3. In class, we looked at an example where all the attributes were binary (i.e., yes/no valued). Consider an example where instead of the attribute “Morning?”, we had an attribute “Time” which specifies when the class begins.

(a) We can pick a threshold τ and use (Time < τ)? as a criteria to split the data in two. Explain how you might pick the optimal value of τ .

Yes, for continuous attributes we can pick a threshold and keep the deciding factor for branching of tree as < the threshold or greater than the threshold. The value can be initially selected by looking at the range of values in the training data set.

We could use a sliding window to pick the time value. A fixed window is selected and then we check the points in that range , then we repeat the step with the next range of window to obtain the optimum threshold. We could pick time 3 a.m. first, one window would be 12-3 a.m and the other window would be 3am to 12 p.m. and so on. We plot a histogram for the favorable and unfavorable outcomes for each range, then move the next range and keep doing it till we get the optimum threshold.

This value can be picked by first checking the range of this value in the test data and then probably by taking half of the higher end of the threshold as the root node decision point or if we know that we get maximum negative points below X p.m. then we can use this X as a deciding point. In some cases, we might have a better deciding attribute which could be used instead.

We could also calculate the likes and dislikes and find the avg time for each of these , the calculate the mean of that to get the optimal threshold. Eg like=[1,2,3,4,5,6] notLike=[6,7,8,9,10,11] averageLike = 3.5 averageNotLike = 8.5 .Threshold = 6

(b) In the decision tree learning algorithm discussed in class, once a binary attribute is used, the subtrees do not need to consider it. Explain why when there are continuous attributes this may not be the case.

When the attribute is binary the first decision made using the attribute would be with it’s 2 possible values 1 or 0. Once these 2 values are considered there is no need to check for it again as each branch would contain just that one value for the attribute. The continuous value attribute on the other hand would have the decision made based on a range of values , once decision is made at a branch each branch could again consider values from the range to further branch. The child node values need to consider the parent node decision to check for deviations from the main range.

4. Why memorizing the training data and doing table lookups is a bad strategy for learning? How do we

prevent that in decision trees?

Memorizing the training data and doing table lookups could result in a very well-trained model just with respect to training data, the model would not work great when tested with new test data as it is not ready for anything but the train data. This leads to overfitting where labels get perfectly predicted for training data but fails miserably in predicting test data labels. Instead we could select a part of the training data as validation data set and verify the model, before testing with the test set.

To prevent this, we should perform pre-pruning and post-pruning of data. Pre-pruning stop growing the tree earlier before perfect classification and post pruning allows the tree to classify the training set and then post prune the tree. Basically, we make sure that the tree is not too deep to prevent over fitting .

5. What does the decision boundary of 1-nearest neighbor classifier for 2 points (one positive, one negative)

look like?

In one nearest neighbor, every point will be its own Voronoi cell, which form the decision boundary. If there are 2 points , one positive and one negative then the decision boundary would be a line separating these two points.

6. Does the accuracy of a kNN classifier using the Euclidean distance change if you (a) translate the data

(b) scale the data (i.e., multiply the all the points by a constant), or (c) rotate the data? Explain.

Answer the same for a kNN classifier using Manhattan distance

a) if we translate the test data the point which were closer to one cluster might get closer to another cluster .But, we also translate the test data by the same factor then points would be closer again, therefore no change in the accuracy as compared to before and after translation. This applies to both KNN by Euclidean and Manhattan distance

b)Scaling would result in multiplying the data points by a constant , as scaling would be applied on all the points equally. The difference in the location(distance) of the test to various training points would also get scaled similarly for all the points . Scaling does not affect accuracy of prediction in Manhattan or

Euclidean distance (as both are based distance differences).Though the points get scaled out the ration remain the same.

c)Euclidean distance between train data and test data would rotate equally around the origin before and after the rotation , hence the accuracy would not be affected .Whereas for the Manhattan distance x and y intercept would be affected after rotation , resulting in changing position of the train data, which would be lead to changes in the accuracy.

7. Implement kNN in Matlab or Python for handwritten digit classification and submit all codes and plots: (a) Download MNIST digit dataset (60,000 training and 10,000 testing data points) and the starter code from the course page. Each row in the matrix represents a handwritten digit image. The starter code shows how to visualize an example data point in Matlab. The task is to predict the class (0 to 9) for a given test image, so it is a 10-way classification problem.

(b) Write a Matlab or Python function that implements kNN for this task and reports the accuracy for each class (10 numbers) as well as the average accuracy (one number). [acc acc av] = kNN(images train, labels train, images test, labels test, k) where acc is a vector of length 10 and acc av is a scalar. Look at a few correct and wrong predictions to see if it makes sense. To speed it up, in all experiments, you may use only the first 1000 testing images.

