שאלה 1

א) $S^{(i)}$ הוא מדגם שמתקבל מ-S כאשר מחליפים את התצפית ה-i-ית של S ב-(x',y').

$A(S)$ הוא פלט של A (אלגוריתם למידה) על מדגם אימון S, במקרה זה הפלט הוא פתרון אופטימלי לבעיית RLM, כאשר $R=\lambda\|w\|^2$.

ב) ישירות מסעיף א' ועפ"י הגדרת $S^{(i)}$ ו-$L_S(w)$ נקבל עבור כל i, w בתחום המוגדר:

$$L_S(w)+\frac{1}{m}\left(\ell(w,x',y')-\ell(w,x_i,y_i)\right)=\frac{1}{m}\sum_{j=1}^{m}\ell(w,x_j,y_j)+\frac{1}{m}\left(\ell(w,x',y')-\ell(w,x_i,y_i)\right)$$

$$=\frac{1}{m}\left(\sum_{j=1}^{i-1}\ell(w,x_j,y_j)+\ell(w,x',y')+\sum_{j=i+1}^{m}\ell(w,x_j,y_j)\right)=L_{S^{(i)}}(w)\Rightarrow$$

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

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

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

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

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

ג) מתקיים:

$$f_S(A(S^{(i)}))-f_S(A(S))\overset{ב}{=}\underbrace{\qquad}_{(*)}$$

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

$$+\frac{1}{m}\left(\ell(A(S^{(i)}),x_i,y_i)-\ell(A(S),x_i,y_i)\right)+\frac{1}{m}\left(\ell(A(S),x',y')-\ell(A(S^{(i)}),x',y')\right)$$

לכן ניתן להראות ש-$(*)\le 0$. כיוון ש-$A(S^{(i)})=\arg\min_w f_{S^{(i)}}(w)$, נקבל כי:

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

$$\Rightarrow (*)\le 0\ \blacksquare$$

ד) ראינו בהרצאה למה על קמירות חזקה ו-RLM, ממנה נובע כי: $f_S(w)\Leftarrow 1,2$ - 2λ קמורה חזק.

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

מכאן ומסעיף ב' נובע הנדרש $\blacksquare$

ה) ρ-Lipschitz $\ell(\cdot)$ $\Leftarrow$

$$\begin{cases} \ell(A(S^{(i)}),x_i,y_i)-\ell(A(S),x_i,y_i) \le \rho\|A(S^{(i)})-A(S)\| \\ \ell(A(S^{(i)}),x',y')-\ell(A(S),x',y') \le \rho\|A(S^{(i)})-A(S)\| \end{cases}$$

בהנחה ש- $0<\|A(S^{(i)})-A(S)\|$

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

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

ו) מסעיף ה' נובע:

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

ז) $L_S(w)$ - Empirical Risk ממוצע של שגיאות (loss) של המודל w - ניתן לחשבו עבור S נתון.

$L_D(w)$ - Expected Risk - תוחלת השגיאה של המודל w. ניתן לחשבה בהינתן D, אך בדרך כלל התפלגות D אינה ידועה, לכן בדרך כלל לא ניתן לחשבה.

ח) (הרצ' 6) Theorem 1

$$E_{S\sim D^m}[L_D(A(S))-L_S(A(S))] =$$

$$= E_{(S,(x',y'))\sim D^{m+1},\, i\sim U(m)}[\ell(A(S^{(i)}),x_i,y_i)-\ell(A(S),x_i,y_i)] \overset{ו'}{\le}$$

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

שאלה 2

א) TPR (recall): $\frac{TP}{TP+FN}$ - יחס בין דוגמאות שסווגו נכון כ- positive לבין כל הדוגמאות ה- positive.
נרצה למקסם אותה (את המטריקה), כלומר שכמות הסיווגים הלא נכונים כ- negative תהיה קטנה ככל האפשר.

precision: $\frac{TP}{TP+FP}$ - יחס בין דוגמאות שסווגו נכון כ- positive לבין כל הדוגמאות שסווגו כ- positive.
נרצה למקסם את המטריקה, כלומר שכמות הסיווגים הלא נכונים כ- positive תהיה קטנה ככל האפשר.

