QUANTIZATION AND STATISTICAL ANALYSIS OF DATA USING A BAYES DECISION RULE

MATTEO CAMPANELLI, KELSEY HORAN, & ALEXANDER WOOD
THE GRADUATE CENTER, CUNY

Abstract. Quantization is a method of transforming a large dataset into a smaller, discrete dataset. In this paper, we perform quantization on a large set of data and test decision rules on the data. We attempt to empirically determine the optimal number of quantizing levels for our quantizer. We show that running a classification experiment with a discrete Bayes decision rule yields a 99.8% accuracy rate, while other decision rules such as nearest neighbor, naive Bayes, Gaussian, and logistic regression yield lower classification accuracy rates.

1. Introduction

While many algorithms are built to run on discrete datasets or are more efficient on discrete datasets, in real-world examples we often find ourselves working with a large number of continuous data points. With this in mind, methods by which we transform continuous datasets into discrete datasets constitutes a large field of study [5]. There are multiple methods from which to approach this problem – a common method seen is binning the data uniformly [5]. However, when a classifier is present, we would like to analyze a method which is more tailored for our specific dataset in order to increase the learning accuracy of a classifier on a dataset. Furthermore, learning methods such as a discrete Bayes decision rule require discrete sets of data points [9]. Quantization is a process by which we take some large dataset and map it to a smaller, discrete dataset. Any method by which we perform this quantization is called our quantizer. While the quantization method had existed for many decades, it fell out of favor with researchers for many years, only to resurface again more recently [8].

During the Cold War the US was in the process of planting sonar arrays in the oceans to pick up the sounds created by ships. The president wanted to keep track of where exactly each country's ships were located, and where they were heading. At the time, the Russians did not have high-performance computing, but had propellers on their submarines which were quieter than our submarines, even with our superior computing power. This is because they had a method for solving the massive linear equations in their electrical signal processing based on quantization [4], [8].

Date: December 23, 2015.

Quantization rose back to interest in the 1980s after research into algorithms and coding structures in the 1970s made implementations of quantization on big data more feasible [11]. It continued its comeback into the 1990s when people began making exchanges on the Internet via routers, switches, and super switches. Furthermore, in the 90s, the FBI wished to implement a digital storing method for the millions of fingerprint records they had been keeping since 1924. The FBI's Criminal Justice Information Services Division followed an approach for storing their fingerprint images which is based on scalar quantization [2]. More recently, quantization found use after the 2007-2008 stock market crash in providing more accurate measures of risk evaluation [1].

In this paper we study the effects of quantization on a large dataset by first quantizing our data, then training discrete Bayes decision rule to assign class values. We compare the classification accuracy of this Bayes decision rule on quantized data to the classification accuracy of the Gaussian decision rule, naive Bayes, support vector machine, logistic regression, decision tree, and the nearest neighbor rule.

1.1. **Outline of Paper.** In Section 2 we outline the project. Subsection 2.1 gives an overview of the quantization method and a description of how we calculated our Bayes decision rule, trained our Bayes decision rule, and calculated a probability of correct identification. In Subsection 2.2 we discuss the pros and cons of alternative decision rules that could be used in place of a Bayes decision rule.

Next, in Section 3 we discuss our experimental results. Our experimental results on quantization with a Bayes decision rule were gathered in three stages:

(1) In Subsection 3.2 we describe our experiment on the first dataset where we train the quantizer and calculate a Bayes decision rule.
(2) In Subsection 3.3 we describe our experiment on the second dataset where we choose the number of quantization levels which yields the highest probability of correct identification.
(3) Lastly, in Subsection 3.4 we describe our experiment on the third dataset, where we determine the probability of correct identification using our previous optimizations.

Section 4 summarizes our results running the same experiment using alternate decision rules, namely: the Gaussian decision rule, Naive Bayes, Support Vector Machine, Logistic Regression, Decision Tree, and the Nearest Neighbor Rule.

2. Technical

2.1. **Quantization.** In this section, we describe the process of applying the quantization method of discretization on a real-valued dataset. Using quantization, we transform a set of continuous variables which can take on a quite large number

of values into a set of discrete variables which take on a much smaller number of values [13].

