Comparative Analyses of Statistical Classification Methods

By

Zachary Barnhart

A Thesis Submitted to the Department of Mathematics
& the University Honors College
In Partial Fulfillment of the Requirements
for the University and Departmental Honors Baccalaureate
Millersville University
Defended on April 23, 2025

Comparative Analyses of Statistical Classification Methods

Thesis Approved By:

Patrick Stewart, Ph.D

Thesis Advisor

Jingnan Xie, Ph.D

Kevin S. Robinson, Ph.D

ABSTRACT

In today's computer-driven world, there are many types of data that need to be parsed through. Because manually processing information can be time-consuming and expensive, we need efficient methods to process a large amount of information. One application of processing information is classifying observations into categories based on specific features. With classification methods, we train a computer to accept specific input predictors, and the output is the class the computer thinks those observations belong in.

In this thesis, we will examine the performance of several classification methods under various scenarios. We compare the performance of each method on simulated mixtures of distributions by using the misclassification rate. Furthermore, we extend our findings from the simulation studies by performing each classification method on several real-world datasets.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Dr. Stewart, for his invaluable guidance and support throughout this process. From running simulations to assisting with the writing process, his dedication and expertise have been instrumental in the completion of this thesis. I am incredibly thankful for all the time and effort he has devoted. I also extend my sincere appreciation to Dr. Xie and Dr. Robinson for serving on my committee and providing valuable feedback. I am grateful to the entire Millersville University Mathematics Department. This thesis would not have been possible without everyone within this department who have contributed to my growth, both academically and personally. I would like to thank the Millersville University Honors College for offering me the opportunity to complete a thesis. Finally, I am deeply thankful to my family for their unwavering support and encouragement throughout my entire life. I would not be in this position today without them.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

Introduction

The idea of deciding which group an object belongs to is a rather general problem in the world today. Data analysis often involves predicting a categorical outcome from a set of predictor variables, which is known as classification. Problems involving classification can appear in many disciplines including business, medicine, astrophysics, health care, genetics, and social sciences (Hastie et al., 2021).

1.1 Classification Overview

Classification falls under the larger umbrella of statistical machine learning which is sometimes referred to as simply statistical learning. Statistical machine learning is at the intersection of statistics and computer science, which includes data mining and artificial intelligence, and is a broad topic that refers to a variety of ways of understanding data (Makridakis et al., 2018). Computer science seeks to build or program machines to solve problems while statistics looks at what can be inferred from a data set. Both of these fundamental questions are combined to form their own discipline, statistical machine learning (Mohammed et al., 2017). Statistical machine learning uses data and complex algorithms to convert an input to an output by recognizing patterns in data (Prevos, 2019). Statistical learning is the key technology for extracting information from data (Sugiyama, 2016). It is different from traditional prediction methods because machines themselves create the equations, not the researcher (Prevos, 2019). These methods have gained popularity over time as interest in artificial intelligence has risen (Makridakis et al., 2018). With the recent growth in artificial intelligence chat bots, like ChatGPT, we expect interest in this area to continue to grow. Statistical learning plays a large role in many areas of science, finance, and industry (Hastie et al., 2009).

Statistical learning can be broken into three branches: supervised learning, unsupervised learning, and reinforcement learning (Prevos, 2019). Supervised learning is similar to how humans learn

to classify the world; as children, our parents identify different objects, like a cat, and over time we learn to classify the objects based on how the objects look or act (Prevos, 2019). Supervised learning involves labeled data which means there is an outcome or response variable (Mohammed et al., 2017). In general, supervised learning begins with a set of observations containing data for both predictor and outcome variables, and a model is built to predict or estimate the specified output (Hastie et al., 2021). Classification, the topic of this thesis, and regression, a widely used statistical method, both belong to the category of supervised learning (Prevos, 2019). While classification involves classifying data into groups, regression looks to find relationships between data (Prevos, 2019). The opposite of supervised learning is unsupervised learning. Supervised learning can be thought of as learning with a teacher while unsupervised can be thought of as learning without one. With unsupervised learning, there is no specified output, only inputs, but relationships and structure can still be learned from the data (Hastie et al., 2021). Clustering is an example of unsupervised learning (Hastie et al., 2009). The goal of clustering is to group similar data points together in a cluster (Hastie et al., 2009). Reinforcement learning, the final branch, learns by trial and error rather than direct answers (Sugiyama, 2016). The model is not given the labels during training and is instead given feedback throughout the training. Reinforcement learning can use both supervised and unsupervised learning methods (Sugiyama, 2016).

The focus of this thesis will be on the supervised learning technique, classification. While regression looks to predict a continuous variable, classification is the task of assigning objects to one of a set of predefined groups (Tan et al., 2006). Classification can be used as an explanatory tool to differentiate between objects of different classes or used to predict the label of objects where the label is unknown (Tan et al., 2006). Classification involves learning a target function which maps each object to a predefined class label. The target function is commonly known as a classification model (Tan et al., 2006). There are many different classification methods. Each classification method uses a learning algorithm to develop a model which best relates the attribute set to the class labels (Tan et al., 2006). Predicting a qualitative response for an observation is referred to as classifying the observation (Hastie et al., 2021). In general, the goal for classification

is to predict an outcome category based on a set of features (Hastie et al., 2009).

1.2 Classification Problems

Classification problems are more common than regression problems in the real world (Hastie et al., 2021). The models built to solve these problems are used to make real-world decisions that can have significant consequences (Kabacoff, 2022). There are many examples of classification problems which are common in every day life. Each simple example below is a categorical outcome based on a set of predictors, also called features (Kabacoff, 2022).

- Identifying whether an emergency room patient is having a heart attack based on their symptoms and vital signs (Kabacoff, 2022).
- Predicting whether a patient hospitalized due to a heart attack will have a second heart attack based on demographics, diet, and clinical measurements (Hastie et al., 2009).
- Diagnosing diseases based on a doctor's years of medical training and practice by assigning the patient to a disease group based on the patient's symptoms (James, 1985).
- Figuring out which DNA mutations are disease causing and which are not for given DNA sequence data for a number of patients with or without a specified disease (Hastie et al., 2021).
- Predicting whether an individual will repay a loan based on their demographics and financial history (Kabacoff, 2022).
- Deciding guilty versus innocent (James, 1985).
- Predicting the gender of students from a general education course based on weight, height, birth date, last name, and first name (Dean, 2014).
- Determining whether an email is spam based on key words, images, and header information (Kabacoff, 2022).

- Identifying whether a transaction in an online banking service is fraudulent or not based on the user's IP address, past transactions, and others (Hastie et al., 2021).
- Forecasting whether it will rain today (James, 1985).
- Identifying plant species using a set of measurements that allow them to assign any specimen to the correct species (James, 1985).
- Detecting animal liver cancer after a suspected carcinogen, aflatoxin, is fed to the animals at different levels in a study (Ott and Longnecker, 2016).

A more complex example of classification is when an algorithm is presented with thousands of images, with some containing a specific object and some not. The algorithm is then taught how to recognize the specific object based on these images (Prevos, 2019). This can also be used in handwriting analysis.

Another specific example is a dataset which contains stock market data over a five-year period. The goal is to predict whether it will increase or decrease for a given day using data from the previous five days (Hastie et al., 2021). This is a difficult classification problem due to the many variables present in the stock market.

Statistical learning models can provide accurate fault detection diagnosis information for engineering better than model-based approaches. This can be used for system recovery or system decision making to ensure reliable robotic operation (Zhang et al., 2022).

The US Postal Service began using machine learning in the 1960s by using machines to read addresses to sort letters. Optical character recognition (OCR) technology was used to correctly read addresses. The OCR technology is part of pattern recognition, a branch of machine learning. Computer-aided diagnosis uses pattern recognition algorithms to assist doctors in making medical diagnoses too. Medical images in the form of X-rays, MRI, or ultrasound are data describing a patient's condition. Computer-aided diagnosis searches for suspicious structures in the medical images using supervised learning. The algorithm is given thousands of labeled images and then

expected to classify new medical images correctly. Computer-aided diagnosis can be helpful in detecting cancer and other diseases (Mohammed et al., 2017).

1.3 Tackling Classification Problems

The goal of classification is to find an accurate method for deciding which category to place new cases (Kabacoff, 2022). Some common steps for classification problems include data splitting, data preprocessing, model building, model comparisons, and model release (Kabacoff, 2022). Data preprocessing involves dealing with missing data and combining highly correlated variables. All supervised learning techniques, including classification, begin with a set of observations containing data for both predictor and outcome variables. The dataset is then split into a training sample and a testing sample. The training set and testing set contain the same predictor variables and response variable (Zhang, 2016). The goal is to create a training sample that is representative of the population. A predictive model, or learner, is built using the training data (Hastie et al., 2009). After the predictive model is created using the training data, it is tested for accuracy using the testing sample. Both samples are necessary in order to maximize prediction for a given dataset (Kabacoff, 2022). Typically, a larger portion of the data is used for training the model (Mohammed et al., 2017). Traditionally, somewhere between seventy and eighty percent is used in the training set. This leaves between twenty and thirty percent in the testing set. The testing set acts as data that the model has never seen before. Classification predictive models can range from relatively simple to extremely complex. Models often have hyperparameters, which are parameters that control the learning process and must be tuned for the specific problem (Kabacoff, 2022). Once an effective model is created, it can be applied to datasets where only the predictor variables are present (Kabacoff, 2022). Model release involves using it to make future predictions. The model should then be continuously evaluated for effectiveness over time (Kabacoff, 2022). Table 1 outlines the basic steps for a classification problem.

Classification models can be built and analyzed using the statistical software R (R Core Team, 2023). The `sample.split()` function in the caTools library can be used to split the data into

Step	Description
1.	**Data Preprocessing**: Clean and transform raw, labeled data.
2.	**Data Splitting**: Divide data into training and test sets.
3.	**Model Building**: Select and train a classification algorithm on the training set.
4.	**Model Comparison**: Evaluate the model's performance using appropriate metrics and compare different models.
5.	**Model Release**: Deploy the best-performing model for use on future data and continue to evaluate it for effectiveness over time. Refine the model if necessary.

Table 1: Common Steps for Classification Problems

the training and testing sets. The function requires a vector of data labels (the response variable) and a split ratio (a value between 0 and 1). Using this function, the proportion of a class of the response variable in the training set matches the proportion in the testing set and the proportion in the overall dataset. The functions for running specific classification methods in R will be discussed in Chapter II.

1.3.1 Evaluation Metrics

After a model is built, it is important to evaluate the performance of the model to determine its usefulness. The performance of any classification model can easily be evaluated since the actual

category of each observation is known for the testing dataset (Zhang, 2016).

Evaluating the performance of a model is focused more on how the model predicts the test set because the goal of classification is to be able to predict future, previously unseen data correctly (Tan et al., 2006). The results from a model are traditionally recorded in a confusion matrix. An example confusion matrix for a two-class problem is Table 2.

		Actual Class	
		Positive	Negative
Predicted Class	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

Table 2: Confusion Matrix Example

It can be seen that the points that fall into the true positive and true negative categories have been correctly classified, while the points in the false positive and false negative have been incorrectly classified.

A performance metric such as accuracy (Equation 1.3.1) or error rate (Equation 1.3.2) can be used as a single number summary of the model's performance. Most models seek to have the highest accuracy, or lowest error rate, on the test set (Tan et al., 2006).

Precision (Equation 1.3.3) and recall (Equation 1.3.4) are used where the classification of a single class is more important than all of the other classes (Tan et al., 2006). The goal is to have a high recall and precision values on the test set.

Sensitivity (Equation 1.3.4) and specificity (Equation 1.3.5) are commonly used metrics for evaluating how a model works on a two-class problem (Tan et al., 2006). In larger class problems, sensitivity and specificity can be calculated for each individual class. The goal is to have a high specificity and sensitivity value, but there is typically a trade off associated between them (Kabacoff, 2022).

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \tag{1.3.1}$$

$$\text{Error Rate (Misclassification Rate)} = \frac{FP + FN}{TP + FP + FN + TN} \tag{1.3.2}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1.3.3}$$

$$\text{True Positive Rate (Sensitivity or Recall)} = \frac{TP}{TP + FN} \tag{1.3.4}$$

$$\text{True Negative Rate (Specificity)} = \frac{TN}{TN + FP} \tag{1.3.5}$$

Each of the evaluation metrics take on values from 0 to 1. Error rate is frequently referred to as misclassification rate.

For problems of more than two classes, the accuracy and error rate are still easy to calculate. The accuracy can be computed by summing the main diagonal and dividing by the total records in the confusion matrix. The error rate can be computed by summing every number not on the main diagonal and dividing by the total records in the confusion matrix.

Choosing the best model is also an important part of the process. Comparison of models involves comparing the evaluation metrics and choosing the model with the lowest (or highest) values. Occam's Razor says that given two models with same generalization errors, the simpler model should be chosen (Tan et al., 2006). Many classification methods require large amounts of storage to be used for future use, so choosing the simpler model may be more efficient. However, there is no single answer on how to evaluate a model or choose the best one. Each classiification problem may require different analysis.

1.3.2 Potential Issues

There are several general issues that can arise in classification problems.

The first issue is the choice of variables. Classification is best used for data with binary or nominal categories. It is less effective for ordinal categories because the model does not consider the order of the categories (Tan et al., 2006).

Additionally, the consequences of misclassification, an incorrect classification of a data point, depend on the problem. For example, it is likely more serious to classify a person as not having a second heart attack when they actually will than vice-versa (Hastie et al., 2009).

Another challenge is the curse of dimensionality, which arises when a dataset contains too many features. As the number of features grows, misclassifications tend to increase because every feature influences the classification process, even if it has little or no actual relevance to the data point. This issue can be resolved by removing some of the features in the dataset before running the classification method.

Another important consideration for classification is the size of the training set. When the set is too large, the algorithm will find patterns that do not exist in the population. On the other hand, a training set that is too small will not be able to find any patterns (Prevos, 2019). It is important to find the right balance so that the training set is representative of the entire population.

There are two basic types of errors in classification models: training errors and generalization errors. Training error is the number of misclassifications on the training data set. Generalization error is the expected error on new records. Typically, the number of misclassifications on the testing set is used to estimate the generalization error. A good classification model not only fits the training data set, but also accurately classifies records it has never seen before (Tan et al., 2006).

Model overfitting, perhaps the biggest issue in classification problems, is when the training error is low and the generalization error is high (Tan et al., 2006). Overfitting is typically described as the model fits the training set too well and is unable to classify new observations. In other words, the training misclassification rate is low, but the testing misclassification rate is high which means that new cases are not classified well (Kabacoff, 2022). Overfitting can be a result of noise from outliers or from fitting the training set too well. Models built on a small number of training samples are also prone to overfitting due to a lack of representation in the training set.

Class imbalance is another major issue in classification problems (Tan et al., 2006). Class imbalance is when one of the classes in the response variable has many more records than the other or others. For example, an automated inspection system that monitors products coming off

an assembly line will likely find fewer defective products than non-defective products. Another example is credit card fraud detection where fraudulent transactions are much less common than legitimate transactions. Despite this imbalance, it may be even more important to correctly classify the smaller class than the larger class (Tan et al., 2006). The accuracy metric may not be as helpful in analyzing the performance of a model in these instances. For example, if one class has 99% of the data, classifying every point into that class would give an accuracy of 99%. Some of the evaluation metrics discussed in the previous section would be more useful to use when evaluating models than accuracy.

CHAPTER II

Classification Methods

In this chapter, we will discuss the following classification methods: k-nearest neighbors, logistic regression, classification trees, and random forest. Each one meets the criterion for supervised learning and classification. There are many other classification methods which will not be discussed in this thesis.

2.1 k-Nearest Neighbors

The k-nearest neighbors method, commonly abbreviated as KNN, is a nonparametric classification method that utilizes the observations in the training set closest to each point to form the predicted responses (Hastie et al., 2009). The user-defined input is the k closest neighbors in the training set to a given observation, and the output is the class membership of the observation based on the majority vote of its k-nearest neighbors (Mohammed et al., 2017). Although it is visually easy to show how KNN works with two predictors, it can be used on a dataset with any number of predictors (Zhang, 2016). Figure 1 displays a two-predictor, two-class classification problem with each color being a different class using $k = 7$ (Hastie et al., 2009).

The KNN method is memory based because it stores the entire training set instead of a model. Given a point x_0, the k training points closest in distance to x_0 are found, and then x_0 is classified by the majority vote among those k neighbors. Typically k is chosen such that there are no ties in the majority vote; however, this approach is not always feasible. In such cases, ties are traditionally broken at random (Hastie et al., 2009). Sometimes, each neighbor is given a weight so that nearer neighbors contribute more than more distant ones. A common weighing scheme is $\frac{1}{d}$, where d is the distance from the observation (Mohammed et al., 2017). Giving each neighbor a weight can reduce the chances of a tie occurring. Table 3 outlines the basic steps for the k-nearest neighbor method (Zhang et al., 2022).

There are many important considerations when it comes to implementing KNN. One impor-

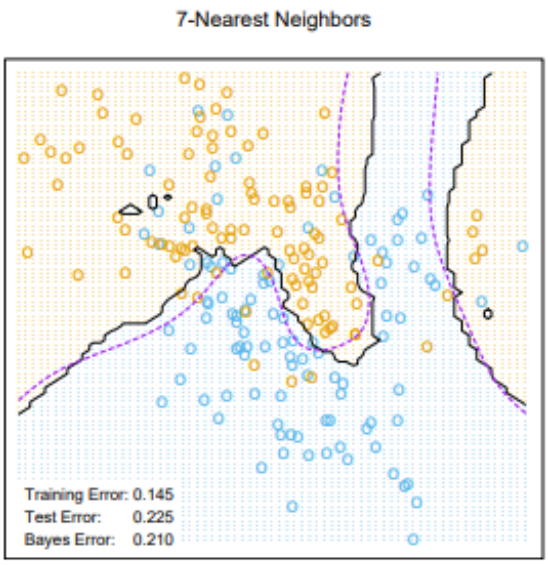


Figure 1: Example KNN Graph

Steps of the KNN Algorithm
1. Calculate the distance between the record to be classified and the records in the training set.
2. Sort the records in ascending order based on their distance.
3. Select the k records with the smallest distance.
4. Count the frequency of each category among the k nearest records.
5. Assign the record to the category with the highest frequency.

