PERFORMANCE OF GAUSSIAN NAÏVE BAYES FOR CLASSIFICATION WITH DEPENDENCIES FROM ARCHEMEDIAN COPULA

By

Hugh E. Winston, B.S.

A Project Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Statistics

University of Alaska Fairbanks

May 2022

APPROVED:

Dr. Ronald Barry, Committee Chair
Dr. Scott Goddard, Committee Member
Dr. Margaret Short, Committee Member
Dr. John Rhodes, Chair
Department of Mathematics and Statistics

Abstract

Naive Bayes is an application of Bayes theorem in which the likelihood function is factored into marginals by making the assumption that the variables are independent. Naive Bayes is typically used for classification problems in which the goal is to find the class with the largest probability given the data on hand. When the data on hand are continuous real numbers we can further assume they are class conditionally normally distributed, which is a particular version of Naive Bayes called Gaussian Naive Bayes. This paper explores when Gaussian Naive Bayes classification problems work well vs when they do not. Typically when assumptions are not valid, valid conclusions cannot be drawn. However, Naive Bayes is known to be robust even when the independence assumption is not met. We show using simulations that binary classification accuracy of Naive Bayes is much more sensitive to differences in the class conditional marginal distributions than the correlation between predictors. Additionally we show that Naive Bayes completely fails when predictors are generated using a Gumbel copula and compare results with a general Bayes classifier and the K-Nearest Neighbors classifier.

1 Introduction

Assumptions play an important part in data analysis. In many cases, perhaps all, assumptions must be made in order to actually proceed with calculations and deductions. These assumption are useful, possibly allowing one to use previously devised theory or simplifying a complex aspect of the problem. However, what happens when those assumptions are not valid? What can we say about the conclusions we reach after assuming a false premise? In traditional logic we would have to say that our conclusions are most likely false if the assumptions we made to reach them were false. There are however surprising situations where an assumption is clearly false but this does not have as big an effect on the conclusions as we might think. In problems such as these, making a false assumption paradoxically *improves* the analysis. Naive Bayes is one such method.

Naive Bayes is a method for calculating the posterior probability of a response given a vector of predictor variables. The method involves assuming the conditional distributions for each variable are mutually independent, which is quite a strong assumption in that, for many real datasets it will be flat out false. For example, in Fisher's Iris dataset[14] the predictor variables are the lengths and widths of an Iris flowers petal and sepal. Clearly the length and width of a flowers petal are not independent measurements. It is by this assumption the method gets its name.

For a classification problem with K different classes, let C_k be the class of the i^{th} observation, where $i = 1, 2, ..n$ and let $\mathbf{x} = (x_1, x_2, ... x_n)$ be an instance vector of predictor variables. Then by Bayes Theorem;

$$P(C_k \mid \mathbf{x}) = \frac{P(C_k)P(\mathbf{x} \mid C_k)}{P(\mathbf{x})}$$

Since $P(\mathbf{x})$ doesn't depend on C_k, it only serves as a normalizing constant in the calculations and is the same for each class. Therefore, in the context of classification, it can be ignored and rather than equality we can replace with a "proportional to" statement;

$$P(C_k \mid \mathbf{x}) \propto P(C_k)P(\mathbf{x} \mid C_k)$$

If the joint class conditional distributions are known then $P(C_k \mid \mathbf{x})$ can be calculated exactly. Otherwise they need to be estimated or a simplifying assumption can be made to make the calculations easier or in certain situations even possible, namely that the predictors are mutually conditionally independent. Under this assumption the $P(\mathbf{x} \mid C_k)$ can be decomposed as the product of the class conditional marginals and, if they are known, $P(C_k \mid \mathbf{x})$ can be calculated using;

$$P(C_k \mid \mathbf{x}) \propto P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k)$$

A Bayes classifier can be built from this statement by calculating $P(C_k)\prod_{i=1}^{n} P(x_i \mid C_k)$ for each class, C_k, and choosing the largest value as the outputted predicted class. Bayes classifiers are known to be optimal if the prior and the likelihood can be calculated precisely[2]. Optimality, in this context, is to say that the probability of mis-classification is minimized when using a Bayes classifier, and therefore the Naive Bayes classifier will definitely be optimal when the independence assumption is true and the true prior is known[1]. A surprising result is that the Naive Bayes classifier can still be very accurate even when the independence assumption is false[4]. Therefore, in many cases the Naive Bayes classifier performs comparatively well compared to other more sophisticated classifiers, regardless of whether the assumption is met or not.

Naive Bayes can be used with a wide variety of data types. When the predictor variables are continuous and the output variable is discrete, we can replace probability statements in Bayes theorem with probability densities;

$$P(C_k \mid \mathbf{x}) = \frac{P(C_k) f(\mathbf{x} \mid C_k)}{f(\mathbf{x})}$$

$$P(C_k \mid \mathbf{x}) \propto P(C_k) \prod_{i=1}^{n} f(x_i \mid C_k)$$

Therefore Naive Bayes can be used when the class conditional marginal distribution of the data are known, or can be reasonable assumed. Deciding the distribution of the class conditional marginals gives many different versions of Naive Bayes, such as Bernoulli Naive Bayes, Multinomial Naive Bayes, and Gaussian Naive Bayes[5]. For the purposes of this paper we will assume the class conditional marginals are normally distributed, therefore we will be using Gaussian Naive Bayes.

