Machine Learning Assignment 1

Shivangi Aneja

29-October-2017

The given data set is as follows

i	$x_{i,1}$	$x_{i,2}$	$x_{i,3}$	y_i
A	5.5	0.5	4.5	2
B	7.4	1.1	3.6	0
C	5.9	0.2	3.4	2
D	9.9	0.1	0.8	0
E	6.9	-0.1	0.6	2
F	6.8	-0.3	5.1	2
G	4.1	0.3	5.1	1
H	1.3	-0.2	1.8	1
I	4.5	0.4	2.0	0
J	0.5	0.0	2.3	1
K	5.9	-0.1	4.4	0
L	9.3	-0.2	3.2	0
M	1.0	0.1	2.8	1
N	0.4	0.1	4.3	1
O	2.7	-0.5	4.2	1

Problem 1

We have following data at root node

TargetAttribute	Count
0	5
1	6
2	4

The formula for Gini Index is given as :

$$i_G(t) = 1 - \sum_{c_i \in C} \pi_{c_i}^2$$

At root node

$$i_G(t) = 1 - [(5/15)^2 + (6/15)^2 + (4/15)^2] = 0.658$$

Testing all possible splits at root node for attribute $x_{i,1}$

Target	0.4		0.5		1		1.3		2.7		4.1		4.5	
	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$
$y = 0$	0	5	0	5	0	5	0	5	0	5	0	5	1	4
$y = 1$	1	5	2	4	3	3	4	2	5	1	6	0	6	0
$y = 2$	0	4	0	4	0	4	0	4	0	4	0	4	0	4
Gini	0.0	0.663	0.0	0.662	0.0	0.652	0.0	0.628	0.0	0.58	0.0	0.494	0.245	0.5
Impurity	0.6188		0.574		0.5216		0.460		0.387		0.2968		0.381	

Target	5.5		5.9		6.8		6.9		7.4		9.3		9.9	
	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$
$y = 0$	1	4	2	3	2	3	2	3	3	2	4	1	5	0
$y = 1$	6	0	6	0	6	0	6	0	6	0	6	0	6	0
$y = 2$	1	3	2	2	3	1	4	0	4	0	4	0	4	0
Gini	0.406	0.489	0.56	0.48	0.595	0.375	0.61	0.0	0.639	0.0	0.653	0.0	0.657	N.D
Impurity	0.445		0. 533		0.536		0.488		0.5538		0.6095		N.A	

Testing all possible splits at root node for attribute $x_{i,2}$

Target	-0.5		-0.3		-0.2		-0.1		0.0		0.1		0.2	
	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$
$y = 0$	0	5	0	5	1	4	2	3	2	3	3	2	3	2
$y = 1$	1	5	1	5	2	4	2	4	3	3	5	1	5	1
$y = 2$	0	4	1	3	1	3	2	2	2	2	2	2	3	1
Gini	0.0	0.663	0.5	0.651	0.625	0.661	0.67	0.642	0.653	0.656	0.62	0.64	0.645	0.625
Impurity	0.6188		0.631		0.6514		0.6532		0.6546		0.627		0.644	

Target	0.3		0.4		0.5		1.1	
	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$
$y = 0$	3	2	4	1	4	1	5	0
$y = 1$	6	0	6	0	6	0	6	0
$y = 2$	3	1	3	1	4	0	4	0
Gini	0.625	0.45	0.639	0.5	0.653	0	0.658	N.D.
Impurity	0.59		0.6205		0.6095		N.A.	

Testing all possible splits at root node for attribute $x_{i,3}$

Target	0.6		0.8		1.8		2.0		2.3		2.8		3.2	
	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$
$y = 0$	0	5	1	4	1	4	2	3	2	3	2	3	3	2
$y = 1$	0	6	0	6	1	5	1	5	2	4	3	3	3	3
$y = 2$	1	3	1	3	1	3	1	3	1	3	1	3	1	3
Gini	0.0	0.643	0.5	0.639	0.67	0.653	0.625	0.644	0.64	0.66	0.61	0.67	0.612	0.656
Impurity	0.60013		0.620		0.6564		0.638		0.653		0.646		0.635	

Target	3.4		3.6		4.2		4.3		4.4		4.5		5.1	
	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$
$y = 0$	3	2	4	1	4	1	4	1	5	0	5	0	5	0
$y = 1$	3	3	3	3	4	2	5	1	5	1	5	1	6	0
$y = 2$	2	2	2	2	2	2	2	2	2	2	3	1	4	0
Gini	0.656	0.653	0.642	0.61	0.64	0.64	0.628	0.625	0.625	0.45	0.651	0.5	0.658	N.D
Impurity	0.6546		0. 6292		0.64		0.6272		0.59		0.6308		N.A	

The least Gini Index is for the split value $x_{i,1} \leq 4.1$. Thus this is the splitting condition for the root node.