Say we have a collection of N-dimensional data points, where each data point has an assigned class c_0 or c_1. We would like to apply a decision rule on this data in order to determine each data point's classification. Our idea is to split our data points into L distinct *bins* along each component, leaving us with a total of L^N possible total bins in our measurement space [9]. We will call our collection of bins $\mathcal{B}$. From here, we apply a decision rule and run tests on this decision rule's accuracy.

More precisely, to *quantize* our data we defined a function called a *quantizer*:

Definition. *A* quantizer q *is a monotonically increasing function that takes in a real number and produces a non-negative integer between* 0 *and* $L-1$*, where* L *is the number of quantizing levels.*[9]

We define the *quantizing interval* as follows:

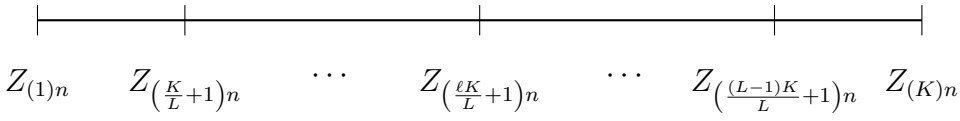
Definition. *The* quantizing interval Q_ℓ *associated with the integer* ℓ *is defined by*

$$Q_\ell = \{x : q(x) = \ell\} \text{ [9].}$$

When we are working with n-dimensional data, a data point $Z = (z_1, z_2, \ldots, z_n)$ is quantized as $(q_1(z_1), q_2(z_2), \ldots, q_n(z_N))$, where q_i is the quantizer for the Nth dimension. To determine our initial quantizing interval boundaries, consider a set of K N-dimensional data points $Z_1, Z_2, \ldots, Z_K$. Denote the kth tuple by $Z_k = (z_{k1}, z_{k2}, \ldots, z_{kN})$. Consider the values of the nth component for each of the K data points, $z_{1n}, z_{2n}, \ldots, z_{Kn}$. Arrange these values in ascending order, denoted

$$z_{(1)n}, z_{(2)n}, \ldots, z_{(K)n}.$$

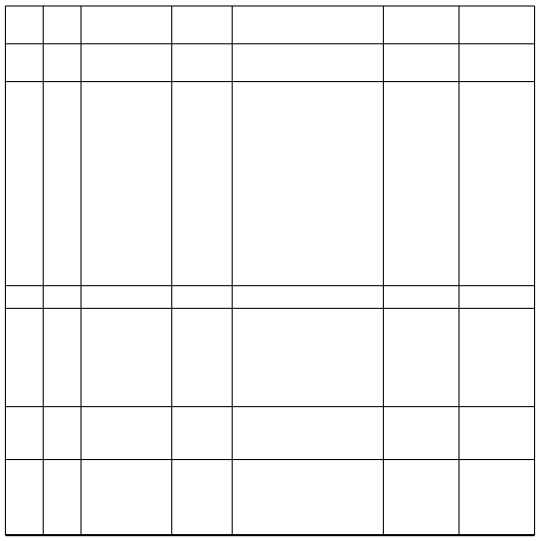
We wish to split our data into L bins along each component, each of which contains an equal number of data points along that component. Define the *left interval quantizing boundaries* along the nth component by:



In other words, the ℓth quantizing interval $[c_{n\ell}, d_{n\ell})$ for the nth component is given by

$$[c_{n\ell}, d_{n\ell}) = \left[z_{\left(\frac{(\ell-1)K}{L+1}\right)n}, z_{\left(\frac{\ell K}{L+1}\right)n}\right) \text{ [9].}$$

Applying this method on each dimension gives us our quantizing bins, indexed by the left interval boundary in each component, for a total of L^N bins. For clarity, pictured below is a sample quantization in $N = 2$ dimensions with $L = 7$ quantizing intervals for a total of 7^2 bins.