In order to apply Gaussian Naive Bayes, the following parameters need to be estimated:

- $P(C_k)$: The probability of each individual class. This can be estimated by the maximum likelihood estimator, $\frac{n_k}{N}$, or provided as a categorical prior distribution.
- μ_{k_i}: The class conditional means, estimated using the class conditional sample means for class k and predictor i
- $\sigma^2_{k_i}$: The class conditional variances, estimated using the class conditional sample variances for class k and predictor i

Then the probability of each class given an instance of the data can then be estimated and a Bayes classifier can be built by taking the largest of these;

$$\widehat{C} = \arg\max_k \frac{n_k}{N} \prod_{i=1}^{n} \frac{1}{\sigma_{k_i}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_i - \mu_{k_i}}{\sigma_{k_i}}\right)^2}$$

Next we demonstrate that even with a high degree of dependency between the predictor variables, Gaussian Naive Bayes can still achieve satisfying prediction accuracy.

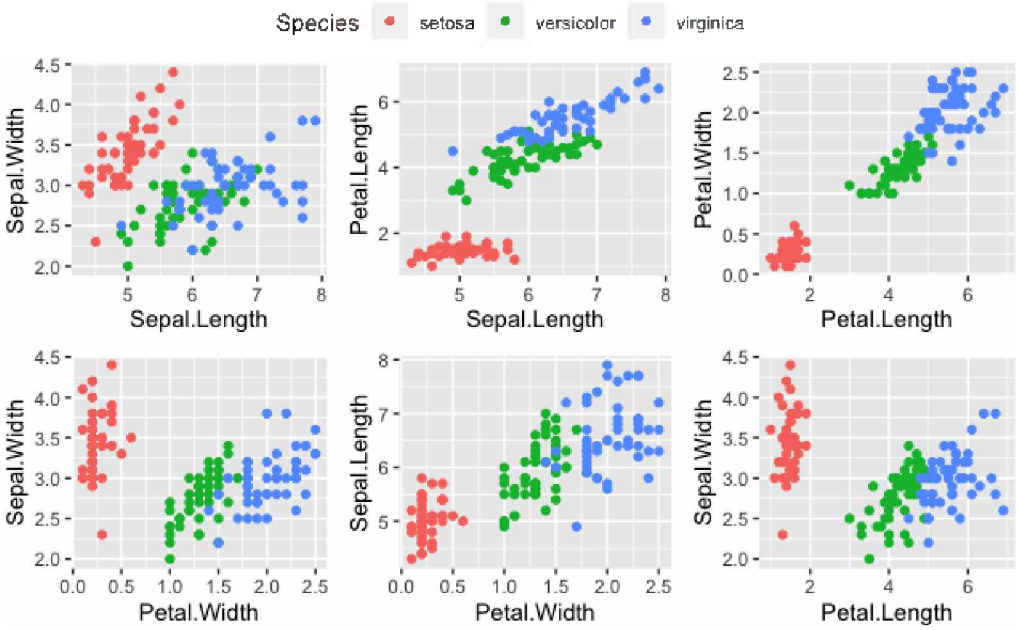
2 Demonstration

In order to demonstrate that Naive Bayes is still able to give satisfying accuracy performance in the face of feature dependencies we will use Sir Ronald Fisher's Iris dataset[14]. The data consists of measurements from three species of Iris flowers; Iris Setosa, Iris Virginica and Iris Versicolor. For each flower the petal length and width and the sepal length and width were measured. Intuitively we can reason that these four predictor variables are not conditionally independent since usually a flowers petals and sepals widths don't grow independently from their respective lengths. Analytically we can see from the scatterplots that there are most likely dependencies and the sample correlation matrices for each class are as follows.

I.Setosa	*s.len*	*s.wid*	*p.len*	*p.wid*
s.len	1	**0.74**	0.26	0.27
s.wid	**0.74**	1	0.17	0.23
p.len	0.26	0.17	1	0.33
p.wid	0.27	0.23	0.33	1

I.Versacolor	*s.len*	*s.wid*	*p.len*	*p.wid*
s.len	1	0.44	0.72	0.57
s.wid	0.44	1	0.26	0.36
p.len	0.72	0.26	1	**0.87**
p.wid	0.57	0.36	**0.87**	1

I.Virginica	*s.len*	*s.wid*	*p.len*	*p.wid*
s.len	1	0.46	**0.86**	0.33
s.wid	0.46	1	0.40	0.53
p.len	**0.86**	0.40	1	0.40
p.wid	0.33	0.53	0.40	1



However, in order to use Naive Bayes to classify these flower species based on the petal and sepal measurements we must assume that their length and widths are independent of each other within each class. Since we will use Gaussian Naive Bayes, we also verify that the class conditional features are approximately normal using normal probability QQ plots. The plots are shown in the appendix. It appears all class conditional features are approximately normal.

The data has 150 observations, 50 from each class. We randomly split the data into 120 observation used to train the model, and 30 observations used to test accuracy. Results on the test data are as follows.

```
Confusion Matrix and Statistics
                    Reference
Prediction     setosa versicolor virginica
setosa         15          0         0
versicolor      0          9         2
virginica       0          0         4

Overall Statistics
Accuracy : 0.9333
95% CI : (0.7793, 0.9918)
```

The model reaches 93% accuracy even with feature dependencies. As we can see, even when the independence assumption is not met, the Naive Bayes model can still produce satisfying prediction accuracy.

3 Dependence Structure from Copulas

In order to create a situation where Naive Bayes must fail we explore the concept of a copula, defined next.

Definition 1. A copula is a joint cumulative distribution function such that each one of the marginals is uniformly distributed across the interval from zero to one.

A copula can be constructed in a number of ways. We discuss a few below.

