1. Loss Functions for Linear Regression

Part 1:

a. This function could be a good choice for regression. However, it is more sensitive to any outliers compared to the standard squared loss function. Sub-gradient descent is applicable here since it's an absolute value function.

b. This function would not be a good choice for regression. Since its cubic, it would not be symmetric at 0 which could lead to great error. Gradient descent could be used here.

c. This function would also not be a good choice. This is an exponential function which means it will continue to grow exponentially. Because of this, the optimization of this function would be unstable. We can use gradient descent here as well.

d. This function is also not a good choice. This function is used for hinge loss which deals with classification, not regression. We can use sub-gradient descent here as well since the max function is not differential at 0.

Part 2:

a.

$\partial L_i/\partial w = 3|f(w,b,x_i) - y_i|^2 * \text{sign}(f(w,b,x_i) - y_i) * x_i$
$\partial L_i/\partial b = 3|f(w,b,x_i) - y_i|^2 * \text{sign}(f(w,b,x_i) - y_i)$

b.

$\partial L_i/\partial w = 3(f(w,b,x_i) - y_i^2)* x_i$
$\partial L_i/\partial b = 3(f(w,b,x_i) - y_i^2)$

c.

$\partial L_i/\partial w = \exp(f(w,b,x_i) - y_i) * x_i$
$\partial L_i/\partial b = \exp(f(w,b,x_i) - y_i)$

d.

When $1 - y_i * f(w,b,x_i) > 0$:

$\partial L_i/\partial w = -y_i x_i$
$\partial L_i/\partial b = -y_i$

When $1 - y_i * f(w,b,x_i) < 0$:

$\partial L_i/\partial w = 0$
$\partial L_i/\partial b = 0$

2. Loss Functions for Classification

Part 1:

a. This loss function is a good choice for classification. This function is better than the original hinge loss function as it will penalize incorrect classifications more strongly. Gradient descent can be used when the function is not equal to 0.

b. This loss function is not a good choice for classification. This function works best for regression as it will penalize correct classifications whose prediction is too far from the true value. Gradient descent can be used here as the function is differential.

c. This loss function is not a good choice for classification since it is an exponential function. Since its exponential, it can lead to unstable training. We can use gradient descent here since its also differential.

d. This loss function is a good choice for classification. We can use gradient descent here as well.

Part 2:

a.

If $1 - y_i * f(w,b,x_i) <= 0$, $L_i(w,b) = 0$ with the gradient being 0 as well
If $1 - y_i * f(w,b,x_i) > 0$, $L_i(w,b) = (1 - y_i * f(w,b,x_i))^2$

$\partial L_i/\partial w = 2(1 - y_i * f(w,b,x_i))(-y_ix_i) = -2/n (y_ix_i(1 - y_i * f(w,b,x_i)))$
$\partial L_i/\partial b = 2(1 - y_i * f(w,b,x_i))(-y_i) = -2/(ny_i(1 - y_i * f(w,b,x_i)))$

b.

$\partial L_i/\partial w = 4(y_i - f(w,b,x_i))^3(x_i) = 4/n (x_i(y_i - f(w,b,x_i))^3)$
$\partial L_i/\partial b = 4(y_i - f(w,b,x_i))^3(-1) = 4/n ((y_i - f(w,b,x_i))^3)$

c.

$\partial L_i/\partial w = \exp(f(w,b,x_i) - y_i) * x_i$
$\partial L_i/\partial b = \exp(f(w,b,x_i) - y_i)$

d.

