Module 3: Critical Thinking Option 1

Jocelyn Strmec

Colorado State University Global

CSC580 Applying Machine Learning and Neural Networks - Capstone

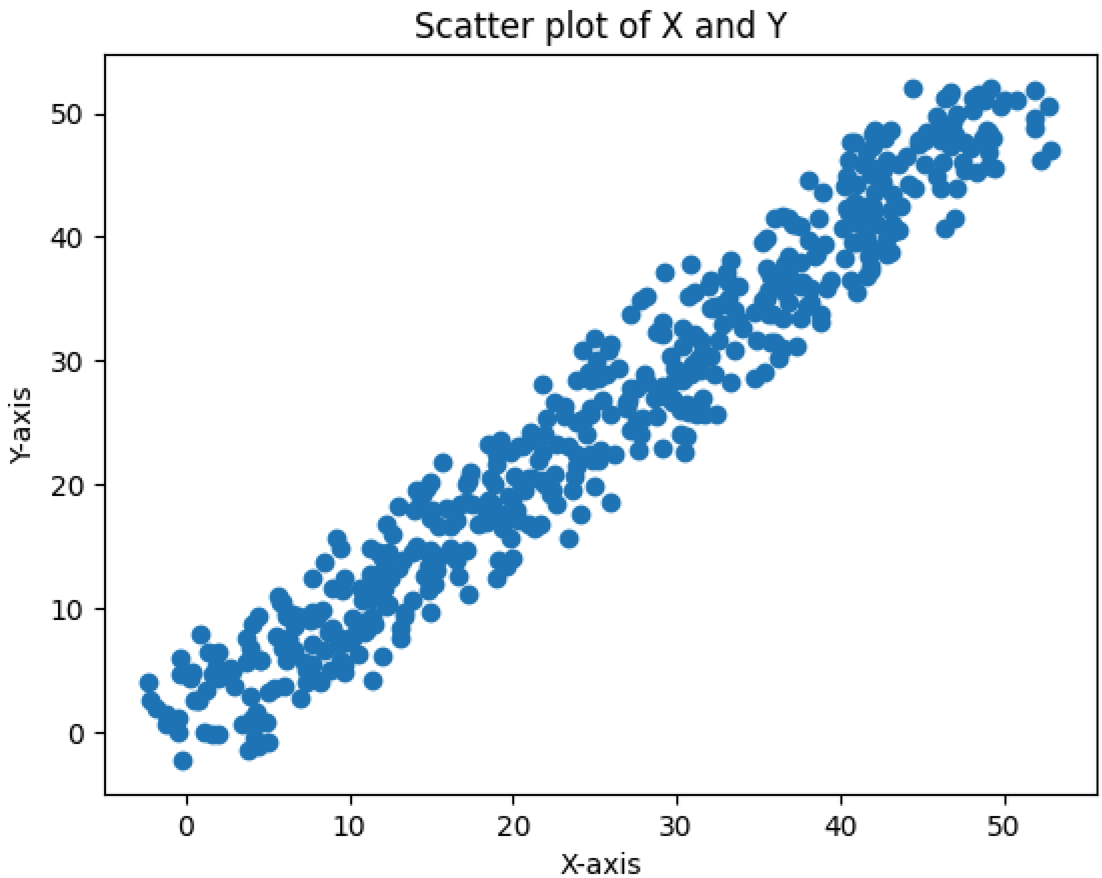
Pubali Banerjee

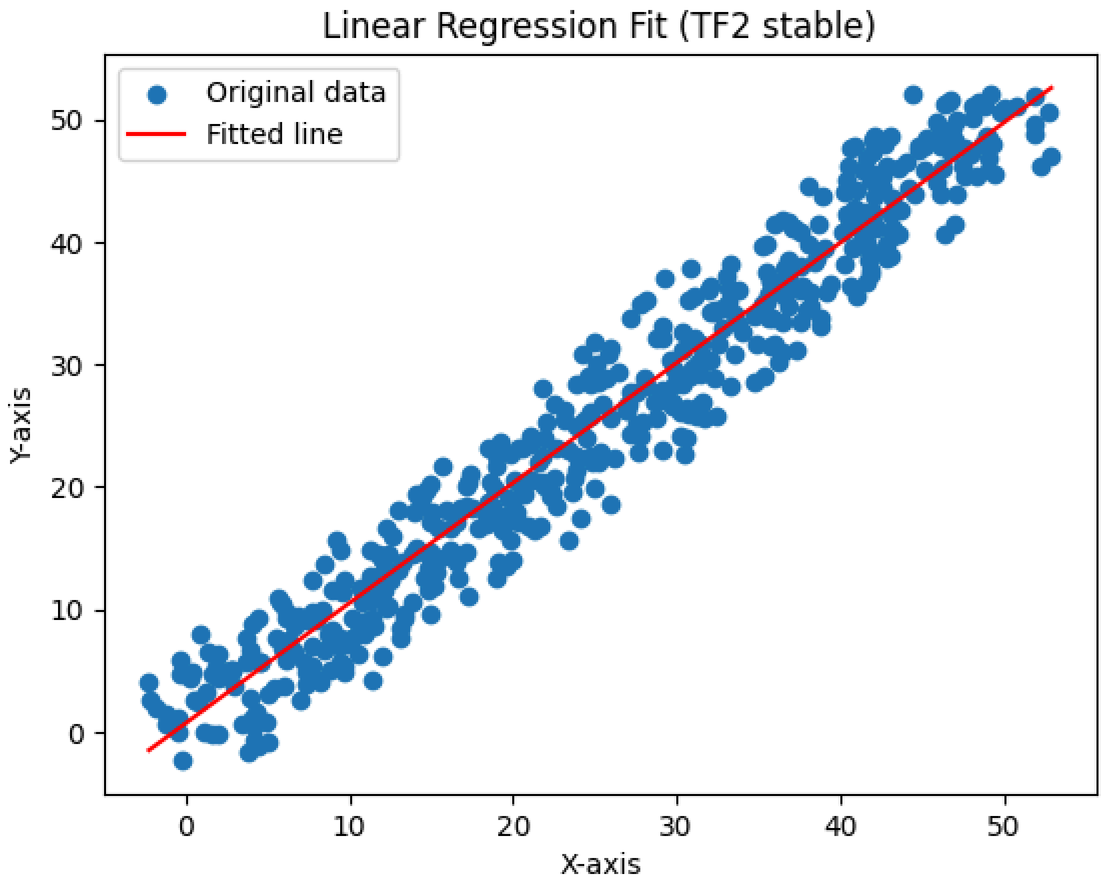
08/31/2025

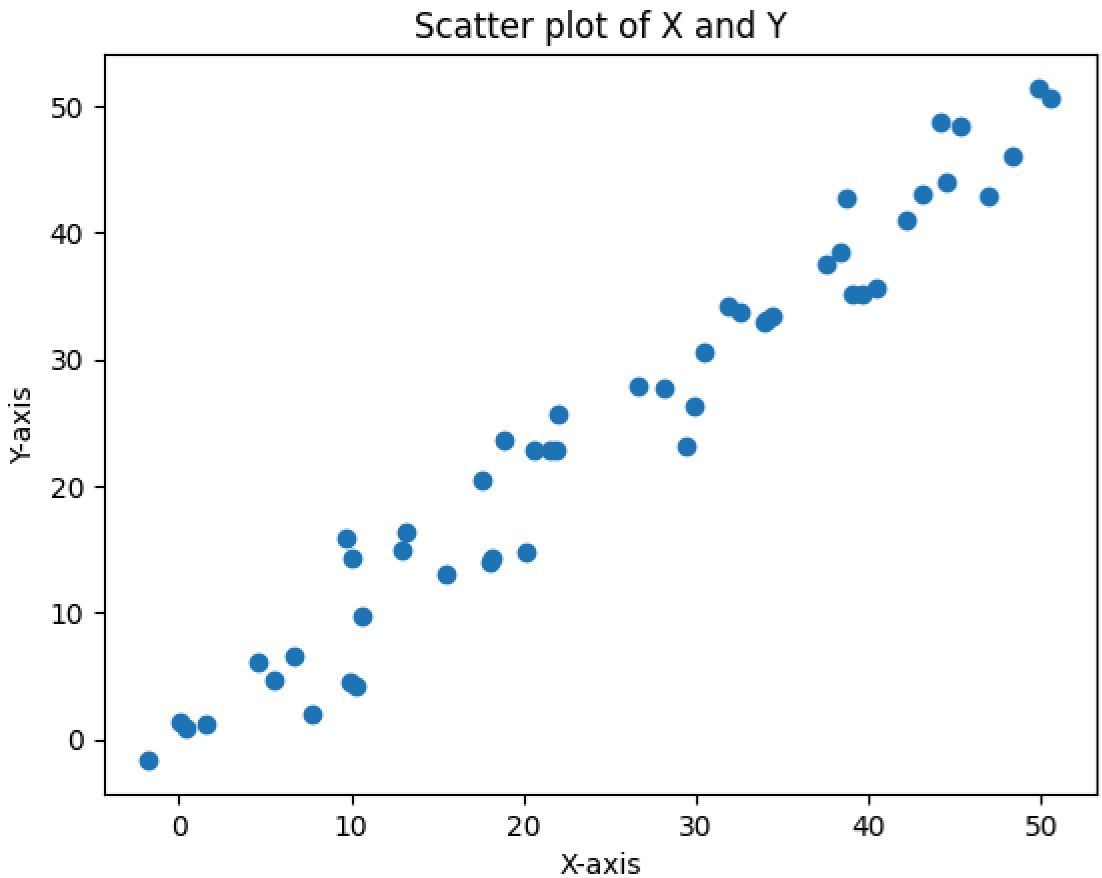
Linear regression works to predict the relationship between x and y parameters. In general regression is a low resource predictor and can suffice on minimal data points to make a fairly accurate prediction. In the code synthetic points are generated for x and y pairs, then plotted to show the raw distribution. Using a bounded random initialization for weight and bias of the regression, we can randomize a bit while binding the function from becoming too crazy. Gradient descent paired with Mean Squared Error (MSE) is used as the loss function to adjust W (weight) and b (bias or slope). This process runs for one thousand epochs outputting cost and parameter updates. Once the model is trained the predicted line is plotted. This allows the models progress and accuracy against the data to be evaluated. Linear regression models can accurately predict the random number points because it assumes a straight-line relationship between x and y in the form $y = Wx + b$. Gradient descent minimizes the errors, which measures how far predictions are from true values adjusting the predictions. Even though data has random noise, with repeated adjustments, gradually reducing error, the model is able to produce the best representation from the data's trend. Linear regression can be applied to various things such as housing prices using price and year, modeling patient outcomes using blood pressure vs. age or estimating linear relationships in measurements. Additionally, applying linear regression to predicting water quality proves quite useful because of linear regression's ability to work on minimal data, so researchers can make predictions even with smaller or limited sampling (Fernandes et al., 2023). As can be seen in the images even with ten data points the model is able to predict.

