שאלה 1:

אם $S^{(i)}$ - סט האימון בו אנו החלפנו את הדוגמא ה-i (x_i, y_i) בדוגמא (x', y').

$A(S)$ - הקונפיגורציה האופטימלית שמ"מ שממזערת את פו' ה-loss - $f_S(w)$.

הס

$$f_S(v) = L_S(v) + \lambda\|v\|^2 =$$

$$= \frac{1}{m}\sum_{j=1}^{m} \ell(v, x_j, y_j) + \lambda\|v\|^2 =$$

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

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

$$= L_{S^{(i)}}(v) + \lambda\|v\|^2 + \frac{1}{m}\ell(v, x_i, y_i) - \frac{1}{m}\ell(v, x', y')$$

באותה דרך עבור $f_S(u)$ ונקבל:

$$f_S(u) = L_{S^{(i)}}(u) + \lambda\|u\|^2 + \frac{1}{m}\ell(u, x_i, y_i) - \frac{1}{m}\ell(u, x', y')$$

$$\Rightarrow f_S(v) - f_S(u) = L_{S^{(i)}}v + \lambda\|v\|^2 - (L_{S^{(i)}}(u) + \lambda\|u\|^2)$$

$$+ \frac{1}{m}(\ell(v, x_i, y_i) - \ell(u, x_i, y_i)) + \frac{1}{m}(\ell(u, x', y') - \ell(v, x', y'))$$

אם נשתמש בסעיף ב' ונשים: $u = A(S)$, $v = A(S^{(i)})$. נקבל:

מכיוון ש- $A(S^{(i)})$ היא ההיפותיזה האופטימלית

מתקיים: $f_S(A(S^{(i)})) \le f_S(A(S))$ מעבר ההשמה ומכאן

$$f_S(A(S^{(i)})) = L_{S^{(i)}}(A(S^{(i)})) + \lambda\|A(S^{(i)})\|^2 \le$$

$$L_S(A(S)) + \lambda\|A(S)\|^2 = f_S(A(S))$$

$$\Rightarrow L_{S^{(i)}}(A(S^{(i)})) + \lambda\|A(S^{(i)})\|^2 - (L_S(A(S)) + \lambda\|A(S)\|^2) \le 0$$

$$\Rightarrow f_S(A(S^{(i)})) - f_S(A(S)) \underset{\text{סעיף ב'}}{\le}$$

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

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

$$\underset{\text{הפחתת מספר שלילי}}{\le} \frac{\ell(A(S^{(i)}),x_i,y_i) - \ell(A(S),x_i,y_i)}{m} + \frac{\ell(A(S),x',y') - \ell(A(S^{(i)}),x',y')}{m} \quad \blacksquare$$

ד. ראינו בהרצאה שמתקיים $f_S(w)$ הינו λ-קמורה מדד לכן:

$$\lambda\|A(S^{(i)}) - A(S)\|^2 \underset{\text{טענה 3}}{\le} f_S(A(S^{(i)})) - f_S(A(S)) \le$$

$$\underset{\text{סעיף ג'}}{\le} \frac{\ell(A(S^{(i)}),x_i,y_i) - \ell(A(S),x_i,y_i)}{m} + \frac{\ell(A(S),x',y') - \ell(A(S^{(i)}),x',y')}{m} \quad \blacksquare$$

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

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

$$\underset{\text{הזזת ρ-ליפשיצ'ות}}{\le} \frac{\rho\|A(S^{(i)}) - A(S)\|}{m} + \frac{\rho\|A(S^{(i)}) - A(S)\|}{m} =$$

$$= \frac{2\rho\|A(S^{(i)}) - A(S)\|}{m} \quad \Big/ \cdot \frac{1}{\lambda\|A(S^{(i)}) - A(S)\|^2}$$

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

ו. $\ell(A(S^{(i)}),x_i,y_i) - \ell(A(S),x_i,y_i) \le$

$$\underset{\text{הזזת ρ-ליפשיצ'ות}}{\le} \rho\|A(S^{(i)}) - A(S)\| \underset{\text{סעיף ה'}}{\le} \rho \cdot \frac{2\rho}{\lambda m} = \frac{2\rho^2}{\lambda m} \quad \blacksquare$$

ז. $L_D(w) = \mathbb{E}_{(x,y)\sim D}[\ell(w,x,y)]$ - התוחלת הבדידה של ה- loss, בעבור השגיאה הבדידה של w (Expected Risk) מבין כל הערכים עם התפלגות D.

