Chocolate Bar Rating Class Project

HarvardX Data Science Capstone

Joseph Cayetano

2022-10-10

1. Introduction/Overview/Executive Summary

There's no doubt that Americans love chocolate. In fact, Americans consume about 2.8 billion pounds of chocolate each year. To put into perspective, the average American eats about 12 pounds of chocolate each year. That's equivalent to 1 pound per month, or the amount in 9 Hershey's chocolate bars. However, people from different countries prefer different kinds of chocolate and not all chocolate bars are created equal.

1.1 Dataset

For this project, we will be using the Chocolate Bar Ratings dataset from Kaggle to create visualizations of the different variables of the chocolate bars and develop predictive machine learning models. The dataset contains expert ratings of about 1,700 chocolate bars, along with information on what country they came from, the amount of cocoa they have, the type of chocolate bean used and where the beans were grown.

The ratings in the dataset were collected by Brady Brelinksi, who is one of the founders of the Manhattan Chocolate Society. To see more of Brelinksi's work regarding how chocolates are made, visit FlavorsOfCacao.com.

1.2 Goal of the Project

The goal of this project is to develop the best machine learning model that accurately predicts the Chocolate Bar Rating Class. We will train and test 3 different predictive models. The model that gets the highest accuracy score will be considered the best.

The rating class for all the chocolate bars in the dataset is influenced by the Flavors of Cacao Rating System.

Rating Scale:

5 = Elite (Transcending beyond the ordinary limits)

4 = Premium (Superior flavor development, character, and style)

3 = Satisfactory(3.0) to praiseworthy(3.75) (well made with special qualities)

2 = Disappointing (Passable but contains at least one significant flaw)

1 = Unpleasant (mostly unpalatable)

1.3 Key Steps Performed

We will be executing these following steps to achieve the goal of the project:

Step 1: Download the Chocolate Bar Ratings dataset.

Step 2: Clean, explore, and create visualizations of the dataset. We will create visualizations of the different variables of the chocolate bars. Doing this, we will be able to determine the features that we will use for our machine learning models.

Step 3: Split the dataset into training and testing sets. The training set will make up about 80 percent of the data, while the testing set will make up about 20 percent of the data.

Step 4: Build and train the machine learning models. Then, evaluate the accuracy of their predictions by using the test set.

Step 5: Discuss the results and provide a conclusion.

2. Methods/Analysis

We will begin the analysis by downloading the Chocolate Bar Ratings dataset. After that, we will remove non printable characters from the data to avoid potential issues that may arise when calculating the count for each variable.

```
# Defining the URL for the Chocolate Bar Ratings dataset.
choco_data_raw <- "https://raw.githubusercontent.com/josephcayetano/HarvardX-DataScience-CYOP/main/flav

# Downloading the Chocolate Bar Ratings dataset and removing non printable characters.
choco <- read_csv(gsub("[^[:print:]]","",choco_data_raw),
                            na = c("","  ","NA"))
```

2.1 Data Exploration

To get familiar with the dataset, We will pull a few rows of it as seen below. The dataset has 9 columns or variables, and they are:

Company (Maker-if known)	Specific Bean Origin or Bar Name	REF	Review Date	Cocoa Percent	Company Location	Rating	Bean Type	Broad Bean Origin
A. Morin	Agua Grande	1876	2016	63%	France	3.75		Sao Tome
A. Morin	Kpime	1676	2015	70%	France	2.75		Togo
A. Morin	Atsane	1676	2015	70%	France	3.00		Togo
A. Morin	Akata	1680	2015	70%	France	3.50		Togo
A. Morin	Quilla	1704	2015	70%	France	3.50		Peru
A. Morin	Carenero	1315	2014	70%	France	2.75	Criollo	Venezuela

Company (Maker-if known) – The name of the company that produces the chocolate bar.

Specific Bean Origin or Bar Name – The specific place of where the bar came from or the bar's name.

REF – The higher the value, the more recent it was entered in the database.

Review Date – The publication date of the review.

Cocoa Percent – The percentage of cocoa of the bar.

Company Location – The country where the company is in.

Rating – The expert rating for the bar (Scale of 1 – 5, with 5 being the best).

Bean Type – The kind of bean used, if provided.

Broad Bean Origin – The broad place of where the bean came from.