Table 3: k-Nearest Neighbors Algorithm Steps

tant concept is the distance metric used to define which k points are closest to x_0 Zhang (2016). The Minkowski distance is a general distance formula described by Equation 2.1.1, where r is a parameter and n is the number of predictors for two data point vectors, **x** and **y** (Tan et al., 2006).

$$d(\mathbf{x}, \mathbf{y}) = (\sum_{i=1}^{n} |x_i - y_i|^r)^{1/r} \tag{2.1.1}$$

Note that r is not the number of predictors and is instead chosen by the user. Two common measures of distance are Manhattan distance and Euclidean distance (Upton and Brawn, 2023).

When $r = 1$, Equation 2.1.1 is the Manhattan distance as shown in Equation 2.1.2, which is also known as city-block distance (Upton and Brawn, 2023). When $r = 2$, Equation 2.1.1 is the Euclidean distance as shown in Equation 2.1.3. Euclidean distance is the most commonly used distance measure (Hastie et al., 2009). When r = ∞, Equation 2.1.1 is the Supremum distance (L_{max} or L_∞ norm) as shown in Equation 2.1.4.

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i| \tag{2.1.2}$$

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.1.3}$$

$$d(\mathbf{x}, \mathbf{y}) = \lim_{r \to \infty} (\sum_{i=1}^{n} |x_i - y_i|^r)^{1/r} \tag{2.1.4}$$

Oftentimes, each predictor is standardized to have a mean of 0 and a standard deviation of 1 before the method is run, since each predictor could be measured in different units (Hastie et al., 2009). If scale of the attributes is not taken into consideration, the distance measure may be dominated by a single attribute (Tan et al., 2006).

Another important concept is the value of the parameter k which determines how many neighbors are chosen (Zhang, 2016). The choice of k has a significant impact on the performance of the algorithm. A large k value reduces the impact of the variance caused by random error while potentially missing smaller patterns (Zhang, 2016). A small k value is susceptible to noise (Tan et al., 2006). In order to choose the best k value, it is important to find the right balance of overfitting and underfitting. With large amounts of data, it is more sensible to choose a larger k value (Upton and Brawn, 2023). A common choice is choosing k be the square root of the total number of observations in the training dataset (Zhang, 2016). It is also common to minimize the training misclassification rate, or another evaluation metric, by analyzing each potential choice of k.

While numerical variables are more common, categorical variables can also be included into KNN by creating dummy variables. Similar to numerical variables, the dummy variables should

be standardized before running KNN (Upton and Brawn, 2023).

This method can be run in R with the `knn()` function inside the class library (Hastie et al., 2021). The parameters for the function are the following: a matrix of the predictors for the training data, a matrix of the predictors for the testing data, a vector of the labels for the training data, and a value for k. The function develops a model using the training set and the specified k value and outputs the labels for the testing set. R can also be used to find the best choice of k by calculating the training error for each value of k.

KNN is one of the simplest methods in machine learning, but it has still been successful in many classification problems. Some of the advantages of the KNN algorithm include its simplicity, efficiency, low retaining cost, low complexity of the algorithm, and suitability for automatic classification of large samples (Zhang et al., 2022). One negative of nearest-neighbor algorithms is the computational load, including finding the neighbors and storing the training set (Hastie et al., 2009). The KNN algorithm is also sensitive to the local structure of the data, as decisions are made locally rather than attempting to find a more global model (Mohammed et al., 2017). Too large of a sample space can have a negative impact on the KNN algorithm (Zhang et al., 2022). KNN struggles with high-dimensional and larger data. Additionally, KNN is a black-box model, which means that the predictor values go in and class predictions come out, but it is unknown what is happening in the model or the importance of each predictor.

2.2 Logistic Regression

Logistic regression is a type of generalized linear model when there are two levels of a qualitative response variable. Logistic regression is useful when predicting a binary outcome from continuous and categorical predictor variables (Kabacoff, 2022). The response variable can be converted into a dummy variable with one category mapping to zero and the other mapping to one. This is an adjustment from the traditional simple or multiple linear regression. It is a parametric method as there are coefficients which have to be estimated. Despite containing regression in its name, logistic regression can be used for classification.

Logistic regression models the probability that the traditional regression response Y belongs to a particular category (Hastie et al., 2021). Similar to linear regression, logistic regression can used with one or more predictors. Equation 2.2.1 gives the equation for logistic regression with p predictors. The left-hand side of the equation is called the log odds or logit while the right hand side contains the traditional β values and predictors associated with linear regression. Let X be the vector of p predictor variables (Ott and Longnecker, 2016). The odds, $\frac{p(X)}{1-p(X)}$, can take any value larger than 0 (Hastie et al., 2021). This means the log odds can be any real number. Odds values close to negative infinity indicate a very low probability that $Y = 1$, and odds values close to infinity indicate a very high probability that $Y = 1$. The quantity $p(X)$ is the probability that $Y = 1$ for given values of the p predictors (Ott and Longnecker, 2016). The logistic regression model has a logit that is linear, so increasing x_i by 1 increases the log odds by a factor of β_i (Hastie et al., 2021).

Once the coefficients have been computed, the probability that $Y = 1$ for any set of observations (X) can be computed using Equation 2.2.2. This equation is known as the logistic function. It gives the probability that $Y = 1$ for a given X. It produces an S-shaped curve, so that a sensible prediction (probabilities falling between 0 and 1) for a given record will occur (Hastie et al., 2021).

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p \tag{2.2.1}$$

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}} \tag{2.2.2}$$

Similar to linear regression problems, variables can be removed from the final logistic regression model to create a smaller model with less noise using stepwise variable selection or any other subset method (Kabacoff, 2022). A logistic regression model should be built using the training set.

To use logistic regression for classification, a cut point, c, is chosen where $p(X) < c$ classifies the data point X into the $Y = 0$ class, and $p(X) \geq c$ classifies X into the $Y = 1$ class and $0 < c < 1$. A common cutoff choice is 0.5 since it falls in the middle of 0 and 1 (Kabacoff, 2022). However, choosing a cutoff of 0.5 does not take into account the importance of classifying

each class. Another approach is to loop through possible cutoff values to find the one with the greatest accuracy or lowest misclassification rate. The training and test sets are then classified using the logistic regression model and the cut point. The classification of a new data point is done by computing $p(X)$ and then comparing it with the cut point to place it within one of two classes.

The `glm()` function in base R is used for fitting a logistic regression model (Kabacoff, 2022). This function requires an input of the formula of the model with the response and predictors specified, the dataset to be used, and specifying the family as binomial. Categorical predictors are automatically coded as dummy variables. The function is run using the training data. The `predict()` function returns the probability of each testing data point falling into one group. The function requires the model, the data to be predicted, and specifying the type as class. A confusion matrix can be constructed to analyze actual status versus predicted status using the `confusionMatrix()` function in the caret library. This function requires the actual class labels and the predicted class labels.

Logistic regression is a rather simple method, but there are still some advantages and disadvantages. Logistic regression is relatively easy to interpret since it is similar to linear regression. Variable importance can be learned from the estimated coefficients as well as the direction of the impact of each variable, since a regression equation is created. Since this method is not a black-box model, it can be seen how each data point is classified. On the other hand, logistic regression struggles with complex relationships that are nonlinear. It can also be sensitive to the selected cutoff point. Another obvious disadvantage is that it can only be used for binary outcome variables which means it is only helpful in two-class problems.

2.3 Classification Tree

Decision tree is a statistical model that can be used for both classification and regression. Decision trees can be constructed with both quantitative and qualitative predictor variables (Hastie et al., 2021). A decision tree used for classification, predicting a qualitative response, is referred to as a classification tree (Mohammed et al., 2017). There are no parameters to be estimated, so

the method is considered nonparametric (Upton and Brawn, 2023). A simple way to describe a classification tree is to think of it as a series of yes/no questions until a decision is reached (Tan et al., 2006).

Classification trees involve creating a set of binary splits on the predictor variables to create a tree that can be used to classify new observations into one of two or more groups (Kabacoff, 2022). It is a top-down induction of decision trees (Mohammed et al., 2017). Each split divides a single group into two smaller groups. The rules of the tree divisions depend on the computer algorithm used (Upton and Brawn, 2023). An example of a classification tree can be seen in Figure 2 (Kabacoff, 2022). The data comes from a breast cancer dataset and is attempting to classify each data point as benign or malignant. The first split is based on the variable sizeUniformity. Any data point with a sizeUniformity greater than 3.5 is classified as malignant while the rest requires another split using the bareNuclei variable. The proportion of each class in each node can also be seen in the visualization of the tree.

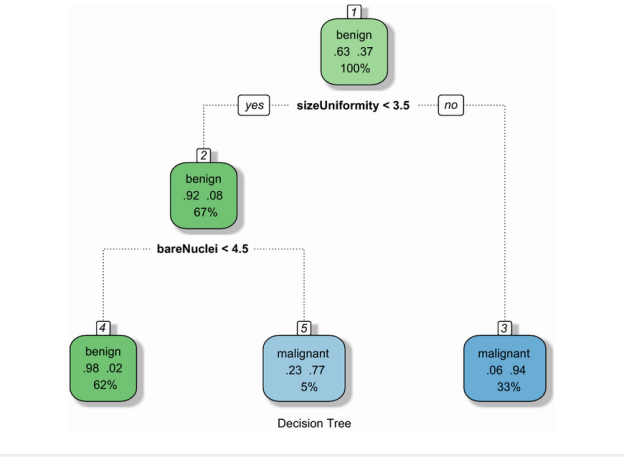


Figure 2: Example Classification Tree

A classification tree contains multiple types of nodes. A root node has no incoming edges and zero or more outgoing edges. The root represents the variable that plays an important role in classification. An internal node has one incoming edge and two or more outgoing edges. A leaf or terminal node has exactly one incoming edge and no outgoing edges. Each leaf node has a corresponding class label in a classification tree. The non-terminal nodes contain test conditions

to separate records with different attributes (Tan et al., 2006).

In theory, there are exponentially many trees which can be constructed from a given set of attributes. Some trees would be more accurate than others, but using this approach would be very difficult given the vast number of possibilities (Tan et al., 2006). Instead, efficient algorithms have been developed using a greedy approach where locally optimal splits are chosen (Tan et al., 2006). At each split, the model chooses the best split of the data at that point without considering the overall impact on the tree.

Recursive binary splitting is used to grow a classification tree (Hastie et al., 2021). The first step is to choose the predictor that best splits the data into two groups so that the purity of the outcome of the split is maximized (Kabacoff, 2022). If the predictor is numerical, a cut point is chosen to maximize purity, with the split being chosen by parsing through possible split values and determining the best one using some metric (Tan et al., 2006). If the predictor is categorical, the categories are combined into two groups if there are more than two and then the split is analyzed (Kabacoff, 2022).

The best, or purest, split can be chosen using a variety of metrics. The best is considered when the purest subgroups are created or the most information is gained from the split (Dean, 2014). That means that the split divides the data into two distinct groups relative to the response variable. The splits can be chosen using classification error rate, as shown in Equation 2.3.2 (Hastie et al., 2021). The classification error rate is the proportion of training observations in that region that do not fall into the most common class (Hastie et al., 2021). Another measure that can be used is the Gini index (Equation 2.3.3), which is a measure of variance across K classes or a measure of node purity (Hastie et al., 2021). Another measure is entropy (Equation 2.3.4), which is similar to the Gini index numerically (Hastie et al., 2021). When building a classification tree, the classification error rate, Gini index, or entropy are used to evaluate the quality of each split (Hastie et al., 2021).

Equation 2.3.1 is for node m representing a region R_m with N_m observations for the proportion of class k observations in node m (Hastie et al., 2009). It is the proportion of observations which fall into class k in a given node. Figure 3 displays the three measures of split for all values of $\hat{p}_{mk}$

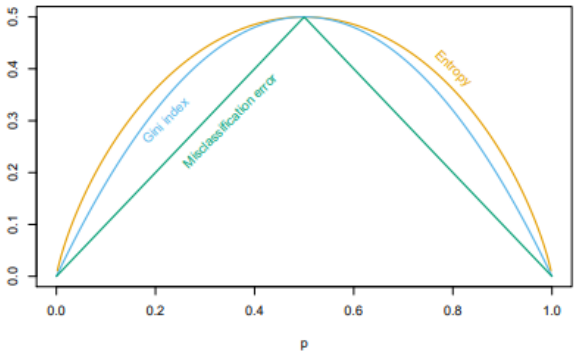


Figure 3: Classification Tree Measures of Split

(Hastie et al., 2009). Depending on the value of $\hat{p}_{mk}$, each metric takes on different values except at $\hat{p}_{mk}$ values of 0, 0.5, and 1.

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k) \tag{2.3.1}$$

$$\text{Classification Error Rate:} \frac{1}{N_m} \sum_{i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)} \tag{2.3.2}$$

$$\text{Gini Index:} \sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{2.3.3}$$

$$\text{Entropy:} - \sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk} \tag{2.3.4}$$

Instead of binary splits, splits into more than two groups can also be considered. However, this typically fragments the data too quickly, leaving not enough values at the next level (Hastie et al., 2009). Variables with large amounts of categories should be avoided as it leads to overfitting. Another option is to reduce the large amount of categories to only two categories.

Once the best split is chosen, the second step is to split the data into two groups using the chosen split and continue the process from step one in each subgroup (Kabacoff, 2022). At any split, if all of the points in the node are of the same class, it becomes a leaf node. If the points

belong to more than one class, then the previous step is performed to partition the node. The chosen metric of evaluating each split is consistent throughout the entire tree (Kabacoff, 2022).

The process is repeated with each subgroup until a stopping condition for growing the tree has been met. A stopping condition is often needed to end the tree growing process. While it may seem best to stop once all nodes contain the same class, early termination is often better to avoid overfitting (Tan et al., 2006). Stopping points for growing the tree could be insufficient data for additional splits, additional splits do not improve the model, or the tree has reached a specified depth (Dean, 2014). Once the final tree has been constructed, each terminal node is classified as the most commonly occurring class of training observations in the node (Hastie et al., 2021). Table 4 gives an outline of the method (Mohammed et al., 2017).

Steps of the Classification Tree Algorithm
1. Place all training records in a root.
2. Divide training records based on selected attributes.
3. Choose the best attribute based on some statistical measure.
4. Perform recursive partitioning until a stopping condition is met.

Table 4: Classification Tree Algorithm Steps

Often, this process still produces a large tree that suffers from overfitting (Kabacoff, 2022). Typically, increasing the number of nodes in a classification tree will decrease the training error, but at some point the testing error will begin to increase which is overfitting (Tan et al., 2006). Pruning is a process whose goal is to reduce this issue and make the model better for the general population (Dean, 2014). The complexity parameter (cp) is used to penalize larger trees (Kabacoff, 2022). A tree can be pruned using the cross-validated error. A good choice for the final tree size is the smallest tree with a cross-validated error within one standard error of the minimum cross-validated error value (Kabacoff, 2022). Any of the three approaches (Gini, entropy, classification error rate) can be used when pruning the tree, but classification error rate is typically used if

prediction accuracy of the final pruned tree is the goal (Hastie et al., 2021). Figure 4 shows an example a plot of the complexity parameter versus the cross-validated error (Kabacoff, 2022). The dotted line indicates the one standard deviation rule discussed previously. The general rule is to choose the size of the tree with the leftmost cp value below the dotted line (Kabacoff, 2022). This choice minimizes both overfitting and underfitting.

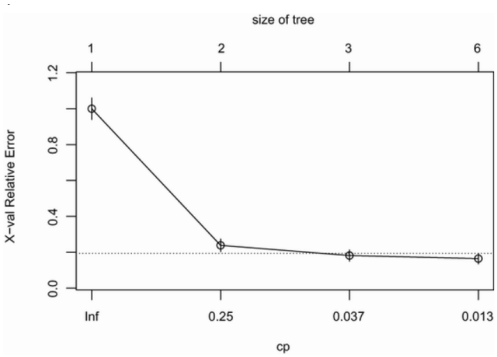


Figure 4: Example CP Plot

When looking to classify a specific observation, run it through the tree to a terminal node, and then assign it to the outcome of that terminal node (Kabacoff, 2022). For interpretation, it is common to be interested in not only the class prediction but also in the class proportions among the training observations that are in that respective region (Hastie et al., 2021).

This method can be run in R. Classification trees can be constructed using the `rpart()` function from the rpart package (Kabacoff, 2022). The function requires a formula with the response and predictors specified and the dataset to be used. The function is run using the training data. The default split measure is the Gini index. Again, the `predict()` and `confusionMatrix()` functions can be run for the testing set results. The `cptable` component from the `rpart()` output can be used for pruning. It is typical to select the tree size associated with the largest complexity parameter below the line in the `plotcp()` figure (Kabacoff, 2022). The `prune()` function can then be used to adjust the tree. The function takes the original tree and a cp parameter to cut beneath (Kabacoff, 2022). The `fancyRpartPlot()` function allows for a visualization of the tree with its input simply being the model created by the `rpart()` function (Kabacoff,

2022).

Decision trees have a variety of advantages and disadvantages (Hastie et al., 2021). An advantage of the recursive binary tree is that it is easy to interpret, as the entire model is described by a single tree (Hastie et al., 2009). Trees are easy to explain to other people and are believed to more closely mirror human decision-making (Hastie et al., 2021). Trees can be displayed graphically, especially if they are small (Hastie et al., 2021). While it may be difficult to draw with too many predictors, it still works in the same way as a simpler tree (Hastie et al., 2009). Trees are able to handle qualitative predictors easily without the creation of dummy variables and do not require the data to be scaled. Tree methods are popular for predictive modeling because they are simple with good predictive power and very few assumptions (Dean, 2014).

A major issue with trees is the high variance (Hastie et al., 2009). Often a small change in the data can completely alter the series of splits in the tree. This is due to the hierarchical design of the model where the error at the first split is carried down throughout the model (Hastie et al., 2009). In general, a single classification tree is traditionally less accurate than other methods, especially on complex datasets with complex interactions (Speiser et al., 2019). It is typically difficult to get variables with perfect splits in real-world scenarios (Dean, 2014).