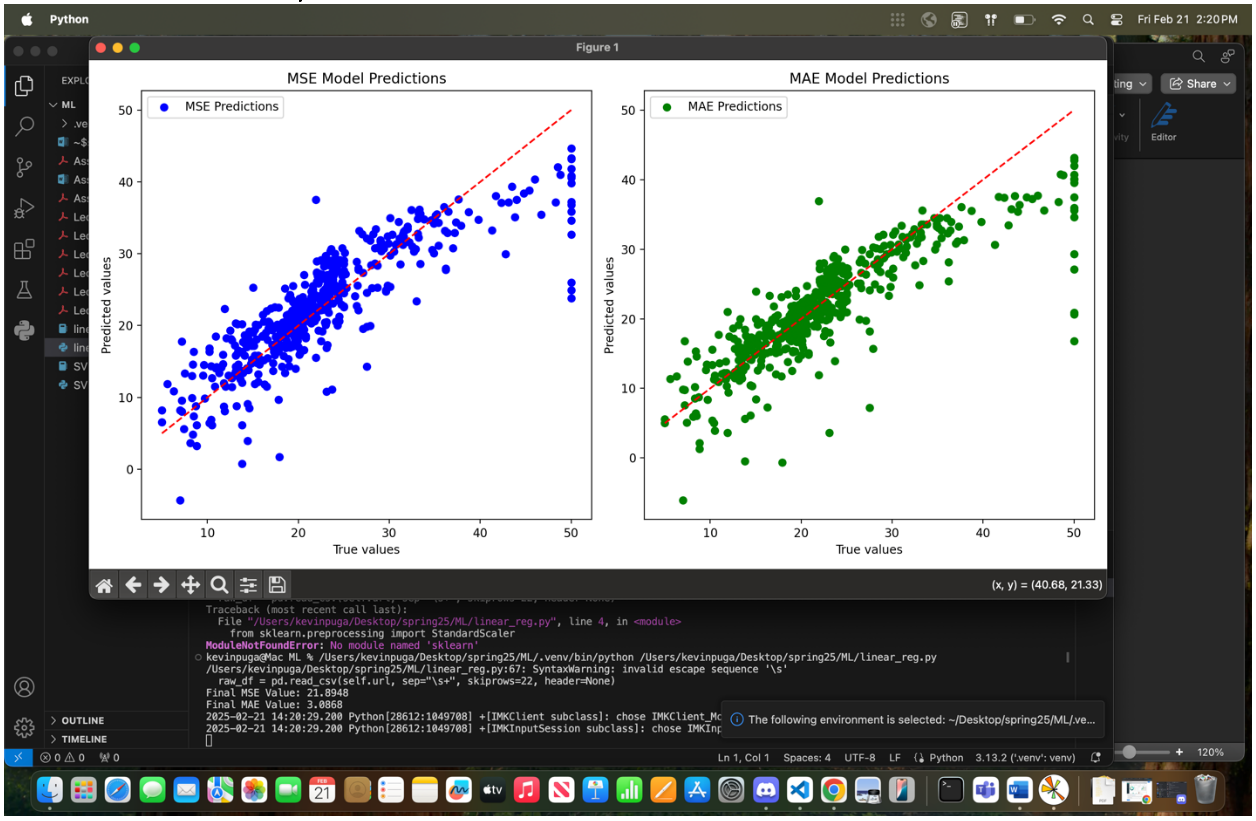
$\partial L_i/\partial w = \exp(-y_i * f(w,b,x_i))(y_ix_i) = -y_ix_i * \exp(-y_i * f(w,b,x_i))$
$\partial L_i/\partial b = \exp(-y_i * f(w,b,x_i))(-y_i) = -y_i* \exp(-y_i * f(w,b,x_i))$

3. <u>Polynomial and Higher Order features</u>

a. The problem with trying to fit a linear regression model without polynomial features is that it will lead to a poor fit and inaccurate predictions, making the model underfitting.
b. The polynomial features I would use are all powers currently present withing the function (cubic, square, etc.) In this case, the optimal weight vector would be the set of coefficients that minimizes the error between the predicted set and the actual set of values.

4. <u>Implement Linear Regression by Scratch</u>

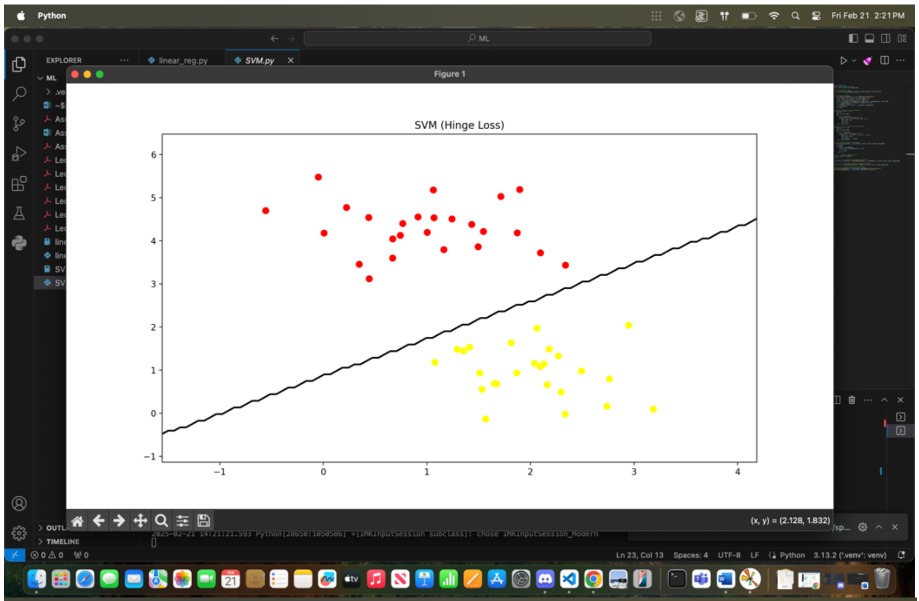
a. When comparing my values to the demo, my mse value was lower than that of the demo. I got 21.9 when the demo had a value of 24.3 or so. In terms of mae, I was very close to the demo as mine was 3.07 and the demo was 3.2.