The node N1 is a pure node, thus no further splitting possible.The data at node N1 is shown below

i	$x_{i,1}$	$x_{i,2}$	$x_{i,3}$	y_i
G	4.1	0.3	5.1	1
H	1.3	-0.2	1.8	1
J	0.5	0.0	2.3	1
M	1.0	0.1	2.8	1
N	0.4	0.1	4.3	1
O	2.7	-0.5	4.2	1

The node N2 is not a pure node and thus it is needed to split further. The data at node N2 is as follows :

i	$x_{i,1}$	$x_{i,2}$	$x_{i,3}$	y_i
A	5.5	0.5	4.5	2
B	7.4	1.1	3.6	0
C	5.9	0.2	3.4	2
D	9.9	0.1	0.8	0
E	6.9	-0.1	0.6	2
F	6.8	-0.3	5.1	2
I	4.5	0.4	2.0	0
K	5.9	-0.1	4.4	0
L	9.3	-0.2	3.2	0

Testing all possible splits at node N2 for attribute $x_{i,1}$

Target	4.5		5.5		5.9		6.8		6.9		7.4		9.3	
	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$
$y = 0$	1	4	1	4	2	3	2	3	2	3	3	2	4	1
$y = 2$	0	4	1	3	2	2	3	1	4	0	4	0	4	0
Gini	0.0	0.5	0.5	0.489	0.5	0.48	0.48	0.375	0.45	0.0	0.489	0	0.5	0.0
Impurity	0.44		0.491		0.49		0.43		0.3		0.38		0.45	

Testing all possible splits at node N2 for attribute $x_{i,2}$

Target	-0.3		-0.2		-0.1		0.1		0.2		0.4		0.5	
	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$
$y = 0$	0	5	1	4	2	3	3	2	3	2	4	1	4	1
$y = 2$	1	3	1	3	2	2	2	2	3	1	3	1	4	0
Gini	0.0	0.468	0.5	0.489	0.5	0.48	0.48	0.5	0.5	0.44	0.489	0.5	0.5	0.0
Impurity	0.416		0.491		0.49		0.49		0.48		0.491		0.44	

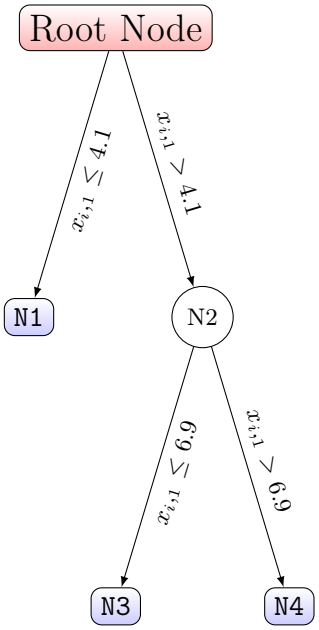
Testing all possible splits at node N2 for attribute $x_{i,3}$

Target	0.6		0.8		2.0		3.2		3.4		3.6		4.4	
	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$
$y = 0$	0	5	1	4	2	3	3	2	3	2	4	1	5	0
$y = 2$	1	3	1	3	1	3	1	3	2	2	2	2	2	2
Gini	0.0	0.468	0.5	0.489	0.44	0.5	0.375	0.48	0.48	0.5	0.44	0.44	0.408	0.0
Impurity	0.416		0.491		0.48		0.43		0.48		0.44		0.317	

Target	4.5		5.1	
	$\leq$	$>$	$\leq$	$>$
$y = 0$	5	0	5	0
$y = 2$	3	1	4	0
Gini	0.468	0.0	0.493	N.D
Impurity	0.416		N.A.	

The least Gini Index at node N2 is for the split value $x_{i,1} \leq 6.9$. Thus this is the splitting condition for the node N2.

