Assignment No 03: Decision tree

CSE-0408 Summer 2021

Nomrota Howlader Seme
Department of Computer Science and Engineering
State University of Bangladesh (SUB)
Dhaka, Bangladesh
email:nomrotahowladersema@gmail.com

***Abstract*—Decision Trees are considered to be one of the most popular approaches for representing classifiers. Researchers from various disciplines such as statistics, machine learning, pattern recognition, and Data Mining have dealt with the issue of growing a decision tree from available data. This paper presents an updated survey of current methods for constructing decision tree classifiers in a top-down manner. The chapter suggests a unified algorithmic framework for presenting these algorithms and describes various splitting criteria and pruning methodologies.**

***Index Terms*—python**

I. Introduction

A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.

II. Literature Review

compared the decision tree classification algorithm and developed the Weka method and it is based on choosing the file and selecting attributes to convert .csv file to flat file. The decision tree algorithms are presented and achieved a high rate of accuracy for classify the data into the correctly and incorrectly instances. Anshul Goyal et .al [7] background study a performance evaluation of Naïve bayes and J48 classification algorithms. J48 gives more classification accuracy for bank dataset having two values Male and Female. The result shows that j48 and Naïve Bayes gives better accuracy.

ShwetaKharya et.al [8] examined various data mining approaches that have been applied for breast cancer diagnosis and prognosis. Decision tree is search to be the best forecaster with 93.62on benchmark dataset and also on SEER data set. Abdullah H. Wahbeh et. al [9] had presented a performance evaluation of Naïve Bayes, J48 classification, sequential Minimal Optimization (SMO) classifier. Compared these three classification techniques based on two main aspects such as accuracy and execution time. In term of accuracy, results showed that the Naïve Bayes classifier achieved the highest accuracy, followed by SMO and J48 classifier. In term of execution time, results showed that the SMO model takes less execution time followed by the NB model and J48 classifier.

III. Proposed Methodology

The main objective of the study is to find the best decision tree based classification algorithms from five algorithms namely ID3, C4.5, C5.0, PART and Bagging CART. The classification algorithms are validated based on the performance measures such as precision, recall, f-measures, accuracy and kappa statistic

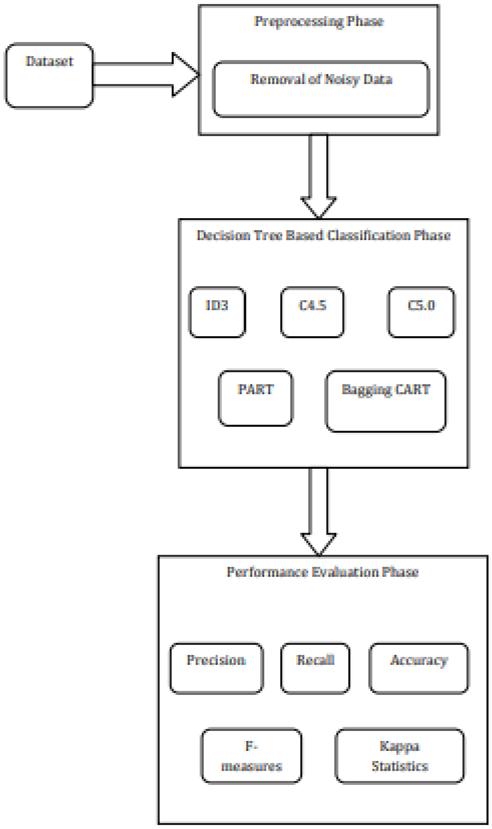


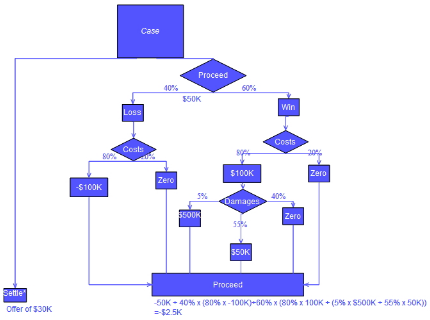
fig:Decision Tree based Classification Algorithms

IV. Decision rules

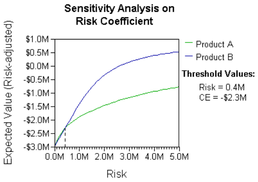
The decision tree can be linearized into decision rules,[2] where the outcome is the contents of the leaf node, and the conditions along the path form a conjunction in the if clause. In general, the rules have the form:

if condition1 and condition2 and condition3 then outcome. Decision rules can be generated by constructing association rules with the target variable on the right. They can also denote temporal or causal relations.