The dataset contains 1795 ratings and 9 variables.

```
## [1] 1795    9
```

The column names look a bit messy. Let's fix the column names not only to make them more readable, but also to make it easier for us to reference them in the code.

```
# Changing column names for consistency.
colNames <- c("CompanyMaker",
                  "ChocoName",
                  "Reference",
                  "ReviewDate",
                  "CocoaPercent",
                  "CompanyLocation",
                  "Rating",
                  "BeanType",
                  "BroadBeanOrigin")

names(choco) <- colNames

# Fixed column names.
names(choco)
```

```
## [1] "CompanyMaker"    "ChocoName"       "Reference"       "ReviewDate"
## [5] "CocoaPercent"    "CompanyLocation" "Rating"          "BeanType"
## [9] "BroadBeanOrigin"
```

We will remove the extra whitespaces and replace all empty values with NA.

```
# Removing extra whitespaces.
choco <- choco %>%
  mutate(across(where(is.character), str_trim))

# Replacing all empty values with NA.
choco <- choco %>%
  mutate_all(na_if, "")
```

There is a lot of missing values in the data, and they are all from the BeanType and BroadBeanOrigin columns.

```
## # A tibble: 9 x 2
##   `Column Name`   `Num of Missing Values`
##   <chr>                            <int>
## 1 CompanyMaker                         0
## 2 ChocoName                            0
## 3 Reference                            0
## 4 ReviewDate                           0
## 5 CocoaPercent                         0
## 6 CompanyLocation                      0
## 7 Rating                               0
## 8 BeanType                           888
## 9 BroadBeanOrigin                     74
```

2.2 Data Cleaning

Looking at the values for the CompanyLocation column, we see some inconsistencies in the data. For example, there's an "Ecuador" and "Eucador". There's a "Nicaragua" and "Niacragua". Also, "Domincan Republic" is misspelled when it should be "Dominican Republic."

```
##  [1] "Amsterdam"          "Argentina"          "Australia"
##  [4] "Austria"            "Belgium"            "Bolivia"
##  [7] "Brazil"             "Canada"             "Chile"
## [10] "Colombia"           "Costa Rica"         "Czech Republic"
## [13] "Denmark"            "Domincan Republic"  "Ecuador"
## [16] "Eucador"            "Fiji"               "Finland"
## [19] "France"             "Germany"            "Ghana"
## [22] "Grenada"            "Guatemala"          "Honduras"
## [25] "Hungary"            "Iceland"            "India"
## [28] "Ireland"            "Israel"             "Italy"
## [31] "Japan"              "Lithuania"          "Madagascar"
## [34] "Martinique"         "Mexico"             "Netherlands"
## [37] "New Zealand"        "Niacragua"          "Nicaragua"
## [40] "Peru"               "Philippines"        "Poland"
## [43] "Portugal"           "Puerto Rico"        "Russia"
## [46] "Sao Tome"           "Scotland"           "Singapore"
## [49] "South Africa"       "South Korea"        "Spain"
## [52] "St. Lucia"          "Suriname"           "Sweden"
## [55] "Switzerland"        "U.K."               "U.S.A."
## [58] "Venezuela"          "Vietnam"            "Wales"
```

We will fix all the problematic values in the CompanyLocation column for consistency.

```
# Fixing the inconsistencies in the CompanyLocation column
choco$CompanyLocation[choco$CompanyLocation == "Domincan Republic"] <- "Dominican Republic"
choco$CompanyLocation[choco$CompanyLocation == "Eucador"] <- "Ecuador"
choco$CompanyLocation[choco$CompanyLocation == "Niacragua"] <- "Nicaragua"
choco$CompanyLocation[choco$CompanyLocation == "Amsterdam"] <- "Netherlands"
choco$CompanyLocation[choco$CompanyLocation == "St. Lucia"] <- "Saint Lucia"
choco$CompanyLocation[choco$CompanyLocation == "Sao Tome"] <- "Sao Tome and Principe"
choco$CompanyLocation[choco$CompanyLocation == "Scotland"] <- "United Kingdom"
choco$CompanyLocation[choco$CompanyLocation == "U.K."] <- "United Kingdom"
choco$CompanyLocation[choco$CompanyLocation == "U.S.A."] <- "United States of America"
choco$CompanyLocation[choco$CompanyLocation == "Wales"] <- "United Kingdom"
choco$CompanyLocation[choco$CompanyLocation == "Martinique"] <- "France"
```