לא ניתן לחשב באופן מדויק את ערך זה בהינתן תת מדגם S שכן לא קיימים מספיק נתונים על התפלגות D.

$L_S(w) = \frac{1}{m}\sum_{i=1}^{m}[\ell(w, x_i, y_i)]$ הוא ה- loss הממוצע, דומה

השגיאה האמפירית (Empirical risk) מהיטב את מבחן S

מהתפלגות D. ניתן לחישוב באמצעות S ו-w.

לא רשמנו מהרצאה את המשפט

$$\mathbb{E}_{S\sim D^m}[L_D(A(S)) - L_S(A(S))] \overset{\downarrow}{=}$$

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

$$\overset{\text{סעיף ו'}}{\leq} \mathbb{E}_{(S,x',y')\sim D^{m+1},\, i\sim U(m)}\left(\frac{2\rho^2}{\lambda m}\right) = \frac{2\rho^2}{\lambda m} \quad \blacksquare$$

שאלה 2:

א.

$$TPR = Recall = \frac{TP}{TP+FN}$$

אחוז החיובים שזיהינו מתוך כלל החיובים (האמיתיים) באוכלוסיה

$$Precision = \frac{TP}{TP+FP}$$

אחוז החיובים שזיהינו מתוך החיובים שחזינו.

ב. אילוץ מחלה מדבקת - נרצה לצמצם כמה שיותר חולים במערכה מדבקת מכלל החולים באוכלוסיה עלול לגרום הדבקה,

ולכן נרצה TPR גבוה.

ג. אילוץ מחלה קשה כגון סרטן - נרצה דיוק גבוה עלול לגרום נבדקים שיקבלו "טיפול" שעלול להיות להם שאינו לטובתם.

ולכן נרצה Precision גבוה.

ד.

$$P_w(y=1|w) = P(\sigma(-0.3-0.5x_1+0.5x_2))$$

נזכר כי:

$$\sigma(w^T x_i) = \frac{e^{w^T x_i}}{1+e^{w^T x_i}}$$

נחשב:

$$P_w(y_1=1|x_1) = \frac{e^{-0.3-0.5\cdot 8+0.5\cdot 2}}{1+e^{-0.3-0.5\cdot 8+0.5\cdot 2}} = 0.035$$

$$P_w(y_1=1|x_2) = \frac{e^{-0.3-0.5\cdot 2+0.5\cdot 4}}{1+e^{-0.3-0.5\cdot 2+0.5\cdot 4}} = 0.668$$

$$P_w(y_1=1|x_3) = \frac{e^{-0.3-0.5\cdot 16+0.5\cdot 2}}{1+e^{-0.3-0.5\cdot 16+0.5\cdot 2}} = 0.0007$$

$$P_w(y_1=1|x_4) = \frac{e^{-0.3-0.5\cdot 4+0.5\cdot 2}}{1+e^{-0.3-0.5\cdot 4+0.5\cdot 2}} = 0.214$$

$$P_w(y_1=1|x_5) = \frac{e^{-0.3-0.5\cdot 9+0.5\cdot 2}}{1+e^{-0.3-0.5\cdot 9+0.5\cdot 2}} = 0.022$$

ה. נשתמש בתוצאות הסעיף הקודם בתור ספים לסיווג 'נ':

לפי: >0: $TPR = \frac{TP}{TP+FN} = \frac{2}{2+0} = 1$ $FPR = \frac{FP}{FP+TN} = \frac{3}{3+0} = 1$

>0.007: $TPR = \frac{TP}{TP+FN} = \frac{1}{1+1} = \frac{1}{2}$ $FPR = \frac{FP}{FP+TN} = \frac{3}{3+0} = 1$