['Class : 0 - Accuracy : 99.03', 'Class : 1 - Accuracy : 100.0', 'Class : 2 - Accuracy : 86.87', 'Class : 3 - Accuracy : 93.91', 'Class : 4 - Accuracy : 90.8', 'Class : 5 - Accuracy : 96.81', 'Class : 6 - Accuracy : 95.35', 'Class : 7 - Accuracy : 93.97', 'Class : 8 - Accuracy : 85.57', 'Class : 9 - Accuracy : 92.16']

average accuracy 93.5

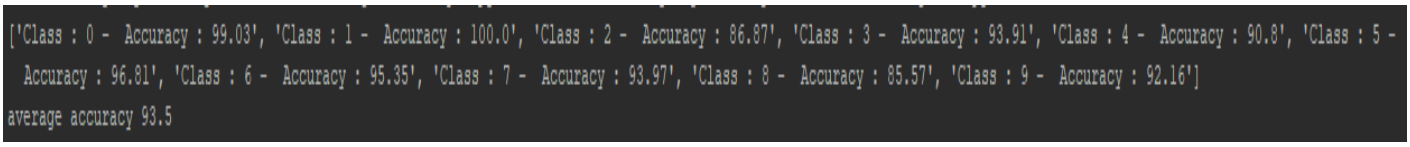

Figure 1 : Console output of Average accuracy and accuracy for each class using k=5

(c) For k = 1, change the number of training data points (30 to 10,000) to see the change in performance. Plot the average accuracy for 10 different dataset sizes. You may use command logspace in Matlab. In the plot, x-axis is for the number of training data and y-axis is for the accuracy.

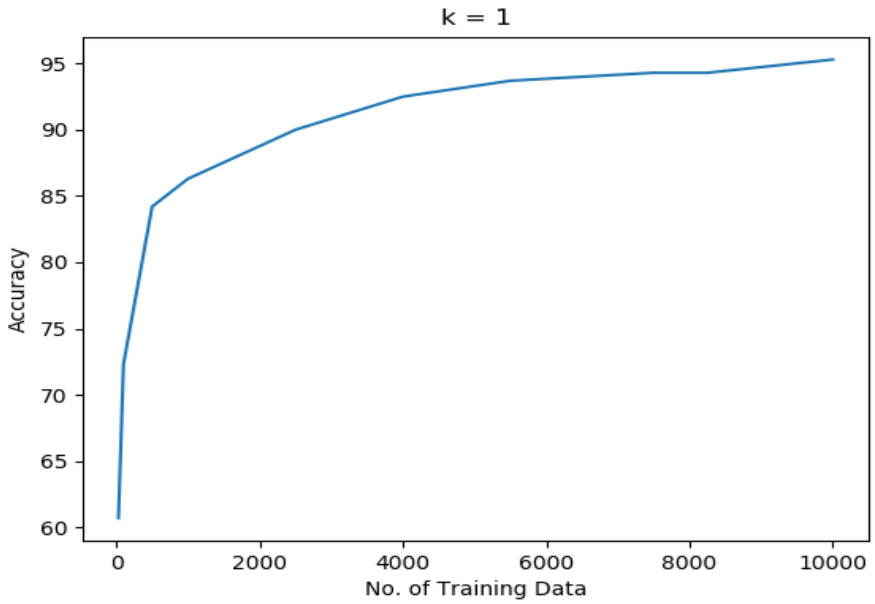

Figure 2: Variations in Accuracy with respect to No of training Data keeping k =1

(d) Show the effect of k on the accuracy. Make a plot similar to the above one with multiple colored curves on the top of each other (each for a particular k in [1 2 3 5 10].) You may use command legend in Matlab to name different colors.