References

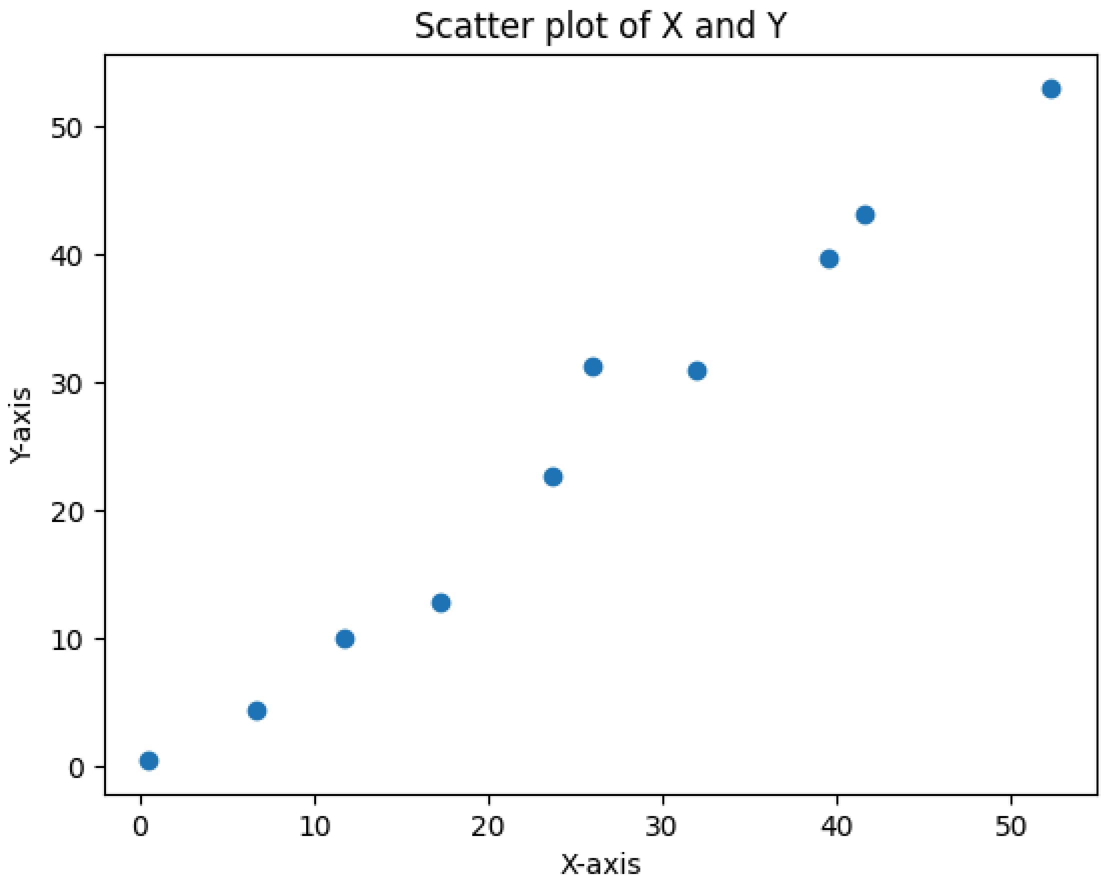
P Fernandes, A. C., R Fonseca, A., Pacheco, F. A. L., & Sanches Fernandes, L. F. (2023). Water quality predictions through linear regression - A brute force algorithm approach. *MethodsX*, *10*, 102153. https://doi.org/10.1016/j.mex.2023.102153