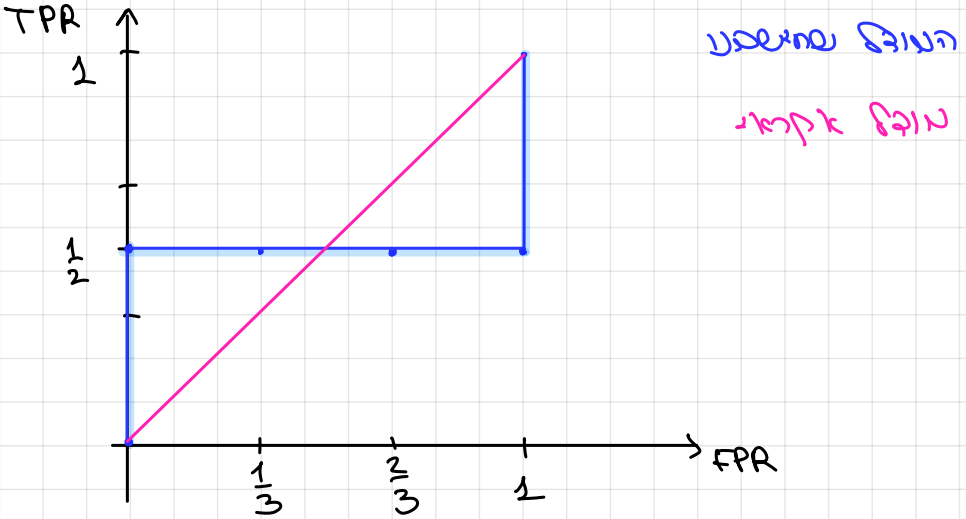
>0.022: $TPR = \frac{TP}{TP+FN} = \frac{1}{1+1} = \frac{1}{2}$ $FPR = \frac{FP}{FP+TN} = \frac{2}{2+1} = \frac{2}{3}$

>0.035: $TPR = \frac{TP}{TP+FN} = \frac{1}{1+1} = \frac{1}{2}$ $FPR = \frac{FP}{FP+TN} = \frac{1}{1+2} = \frac{1}{3}$

>0.214: $TPR = \frac{TP}{TP+FN} = \frac{1}{1+1} = \frac{1}{2}$ $FPR = \frac{FP}{FP+TN} = \frac{0}{0+3} = 0$

>0.668: $TPR = \frac{TP}{TP+FN} = \frac{0}{0+2} = 0$ $FPR = \frac{FP}{FP+TN} = \frac{0}{0+0} = 0$

כעת נשרטט את עקומת ה-ROC.



ו. ה- AUC-ROC הוא השטח מתחת לגרף.

לפי המודל שלנו מדובר ב: $S_{\text{חישבנו}} = \frac{1}{2} \cdot 1 = \frac{1}{2}$

$S_{\text{אקראי}} = \frac{1 \cdot 1}{2} = \frac{1}{2}$

משום שהשטחים שמתחת לגרפים זהים, לא ניתן לקבוע שהמודל שחישבנו טוב יותר מהמודל האקראי.

:3 שלום

Question 3.1

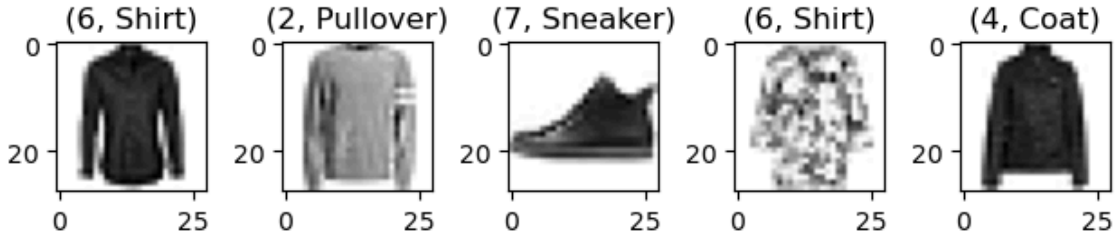
In [44]:
```
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_openml
import matplotlib.pyplot as plt
from sklearn.svm import SVC as svc
```