2.4 Random Forest

While a single tree is simple and easy to interpret, it is not very accurate for predictions. The random forest method was developed to overcome this limitation that single trees have. The random forest is a nonparametric method that uses multiple trees which are combined to produce a single prediction (Hastie et al., 2021). This makes the random forest an ensemble method, where classification trees are the simple model. An ensemble method combines many simple models to create a single, powerful model (Hastie et al., 2021). The simple, or building block, models are sometimes referred to as weak learners because they have mediocre accuracy on their own. While combining large amounts of trees reduces interpretability, prediction accuracy is traditionally increased. Random forest is popular due to the high accuracy and very little tuning required (Hastie

et al., 2009).

Random forest uses the bagging technique to reduce the variance present in a single tree. Bagging, short for bootstrap aggregation, is a way to reduce variance for a prediction function. Random forest builds a large amount of de-correlated trees and averages them. Building many noisy, but roughly unbiased models reduces the variance (Hastie et al., 2009).

The general process, as described by Hastie et al. (2009, 2021), is to take a bootstrap sample of size N from the training data and grow a tree based on m randomly selected variables from p total variables at each split until the minimum terminal node size is achieved. For each tree at each split, a random sample of m predictors are chosen as split candidates from the full set of p predictors. The split is only allowed to use one of the m predictors, and a new random sample of size m is chosen at each split. Typically, m is chosen to be the square root of p or $\log_2(p) + 1$ (Tan et al., 2006). Choosing a small value of m is useful when there is a large number of correlated predictors, so m can edited depending on the problem. Reducing the number of features to analyze at each split also decreases the runtime (Tan et al., 2006). The default minimum node size is typically one which is used as the stopping criteria for each tree. Then all of the trees are outputted. Random forests take the class vote from each tree and then classifies each point using the majority vote of all of the trees. Variable importance plots can also be made from random forests. New records are classified by running them through all the trees in the forest and taking a vote (Kabacoff, 2022).

Random forest can be run in R using the `randomForest()` function found in the randomForest library (Kabacoff, 2022). The function requires a formula with the response and predictors specified and a dataset. The default number of trees is 500, and the minimum node size is 1. The default number of variables sampled at each node is $\sqrt{m}$, where m is the total number of predictors. The model is fit using the training data. Again, the `predict()` and `confusionMatrix()` functions can be run for the testing set results. Majority vote ties are broken randomly in R.

There are many advantages to the random forest method. Random forest can easily handle large datasets with large numbers of predictors (Speiser et al., 2019). Random forests reduce the likelihood of overfitting by using bagging (Hastie et al., 2009). Limiting the set of predictors

prevents all of the trees in the forest from being identical if there is a strong predictor which is always chosen as the initial split which can be thought of as de-correlating the trees (Hastie et al., 2021). This makes the average of the resulting trees less variable and therefore more reliable. As noted previously, random forest can also measure variable importance.

Still, there are some disadvantages to the random forest method. In the case where the number of variables is large, but the number of relevant variables is small, the method will perform poorly with a small m (Hastie et al., 2009). This is due to the only small chance that a relevant variable is an option to be chosen at each split. The entire forest has to be stored to make future predictions, which means that the random forest method takes up a large amount of space (Kabacoff, 2022). It is difficult to understand the classification rules and communicate them to others with a large number of trees, as it is a black-box method (Kabacoff, 2022). Lastly, random forest works better with a large number of predictors.

CHAPTER III

Simulation Studies

Chapter III discusses simulations performed in R to compare the four classification methods outlined in Chapter II under various distributional scenarios and compares their results.

3.1 Introduction to Simulations

Each simulation involves a categorical response variable with either two or three classes. The predictor variables of each class were randomly generated using the built-in functions in R. The goal was to determine how well each classification method performed on the testing set in each simulation. Each scenario was simulated 10,000 times, though for KNN simulations with a large number of data points, only 1,000 or 5,000 simulations were conducted for time efficiency.

Each classification method involves a unique set of R code. For all methods, the first step is to randomly create the data and label it based on the distribution it was generated from. The next step is to randomly split the data into the training and testing data sets. Seventy-five percent of the data is used in the training set, and twenty-five percent is used in the testing set for each simulation.

Each method is then trained on the training data set to build the model. Information is collected based on the performance of the method on the training set. For KNN, the parameter k is chosen based on minimizing the misclassification rate in the training data set. This is done by looping through each potential value for k, beginning at 2 and going up to the number of points in the testing set and recording the misclassification rate at each k value. The k value with the lowest misclassification rate is chosen for the final model. For logistic regression, the cutoff value is chosen based on minimizing the misclassification error in the training set. This is done by looping through the cutoff values from 0.01 to 0.99 in steps of 0.01 and recording the misclassification rate for each cutoff value. The cutoff value with the lowest misclassfication rate is chosen for the final model. For classification trees, no pruning is performed. For random forest, five hundred trees are constructed in each simulation. Neither classification trees nor random forest require any tuning

parameters. Information is then collected on the performance of the final model on the testing set.

In every simulation, the training misclassification rate and testing misclassification rate are recorded. For the two-class mixtures, the sensitivity and specificity are recorded for both the training and testing sets. Additionally, the k values for KNN, cutoff values for logistic regression, and depth for classification trees are also recorded. An average and median value for each statistic is computed as a summary of the simulation case. Only the average misclassification rates are included in the tables below for consistency.

The random sampling is done using some built-in functions in R. Table 5 highlights the important information for the three univariate distributions. The `rnorm()` function is used for random sampling from a normal distribution. The number of samples, mean of the normal distribution, and standard deviation need to be specified. The `rexp()` function is used for random sampling from an exponential distribution. The number of samples and rate need to be specified. The `rpois()` function is used for random sampling from a Poisson distribution. The number of samples and λ value need to be specified. The `rmvnorm()` function is used for sampling from a multivariate normal distribution. The number of samples, the means of each normal distribution, and a covariance matrix need to be specified. The multivariate normal distribution is too complex to include in Table 5.

Distribution	Parameters	PDF	Domain	Mean	Variance
Normal	μ, σ	$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	$-\infty < x < \infty$	μ	σ^2
Exponential	λ	$f(x) = \lambda e^{-\lambda x}$	$x \geq 0$	$\frac{1}{\lambda}$	$\frac{1}{\lambda^2}$
Poisson	λ	$p(x) = \frac{\lambda^x e^{-\lambda}}{x!}$	x = 0,1,2,...	λ	λ

Table 5: Univariate Distributions for Simulations

The distributions in each simulation case were chosen to give a variety of potential scenarios. The goal in each case was that the distributions would slightly overlap, making the classification harder. Clearly, not all possible scenarios were covered as part of this simulation work, and there are more distributions which could be sampled from. It is worth noting that logistic regression

only works on a binary outcome and random forest works best on data sets with multiple predictors. These limitations were taken into consideration when deciding what methods to use on each simulation case.

For both the two-class and three-class simulations, five different sample sizes are included. Each class contains 20, 50, 80, 100, or 500 data points. In all of these cases, the number of data points in each class is equal, so the classes are perfectly balanced. There is one instance in which the sample sizes are not equal, and this is noted later in the chapter. Given that the classes are equal, aside from one instance, misclassification rate should be a good evaluation metric.

For the first two sections, each table contains two or three columns indicating the sampling distributions. $N(x, y)$ indicates a normal distribution with a mean of x and a standard deviation of y. $E(x)$ indicates an exponential distribution with a rate of x. $P(x)$ indicates a Poisson distribution with a λ value of x. For each section, the column labeled N indicates the number of data points in each class. The columns with a method header contain the average testing misclassification rate with the average training misclassification rate in parentheses directly after. A $*$ next to a rate in any table indicates that less than 10,000 simulations were performed. The graphs in each section display the testing misclassification rate at each class size for a given simulation case, and each line represents a different method. When referring to the performance of a method, we are generally referring to the testing misclassification rates.

3.2 Two-Class Mixtures with Single Predictor

This section includes four types of two-class mixtures with only a single predictor. Each mixture is simulated on KNN, logistic regression, and classification tree. Random forest is not run because there is only one predictor in each simulation case. Figure 5 displays an example of a mixture of two normal distributions being analyzed in this section. There is an overlap between the two classes, which is why the mixtures cannot be easily classified.

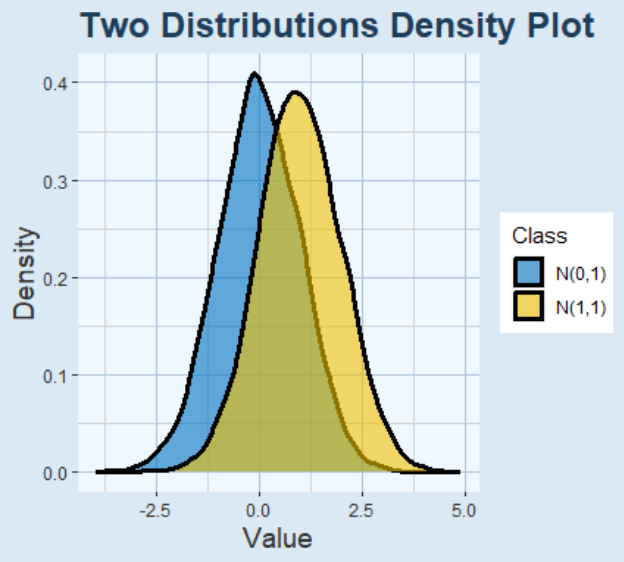


Figure 5: Two-Class Example Density Graph

3.2.1 Normal Mixtures

Table 6 contains the simulation results for four mixtures whose classes originated from normal distributions.

For the mixture in Figure 6a, logistic regression performs the best across all sample sizes. KNN consistently performs the worst, with classification tree falling in between. For the mixture in Figure 6b, the classification tree does the best, and logistic regression does the worst. KNN falls in between those two methods. For the mixture in Figure 6c, again logistic regression does the best, with KNN being consistently the worst. For the mixture in Figure 6d, logistic regression and classification tree performs very similarly with different leaders based on the sample size. At smaller class sizes, classification tree performs the best, and at larger class sizes logistic regression performs the best.

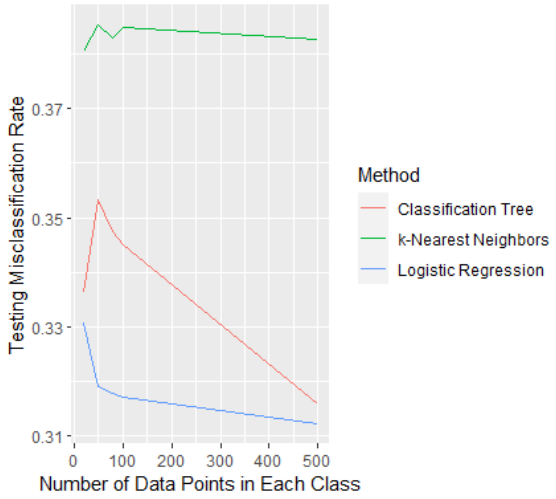
In each case, the misclassification rates across all class sizes are nearly identical for KNN. For logistic regression and classification tree, the misclassification rate decreases as class size increases. Logistic regression only performs the worst in the case where the means of each class are the same; this case also has the highest testing misclassification rate compared to the other methods. In general, KNN has the lowest training misclassification rate with the testing rate being

Table 6: Simulation Results for Normal Mixtures

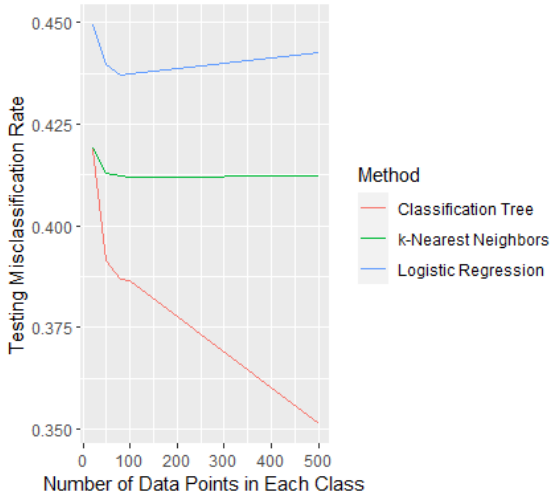
Class One	Class Two	N	k-Nearest Neighbors	Logistic Regression	Classification Tree
$N(0,1)$	$N(1,1)$	20	0.381 (0.199)	0.331 (0.249)	0.336 (0.245)
		50	0.385 (0.198)	0.319 (0.274)	0.353 (0.240)
		80	0.383 (0.197)	0.318 (0.282)	0.347 (0.240)
		100	0.383 (0.197)	0.317 (0.285)	0.345 (0.242)
		500	0.383 (0.198)*	0.312 (0.301)	0.316 (0.296)
$N(0,1)$	$N(0,2)$	20	0.419 (0.212)	0.449 (0.346)	0.419 (0.288)
		50	0.413 (0.208)	0.440 (0.386)	0.391 (0.256)
		80	0.413 (0.208)	0.437 (0.398)	0.387 (0.255)
		100	0.412 (0.208)	0.437 (0.404)	0.387 (0.256)
		500	0.412 (0.209)*	0.443 (0.433)	0.351 (0.320)
$N(0,1)$	$N(2,1)$	20	0.206 (0.109)	0.181 (0.117)	0.180 (0.115)
		50	0.207 (0.110)	0.168 (0.133)	0.179 (0.129)
		80	0.208 (0.112)	0.166 (0.140)	0.174 (0.132)
		100	0.208 (0.111)	0.166 (0.143)	0.174 (0.134)
		500	0.208 (0.110)*	0.161 (0.153)	0.162 (0.152)
$N(-2,2)$	$N(5,4)$	20	0.155 (0.081)	0.135 (0.081)	0.130 (0.115)
		50	0.152 (0.082)	0.123 (0.093)	0.124 (0.092)
		80	0.153 (0.083)	0.120 (0.098)	0.122 (0.096)
		100	0.152 (0.083)	0.119 (0.100)	0.122 (0.098)
		500	0.154 (0.084)*	0.115 (0.108)	0.115 (0.108)

close to double the training rate. This is an indication that KNN is overfitting the training data.

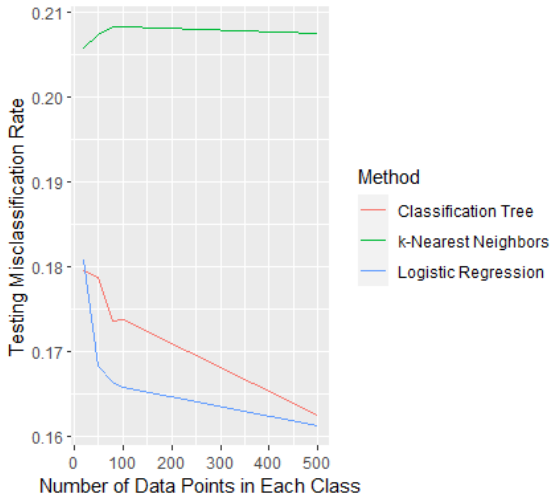
Figure 6: Graphs of Simulation Results for Normal Mixtures



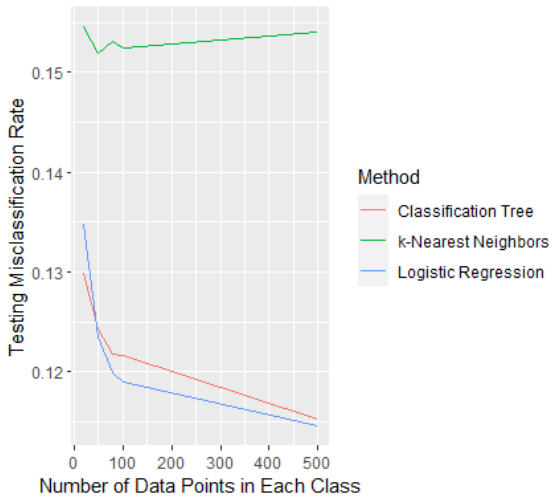
(a) $N(0,1)$ & $N(1,1)$



(b) $N(0,1)$ & $N(0,2)$



(c) $N(0,1)$ & $N(2,1)$



(d) $N(-2,2)$ & $N(5,4)$

3.2.2 Exponential Mixtures

Table 7 contains the simulation results for four mixtures whose classes originate from exponential distributions.

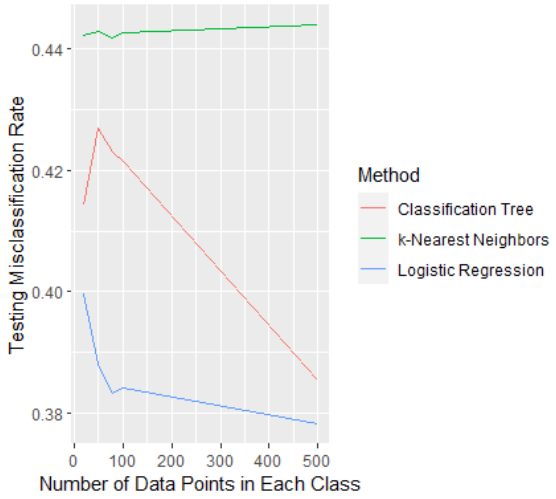
For the mixture in Figure 7a, logistic regression performs the best with KNN doing the worst. For the in Figure 7b, logistic regression again performs the best, and KNN again does the worst.