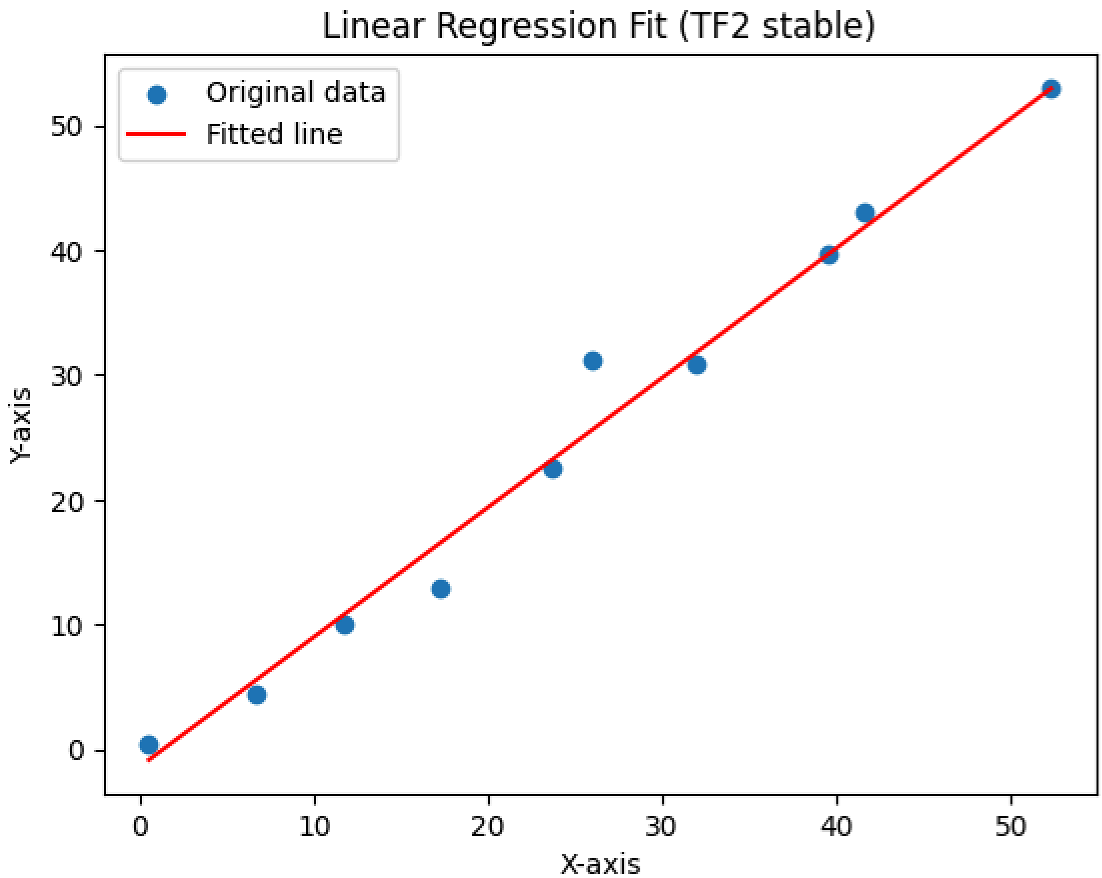

Scatter plot of X and Y
Y-axis
X-axis
50
40
30
20
10
0
0
10
20
30
40
50