V. Decision tree using flowchart symbols



VI. Analysis example



The basic interpretation in this situation is that the company prefers B's risk and payoffs under realistic risk preference commonly used in operations research courses, is the distribution of lifeguards on beaches . The example describes two beaches with lifeguards to be distributed on each beach. There is maximum budget B that can be distributed among the two beaches (in total), and using a marginal returns table, analysts can decide how many lifeguards to allocate to each beach.

VII. Advantages and disadvantages

Among decision support tools, decision trees (and influence diagrams) have several advantages. Decision trees:

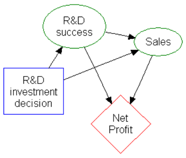
Are simple to understand and interpret. People are able to understand decision tree models after a brief explanation. Have value even with little hard data. Important insights can be generated based on experts describing a situation (its alternatives, probabilities, and costs) and their preferences for outcomes. Help determine worst, best and expected values for different scenarios. Use a white box model. If a given result is provided by a model. Can be combined with other decision techniques.

Disadvantages of decision trees:

They are unstable, meaning that a small change in the data can lead to a large change in the structure of the optimal decision tree. They are often relatively inaccurate. Many other predictors perform better with similar data. This can be remedied by replacing a single decision tree with a random forest of decision trees, but a random forest is not as easy to interpret as a single decision tree. For data including categorical variables with different number of levels, information gain in decision trees is biased in favor of those attributes with more levels.[7] Calculations can get very complex, particularly if many values are uncertain and/or if many outcomes are linked

VIII. Influence diagram

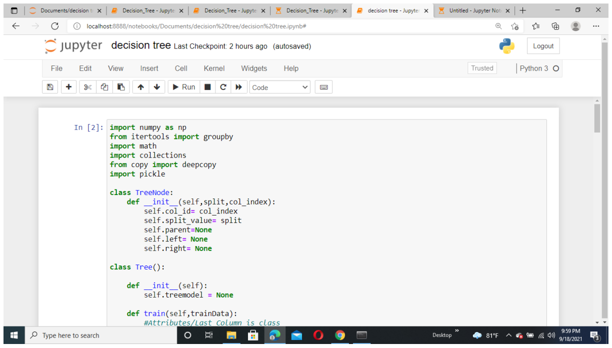
Much of the information in a decision tree can be represented more compactly as an influence diagram, focusing attention on the issues and relationships between events.



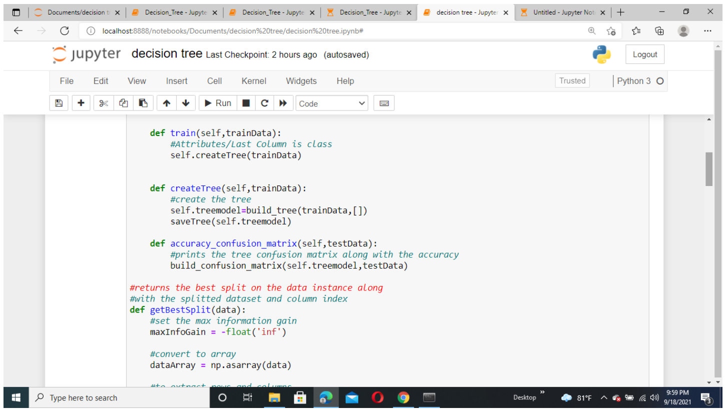
IX. Conclusion

Data mining is used to extract useful knowledge from large data repositories. Recently data mining techniques have enclosed every field in our life. Data mining have numerous algorithms to use for different purpose. In this paper discussed about the classification techniques. From this, the decision tree based classification algorithms namely ID3, C4.5, C5.0, PART and Bagging CART are used to perform classification process. The four data sets Iris, Balance scale, Contact lenses, Pima Indian Diabetes have been applied and performance is validated based on Accuracy (CA), Precision, Recall, F-Measure and Kappa Statistics.

