COMPSCI 589 Homework 1 - Spring 2024

Noshitha Padma Pratyusha Juttu

Notes on Submission

- I have made a different files for KNN and decision tree . I made the python files from python notebook.
- For KNN , please use below commands to run : python3 knn.py
- For Decision Trees, please use the below commands to run it python3 DecisionTree.py

Evaluating the *k*-NN Algorithm

(50 Points Total)

Q1.1 (10 Points) In the first graph, you should show the value of k on the horizontal axis, and on the vertical axis, the average accuracy of models trained over the *training set*, given that particular value of k. Also show, for each point in the graph, the corresponding standard deviation; you should do this by adding error bars to each point.

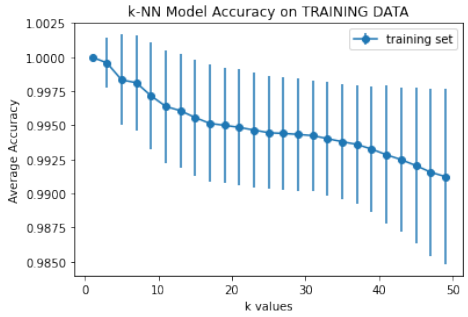

Figure 1: average accuracy of models trained over the *training set*,

Q1.2 (10 Points) In the second graph, you should show the value of k on the horizontal axis, and on the vertical axis, the average accuracy of models trained over the *testing set*, given that particular value of k. Also show, for each point in the graph, the corresponding standard deviation by adding error bars to the point.

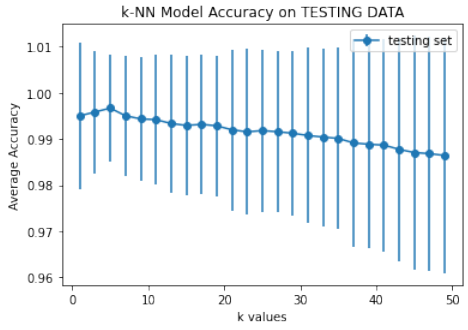

Figure 2: average accuracy of models trained over the *testing set*,

Q1.3 (8 Points) Explain intuitively why each of these curves looks the way they do. First, analyze the graph showing performance on the training set as a function of k. Why do you think the graph

looks like that? Next, analyze the graph showing performance on the testing set as a function of k. Why do you think the graph looks like that?

• As k increases, the model becomes less sensitive to noise and specific patterns in the training data, leading to a slight decrease in training accuracy. At the initial case of k=1, we do get 100% accuracy, and then remained approximately same for few k's and then dropped significantly and settled at a level after a certain k.

• As k increases, testing set accuracies initially improve because a larger dataset supports better predictions. However, past a certain point, the increasing number of neighbors leads to a decline, reflecting an optimal k value. Therefore, a slight reduction in testing accuracy occurs because of considering more neighbors in decision boundary formation.

Q1.4 (6 Points) We say that a model is *underfitting* when it performs poorly on the training data (and most likely on the testing data as well). We say that a model is *overfitting* when it performs well on training data but it does not generalize to new instances. Identify and report the ranges of values of k for which k-NN is underfitting, and ranges of values of k for which k-NN is overfitting.

• The model exhibits overfitting for k values among 1 to 5, showcasing strong performance on the training set but comparatively lower performance on the testing set.

• Conversely, for k values exceeding 13, the model experiences underfitting, indicated by a decline in performance on both the training and testing sets.

Q1.5 (6 Points) Based on the analyses made in the previous question, which value of k would you select if you were trying to fine-tune this algorithm so that it worked as well as possible in real life? Justify your answer.

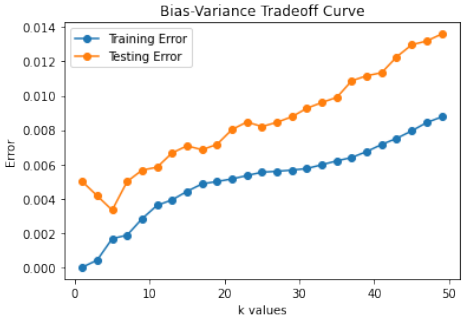

Figure 3: Training and testing error graph

- The recommendation for selecting a value of k will be within 3-10, where testing errors are minimized, and the model demonstrates a good balance between bias and variance. This choice would likely lead to a well-performing model in real-life scenarios by achieving effective generalization to new data.

