A(S^{(i)})\|^2 - \left(L_{S^{(i)}}(A(S^{(i)})) + \lambda\|A(S)\|^2\right)}^{\leq 0,\ \text{לפי } (*)} +$$

$$+ \frac{\ell(A(S^{(i)}), x_i, y_i) - \ell(A(S), x_i, y_i)}{m} + \frac{\ell(A(S), x', y') - \ell(A(S^{(i)}), x', y')}{m} \leq$$

$$\frac{\ell(A(S^{(i)}), x_i, y_i) - \ell(A(S), x_i, y_i)}{m} + \frac{\ell(A(S), x', y') - \ell(A(S^{(i)}), x', y')}{m}$$

■

(*) $A(S^{(i)})$ זוהי נקודה שממזערת את הפונקציה $f_{S^{(i)}}$ ואילו $A(S)$ זוהי נקודה כלשהי ב- $f_{S^{(i)}}$. לכן בהכרח מתקיים:

$$f_{S^{(i)}}(A(S^{(i)})) \leq f_{S^{(i)}}(A(S)) \Rightarrow L_{S^{(i)}}(A(S^{(i)})) + \lambda\|A(S^{(i)})\|^2 \leq$$

$$\left(L_{S^{(i)}}(A(S)) + \lambda\|A(S)\|^2\right) \Rightarrow L_{S^{(i)}}(A(S^{(i)})) + \lambda\|A(S^{(i)})\|^2 -$$

$$- \left(L_{S^{(i)}}(A(S)) + \lambda\|A(S)\|^2\right) \leq 0$$

סעיף ד':

נגדיר: $h(w)=\lambda\|w\|^2$, $g(w)=L_S(w)$ ומתקיים: $f(w)=g(w)+h(w)$

אזי: לפי למה 1 מההרצאה, מתקיים: $h(w)$ היא 2λ קמורה חזק.
$g(w)$ היא פונקציה קמורה (כסכום של פונק' קמורות) ולכן לפי למה 2
מההרצאה מתקיים כי $f(w)=g(w)+h(w)$ היא 2λ קמורה חזק.
נגדיר u - הנק' שמביאה את הפונק' f למינימום. לפי למה 3 מההרצאות,
כיוון ש- $f(w)$ היא 2λ קמורה חזק ו- u מביאה את f
למינימום מתקיים לכל w:

$$f(w)-f(u)\ge \frac{2\lambda}{2}\|w-u\|^2=\lambda\|w-u\|^2$$

לכן, קיבלנו כי f_S הינה פונקציה 2λ קמורה חזק.
מההגדרה מתקיים כי $A(S)$ היא מינימום של הפונקציה f_S ולכן
לכל נקודה w, ובפרט $w=A(S^{(i)})$ מתקיים:

$$f_S(A(S^{(i)}))-f_S(A(S))\ge\lambda\|A(S^{(i)})-A(S)\|^2$$

בשילוב עם סעיף ג' נקבל:

$$\lambda\|A(S^{(i)})-A(S)\|^2\le f_S(A(S^{(i)}))-f_S(A(S))\le$$

$$\le\frac{\ell(A(S^{(i)}),x_i,y_i)-\ell(A(S),x_i,y_i)}{m}+\frac{\ell(A(S),x',y')-\ell(A(S^{(i)}),x',y')}{m}$$

■

סעיף ה':

נתון כי $\ell(\cdot)$ היא פונקציה ρ-ליפשיצית ולכן מתקיים:

$$* \; \|\ell(A(S^{(i)}),x_i,y_i)-\ell(A(S),x_i,y_i)\|\le\rho\|A(S^{(i)})-A(S)\|$$

$$* \; \|\ell(A(S^{(i)}),x',y')-\ell(A(S),x',y')\|\le\rho\|A(S^{(i)})-A(S)\|$$

לכן, מתקיים:

$$\lambda\|A(S^{(i)})-A(S)\|^2 \overset{\text{לפי סעיף ג'}}{\le} \frac{\ell(A(S^{(i)}),x_i,y_i)-\ell(A(S),x_i,y_i)}{m}+\frac{\ell(A(S),x',y')-\ell(A(S^{(i)}),x',y')}{m} \overset{\text{לפי } * *}{\le}$$

$$\le \frac{1}{m}\rho\|A(S^{(i)})-A(S)\|+\frac{1}{m}\rho\|A(S^{(i)})-A(S)\|=\frac{2\rho}{m}\|A(S^{(i)})-A(S)\| \Rightarrow$$

$$\Rightarrow \lambda\|A(S^{(i)})-A(S)\|^2\le\frac{2\rho}{m}\|A(S^{(i)})-A(S)\| \Rightarrow \|A(S^{(i)})-A(S)\|\le\frac{2\rho}{\lambda m}$$