In [45]:
```
def fetch_mnist():
    #Download MNIST dataset
    X, y = fetch_openml('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), 7000, replace=False)
    X, y = X[indices], y[indices]
    return X, y

X, y = fetch_mnist()
print(X.shape, y.shape)
```

```
C:\Users\emili\anaconda3\envs\data_analysis\lib\site-packages\sklearn\data
sets\_openml.py:932: FutureWarning: The default value of `parser` will cha
nge from `'liac-arff'` to `'auto'` in 1.4. You can set `parser='auto'` to
silence this warning. Therefore, an `ImportError` will be raised from 1.4
if the dataset is dense and pandas is not installed. Note that the pandas
parser may return different data types. See the Notes Section in fetch_ope
nml's API doc for details.
  warn(

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

Question 3.2

In [46]:
```
labels={'0': ' T-shirt/ top', '1': ' Trouser',
        '2': ' Pullover', '3': ' Dress', '4': ' Coat',
        '5': ' Sandal', '6': ' Shirt', '7': ' Sneaker', '8': ' Bag',
        '9': ' Ankle'}

X_im = np.apply_along_axis(lambda row: np.reshape(row, (28, 28)),
                           axis=1, arr=X)

# Create a figure with a 1x10 grid of subplots
fig, axs = plt.subplots(2, 5)

# Plot data on the first subplot
for j in range(2):
    for i in range(5):
        axs[j,i].imshow(X_im[i],cmap="binary")
        axs[j,i].set_title(f'({y[i]},{labels[y[i]]})')

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



Question 3.3

In [47]:
```
def cross_validation_error(X, y, model, n_folds):
    indices = np.arange(len(X))
    np.random.shuffle(indices)
    folds = np.array_split(indices, n_folds)

    train_errors = []
    validation_errors = []

    for fold in range(n_folds):
        val_indices = folds[fold]
        train_indices =
        np.concatenate([folds[i] for i in range(n_folds) if i != fold])

        X_train, y_train = X[train_indices], y[train_indices]
        X_val, y_val = X[val_indices], y[val_indices]

        model.fit(X_train, y_train)
        train_preds = model.predict(X_train)
        val_preds = model.predict(X_val)

        train_errors.append(np.mean(train_preds != y_train))
        validation_errors.append(np.mean(val_preds != y_val))

    avg_train_error = np.mean(train_errors)
    avg_val_error = np.mean(validation_errors)

    return avg_train_error, avg_val_error
```

In [52]:
```
def SVM_results(X_train, y_train, X_test, y_test):
    folds = 4
    results = {}

    # Polynomial and RBF parameter values
    pol_vals = [2, 4, 6, 8]
    RBF_vals = [0.001, 0.01, 0.1, 1.0, 10]

    # Linear SVM
    model = svc(kernel='linear')
    errors = cross_validation_error(X_train, y_train, model, folds)
    model.fit(X_train, y_train)
    test_predict = model.predict(X_test)
    test_error = np.sum(test_predict != y_test) / len(test_predict)
    results[f'SVM linear'] = (errors[0], errors[1], test_error)

    # Polynomial SVMs
    for d in pol_vals:
        model = svc(kernel='poly', degree=d)
        errors = cross_validation_error(X_train, y_train, model, folds)
        model.fit(X_train, y_train)
        test_predict = model.predict(X_test)
        test_error = np.sum(test_predict != y_test) / len(test_predict)
        results[f'SVM poly {d}'] = (errors[0], errors[1], test_error)

    # RBF SVMs
    for gamma in RBF_vals:
        model = svc(kernel='rbf', gamma=gamma)
        errors = cross_validation_error(X_train, y_train, model, folds)
        model.fit(X_train, y_train)
        test_predict = model.predict(X_test)
        test_error = np.sum(test_predict != y_test) / len(test_predict)
        results[f'SVM RBF {gamma}'] = (errors[0], errors[1], test_error)

    return results
```