The decision tree is as follows:



The following table shows the distribution of classes and Gini Index at the nodes

Node	Gini Index	y=0	y=1	y=2
N1	0.0	0	6	0
N2	0.494	5	0	4
N3	0.45	2	0	4
N4	0.0	3	0	0

Problem 2

We are given 2 points as follows

(a) x_a = (4.1,-0.1,2.2)

Using the Decison tree of Problem1, x_a,1 = 4.1, this data point will lie in node N1

Also, this is a pure leaf node with all target values $y = 1$

p($c = 0 \mid x_a$,T) = 0

p($c = 1 \mid x_a$,T) = 1

p($c = 2 \mid x_a$,T) = 0

$\hat{y}$ = arg max p(y = c | x) = p($c = 1 \mid x_a$,T)

Thus for x_a , the predicted value of $y = 1$

(b) x_b = (6.1,0.4,1.3)

Using the Decison tree of Problem1, x_b,1 = 6.1, this data point will lie in node N3

p($c = 0 \mid x_b$,T) = 1/3

p($c = 1 \mid x_b$,T) = 0

p($c = 2 \mid x_b$,T) = 2/3

$\hat{y}$ = arg max p(y = c | x) = p($c = 2 \mid x_b$,T)

Thus for x_b , the predicted value of $y = 2$

Problem 3

This programming assignment is attached below

October 29, 2017

1 Programming assignment 1: k-Nearest Neighbors classification

```
In [1]: import numpy as np
        from sklearn import datasets, model_selection
        import matplotlib.pyplot as plt
        %matplotlib inline
```

1.1 Introduction

For those of you new to Python, there are lots of tutorials online, just pick whichever you like best :)

If you never worked with Numpy or Jupyter before, you can check out these guides * https://docs.scipy.org/doc/numpy-dev/user/quickstart.html * http://jupyter.readthedocs.io/en/latest/

1.2 Your task

In this notebook code to perform k-NN classification is provided. However, some functions are incomplete. Your task is to fill in the missing code and run the entire notebook.

In the beginning of every function there is docstring, which specifies the format of input and output. Write your code in a way that adheres to it. You may only use plain python and `numpy` functions (i.e. no scikit-learn classifiers).

Once you complete the assignments, export the entire notebook as PDF using nbconvert and attach it to your homework solutions. On a Linux machine you can simply use `pdfunite`, there are similar tools for other platforms too. You can only upload a single PDF file to Moodle.

1.3 Load dataset

The iris data set (https://en.wikipedia.org/wiki/Iris_flower_data_set) is loaded and split into train and test parts by the function `load_dataset`.

```
In [2]: def load_dataset(split):
            """Load and split the dataset into training and test parts.

            Parameters
            ----------
            split : float in range (0, 1)
                Fraction of the data used for training.
```

