CSE-575: Statistical Machine Learning
Assignment 2

Name: Kunal Vinay Kumar Suthar
ASURite ID: 1215112535

---

1. SUPPORT VECTOR MACHINES

Solution 1.(a)
Lagrange Multipliers are used to perform maximum separation between classes in SVM. This is a dual function:

$$L(w,\ \alpha)\ =\ (1/2)(w.w)\ \ -\Sigma_j\ \alpha_j\ [(w.x_j\ +\ b)y_j - 1]\ where\ \alpha_j \geq 0,\ \ \forall j\ .$$

We will equate dL/dw = 0 to find the extremities and maximize the separation.

$$\Rightarrow dL/dw\ =\ w - \Sigma_j \alpha_j y_j x_j\ =\ 0\ .$$

$$\Rightarrow w\ = \Sigma_j \alpha_j y_j x_j \text{ and hence } b = y_k - w.x_k$$

Then $dL/db\ =\ -\ \Sigma_j \alpha_j y_j\ = 0$

Putting the values back in L, we get:

$$L = Max_\alpha \sum_j \alpha_j\ -\ \tfrac{1}{2} \sum_j \sum_k \alpha_j \alpha_k x_j x_k y_j y_k \qquad \text{Eq(1)}$$

We are given 2 support vectors in the question x_j , $y_j = 1$ and x_k, $y_k = -1$.

Using these we get:

$$w\ =\ \alpha_j x_j\ -\alpha_k x_k \text{ and } b\ =\ 1-w.x_j \text{ and } b\ =\ -1-w.x_k\ .$$

Hence, using the support vectors Eq(1) boils down to:

$$Max_\alpha\ (\alpha_j\ +\ \alpha_k) -\tfrac{1}{2}(x_j{}^2 \alpha_j{}^2\ +\ x_k{}^2 \alpha_k{}^2\ -2\alpha_j \alpha_k x_j x_k) \qquad \text{Eq(2)}$$

But $\sum_j \alpha_j y_j = 0\ from\ \frac{dL}{db}$

Hence $\alpha_j = \alpha_k = \alpha$

Therefore Eq(2) boils down to

$$Max_{\alpha}\ (2\alpha) - \tfrac{1}{2}\alpha^2 (x_j - x_k)^2 \qquad \text{Eq(3)}$$

Now, given in question:

$$arg\ max_{w,b} d = d^+ - d^- = \frac{w^T}{|w|}(x_j - x_k)$$

Now as $\frac{w^T}{|w|}$ is a unit vector, we can write $d^+ - d^- = x_j - x_k$ Eq(4)

Now, from Eq(3) and (4), We formulate the optimization problem for the hard-margin linear SVM as:

$$Max_{\alpha}\ (2\alpha) - \tfrac{1}{2}\alpha^2 (d^+ - d^-)^2$$

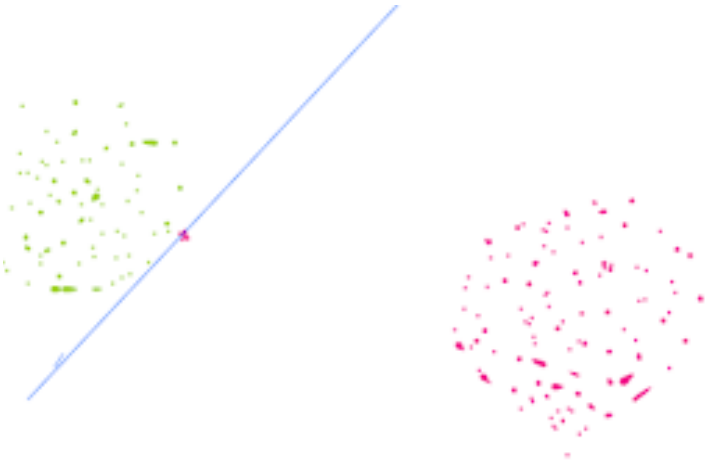
Where the constraints are

$\alpha > 0$,

$w = \alpha(x_j - x_k)$,

$b = 1 - w.x_k$ or $b = -1 - w.x_k$

Solution 1.(b) The limitation of the hard-margin linear SVM is that it only works correctly for linearly separable examples with no error or outliers. Otherwise it overfits. The example which demonstrates this limitation with a 2-dimensional feature space is as follows:



In the diagram above, a single pink outlier essentially determines the decision boundary.