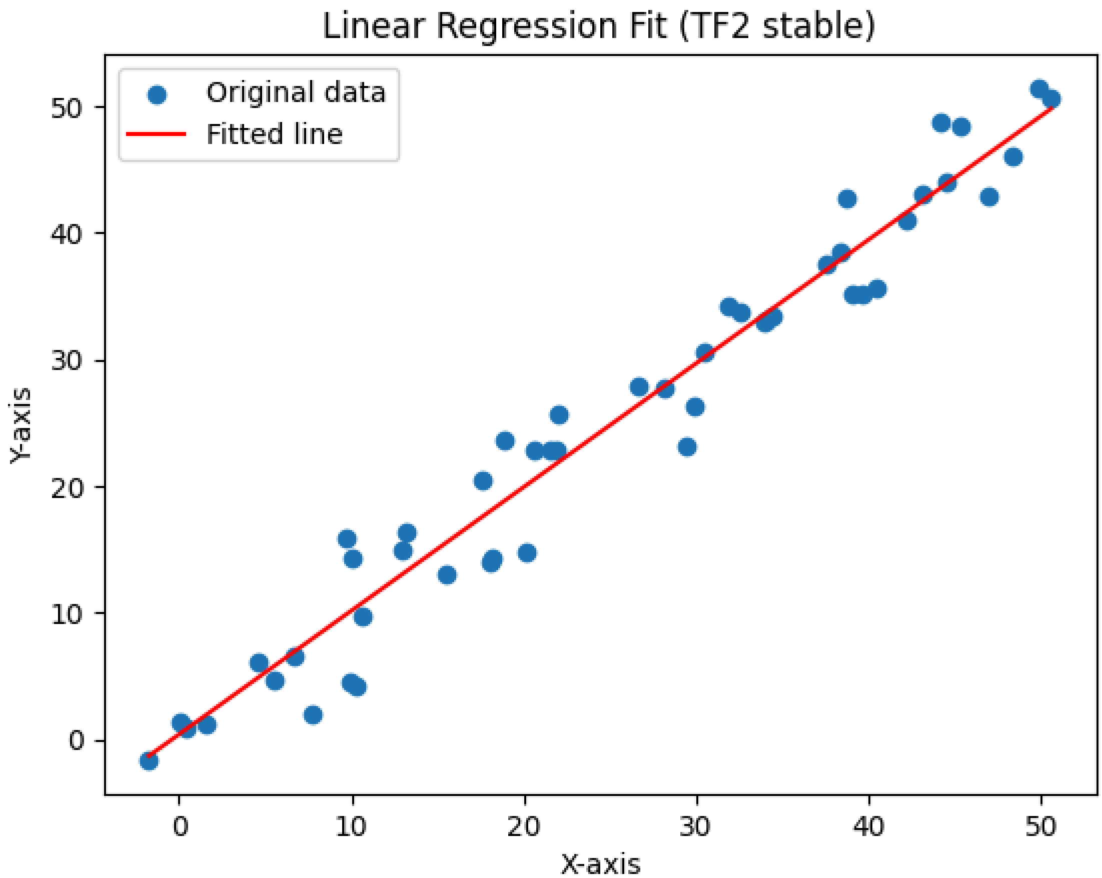

Linear Regression Fit (TF2 stable)
Original data
Fitted line
Y-axis
X-axis
0
10
20
30
40
50