Table 7: Simulation Results for Exponential Mixtures

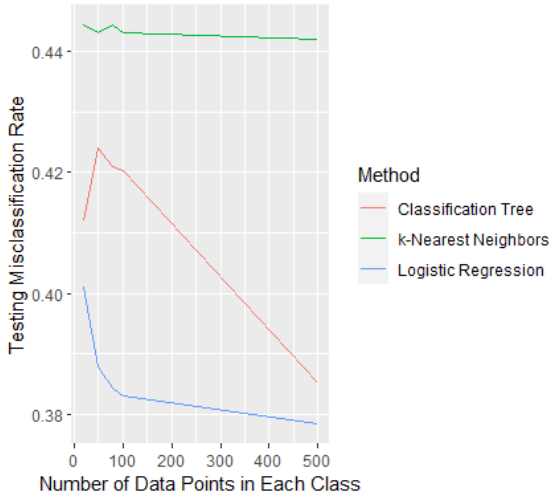
Class One	Class Two	N	k-Nearest Neighbors	Logistic Regression	Classification Tree
$E(0.5)$	$E(1)$	20	0.442 (0.227)	0.400 (0.310)	0.414 (0.294)
		50	0.443 (0.224)	0.388 (0.337)	0.427 (0.277)
		80	0.442 (0.223)	0.383 (0.347)	0.423 (0.275)
		100	0.443 (0.223)	0.383 (0.350)	0.421 (0.276)
		500	0.444 (0.223)*	0.378 (0.367)	0.386 (0.354)
$E(1)$	$E(2)$	20	0.444 (0.227)	0.401 (0.311)	0.412 (0.294)
		50	0.443 (0.224)	0.388 (0.338)	0.424 (0.277)
		80	0.443 (0.224)	0.384 (0.347)	0.421 (0.275)
		100	0.443 (0.223)	0.383 (0.351)	0.420 (0.276)
		500	0.442 (0.224)*	0.378 (0.367)	0.385 (0.354)
$E(0.5)$	$E(2)$	20	0.335 (0.175)	0.284 (0.214)	0.290 (0.210)
		50	0.335 (0.175)	0.274 (0.234)	0.303 (0.212)
		80	0.336 (0.174)	0.273 (0.241)	0.299 (0.214)
		100	0.337 (0.175)	0.271 (0.244)	0.296 (0.215)
		500	0.338 (0.174)*	0.266 (0.257)	0.271 (0.254)
$E(0.25)$	$E(4)$	20	0.155 (0.081)	0.130 (0.080)	0.129 (0.080)
		50	0.150 (0.081)	0.119 (0.092)	0.122 (0.090)
		80	0.151 (0.081)	0.117 (0.096)	0.119 (0.094)
		100	0.151 (0.082)	0.117 (0.098)	0.119 (0.095)
		500	0.151 (0.083)*	0.113 (0.106)	0.113 (0.106)

For the mixture in Figure 7c, logistic regression again performs the best, with KNN again doing the worst. For the mixture in Figure 7d, classification tree and logistic regression performed similarly and produced the best results.

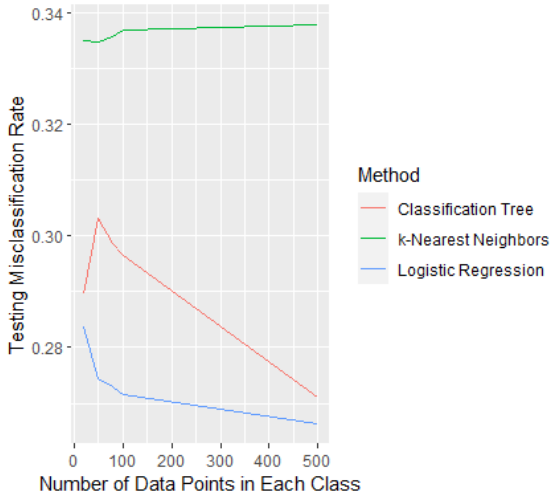
Figure 7: Graphs of Simulation Results for Exponential Mixtures



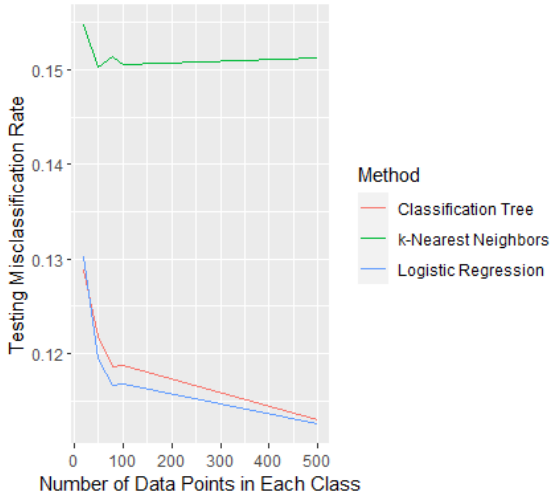
(a) $E(0.5)$ & $E(1)$



(b) $E(1)$ & $E(2)$



(c) $E(0.5)$ & $E(2)$



(d) $E(0.25)$ & $E(4)$

Logistic regression consistently performs the best, except for the last case where classification tree performs similarly. The last case features the class means which were the furthest apart; this results in the lowest testing misclassification rate across each method. The testing rate for KNN remains consistent across all class sizes while the rate decreases as class size increases for logistic regression and classification tree.

3.2.3 Poisson Mixtures

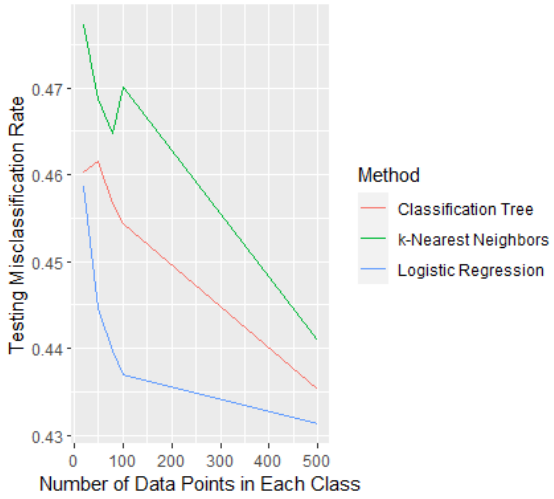
Table 8 contains simulation results for three mixtures whose classes originate from Poisson distributions.

Table 8: Simulation Results for Poisson Mixtures

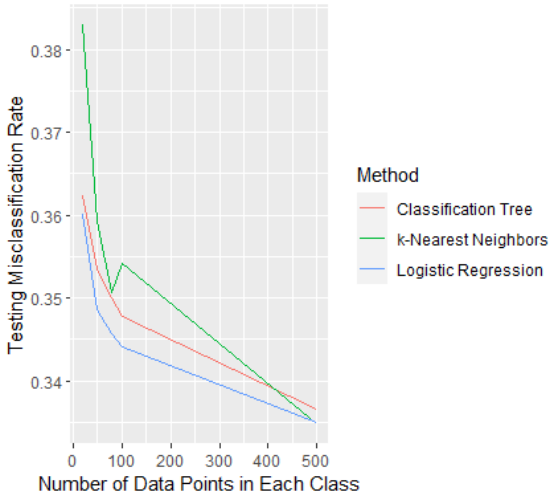
Class One	Class Two	N	k-Nearest Neighbors	Logistic Regression	Classification Tree
$P(7)$	$P(8)$	20	0.477 (0.308)	0.459 (0.445)	0.460 (0.357)
		50	0.469 (0.351)	0.445 (0.394)	0.462 (0.366)
		80	0.464 (0.372)	0.440 (0.403)	0.457 (0.379)
		100	0.470 (0.404)	0.437 (0.406)	0.454 (0.386)
		500	0.441 (0.418)*	0.431 (0.422)	0.435 (0.421)
$P(1)$	$P(2)$	20	0.383 (0.309)	0.360 (0.310)	0.362 (0.315)
		50	0.359 (0.327)	0.349 (0.327)	0.354 (0.325)
		80	0.351 (0.331)	0.346 (0.331)	0.350 (0.332)
		100	0.354 (0.339)	0.344 (0.332)	0.348 (0.332)
		500	0.335 (0.335)*	0.335 (0.335)	0.337 (0.335)
$P(3)$	$P(8)$	20	0.162 (0.114)	0.153 (0.112)	0.157 (0.113)
		50	0.150 (0.126)	0.144 (0.125)	0.148 (0.125)
		80	0.144 (0.129)	0.142 (0.128)	0.144 (0.128)
		100	0.144 (0.131)	0.141 (0.130)	0.143 (0.130)
		500	0.138 (0.135)*	0.139 (0.135)	0.139 (0.136)

For the mixture in Figure 8a, logistic regression performs the best. However, KNN and classification tree are not far behind. For the mixture in Figure 8b, logistic regression performs the best and ties KNN for the best rate with a class size of 500. Classification tree trailed by only a small margin. For the mixture in Figure 8c, logistic regression performs the best, but all three methods

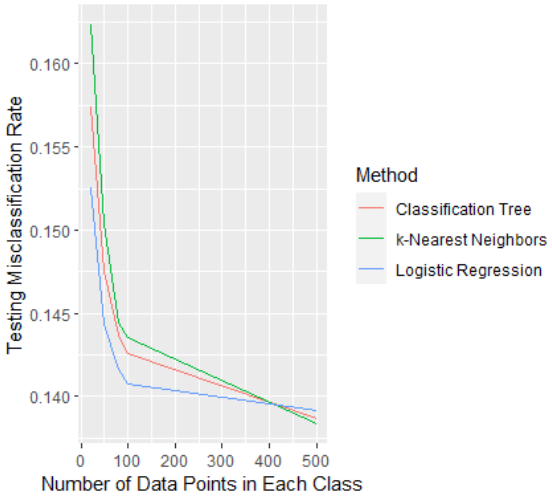
Figure 8: Graphs of Simulation Results for Poisson Mixtures



(a) $P(7)$ & $P(8)$



(b) $P(1)$ & $P(2)$



(c) $P(3)$ & $P(8)$

perform similarly at each class size.

In all three cases, logistic regression performs the best except for class sizes of 500 when all of the methods perform similarly. For all of the methods, as class size increases, the testing rate decreases. In general, there is no significant margin between the testing rates for all of the methods.

3.2.4 Multi-Distribution Mixtures

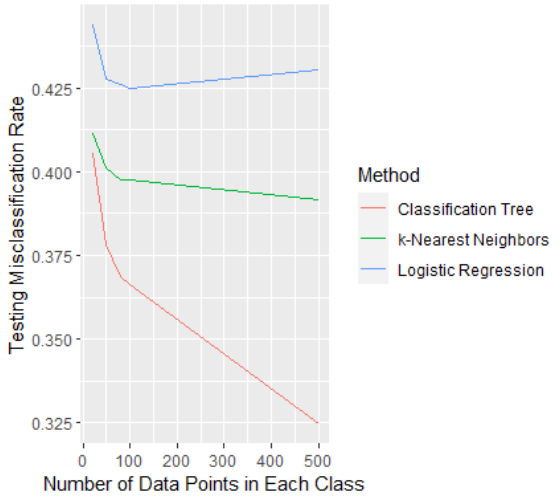
The first three mixtures for two-classes with a single predictor can be found in Table 9. These three mixtures contain one class from a normal distribution and one class from an exponential distribution.

Table 9: Simulation Results for Multi-Distribution Mixtures Part I

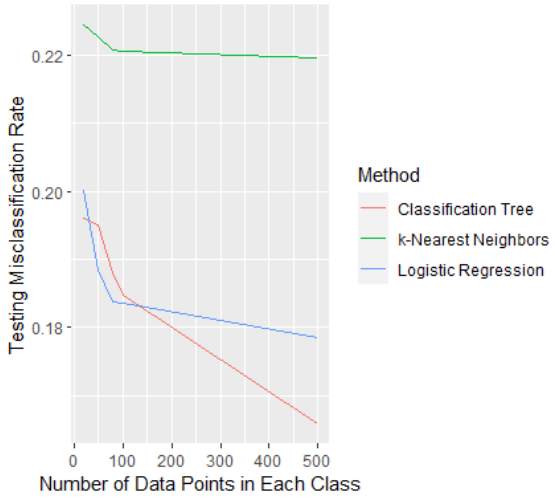
Class One	Class Two	N	k-Nearest Neighbors	Logistic Regression	Classification Tree
$N(2,1)$	$E(0.5)$	20	0.412 (0.208)	0.444 (0.342)	0.406 (0.284)
		50	0.401 (0.203)	0.428 (0.379)	0.378 (0.256)
		80	0.398 (0.203)	0.426 (0.392)	0.369 (0.253)
		100	0.398 (0.201)	0.425 (0.396)	0.366 (0.254)
		500	0.392 (0.200)*	0.431 (0.422)	0.325 (0.297)
$N(4,1)$	$E(0.5)$	20	0.224 (0.119)	0.200 (0.143)	0.196 (0.139)
		50	0.223 (0.119)	0.188 (0.156)	0.195 (0.146)
		80	0.221 (0.117)	0.184 (0.161)	0.188 (0.145)
		100	0.221 (0.117)	0.183 (0.163)	0.185 (0.144)
		500	0.220 (0.117)*	0.179 (0.172)	0.166 (0.153)
$N(4,1)$	$E(2)$	20	0.028 (0.013)	0.031 (0.008)	0.028 (0.008)
		50	0.029 (0.014)	0.027 (0.012)	0.025 (0.012)
		80	0.028 (0.014)	0.025 (0.013)	0.024 (0.013)
		100	0.027 (0.014)	0.025 (0.014)	0.024 (0.014)
		500	0.027 (0.015)*	0.021 (0.018)	0.022 (0.017)

For the first mixture in Figure 9a, classification tree clearly performs the best, and logistic regression performs the worst. For the second mixture in Figure 9b, logistic regression does the best except with class sizes of 20 and 500 when classification tree does better. For the third mixture

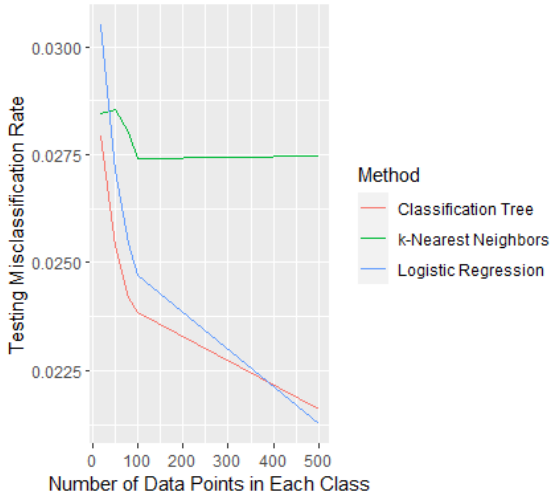
Figure 9: Graphs for Simulation Results for Multi-Distribution Mixtures Part I



(a) $N(2,1)$ & $E(0.5)$



(b) $N(4,1)$ & $E(0.5)$



(c) $N(4,1)$ & $E(2)$

in Figure 9c, classification tree does the best except for a class size of 500 when logistic regression does the best. In general, the misclassification rates are very close for all three methods.

The final four mixtures for two-classes with a single predictor can be found in Table 10. Each of these mixtures contains one class from a Poisson distribution.

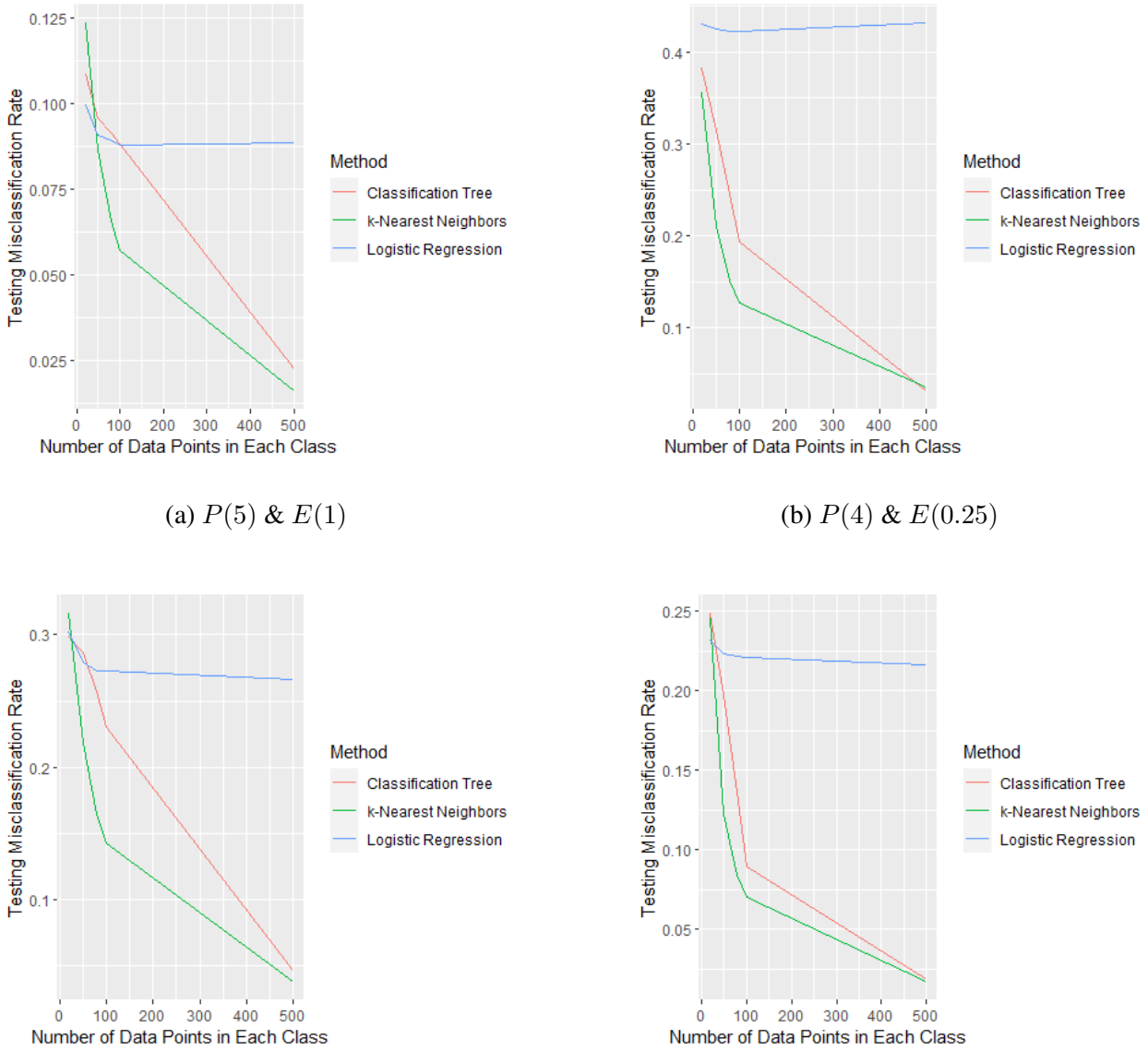
For the mixture in Figure 10a, logistic regression performs the best with a class size of 20, but KNN performs the best for every other class size. Logistic regression clearly performs the worst

