You may complete the following entirely in a Jupyter notebook. Ensure that the notebook has your name on it. Save the notebook as a PDF and submit the PDF through Canvas and also upload your notebook file to Canvas if your professor is not having you submit it another way, such as via GitHub.

1) In lecture, we discuss evaluating models on a grid of points to estimate and visualize the decision boundaries learned by a model. In this problem, you are going to walk through that process.

(a) Using `np.linspace` and `np.meshgrid` functions from NumPy, create grids of x_1 values and x_2 values in the 2-dimensional space bounded by $0 \leq x_1 \leq 10$ and $0 \leq x_2 \leq 10$. Use 20 points along each dimension (400 points total).

(b) Save the grids of x_1 values and x_2 values returned by `np.meshgrid` in a matrix of observations X. The matrix should have 2 columns (first column for x_1 and second column for x_2) and 400 rows. *Hint: To transform the grids of x_1 values into a column vector, use `np.reshape`. Same thing for x_2. To create the 400×2 matrix, use `np.hstack`.* Plot the observations in X using a scatter plot.

(c) We will learn later how for a linear classification model (like logistic regression), the weighted sum of the features can be mapped to a certainty measure that tells us how likely a given feature vector belongs to class 1 or 0. If the weighted sum is positive: the higher the sum, the more certain we are that the sample belongs to class 1. If the weighted sum is negative: the lower the sum, the more certain we are that the sample belongs to class 0. In both cases, the closer the weighted sum is to 0, the less certain we become. In the next questions, you will visualize the output of the weighted sum of a trained linear classification model.

Assume we trained a linear classification model on a 2-dimensional data and we found that $\mathbf{w} = (5, 3)$ and $w_0 = -10$. For each point in X (for each row), you will need to find the weighted sum, i.e., create a vector $\mathbf{v}$ such as $\mathbf{v} = X\mathbf{w} + w_0$.

(d) To visualize the values of the weighted sum in the feature space, use Seaborn's `heatmap` function to plot the values of the weighted sum. You might not have the seaborn package installed. If needed, after activating your local csc4601 environment, execute conda install seaborn. *Hint: You'll need first to convert $\mathbf{v}$ to a square 20×20 array using `np.reshape` and then pass the matrix to the function heatmap of Seaborn.*

(e) You may have noticed that $\mathbf{v}$ contains continuous values. We can convert the continuous values to categorical by using a threshold. Whenever a value ≤ 0, the label should be 0. When the value is > 0, the label should be 1. Create a vector of predicted (categorical) labels $\hat{\mathbf{y}}$. *Hint: y_hat = v > 0 works here because True will be treated as 1 and False as 0, but more generally we would need to use mask arrays as indexes to set the various class labels.*

(f) Use Matplotlib's `contourf` function to create a contour plot from the binary values $\hat{\mathbf{y}}$. *Hint1: You'll need to ensure the binary vector $\hat{\mathbf{y}}$ is 20×20 square array; use `np.reshape` if needed. Hint2: An example that uses `contourf` can be found here.*

2) In lecture, we discuss geometric objects (e.g., points, vectors, and planes) and how they can be used to model linear decision boundaries. This problem explores some of their properties and how to apply these objects to classification. Stating the parametric equation of a line $\mathbf{x}$ in terms of a normal vector $\mathbf{n}$ and a point on the line $\mathbf{p}$, we have $\mathbf{n} \cdot (\mathbf{x} - \mathbf{p}) = 0$.

(a) Write the equations for each plane with normal vector $\mathbf{n}$ and intersecting the point $\mathbf{r}_0$:

(i) $\mathbf{n} = (1, 1)$, $\mathbf{r}_0 = (5, 7)$

(ii) $\mathbf{n} = (-1, -1)$, $\mathbf{r}_0 = (5, 7)$

(iii) *Optional* $\mathbf{n} = (1, 1, 0)$, $\mathbf{r}_0 = (1, 2, 3)$

(iv) *Optional* $\mathbf{n} = (-1, -1, 0)$, $\mathbf{r}_0 = (1, 2, 3)$

(v) Explain why you cannot find a plane from the following: $\mathbf{n} = (1, 3, 5, 11)$, $\mathbf{r}_0 = (13, 17)$

(vi) Simplify the equations from (i-iv) as much as possible. What do you notice about the pairs (i) and (ii) and (iii) and (iv)?

(vii) *Optional* Plot the normal vectors, points, and planes from (i-iv). Hints: Planes in 2D spaces are lines. For the 3D case, create a 3D projection and use the `plot_surface` function to create the plane.

(b) Planes divide spaces in half. We can use the sign of a dot product operation to determine which side of the plane a point lies. Here is the equation of the line; remember the dot product is 0 only for points that lie on the line.

$$(2, 2) \cdot (x_1 - 5, x_2 - 5) = 0$$

x_1	x_2
2	15
3	17
4	13
4	1
5	3
6	2

(i) Calculate the sign for each of the given points. np.dot can calculate all the dot products at once if you give it the 6x2 matrix of differences as its first argument and the length-2 normal vector as its second argument. Always check the NumPy documentation for shortcuts like these.