There's also a lot of inconsistent values in the column BroadBeanOrigin. It's clear that there are some rows with multiple bean origins delimited by ",", "/", "&", or "and". We will only use "|" as the delimiter for the rows with multiple bean origin values to make the data more consistent.

```
##  [1] "Africa, Carribean, C. Am."     "Australia"
##  [3] "Belize"                        "Bolivia"
##  [5] "Brazil"                        "Burma"
##  [7] "Cameroon"                      "Carribean"
##  [9] "Carribean(DR/Jam/Tri)"         "Central and S. America"
## [11] "Colombia"                      "Colombia, Ecuador"
## [13] "Congo"                         "Cost Rica, Ven"
```

```
## [15] "Costa Rica"                    "Cuba"
## [17] "Dom. Rep., Madagascar"         "Domincan Republic"
## [19] "Dominican Rep., Bali"          "Dominican Republic"
## [21] "DR, Ecuador, Peru"             "Ecuador"
## [23] "Ecuador, Costa Rica"           "Ecuador, Mad., PNG"
## [25] "El Salvador"                   "Fiji"
## [27] "Gabon"                         "Ghana"
## [29] "Ghana & Madagascar"            "Ghana, Domin. Rep"
## [31] "Ghana, Panama, Ecuador"        "Gre., PNG, Haw., Haiti, Mad"
## [33] "Grenada"                       "Guat., D.R., Peru, Mad., PNG"
## [35] "Guatemala"                     "Haiti"
## [37] "Hawaii"                        "Honduras"
## [39] "India"                         "Indonesia"
## [41] "Indonesia, Ghana"              "Ivory Coast"
## [43] "Jamaica"                       "Liberia"
## [45] "Mad., Java, PNG"               "Madagascar"
## [47] "Madagascar & Ecuador"          "Malaysia"
## [49] "Martinique"                    "Mexico"
## [51] "Nicaragua"                     "Nigeria"
## [53] "Panama"                        "Papua New Guinea"
## [55] "Peru"                          "Peru(SMartin,Pangoa,nacional)"
## [57] "Peru, Belize"                  "Peru, Dom. Rep"
## [59] "Peru, Ecuador"                 "Peru, Ecuador, Venezuela"
## [61] "Peru, Mad., Dom. Rep."         "Peru, Madagascar"
## [63] "Philippines"                   "PNG, Vanuatu, Mad"
## [65] "Principe"                      "Puerto Rico"
## [67] "Samoa"                         "Sao Tome"
## [69] "Sao Tome & Principe"           "Solomon Islands"
## [71] "South America"                 "South America, Africa"
## [73] "Sri Lanka"                     "St. Lucia"
## [75] "Suriname"                      "Tanzania"
## [77] "Tobago"                        "Togo"
## [79] "Trinidad"                      "Trinidad-Tobago"
## [81] "Trinidad, Ecuador"             "Trinidad, Tobago"
## [83] "Uganda"                        "Vanuatu"
## [85] "Ven, Bolivia, D.R."            "Ven, Trinidad, Ecuador"
## [87] "Ven., Indonesia, Ecuad."       "Ven., Trinidad, Mad."
## [89] "Ven.,Ecu.,Peru,Nic."           "Venez,Africa,Brasil,Peru,Mex"
## [91] "Venezuela"                     "Venezuela, Carribean"
## [93] "Venezuela, Dom. Rep."          "Venezuela, Ghana"
## [95] "Venezuela, Java"               "Venezuela, Trinidad"
## [97] "Venezuela/ Ghana"              "Vietnam"
## [99] "West Africa"
```

```
# Fixing the inconsistencies in the BroadBeanOrigin column.
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Africa, Carribean, C. Am."] <- "Western Africa|Caribbean
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Burma"] <- "Myanmar"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Carribean"] <- "Caribbean"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Carribean(DR/Jam/Tri)"] <- "Dominican Republic|Jamaica|T
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Central and S. America"] <- "Meso-America|South America
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Colombia, Ecuador"] <- "Colombia|Ecuador"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Congo"] <- "Republic of the Congo"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Cost Rica, Ven"] <- "Costa Rica|Venezuela"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Dom. Rep., Madagascar"] <- "Dominican Republic|Madagasca
```

```
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Dominican Rep., Bali"] <- "Dominican Republic|Indonesia"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "DR, Ecuador, Peru"] <- "Dominican Republic|Ecuador|Peru"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Ecuador, Costa Rica"] <- "Ecuador|Costa Rica"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Ecuador, Mad., PNG"] <- "Ecuador|Madagascar|Papua New Gu
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Ghana & Madagascar"] <- "Ghana|Madagascar"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Ghana, Domin. Rep"] <- "Ghana|Dominican Republic"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Ghana, Panama, Ecuador"] <- "Ghana|Panama|Ecuador"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Gre., PNG, Haw., Haiti, Mad"] <- "Grenada|Papua New Gui