Table 10: Simulation Results for Multi-Distribution Mixtures Part II

Class One	Class Two	N	k-Nearest Neighbors	Logistic Regression	Classification Tree
$P(5)$	$E(1)$	20	0.124 (0.067)	0.099 (0.068)	0.108 (0.066)
		50	0.086 (0.047)	0.091 (0.075)	0.096 (0.074)
		80	0.066 (0.035)	0.089 (0.078)	0.091 (0.074)
		100	0.057 (0.031)	0.088 (0.079)	0.088 (0.071)
		500	0.016 (0.009)*	0.089 (0.084)	0.022 (0.015)
$P(4)$	$E(0.25)$	20	0.356 (0.185)	0.430 (0.340)	0.382 (0.281)
		50	0.211 (0.107)	0.425 (0.378)	0.315 (0.222)
		80	0.152 (0.075)	0.423 (0.391)	0.245 (0.167)
		100	0.127 (0.062)	0.422 (0.395)	0.194 (0.125)
		500	0.035 (0.017)*	0.432 (0.423)	0.031 (0.015)
$P(6)$	$E(0.25)$	20	0.316 (0.174)	0.303 (0.243)	0.298 (0.223)
		50	0.219 (0.116)	0.279 (0.254)	0.287 (0.209)
		80	0.165 (0.085)	0.272 (0.256)	0.257 (0.190)
		100	0.143 (0.073)	0.273 (0.257)	0.230 (0.170)
		500	0.038 (0.019)*	0.266 (0.263)	0.046 (0.027)
$N(3,1)$	$P(5)$	20	0.246 (0.120)	0.232 (0.196)	0.249 (0.188)
		50	0.122 (0.062)	0.223 (0.206)	0.197 (0.140)
		80	0.084 (0.043)	0.222 (0.209)	0.137 (0.091)
		100	0.070 (0.036)	0.221 (0.210)	0.089 (0.054)
		500	0.017 (0.009)*	0.217 (0.215)	0.019 (0.009)

with a class size of 500. For the mixture in Figure 10b, KNN performs the best except for a class size of 500 when classification tree does the best. Again, logistic regression performs the worst. For the mixture in Figure 10c, KNN does the best except for a class size of 20 which classification

Figure 10: Graphs for Simulation Results for Multi-Distribution Mixtures Part II



(a) $P(5)$ & $E(1)$

(b) $P(4)$ & $E(0.25)$

(c) $P(6)$ & $E(0.25)$

(d) $N(3,1)$ & $P(5)$

tree performed the best. Again, logistic regression does the worst. For the last mixture in Figure 10d, KNN performs the best except for when the class size is 20 which logistic regression performs the best, and classification tree performs very similarly to KNN. Otherwise, logistic regression performs the worst at larger class sizes.

In all of the cases which contain a class from a Poisson distribution, KNN and classification tree see a significant decrease in misclassification rates as class size increases. For example, for

the mixture in Figure 10b, the testing rate for the class size of 20 was nearly 40% while it was only 3% for a class size of 500 for KNN and classification tree. Logistic regression on the other hand, sees only small decreases in testing rate as class size increases. For the cases which do not contain a class from a Poisson distribution, the decreases in testing rate are still present, but much smaller for logistic regression. Thus, there seems to be a clear detrimental effect on logistic regression when a continuous distribution is mixed with a discrete distribution.

3.3 Three-Class Mixtures with Single Predictor

This section contains three-class mixtures with a single predictor. Each mixture is simulated on KNN and classification tree. Logistic regression is not run because there are three-classes, and random forest is not run because there is only one predictor. Figure 11 displays an example of a mixture within this section.

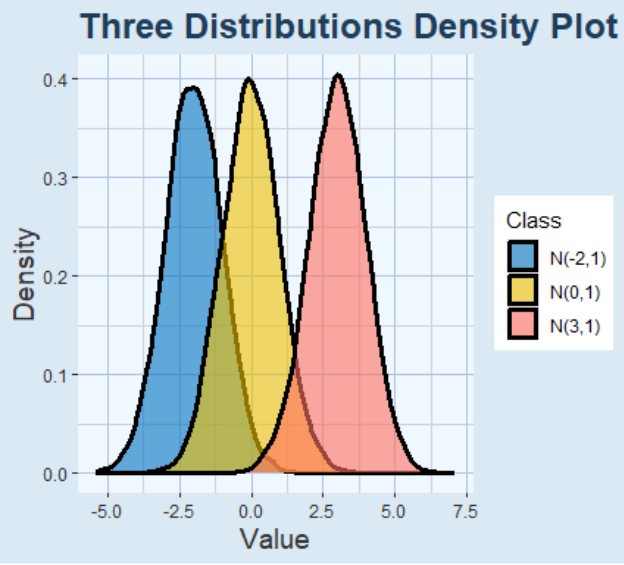


Figure 11: Three-Class Example Density Graph

The first four mixtures for three-classes with a single predictor can be found in Table 11. Each of these mixtures contain classes that all originate from the same type of distribution.

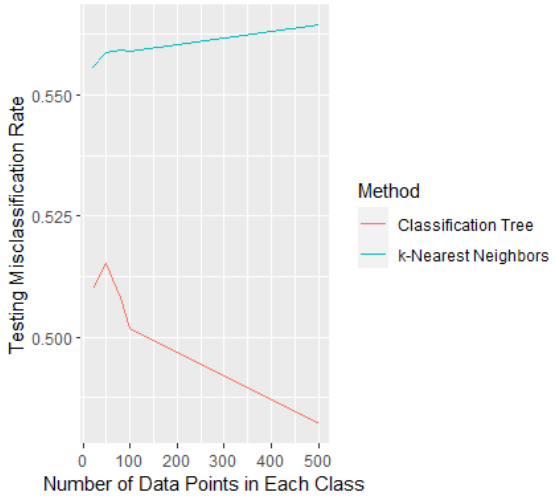
For the first three mixtures (Figures 12a, 12b, and 12c), classification tree does better at every class size. The error rate for KNN remains roughly the same at each class size while it slightly

Table 11: Simulations Results for Three-Class Mixtures with Single Predictor Part I

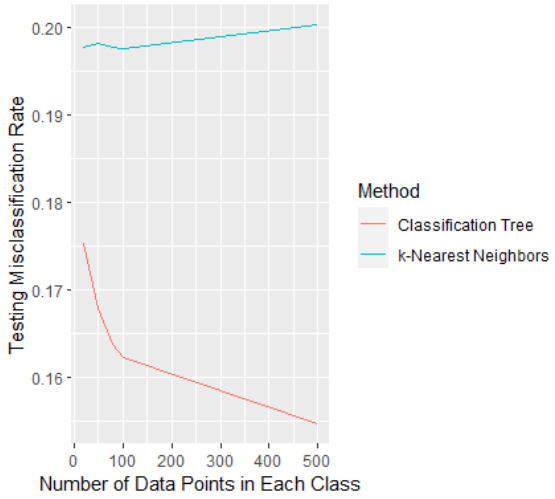
Class One	Class Two	Class Three	N	k-Nearest Neighbors	Classification Tree
$N(-0.5, 1)$	$N(0, 1)$	$N(1, 1)$	20	0.556 (0.295)	0.510 (0.379)
			50	0.559 (0.289)	0.515 (0.369)
			80	0.559 (0.287)	0.508 (0.377)
			100	0.559 (0.287)	0.502 (0.391)
			500	0.564 (0.283)*	0.482 (0.463)
$N(-2, 1)$	$N(0, 1)$	$N(3, 1)$	20	0.198 (0.106)	0.175 (0.104)
			50	0.198 (0.106)	0.168 (0.120)
			80	0.198 (0.106)	0.164 (0.125)
			100	0.198 (0.106)*	0.162 (0.129)
			500	0.200 (0.107)*	0.155 (0.143)
$E(0.5)$	$E(1)$	$E(4)$	20	0.517 (0.274)	0.463 (0.351)
			50	0.521 (0.271)	0.469 (0.341)
			80	0.521 (0.269)	0.460 (0.349)
			100	0.523 (0.268)*	0.454 (0.361)
			500	0.529 (0.266)*	0.437 (0.419)
$P(2)$	$P(5)$	$P(8)$	20	0.367 (0.268)	0.352 (0.287)
			50	0.347 (0.298)	0.342 (0.300)
			80	0.338 (0.307)	0.339 (0.309)
			100	0.334 (0.311)*	0.336 (0.312)
			500	0.323 (0.319)*	0.328 (0.321)

decreases for classification tree as class size increases. For the mixture in Figure 12d, classification tree does slightly better for class sizes of 20 and 50 while KNN does slightly better for the other class sizes.

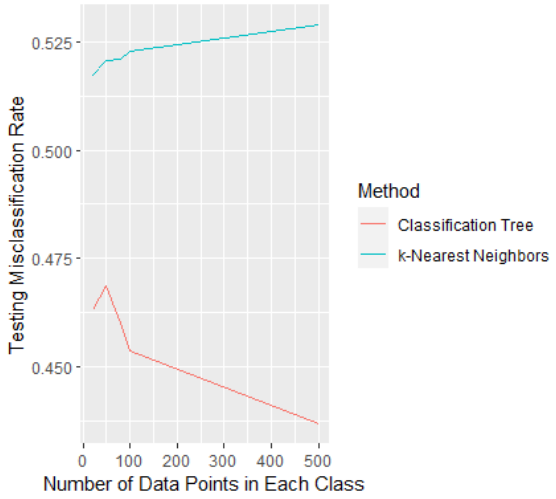
Figure 12: Graphs for Simulation Results for Three-Class Mixtures with Single Predictor Part I



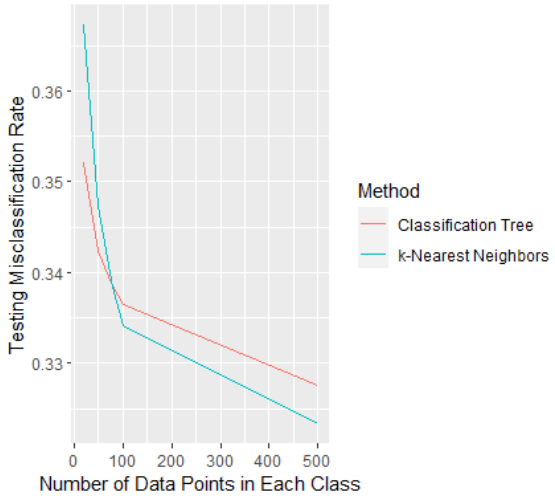
(a) $N(-0.5, 1)$ & $N(0, 1)$ & $N(1, 1)$



(b) $N(-2, 1)$ & $N(0, 1)$ & $N(3, 1)$



(c) $E(0.5)$ & $E(1)$ & $E(4)$



(d) $P(2)$ & $P(5)$ & $P(8)$

The final three mixtures for three-classes with a single predictor can be found in Table 12.

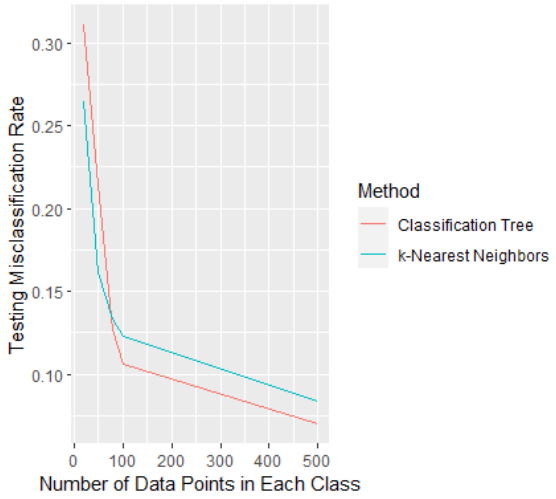
For the mixture in Figure 13a, KNN does the best for class sizes of 20, 50, and 80 while classification tree does the best for the other two. However, they both perform roughly similar at each class size. For the mixture in Figure 13b, KNN did the best at every class size. However, there is no significant difference at any class size. For the mixture in Figure 13c, classification tree clearly does the best at all class sizes.

Table 12: Simulations Results for Three-Class Mixtures with Single Predictor Part II

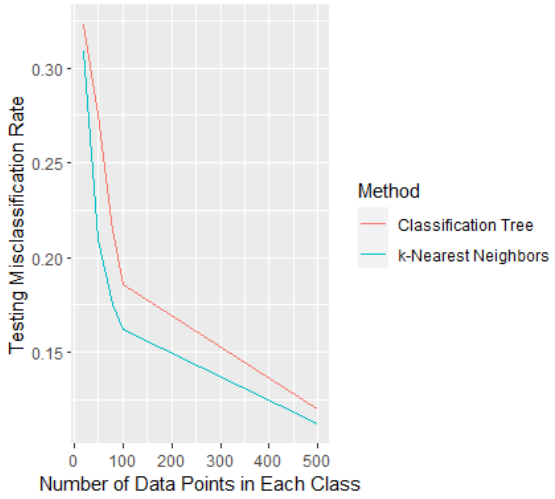
Class One	Class Two	Class Three	N	k-Nearest Neighbors	Classification Tree
$N(4,1)$	$E(1)$	$P(2)$	20	0.265 (0.136)	0.311 (0.243)
			50	0.161 (0.083)	0.214 (0.158)
			80	0.134 (0.069)	0.127 (0.089)
			100	0.123 (0.063)*	0.106 (0.075)
			500	0.084 (0.044)*	0.070 (0.061)
$E(0.5)$	$P(3)$	$P(8)$	20	0.309 (0.185)	0.323 (0.249)
			50	0.209 (0.145)	0.276 (0.214)
			80	0.175 (0.130)	0.214 (0.170)
			100	0.162 (0.124)*	0.186 (0.151)
			500	0.112 (0.101)*	0.120 (0.111)
$N(4,2)$	$E(0.5)$	$E(2)$	20	0.437 (0.232)	0.384 (0.284)
			50	0.440 (0.230)	0.386 (0.289)
			80	0.440 (0.228)	0.378 (0.296)
			100	0.440 (0.228)*	0.372 (0.303)
			500	0.440 (0.225)*	0.353 (0.335)

In this section, there is a clear difference between mixtures that contained a class generated from a Poisson distribution and mixtures that did not. For mixtures that did not, the testing rate remained consistent across all class sizes for KNN and decreased only slightly as class size increased for classification tree. For mixtures that did contain a Poisson, the testing rate decreased as class size increased for both KNN and classification tree. For the mixtures with at least one Poisson and at least one non-Poisson, the rates decreased dramatically as class size increased.

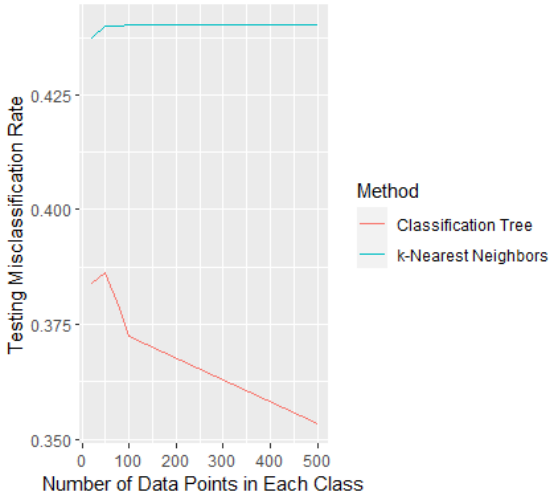
Figure 13: Graphs for Simulation Results for Three-Class Mixtures with Single Predictor Part II



(a) $N(4,1)$ & $E(1)$ & $P(2)$



(b) $E(0.5)$ & $P(3)$ & $P(8)$



(c) $N(4,2)$ & $E(0.5)$ & $E(2)$

3.4 Multivariate Normal Mixtures

This section contains two-class mixtures with two predictors from a multivariate distribution. Each mixture is simulated on KNN, logistic regression, and classification tree. Random forest is not run because there are only two predictors.

The first multivariate normal distribution being sampled from has means of 1 and 2 and a

covariance matrix of 3.4.1a, where r is the first distribution's correlation. The second multivariate normal distribution being sampled from has means of 2 and 3 and a covariance matrix of 3.4.1b, where s is the second distribution's correlation.

$$\begin{bmatrix} 1 & r \\ r & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & s \\ s & 2 \end{bmatrix} \tag{3.4.1a, 3.4.1b}$$

The r and s values change for each case, and the values are listed in the first two columns of the table. Each data point simulated from the multivariate mixtures above contains an x_1 and x_2 value, which means there are two predictor variables for these mixtures.

A summary of the multivariate normal mixture simulation results can be found in Table 13.

For the first case in Figure 14a, logistic regression is the best for every class size. Although the testing rate for classification trees decreases as class size increases, it is not lower than logistic regression. KNN is consistently the worst across all class sizes.

For the second case in Figure 14b, logistic regression slightly outperforms classification trees for class sizes of 20, 50, and 80, but classification trees have a lower misclassification rate for class sizes of 100 and 500. KNN is consistently the worst across all class sizes. At a class size of 500, classification tree clearly performs better than the other two methods.

For the third case in Figure 14c, logistic regression is best for every class size except for 500. At 500, classification trees outperform logistic regression. Classification tree is the worst for a class size of 20, but KNN clearly performs the worst for every other class size.

For the fourth case in Figure 14d, logistic regression is significantly better than the other two methods for each class size. Classification tree performs the worst at class sizes of 20, 50, and 80 while KNN performs the worst at a class size of 500.