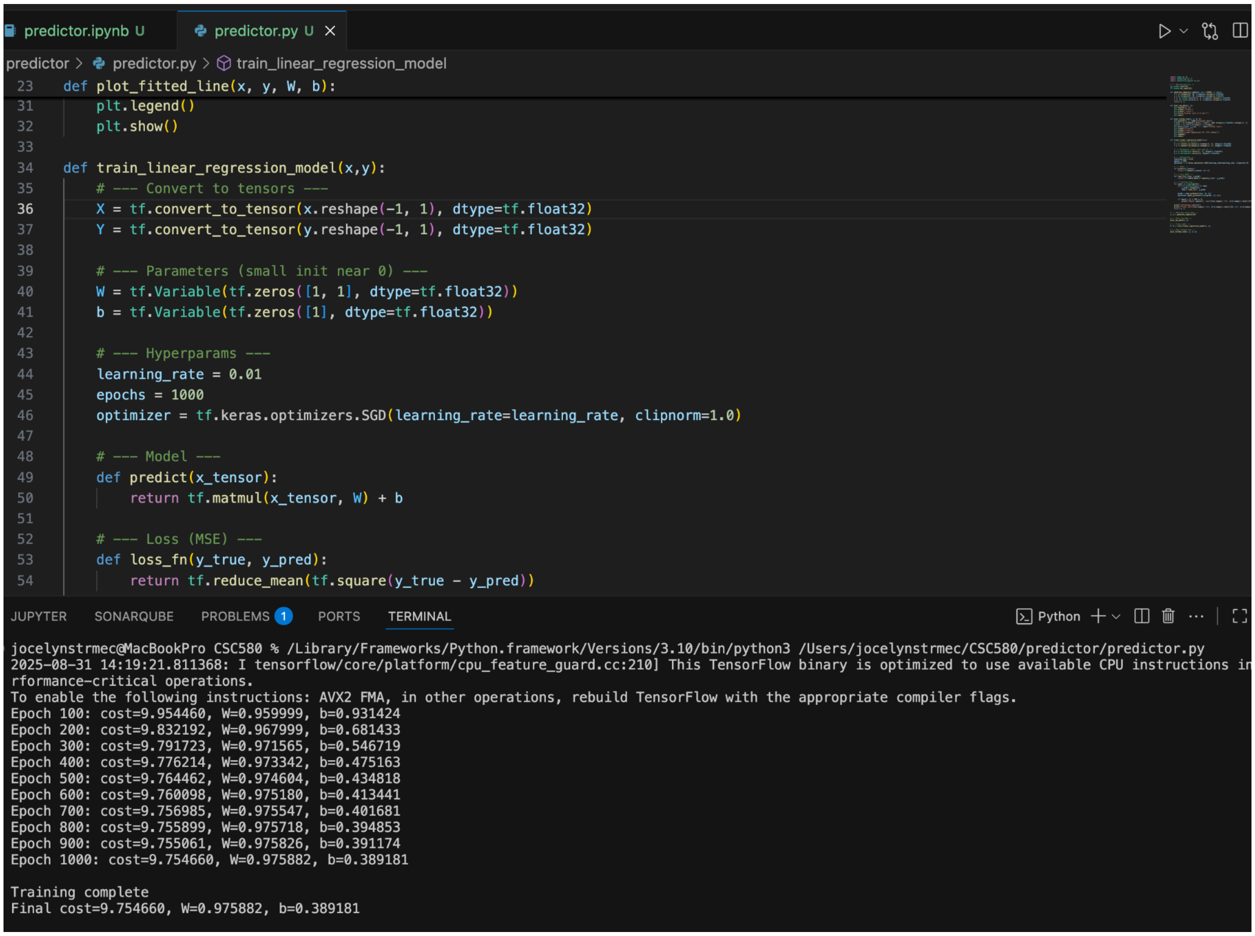


Scatter plot of X and Y
Y-axis
X-axis
0
10
20
30
40
50


Scatter plot of X and Y
Y-axis
X-axis
0
10
20
30
40
50
0
10
20
30
40
50


Linear Regression Fit (TF2 stable)
Original data
Fitted line
Y-axis
X-axis
0
10
20
30
40
50


Linear Regression Fit (TF2 stable)
Original data
Fitted line
Y-axis
X-axis
0
10
20
30
40
50



```
def plot_fitted_line(x, y, W, b):
    plt.legend()
    plt.show()

def train_linear_regression_model(x,y):
    # --- Convert to tensors ---
    X = tf.convert_to_tensor(x.reshape(-1, 1), dtype=tf.float32)
    Y = tf.convert_to_tensor(y.reshape(-1, 1), dtype=tf.float32)

    # --- Parameters (small init near 0) ---
    W = tf.Variable(tf.zeros([1, 1], dtype=tf.float32))
    b = tf.Variable(tf.zeros([1], dtype=tf.float32))

    # --- Hyperparams ---
    learning_rate = 0.01
    epochs = 1000
    optimizer = tf.keras.optimizers.SGD(learning_rate=learning_rate, clipnorm=1.0)

    # --- Model ---
    def predict(x_tensor):
        return tf.matmul(x_tensor, W) + b

    # --- Loss (MSE) ---
    def loss_fn(y_true, y_pred):
        return tf.reduce_mean(tf.square(y_true - y_pred))
```

```
jocelynstrmec@MacBookPro CSC580 % /Library/Frameworks/Python.framework/Versions/3.10/bin/python3 /Users/jocelynstrmec/CSC580/predictor/predictor.py
2025-08-31 14:19:21.811368: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in
rformance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
Epoch 100: cost=9.954460, W=0.959999, b=0.931424
Epoch 200: cost=9.832192, W=0.967999, b=0.681433
Epoch 300: cost=9.791723, W=0.971565, b=0.546719
Epoch 400: cost=9.776214, W=0.973342, b=0.475163
Epoch 500: cost=9.764462, W=0.974604, b=0.434818
Epoch 600: cost=9.760098, W=0.975180, b=0.413441
Epoch 700: cost=9.756985, W=0.975547, b=0.401681
Epoch 800: cost=9.755899, W=0.975718, b=0.394853
Epoch 900: cost=9.755061, W=0.975826, b=0.391174
Epoch 1000: cost=9.754660, W=0.975882, b=0.389181

Training complete
Final cost=9.754660, W=0.975882, b=0.389181
```