Theorem. *Let X be any continuous random variable with CDF $F_X(x)$. Then the random variable $Y = F_X(X) \sim U(0,1)$.*

Proof. Since X is a continuous random variable it's CDF is also continuous and $F_X^{-1}(y)$ exists. Then for $0 \leq y \leq 1$,

$$\begin{aligned} F_Y(y) &= P(Y \leq y) = P(F_X(X) \leq y) \\ &= P\left(F_X^{-1}(F_X(X)) \leq F_X^{-1}(y)\right) \\ &= P\left(X \leq F_X^{-1}(y)\right) = F_X(F_X^{-1}(y)) = y \end{aligned}$$

Differentiating to obtain the PDF of Y gives $f_Y(y) = I(0 < y < 1)$, which is the PDF of a uniformly distributed random variable across the interval from zero to one. □

Now, let $\mathbf{X} = (X_1, X_2, \ldots X_n)$ be a vector of random variables with continuous marginals and apply the following transformation to each element of the vector...

$$\mathbf{U} = (F_1(X_1), F_2(X_2), ..., F_n(X_n))$$

This results in a random vector of marginally distributed $U(0,1)$ random variables, and the joint CDF of such a vector is a copula.

$$C(u_1, u_2, \ldots u_n) = P(X_1 \leq F_1^{-1}(u_1), X_2 \leq F_2^{-1}(u_2), ..., X_n \leq F_n^{-1}(u_n))$$

Therefore, a copula is nothing more than any joint CDF with uniform marginals and, in fact, any joint CDF with continuous marginals can be written as a copula due to Sklar's Theorem[13].

Many useful copulas have been defined and explored as they give one the ability to model dependence structure without modeling the marginals. This idea is useful in areas such as quantitative financial analysis for risk assessments[15]. An especially useful class of copula are the Archimedean copulas because they admit an explicit closed form function. Archimedean copulas are defined through the use of a generator function with specific properties.

Definition 2. A bivariate copula with parameter θ and generator function $\psi(t)$ is called Archimedean if it can be written in the form...

$$C(u, v; \theta) = \psi^{-1}\left(\psi(u; \theta) + \psi(v; \theta); \theta\right)$$

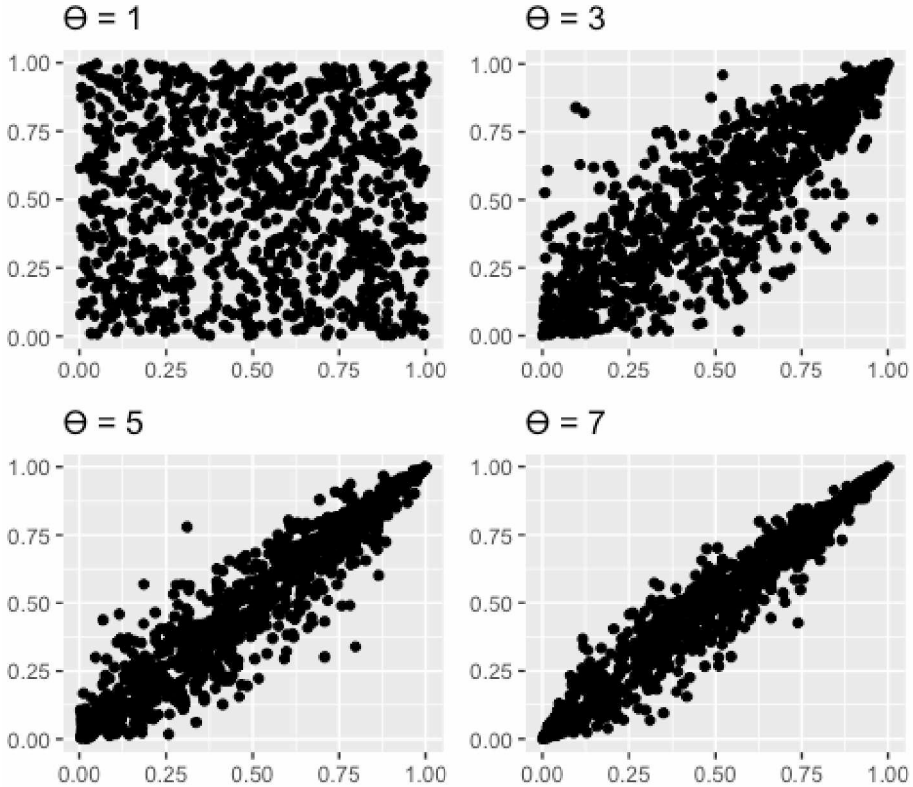
The generator function must have the following properties...

1. $\psi(1) = 0$
2. $\psi(t)$ is continuous and $\psi : [0, 1] \to [0, \infty)$
3. $\psi(t)$ is strictly decreasing
4. $\psi(t)$ is convex

For this paper we generate data from the bivariate Gumbel copula with dependence parameter θ and generator function $-(ln(t))^{\theta}$. Therefore the CDF is....

$$C(u, v; \theta) = e^{-\left((-ln(u))^{\theta} + (-ln(v))^{\theta}\right)^{\frac{1}{\theta}}}$$

The PDF can be calculated by partial differentiation. Data was generated using the R package *gumbel*[11]. Below we show 1000 randomly generated points from a Gumbel copula for $\theta = 1, 3, 5, 7$.