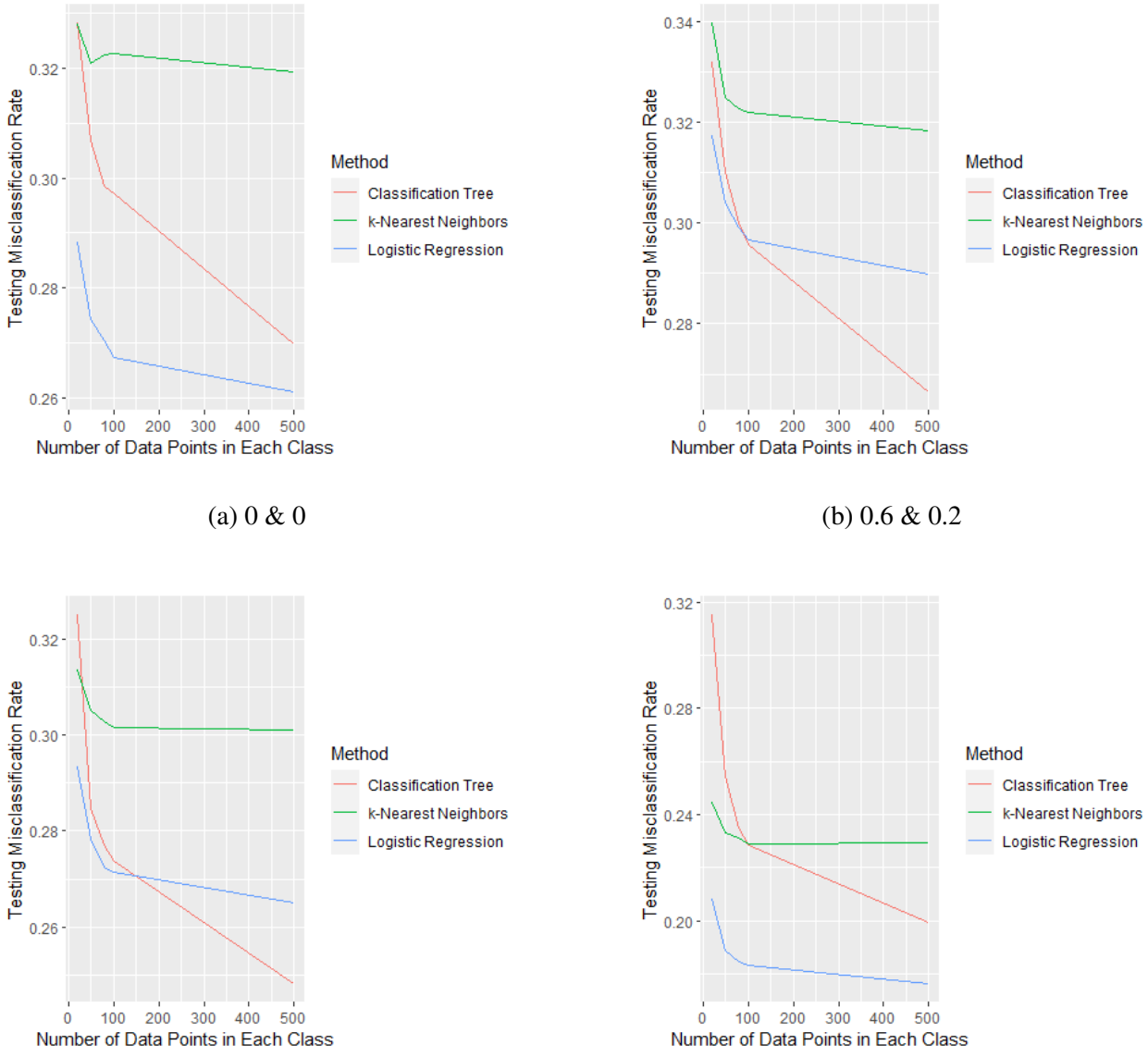
In general for each method, the testing error decreases as class size increases. The testing errors change greatly between 20 data points per class and 500 data points per class for classification tree while the change is smaller for logistic regression and even smaller for KNN. KNN consistently performs the worst for each case, except with the two negative correlations. On the other

Table 13: Simulations Results for Multivariate Normal Mixtures

First Correlation	Second Correlation	N	k-Nearest Neighbors	Logistic Regression	Classification Tree
0	0	20	0.328 (0.165)	0.288 (0.193)	0.328 (0.205)
		50	0.321 (0.165)	0.274 (0.221)	0.307 (0.195)
		80	0.322 (0.166)	0.271 (0.231)	0.299 (0.192)
		100	0.323 (0.166)	0.267 (0.234)	0.297 (0.191)
		500	0.319 (0.165)*	0.261 (0.250)	0.270 (0.233)
0.6	0.2	20	0.340 (0.168)	0.317 (0.216)	0.332 (0.213)
		50	0.325 (0.166)	0.304 (0.248)	0.310 (0.196)
		80	0.323 (0.165)	0.299 (0.258)	0.300 (0.191)
		100	0.322 (0.164)	0.297 (0.262)	0.296 (0.190)
		500	0.318 (0.165)*	0.290 (0.278)	0.266 (0.236)
0.4	-0.3	20	0.314 (0.157)	0.293 (0.197)	0.325 (0.204)
		50	0.305 (0.156)	0.278 (0.226)	0.285 (0.183)
		80	0.303 (0.156)	0.273 (0.235)	0.277 (0.178)
		100	0.301 (0.156)	0.271 (0.239)	0.274 (0.178)
		500	0.301 (0155)*	0.265 (0.254)	0.248 (0.220)
-0.7	-0.4	20	0.245 (0.120)	0.208 (0.126)	0.315 (0.188)
		50	0.233 (0.120)	0.189 (0.148)	0.254 (0.160)
		80	0.231 (0.120)	0.185 (0.153)	0.235 (0.150)
		100	0.229 (0.120)	0.183 (0.156)	0.229 (0.147)
		500	0.229 (0.120)*	0.176 (0.168)	0.199 (0.162)

hand, KNN has the lowest training rates across all class sizes. Again, this indicates KNN possibly overfitting the training data.

Figure 14: Graphs for Simulation Results for Multivariate Normal Mixtures



(a) 0 & 0

(b) 0.6 & 0.2

(c) 0.4 & -0.3

(d) -0.7 & -0.4

3.5 Two-Class Mixtures with Multiple Predictors

The mixtures in this section are broken into three parts: without noise, with noise, and a third case. Each part is a two-class mixture with multiple predictors. Each mixture is simulated on KNN, logistic regression, classification tree, and random forest.

The mixtures without noise contain four predictors. Each predictor is randomly chosen from

a set distribution. The first class contains x_1 (chosen from $N(1,1)$), x_2 (chosen from $N(1,2)$), x_3 (chosen from $E(0.5)$), and x_4 (chosen from $P(8)$). The second class contains x_1 (chosen from $N(-1,1)$), x_2 (chosen from $N(1,1)$), x_3 (chosen from $E(1.5)$), and x_4 (chosen from $P(6)$).

The mixtures with noise contain the same four predictors as the mixtures without noise. To generate noise, two more predictors added. x_5 and x_6 are randomly generated for both classes from $N(0,1)$.

The third case is unique in how it is constructed. A model is specified with eight predictors and nine β values as given in Equation 3.5.1.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6 + \beta_7 x_7 + \beta_8 x_8 + \epsilon \tag{3.5.1}$$

The predictors, x's, are randomly generated from a variety of uniform distributions. The β values are chosen in such a way as to produce roughly a 60/40 split in class labels. The noise term is randomly generated from a normal distribution whose standard deviation is the variance of y. The class labels are randomly generated from a Bernoulli distribution where p is the inverse logit (or logistic) function of y as shown in Equation 2.2.2, where $p = 8$.

A summary of the two-class mixtures with multiple predictor results can be found in Table 14. $\bar{N}$ indicates the total number of data points in the entire mixture rather than in each class due to the unique construction of the these class labels.

For the case with no noise in Figure 15a, KNN and classification trees perform similarly while logistic regression and random forest perform similarly. Random forest performs the best for class sizes of 20 and 500 while logistic regression performs the best for a class size of 80. For class sizes of 50 and 100, they perform identically.

For the case with noise in Figure 15b, random forest is consistently the best across all class sizes. Logistic regression trails random forest by less than a percentage point for class sizes greater than 20. KNN and classification trees perform similarly for class sizes greater than 20.

For the third case in Figure 15c, logistic regression performs the best for total data point sizes greater than 40. For a total data point size of 40, random forest performs the best. In contrast to

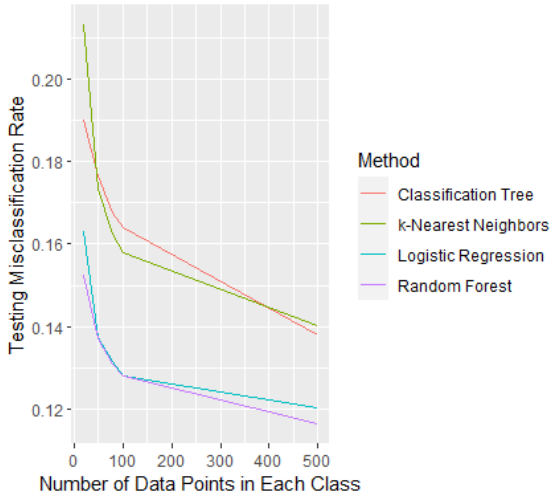
Table 14: Simulations Results for Two-Class Mixtures with Multiple Predictors

Simulation Case	N	k-Nearest Neighbors	Logistic Regression	Classification Tree	Random Forest
No Noise	20	0.213 (0.093)	0.163 (0.052)	0.190 (0.110)	0.152 (0.164)
	50	0.173 (0.081)	0.137 (0.085)	0.176 (0.113)	0.137 (0.141)
	80	0.163 (0.079)	0.132 (0.094)	0.168 (0.107)	0.131 (0.133)
	100	0.158 (0.078)	0.128 (0.097)	0.164 (0.103)	0.128 (0.131)
	500	0.140 (0.075)	0.120 (0.111)	0.138 (0.108)	0.116 (0.116)
Noise	20	0.222 (0.096)	0.187 (0.032)	0.192 (0.109)	0.156 (0.170)
	50	0.186 (0.086)	0.144 (0.078)	0.178 (0.112)	0.135 (0.139)
	80	0.173 (0.084)	0.134 (0.090)	0.171 (0.106)	0.129 (0.131)
	100	0.169 (0.083)	0.132 (0.094)	0.166 (0.103)	0.126 (0.128)
	500	0.148 (0.077)	0.121 (0.110)	0.139 (0.107)	0.114 (0.114)
	$\bar{N}$				
Third Case	40	0.437 (0.218)	0.432 (0.186)	0.442 (0.242)	0.407 (0.432)
	100	0.449 (0.223)	0.417 (0.286)	0.452 (0.218)	0.413 (0.426)
	160	0.451 (0.225)	0.409 (0.316)	0.454 (0.213)	0.412 (0.422)
	200	0.451 (0.225)	0.404 (0.327)	0.451 (0.212)	0.412 (0.419)
	1000	0.456 (0.228)	0.383 (0.366)	0.418 (0.279)	0.402 (0.405)

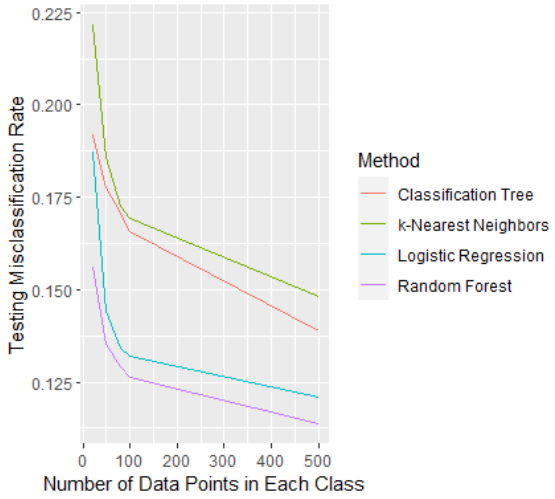
the other methods, the testing rate for KNN increases as the total number of data points increases.

Overall, random forest generally performs the best while KNN consistently performs the worst. Logistic regression also does a good job overall, especially in the third case. This is not surprising due to the way the third case was constructed. In general, classification tree tends to struggle with these mixtures. The inclusion of noise did not appear to greatly impact any of the methods. Even KNN and logistic regression only saw an increase of roughly one percent in testing rate.

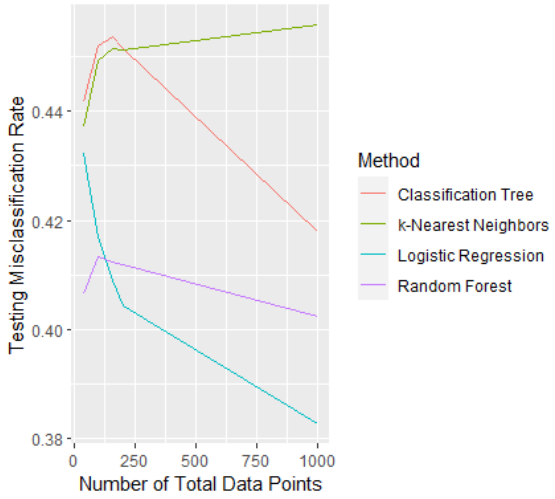
Figure 15: Graphs for Simulation Results for Two-Class Mixtures with Multiple Predictors



(a) No Noise



(b) Noise



(c) Third Case

3.6 Three-Class Mixtures with Multiple Predictors

The mixtures in this section are split into two parts: a mixture without noise and a mixture with noise. Each part is a three-class mixture with multiple predictors. Each mixture is simulated on KNN, classification tree, and random forest. Logistic regression is not run because there are three classes.

The mixtures without noise contain four predictor variables. The first class contains x_1 (chosen from $N(1,1)$), x_2 (chosen from $P(6)$), x_3 (chosen from $E(4)$), and x_4 (chosen from $P(11)$). The second class contains x_1 (chosen from $N(1,1)$), x_2 (chosen from $N(5,2)$), x_3 (chosen from $E(2)$), and x_4 (chosen from $P(6)$). The third class contains x_1 (chosen from $N(0,1)$), x_2 (chosen from $N(6,1)$), x_3 (chosen from $E(0.5)$), and x_4 (chosen from $P(8)$). The mixtures with noise contain the same four predictor variables as the mixtures without noise along with two more predictor variables. These variables, x_5 and x_6, are generated from $N(0,1)$ for both classes.

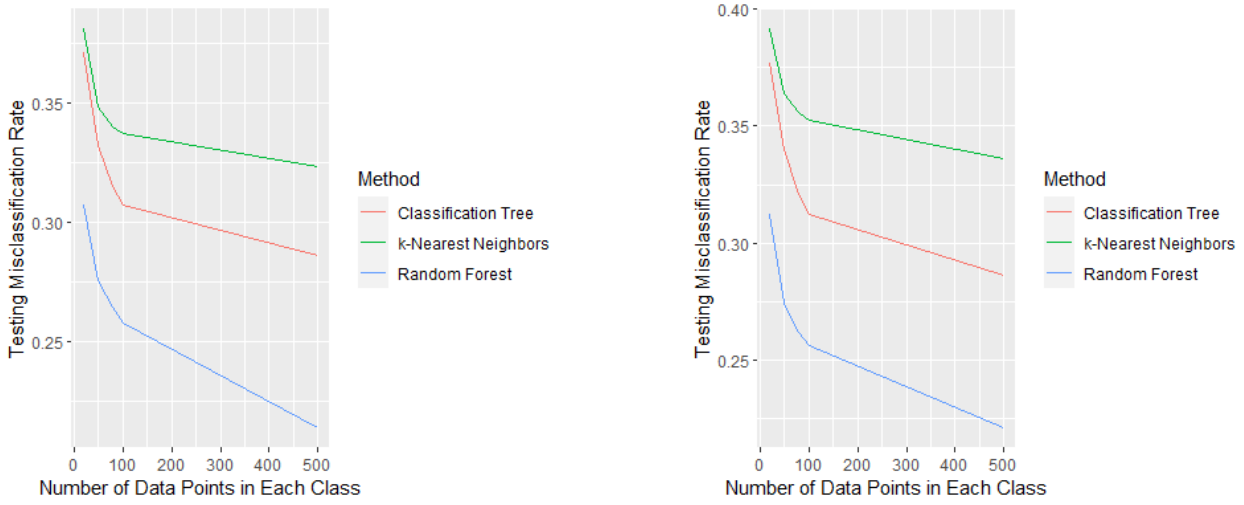
A summary of the three-class mixtures with multiple predictor results can be found in Table 15.

Table 15: Simulation Results for Three-Class Mixtures with Multiple Predictors

Simulation Case	N	k-Nearest Neighbors	Classification Tree	Random Forest
No Noise	20	0.381 (0.190)	0.372 (0.233)	0.307 (0.325)
	50	0.349 (0.178)	0.333 (0.219)	0.276 (0.282)
	80	0.340 (0.175)	0.315 (0.214)	0.264 (0.268)
	100	0.337 (0.174)	0.307 (0.214)	0.257 (0.262)
	500	0.324 (0.159)	0.286 (0.258)	0.214 (0.215)
Noise	20	0.391 (0.196)	0.377 (0.230)	0.312 (0.335)
	50	0.364 (0.185)	0.340 (0.216)	0.274 (0.282)
	80	0.356 (0.181)	0.321 (0.211)	0.262 (0.267)
	100	0.352 (0.180)	0.312 (0.211)	0.256 (0.260)
	500	0.336 (0.173)	0.286 (0.258)	0.221 (0.223)

For the case with no noise in Figure 16a, random forest clearly performs the best across all class sizes. Classification trees outperform KNN by a good margin across all class sizes too. For the case with noise in Figure 16b, random forest clearly performs the best across all class sizes. Classification trees outperform KNN by a good margin across all class sizes too.

Figure 16: Graphs for Simulation Results for Three-Class Mixtures with Multiple Predictors



(a) No Noise

(b) Noise

In both cases, the testing rates decrease as class size increases. Both tree methods see substantial decreases from class sizes of 20 to 500. KNN has the lowest training rates while random forest has the highest. Again, this indicates KNN possibly overfitting the training data. The inclusion of noise did not appear to greatly impact any of the methods.

3.7 Conclusion

These simulations display that there is no one single best method for every dataset. Within each section, the best method tended to vary. In many cases, there was not a significant difference between methods. Including the Poisson distribution as part of a mixture had a significant impact on the performance of the methods. Class size tended to have an impact on the performance of the methods as well.