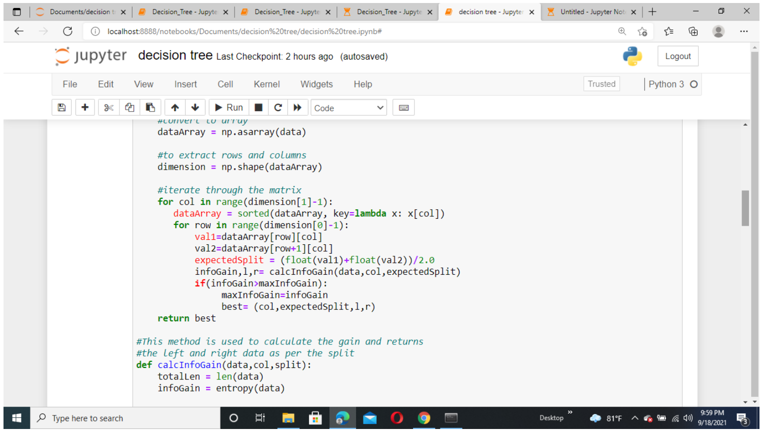
X. code



```
In [2]: import numpy as np
        from itertools import groupby
        import math
        import collections
        from copy import deepcopy
        import pickle

        class TreeNode:
            def __init__(self,split,col_index):
                self.col_id= col_index
                self.split_value= split
                self.parent=None
                self.left= None
                self.right= None

        class Tree():

            def __init__(self):
                self.treemodel = None

            def train(self,trainData):
                #Attributes/Last Column is class
```



```
    def train(self,trainData):
        #Attributes/Last Column is class
        self.createTree(trainData)


    def createTree(self,trainData):
        #create the tree
        self.treemodel=build_tree(trainData,[])
        saveTree(self.treemodel)

    def accuracy_confusion_matrix(self,testData):
        #prints the tree confusion matrix along with the accuracy
        build_confusion_matrix(self.treemodel,testData)

#returns the best split on the data instance along
#with the splitted dataset and column index
def getBestSplit(data):
    #set the max information gain
    maxInfoGain = -float('inf')

    #convert to array
    dataArray = np.asarray(data)

    #to extract rows and columns
```



```
    #convert to array
    dataArray = np.asarray(data)

    #to extract rows and columns
    dimension = np.shape(dataArray)

    #iterate through the matrix
    for col in range(dimension[1]-1):
       dataArray = sorted(dataArray, key=lambda x: x[col])
       for row in range(dimension[0]-1):
           val1=dataArray[row][col]
           val2=dataArray[row+1][col]
           expectedSplit = (float(val1)+float(val2))/2.0
           infoGain,l,r= calcInfoGain(data,col,expectedSplit)
           if(infoGain>maxInfoGain):
                maxInfoGain=infoGain
                best= (col,expectedSplit,l,r)
    return best

#This method is used to calculate the gain and returns
#the left and right data as per the split
def calcInfoGain(data,col,split):
    totalLen = len(data)
    infoGain = entropy(data)
```

ACKNOWLEDGMENT

I would like to thank my honourable **Khan Md. Hasib Sir** for his time, generosity and critical insights into this project.

REFERENCES

[1] Osmar R.; Zaine. (1999): Introduction to DataMining, CMPUT690 Principles of Knowledge Discovery in Databases, University of Alberta, Department of Computing Science. .

[2] Han Wu.; Shangqi Yang.; Zhangqin Hung.; Jian He.; Xiaoyi Wang. (2018): Type 2 diabetes mellitus prediction model based on data mining, Informatics in Medicine Unlocked

[3] K. Karimi and H.J. Hamilton (2011), ”Generation and Interpretation of Temporal Decision Rules”, International Journal of Computer Information Systems and Industrial Management Applications, Volume 3

[4] Wagner, Harvey M. (1 September 1975). Principles of Operations Research: With Applications to Managerial Decisions (2nd ed.). Englewood Cliffs, NJ: Prentice Hall. ISBN 9780137095926.