For $\theta = 1$ the variables are independent. As the value of θ increases, the strength of the dependence of the two variables increases. Naive Bayes relies entirely on the class conditional marginals to be different in order to correctly classify the target classes since if they are equal the probability estimates will be approximately equal. Therefore, data generated from the copula should force the algorithm to fail since each marginal distribution will be the same.

4 Methods

For the purposes of this paper we focus on binary classification, K = 2, and we artificially simulate the class conditional features as normally distributed real numbers. We show multiple situations where Gaussian Naive Bayes works well, with and without class conditional independence. In order to create a situation where Gaussian Naive Bayes preforms poorly we use a Gumbel copula.

For the simulation we generate $n = 800$ points for each of two classes labeled "1" and "2" in three different scenarios;

- truly independent predictors within each class
- predictors from a bivariate normal distribution with covariance matrix $\mathbf{\Sigma}$
- predictors from a Gumbel copula with dependence parameter θ

In each scenario there are two extreme cases. One where the data points in each class are linearly separable, and one where the they have been generated from the exact same distribution. In the first case we could achieve perfect accuracy trivially and in the second we would expect 50% accuracy, which is to say the model does no better than random guessing. Therefore, these two extreme cases will not be considered. Outside of those extremes we look at average classification accuracy over 1000 iterations for various distances between the class centers while the class standard deviations are equal and held fixed. For multivariate normal predictors, the correlation coefficients for each class are equal. The results are as follows.

5 Results

5.1 Independent Predictors

The table below represents average accuracy calculations over 1000 iterations at various distances between the class centers given by $\Delta\mu$.

$\Delta\mu$	Average accuracy
0.00	0.52
0.25	0.54
0.5	0.59
0.75	0.64
1.00	0.69
1.25	0.73
1.50	0.77
1.75	0.81
2.00	0.84
2.25	0.87
2.50	0.89
2.75	0.92
3.00	0.93

5.2 Multivariate Normal Predictors

Each of the tables below represents average accuracy calculations over 1000 iterations at various distances between the class centers given by $\Delta\mu$ while the value of ρ is held fixed and equal for both classes.

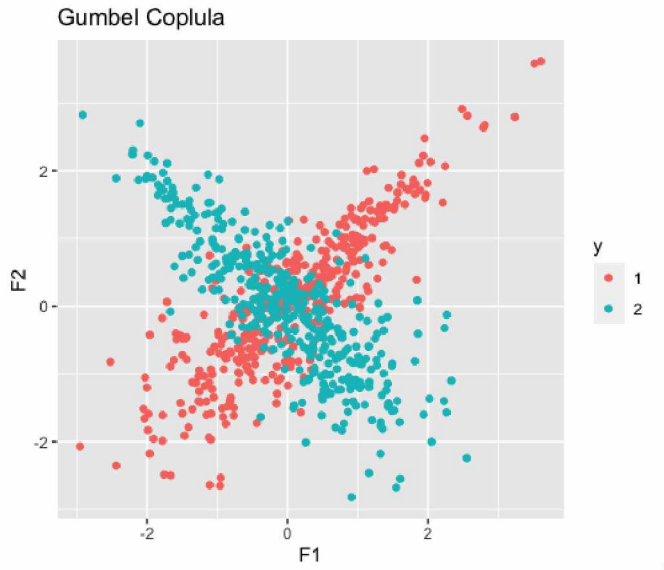
$\rho = 0.2$			$\rho = 0.4$	
$\Delta\mu$	Average accuracy		$\Delta\mu$	Average accuracy
0.00	0.49		0.00	0.49
0.25	0.54		0.25	0.54
0.5	0.59		0.5	0.59
0.75	0.64		0.75	0.64
1.00	0.68		1.00	0.69
1.25	0.73		1.25	0.73
1.50	0.77		1.50	0.77
1.75	0.81		1.75	0.81
2.00	0.84		2.00	0.84
2.25	0.87		2.25	0.87
2.50	0.89		2.50	0.89
2.75	0.91		2.75	0.92
3.00	0.93		3.00	0.93

$\rho = 0.6$			$\rho = 0.8$	
$\Delta\mu$	Average accuracy		$\Delta\mu$	Average accuracy
0.00	0.50		0.00	0.49
0.25	0.54		0.25	0.54
0.5	0.59		0.5	0.59
0.75	0.64		0.75	0.64
1.00	0.69		1.00	0.69
1.25	0.73		1.25	0.73
1.50	0.77		1.50	0.77
1.75	0.81		1.75	0.81
2.00	0.84		2.00	0.84
2.25	0.87		2.25	0.87
2.50	0.89		2.50	0.89
2.75	0.92		2.75	0.92
3.00	0.93		3.00	0.93