In general for KNN, the testing rate is twice the training rate. This is a clear indication of overfitting. For random forest, the testing rate is only slightly higher than the training rate.

For two-class mixtures with a single predictor, logistic regression and classification trees tended to perform the best with KNN consistently performing the worst. For three-class mixtures with a

single predictor, classification tree outperformed KNN in almost every scenario. For the multivariate normal mixtures, classification tree and logistic regression tended to outperform KNN. For two-class mixtures with multiple predictors, logistic regression and random forest consistently performed the best over KNN and classification tree. For three-class mixtures with multiple predictors, random forest clearly outperformed KNN and classification tree.

CHAPTER IV

Applications

Chapter IV features some real-world dataset applications. We will look at the four methods discussed in Chapter II on each of the datasets, when possible. The datasets discussed in this chapter do not contain any missing values, but there was some data cleaning performed to reduce the number of columns used for some of the datasets. For each individual method, the data was randomly split into a training set and testing set with seventy-five percent of the data in the training set and twenty-five percent of the data in the testing set. This means that the training set for each method is not identical. For the summary tables, the training and testing rates refer to the respective misclassification rates.

4.1 Iris

The Iris dataset can be found in base R (R Core Team, 2023). It contains 150 records with four numerical variables and one categorical variable. The four numerical variables are sepal length, sepal width, petal length, and petal width. The categorical variable is the species, which is either setosa, versicolor, or virginica. There are fifty of each species in the whole dataset. For each method, the species is the variable that we are trying to predict. The four numerical variables are used as predictors. For the first three methods, the testing set contains 36 records. Since there are three categories in the response variable, adjustments were made to the dataset in order to run logistic regression. For logistic regression, there are 24 records in each of the testing sets.

4.1.1 k-Nearest Neighbors

The four predictor variables are first scaled. For each value of k from 1 to 36, the training and testing misclassification rates are recorded.

The graph in Figure 17 shows the misclassification rates for each value of k. The lowest misclassification training rate occurred at k values of 7 and 13. We selected 13 as it was the largest

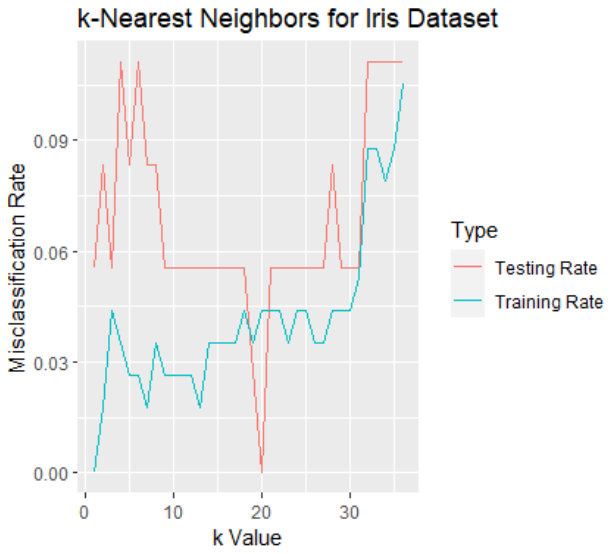


Figure 17: Iris Dataset k Selection

k value which minimized the training rate.

Table 16: Iris k-Nearest Neighbors Confusion Matrices

		Actual Class		
		Setosa	Versicolor	Virginica
Predicted Class	Setosa	38	0	0
	Versicolor	0	37	2
	Virginica	0	1	36

(a) Training Set Confusion Matrix

		Actual Class		
		Setosa	Versicolor	Virginica
Predicted Class	Setosa	12	0	0
	Versicolor	0	11	1
	Virginica	0	1	11

(b) Testing Set Confusion Matrix

The results from KNN on the Iris dataset can be found in Table 16. In the training set, there were only 3 records which were incorrectly classified. There were only 2 records incorrectly classified in the testing set. All of the setosa records were correctly classified in both the testing and training sets.

4.1.2 Classification Tree

A classification tree is created using the training dataset and can be seen in Figure 18. No pruning is performed since the tree is already small.

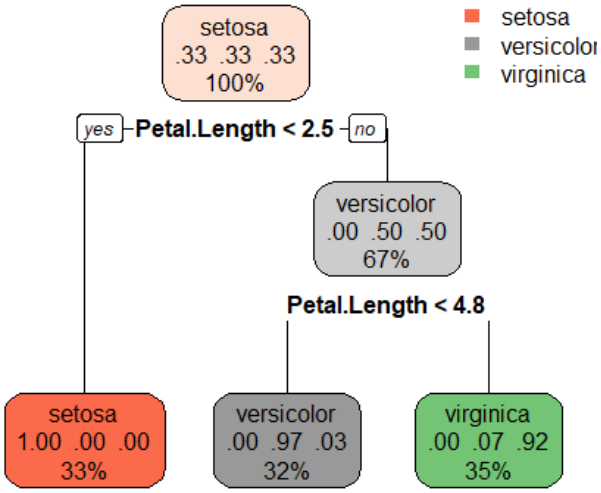


Figure 18: Iris Dataset Classification Tree Visualization

The tree first splits the data by petal length. A petal length less than 2.5 leads to classification as a setosa. A petal length greater than or equal to 2.5 leads to another split. For the remaining data points, petal length is again used to create a split. A petal length of less than 4.8 leads to being classified as versicolor while greater than or equal to 4.8 leads to being classified as virginica. There are only two splits in this tree, so again pruning would not be worthwhile or improve performance.

The performance results of a single classification tree on the Iris dataset can be seen in Table 17. There were 4 records incorrectly classified in the training set and 3 records incorrectly classified in the testing set. Setosa was perfectly classified in both the training and testing sets.

4.1.3 Random Forest

A forest of 500 trees was built on the training set.

Table 17: Iris Classification Tree Confusion Matrices

		Actual Class		
		Setosa	Versicolor	Virginica
Predicted Class	Setosa	38	0	0
	Versicolor	0	35	1
	Virginica	0	3	37

(a) Training Set Confusion Matrix

		Actual Class		
		Setosa	Versicolor	Virginica
Predicted Class	Setosa	12	0	0
	Versicolor	0	9	0
	Virginica	0	3	12

(b) Testing Set Confusion Matrix

Table 18: Iris Random Forest Confusion Matrices

		Actual Class		
		Setosa	Versicolor	Virginica
Predicted Class	Setosa	38	0	0
	Versicolor	0	35	4
	Virginica	0	3	34

(a) Training Set Confusion Matrix

		Actual Class		
		Setosa	Versicolor	Virginica
Predicted Class	Setosa	12	0	0
	Versicolor	0	12	2
	Virginica	0	0	10

(b) Testing Set Confusion Matrix

The confusion matrices in Table 18 display the results of random forest on the Iris dataset. There were 7 records incorrectly classified in the training set and 2 incorrectly classified in the testing set. Setosa was perfectly classified in both the training and testing sets.

4.1.4 Special Logistic Regression

Logistic regression can only be performed on binary response variables. However, the species variable contains three unique values. In order to run logistic regression, one of the species was removed, and logistic regression was run three separate times. A cutoff value of 0.5 was chosen in all three instances based on minimizing the training misclassification rate.

The first case uses only versicolor and virginica with the results in Table 19.

While the training set was perfectly classified, there were 2 records in the testing set that were

		Actual Class	
		Versicolor	Virginica
Predicted Class	Versicolor	38	0
	Virginica	0	38

(a) Training Confusion Matrix

		Actual Class	
		Versicolor	Virginica
Predicted Class	Versicolor	10	0
	Virginica	2	12

(b) Testing Confusion Matrix

Table 19: Iris Versicolor and Virginica Logistic Regression Confusion Matrices

incorrectly classified. Both records were classified as virginica but were actually versicolor.

The second case uses only setosa and virginica with the results in Table 20.

		Actual Class	
		Setosa	Virginica
Predicted Class	Setosa	38	0
	Virginica	0	38

(a) Training Confusion Matrix

		Actual Class	
		Setosa	Virginica
Predicted Class	Setosa	12	0
	Virginica	0	12

(b) Testing Confusion Matrix

Table 20: Iris Setosa and Virginica Logistic Regression Confusion Matrices

Both the training and testing sets had perfect classification.

The third case uses only setosa and versicolor with the results in Table 21.

		Actual Class	
		Setosa	Versicolor
Predicted Class	Setosa	38	0
	Versicolor	0	38

(a) Training Confusion Matrix

		Actual Class	
		Setosa	Versicolor
Predicted Class	Setosa	12	0
	Versicolor	0	12

(b) Testing Confusion Matrix

Table 21: Iris Setosa and Versicolor Logistic Regression Confusion Matrices

Both the training and testing sets again had perfect classification.

4.1.5 Summary of Results

Method	Training Rate	Testing Rate
k-Nearest Neighbors	0.026	0.056
Classification Tree	0.035	0.083
Random Forest	0.061	0.056

Table 22: Summary of Results for Iris Dataset

The summary of results for the Iris dataset can be found in Table 22. KNN and random forest had the exact same misclassification rate on the testing set. Classification tree performed only slightly worse on the testing set. KNN performed the best on the training set while random forest performed the worst. Random forest appears to be the best method for the Iris dataset as KNN displays some potential overfitting.

Missing Flower	Training Rate	Testing Rate
Setosa	0	0.083
Versicolor	0	0
Virginica	0	0

Table 23: Summary of Results for Logistic Regression on Iris Dataset

The results for logistic regression on the Iris dataset can be found in Table 23. Logistic regression did a great job of classifying setosa versus virginica and setosa versus versicolor but had some issues with classifying virginica and versicolor. There appears to be some overfitting in this case because the training error was 0 while the testing error was higher. Versicolor and virginica appear to be similar flowers, while setosa is distinct. This led to the misclassifications in the first logistic regression. This trend was also present in the other three methods.

4.2 Titanic

The Titanic dataset can be found in base R (R Core Team, 2023). It features a column of survived (yes/no) which is what we are trying to classify. There are three other columns: sex (male/female), age (adult/child), and class (1st/2nd/3rd/crew). Each of the predictors is a categorical variable. In the dataset, there are 711 people who survived and 1490 people who did not which indicates some slight class imbalance. There are 550 records in the testing set for each method.

4.2.1 k-Nearest Neighbors

The predictors were first converted to dummy variables. Only k values of 1 to 75 were considered as there were too many ties at larger values of k.

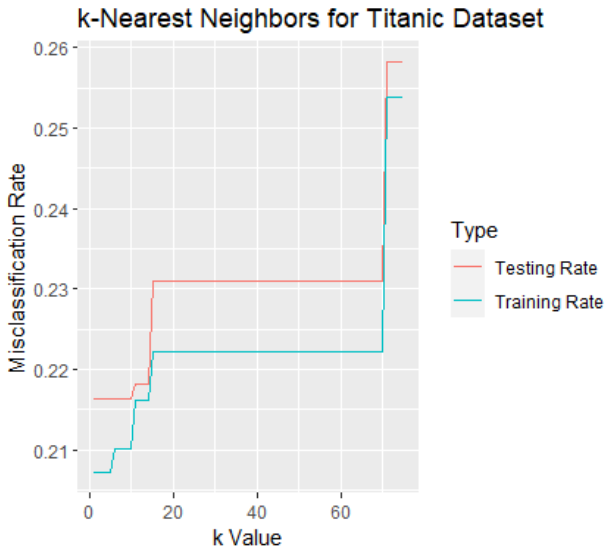


Figure 19: Titanic Dataset k Selection

The lowest training rate occurred with a k values of 2 through 5 as seen in Figure 19. So, a k of 5 was used as it is the largest k with the lowest training misclassification rate.

The confusion matrices in Table 24 display the performance of KNN on the Titanic dataset. The patterns in the training confusion matrix were also present in the testing confusion matrix where a larger amount of people who actually survived were classified as no than yes.

Table 24: Titanic k-Nearest Neighbors Confusion Matrices

		Actual Class	
		Yes	No
Predicted Class	Yes	205	14
	No	328	1104

(a) Training Set Confusion Matrix

		Actual Class	
		Yes	No
Predicted Class	Yes	65	6
	No	113	366

(b) Testing Set Confusion Matrix

4.2.2 Logistic Regression

The best cutoff is chosen as 0.58 for minimizing training error.

Table 25: Titanic Logistic Regression Confusion Matrices

		Actual Class	
		Yes	No
Predicted Class	Yes	219	29
	No	314	1089

(a) Training Set Confusion Matrix

		Actual Class	
		Yes	No
Predicted Class	Yes	54	8
	No	124	364

(b) Testing Set Confusion Matrix

The results from logistic regression on the Titanic dataset are displayed in Table 25. The trends from KNN were also present in the logistic regression confusion matrices where more people who actually survived were classified as a no than a yes.

4.2.3 Classification Tree

The first tree created from the training set can be seen in Figure 20.

The original tree is pruned to reduce its size without greatly impacting the misclassification rate. The new tree can be seen in Figure 21. This tree is used as the final model and contains fewer splits than the original model.

The root of the pruned tree splits males with yes being classified as did not survive. With

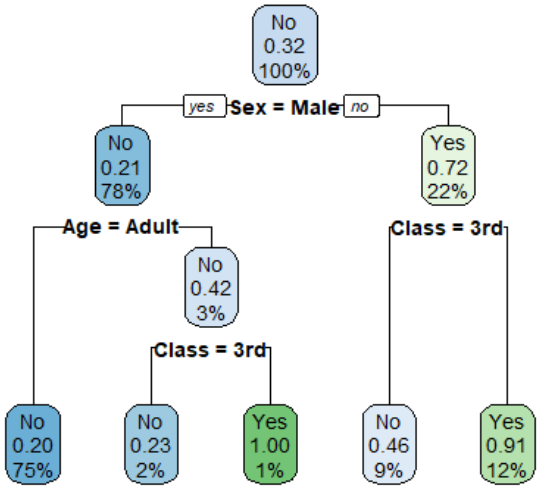


Figure 20: Titanic Dataset Classification Tree Visualization Part I

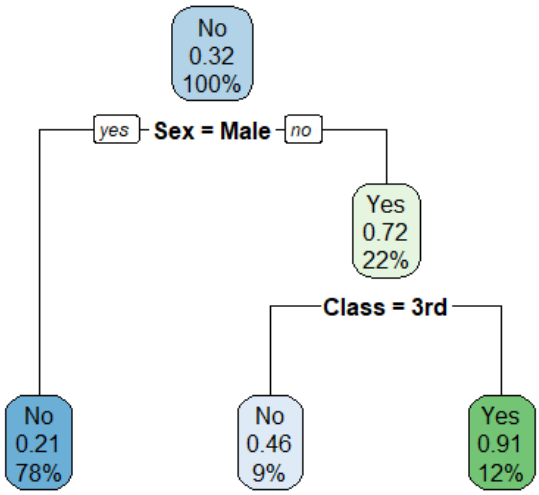


Figure 21: Titanic Dataset Classification Tree Visualization Part II

Table 26: Titanic Classification Tree Confusion Matrices

		Actual Class	
		Yes	No
Predicted Class	Yes	188	18
	No	345	1100

(a) Training Set Confusion Matrix

		Actual Class	
		Yes	No
Predicted Class	Yes	66	2
	No	112	370

(b) Testing Set Confusion Matrix

males being no, there is another split over the class variable. If the class is 3rd, then there is a classification of survived, while if the class is not 3rd then the classification is did not survive.

The results from a single classification tree on the Titanic dataset can be found in Table 26. The patterns present in the KNN and logistic regression confusion matrices were again present for the classification tree where more people who actually survived were classified as a no than a yes.

4.2.4 Random Forest

A forest of 500 trees was created using the training set.

Table 27: Titanic Random Forest Confusion Matrices

		Actual Class	
		Yes	No
Predicted Class	Yes	264	86
	No	269	1032

(a) Training Set Confusion Matrix

		Actual Class	
		Yes	No
Predicted Class	Yes	76	30
	No	102	342

(b) Testing Set Confusion Matrix

The performance of the random forest method on the Titanic dataset can be viewed in Table 27. Similar to the results from KNN, logistic regression, and classification tree, more people who survived were classified as no than yes. However, the issue appears to be less severe in the random forest confusion matrices.

4.2.5 Summary of Results

Method	Training Rate	Testing Rate	Training Sensitivity	Testing Sensitivity
k-Nearest Neighbors	0.207	0.216	0.385	0.365
Logistic Regression	0.208	0.240	0.411	0.303
Classification Tree	0.220	0.207	0.353	0.371
Random Forest	0.215	0.240	0.495	0.427

Table 28: Summary of Results From Titanic Dataset

A summary of the performance of each method on the Titanic dataset can be found in Table 28. Classification tree had the lowest testing misclassification rate. Surprisingly, the misclassification rate decreased from training to testing for classification tree. There was a significant increase in misclassification rate from training to testing for random forest which was also surprising. KNN had the lowest misclassification rate on the training set, only slightly lower than logistic regression. In general, the methods performed very similarly on the testing set.

Since it may be more important to classify those who survived, we also included sensitivity in the results (using yes as the positive class). Random forest had the highest sensitivity on both the training and testing sets. KNN, logistic regression, and classification tree both had much lower sensitivity on the testing sets depite having lower misclassification rates. While logistic regression and random forest had the same testing rate, the testing sensitivity values are vastly different. When taking sensitivity into account, random forest appears to be the best method for classifying the Titanic dataset.

4.3 Phones

The Phones dataset was found on the website Kaggle (Khal, 2024). The data contains information for various phones currently on the market as of 2024. There are 38 columns and 1708 rows

in the dataset. The phone_brand column was chosen as the variable to try and predict. There are 22 unique phone brands in the dataset, so we chose to only use the top four most popular phone brands in the dataset: Apple, Motorola, Samsung, and Xiaomi. Using that subset results in 1191 rows. There are 194 Apple phones, 142 Motorola phones, 428 Samsung phones, and 427 Xiaomi phones. Since there are 38 columns, only the numerical and binary columns were selected to make the analysis simpler. The columns selected are price_usd, storage, ram, weight, display_size, battery, year, foldable, ppi_density, and colors_available, in addition to the phone_brand column that is being predicted. Now, there are 10 predictors instead of the original 37. There are 298 records in each of the testing sets. Logistic regression was not run because there are four categories in the response variable.