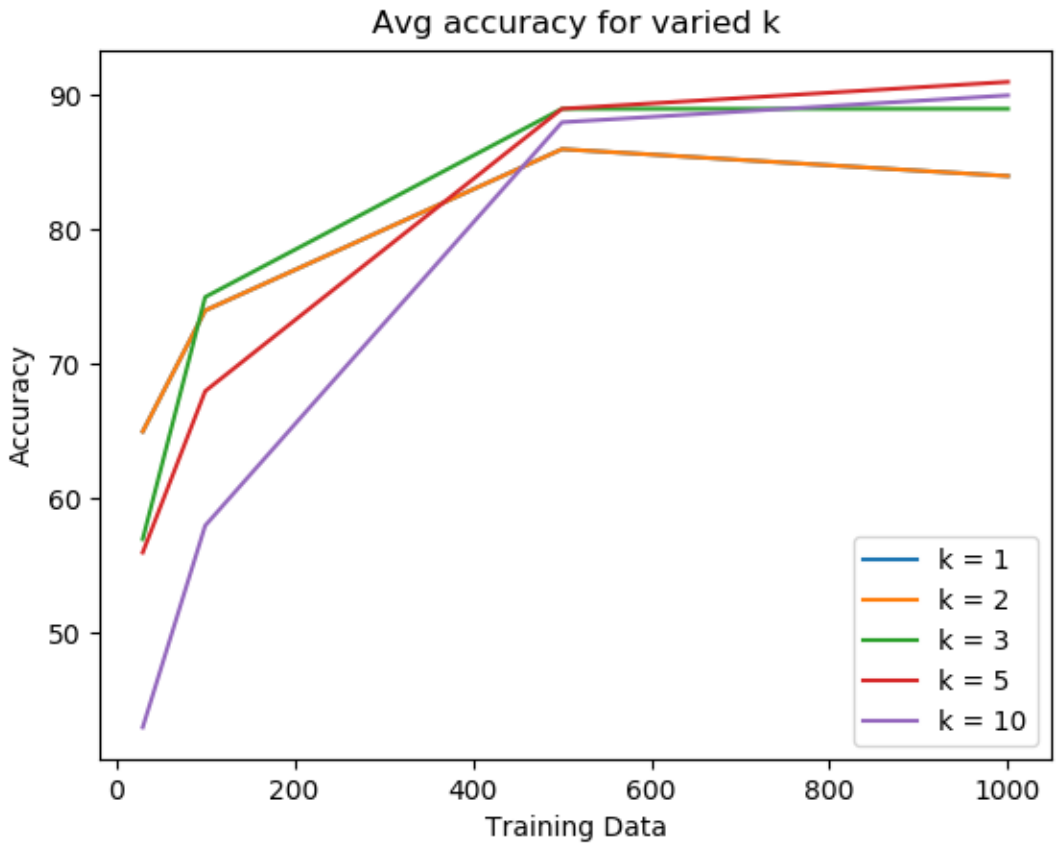

Figure 3: The plot above shows the effect of varied values of k on the accuracy of predictions calculated over varying number of training data.

(e) Choose the best k for 2,000 total training data by splitting the training data into two halves (the first for training and the second for validation). You may plot the average accuracy wrt k for this. Note that in this part, you should not use the test data. You may search for k in this list: [1 2 3 5 10].

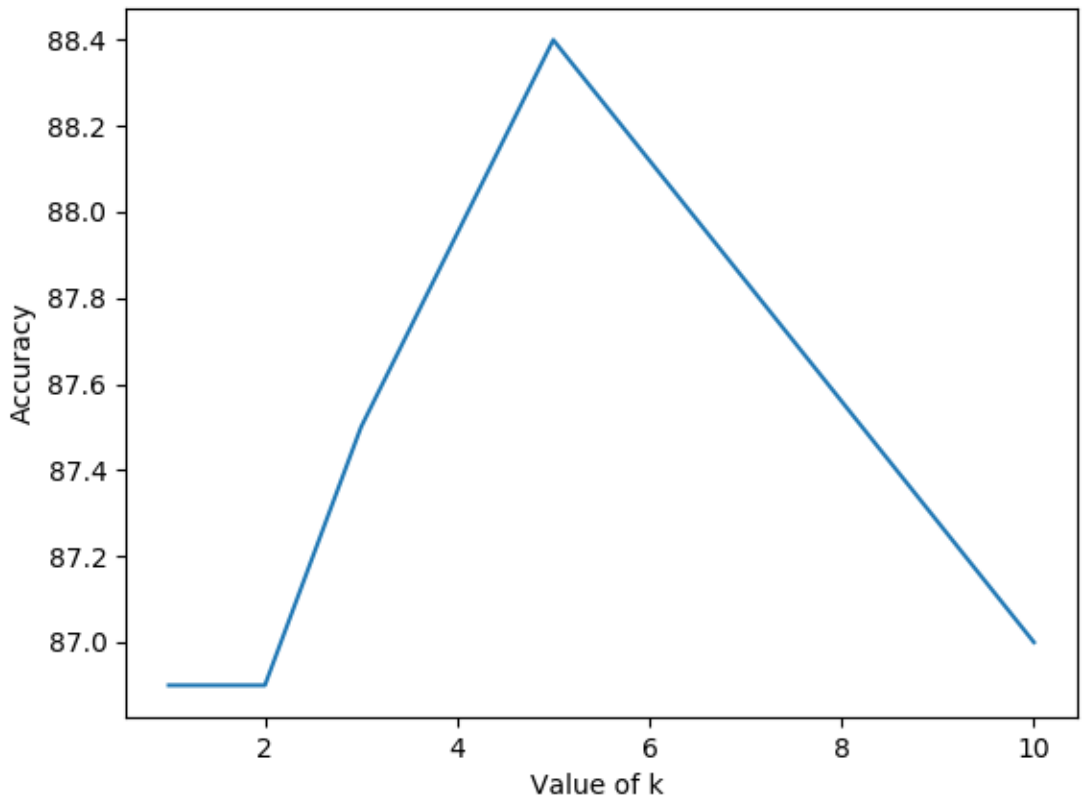

Figure 4: Plot of Varying Accuracy with respect to k for a fixed number of training and Validation data points(1000 each)