After applying our quantizer q on our K data points, we now wish to compute the maximum likelihood estimate of the probability for each bin. For a quantized tuple $q(Z_k) = (q_1(z_{k1}), q_2(z_{k2}), \ldots, q_n(z_{kN}))$, let $a(q(Z_k))$ denote its bin address. For a collection $\mathcal{B}$ of b bins, let t_m denote the number of quantized tuples in bin m and let p_m denote the maximum probability estimate of a data point falling in bin m. More precisely, we have:

$$t_m = \#\{k : a(q(Z_k)) = m\}, \quad p_m = \frac{t_m}{K}.$$

2.1.1. *Applying a Discrete Bayes Decision Rule.* We now want to apply a discrete Bayes decision rule on our quantized values in order to classify our data. Pick as sample set of K' data points from the set of K total data points on which to train the classifier. So, our sample set of points is $\{Z_1, Z_2, \ldots, Z_{K'}\}$. Say the total number of bins after quantization is M. Calculate the quantizer and the probability associations p_C, and then calculate a Bayes decision rule maximizing expected gain [9].

A discrete Bayes decision rule is a decision rule which satisfies Bayes theorem. Let c denote the class of a data point and let m denote the bin of the data point. Bayes theorem states the following:

$$P(c|m) = \frac{P(c)P(m|c)}{P(m)}.$$

Therefore, given a data point which falls into either class c_0 or c_1, we use our quantizer to compute m, and apply a decision rule to determine $P(c_0|m)$ and $P(c_1|m)$. The assigned class of the data point is whichever probability is higher. Observe that $P(m) = p_m$ was computed above.

To compute $P(m|c)$, we set up a table. Every row corresponds to a bin, and the columns correspond to classes c_0 and c_1. For column c and row m, we place

in the table $P(m|c)$, computed by looking at all of the data points in class c and setting

$$P(m|c) = \frac{\text{number of class } c \text{ data points in bin } m}{\text{total number of class } c \text{ data points}}.$$

The class probability is simply computed as

$$P(c) = \frac{\text{number of class } c \text{ data points}}{\text{total number of data points}}.$$

This defines our discrete Bayes decision rule.

Say a data point has true class c^i and assigned class c^j. Let $\Pr_{TA}(c^i, c^j)$ denote the probability that our data points have true class c^i and assigned class c^j. To compute the probability of correct identification under our decision rule, we first construct the *confusion matrix*:

		Assigned Class					
		c_1	c_2	$\cdots$	c_j	$\cdots$	c_ℓ
True Class	c_1						
	c_2						
	$\vdots$			$\ddots$	$\vdots$	$\iddots$	
	c_i			$\cdots$	$P_{TA}(c^i, c^j)$	$\cdots$	
	$\vdots$			$\iddots$	$\vdots$	$\ddots$	
	c_ℓ						

The probability of correct identification, P_C, occurs by adding up the diagonal entries of the confusion matrix [7]:

$$P_C = \sum_{i=i}^{\ell} P_{TA}(c^i, c^i).$$

2.1.2. *Optimizing the Quantizer Boundaries.* After applying a discrete Bayes decision rule, we wish to optimize our quantizer boundaries. In order to optimize our quantizer boundaries, we apply the following *greedy algorithm.* Let $b_{1n}, b_{2n}, \ldots, b_{Ln}$ denote the quantizing bins for dimension component n, $1 \leq n \leq N$. Repeat the following steps until there is no change [9]

- Randomly choose a component n and a quantizing interval ℓ;
- Randomly choose a small perturbation δ;
- Randomly choose a small integer M, such that we have no collision with our neighboring boundaries;
- Set $b_{\ell n}^{NEW} = b_{\ell n} - \delta(M+1)$;
- For $(m = 0; m \leq 2M; m++)$:
  - $b_{\ell n}^{NEW} = b_{\ell n}^{NEW} + \delta$
  - Compute new probabilities

  - Recompute Bayes rule
  - Save expected gain
- Replace $b_{\ell n}$ by the value associated with the highest expected gain

2.1.3. *Testing the Decision Rule.* After quantizing our data, applying a discrete Bayes decision rule, and optimizing the quantizer boundaries, we are ready to test our classification method for accuracy. On a new set of data points, run the entire experiment again to compute an unbiased probability of correct identification $\Pr_C$.