[5] Geetha Kashyap.; Ekta chauhan. (2016): Parametric comparisons of classification techniques in Data mining application, International journal of Engineering Development Research 4(2), pp. 1117-1123.

[6] Utgoff, P. E. (1989). Incremental induction of decision trees. Machine learning, 4(2), 161–186 .

[7] Deng,H.; Runger, G.; Tuv, E. (2011). Bias of importance measures for multi-valued attributes and solutions. Proceedings of the 21st International Conference on Artificial Neural Networks (ICANN).

Assignment No 04:K-Nearest Neighbors

CSE-0408 Summer 2021, FINAL EXAM

Nomrota Howlader Sema
Department of Computer Science and Engineering
State University of Bangladesh (SUB)
Dhaka, Bangladesh
email:nomrotahowladersema@gmail.com

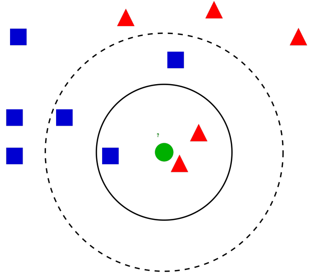
***Abstract*—The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.KNN is a lazy learning, non-parametric algorithm. It uses data with several classes to predict the classification of the new sample point. KNN is non-parametric since it doesn't make any assumptions on the data being studied, i.e., the model is distributed from the data.**

***Index Terms*—k-nn,python**

I. Introduction

k-nearest neighbors algorithm (k-NN) is a non-parametric classification method first developed by Evelyn Fix and Joseph Hodges in 1951,[1] and later expanded by Thomas Cover.[2] It is used for classification and regression. In both cases, the input consists of the k closest training examples in data set. The output depends on whether k-NN is used for classification or regression:

In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor. In k-NN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors. k-NN is a type of classification where the function is only approximated locally and all computation is deferred until function evaluation. Since this algorithm relies on distance for classification, if the features represent different physical units or come in vastly different scales then normalizing the training data can improve its accuracy dramatically.



II. Literature Review

Thomas Cover and Peter Hart proved an upper bound error rate with multiclass kNN classifications in 1967 (twice the Bayes error rate). Research paper: Nearest neighbor pattern classification (1967) Cover Hart's discovery paved the way for a number of new rejection studies such as:

In 1970: Edward Hellman examined “the (k,k?) nearest neighbor rule with a reject option“. In 1975: Fukunaga and Hostetler made refinements with respect Bayes error rate In 1976: Dudani; in 1978 Bailey and Jain published distance weighted approaches
In 1983: Adam Jozwik introduced: A learning scheme for a fuzzy k-NN rule
In 1985: James Keller et al developed FKNN (Fuzzy kNN): A fuzzy k-nearest neighbor algorithm
In 2000: Bermejo and Cabestany published: Adaptive soft k-nearest-neighbour classifiers.
There has been many improvements to kNNs since those days and new approaches continue to emerge to this day KNN algorithm applies the birds of a feather. It assumes that similar things are near to each other; that is, they are nearby.

