```
classdef NNClassifier
    properties
        x_train = [] % 785 x 5000 vector
        y_train = [] % 5000 x 1 vector
    end

    methods

        function obj = create(obj, x_train, y_train)
            obj.x_train = x_train;
            obj.y_train = y_train;
        end

        function prediction = predict(obj, x_test, y_test)
            K = 1;
            theResult = zeros(length(y_test), 1);
            labelVector = unique(y_test);
            theResult = zeros(length(y_test), 1);
            for testIndex = 1:length(y_test)
                for trainIndex = 1:length(obj.y_train)
                    distance(trainIndex) = norm(x_test(:, testIndex) - obj.x_train(:,↙
trainIndex));
                end

                [minimum, index] = min(distance(distance > 0));
                winnerNumber = obj.y_train(index);
                winnerIndex = find(labelVector == winnerNumber);
                theResult(testIndex) = labelVector(winnerIndex);

            end
            prediction = theResult;
        end

        function prediction = predictNorm(obj, x_test, y_test)
            K = 1;
            theResult = zeros(length(y_test), 1);
            labelVector = unique(y_test);
            theResult = zeros(length(y_test), 1);
            for testIndex = 1:length(y_test)
                for trainIndex = 1:length(obj.y_train)
                    cov_xy  = (obj.x_train(:, trainIndex).' * x_test(:,  testIndex));
                    variance = (obj.x_train(:, trainIndex).' * obj.x_train(:,↙
trainIndex));
                    scaleA = cov_xy / variance;
                    newTest = x_test(:,testIndex) / scaleA;
                    distance(trainIndex) = norm(newTest - obj.x_train(:,trainIndex));
                end

                [minimum, index] = min(distance(distance > 0));
                winnerNumber = obj.y_train(index);
                winnerIndex = find(labelVector == winnerNumber);
```

```
                theResult(testIndex) = labelVector(winnerIndex);

            end
            prediction = theResult;
        end

        function error = getError(obj, y_pred, labelTest)
            labelVector = unique(y_pred);
                theResult = zeros(size(labelVector));
                for indexPrediction = 1:length(y_pred)
                    if(y_pred(indexPrediction) ~= labelTest(indexPrediction))
                        errorIndex = find(labelVector == y_pred(indexPrediction));
                        theResult(errorIndex) = theResult(errorIndex) + 1;
                    end
                end

                for indexCount = 1:length(labelVector)
                    count(indexCount) = sum(y_pred == labelVector(indexCount));
                end
                error(:,1) = theResult;
                error(:,2) = count;

                for errorRateIndex = 1:length(labelVector)
                    rate(errorRateIndex) = theResult(errorRateIndex)/count↙
(errorRateIndex);
                end

                error(:,3) = rate;
        end

        function images = getImages(obj, x_test, y_test)
            imageCollection = zeros(10,2);
            c = 1;
            for testIndex = 1:length(y_test)
                for trainIndex = 1:length(obj.y_train)
                    distance(trainIndex) = norm(x_test(:, testIndex) - obj.x_train(:,↙
trainIndex));
                end

                [minimum, index] = min(distance(distance > 0));
                winnerNumber = obj.y_train(index);
                if(winnerNumber ~= y_test(testIndex) && c ~= 11)
                    imageCollection(c, 1) = index;
                    imageCollection(c, 2) = testIndex;
                    c = c + 1;
                end
                images = imageCollection;
            end
        end
```

```
    end

end
```