2.2. **Comparisons With Other Decision Rules.**

Gaussian. A Gaussian classifier applies the Maximum A Posteriori (MAP) rule (where we classify a point x choosing $\mathsf{argmax}_{C_j} P(x_1, ..., x_d|C_j)$) and models class-conditional distributions $P(x_1, ..., x_d|C_j)$ as multivariate Gaussian distributions.

Naive Bayes. The concept of Naive Bayes classification is based on Bayes Theorem and a strong (naive) assumption of independence assumption between the features. The assumption that features are independent does simplify the classification task dramatically, since it allows the class conditional densities $P(x_k|C_j)$ to be calculated separately for each variable, i.e., it reduces a multidimensional task to a number of one-dimensional ones. Following Bayes rule:

$$P(x_1, ..., x_d|C_j) \propto P(x_1, ..., x_d|C_j)P(C_j)$$

Since Naive Bayes assumes that the conditional probabilities of the independent variables are statistically independent we can decompose the likelihood to a product of terms:

$$P(x_1, ..., x_d|C_j) \propto \prod_{k=1}^{d} P(x_k|C_j)$$

The posterior becomes:

$$P(C_j|x_1, ..., x_d) \propto P(C_j) \prod_{k=1}^{d} P(x_k|C_j)$$

Support Vector Machine. A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. Among the possible separating hyperplanes, SVMs find the hyperplane that gives the largest minimum distance to the training examples. SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

Logistic Regression. Logistic regression fits a logistic curve to the relationship between data and classes. Logistic curve is an S-shaped or sigmoid curve that can be expressed, in an extended form, as:

$$y = \frac{1}{1 + e^{-(\alpha+\beta x)}}$$

Logistic regression fits α and β, using the method of maximum likelihood of observing the sample values. Maximum likelihood will provide values of α and β which maximize the probability of obtaining the data set.

Decision Tree. The decision tree is a natural model of learning. The training process builds a binary tree, where each internal node represents a yes/no question we can ask on a data point. When we want to classify a point $x = (x_1, ..., x_d)$ we start from the root and follow the path generated by the answers to the questions. Each leaf in the decision tree represents a class, thus the leaf at the end of x's path determine how we classify it. Questions may ask whether a feature x_i or a linear combination of all features is above a certain value. At each level questions are meant to maximize the information gain.

Nearest Neighbor Rule. The Nearest Neighbor (NN) Rule is an extremely simple non-parametric classifier. It assumes that data are represented in a metric space. Given a training set the algorithm makes no generalization about it. In order to predict the class of a point x, it uses the class of the point x' in the training set that has minimum distance from x.

Implementation. We implemented all the comparison classifiers in Python. While we had to implement Naive Bayes from scratch, the other classifiers could be used off the shelf from the scikit-learn package.

3. Experimental Results on Quantization

Consider a $10,000,000$ point dataset of 5-dimensional data points, each with an associated class, c_0 or c_1. Let L denote the number of quantizing intervals. Observe that after quantizing with L intervals on 5-dimensional data points, we will have L^5 bins. Call the collection of bins $\mathcal{B}$, where $|\mathcal{B}| = L^5$.

For each experiment we selected a number of quantizing intervals L. Numbers chosen were $L =$(FILL THIS IN).

For each L, we began our experiment by training the quantizer on our first $3,333,333$ data points. We then estimated the bin probabilities and calculated our Bayes decision rule, as outlined in Section 2. From here, we used the second $3,333,333$ data points to estimate a confusion matrix in order to calculate the probability of correct identification with L quantizing intervals, denoted $p_C(L)$.

We repeated this experiment while varying the value of L and recording which value thus far had given the highest probability of correct identification $p_C(L)$.

Once we identified the highest value for $p_C(L)$, we stored this value of L as L_{BEST}. The final step in our experiment was to apply our discrete Bayes decision rule on the last $3,333,334$ data points with L_{BEST} quantizing intervals to determine an unbiased estimate of $p_C(L_{BEST})$.