Solution 2.(a)

Soft-margin SVM allows some amount of error in training as penalty C with slack ξ_i (for the ith example in the dataset)for each of the misclassified example to account for a dataset which is not linearly separable. Hard-margin does not include any penalty and thus its slack value ξ_i for every example is zero. Therefore, a Soft-margin SVM with its slack values ξ_i =0 for all examples is basically a Hard-Margin SVM. Therefore, a Hard-margin SVM does not allow any error while finding the decision boundary, whereas Soft-margin SVM does. Therefore, a Soft-margin SVM is more generalizable than Hard-Margin SVM.

Soft-margin SVM is given by:

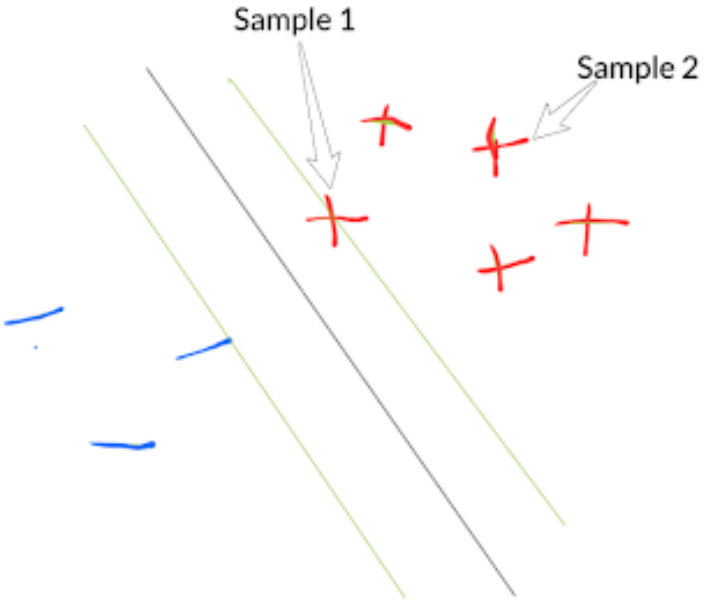
$$min_{w,b}\ w.w\ +\ C\sum_j \xi_j$$

$$(w.x_j + b)y_j \geq 1 - \xi_j,\ \ \forall j\ and\ \xi_j \geq 0\ \ \forall j$$

Solution 2.(b)

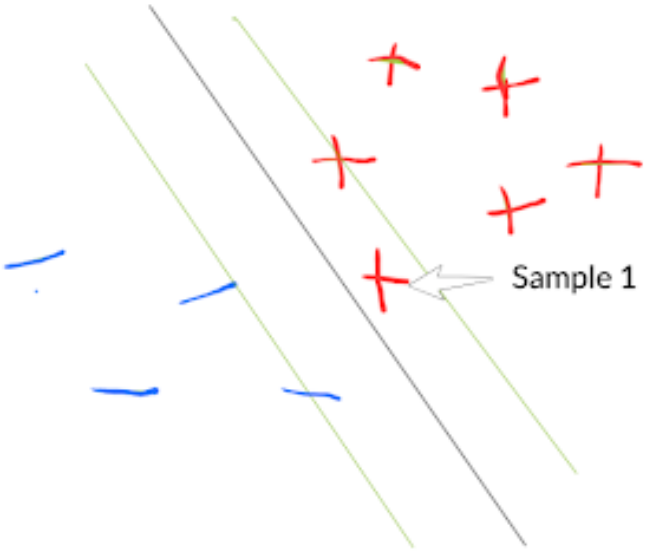
The three categories based on the value of ξ_j is as follows:

i) ξ_j **=0 :** This case is the hard-margin SVM. Consider the following example, in which all examples are correctly classified.