■

סעיף ו':

נתון כי $\ell(\cdot)$ היא פונקציה ρ-ליפשיצית ולכן מתקיים:

$$\ell(A(S^{(i)}),x_i,y_i)-\ell(A(S),x_i,y_i)\le\rho\|A(S^{(i)})-A(S)\| \overset{\text{סעיף ה'}}{\le} \rho\cdot\frac{2\rho}{\lambda m}=\frac{2\rho^2}{\lambda m}$$

■

סעיף ז':

$L_S(w)=\frac{1}{m}\sum_{i=1}^{m}[\ell(w,x_i,y_i)]$ היא ממוצע השגיאות של הלומד w, בהינתן מדגם האימון S.

$L_D(w)=\mathbb{E}_{(x,y)\sim D}[\ell(w,x,y)]$ הוא תוחלת שגיאות הלומד w, כאשר התוחלת נלקחת מהתפלגות D כלשהי-לא ידועה (מניחים שהדגימות מפולגים D).

- בהינתן מדגם אימון S ולומד w - אנו יכולים לחשב את $L_S(w)$ כי אנחנו יודעים את S ו-w.
- לא נוכל לחשב את $L_D(w)$ כיוון שהוא תלוי בהתפלגות D שאינה ידועה לנו ולכן לא נוכל לחשב את התוחלת.

סעיף ה':

נשים לב כי מתקיים:

משפט היציבות מההרצאה

$$\mathbb{E}_{S\sim \mathcal{D}^m}\left[L_{\mathcal{D}}(A(S)) - L_S(A(S))\right] =$$

$$\mathbb{E}_{(S,x',y')\sim \mathcal{D}^{m+1},\ i\sim U(m)}\left[\ell(A(S^{(i)}),x_i,y_i) - \ell(A(S),x_i,y_i)\right] \le$$

לפי סעיף ז' + לינאריות התוחלת

$$\le \mathbb{E}_{(S,x',y')\sim \mathcal{D}^{m+1},\ i\sim U(m)}\left[\frac{2\rho^2}{\lambda m}\right] = \frac{2\rho^2}{\lambda m} \quad \blacksquare$$

Q2

Q2.1

TPR=recall = TP/(TP+FN) – מודד מה יחס של הסיווגים שסיווגנו כנכונים והם באמת נכונים מתוך כל מה שבאמת היה צריך להיות מסווג נכון. את המדד הזה נרצה למקסם

Presicion = TP/(TP+FP) - מודד מה יחס של הסיווגים שסיווגנו כנכונים והם באמת נכונים מתוך כלל הסיווגים שסיווגנו כנכונים. גם את המדד הזה נרצה למקסם

Q2.2

נבחר לדוגמה בעיית סיווג רפואית, כגון אבחון של סרטן. במקרה זה חשוב לנו לאבחן נכון כמה שיותר חולים (TPR) גם במחיר של לאבחן כמה שהם לא חולים כחולים (presicion). זאת משום שמסוכן יותר שחולה סרטן לא יהיה מאובחן מאשר שאדם בריא יהיה מאובחן ויעבור בדיקות נוספות

Q2.3

נבחר לדוגמא בעיית סיווג "לא דחופה", כגון סיווג של מייל ספאם. כאן יהיה לנו חושב יותר לא לפספס מיילים חשובים (presicion) מאשר שנצטרך לעבור על כמה מיילים שהם ספאם (TPR)

Q2.4

$$P_w(y=1|x)=\sigma(-0.3-0.5x_1+0.5x_2)=\frac{\exp(-0.3-0.5x_1+0.5x_2)}{1+\exp(-0.3-0.5x_1+0.5x_2)}$$