3.1. **The Dataset.** The global dataset provided is composed of 10^7 rows in total. Each row corresponds to a labeled five-dimensional point $x \in [0,1]^5$. The label is either 0 or 1, respectively denoting classes c_0 and c_1 which x can belong to.

We partitioned the dataset differently for the quantizer and the comparison classifiers. For classification using the quantizer we divided the dataset in subsets containing one third of the rows. We used the first, second and third subset respectively as training, development and testing dataset. For the comparison classifiers we used the first two thirds of the data for training and the remaining third for testing.

Figures 1 and 2 provide a visual overview of the data, respectively in one and two dimensions. Figure 1 shows the distribution of each component of the data. Figure 2 shows the joint distribution for every possible pair of components.

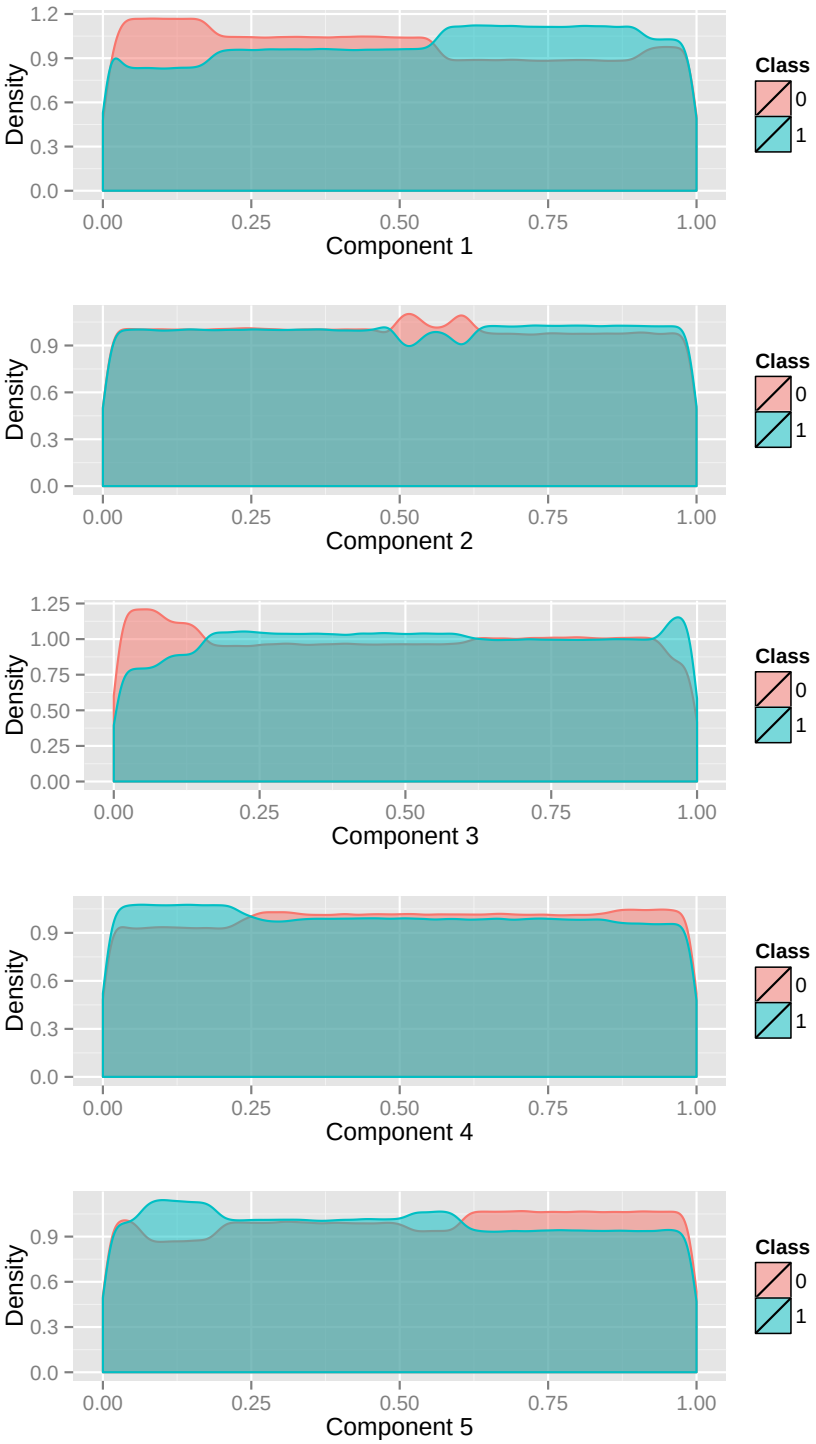


FIGURE 1. Distribution of values for each of the five components grouped by classes.