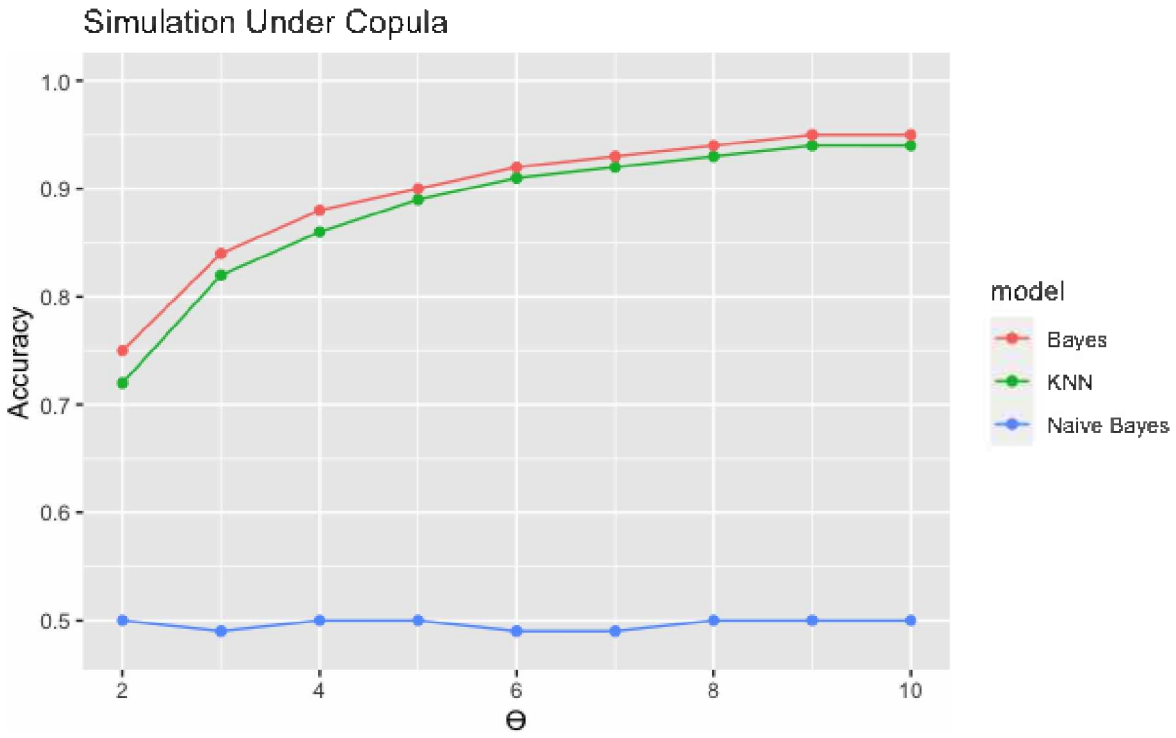
$\rho = 0.9$			$\rho = 0.99$	
$\Delta\mu$	Average accuracy		$\Delta\mu$	Average accuracy
0.00	0.50		0.00	0.50
0.25	0.54		0.25	0.54
0.5	0.60		0.5	0.60
0.75	0.65		0.75	0.65
1.00	0.69		1.00	0.69
1.25	0.73		1.25	0.73
1.50	0.77		1.50	0.77
1.75	0.81		1.75	0.81
2.00	0.84		2.00	0.84
2.25	0.87		2.25	0.87
2.50	0.89		2.50	0.89
2.75	0.92		2.75	0.92
3.00	0.93		3.00	0.93

5.3 Gumbel Copula Predictors

In this section we generate data from the bivariate Gumbel Copula and use a transformation so that that data are marginally standard normal. Then class 2 is horizontally translated to produce data that is classifiable as seen in the image below.



Average Accuracy	θ								
	2	3	4	5	6	7	8	9	10
Naive Bayes	0.50	0.49	0.50	0.50	0.49	0.49	0.50	0.50	0.50
Bayes	0.75	0.84	0.88	.90	0.92	0.93	0.94	0.95	0.95
KNN	0.72	0.82	0.86	0.89	0.91	0.92	0.93	0.94	0.94



6 Discussion

We can see that the level of class conditional correlation between the predictor variables barely makes a difference when determining the average classification accuracy using Naive Bayes. Even when the predictors are almost perfectly correlated, Naive Bayes can still achieve excellent accuracy results as long as the class centers are different. Additionally, we can see that Naive Bayes completely breaks down when the class conditional marginal distributions are the same, regardless of the dependence structure between the predictors. In order for Naive Bayes to have any predictive classification power, the class condition marginals must be, at least slightly, distinct. When data is generated from a copula Naive Bayes cannot perform at all because by definition each of its marginal distributions are the same. Using a general Bayes classifier we can see that when the dependance structure is taken into account we get much better average accuracy. However the real utility of the Naive Bayes method is that one does not need to know the joint distribution in order to build the model. In situations where the class conditional marginals are equal, another classification method should be considered. We can see that KNN outperforms Naive Bayes in these situations.

7 Conclusion

Naive Bayes turns out to be a very robust technique given that its main assumption is often violated. However in order for Naive Bayes to be able to have predictive power, the class conditional marginal distributions cannot be the same. Since data generated from a copula produces equal marginals, Naive Bayes is unable to be useful in that situation. Ultimately, it is important to evaluate the structure of data before or in tandem with deciding what methods one will use to perform the analysis.

References

[1] Zhang, Harry. (2004). The Optimality of Naive Bayes. Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004. 2.

[2] Domingos, Pedro & Pazzani, Michael. (1998). On the Optimality of the Simple Bayesian Classifier Under Zero-One Loss. Machine Learning. 29. 10.1023/A:1007413511361.

[3] Frank, Eibe & Trigg, Len & Holmes, Geoffrey & Witten, Ian. (2000). Naive Bayes for Regression (Technical Note).. Machine Learning. 41. 5-25. 10.1023/A:1007670802811.

[4] Rish, Irina. (2001). An Empirical Study of the Naïve Bayes Classifier. IJCAI 2001 Work Empir Methods Artif Intell. 3.

[5] Mccallum, Andrew & Nigam, Kamal. (2001). A Comparison of Event Models for Naive Bayes Text Classification. Work Learn Text Categ. 752.

[6] Rennie, Jason & Shih, Lawrence & Teevan, Jaime & Karger, David. (2003). Tackling the Poor Assumptions of Naive Bayes Text Classifiers.