Q1.6 (10 Points) In the experiments conducted earlier, you normalized the features before running k-NN. This is the appropriate procedure to ensure that all features are considered equally important when computing distances. Now, you will study the impact of *omitting* feature normalization on the performance of the algorithm. To accomplish this, you will repeat Q1.2 and create a graph depicting the average accuracy (and corresponding standard deviation) of k-NN as a function of k, when evaluated on the *testing set*. However, this time you will run the algorithm *without* first normalizing the features. This means that you will run k-NN directly on the instances present in the original dataset without performing any pre-processing normalization steps to ensure that all features lie in the same range/interval. Now ***(a)*** present the graph you created; ***(b)*** based on this graph, identify the best value of k; that is, the value of k that results in k-NN performing the best on the testing set; and ***(c)*** describe how the performance of this version of k-NN (without feature

normalization) compares with the performance of k-NN *with* feature normalization. Discuss intuitively the reasons why one may have performed better than the other.

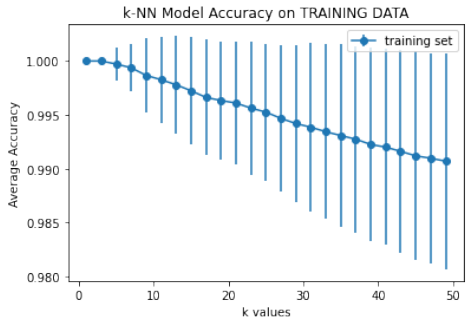

Figure 4: average accuracy of models trained over the training set without normalization

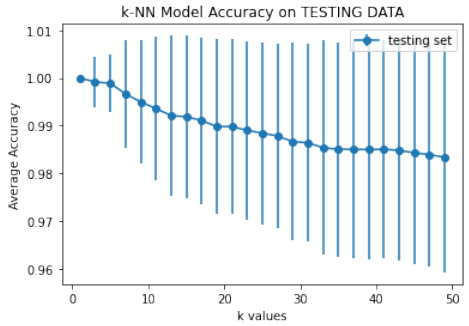

Figure 5: average accuracy of models trained over the training set without normalization

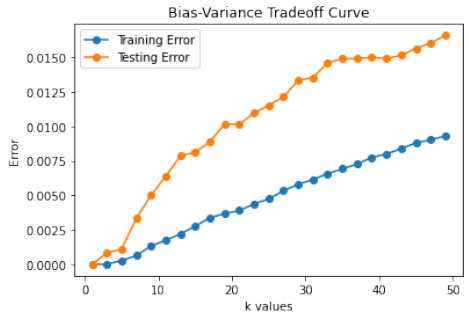

Figure 6: Training and testing error graph without normalization

- Training data accuracy begins at a perfect 1.0, diminishing gradually with increasing model complexity, indicating reduced accuracy. Similarly, testing data accuracy, starting at 1.0, experiences a slight initial decline with growing complexity and a more significant decrease as overfitting occurs with excessive model complexity.
- Training errors and testing errors both increase as model complexity grows, with testing errors showing a more pronounced increase. This reflects overfitting, where the model fits the training data too closely, leading to a decline in performance on new data.

Evaluating the Decision Tree Algorithm

(50 Points Total)

In this question, you will implement the Decision Tree algorithm, as presented in class, and evaluate it on the *1984 United States Congressional Voting* dataset. This dataset includes information about how each U.S. House of Representatives Congressperson voted on 16 key topics/laws. For each topic/law being considered, a congressperson may have voted yea, nay, or may not have voted. Each of the 16 attributes associated with a congressperson, thus, has 3 possible categorical values. The goal is to predict, based on the voting patterns of politicians (i.e., on how they voted in those 16 cases), whether they are Democrat (class/label 0) or Republican (class/label 1). **You can download the dataset here.**

Notice that this dataset contains 435 instances. Each instance is stored in a row of the CSV file. The first row of the file describes the name of each attribute. The attributes of each instance are stored in the first 16 columns of each row, and the corresponding class/label is stored in the last column of that row. For each experiment below, you should repeat the steps *(a)* through *(e)* described in the previous question—but this time, you will be using the Decision Tree algorithm rather than the k-NN algorithm. You should use the Information Gain criterion to decide whether an attribute should be used to split a node.

You will now construct two histograms. The first one will show the accuracy distribution of the Decision Tree algorithm when evaluated on the training set. The second one will show the accuracy distribution of the Decision Tree algorithm when evaluated on the testing set. You should train the algorithm 100 times using the methodology described above (i.e., shuffling the dataset, splitting the dataset into disjoint training and testing sets, computing its accuracy in each one, etc.). This process will result in 100 accuracy measurements for when the algorithm was evaluated over the training data, and 100 accuracy measurements for when the algorithm was evaluated over testing data.