עבור כל $i\in\{0\ldots4\}$

$$P_w(y_0=1|x_0)=0.0356$$

$$P_w(y_1=1|x_1)=0.6682$$

$$P_w(y_2=1|x_2)=0.0007$$

$$P_w(y_3=1|x_3)=0.2142$$

$$P_w(y_4=1|x_4)=0.0219$$

Q2.5

נחשב את ערכי הTPR והFPR עבור כל אחד מהתצפיות

$$for\ 0{:}\,TPR=\frac{2}{2}=1, FPR=\frac{3}{3}=1$$

$$for\ 0.0007{:}\,TPR=\frac{1}{2}, FPR=\frac{3}{3}=1$$

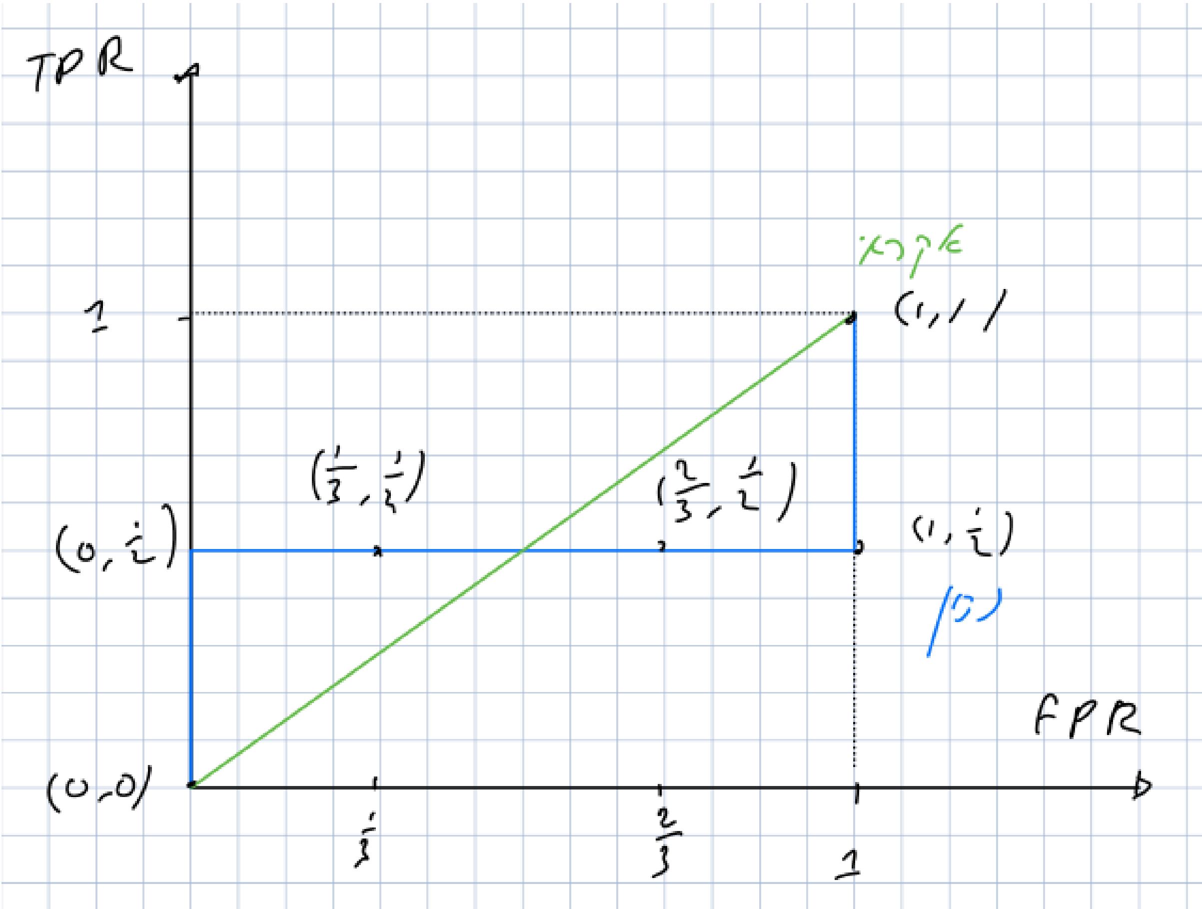
$$for\ 0.0219{:}\,TPR=\frac{1}{2}=1, FPR=\frac{2}{3}$$

$$for\ 0.0356{:}\,TPR=\frac{1}{2}=1, FPR=\frac{1}{3}$$

$$for\ 0.2142: TPR=\frac{1}{2}=1, FPR=\frac{0}{3}=0$$

$$for\ 0.6682: TPR=\frac{0}{2}=0, FPR=\frac{0}{3}=0$$

מכאן:



Q2.6

נחשב את השטח מתחת לגרף:

$$S_{\Delta_{\mathrm{Roc}}}=1*\frac{1}{2}=\frac{1}{2}$$

$$S_{\Delta_{\mathrm{rand}}}=\frac{1*1}{2}=\frac{1}{2}$$

מכאן המודלים שווים אחד לשני, ומודל הAUC-ROC הוא לא טוב יותר

Q3

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

idx2class = {'0': 'T-shirt/top', '1': 'Trouser', '2': 'Pullover', '3': 'Dress', '4': 'Coat',
             '5': 'Sandal', '6': 'Shirt', '7': 'Sneaker', '8': 'Bag', '9': 'Ankle boot'}

1 usage
def fetch_mnist():
    # Download MNIST dataset
    X, y = fetch_openml( name: 'Fashion-MNIST', version=1, return_X_y=True)
    X = X.to_numpy()
    y = y.to_numpy()

    # Randomly sample 7000 images
    np.random.seed(2)
    indices = np.random.choice(len(X),  size: 7000, replace=False)
    X, y = X[indices], y[indices]
    return X, y

1 usage
def show_top_10_images(X, y):
    fig, axs = plt.subplots( nrows: 2,  ncols: 5, figsize=(15, 6))
    axs = axs.ravel()

    for i in range(10):
        img = X[i].reshape(28, 28)
        axs[i].imshow(img, cmap="binary")
        axs[i].axis('off')
        class_index = y[i]
        class_name = idx2class[class_index]
        axs[i].set_title(f"({class_index}, {class_name})")

    plt.tight_layout()
    plt.show()


import numpy as np
```

```python
class SimpleSVM:
    def __init__(self, learning_rate=0.01, lambda_param=0.01, n_iters=1000):
        self.lr = learning_rate
        self.lambda_param = lambda_param
        self.n_iters = n_iters
        self.w = None
        self.b = None

    1 usage (1 dynamic)
    def fit(self, X, y):
        n_samples, n_features = X.shape

        # Initialize parameters
        self.w = np.zeros(n_features)
        self.b = 0

        # Gradient descent
        for _ in range(self.n_iters):
            for idx, x_i in enumerate(X):
                condition = y[idx] * (np.dot(x_i, self.w) - self.b) >= 1
                if condition:
                    self.w -= self.lr * (2 * self.lambda_param * self.w)
                else:
                    self.w -= self.lr * (2 * self.lambda_param * self.w - np.dot(x_i, y[idx]))
                    self.b -= self.lr * y[idx]

    2 usages (2 dynamic)
    def predict(self, X):
        approx = np.dot(X, self.w) - self.b
        return np.sign(approx)
```

```python
def cross_validation_errors(X, y, model, folds):
    n = len(X)
    fold_size = n // folds
    train_errors = []
    val_errors = []

    for i in range(folds):
        start = i * fold_size
        end = (i + 1) * fold_size if i < folds - 1 else n

        X_train = np.concatenate((X[:start], X[end:]))
        y_train = np.concatenate((y[:start], y[end:]))
        X_test = X[start:end]
        y_test = y[start:end]

        # Train the model
        model.fit(X_train, y_train)

        # Predict on training set
        y_train_pred = model.predict(X_train)
        train_error = np.mean(y_train_pred != y_train)
        train_errors.append(train_error)

        # Predict on validation set
        y_test_pred = model.predict(X_test)
        val_error = np.mean(y_test_pred != y_test)
        val_errors.append(val_error)

    average_train_error = np.mean(train_errors)
    average_val_error = np.mean(val_errors)

    return average_train_error, average_val_error
```

```python
def SVM_results(X_train, y_train, X_test, y_test):
    results = {}

    # Linear kernel
    model = SVC(kernel='linear')
    train_error, val_error = cross_validation_errors(X_train, y_train, model, folds: 4)
    model.fit(X_train, y_train)
    test_error = np.mean(model.predict(X_test) != y_test)
    results['SVM_linear'] = (train_error, val_error, test_error)