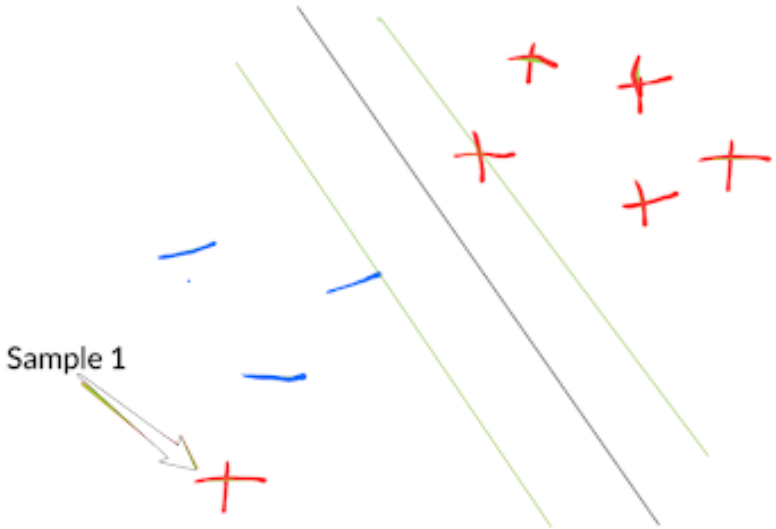


If sample 1 is removed then the decision boundary would change as it is on the support vector.
If sample 2 is removed then the decision boundary would not change as it is not on the support vector.

ii) $0 < \xi_j \leq 1$ **:** This case is the soft margin SVM which involves some slack. In the figure below, Sample 1 is correctly classified and removing it would not have any effect on the decision boundary.



iii) $\xi_j > 1$ **:** In this scenario, samples will be on the wrong side of the decision boundary and its removal will not affect the decision boundary.



---

2. ADABOOST

Solution 1.(a)

In the case of a balanced data set, the basic requirement of the Adaboost algorithm regarding the adopted weak classifiers should be that each classifier should be better than random.

Solution 1.(b)

We are given weak binary classifiers whose classification accuracy is less than 50%, given to be 30%. Then we know that $\varepsilon_t = 0.7$ (from 1-accuracy). We also know that for binary weak classifiers : $\alpha_t = \frac{1}{2} ln(\frac{1-\varepsilon_t}{\varepsilon_t})$. In our case it will be $\alpha_t = \frac{1}{2} ln(\frac{0.3}{0.7}) =$ -0.42364893. Therefore, we need to invert the answer of every classifier $sign(\alpha_t h_t(x))$. If $h_t(x)$ predicts positive and α_t is negative, the final output will predict negative. If $h_t(x)$ predicts negative and α_t is negative, the final output will predict positive. So, the final prediction will be the inversion of the prediction of the weak classifier and the performance of the combined classifier will improve with more iterations.

Solution 2.(a)

ITERATION 1:

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1

We are given $\theta = 2.5$. Then the weak classifier:

$$C(x) = \begin{cases} +1 & x < 2.5 \\ -1 & x \geq 2.5 \end{cases}$$

Correctly classified instances/examples, x = {0,1,2,3,4,5,9}
Incorrectly classified instances/examples, x = {6,7,8}

$D_1(j) = \frac{1}{10} \quad \forall j = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9$

$\Rightarrow Error, \varepsilon_t = \frac{1}{\sum_{\forall i} D_t(i)} \sum_{\forall i} D_t(i)\, \delta(h_t(x_i) \neq y_i)$

$\Rightarrow$ Therefore, $\varepsilon_1 = \frac{1}{1}(0.1 + 0.1 + 0.1) = 0.3$

$\Rightarrow \alpha_t = \frac{1}{2} ln(\frac{1-\varepsilon_t}{\varepsilon_t})$

$\Rightarrow \alpha_1 = \frac{1}{2} ln(\frac{1-0.3}{0.3}) = 0.42364893$

Now, for correctly predicted instances:

$D_{t+1}(i) = \frac{D_t(i)}{Z_t} exp(-\alpha_t)$

$\Rightarrow D_2(i) = \frac{(0.1)}{Z_1} exp(-0.42364893)\, for\, i = 0, 1, 2, 3, 4, 5, 9$

$\Rightarrow D_2(i) = \frac{0.0655}{Z_1}$

Now, for incorrectly predicted instances:

$D_{t+1}(i) = \frac{D_t(i)}{Z_t} exp(\alpha_t)$

$\Rightarrow D_2(i) = \frac{(0.1)}{Z_1} exp(0.42364893)\ for\ i = 6, 7, 8$

$\Rightarrow D_2(i) = \frac{0.1527}{Z_1}$

Now, we calculate Z_t:

$$Z_t = \sum_{\forall\ correctly\ classified\ i's} D_t(i)\ exp(-\alpha_t) + \sum_{\forall\ incorrectly\ classified\ i's} D_t(i)\ exp(\alpha_t)$$

$\Rightarrow Z_1 = 0.0655 * 7 + 0.1527 * 3 = 0.9166$

For correctly classified instances i= 0,1,2,3,4,5,9 : $D_2(i) = \frac{0.0655}{0.9166} = 0.0715$
For incorrectly classified instances i= 6,7,8 : $D_2(i) = \frac{0.1527}{0.9166} = 0.1666$
Hence,

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1
D_2	0.0715	0.0715	0.0715	0.0715	0.0715	0.0715	0.1666	0.1666	0.1666	0.0715

ITERATION 2:
The weights have been updated. Let us update θ also. Let $\theta = 8.2$ to get the minimum error.
Then the weak classifier:

$$C(x) = \begin{cases} +1 & x < 8.2 \\ -1 & x \geq 8.2 \end{cases}$$

Correctly classified instances/examples, x = {0,1,2,6,7,8,9}
Incorrectly classified instances/examples, x = {3,4,5}

$\Rightarrow$ Therefore, $\varepsilon_2 = \frac{1}{1}(0.0715 + 0.0715 + 0.0715) = 0.2145$
$\Rightarrow$ Hence, $\alpha_2 = \frac{1}{2} ln(\frac{1-0.2145}{0.2145}) = 0.649$

$Z_2 = 0.0374 * 4 + 0.0871 * 3 + 0.1368 * 3 = 0.8213$

For correctly predicted instances:
i) $D_3(i) = \frac{(0.0715)}{0.8213} exp(-0.649) = 0.0455 \qquad for\ i = 0, 1, 2, 9$

ii) $D_3(i) = \frac{0.1666}{0.8213} exp(-0.649) = 0.1061 \quad for\ i = 6, 7, 8$

For incorrectly predicted instances:
i) $D_3(i) = \frac{(0.0715)}{0.8213} exp(0.649) = 0.1666 \qquad for\ i = 3, 4, 5$

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1
D_3	0.0455	0.0455	0.0455	0.1666	0.1666	0.1666	0.1061	0.1061	0.1061	0.0455

ITERATION 3:

The weights have been updated. Let us update θ so that it minimizes classification error at this iteration. Let θ=2.5 again

Then the weak classifier:

$$C(x) = \begin{cases} +1 & x < 2.5 \\ -1 & x \geq 2.5 \end{cases}$$

Correctly classified instances/examples, x = {0,1,2,3,4,5,9}

Incorrectly classified instances/examples, x = {6,7,8}

$\Rightarrow$ Therefore, $\varepsilon_3 = \frac{1}{1}(0.1061 + 0.1061 + 0.1061) = 0.3183$

$\Rightarrow$ Hence, $\alpha_3 = \frac{1}{2} ln(\frac{1-0.3183}{0.3183}) = 0.3808$

$Z_3 = 0.0455 * 0.683 * 4 + 0.1666 * 0.683 * 3 + 0.1061 * 1.463 * 3 = 0.9309$

For correctly predicted instances:

i) $D_4(i) = \frac{(0.0455)}{0.9309} exp(-0.3808) = 0.0333 \quad for\ i = 0, 1, 2, 9$

ii) $D_4(i) = \frac{0.1666}{0.9309} exp(-0.3808) = 0.1222 \quad for\ i = 3, 4, 5$

For incorrectly predicted instances:

i) $D_4(i) = \frac{(0.1061)}{0.9309} exp(0.3803) = 0.1667 \quad for\ i = 6, 7, 8$

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1
D_4	0.0333	0.0333	0.0333	0.1222	0.1222	0.1222	0.1667	0.1667	0.1667	0.0333

ITERATION 4:

The weights have been updated. Let us update θ so that it minimizes classification error at this iteration. Let $\theta = 8.2$ again.
Then the weak classifier:

$$C(x) = \begin{cases} +1 & x < 8.2 \\ -1 & x \geq 8.2 \end{cases}$$

Correctly classified instances/examples, x = {0,1,2,6,7,8,9}
Incorrectly classified instances/examples, x = {3,4,5}