ב) recall יהיה חשוב יותר מה-precision. למשל כשחשוב לסווג בין אדם חולה כחולה (למשל אם המחלה מדבקת מאוד) וכמות קטנים אם הסיווג של בן אדם בריא יהיה שגוי.

ג) precision יהיה יותר חשוב מה-recall. לדוגמא, צריך לסווג תאים סרטניים במוח כדי להעביר ניתוח מדויק ולהסיר רק את התאים הסרטניים, וסיווג לא מדויק יכול לסכן את האזורים הסביבים של המוח (בעיקר, כשניתוחי מוח יהיו אוטומטיים...)

ד) המודל רגרסיה לוגיסטית:
$$P_w(y=1\mid x)=\frac{e^{w^Tx+b}}{1+e^{w^Tx+b}}=\frac{e^{-0.3-0.5x_1+0.5x_2}}{1+e^{-0.3-0.5x_1+0.5x_2}}$$

$$P_w(y_1=1\mid x_1=(8,2))=\frac{e^{-0.3-0.5\cdot8+0.5\cdot2}}{1+e^{-0.3-0.5\cdot8+0.5\cdot2}}\cong 0.03557$$

$$P_w(y_2=1\mid x_2=(2,4))=\frac{e^{-0.3-0.5\cdot2+0.5\cdot4}}{1+e^{-0.3-0.5\cdot2+0.5\cdot4}}\cong 0.66819$$

$$P_w(y_3=1\mid x_3=(16,2))=\frac{e^{-0.3-0.5\cdot16+0.5\cdot2}}{1+e^{-0.3-0.5\cdot16+0.5\cdot2}}\cong 0.00067$$

$$P_w(y_4=1\mid x_4=(4,2))=\frac{e^{-0.3-0.5\cdot4+0.5\cdot2}}{1+e^{-0.3-0.5\cdot4+0.5\cdot2}}\cong 0.21416$$

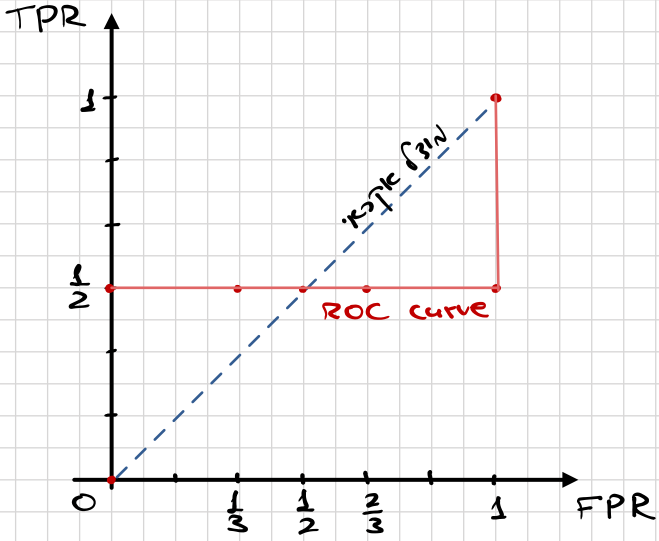
$$P_w(y_5=1\mid x_5=(9,2))=\frac{e^{-0.3-0.5\cdot9+0.5\cdot2}}{1+e^{-0.3-0.5\cdot9+0.5\cdot2}}\cong 0.02188$$

ה)

כמות מעל הסף בכל אינטרוול

τ	0 – 0.00067	0.00067 – 0.02188	0.02188 – 0.03557	0.03557 – 0.21416	0.21416 – 0.66819	0.66819 – 1
Pred_TP	2	1	1	1	1	0
Pred_FP	3	3	2	1	0	0
TPR	$2/2=1$	$1/2$	$1/2$	$1/2$	$1/2$	$0/2=0$
FPR	$3/3=1$	$3/3=1$	$2/3$	$1/3$	$0/3=0$	$0/3=0$

מס' השליליים הכולל: 3, מס' החיוביים הכולל: 2.