[7] Metsis, Vangelis & Androutsopoulos, Ion & Paliouras, Georgios. (2006). Spam Filtering with Naive Bayes - Which Naive Bayes?. In CEAS.

[8] RStudio Team (2020). RStudio: Integrated Development for R. RStudio, PBC, Boston, MA URL http://www.rstudio.com/.

[9] H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

[10] Majka M (2019). naivebayes: High Performance Implementation of the Naive Bayes Algorithm in R. R package version 0.9.7, <URL: https://CRAN.R-project.org/package=naivebayes>.

[11] C. Dutang (2015), gumbel: package for Gumbel copula.

[12] Max Kuhn (2021). caret: Classification and Regression Training. R package version 6.0-90. https://CRAN.R-project.org/package=caret

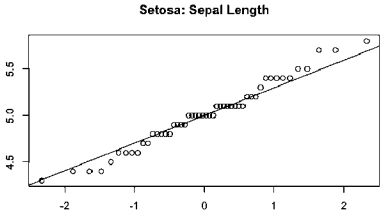
[13] Sklar, A. (1973). Random variables, joint distribution functions, and copulas. Kybernetika, 9, 449-460.

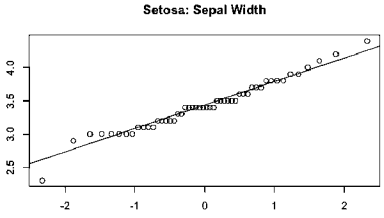
[14] Fisher, R.A.. (1936). The Use of Multiple Measurements in Taxonomic Problems. Annals of eugenics. 7. 179-188.

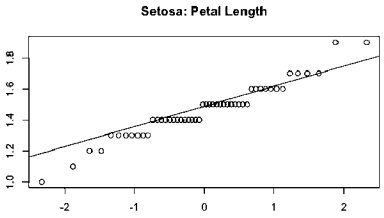
[15] Roger B. Nelsen. 2010. An Introduction to Copulas. Springer Publishing Company, Incorporated.

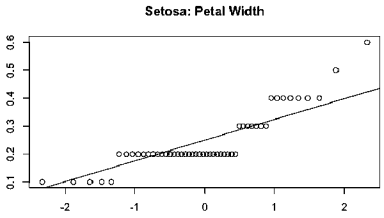
Appendix

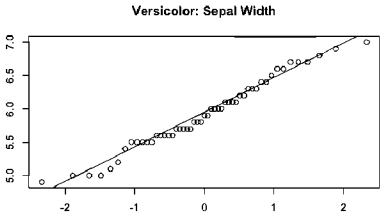
Normal QQ plots:

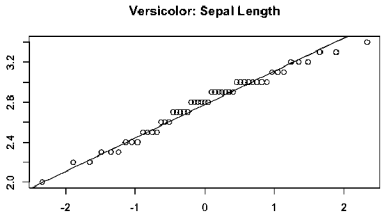


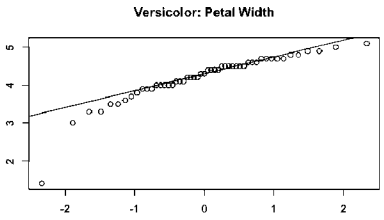


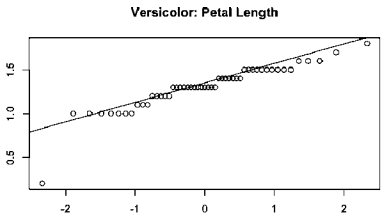


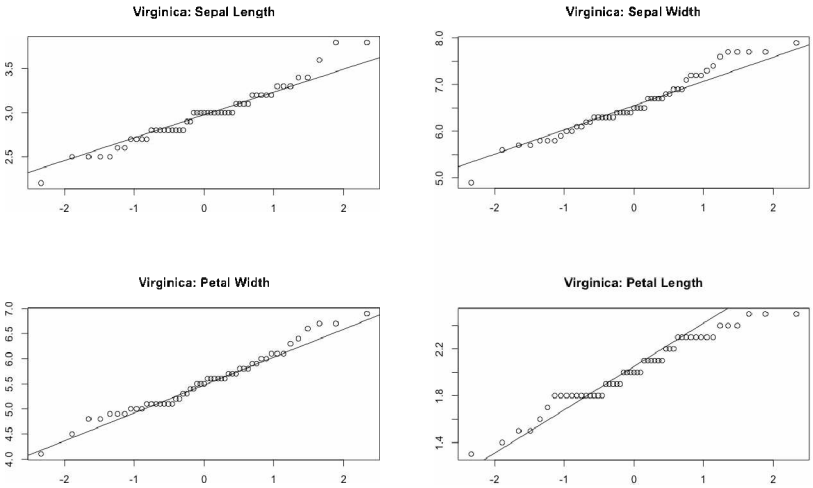











Virginica: Sepal Length
Virginica: Sepal Width
Virginica: Petal Width
Virginica: Petal Length