$\Rightarrow$ Therefore, $\varepsilon_4 = \frac{1}{1}(0.1222 + 0.1222 + 0.1222) = 0.3666$

$\Rightarrow$ Hence, $\alpha_4 = \frac{1}{2} ln(\frac{1-0.3666}{0.3666}) = 0.2734$

$Z_4 = 0.0333 * 0.7607 * 4 + 0.1667 * 0.7607 * 3 + 0.1222 * 1.3144 * 3 = 0.9635$

For correctly predicted instances:
i) $D_5(i) = \frac{(0.0333)}{0.9635} exp(-0.2734) = 0.0262 \quad for\ i = 0, 1, 2, 9$

ii) $D_5(i) = \frac{0.1667}{0.9635} exp(-0.2734) = 0.1316 \quad for\ i = 6, 7, 8$

For incorrectly predicted instances:
i) $D_5(i) = \frac{(0.1222)}{0.9635} exp(0.2734) = 0.1667 \quad for\ i = 3, 4, 5$

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1
D_5	0.0262	0.0262	0.0262	0.1667	0.1667	0.1667	0.1316	0.1316	0.1316	0.0262

ITERATION 5:
The weights have been updated. Let us update θ so that it minimizes classification error at this iteration. Let $\theta = 2.5$ again.
Then the weak classifier:

$$C(x) = \begin{cases} +1 & x < 2.5 \\ -1 & x \geq 2.5 \end{cases}$$

Correctly classified instances/examples, x = {0,1,2,3,4,5,9}
Incorrectly classified instances/examples, x = {6,7,8}

$\Rightarrow$ Therefore, $\varepsilon_5 = \frac{1}{1}(0.1316 + 0.1316 + 0.1316) = 0.3948$

$\Rightarrow$ Hence, $\alpha_5 = \frac{1}{2} ln(\frac{1-0.3948}{0.3948}) = 0.2136$

$Z_5 = 0.0262 * 0.80767 * 4 + 0.1667 * 0.80767 * 3 + 0.1316 * 1.2381 * 3 = 0.9773$

For correctly predicted instances:

i) $D_6(i) = \frac{(0.0262)}{0.9773} exp(-0.2136) = 0.02165 \qquad for\ i = 0, 1, 2, 9$

ii) $D_6(i) = \frac{0.1667}{0.9773} exp(-0.2136) = 0.1377 \quad for\ i = 3, 4, 5$

For incorrectly predicted instances:

i) $D_6(i) = \frac{(0.1316)}{0.9773} exp(0.2136) = 0.1667 \qquad for\ i = 6, 7, 8$

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1
D_6	0.0216	0.0216	0.0216	0.1377	0.1377	0.1377	0.1667	0.1667	0.1667	0.0216

ITERATION 6:

The weights have been updated. Let us update θ so that it minimizes classification error at this iteration. Let $\theta = 8.2$ again.

Then the weak classifier:

$$C(x) = \begin{cases} +1 & x < 8.2 \\ -1 & x \geq 8.2 \end{cases}$$

Correctly classified instances/examples, x = {0,1,2,6,7,8,9}

Incorrectly classified instances/examples, x = {3,4,5}

$\Rightarrow$ Therefore, $\varepsilon_6 = \frac{1}{1}(0.1377 + 0.1377 + 0.1377) = 0.4131$

$\Rightarrow$ Hence, $\alpha_6 = \frac{1}{2} ln(\frac{1-0.4131}{0.4131}) = 0.1755$

$Z_6 = 0.0216 * 0.839 * 4 + 0.1667 * 0.839 * 3 + 0.1377 * 1.19 * 3 = 0.9834$

For correctly predicted instances:

i) $D_6(i) = \frac{(0.0216)}{0.9834} exp(-0.1755) = 0.0184 \qquad for\ i = 0, 1, 2, 9$

ii) $D_6(i) = \frac{0.1667}{0.9834} exp(-0.1755) = 0.1422 \quad for\ i = 6, 7, 8$