Q2.1 (12 Points) In the first histogram, you should show the accuracy distribution when the algorithm is evaluated over *training data*. The horizontal axis should show different accuracy values, and the vertical axis should show the frequency with which that accuracy was observed while conducting these 100 experiments/training processes. The histogram should look like the one in Figure **??** (though the "shape" of the histogram you obtain may be different, of course). You should also report the mean accuracy and its standard deviation.

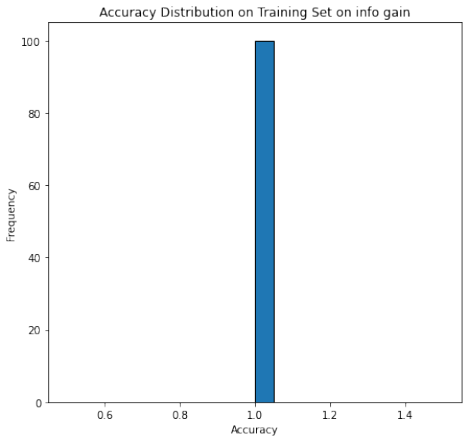

Figure 7: Histogram of the accuracy of the decision tree model on training set

- Mean Training Accuracy: 1.0
- Standard deviation training: 0.0

Q2.2 (12 Points) In the second histogram, you should show the accuracy distribution when the algorithm is evaluated over *testing data*. The horizontal axis should show different accuracy values, and the vertical axis should show the frequency with which that accuracy was observed while conducting these 100 experiments/training processes. You should also report the mean accuracy and its standard deviation.

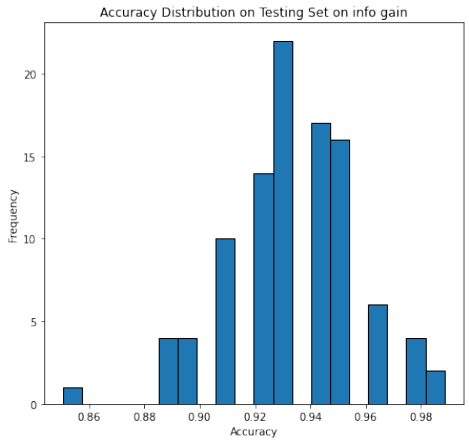

Figure 8: Histogram of the accuracy of the decision tree model on testing set

- Mean Testing Accuracy: 0.9313793103448276
- Standard deviation testing: 0.023129167362391723

Q2.3 (12 Points) Explain intuitively why each of these histograms looks the way they do. Is there more variance in one of the histograms? If so, why do you think that is the case? Does one histogram show higher average accuracy than the other? If so, why do you think that is the case?

- The absence of variance in training accuracies (standard deviation of 0.0) indicates that the model consistently memorizes and perfectly fits the training data in each iteration, leading to overfitting
- The higher average accuracy (mean of 1.0) in the training accuracy histogram is attributed to the model consistently achieving a perfect fit on the same dataset, while the lower average accuracy (mean of 0.9314) in the testing accuracy histogram reflects the model's performance variability on unseen data, encountering potential variations and challenges.

Q2.4 (8 Points) By comparing the two histograms, would you say that the Decision Trees algorithm, when used in this dataset, is underfitting, overfitting, or performing reasonably well? Explain your reasoning.

- The Decision Trees algorithm, as indicated by the histograms, is likely overfitting to the training data in this dataset. The consistently perfect training accuracy (mean of 1.0) and zero standard deviation suggest that the model has memorized the training data, achieving a perfect fit in every iteration.
- However, the lower but still high average testing accuracy (mean of 0.9314) and a small standard deviation (0.0231) indicate that the model's performance on unseen data is slightly variable but generally good.

- The overfitting is inferred from the perfect training accuracy, which suggests that the model has learned the training data too well, capturing noise and specificities that do not generalize to new data. The slight variability in testing accuracies further supports this, indicating that the model's performance varies across different test datasets.

Q2.5 (6 Points) In class, we discussed how Decision Trees might be non-robust. Is it possible to experimentally confirm this property/tendency via these experiments, by analyzing the histograms you generated and their corresponding average accuracies and standard deviations? Explain your reasoning.

- The experiments indicate that Decision Trees might not be very robust. The fact that the model consistently having the training data perfectly every time (with a mean accuracy of 1.0 and no variability) suggests it's memorizing details, making it less adaptable to new, different situations.
- While it still does decently well on new data (mean testing accuracy of 0.9314 with some variability), the big difference between its performance on familiar and unfamiliar data implies it might struggle with real-world variability.

[QE.1] Extra points (15 Points) Repeat the experiments Q2.1 to Q2.4, but now use the Gini criterion for node splitting, instead of the Information Gain criterion.