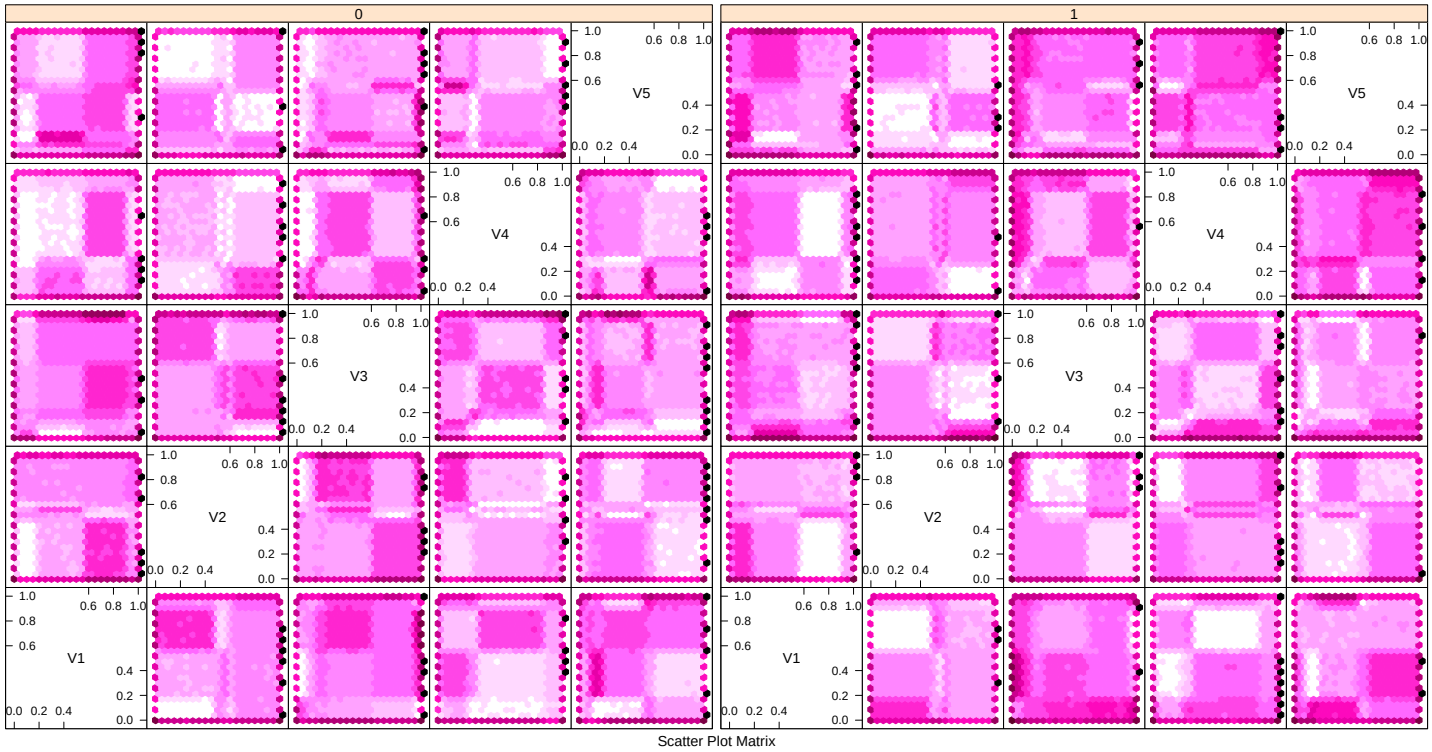


FIGURE 2. Distribution of values for each possible pair of components. The labels V1,...,V5 represent the five components. Plots for points in class 0 and 1 are respectively in the left and right frames. In each frame, a plot in row i and column j has component i on the y axis and component j on the x axis. Position $(1, 1)$ corresponds to bottom left