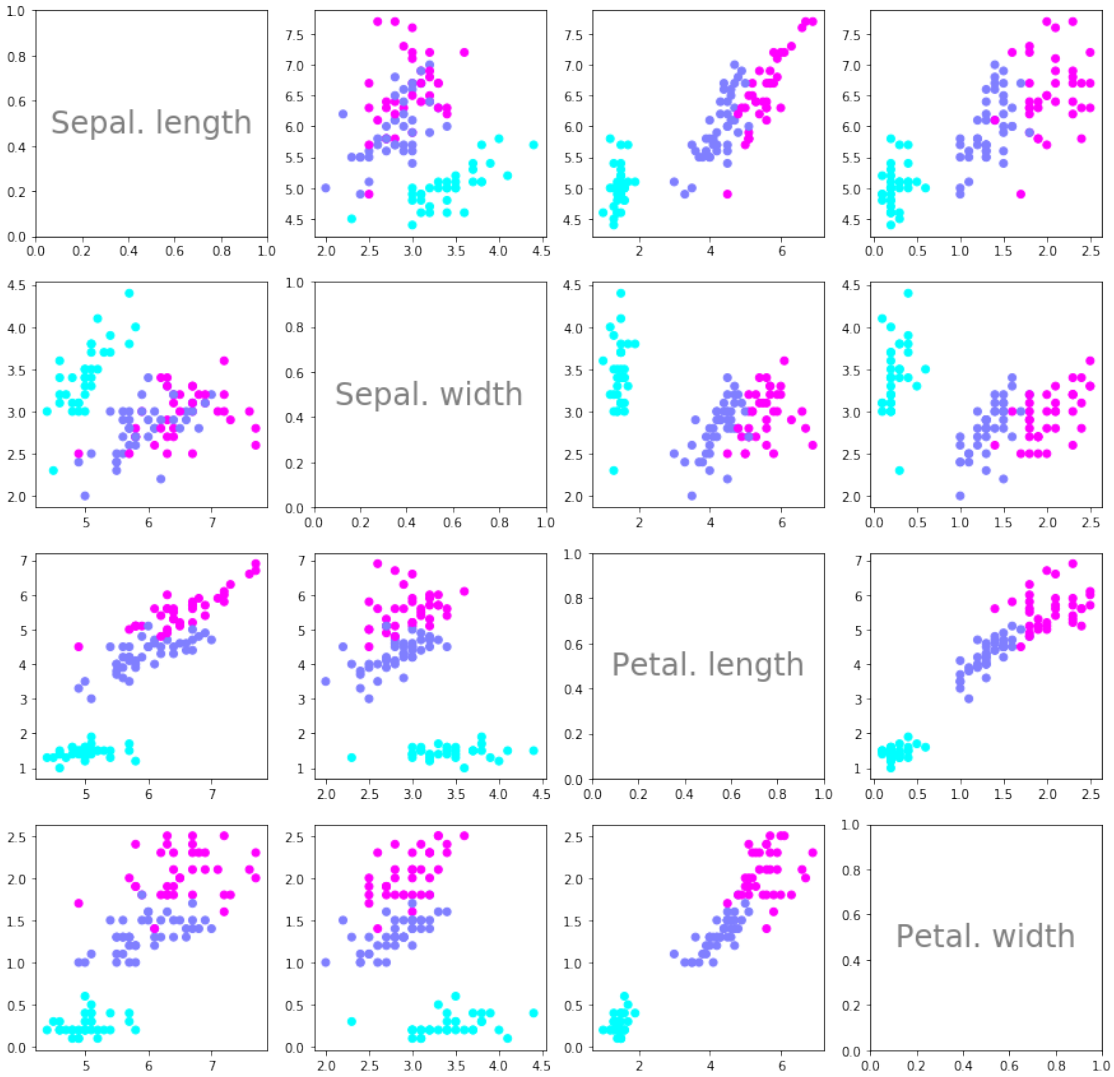
```
        Returns
        -------
        X_train : array, shape (N_train, 4)
            Training features.
        y_train : array, shape (N_train)
            Training labels.
        X_test : array, shape (N_test, 4)
            Test features.
        y_test : array, shape (N_test)
            Test labels.
        """
        dataset = datasets.load_iris()
        X, y = dataset['data'], dataset['target']
        X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, random_sta
        return X_train, X_test, y_train, y_test
```

```
In [3]: # prepare data
        split = 0.75
        X_train, X_test, y_train, y_test = load_dataset(split)
```

1.4 Plot dataset

Since the data has 4 features, 16 scatterplots (4x4) are plotted showing the dependencies between each pair of features.

```
In [4]: f, axes = plt.subplots(4, 4,figsize=(15, 15))
        for i in range(4):
            for j in range(4):
                if j == 0 and i == 0:
                    axes[i,j].text(0.5, 0.5, 'Sepal. length', ha='center', va='center', size=24,
                elif j == 1 and i == 1:
                    axes[i,j].text(0.5, 0.5, 'Sepal. width', ha='center', va='center', size=24,
                elif j == 2 and i == 2:
                    axes[i,j].text(0.5, 0.5, 'Petal. length', ha='center', va='center', size=24,
                elif j == 3 and i == 3:
                    axes[i,j].text(0.5, 0.5, 'Petal. width', ha='center', va='center', size=24,
                else:
                    axes[i,j].scatter(X_train[:,j],X_train[:,i], c=y_train, cmap=plt.cm.cool)
```



1.5 Task 1: Euclidean distance

Compute Euclidean distance between two data points.

```
In [5]: def euclidean_distance(x1, x2):
            """Compute Euclidean distance between two data points.

            Parameters
            ----------
            x1 : array, shape (4)
                First data point.
            x2 : array, shape (4)
                Second data point.
```

```
    Returns
    -------
    distance : float
        Euclidean distance between x1 and x2.
    """
    # TODO
     # Calculate Eucledian Distance between 2 points
    return np.linalg.norm(np.array(x1)-np.array(x2))
```