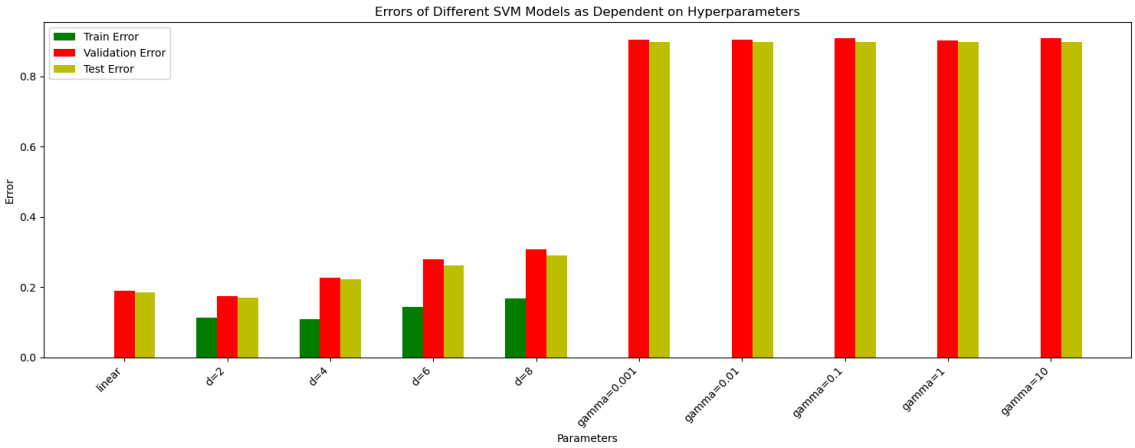
Question 3.4

In [53]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
                    train_test_split(X, y, test_size=0.25, random_state=42)
```

In [54]:
```
results = SVM_results(X_train, y_train, X_test, y_test)
```

In [55]:
```
x = np.arange(10)
labels = [
    'linear', 'd=2', 'd=4', 'd=6', 'd=8',
    'gamma=0.001', 'gamma=0.01', 'gamma=0.1', 'gamma=1', 'gamma=10'
]

train_errors = []
val_errors = []
test_errors = []

# Polynomial and RBF parameter values
pol_vals = [2, 4, 6, 8]
rbf_vals = [0.001, 0.01, 0.1, 1.0, 10]

# Append errors for linear SVM
train_errors.append(results['SVM linear'][0])
val_errors.append(results['SVM linear'][1])
test_errors.append(results['SVM linear'][2])

# Append errors for polynomial SVMs
for d in pol_vals:
    train_errors.append(results[f'SVM poly {d}'][0])
    val_errors.append(results[f'SVM poly {d}'][1])
    test_errors.append(results[f'SVM poly {d}'][2])

# Append errors for RBF SVMs
for gamma in rbf_vals:
    train_errors.append(results[f'SVM RBF {gamma}'][0])
    val_errors.append(results[f'SVM RBF {gamma}'][1])
    test_errors.append(results[f'SVM RBF {gamma}'][2])

# Plotting the errors
plt.figure(figsize=(15, 6))

bar_width = 0.2

# Plotting train errors
plt.bar(x, train_errors, width=bar_width, label='Train Error', color='g')

# Plotting validation errors
plt.bar(x + bar_width, val_errors, width=bar_width,
        label='Validation Error', color='r')

# Plotting test errors
plt.bar(x + 2 * bar_width, test_errors, width=bar_width,
        label='Test Error', color='y')

# Setting labels and titles
plt.xlabel('Parameters')
plt.ylabel('Error')
plt.title('Errors of Different SVM Models as Dependent on Hyperparameters')

# Setting x-ticks with appropriate labels
plt.xticks(x + bar_width, labels, rotation=45, ha='right')

# Displaying the legend
plt.legend()

# Show plot
plt.tight_layout()
```

```
plt.show()
```



The best model for both CV and Test set is the polynomial kernel with d=2.