    # Polynomial kernel
    for d in [2, 4, 6, 8]:
        model = SVC(kernel='poly', degree=d)
        train_error, val_error = cross_validation_errors(X_train, y_train, model, folds: 4)
        model.fit(X_train, y_train)
        test_error = np.mean(model.predict(X_test) != y_test)
        results[f'SVM_poly_{d}'] = (train_error, val_error, test_error)

    # RBF kernel
    for gamma in [0.001, 0.01, 0.1, 1.0, 10]:
        model = SVC(kernel='rbf', gamma=gamma)
        train_error, val_error = cross_validation_errors(X_train, y_train, model, folds: 4)
        model.fit(X_train, y_train)
        test_error = np.mean(model.predict(X_test) != y_test)
        results[f'SVM_rbf_{gamma}'] = (train_error, val_error, test_error)

    return results
```

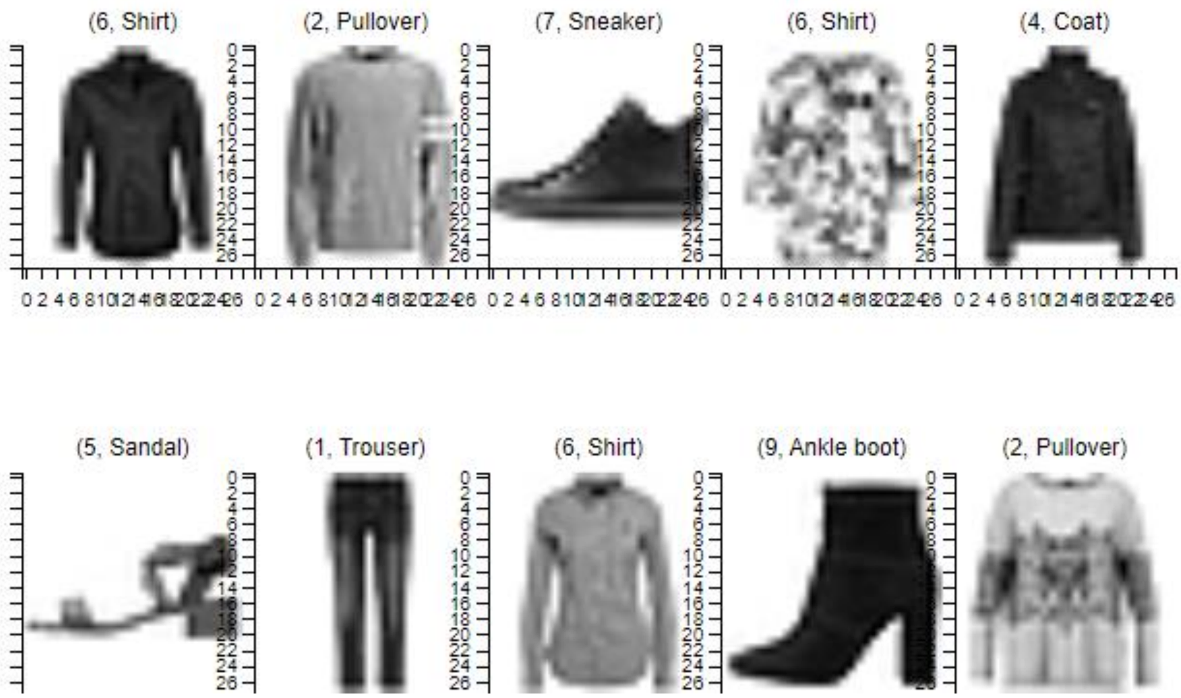
```python
def plot_results(results):
    # Extract keys, and separate the errors
    models = list(results.keys())
    train_errors = [result[0] for result in results.values()]
    val_errors = [result[1] for result in results.values()]
    test_errors = [result[2] for result in results.values()]

    # Set up the bar width and positions
    bar_width = 0.25
    r1 = np.arange(len(models))
    r2 = [x + bar_width for x in r1]
    r3 = [x + bar_width for x in r2]

    # Create the bars
    plt.figure(figsize=(12, 8))
    plt.bar(r1, train_errors, color='b', width=bar_width, edgecolor='grey', label='Train Error')
    plt.bar(r2, val_errors, color='g', width=bar_width, edgecolor='grey', label='Validation Error')
    plt.bar(r3, test_errors, color='r', width=bar_width, edgecolor='grey', label='Test Error')