1.6 Task 2: get k nearest neighbors' labels

Get the labels of the *k* nearest neighbors of the datapoint *x_new*.

```
In [6]: def get_neighbors_labels(X_train, y_train, x_new, k):
    """Get the labels of the k nearest neighbors of the datapoint x_new.

    Parameters
    ----------
    X_train : array, shape (N_train, 4)
        Training features.
    y_train : array, shape (N_train)
        Training labels.
    x_new : array, shape (4)
        Data point for which the neighbors have to be found.
    k : int
        Number of neighbors to return.

    Returns
    -------
    neighbors_labels : array, shape (k)
        Array containing the labels of the k nearest neighbors.
    """
    # TODO
    # euc_dist_array is the array that stores Eucledian distance value between every poi
    euc_dist_array = np.array([])
    # TODO
    for x_data in X_train:
         result = euclidean_distance(x_data,x_new)
         euc_dist_array = np.append(euc_dist_array,result)

    # min_dis_index_arr stores the index of X-train datapoints with minimum eucledian di
    min_dis_index_arr = euc_dist_array.argsort()[:k]


    y_predicted_arr = np.array([])
    for y_index in min_dis_index_arr:
        y_predicted_arr = np.append(y_predicted_arr,y_train[y_index])
```

```
    return y_predicted_arr
```

1.7 Task 3: get the majority label

For the previously computed labels of the k nearest neighbors, compute the actual response. I.e. give back the class of the majority of nearest neighbors. In case of a tie, choose the "lowest" label (i.e. the order of tie resolutions is 0 > 1 > 2).

```
In [7]: def get_response(neighbors_labels, num_classes=3):
            """Predict label given the set of neighbors.

            Parameters
            ----------
            neighbors_labels : array, shape (k)
                Array containing the labels of the k nearest neighbors.
            num_classes : int
                Number of classes in the dataset.

            Returns
            -------
            y : int
                Majority class among the neighbors.
            """
            # TODO
            u, indices = np.unique(neighbors_labels, return_inverse=True)
            return u[np.argmax(np.bincount(indices))]
```

1.8 Task 4: compute accuracy

Compute the accuracy of the generated predictions.

```
In [29]: def compute_accuracy(y_pred, y_test):
             """Compute accuracy of prediction.

             Parameters
             ----------
             y_pred : array, shape (N_test)
                 Predicted labels.
             y_test : array, shape (N_test)
                 True labels.
             """
             y_pred_np = np.array(y_pred)
             y_test_np = np.array(y_test)

             acc_count = np.sum(y_pred == y_test)

             acc_per = float(acc_count)/float(y_test.size)

             return acc_per
```

```
In [9]: # This function is given, nothing to do here.
        def predict(X_train, y_train, X_test, k):
            """Generate predictions for all points in the test set.

            Parameters
            ----------
            X_train : array, shape (N_train, 4)
                Training features.
            y_train : array, shape (N_train)
                Training labels.
            X_test : array, shape (N_test, 4)
                Test features.
            k : int
                Number of neighbors to consider.

            Returns
            -------
            y_pred : array, shape (N_test)
                Predictions for the test data.
            """
            y_pred = []
            for x_new in X_test:
                neighbors = get_neighbors_labels(X_train, y_train, x_new, k)
                y_pred.append(get_response(neighbors))
            return y_pred
```

1.9 Testing

Should output an accuracy of 0.9473684210526315.

```
In [32]: # prepare data
         split = 0.75
         X_train, X_test, y_train, y_test = load_dataset(split)
         print('Training set: {0} samples'.format(X_train.shape[0]))
         print('Test set: {0} samples'.format(X_test.shape[0]))

         # generate predictions
         k = 3
         y_pred = predict(X_train, y_train, X_test, k)
         accuracy = compute_accuracy(y_pred, y_test)
         print('Accuracy = {0}'.format(accuracy))
```

```
Training set: 112 samples
Test set: 38 samples
Accuracy = 0.947368421053
```

Problem 4

K Nearest Neighbour Algorithm with $k = 3$ and Euclidean Distance.

