Question 1

Section A

$S^{(i)}$ is the training set with the (i)th point replaced by point (x', y').
$A(S)$ is the optimal vector w that minimizes the regularized loss function $f_s(w)$. In other words, $A(S)$ is a weight vector that minimizes the empirical loss on the training data while also applying regularization, which encourages larger margins, improves generalization to unseen data, and helps prevent overfitting.

Section B

The equality is correct. We replaced the regularized loss function $f_s(w)$ with $f_{S^{(i)}}(w)$ for each weight vector u, v. We know that $f_s(w) = L_s(w) + \lambda \| w \|$, while $L_S(w) = \frac{1}{m} \sum_{i=1}^{m} l(w, x_i, y_i)$. So, to maintain equality we need to remove the empirical loss (divided by m) of (x', y') and add the empirical loss of (x_i, y_i). In math, we would get the following:

$$f_S(v) = L_S(v) + \lambda \| v \|^2 = \frac{1}{m} \sum_{j=1}^{m} l(v, x_j, y_j) + \lambda \| v \|^2 = \frac{1}{m} \left(\sum_{j=1}^{m} l(v, x_j, y_j) + l(v, x', y') - l(v, x', y') \right) + \lambda \| v \|^2$$

$$f_S(v) = \frac{1}{m} \left(\sum_{j=1, j \neq i}^{m} l(v, x_j, y_j) + l(v, x', y') + l(v, x_i, y_i) - l(v, x', y') \right) + \lambda \| v \|^2$$

$$f_S(v) = \frac{1}{m} \left(\sum_{j=1, j \neq i}^{m} l(v, x_j, y_j) + l(v, x', y') \right) + \frac{1}{m} \left(l(v, x_i, y_i) - l(v, x', y') \right) + \lambda \| v \|^2$$

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

The opening for $f_S(u)$ is the same, but with u instead of v. Combining both of the equations, we get the required equality.

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

Section C

First, $A(S^{(i)})$ is the optimal solution to the loss function (minimum), and therefore $A(S)$ will produce a loss bigger than or equal to it.
Therefore:

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

We take $v = A\left(S^{(i)}\right)$ and $u = A(S)$. We get:

$$f_S\left(A\left(S^{(i)}\right)\right)-f_S\left(A(S)\right)\overset{\text{Section B}}{=} ¿$$

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

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

Then from (*) we get:

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

As required.

Section D

According to the lemma we've seen in class, $f_S(w)$ is 2λ-strongly convex and $A\left(S^{(i)}\right)$ minimizes f, then for $A(S)$ it holds that:

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

Now, from Section C it holds that:

$$\lambda\left\|A\left(S^{(i)}\right)-A(S)\right\|^2\le\frac{l\left(A\left(S^{(i)}\right),x_i,y_i\right)-l\left(A(S),x_i,y_i\right)}{m}+\frac{l\left(A(S),x',y'\right)-l\left(A\left(S^{(i)}\right),x',y'\right)}{m}$$

As required.

Section E

First, If $\left\|A\left(S^{(i)}\right)-A(S)\right\|=0$ the inequality is trivial.

Otherwise,

We know that $l(\cdot)$ is ρ-Lipschitz. Then, from Section D it holds that:

$$\lambda\left\|A\left(S^{(i)}\right)-A(S)\right\|^2\le\frac{l\left(A\left(S^{(i)}\right),x_i,y_i\right)-l\left(A(S),x_i,y_i\right)}{m}+\frac{l\left(A\left(S^{(i)}\right),x',y'\right)-l\left(A(S),x',y'\right)}{m}$$

$$\overset{\rho\text{-Lipschitz's definition}}{\le}\frac{\rho\left\|A\left(S^{(i)}\right)-A(S)\right\|}{m}+\frac{\rho\left\|A\left(S^{(i)}\right)-A(S)\right\|}{m}=\frac{2\rho\left\|A\left(S^{(i)}\right)-A(S)\right\|}{m}$$

Therefore, we get:

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

Section F

If $l(\cdot)$ is ρ-Lipschitz, Then:

$$l\left(A\left(S^{(i)}\right),x_i,y_i\right)-l\left(A(S),x_i,y_i\right)\le\rho\ \|\ A\left(S^{(i)}\right)-A(S)\ \|$$

From Section E we get that:

$$l\left(A\left(S^{(i)}\right),x_i,y_i\right)-l\left(A(S),x_i,y_i\right)\le\frac{2\rho^2}{\lambda m}$$

Section G

$L_D(w)$ is the expected risk (expected value of loss) of w over all the data points with distribution D. It is not possible to calculate this value given only a subset S and vector w, Because we need to know also the original distribution D for the expectation.