Each component is distributed almost uniformly, except for a few "patches" with slightly higher or lower distribution (Figure 1). For each component, the distribution of the two classes is roughly the dual of the other withing patches. For example, notice Component 2 in Figure 1: for any value within the central patch, points of Class 0 increase in density roughly as much as points of Class 1 decrease in density. The joint distribution of pairs of components (Figure 2) shows that points are organized in rectangular patches of higher or lower densities. Also in this case we can observe some duality: if a rectangular patch shows higher density for one class, it will show lower density for the other.

3.2. **Training the Quantizer and Calculating a Decision Rule.**

3.3. **Determining Number of Quantizing Intervals on Dataset 2.** Note that if our number of bins M is too large, we will suffer from memorization and over-fitting, while if M is too small we will suffer from over-generalization and under-fitting [9]. Thus our goal in determining the number of quantizing intervals is to find a happy medium which avoids both of these pitfalls.

3.4. **Determing Probability of Correct Identification on Dataset 3.**

4. Experimental Results on Other Decision Rules: A Comparative Analysis

Figure 3 summarizes the accuracy results for the comparison classifiers. The Decision Tree classifier achieves an almost perfect accuracy of 99.93%. The Nearest Neighbors is the second best performing classifier with around 87.86% accuracy. Other classifiers achieve less than 60% accuracy.

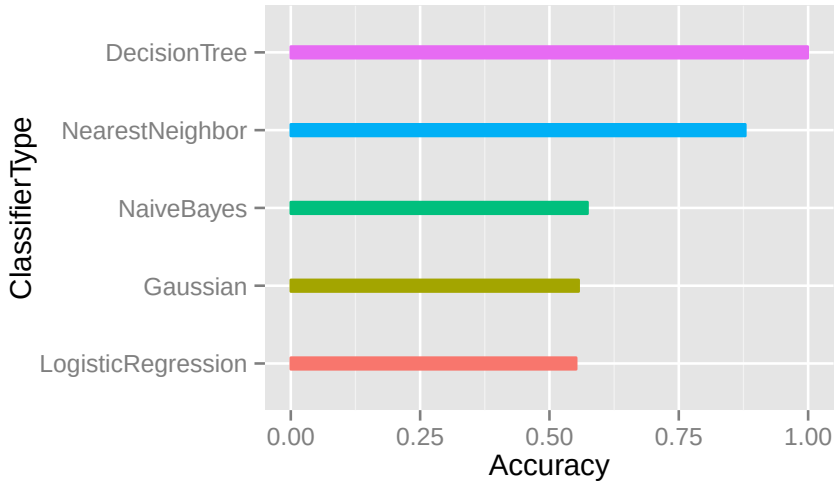


Figure 3. Accuracy results for comparison classifiers.

What follows are a few comments on the results above in light of the characteristics of the dataset (see Subsection 3.1). Although this final project explicitly required decision trees for comparison, this type of classifier would have been a natural choice in a domain with data similar to those provided. In fact, even basic types of decision trees find rectangular boundaries aligned with axis (see Figure 2). On the other hand, a classifier like a Naive Bayes would have been a poor choice. In fact, from Figures 1 and 2 we can see that the independence assumption does not hold. For example notice the patch $[0, 0.4] \times [0, 0.2]$ for class 0 and components 1 and 2 (i.e. plot at row 1 and column 2 in Figure 2). The density of class 0 points in that patch is nearly 0. But taking the respective projections of a point in the patch and computing the product of the separate distributions for components 1 and 2 would give us something close to 25% (see first two plots in Figure 1).