    # Add labels
    plt.xlabel( xlabel: 'Models', fontweight='bold')
    plt.ylabel( ylabel: 'Error', fontweight='bold')
    plt.xticks([r + bar_width for r in range(len(models))], models, rotation=45, ha="right")
    plt.title('SVM Model Errors Comparison')
    plt.legend()

    # Show plot
    plt.tight_layout()
    plt.show()
```

```python
if __name__ == "__main__":
    # Q3.1
    X, y = fetch_mnist()
    print(f"Q3.1: {X.shape}, {y.shape}")

    # Q3.2
    show_top_10_images(X, y)

    # Q3.3
    X_train, X_test, y_train, y_test = train_test_split( *arrays: X, y, test_size=0.25, random_state=42)
    results = SVM_results(X_train, y_train, X_test, y_test)
    print(results)

    # Q3.4
    plot_results(results)
```

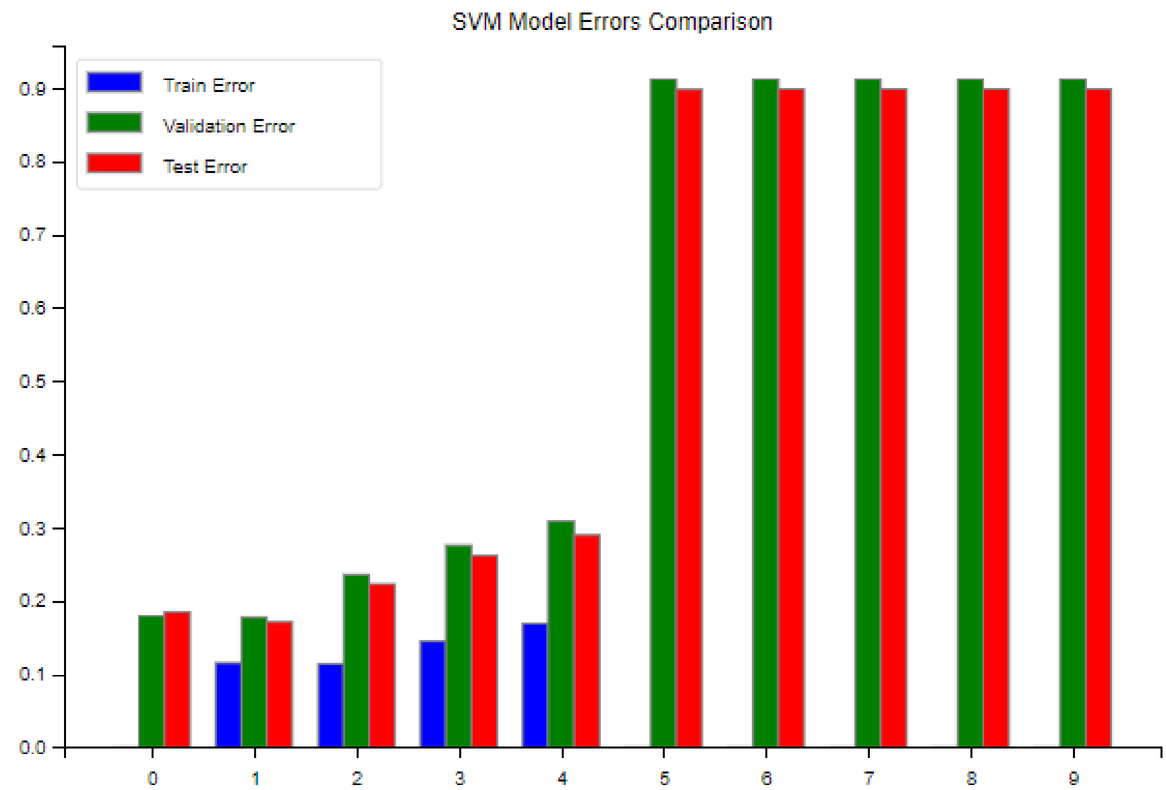
Q3.1

```
(7000, 784), (7000,)
```

Q3.2



Q3.3



We can see that the best modle is the poynomial modle with param d = 2 (1).

For the test error, we can aobserve that the linear modle and the RBF modles have a test error of 0. If we look at the modles that have a test error, the best will still be the polynomial one with d = 2 (1)