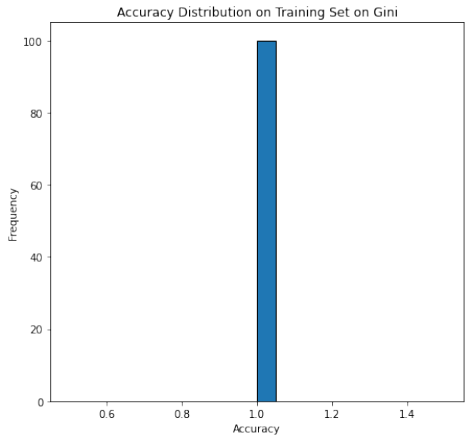

Figure 9: Histogram of the accuracy of the decision tree model on testing set on Gini

- Mean Training Accuracy: 1.0
- Mean Testing Accuracy: 0.9380459770114943
- Standard Deviation Training : 0.0
- Standard Deviation Testing : 0.02360635244609429
- It showcases potential overfitting tendency, as the model appears to have memorized the training data exceptionally well, leading to perfect training accuracy but possibly limiting its adaptability to new, diverse datasets. Despite this, the model still demonstrates good generalization to testing data, as indicated by the high average testing accuracy and relatively low standard deviation.

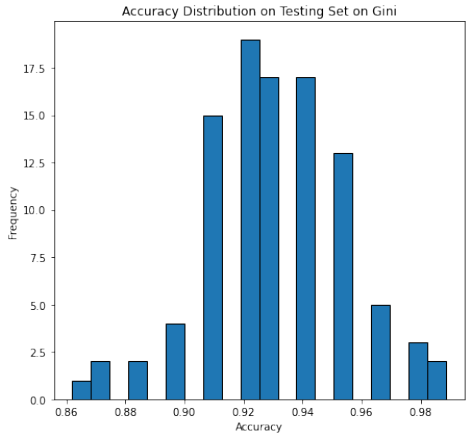

Figure 10: Histogram of the accuracy of the decision tree model on testing set on Gini

[QE.2] Extra points (15 Points) Repeat the experiments Q2.1 to Q2.4 but now use a simple heuristic to keep the tree from becoming too "deep"; i.e., to keep it from testing a (possibly) excessive number of attributes, which is known to often cause overfitting. To do this, use an *additional* stopping criterion: whenever more than 85% of the instances associated with a decision node belong to the same class, do not further split this node. Instead, replace it with a leaf node whose class prediction is the majority class within the corresponding instances. E.g., if 85% of the instances associated with a given decision node have the label/class *Democrat*, do not further split this node, and instead directly return the prediction *Democrat*.

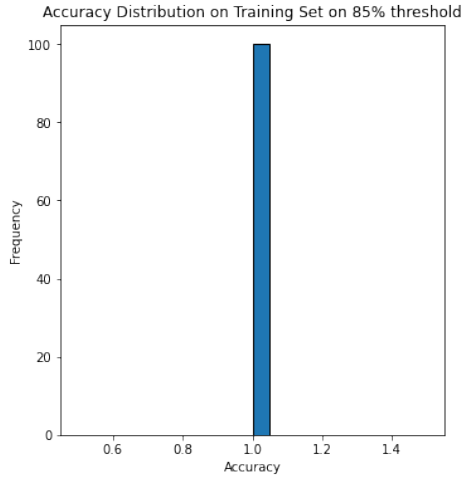

Figure 11: Histogram of the accuracy of the decision tree model on training set

- Mean Training Accuracy: 1.0
- Mean Testing Accuracy: 0.9331034482758621
- Standard Deviation Training : 0.0
- Standard Deviation Testing : 0.02456545313335337

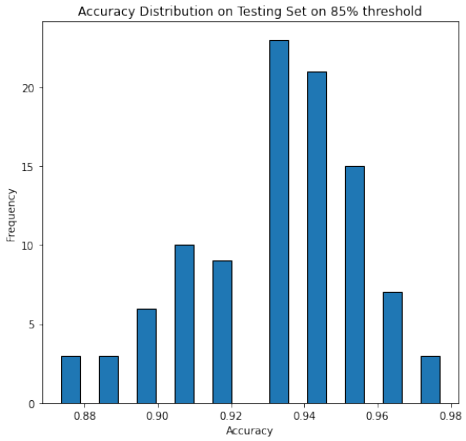

Figure 12: Histogram of the accuracy of the decision tree model on testing set

- The Decision Trees model continues to demonstrate a flawless fit on the training data, yielding perfect accuracy and zero variability. While its testing accuracy remains high (0.933) with a small standard deviation (0.0246), suggesting effective generalization to new data, the lack of improvement in training or testing metrics indicates a consistent performance without notable enhancements over previous models. The model's capacity to capture intricate details in the training data might hinder its adaptability to diverse real-world scenarios.