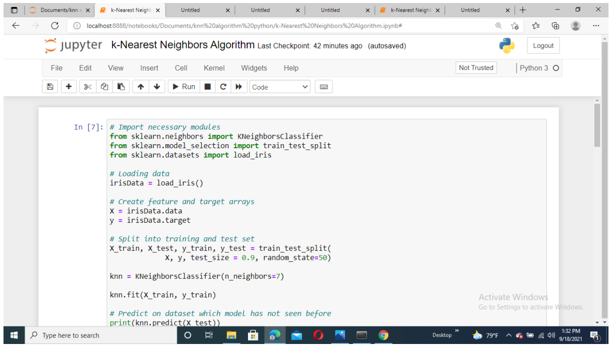
III. Proposed Methodology

we classified a data set including large number of unlabeled data, if only utilize the few training examples available, then we can't obtain a high accuracy classifier with inadequate training examples; if we want to obtain a classifier of high performance, then labeling the unlabeled data is necessary, but labeling vast unlabeled data wastes time and consumes strength. In this paper, we propose a novel method which uses SVM cooperated with KNN for classification based on semi supervised learning theory. The general model is depicted as above (See Figure 2). To begin with, we construct a weaker classifier SVM according to the few training examples available, then using the weaker SVM classifies the remaining large number of unlabeled data in the data set, picking out n examples belonging to each class around the decision boundary by calculating Euclidean distance in the feature space, because the examples located around the boundary are easy to be misclassified, but they are likely to be the support vectors, we call them boundary vectors, so picking out these boundary vectors whose labels are fuzzy labeled by the weaker classifier SVM. Secondly we recognize these boundary vectors as testing set while recognize initial training examples as training set, use KNN method to classify them and recognize the results as the labels for boundary vectors. In the end, we put these boundary vectors and their labels into initial training set to enlarge the number of the training examples then retrain a SVM, iteratively until the number of the training examples is m times of the whole data set. The experimental results on three UCI data sets indicate that the final classifier SVM has significant improvement on accuracy.

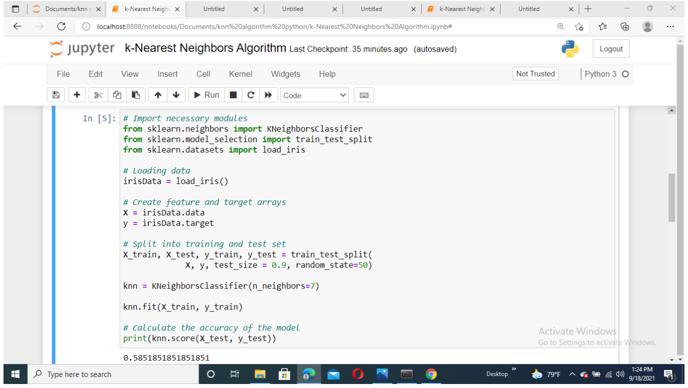
IV. Conclusion

Every learning algorithm will tend to suit some problem types better than others, and will typically have many different parameters and configurations to be adjusted before achieving optimal performance on a dataset, AdaBoost (with decision trees as the weak learners) is often referred to as the best outofthe-box classifier. When used with decision tree learning, information gathered at each stage of the AdaBoost algorithm about the relative ’hardness’ of each training sample is fed into the tree growing algorithm such that later trees tend to focus on harder-to-classify examples. The supervised machine learning algorithms such as decision trees and support vector machine are capable enough to deal with big data mining tasks. Even though the algorithms efficiency considerably improving there is a need for adaptive boosting process required in order to increase the predictive accuracy much more In future, what you bring in your project and the idea of your work.

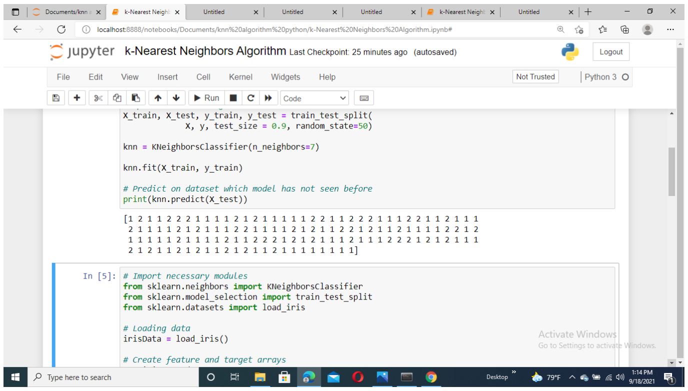
V. CODE


jupyter k-Nearest Neighbors Algorithm Last Checkpoint: 42 minutes ago (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Code

