CS 4372

ASSIGNMENT 1

Names of students in your group:

James Hooper

Hritik Panchasara

Number of free late days used: 0

Note: You are allowed a **total** of 4 free late days for the **entire semester**. You can use at most 2 for each assignment. After that, there will be a penalty of 10% for each late day.

Please list clearly all the sources/references that you have used in this assignment.

For SGD-Regressor model, metrics, & preprocessing data split

https://scikit-learn.org/stable/

For Adam Optimizer understanding

https://towardsdatascience.com/10-gradient-descent-optimisation-algorithms-86989510b5e9

https://hackernoon.com/demystifying-different-variants-of-gradient-descent-optimization-algorithm-19ae9ba2e9bc

https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam

For graphing data

https://seaborn.pydata.org/

https://matplotlib.org/

For data manipulation

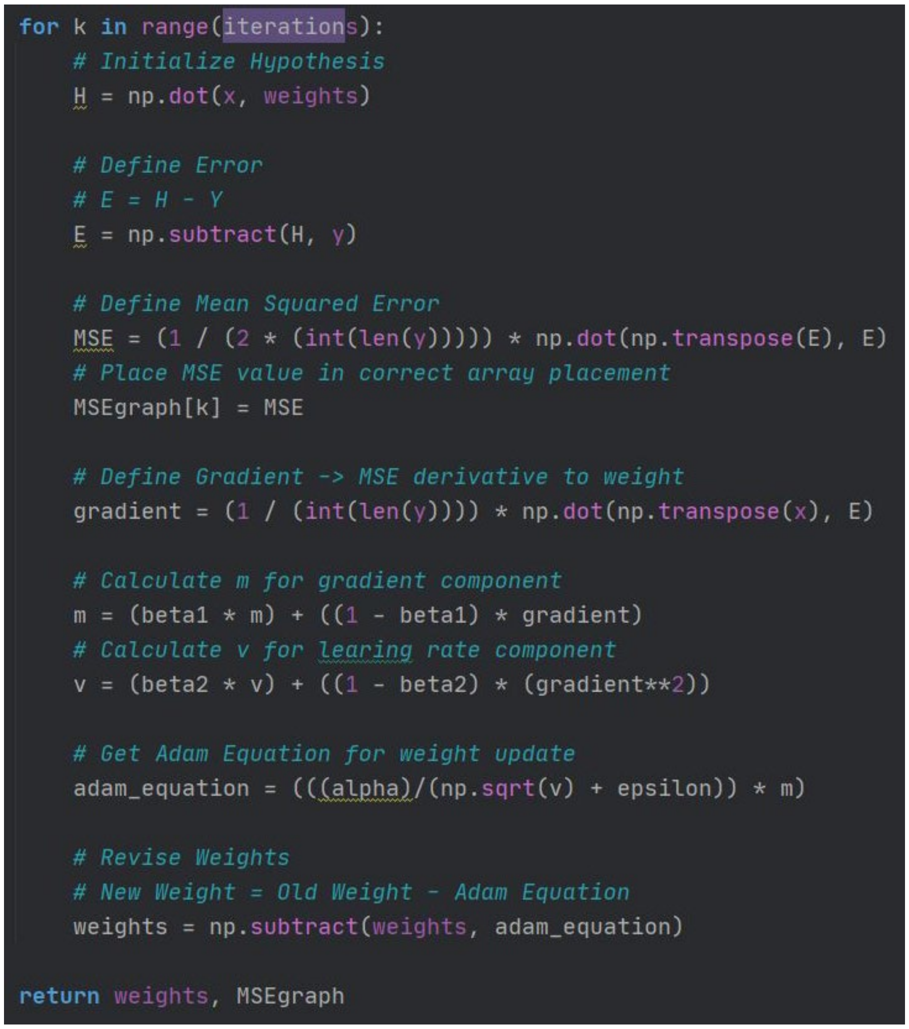
https://pandas.pydata.org/

https://numpy.org

Details of Enhanced Gradient Descent Algorithm: *Adam Optimizer*

The Adam Optimizer, also known as Adaptive moment estimation, is an enhancement to the vanilla gradient descent model. Adam utilizes two key concepts: an exponential moving average of gradients that is akin to momentum & a type of adaptive learning rate that changes upon each iteration. The parameters/equations used for the Adam optimization are as follows:

- alpha – the learning rate / step size
- beta1 – exponential decay rate for the first moment estimates
- beta2 – exponential decay rate for the second moment estimates
- epsilon – constant for numerical stability
- m & v – moving averages
    - m is "similar to the history used in Momentum GD"
    - v is "similar to the history used in RMSProp"
- g – gradient
- *The equations for the calculated values upon each iteration can be found in the snippet of code below.*



```
for k in range(iterations):
    # Initialize Hypothesis
    H = np.dot(x, weights)

    # Define Error
    # E = H - Y
    E = np.subtract(H, y)

    # Define Mean Squared Error
    MSE = (1 / (2 * (int(len(y))))) * np.dot(np.transpose(E), E)
    # Place MSE value in correct array placement
    MSEgraph[k] = MSE

    # Define Gradient -> MSE derivative to weight
    gradient = (1 / (int(len(y)))) * np.dot(np.transpose(x), E)

    # Calculate m for gradient component
    m = (beta1 * m) + ((1 - beta1) * gradient)
    # Calculate v for learing rate component
    v = (beta2 * v) + ((1 - beta2) * (gradient**2))

    # Get Adam Equation for weight update
    adam_equation = (((alpha)/(np.sqrt(v) + epsilon)) * m)

    # Revise Weights
    # New Weight = Old Weight - Adam Equation
    weights = np.subtract(weights, adam_equation)

return weights, MSEgraph
```