Code:

```
#MV Normal Simulation
#Author: Hugh Winston
library(MASS)
library(MBESS)
library(mvtnorm)
library(naivebayes)
library(ggplot2)

options(scipen = 999)#turn off scientific notation

ch <- seq(from=0, to=3, by=0.25)
rho <- .99

for(j in 1:length(ch)){
  reps = 1000
  history <- rep(NA, reps)

  for(i in 1:reps){
    n = 400 #data points in each class
    mu1 <- c(ch[j], 0) #class 1 center
    rho1 <- 0.99
    sig1 <- matrix(c(1,rho,rho, 1), 2, 2) #class 1 correlation matrix
    sd1 <- c(1,1) #class 1 standard deviations
    sig1 <- cor2cov(sig1, sd1) #class 1 covariance matrix


    mu2 <- c(0, 0)
    rho2 <- -0.9
    sig2 <- matrix(c(1,-rho,-rho, 1), 2, 2)
    sd2 <- c(1,1)
    sig2 <- cor2cov(sig2, sd2)

    y <- c(rep("1", n), rep("2", n))

    d <- as.data.frame(rbind(mvrnorm(n, mu1, sig1), mvrnorm(n, mu2, sig2)))
    d[3] <- y

    g <- ggplot(d, aes(V1, V2)) +
      geom_point(aes(col = V3)) +
      ggtitle("MV Norm")
    plot(g)

    s <- sample(1:nrow(d), 0.8*nrow(d), replace = F)
    d_train <- d[s, ]
    d_test <- d[-s, ]
```

```
    nb <- naive_bayes(V3 ~ ., data = d_train)
    history[i] <- mean(predict(nb, d_test[-3], type = "class") == d_test$V3)
  }
  print(mean(history))
}



#Copula Simulation
#Author: Hugh Winston
library(ggplot2)
library(gumbel)
library(naivebayes)
library(caret)
library(class)

reps = 1000 #number of iterations of simulation
theta1 = 10  #dependence parameter for class 1
theta2 = 10  #dependence parameter for class 2
history <- rep(NA, reps)        #Accuracy output of each run for Naive Bayes
history_KNN <- rep(NA, reps)    #Accuracy output of each run for K-NN
history_Bayes <- rep(NA, reps)  #Accuracy output of each run for general Bayes

for(i in 1:reps){

  n = 800

  x1 <- rgumbel(n/2, theta1)  #
  t1 <- qnorm(x1[ , 1])
  t2 <- qnorm(x1[ , 2])

  x2 <- rgumbel(n/2, theta2)
  t3 <- -qnorm(x2[ , 1])
  t4 <- qnorm(x2[ , 2])

  F1 <- c(t1, t3)
  F2 <- c(t2, t4)

  y <- c(rep("1", n/2), rep("2", n/2)) #Balanced classes, therefore P(C) = 0.5
  d <- data.frame(y = y, F1 = F1, F2 = F2)

  # g <- ggplot(d, aes(x = F1, y = F2)) +
  #   geom_point(aes(color = y)) +
  #   ggtitle("Gumbel Coplula")
  # plot(g)
  # h <- ggplot(d, aes(x = F1)) +
  #   geom_histogram()
  # plot(h)

  s <- sample(1:nrow(d), 0.8*nrow(d), replace = F)
  d_train <- d[s, ]
  d_test <- d[-s, ]
```

```
  nb <- naive_bayes(y ~ ., data = d_train) #NB model with all predictors assumed normal dist

  exact <- data.frame(one = NA, two = NA) #set up blank df

   for(j in 1:n){
     evidence <- (0.5)*dgumbel(pnorm(d[j, 2]), pnorm(d[j, 3]), alpha=theta1) + (0.5)*dgumbel(pnorm(-d[j
     exact[j, 1] <- ((0.5)*dgumbel(pnorm(d[j, 2]), pnorm(d[j, 3]), alpha=theta1)) / evidence
     exact[j, 2] <- ((0.5)*dgumbel(pnorm(-d[j, 2]), pnorm(d[j, 3]), alpha=theta2)) / evidence
   }

  history[i] <- mean(predict(nb, d_test[-1], type = "class") == d_test$y)
  history_KNN[i] <- mean(knn(d_train[-1], d_test[-1], as.factor(d_train[[1]]), k = 5) == d_test[[1]])
  history_Bayes[i] <- mean(as.numeric(exact[-s, 1] > exact[-s, 2]) == as.numeric(d_test[[1]] == 1))
}
mean(history)
mean(history_KNN)
mean(history_Bayes)
#######################################
#for ploting outside the main loop
g <- ggplot(d, aes(x = F1, y = F2)) +
  geom_point(aes(color = y)) +
  ggtitle("Gumbel Coplula")
plot(g)
h <- ggplot(d, aes(x = F1)) +
  geom_histogram()
plot(h)
#######################################
#for testing outside the main loop
exact <- data.frame(one = NA, two = NA)

for(j in 1:n){
  evidence <- (0.5)*dgumbel(pnorm(d[j, 2]), pnorm(d[j, 3]), alpha=theta1) + (0.5)*dgumbel(pnorm(-d[j, 2
  exact[j, 1] <- ((0.5)*dgumbel(pnorm(d[j, 2]), pnorm(d[j, 3]), alpha=theta1)) / evidence
  exact[j, 2] <- ((0.5)*dgumbel(pnorm(-d[j, 2]), pnorm(d[j, 3]), alpha=theta2)) / evidence
}

mean(as.numeric(exact[-s, 1] > exact[-s, 2]) == as.numeric(d_test[[1]] == 1))
#####################################################3
#results
t <- 2:10 #values of theta
n <- c(.5, .49, .5, .5, .49, .49, .5, .5, .5) #NB accuracy results
k <- c(.72, .82, .86, .89, .91, .92, .93, .94, .94) #KNN accuracy results