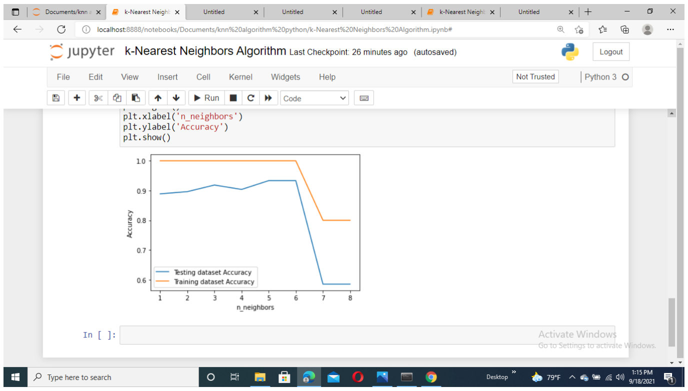
```
In [7]: # Import necessary modules
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.model_selection import train_test_split
        from sklearn.datasets import load_iris

        # Loading data
        irisData = load_iris()

        # Create feature and target arrays
        X = irisData.data
        y = irisData.target

        # Split into training and test set
        X_train, X_test, y_train, y_test = train_test_split(
                     X, y, test_size = 0.9, random_state=50)

        knn = KNeighborsClassifier(n_neighbors=7)

        knn.fit(X_train, y_train)

        # Predict on dataset which model has not seen before
        print(knn.predict(X_test))
```


jupyter k-Nearest Neighbors Algorithm Last Checkpoint: 35 minutes ago (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Code

```
In [5]: # Import necessary modules
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.model_selection import train_test_split
        from sklearn.datasets import load_iris

        # Loading data
        irisData = load_iris()

        # Create feature and target arrays
        X = irisData.data
        y = irisData.target

        # Split into training and test set
        X_train, X_test, y_train, y_test = train_test_split(
                     X, y, test_size = 0.9, random_state=50)

        knn = KNeighborsClassifier(n_neighbors=7)

        knn.fit(X_train, y_train)

        # Calculate the accuracy of the model
        print(knn.score(X_test, y_test))

        0.5851851851851851
```


jupyter k-Nearest Neighbors Algorithm Last Checkpoint: 25 minutes ago (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Code

```
        X_train, X_test, y_train, y_test = train_test_split(
                     X, y, test_size = 0.9, random_state=50)

        knn = KNeighborsClassifier(n_neighbors=7)

        knn.fit(X_train, y_train)

        # Predict on dataset which model has not seen before
        print(knn.predict(X_test))

        [1 2 1 1 2 2 2 1 1 1 1 2 1 2 1 1 1 1 1 2 2 1 1 2 2 2 1 1 1 2 2 1 1 2 1 1 1
         2 1 1 1 1 2 1 2 1 1 1 2 2 1 1 1 1 2 1 2 1 1 2 2 1 2 1 1 2 1 1 1 1 2 2 1 2
         1 1 1 1 1 2 1 1 1 1 2 1 1 2 2 2 1 2 1 2 1 1 1 2 1 1 1 2 2 2 1 2 1 2 1 1 1
         2 1 2 1 1 2 1 2 1 1 2 1 2 1 1 2 1 1 1 1 1 1 1 1]

In [5]: # Import necessary modules
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.model_selection import train_test_split
        from sklearn.datasets import load_iris

        # Loading data
        irisData = load_iris()

        # Create feature and target arrays
```



```
        plt.xlabel('n_neighbors')
        plt.ylabel('Accuracy')
        plt.show()

In [ ]:
```

Acknowledgment

I would like to thank my honourable **Khan Md. Hasib Sir** for his time, generosity and critical insights into this project.

References

[1] X.J. Zhu. Semi-supervised learning literature survey[R]. Technical Report 1530, Department of Computer Sciences, University of Wisconsin at Madison, Madison, WI, December, 2007.

[2] Everitt, Brian S.; Landau, Sabine; Leese, Morven; and Stahl, Daniel (2011) ”Miscellaneous Clustering Methods”, in Cluster Analysis, 5th Edition, John Wiley Sons, Ltd., Chichester, UK

[3] Fung, G., and Mangasarian, O. Semi-supervised support vector machines for unlabeled data classification (Technical Report 99-05). Data Mining Institute, University of Wisconsin Madison, 1999

[4] Hastie, Trevor. (2001). The elements of statistical learning : data mining, inference, and prediction : with 200 full-color illustrations. Tibshirani, Robert., Friedman, J. H. (Jerome H.). New York: Springer.

[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.