For incorrectly predicted instances:

i) $D_6(i) = \frac{(0.1377)}{0.9834} exp(0.1755) = 0.1667 \qquad for\ i = 3, 4, 5$

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1
D_6	0.0184	0.0184	0.0184	0.1667	0.1667	0.1667	0.1422	0.1422	0.1422	0.0184

Solution 2.(b)
After 4 iterations, the weights of the training samples change in such a way that the training error ε_t increases with every iteration. We get the minimum error i.e. 0.2145 at the end of the second iteration. We can stop after this iteration, because the error increases from here onwards. Therefore, the fifth and sixth weak classifiers do not improve the performance of the combined classifier. The value of θ also keeps oscillating between values in two ranges, so that we can get the minimum possible error at that iteration.

---

3. KNN CLASSIFIER

Solution 1.(a)
When a new training sample becomes available, among SVM, Naive Bayes and KNN, only **SVM** has to be re-trained from scratch. This is because in the case of SVM, the additional new data might change the identities of the support vector. Therefore, the SVM model needs to be trained again. For Naive Bayes, the count of data-points for every class would change, which would result in different probability values for prior and class conditional probability. These can be updated easily and hence no re-training is required. For KNN, we just need to append the new data to the training set and then we can easily predict/classify. Hence, there is again no need for re-training.

Solution 1.(b)
When a new test sample becomes available, among SVM, Naive Bayes and KNN, the classifier which needs the most computation to infer the class label for this sample is **KNN**. This is because, when a new test sample becomes available in the case of KNN, we need to compute this sample's distance from every point. Then we need to find the k nearest neighbours. Then from the k nearest neighbours, we need to find the majority label. Naive Bayes based classification will only involve calculation through probabilities, which will be comparatively very fast to KNN in terms of computation. SVM based prediction will also be faster as it would only need to compute which side of the decision boundary hyperplane does the sample exist. Hence KNN is the most computationally inefficient algorithm amongst the given algorithms.

<u>Time Complexity for the classifier:</u>
Assumption: The number of features of the sample is **1** , which is significantly smaller than n.
(i) O(n): There will be computation involving calculation of distance of this new sample from every other sample. Calculation of distance between a pair of samples will take O(1) time and there will be n such calculations. Therefore, O(n).

ii) O(nlog(n)): After computing the n distances, we need to find the k nearest/smallest distances. I assume that we use QuickSort algorithm for sorting. We then select the k nearest points. Note: If we use a MaxHeap, we can get the k-nearest points in O(nlog(k)).
iii) O(k): After getting the k nearest points, we need to find the majority class label. For that we need to traverse through all the k-nearest points.

Therefore, in total the time complexity of this algorithm is **O(n) + O(nlog(n)) + O(k)** .

Solution 2.(a) Pseudocode for KNN Classifier:

1. **Loading the data:** Used keras library to load the mnist dataset(it has been mentioned in the assignment that using libraries/packages for the data is allowed) involving 60000 training images and labels and 10000 testing images and labels. Then the dataset size was truncated to the first 6000 training samples and the first 1000 test samples.

2. **Calculation of Euclidean distance between every test and training sample:** The euclidean distance is then calculated between the 1000 test and 6000 training samples, and the result is then stored in a 1000 x 6000 matrix.
   Function:

```
Function calculate_distances(train_data, test_data)
        For i=0 to test_data.size-1
             For j=0 to train_data.size-1
                 difference= (test_data[i]-train_data[j])
                 square= difference^2
                 sum= sum_{k=0}^{783} square_k

                 sqrroot= sqrt(sum)
                 distance_mat[i][j]=sqrroot

         Return distance_mat
```

3. **Prediction of label for the test set using the distance_mat from the above function using different K:**

```
Function apply_KNN()
         For i=0 to test_data.size-1
             distance_from_train= distance_mat[i]
             labels=train_label
```

Sort distance_from_train and also change labels' variable o position accordingly.

Get the first K nearest labels after sorting distances

Store the most frequent label in Predictions

Return Predictions

4. <u>Calculate the accuracy and error of the KNN classifier:</u>

Count the total number of correct predictions made from Predictions in the above functions, and calculate the accuracy and error.

Accuracy = No. of correct predictions/(size of test_data)

Error = 1-Accuracy

Solution 2.(b) Error Curves for training and test errors: