# Classification Modeling for Cacao Quality Using KNN and SVM

Carolina Garza, Mallory Hall, Morgan Hall, Skylar Mai, Justin Steadham, Ray Thomas

5/5/2021

## Introduction

Chocolate, a treat produced from roasted cacao beans, is one of the most popular foods in the world. Each year, Americans collectively consume more than 2.8 billion pounds of chocolate. Around the word, all the chocolate consumed amounts to a more than 100-billion-dollar industry. However, not all chocolate is created equal. The quality of chocolate varies depending on cacao percentage (darkness), bean origin, manufacturing companies, and supplementary ingredients such as sugar, butter, vanilla, salt, et cetera. Here, examining the Flavors of Cacao Chocolate Bar Ratings Database, we analyze what makes a chocolate bar better than another.

The objective of our project is to accurately predict how well-received a chocolate bar is (in other words, its rating, whether recommended or not) given predictors such as cocoa percentage, manufacturer (company name and its location), country of bean origin, and the presence of certain ingredients (beans, sugar or sweetener, cocoa butter, vanilla, lecithin, and salt.) It is an important goal to be able to anticipate whether a chocolate bar will be received positively or negatively because for consumers, they want to be aware if their purchase will be satisfactory; and for manufacturers, they want to ensure that they produce popular and profitable chocolate bars.

## About the Data

The Flavors of Cacao Chocolate Bar Ratings database is a collection of reviews of over 2,300 chocolate bars, ordered based on 10 metrics:

- **REF** (`REF`): Value associated with when the review was entered in the database, where higher value means more recent
- **Manufacturer** (`Manufacturer`): Name of the manufacturing company of the bar
- **Company Location** (`CompanyLocation`): The country in which the company is based
- **Review Date** (`ReviewDate`): The review publication year
- **Bean Origin** (`BeanOrigin`): The country where the cacao bean that comprises the bar is harvested
- **Specific Bean Origin or Bar Name** (`SpecificOrigin`): More specific location of the bean origin or brand name of the chocolate bars
- **Cocoa Percent** (`CocoaPercent`): The percentage of cocoa in the bar
- **Ingredients** (`Ingredients`): String indicating the number of ingredients followed by some or all of the following letters representing the bar's ingredients: beans (“B”), sugar (“S”), sweetener other than white cane or beet sugar ("S*“), cocoa butter (”C“), vanilla (”V“), lecithin (”L“), salt (”Sa")
- **Most Memorable Characteristics** (`Characteristics`): A summary review of the most memorable characteristics of the chocolate bar
- **Rating** (`Rating`): Score representing the quality of chocolate bar as rated by an expert reviewer in six categories: flavor, texture, after-melt, aroma, appearance, and overall subjective opinion. Aroma and appearance are scored in increments of 0.5 on a scale from 0.5 to 2.5, while texture, flavor, after-melt,

and overall opinion are scored in increments of 1 on a scale from 1 to 5. Ratings from the six categories are summed and normalized to a scale from 1.0 to 5.0. The rating system is defined as:

- 4.0 to 5.0: "Outstanding"
- 3.5 to 3.9: "Highly Recommended"
- 3.0 to 3.49: "Recommended"
- 2.0 to 2.9: "Disappointing"
- 1.0 to 1.9: "Unpleasant"

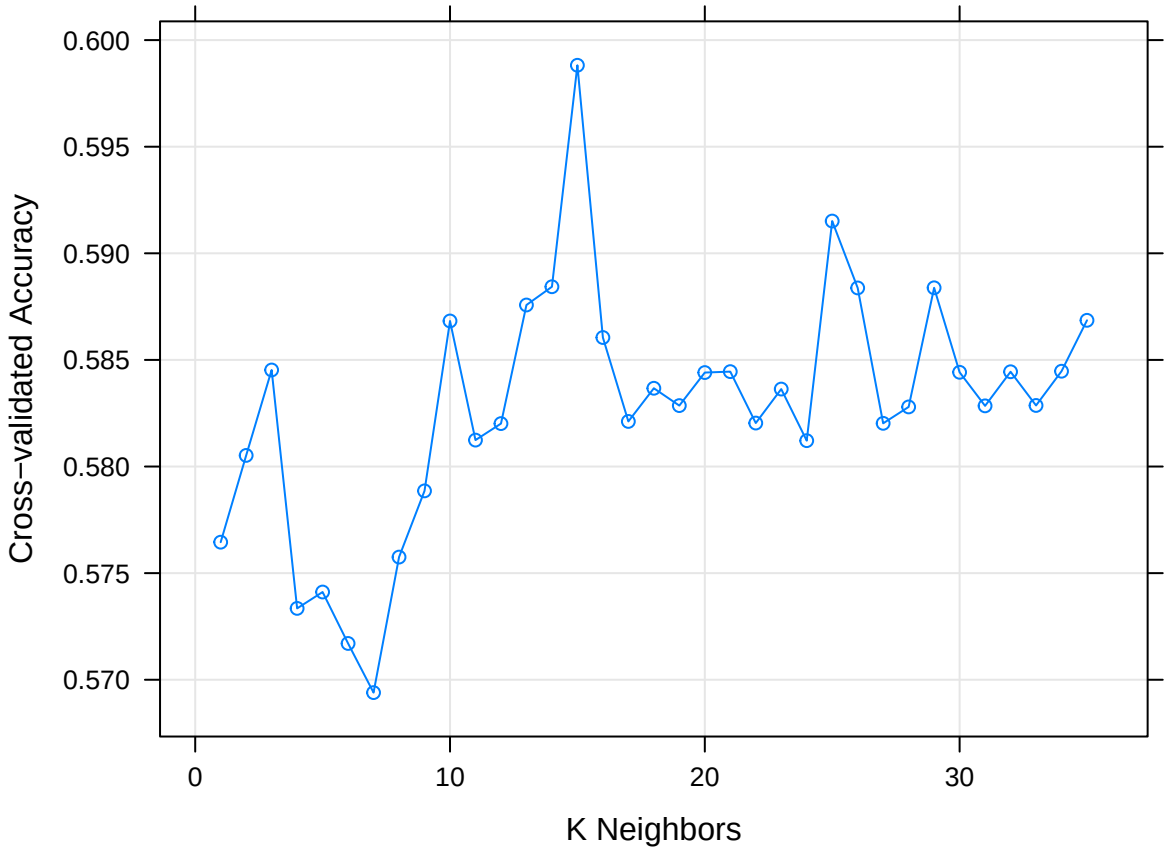
For the purposes of our analysis, we use a modified version of the database. First, we created a binary variable, `Recommended`, based on `Rating` such that it has a value of 1 if the chocolate bar has a rating of 3.0 or above, and a value of 0 otherwise. This redefinition enables us to perform classification tasks using K-nearest neighbors (KNN) and support vector machine (SVM) methods. Additionally, in order to manipulate `Ingredients`, we separated the variable into seven individual binary variables: `Beans`, `Sugar`, `Sweetener`, `CocoaButter`, `Vanilla`, `Lecithin`, and `Salt`, where the variable has a value of 1 if the chocolate bar has the corresponding ingredient and a value of 0 if otherwise. Additionally, given the diversity of `CompanyLocation` and `BeanOrigin`, we group observations by continents into two additional variables `Company` and `Origin`, respectively, and one-hot encode both variables. Finally, we omitted observations with missing values as well as variables `Ingredients`, `Characteristics`, `SpecificOrigin`, `Manufacturer`, `REF` due to the nature of their values.

#Methodology We will be using supervised learning methods to answer our questions, specifically using K-Nearest Neighbor (KNN) and Support Vector Machines (SVM) as our two models. The advantages of KNN are that it is a simple model to use, intuitive, has no assumptions, and provides relatively high accuracy. The advantages of SVM are that it allows us to obtain a general framework of enlarging the feature space for a support vector classifier leading to more efficient computations. For both the KNN and SVM approach, the goal is to obtain the lowest test error. The KNN approach will use K-fold cross validation which is computational, does not lose in estimation quality, and the variability in the error estimates is negligible. The SVM approach uses cross validation to acquire the optimal value of the parameter C. This is done by selecting a grid of values for C and calculating the test error based on the values on the grid. By doing so, the value C with the lowest test error can be obtained.

KNN CV Formula: Pr(Y = j | X = x0) = 1/K E(yi = j) i???N0
SVM Formula: f(x ??? ) = ??0 + X i???S ??iK(x ??? , xi) Linear kernel: K(xi , xj) = X p k=1 xik xjk Polynomial kernel: K(xi , xj) = (1 + hxi , xji)ˆd = (1 + p k=1 xik xjk )ˆd Radial kernel: K(xi , xj) = exp(????? p k=1 (xik ??? xjk ) 2 ), ?? > 0

## K-Nearest Neighbors (Mallory Hall, Morgan Hall, Skylar Mai)

We will begin by fitting our dataset using the K-Nearest Neighbors (KNN) Cross Validation model. We will train and scale the model to determine which predictors will generate an accurate test error rate based on the conditions presented. Variables such as Beans will be excluded because it has a standard deviation of 0. We defined the predictors as dummy variables - "yes" or "no" in order to classify them based on how well they predicted K observations. By resampling our data and tuning our parameters, we will determine which K value obtains the highest accuracy rate and reduces the error rate. The results are shown below:




Plotting accuracy against $K$ for $K$ in the range of 1 to 20, we see that $K = 15$ yielded the best accuracy. Thus, we fitted a KNN model using the value as our $K$ and obtained a confusion matrix of the results as follows. After obtaining the optimal value, we will fit a KNN model using our best parameter (optimal) value to create a confusion matrix. There were 177 out of the 314 observations that were correctly classified, and it obtained a misclassification error rate of 43.6%.

```
set.seed(1)
cacaoKnn <- knn(train = cacaoTrain, test = cacaoTest,
                cl = cacao[sample, "Recommended"], k = knnCv$bestTune)
```

| Predicted | Actual | |
|---|---|---|
| | No | Yes |
| No | 39 | 32 |
| Yes | 106 | 137 |

Next, we scaled the data in order to ensure that our data is measured in the same units. This will allow us to clean up the data and easily obtain a more accurate misclassification error rate. There were 986 out of the 1570 observations that were correctly classified, and the misclassification rate produced was 37.2%.

With the KNN model, we obtained a misclassification error rate of 0.4394904.

```
# Fitting on full dataset
cacaoTrainFull <- scale(cacao[, -ncol(cacao)])
cacaoTestFull <- scale(cacao[, -ncol(cacao)],
                       center = attr(cacaoTrainFull, "scaled:center"),
                       scale = attr(cacaoTrainFull, "scaled:scale"))
set.seed(1)
cacaoKnnFull <- knn(train = cacaoTrainFull,
                    cl = cacao[, "Recommended"],
                    test = cacaoTestFull,
                    k = knnCv$bestTune)
```

| Predicted | Actual | |
|---|---|---|
| | No | Yes |
| No | 258 | 120 |
| Yes | 439 | 753 |

The results of both of the matrices validated that scaling the data improves the error rate. In addition, both of these values appeared to outperform the cross validation model indicating that the scale model is the best at predicting the most profitable and saleable chocolate bar amongst consumers.

When the model is fitted on the full data set, we obtained a misclassification rate of 0.356051

## Support Vector Machine (Carolina Garza, Justin Steadham, Ray Thomas)

As with the KNN model, we begin by randomly splitting the dataset into 80% for training and 20% for testing. For consistency sake, we are using the trimmed dataset from above. We then proceed to tune the data to a range of hyperparameters appropriate to each of the three kernel types. For the linear kernel, we must determine the best cost hyperparameter. This determines what budget we must leave to allow observations to fall onto the wrong side of the hyperplane. For the polynomial kernel, we must determine both the cost and the degree hyperparameters. In determining the degree, we must exercise caution since a larger degree will result in greater computational costs. For the radial kernel, we must once again determine the cost along with the gamma hyperparameter. Determining the gamma tells us the radius separating the different types.

```
# create an 80% sample
set.seed(1)
sample <- sample(seq_len(nrow(cacao)), size = nrow(cacao) * 0.80)
cacaoTrain <- cacao[sample, ]
cacaoTest <- cacao[-sample, ]
```

Since the SVM function scales the data by default, we do not need manually scale the training/testing split.

```
#Linear Kernel Tuning
set.seed(1)
svm.linear.tune <- tune(method = svm,
                        Recommended~.,
                        data = cacaoTrain,
                        kernel = "linear",
```

```
                        ranges = list(cost = c(0.001,0.01,0.1,1,5,10,100,1000)))
svm.linear.tune$best.parameters
  cost
3  0.1
svm.linear.tune$best.performance
[1] 0.4020698
```

```
#Polynomial Kernel Tuning
set.seed(1)
svm.polynomial.tune <- tune(method = svm,
                      Recommended~.,
                      data = cacaoTrain,
                      kernel = "polynomial",
                      ranges = list(cost = c(0.001,0.01,0.1,1,5,10,100,1000),
                                    degree=c(2,3,4,5)))
svm.polynomial.tune$best.parameters
  cost degree
7  100      2
svm.polynomial.tune$best.performance
[1] 0.3941651
```

```
#Radial Kernel Tuning
set.seed(1)
svm.radial.tune <- tune(method = svm,
                     Recommended~.,
                     data = cacaoTrain,
                     kernel="radial",
                     ranges = list(cost=c(0.001,0.01,0.1,1,5,10,100,1000),
                                   gamma=c(0.5,1,2,3,4)))
svm.radial.tune$best.parameters
  cost gamma
7  100   0.5
svm.radial.tune$best.performance
[1] 0.4083873
```

When looking at the summary outputs of the three kernels, we find what error rates resulted from the best performances of each kernel. The optimal linear kernel results in an error rate of 0.4020698 when the $cost = 0.1$. The optimal polynomial kernel results in an error rate of 0.3941651 when the $cost = 100$, and the $degree = 2$. And the optimal radial kernel results in an error rate of 0.4083873 when the $cost = 100$, and the $gamma = 0.5$.

Once we have attained the best model for each kernel, we proceed to use the testing set to validate which kernel yields the lowest errors when faced with new data.

```
##Linear Kernel:
#Training Error
mean(predict(svm.linear.tune$best.model) != cacaoTrain$Recommended)
[1] 0.4020701

#Confusion Matrix
set.seed(1)
pred.linear.test = predict(svm.linear.tune$best.model, newdata=cacaoTest)
```

| Predicted | True | |
|---|---|---|
| | No | Yes |
| No | 47 | 98 |
| Yes | 18 | 151 |

```
#Testing Error
mean(predict(svm.linear.tune$best.model, newdata = cacaoTest) != cacaoTest$Recommended)
[1] 0.3694268

#False Positive Rate
(linear.FPR = 98/(47+98))
[1] 0.6758621

#False Negative Rate
(linear.FNR = 18/(18+151))
[1] 0.1065089
```

```
##Polynomial Kernel:
#Training Error
mean(predict(svm.polynomial.tune$best.model) != cacaoTrain$Recommended)
[1] 0.3328025

#Confusion Matrix
set.seed(1)
pred.polynomial.test = predict(svm.polynomial.tune$best.model, newdata=cacaoTest)
```

| Predicted | True | |
|---|---|---|
| | No | Yes |
| No | 41 | 104 |
| Yes | 28 | 141 |

```
#Testing Error
mean(predict(svm.polynomial.tune$best.model, newdata = cacaoTest) != cacaoTest$Recommended)
[1] 0.4203822

#False Positive Rate
(polynomial.FPR = 104/(41+104))
[1] 0.7172414

#False Negative Rate
(polynomial.FNR = 28/(28+141))
[1] 0.1656805
```

```
#Radial Kernel:
#Training Error
mean(predict(svm.radial.tune$best.model) != cacaoTrain$Recommended)
[1] 0.2444268

#Confusion Matrix
```

```
set.seed(1)
pred.radial.test = predict(svm.radial.tune$best.model, newdata=cacaoTest)
```

| Predicted | True | |
|---|---|---|
| | No | Yes |
| No | 57 | 88 |
| Yes | 43 | 126 |

```
#Testing Error
mean(predict(svm.radial.tune$best.model, newdata = cacaoTest) != cacaoTest$Recommended)
[1] 0.4171975

#False Positive Rate
(radial.FPR = 88/(57+88))
[1] 0.6068966

#False Negative Rate
(radial.FNR = 43/(43+126))
[1] 0.2544379
```

As we can see from the above results, there are multiple considerations for us to take into account. We see that the training errors range from a low of 0.2444268 for the radial kernel, to a high of 0.4020701 for the linear kernel. We also see that the testing errors range from a low of 0.3694268 for the linear kernel, to a high of 0.4203822 for the polynomial kernel. Furthermore, it is crucial to consider the False Positive Rate (FPR) for each of the three kernels. This is important for future sales since it a low FPR will result in greater customer retention. Since a false positive in this case means that poor quality cocoa has been misclassified to be good quality, customers who buy that chocolate that has been inadvertantly made from that poor cocoa will be less likely to buy from the company in the future. The FPR ranges from 0.6068966 for the radial kernel, to a high of 0.7172414 for the linear kernel. This tells us that even though the linear kernel yields the lowest error rate on the validation set, it might not be the most ideal model since it will likely result in high rates of poor quality cacao being considered as good cacao. And since the error rates are not significantly different from each other, the radial kernel will yield the most well-rounded SVM model for making recommendations.

The final Radial SVM model and results are as follows:

```
#Final Radial SVM Model
set.seed(1)
svm.radial.fit <- tune(method = svm,
                       Recommended~.,
                       data = cacao,
                       kernel ="radial",
                       cost = 100,
                       gamma = 0.5)
#Confusion Matrix
set.seed(1)
pred.radial.fit = predict(svm.radial.fit$best.model, newdata=cacao)
```

| Predicted | True | |
|---|---|---|
| | No | Yes |
| No | 352 | 345 |
| Yes | 62 | 811 |

```
#Misclassification Error
mean(predict(svm.radial.fit$best.model) != cacao$Recommended)
[1] 0.2592357
```

## Model Comparison

By focusing on the KNN model and the SVM model using the radial kernel, we can see that the misclassification rate of our KNN model is 0.356051 and the misclassifcation rate of the SVM model is 0.2592357. Although SVM can be more computationally demanding, the misclassification rate was lower in the Radial SVM model. Because of this, we have chosen the Radial SVM model as our final model.

## Conclusion

We determined that the Radial SVM model would be best indicator for predicting whether or not a chocolate is marketable. Our goal was to create a model that accurately determined the quality of cacao based on certain predictors and access whether or not it would be saleable to consumers. The Radial SVM model yielded the lowest test error rate compared to the KNN model. There was some discrepancies between some of the models and the results produced when it came to quality of the chocolate and buyer and seller integrity. For example, in the SVM model, the linear kernel produced the lowest test error rate; however, its false positive rate masked the poor quality of the chocolate. This can lead to a decrease in customer loyalty due to people buying chocolate that they expect to be of higher quality, but in truth, did not meet the standards in terms of quality and taste.