$L_S(w)$ is the empirical risk (which is the average loss) over a subset of points S from distribution D. This value can be calculated, as we know both S and w.

Section H

First of all weve seen in class that:

$$E_{S\sim D^m}\left[L_D(A(S))-L_S(A(S))\right)=E_{(S,(x_i,y_i))\sim D^{m+1},i\sim U(m)}\left[\ell\left(A\left(S^{(i)}\right),(x_i,y_i)\right)-\ell\left(A(S),(x_i,y_i)\right)\right)$$

From Section G we get that:

$$E_{(S,(x_i,y_i))\sim D^{m+1},i\sim U(m)}\left[\ell\left(A\left(S^{(i)}\right),(x_i,y_i)\right)-\ell\left(A(S),(x_i,y_i)\right)\right)\le E_{(S,(x_i,y_i))\sim D^{m+1},i\sim U(m)}\left[\frac{2\rho^2}{\lambda m}\right)$$

$\frac{2\rho^2}{\lambda m}$ is scalar. Therefore:

$$E_{(S,(x_i,y_i))\sim D^{m+1},i\sim U(m)}\left[\ell\left(A\left(S^{(i)}\right),(x_i,y_i)\right)-\ell\left(A(S),(x_i,y_i)\right)\right)\le\frac{2\rho^2}{\lambda m}$$

We get,

$$E_{S\sim D^m}\left[L_D(A(S))-L_S(A(S))\right)\le\frac{2\rho^2}{\lambda m}$$

As required.

Question 2

Section A

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

We want Precision to be higher. If this is 1, then every person that we identified as sick is actually sick.

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

We want Recall/TPR to be higher. If this is 1, then we managed to correctly identify every sick person as sick.

Section B

An example where recall is more important than precision is identifying rockets sent to Israel during the war. This is because it's better to falsely tell citizens to enter their bomb shelters for a couple of minutes, than it is to take the risk with their lives.

For this reason, it is more important that we correctly label each case where there might be a bomb as cause for an alarm, even at the risk of having more false positives, which is why recall is more important than precision.

Section C

An example where precision is more important than recall is deciding to give someone a death sentence based on a crime.

The reason precision might be more important in this scenario, is due to the severity of the punishment, we want to be as sure as possible that we're only giving it when absolutely necessary, even at the risk of having some people be wrongly sent to prison instead.

For this reason, it is more important that we not have false positives, than it is to have less false negatives, making precision the more critical measurement to maximize.

Section D

We'll use the sigmoid function to classify vectors to labels.

$P_w(y=1 \vee x)=\sigma(w^T x)=\sigma\left(\frac{1}{2}(x_2-x_1)-0.3\right)$.

Placing the vectors $x_i, \forall i=1,2,\ldots,5$ gives:

$P_w(y_1=1 \vee x_1)=\sigma(-3.3)=0.0356$

$P_w(y_2=1 \vee x_2)=\sigma(0.7)=0.6682$

$P_w(y_3=1 \vee x_3)=\sigma(-7.3)=0.0007$

$P_w(y_4=1 \vee x_4)=\sigma(-1.3)=0.2141$

$P_w(y_5=1 \vee x_5)=\sigma(-3.8)=0.0219$

Section E

For each of these values we just calculated, we'll show the ROC if the "passing" value is anything greather than it.

In other words, for $p_i = P_w(y_i = 1 \vee x_i)$ and for each $i \in [5]$, we'll decide if a point gets labeled 1 or 0 based on if its probability is strictly greater than p_i.

We note that the total number of "healthy" is 3, and total number of "sick" is 2.

$i=1$: TP: point 2. FP: point 4. TPR: $\frac{1}{2}$, FPR: $\frac{1}{3}$.