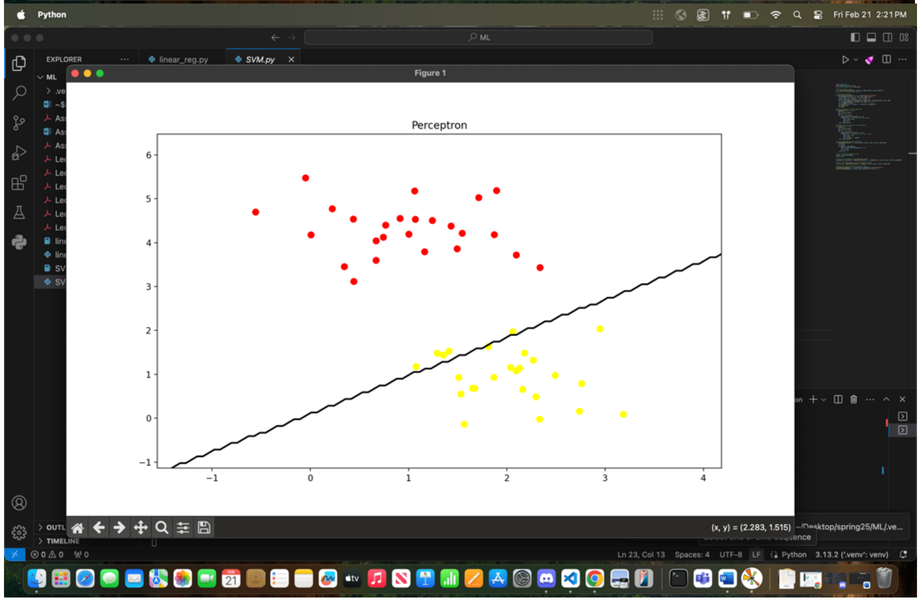
```
Traceback (most recent call last):
  File "/Users/kevinpuga/Desktop/spring25/ML/linear_reg.py", line 4, in <module>
    from sklearn.preprocessing import StandardScaler
ModuleNotFoundError: No module named 'sklearn'
kevinpuga@Mac ML % /Users/kevinpuga/Desktop/spring25/ML/.venv/bin/python /Users/kevinpuga/Desktop/spring25/ML/linear_reg.py
/Users/kevinpuga/Desktop/spring25/ML/linear_reg.py:67: SyntaxWarning: invalid escape sequence '\s'
  raw_df = pd.read_csv(self.url, sep="\s+", skiprows=22, header=None)
Final MSE Value: 21.8948
Final MAE Value: 3.0868
2025-02-21 14:20:29.200 Python[28612:1049708] +[IMKClient subclass]: chose IMKClient_Mc
2025-02-21 14:20:29.200 Python[28612:1049708] +[IMKInputSession subclass]: chose IMKInp
```

5. <u>Implement SVM and Perceptrons from Scratch</u>

a. For the perceptron loss graph, the margin that divides both plot points is going through the yellow data points. For the SVM loss graph, the margin is higher and doesn't go through any data points.



```
2025-02-21 14:21:21.593 Python[28650:1050586] +[IMKInputSession subclass]: chose IMKInputSession_Modern
```


Python
Fri Feb 21 2:21 PM
ML
EXPLORER
linear_reg.py
SVM.py
Figure 1
Perceptron
6
5
4
3
2
1
0
−1
−1
0
1
2
3
4
(x, y) = (2.283, 1.515)
OUTLINE
TIMELINE
Ln 23, Col 13 Spaces: 4 UTF-8 LF Python 3.13.2 ('.venv': venv)