b <- c(.75, .84, .88, .9, .92, .93, .94, .95, .95) #General Bayes accuracy results

v <- data.frame(t=rep(t, 3), data=c(n,k,b), model=c(rep("Naive Bayes", 9), rep("KNN", 9), rep("Bayes",

g <- ggplot(v, aes(t, data)) +
  geom_point(aes(col=model)) +
  geom_line(aes(col=model)) +
  coord_cartesian(xlim=c(2, 10), ylim = c(0.48, 1)) +
  labs(x = "\u03F4", y = "Accuracy")
plot(g)
```

```r
#Independent Predictors Simulation
#Author: Hugh Winston
library(ggplot2)
library(naivebayes)

reps = 1000
history <- rep(NA, reps)

for(i in 1:reps){
  n = 800
  #With 2 classes and 2 predictors there are 4 conditional distributions. For Gaussian NB, each
  #one needs a mean and var.
  d_mu = 0
  d_sig = 3
  #Class 1
  x_11_true <- c(0, 1)
  x_21_true <- c(d_mu, 1 + d_sig)
  #Class 2
  x_12_true <- c(0, 1 + d_sig)
  x_22_true <- c(0, 1)

  #create 4 different normal dists with set parameters
  x_11 <- rnorm(n/2, mean =  x_11_true[1], sd = x_11_true[2])
  x_12 <- rnorm(n/2, mean =  x_12_true[1], sd = x_12_true[2])
  x_21 <- rnorm(n/2, mean =  x_21_true[1], sd = x_21_true[2])
  x_22 <- rnorm(n/2, mean =  x_22_true[1], sd = x_22_true[2])
  #two features that are a combination of two dists
  F1 <- c(x_11, x_12)
  F2 <- c(x_21, x_22)
  #balanced in each class for each feature
  y <- c(rep("1", n/2), rep("2", n/2))

  d <- data.frame(y = y, F1 = F1, F2 = F2)
  #train test split 80/20
  train <- sample(1:n, 0.8*n, replace = F)
  d_train <- d[train, ]
  d_test <- d[-train, ]

  #naive bayes model
  nb <- naive_bayes(y ~ ., data = d_train)
  history[i] <- mean(predict(nb, d_test[-1], type = "class") == d_test$y)
}
mean(history)


#plot F1 vs F2 to verify clustering
g <- ggplot(d, aes(x = F1, y = F2)) +
  geom_point(aes(col = y))
plot(g)


#custom naive bayes model
class1 <- d_train[d_train$y == "1", ]
```

```r
class2 <- d_train[d_train$y == "2", ]

p_class1 <- nrow(class1) / nrow(d_train)
p_class2 <- nrow(class2) / nrow(d_train)

#calculate dist parameters
c1_m <- apply(class1[-1], 2, "mean")
c1_v <- apply(class1[-1], 2, "var")
c2_m <- apply(class2[-1], 2, "mean")
c2_v <- apply(class2[-1], 2, "var")

#predict class on the test set assuming normal dist conditional on class
preds <- c()
for(i in 1:nrow(d_test)){
  t1 <- p_class1 *
    dnorm(d_test[i, 2], mean = c1_m[[1]], sd = sqrt(c1_v[[1]])) *
    dnorm(d_test[i, 3], mean = c1_m[[2]], sd = sqrt(c1_v[[2]]))
  t2 <- p_class2 *
    dnorm(d_test[i, 2], mean = c2_m[[1]], sd = sqrt(c2_v[[1]])) *
    dnorm(d_test[i, 3], mean = c2_m[[2]], sd = sqrt(c2_v[[2]]))
  if(t1 < t2) {
    preds[i] <- "2"
  }
  else{
    preds[i] <- "1"
  }
}
#test set accuracy
mean(preds == d_test[1])



#Naive Bayes on Iris dataset
#Author: Hugh Winston
library(datasets)
library(ggplot2)
library(ggpubr)
library(naivebayes)
library(caret)

i <- iris
plot(i[-5])

g <- ggplot(i, aes(x = Sepal.Length, y = Sepal.Width)) +
  geom_point(aes(color = Species))
h <- ggplot(i, aes(x = Sepal.Length, y = Petal.Length)) +
  geom_point(aes(color = Species))
k <- ggplot(i, aes(x = Petal.Length, y = Petal.Width)) +
  geom_point(aes(color = Species))
j <- ggplot(i, aes(x = Petal.Width, y = Sepal.Width)) +
  geom_point(aes(color = Species))
l <- ggplot(i, aes(x = Petal.Width, y = Sepal.Length)) +
```

```
  geom_point(aes(color = Species))
plot(g)
plot(h)
plot(k)
plot(j)
plot(l)
ggarrange(g, h, k, j, l, common.legend = T)

set <- cor(i[1:50, -5])
vers <- cor(i[50:100, -5])
virg <- cor(i[100:150, -5])

qqnorm(i[1:50, 1])
qqnorm(i[1:50, 2])
qqnorm(i[1:50, 3])
qqnorm(i[1:50, 4])
qqnorm(i[50:100, 1])
qqnorm(i[50:100, 2])
qqnorm(i[50:100, 3])
qqnorm(i[50:100, 4])
qqnorm(i[100:150, 1])
qqnorm(i[100:150, 2])
qqnorm(i[100:150, 3])
qqnorm(i[100:150, 4])

s <- sample(1:nrow(i), 0.8*nrow(i), replace = F)
i_train <- i[s, ]
i_test <- i[-s, ]

nb <- naive_bayes(Species ~ ., data = i_train)
summary(nb)
mean(predict(nb, i_test[-5], type = "class") == i_test$Species)
mean(predict(nb, i_train[-5], type = "class") == i_train$Species)

cm <- confusionMatrix(predict(nb, i_test[-5], type = "class"), i_test$Species)
cm
```