Two vectors are as follows

(a) x_a = (4.1,-0.1,2.2)

Below table shows the Euclidean Distance of the given dataset with the datapoint x_a

Datapoint	Euclidean Distance
A	2.75
B	3.78
C	2.18
D	5.96
E	3.22
F	3.96
G	2.92
H	2.83
I	0.67
J	3.60
K	2.84
L	5.29
M	3.16
N	4.25
O	2.47

The 3 nearest neighbours to point x_a are C($y = 2$),I($y = 0$) and O($y = 1$).

p($c = 0 \mid x_a$,k) = 1/3

p($c = 1 \mid x_a$,k) = 1/3

p($c = 2 \mid x_a$,k) = 1/3

$\hat{y}$ = arg max p(y = c | x) and there are equal probabilities for all the target values. So , to break the tie we need to look for next nearest neighbour which is A($y = 2$)

Thus for x_a , the predicted value of $y = 2$

(b) x_b = (6.1,0.4,1.3)

Below table shows the Euclidean Distance of the given dataset with the datapoint x_b

Datapoint	Euclidean Distance
A	3.25
B	2.73
C	2.11
D	3.84
E	1.17
F	3.92
G	4.29
H	4.86
I	1.74
J	5.703
K	3.14
L	3.76
M	5.32
N	6.44
O	4.55

The 3 nearest neighbours to point x_b are C($y = 2$), E($y = 2$)) and I($y = 0$)

p($c = 0 \mid x_b$,k) = 1/3

p($c = 1 \mid x_b$,k) = 0

p($c = 2 \mid x_b$,k) = 2/3

$\hat{y}$ = arg max p(y = c | x)

Thus for x_b , the predicted value of $y = 2$

Problem 5

The formula for k-NN regression is as follows:

$\hat{y} = \frac{1}{Z} \sum_{i \epsilon N_k(x))} \frac{1}{d(x,x_i)} y_i$

with $Z = \sum_{i \epsilon N_k(x))} \frac{1}{d(x,x_i)}$

(a) x_a = (4.1,-0.1,2.2) : The 3 nearest neighbours to point x_a are C($y = 2$),I($y = 0$) and O($y = 1$).

$d(x, x_C) = 2.18$, $d(x, x_I) = 0.67$, $d(x, x_O) = 2.47$

Z = $\frac{1}{2.18} + \frac{1}{0.67} + \frac{1}{2.47} = 2.356$

$y_C = 2$, $y_I = 0$, $y_O = 1$

$\hat{y} = \frac{1}{2.356} [\frac{2}{2.18} + \frac{0}{0.67} + \frac{1}{2.47}] = 0.56124$

(b) x_a = (6.1,0.4,1.3) : The 3 nearest neighbours to point x_b are C($y = 2$), E($y = 2$)) and I($y = 0$)

$d(x, x_C) = 2.11$, $d(x, x_E) = 1.17$, $d(x, x_I) = 1.74$

Z = $\frac{1}{2.11} + \frac{1}{1.17} + \frac{1}{1.74} = 1.9033$

$y_C = 2$, $y_E = 2$, $y_I = 0$

$\hat{y} = \frac{1}{1.9033} [\frac{2}{2.11} + \frac{2}{1.17} + \frac{0}{1.74}] = 1.39613$

Problem 6

The problems faced in building k-NN model using Euclidean Model are as follows :

(a) **Computationally Expensive** : As the attributes used for prediction of the target increases the more calculations are to be done for computing Euclidean Distance and thus Space and Time Complexity increases. Thus time to compute distance of a new sample from the points in dataset also increases.

(b) **Scaling Issues** : If we compute Euclidean distance without normalizing or standardizing the data , then there is a high possibility that one attribute may influence the data and will thus produce counter intuitive results. For ex., Consider attribute AGE(2 digit number usually) and INCOME(3-4 digit number). In this case even if there is a large difference in age (say 20 years) and a small difference in salary (say 500 euros), the salary will still influence and dominate the value of Euclidean Distance

Thus classifying unknown record using k-NN classifier is expensive

We can do following things to overcome the problem :

(a) To overcome the problem of computation time and space , we can use high configuration processors or GPU's , but these again incur some costs.

(b) To overcome the Scaling issues , we first need to nomalize the data and then apply any distance measure on that data or using Mahalanobis distance.This produces better results. But there again is cost involved in normalizing the data. Also each time we get a new

sample we will have to normalize this new sample always. before applying the distance measure(not for Mahalanobis)

Training Decision Tree is totally different from training a k-NN classifier in the following ways :

(a) We do not need to store any distance from each data point like in K-NN because it works on rules which are made using Impurity metrics. So space complexity is very less as compared to K-NN classifier.

(b)Decision Tree uses rules to classify the new sample data. We do not need to normlize the data for decision trees, because it does not compute distance between sample, instead it computes Impurity Measures like Gini Index , Entropy etc which is scale irrelevant. And these measures only involve the relative probability of the target class.