1) קל לראות לפי הגרף ש- $AUC\text{-}ROC = \frac{1}{2}$.
כמו כן, גם השטח מתחת למודל האקראי הוא $\frac{1}{2}$.
לכן לא נראה שהמודל יותר טוב (או פחות טוב) מהמודל האקראי.

Question 3

In [28]:
```python
# 3.a

import numpy as np
from sklearn.datasets import fetch_openml
```

In [3]:
```python
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
```

In [4]:
```python
X, y = fetch_mnist()
print(X.shape, y.shape)
```

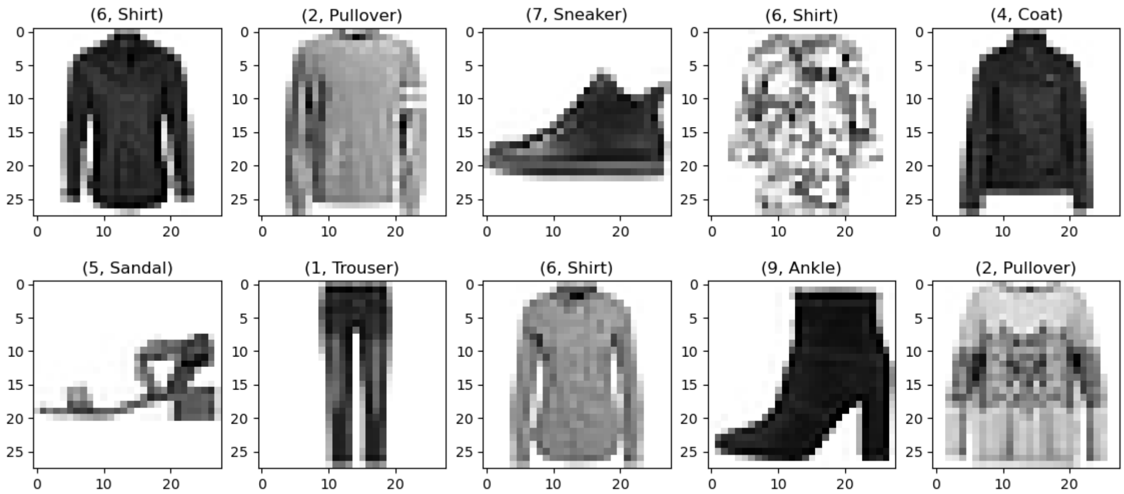
```
(7000, 784) (7000,)
```

In [27]:
```python
# 3.b

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'}
```

In [36]:
```python
head = 10
fig, ax = plt.subplots(2, head//2, figsize=(14, 6))
for i in range(2):
    for j in range(head//2):
        ax[i][j].imshow(X[(head//2)*i + j].reshape(28, 28), cmap='binary')
        ax[i][j].set_title(f"({y[(head//2)*i + j]}, {idx2class[y[(head//2)*i + j]]})")
```



In [38]:
```python
# 3.c

def cross_vlidation_error(X, y, model, folds):

    fold_attrs, fold_labels = np.array_split(X, folds), np.array_split(y, folds)
    val_res, train_res = [], []

    for outer in range(folds):
        x_train, y_train = [], []
        for inner in range(folds):
            if inner != outer:
                x_train.extend(fold_attrs[inner])
                y_train.extend(fold_labels[inner])
        x_train = np.array(x_train)
        y_train = np.array(y_train)
        model.fit(x_train, y_train)

        train_res.append(1-(model.score(x_train, y_train)))

        attrs = fold_attrs[outer]
        labels = fold_labels[outer]
        val_res.append(1-(model.score(attrs, labels)))

        average_train_error = np.array(train_res).mean()
        average_val_error = np.array(val_res).mean()
    return (average_train_error, average_val_error)
```