This dataset is interesting for the fact that there are a large number of predictors and four categories in the response variable. This makes it different from the previously discussed Iris and Titanic datasets.

4.3.1 k-Nearest Neighbors

The predictors were first scaled. k values from 1 to 298 were considered, but the lowest error rates occurred with smaller k values as seen in Figure 22. Figure 23 was also included to make the choice for k more visible.

The k value that minimized the training misclassification rate was a k of 2. We can see in Figure 23 that this coincidentally minimized the testing error.

The results of KNN on the Phones dataset can be found in Table 29. In general, each phone brand was classified fairly well. The biggest issue occurred with the Motorola phones with 7 being misclassified in the testing set.

4.3.2 Classification Tree

The tree created on the training set is so large that visualization would not be readable. It is very large since there are many predictors and four categories in the response variable. Any

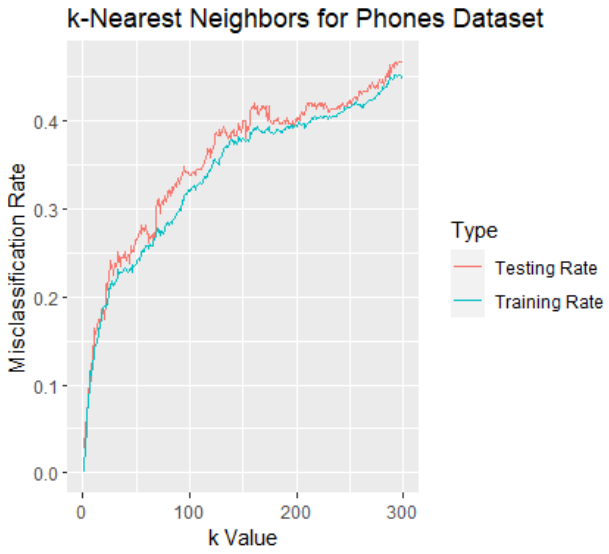


Figure 22: Phones Dataset k Selection Part I

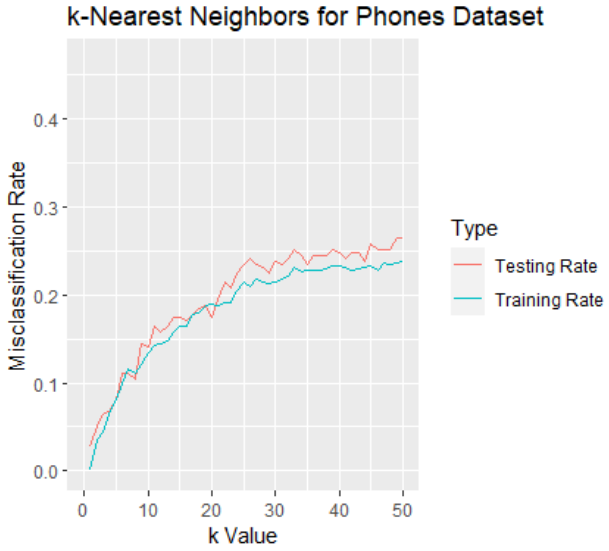


Figure 23: Phones Dataset k Selection Part II

pruning impacted the accuracy without significantly decreasing the size of the tree. So, we chose to keep entire original tree.

The results of a classification tree on the Phones dataset are displayed in Table 30. The classification tree had very few issues classifying Apple phones. There was an increase in misclassifications for Samsung, Motorola, and Xiaomi phones from KNN.

Table 29: Phones k-Nearest Neighbors Confusion Matrices

		Actual Class			
		Apple	Motorola	Samsung	Xiaomi
Predicted Class	Apple	146	0	0	0
	Motorola	0	95	0	1
	Samsung	0	3	315	8
	Xiaomi	0	8	6	311

(a) Training Set Confusion Matrix

		Actual Class			
		Apple	Motorola	Samsung	Xiaomi
Predicted Class	Apple	48	0	0	0
	Motorola	0	29	0	1
	Samsung	0	5	103	3
	Xiaomi	0	2	4	103

(b) Testing Set Confusion Matrix

4.3.3 Random Forest

A forest of 500 trees were created on the training set.

The confusion matrices in Table 31 display the performance of random forest on the Phones dataset. Apple phones were classified perfectly by the random forest method. The training set had some slight issues with the other three phones, but there were only two phones incorrectly classified in the testing set.

Table 30: Phones Classification Tree Confusion Matrices

		Actual Class			
		Apple	Motorola	Samsung	Xiaomi
Predicted Class	Apple	146	0	4	2
	Motorola	0	80	21	13
	Samsung	0	6	280	30
	Xiaomi	0	20	16	275

(a) Training Set Confusion Matrix

		Actual Class			
		Apple	Motorola	Samsung	Xiaomi
Predicted Class	Apple	48	0	1	1
	Motorola	0	25	11	4
	Samsung	0	2	89	8
	Xiaomi	0	9	6	94

(b) Testing Set Confusion Matrix

4.3.4 Summary of Results

A summary of the results from the Phones dataset can be found in Table 32. Random forest clearly did the best out of these three methods with a testing misclassification rate that is nearly zero. However, with many predictors and four categories in the response variable, random forest will take up a lot of space which is an important consideration for model implementation. Classification tree performed better than KNN for both the training and testing sets.

Table 31: Phones Random Forest Confusion Matrices

		Actual Class			
		Apple	Motorola	Samsung	Xiaomi
Predicted Class	Apple	146	0	0	0
	Motorola	0	101	0	5
	Samsung	1	5	310	5
	Xiaomi	0	1	5	314

(a) Training Set Confusion Matrix

		Actual Class			
		Apple	Motorola	Samsung	Xiaomi
Predicted Class	Apple	48	0	0	0
	Motorola	0	35	1	0
	Samsung	0	0	106	0
	Xiaomi	0	1	0	107

(b) Testing Set Confusion Matrix

Method	Training Rate	Testing Rate
k-Nearest Neighbors	0.029	0.050
Classification Tree	0.125	0.141
Random Forest	0.025	0.007

Table 32: Summary of Results From Phones Dataset

4.4 Cancer

The Cancer dataset was found on the UCI machine learning repository website (Wolberg et al., 1993). The dataset is for breast cancer in the state of Wisconsin. The goal of this classification is

to diagnose cells as benign or malignant with malignant meaning cancerous, so this was chosen as the response variable. The predictors of the dataset are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass and describe characteristics of the cell nuclei present in the image. The predictors are radius, texture, perimeter, area, smoothness, compactness, concavity, concave_points, symmetry, and fractal_dimension which are all continuous variables. There are 357 benign and 212 malignant records which indicates some slight class imbalance. There are 142 records in the testing sets.

4.4.1 k-Nearest Neighbors

The ten predictors were first scaled. k values from 1 to 142 were considered.

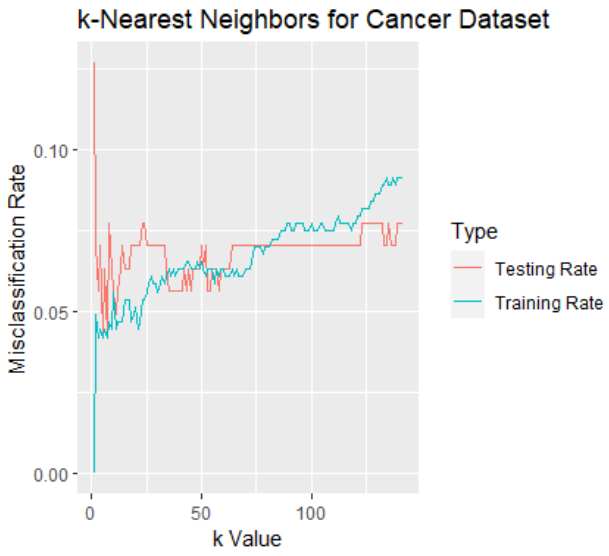


Figure 24: Cancer Dataset k Selection

The k value was chosen as 7 which minimizes the training misclassification rate. Figure 24 is interesting as the training rate continues to rise as the k value increases while the testing rate begins decreasing. We chose 7 as opposed to 3 or 5 because 7 is the largest of the k values which minimized the training misclassification rate.

The confusion matrices found in Table 33 display the results of KNN on the Cancer dataset. There were 18 records incorrectly classified in the training set with only 6 incorrectly classified in

Table 33: Cancer k-Nearest Neighbors Confusion Matrices

		Actual Class	
		Benign	Malignant
Predicted Class	Benign	261	11
	Malignant	7	148

(a) Training Set Confusion Matrix

		Actual Class	
		Benign	Malignant
Predicted Class	Benign	87	4
	Malignant	2	49

(b) Testing Set Confusion Matrix

the testing set. In both cases, there were slightly more false benign classifications.

4.4.2 Logistic Regression

A cutoff value of 0.48 was used based on minimizing the training misclassification rate.

Table 34: Cancer Logistic Regression Confusion Matrices

		Actual Class	
		Benign	Malignant
Predicted Class	Benign	262	10
	Malignant	6	149

(a) Training Set Confusion Matrix

		Actual Class	
		Benign	Malignant
Predicted Class	Benign	80	5
	Malignant	9	48

(b) Testing Set Confusion Matrix

The results from logistic regression on the Cancer dataset can be found in Table 34. There were 16 misclassifications in the training set and 14 in the testing set. In the training set, more records were incorrectly classified as benign while in the testing set more records were incorrectly classified as malignant.

4.4.3 Classification Tree

The tree constructed for the training set can be seen in Figure 25. No pruning was performed because the tree is already rather small.

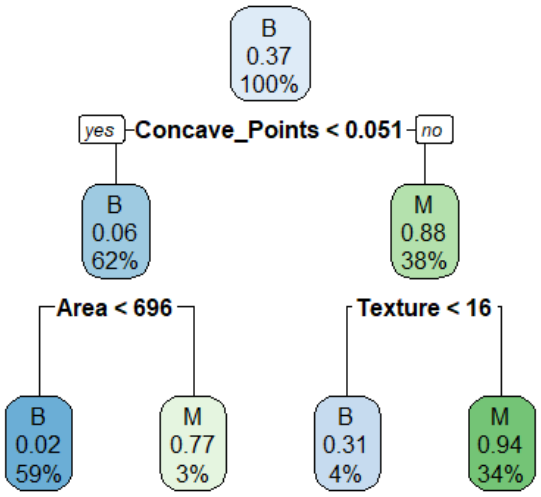


Figure 25: Cancer Dataset Classification Tree Visualization

The tree begins with a split on concave_points at 0.051. If it is less, then it splits at area of 696. If it is greater, then it splits at texture of 16. An area and texture on the smaller side gets classified as benign and larger gets classified as malignant.

Table 35: Cancer Classification Tree Confusion Matrices

		Actual Class	
		Benign	Malignant
Predicted Class	Benign	256	11
	Malignant	12	148

(a) Training Set Confusion Matrix

		Actual Class	
		Benign	Malignant
Predicted Class	Benign	81	7
	Malignant	8	46

(b) Testing Set Confusion Matrix

The confusion matrices in Table 35 display the results of a single tree on the Cancer dataset. In the training set, there were 23 incorrect classifications with 15 in the testing set. The amount of incorrect benign and malignant classifications were nearly identical.

4.4.4 Random Forest

A forest of 500 trees was constructed using the training data.

Table 36: Cancer Random Forest Confusion Matrices

		Actual Class	
		Benign	Malignant
Predicted Class	Benign	258	10
	Malignant	14	145

(a) Training Set Confusion Matrix

		Actual Class	
		Benign	Malignant
Predicted Class	Benign	84	5
	Malignant	5	48

(b) Testing Set Confusion Matrix

The results of random forest on the Cancer dataset can be found in Table 36. In the training set, there are 24 misclassifications with only 10 in the testing set. The training set has more malignant misclassifications while the testing set has an identical number of malignant and benign misclassifications.

4.4.5 Summary of Results

Method	Training Rate	Testing Rate	Training Specificity	Testing Specificity
k-Nearest Neighbors	0.042	0.042	0.931	0.925
Logistic Regression	0.038	0.099	0.937	0.906
Classification Tree	0.054	0.106	0.931	0.868
Random Forest	0.056	0.070	0.935	0.906

Table 37: Summary of Results From Cancer Dataset

The results from the Cancer dataset can be found in Table 37. We do not want to miss a cancer diagnosis, so we also added specificity to the table (using benign as positive class). KNN performed the best on the testing set based on misclassification rate. Surprisingly, the misclassification rate stayed the same from the training set to the testing set for KNN. The testing specificity is highest for both KNN and logistic with only a slight decrease from the training to testing sets. For random forest, the misclassifcation rate increased slightly from the training set to the testing

set. Logistic regression had the lowest error rate and highest specificity in the training set, but the misclassification rate more doubled on the testing set. Clearly, KNN is the best model choice for the Cancer dataset.

CHAPTER V

Conclusion and Future Work

5.1 Conclusion

Classification involves predicting a categorical response from a set of predictors. This makes it a supervised learning method, which falls under the umbrella of statistical machine learning. Simple classification problems are quite common in everyday life, such as predicting if it will rain today or identifying an illness.

Each classification method follows a similar structure and basic outline. There are multiple metrics which can be used to analyze the performance of a model, such as misclassification rate and specificity. Unfortunately, classification problems have their own set of issues, and each method deals with them in a different way. This means there are various advantages and disadvantages associated with each classification method.

In this thesis, four classification methods were analyzed through a serious of simulation studies on various mixtures of distributions. These methods were k-nearest neighbors, logistic regression, classification trees, and random forest. The simulations performed in R showed strong performances for logistic regression and classification trees for mixtures with a small number of predictors, while random forest performed well for mixtures with a larger number of predictors.

Each method was performed on a few real-world datasets. Although k-nearest neighbors struggled with the simulations, the real datasets showed that it and random forest tended to perform the best. This warrants further investigation, particularly into the k-nearest neighbors method.

5.2 Future Work

There are many aspects of classification and similar topics which were not covered extensively in this thesis. Another measure of distance is the Mahalanobis distance which takes into account correlation in the dataset. This could be explored further along with other measures of distance for

use in KNN. There are also other classification methods which could have been included in this thesis, such as support vector machines.

For the simulations, we could alter the parameters for each of our methods to produce dissimilar results. In the simulations performed, the chosen k for KNN tended to be small, so a larger choice of k could impact the testing misclassification rate. A different criteria to choose k, such as simply letting k be the square root of the number of training samples could produce different results for the KNN simulations. Additionally, the data could be scaled before running KNN. Using a different criteria to select the cutoff value in logistic regression, such as choosing based on maximizing specificity and sensitivity, could alter the logistic regression simulation results. More simulations could be performed, and incorporating some different distributions could be useful. Some of the irregularities in the simulation results could also be investigated further.

A topic found in some sources was the inclusion of a validation set. The validation set is a third split of the data, in addition to training and testing, which can be used as part of parameter tuning and model evaluation.

Preliminary research on unsupervised learning was carried out. This includes clustering methods, such as k-means. Performing similar actions, such as simulations and looking at real datasets, would be useful.

Looking at even more real-world datasets would be interesting. The applications of classification methods are nearly endless, so there are many avenues for further exploration.

REFERENCES

Dean, J. (2014). *Big Data, Data Mining, and Machine Learning: Value Creation for Business Leaders and Practitioners*. Wiley.

Hastie, T., G. James, R. Tibshirani, and D. Witten (2021). *An Introduction to Statistical Learning with Applications in R* (Second ed.). Springer Science+Business Media, LLC.

Hastie, T., R. Tibshirani, and J. Friedman (2009). *The Elements of Statistical Learning* (Second ed.). Springer Science+Business Media, LLC.

James, M. (1985). *Classification Algorithms*. John Wiley and Sons, Inc.

Kabacoff, R. I. (2022). *R in Action* (Third ed.). Manning Publications Co.

Khal (2024). Phones 2024: A comprehensive collection of phone information. *Kaggle*. `https://www.kaggle.com/datasets/jakubkhalponiak/phones-2024/data`.

Makridakis, S., E. Spiliotis, and V. Assimakopoulos (2018). Statistical and machine learning forecasting methods: Concerns and ways forward. *PLOS ONE*. `https://doi.org/10.1371/journal.pone.0194889`.

Mohammed, M., M. B. Khan, E. Bashier, and M. Bashier (2017). *Machine Learning: Algorithms and Applications*. CRC Press.

Ott, R. L. and M. Longnecker (2016). *An Introduction to Statistical Methods & Data Analysis* (Seventh ed.). Cengage Learning, Inc.

Prevos, P. (2019). *Principles of Strategic Data Science*. Packt Publishing.

R Core Team (2023). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. `https://www.R-project.org/`.

Speiser, J. L., M. E. Miller, J. Tooze, and E. Ip (2019). A comparison of random forest variable selection methods for classification prediction modeling. *Expert Systems with Applications*. https://doi.org/10.1016/j.eswa.2019.05.028.

Sugiyama, M. (2016). *Introduction to Statistical Machine Learning*. Morgan Kaufmann Publishers.

Tan, P.-N., M. Steinbach, and V. Kumar (2006). *Introduction to Data Mining*. Pearson Education, Inc.

Upton, G. J. G. and D. Brawn (2023). *Data Analysis: A Gentle Introduction For Future Data Scientists*. Oxford: Oxford University Press, Inc.

Wolberg, W., O. Mangasarian, N. Street, and W. Street (1993). Breast cancer wisconsin (diagnostic). *UCI Machine Learning Repository*. https://doi.org/10.24432/C5DW2B.

Zhang, C., P. Zhong, M. Liu, Q. Song, Z. Liang, and X. Wang (2022). Hybird metric k-nearest neighbor algorithm and applications. *Hindawi*. https://doi.org/10.1155/2022/8212546.

Zhang, Z. (2016). Introduction to machine learning: k-nearest neighbors. *Annals of Translational Medicine*. https://pmc.ncbi.nlm.nih.gov/articles/PMC4916348/pdf/atm-04-11-218.pdf.