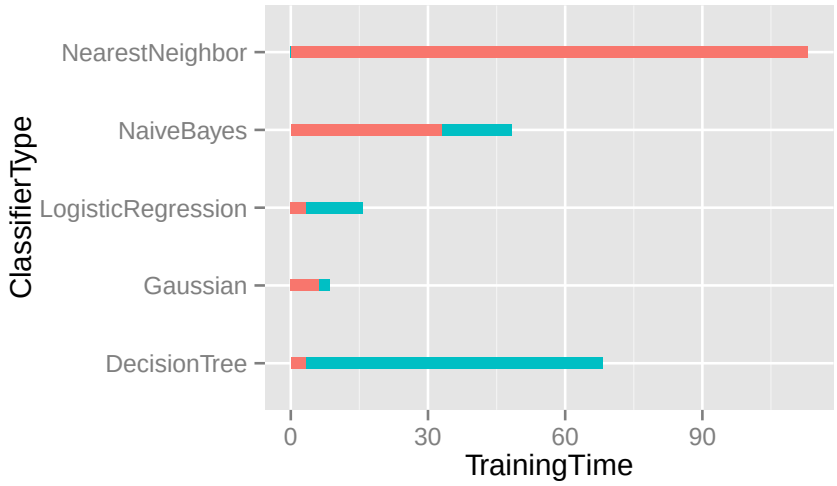


FIGURE 4. Training and testing time for comparison classifiers (in seconds)

5. CONCLUSION

We are the best team with the best paper 100% for everyone goodbye

References

[1] M. Bonollo, L. Di Persio, I. Olivia, & Semmoloni: A Quantization Approach to the Counterparty Credit Exposure Estimation. arXiv:1503.01754v1 [q-fin.PR], `http://arxiv.org/abs/1503.01754`.

[2] Johnathan N. Bradly & Christopher M. Brislawn: The FBI Wavelent/Scalar Quantization Standard for gray-scale fingerprint image compression. SPIE Proceedings, vol. 1961, *Visual Information Processing II*, Orlando, FL (1993), pp. 293-304.

[3] Ellis J. Clarke & Bruce A. Barton: Entropy and MDL Discretization of Continuous Variables for Bayesian Belief Networks. International Journal of Intelligent Systems, Vol. 15 (2000), pp. 61-92.

[4] William Oris Davis & Paul Howard Donaldson: *Signal Processing for Antisubmarine Warfare*. Defense Technical Information Center, 1975.

[5] J. Dougherty, R. Kohavi, & M. Sahami: Supervised and Unsupervised Discretization of Continuous Features. In A. Prieditis & S. J. Russell, eds. Work. Morgan Kaufmann (1995), pp. 194-202.

[6] Usama M. Fayyad & Keki B. Irani: Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. Proceedings of the International Joint Conference on Uncertainty in AI (1993), pp. 1022-1027.

[7] Robert M. Haralick: Discrete Bayes Pattern Recognition. Lecture Notes, Department of Computer Science, The Graduate Center, CUNY. `http://haralick.org/ML/discrete_bayes.pdf`

[8] Robert M. Haralick. Notes from Machine Learning course lecture on October 26, 2015. Department of Computer Science, The Graduate Center, CUNY.

[9] Robert M. Haralick: Quantization. Lecture Notes, Department of Computer Science, The Graduate Center, CUNY. `http://haralick.org/ML/quantization.pdf`

[10] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning, Second Edition.* Springer, 2009.

[11] Allen Gersho & Robert M. Grey: *Vector Quantization and Signal Compression.* Springer Science & Business Media (2012).

[12] Yoseph Linde, André s Buzo, & Robert M. Gray: An Algorithm for Vector Quantizer Design. IEEE Transactions on Communications, Vol. COM-28, No. 1 (1980), pp. 84-95.

[13] Michal Skubacz & Jaakko Hóllmen: Quantization of continuous input variables for binary classification. *Intelligent Data Engineering and Automated Learning - IDEAL 2000*, Volume 1983 of the series Lecture Notes in Computer Science (2002), pp 42-47.

[14] Andrew K. C. Wong & David K. Y. Chiu: Synthesizing Statistical Knowledge from Incomplete Mixed-Mode Data. IEEE Transations on Pattern Analysis and Machine Intelligence, Vol. PAMI-9, No. 6 (1987), pp. 796-805.