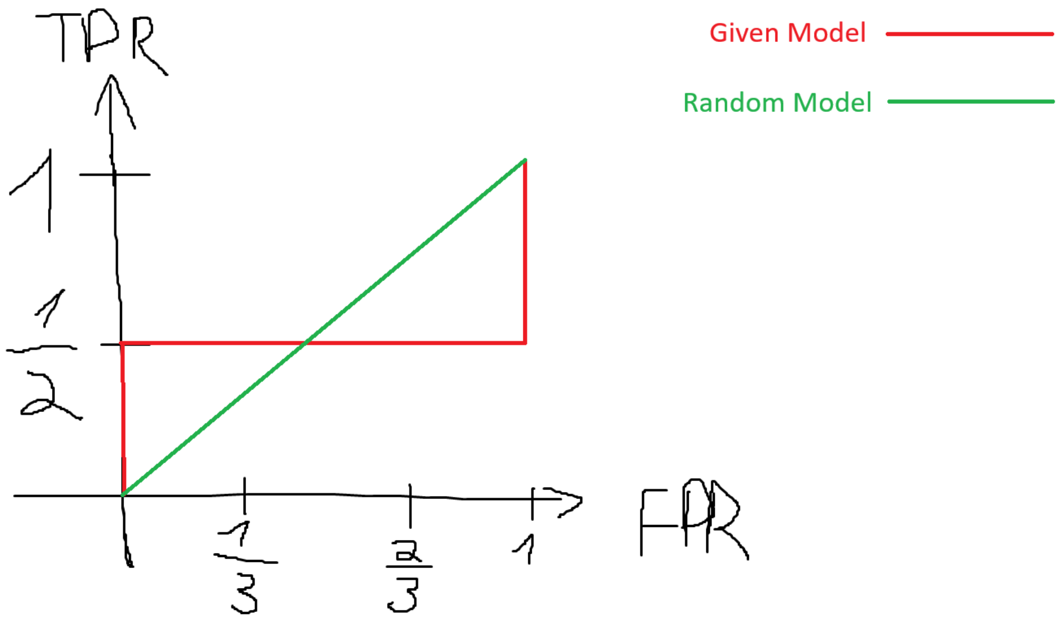
$i=2$: TP: None. FP: 0. TPR: 0, FPR: 0.

$i=3$: TP: points 2+3. FP: points 1,4,5. TPR: $\frac{1}{2}$, FPR: 1.

$i=4$: TP: point 2. FP: None. TPR: $\frac{1}{2}$, FPR: 0.

$i=5$: TP: point 2. FP: points 1,4. TPR: $\frac{1}{2}$, FPR: $\frac{2}{3}$.

6th: Labelling all as 1. In this case, the TPR: 1, and FPR: 1.



Section F

The ROC-AUC of the random model we already know is $\frac{1}{2}$.

Our model's ROC-AUC:

$$\int_0^1 \frac{1}{2} dx = \frac{1}{2}$$

So our model has the same ROC-AUC score as a random classifier, meaning that our model is no better than a totally random classifier.

Question 3

Section A

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

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)
```

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

Section B

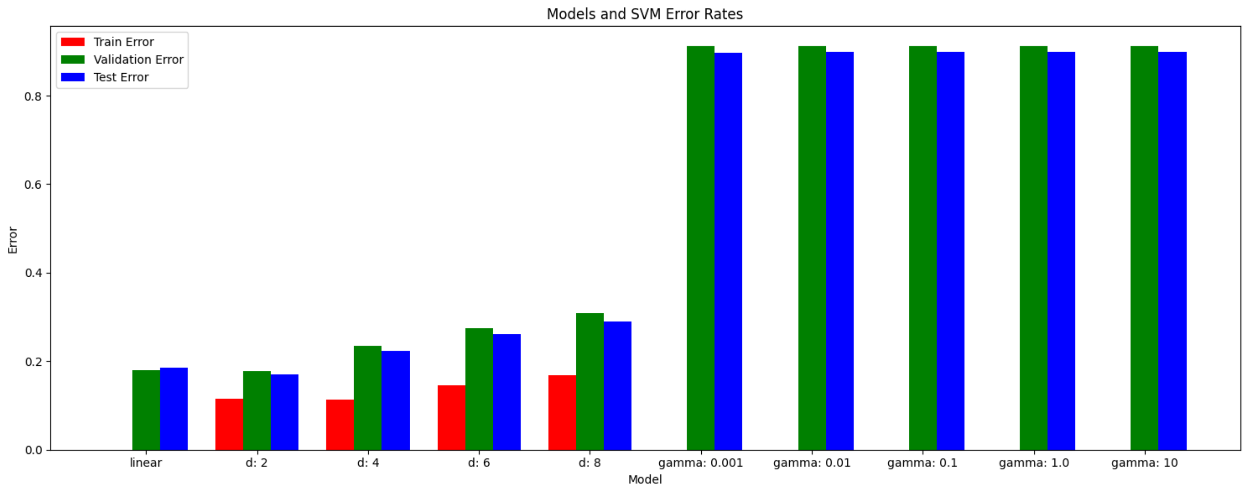
```
import matplotlib.pyplot as plt