In [53]:
```python
from sklearn.svm import SVC
def SVM_results(X_train, y_train, X_test, y_test):
    poly_d = [2, 4, 6, 8]
    RBF_gammas = [0.001, 0.01, 0.1, 1.0, 10.0]
    kernel_configs = [{'name': 'SVM_linear', 'kernel': 'linear', 'C': 1, 'degree': 3, 'gamma': 'scale'}] + \
            [{'name': f"SVM_poly_{d}", 'kernel': 'poly', 'C': 1, 'degree': d, 'gamma': 'scale'} for d in poly_d] + \
            [{'name': f"SVM_rbf_{gamma}", 'kernel': 'rbf', 'C': 1, 'degree': 3, 'gamma': gamma} for gamma in RBF_gammas]

    svm_results = {}
    for i, config in enumerate(kernel_configs):
        print(f"\rModel: {config['name']} ({i+1} of {len(kernel_configs)})", end='')
        model = SVC(kernel=config['kernel'], C=config['C'], degree=config['degree'], gamma=config['gamma'])
        average_train_error, average_validation_error = cross_vlidation_error(X_train, y_train, model, folds=4)
        average_test_error = (model.predict(X_test) != y_test).mean()
        svm_results[config['name']] = (average_train_error, average_validation_error, average_test_error)
    print('\rFinished.')
    return svm_results
```

In [43]:
```python
# 3.d

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]:
```python
svm_results = SVM_results(X_train, y_train, X_test, y_test)
```

```
Finished.M_rbf_10.0 (10 of 10)
```

In [78]:
```
import pandas as pd
df = pd.DataFrame.from_dict(svm_results).transpose().rename(
    columns={0: 'Average Train Error', 1: 'Average Validation Error', 2: 'Average Test Error'})
df.index.name = 'model'
df.reset_index(inplace=True)
df = pd.melt(df, id_vars="model",
             value_vars=['Average Train Error', 'Average Validation Error', 'Average Test Error'], var_name="error")
df.head(2)
```

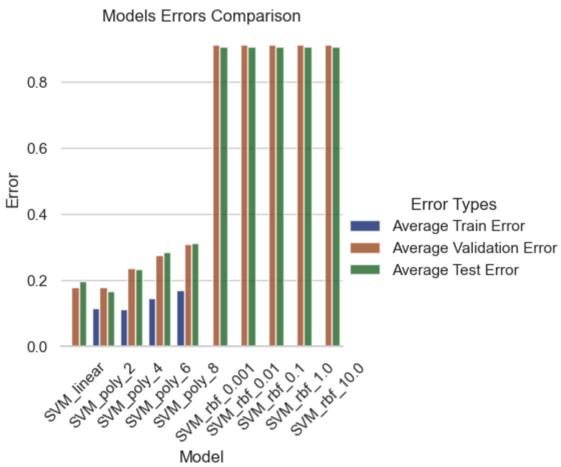
Out[78]:

	model	error	value
0	SVM_linear	Average Train Error	0.000000
1	SVM_poly_2	Average Train Error	0.114921

In [91]:
```
import seaborn as sns

sns.set_theme(style="whitegrid")

g = sns.catplot(data=df, kind="bar", x="model", y="value", hue="error",
                errorbar="sd", palette="dark", alpha=.8, height=4).set(title="Models Errors Comparison")
plt.xticks(rotation=45)
g.despine(left=True)
g.set_axis_labels("Model", "Error")
g.legend.set_title("Error Types")
```



In [92]:
```
print(f"SVM_linear: {svm_results['SVM_linear']}")
print(f"SVM_poly_2: {svm_results['SVM_poly_2']}")
```

```
SVM_linear: (0.0, 0.1780943206304683, 0.1977142857142857)
SVM_poly_2: (0.11492092702316437, 0.17752528653428193, 0.16685714285714287)
```

According to CV method, the best model (with the lowest average validation error = 0.1775) is SVM polynomial with degree of 2. This is also the best model on the test set (with average test error = 0.1669). It is worth to mention that the linear model performance is almost the same.