(ii) Plot the line, normal vector and points. *You have two options here: you can use Matplotlib OR you can sketch everything on paper.*

(iii) Verify that for all points that lie on the same side of the normal vector, the dot product is positive.

3) **Standardization**
Many machine learning algorithms are sensitive to the centering and scaling of features. Let's walk through what this means. To "center" data, we shift the data so that the mean is 0. We do this by subtracting the mean $\bar{x}$ value from every data point.

$$z = x - \bar{x}$$

Here x represents the value of a given feature that corresponds to a particular sample. The mean $\bar{x}$ is the mean of all values of the given feature across all samples (mean of a given column).
To "scale" data, we want to "squish" the data so that the standard deviation is 1. We do this by dividing every data point by the standard deviation σ.

$$z = \frac{x}{\sigma}$$

We often combine these into a single equation:

$$z = \frac{x - \bar{x}}{\sigma}$$

A few notes on best practices: Centering and scaling is done independently for every column in the feature matrix. When using training and testing sets, we calculate $\bar{x}$ and σ from the training set and use both to center and scale the training and testing sets.

(a) Load the data from the file `peaks_lfc.csv` using Pandas or NumPy. (It only contains one column.)

(b) Plot a histogram of the data. *Hint: use Seaborn's `displot` function.*

(c) Center the data. Plot the histogram. What's different? *Hint: Use Numpy's `mean` function.*

(d) Scale the data. Plot the histogram. What's different? *Hint: Use Numpy's `std` function.*

(e) Both center and scale the data. Plot the histogram. What's different?

4) **Observation distances**
Explore how distance from the decision boundary is related to the output of our linear models.

1. Load in the IRIS dataset and put the first two features into a feature matrix. Plot these observations.
2. Plot the following decision boundary on the figure:

$$x_2 = 0.75x_1 - 0.9$$

   To proceed, choose one option:

Option 1 - Longer for complete grade

3. Find the projections from each observation to the line and plot a line segment for each observation
   (a) We are finding the orthogonal projection of a point (observation) to the line (model)
   (b) Note that we are looking for line segments that are orthogonal to the model
      i. The model is in the form of $x_2 = m_{model}x_1 + b$
      ii. We want to solve for the orthogonal slope $m_{orthogonal}$
      iii. $m_{model}m_{orthogonal} = -1 \rightarrow m_{orthogonal} = -\frac{4}{3}$
      iv. This gives us a "line template" of $x_2 = -\frac{4}{3}x_1 + b$, and now we have to solve for b given our point (observation)
      v. Given the first point in the dataset - $Obs_1 = (5.1, 3.5)$, we can solve $3.5 = -\frac{4}{3}(5.1) + b \rightarrow b = 10.3$
      vi. Now we have two equations and two unknowns
         - $x_2 = \frac{3}{4}x_1 - 0.9$
         - $x_2 = -\frac{4}{3}x_1 + 10.3$
      vii. We can put these in a standard form:
         - $\frac{3}{4}x_2 - x_1 = 0.9$
         - $-\frac{4}{3}x_2 - x_1 = -10.3$
      viii. In the standard form we can use linear algebra to solve for the system of equations:
         - $\begin{bmatrix} \frac{3}{4} & -1 \\ -\frac{4}{3} & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.9 \\ -10.3 \end{bmatrix}$
         - $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{3}{4} & -1 \\ -\frac{4}{3} & -1 \end{bmatrix}^{-1} \begin{bmatrix} 0.9 \\ -10.3 \end{bmatrix}$
         - Where (x_1, x_2) is the projection of our point (observation) onto the line (model)
      ix. You now have two points, solve for the distance between the two points
         - ||a-b||→ 0.46
      x. Add a vector to your figure that connects the point on the model to the observation in space, color it according to the distance.
      xi. Do this for all observations in the dataset.

Option 2 - Shorter for partial grade

Suppose a linear classification model has the following decision boundary:

$$\mathbf{w}^T\mathbf{x} + w_0 = 0$$

We can show that the distance from an observation point $\mathbf{x}_{\text{obs}} = (x_1, x_2)$ to the decision boundary is given by:

$$\text{distance from } \mathbf{x}_{\text{obs}} \text{ to the line} = \frac{|\mathbf{w}^T\mathbf{x}_{\text{obs}} + w_0|}{\|\mathbf{w}\|} \tag{1}$$

- How is the weighted sum of the features of $\mathbf{x}_{\text{obs}}$ is related to its distance from the line?
- Rearrange the terms of the given decision boundary

  $$x_2 = 0.75x_1 - 0.9$$

  so that you obtain the line's equation in standard form. Extract the normal vector $\mathbf{w}$ and w_0. Choose two samples/observations from the given dataset. Compute their distances from the line using the formula in (1). Verify that the closer point to the line has indeed a smaller value for the computed distance.