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

plt.figure(figsize=(10, 4))
for i, x in enumerate(X[:10, :]):
    plt.subplot(2, 5, i + 1)
    plt.imshow(x.reshape((28, 28)), cmap="binary")
    plt.title(f"Label: ({y[i]}, {idx2class[y[i]]})")
    plt.axis("off")  # Hide axes for cleaner look

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

Section C

```
from sklearn import svm

def cross_validation_error(X, y, model, folds):
  """
  Perform k-fold Cross Validation
  Returns (avg_train_error, avg_validation_error)
  Assumes model has fit and predict functions

  Taken directly from HW3
  """

  train_error_rates = []
  val_error_rates = []

  fold_size = len(X) // folds

  indices = np.arange(len(X))

  for fold in range(folds):
    val_start_idx = fold * fold_size
    val_end_idx = val_start_idx + fold_size if fold < folds - 1 else
len(X)

    val = indices[val_start_idx: val_end_idx]
    train = np.setdiff1d(np.arange(len(X)), val)


    X_val = X[val]
    y_val = y[val]
    X_train = X[train]
    y_train = y[train]

    model.fit(X_train, y_train)
    y_pred = model.predict(X_val)
    val_error_rate = np.mean(y_pred != y_val)
    val_error_rates.append(val_error_rate)

    y_pred = model.predict(X_train)
    train_error_rate = np.mean(y_pred != y_train)
    train_error_rates.append(train_error_rate)

  return np.mean(train_error_rates), np.mean(val_error_rates)

gamma_list = [0.001, 0.01, 0.1, 1.0, 10]
d_list = [2, 4, 6, 8]


def evaluate_model(model, X_train, y_train, X_test, y_test,
results_dict, key):
  train_err, val_err = cross_validation_error(X_train, y_train, model,
folds=4)
  model.fit(X_train, y_train)
  test_err = 1 - model.score(X_test, y_test)
  results_dict[key] = (train_err, val_err, test_err)


def SVM_results(X_train, y_train, X_test, y_test):
  """
    Train SVM classifiers with linear, polynomial, and RBF kernels on
the training set.
    Evaluate their performance using cross-validation error on
training data and test data.

    Returns:
        dict: Keys are model descriptions, values are (train_error,
test_error) tuples.
    """
  results_dict = dict()
  model = svm.SVC(kernel='linear')
  evaluate_model(model, X_train, y_train, X_test, y_test,
results_dict, 'SVM_linear')

  for d in d_list:
    model = svm.SVC(kernel='poly', degree=d)
    evaluate_model(model, X_train, y_train, X_test, y_test,
results_dict, f'SVM_poly_{d}')

  for gamma in gamma_list:
    model = svm.SVC(kernel='rbf', gamma=gamma)
    evaluate_model(model, X_train, y_train, X_test, y_test,
results_dict, f'SVM_rbf_{gamma}')

  return results_dict
```

Section 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)

results = SVM_results(X_train, y_train, X_test, y_test)

hyper_parameters = ['linear']
hyper_parameters.extend(f'd: {d}' for d in d_list)
hyper_parameters.extend(f'gamma: {gamma}' for gamma in gamma_list)

def plot_SVM_results(results):
  width = 0.25
  plt.figure(figsize=(15, 6))

  models = list(results.keys())
  x_axis = np.arange(len(models))

  train_errors = [results[k][0] for k in models]
  val_errors = [results[k][1] for k in models]
  test_errors = [results[k][2] for k in models]

  plt.bar(x_axis-width, train_errors, width=width, label='Train
Error', color='r')

  plt.bar(x_axis, val_errors, width=width, label='Validation Error',
color='g')

  plt.bar(x_axis+width, test_errors, width=width, label='Test Error',
color='b')

  plt.xticks(x_axis, hyper_parameters)
  plt.xlabel('Model')
  plt.ylabel('Error')

  plt.title('Models and SVM Error Rates')
  plt.legend()
  plt.tight_layout()

  plt.show()

plot_SVM_results(results)
```


Models and SVM Error Rates
Train Error
Validation Error
Test Error
Error
0.0
0.2
0.4
0.6
0.8
linear
d: 2
d: 4
d: 6
d: 8
gamma: 0.001
gamma: 0.01
gamma: 0.1
gamma: 1.0
gamma: 10
Model