Instructor: Shaahid Ahmed Nadeem

Week 6 – Practice Problems

Deep-ML | Calculate Root Mean Square Error (RMSE)

```
import numpy as np
def rmse(y_true, y_pred):
        # Write your code here
        squared_errors = (y_true - y_pred) ** 2
        mse = np.mean(squared_errors)
        rmse_res = np.sqrt(mse)
        return round(rmse_res, 3)
```

Deep-ML | Calculate Mean Absolute Error (MAE)

```
import numpy as np
def mae(y_true, y_pred):
    absolute_errors = np.abs(y_true - y_pred)
    val = np.mean(absolute_errors)
    return round(val, 3)
```

Deep-ML | Calculate Accuracy Score

```
import numpy as np
def accuracy_score(y_true, y_pred):
    correct = np.sum(y_true == y_pred)
    total = len(y_true)
    accuracy = correct / total
    return accuracy
```

Deep-ML | Implement K-Nearest Neighbors

```
def k_nearest_neighbors(points, query_point, k):
    distances = []
    for point in points:
        dist = ((point[0] - query_point[0]) ** 2 + (point[1] - query_point[1]) ** 2) ** 0.5
        distances.append((dist, point))
    distances.sort(key=lambda x: x[0])
    return [point for dist, point in distances[:k]]
```

Deep-ML | Single Neuron

```
import math

def single_neuron_model(features: list[list[float]], labels: list[int], weights: list[float], bias: float) ->
(list[float], float):

    probabilities = []

    for feature_vector in features:

        z = sum(w * x for w, x in zip(weights, feature_vector)) + bias

        # weights -> [1,2,3] and feature vector -> [4,5,6]: zip() -> [(1,4),(2,5),(3,6)] -> 1*4 + 2*5 + 3*6

        # z = wx + b = mx + c

        prob = 1 / (1 + math.exp(-z)) # sigmoid -> [0,1]

        probabilities.append(prob)

        # perceptron

        # if z>0:

            # prob (y_pred) = +1

        #else:

            # -1

    mse = sum((prob - label) ** 2 for prob, label in zip(probabilities, labels)) / len(labels) #mean ->
sum/len

    probabilities = [round(prob, 4) for prob in probabilities]

    mse = round(mse, 4)

    return probabilities, mse
```