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Guat., D.R., Peru, Mad., PNG"] <- "Guatemala|Dominican
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Indonesia, Ghana"] <- "Indonesia|Ghana"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Mad., Java, PNG"] <- "Madagascar|Indonesia|Papua New Gu
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Madagascar & Ecuador"] <- "Madagascar|Ecuador"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Peru(SMartin,Pangoa,nacional)"] <- "Peru"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Peru, Belize"] <- "Peru|Belize"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Peru, Dom. Rep"] <- "Peru|Dominican Republic"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Peru, Ecuador"] <- "Peru|Ecuador"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Peru, Ecuador, Venezuela"] <- "Peru|Ecuador|Venezuela"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Peru, Mad., Dom. Rep."] <- "Peru|Madagascar|Dominican Re
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Peru, Madagascar"] <- "Peru|Madagascar"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "PNG, Vanuatu, Mad"] <- "Papua New Guinea|Vanuatu|Madagas
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Trinidad, Ecuador"] <- "Trinidad and Tobago|Ecuador"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Ven, Bolivia, D.R."] <- "Venezuela|Bolivia|Dominican Rep
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Ven, Trinidad, Ecuador"] <- "Venezuela|Trinidad and Toba
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Ven., Indonesia, Ecuad."] <- "Venezuela|Indonesia|Ecuado
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Ven., Trinidad, Mad."] <- "Venezuela|Trinidad and Tobago
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Ven.,Ecu.,Peru,Nic."] <- "Venezuela|Ecuador|Peru|Nicarag
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Venez,Africa,Brasil,Peru,Mex"] <- "Venezuela|Western Afr
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Venezuela, Carribean"] <- "Venezuela|Caribbean"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Venezuela, Dom. Rep."] <- "Venezuela|Dominican Republic
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Venezuela, Ghana"] <- "Venezuela|Ghana"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Venezuela, Java"] <- "Venezuela|Indonesia"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Venezuela, Trinidad"] <- "Venezuela|Trinidad and Tobago
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Venezuela/ Ghana"] <- "Venezuela|Ghana"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Domincan Republic"] <- "Dominican Republic"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Hawaii"] <- "South Pacific"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Principe"] <- "Sao Tome and Principe"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Sao Tome"] <- "Sao Tome and Principe"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Sao Tome & Principe"] <- "Sao Tome and Principe"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "St. Lucia"] <- "Saint Lucia"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "South America, Africa"] <- "South America|Western Africa
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Tanzania"] <- "United Republic of Tanzania"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Tobago"] <- "Trinidad and Tobago"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Trinidad"] <- "Trinidad and Tobago"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Trinidad, Tobago"] <- "Trinidad and Tobago"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Trinidad-Tobago"] <- "Trinidad and Tobago"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Venezuela"] <- "Venezuela"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Vietnam"] <- "Vietnam"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "Martinique"] <- "Caribbean"
choco$BroadBeanOrigin[choco$BroadBeanOrigin == "West Africa"] <- "Western Africa"
```

Looking at the values for the BeanType column, some of them contain sub-varieties. For example, For the value "Criollo (Porcelana)", the "(Porcelana)" is the sub-variety in this value. Basically, the sub-varieties are the names with parentheses. We will simply remove these sub-varieties since we are only going to focus on the main varieties of the cacao plant, which are forastero, criollo, trinitario, and nacional.

```
##  [1] "Amazon"                   "Amazon mix"
##  [3] "Amazon, ICS"              "Beniano"
##  [5] "Blend"                    "Blend-Forastero,Criollo"
##  [7] "CCN51"                    "Criollo"
##  [9] "Criollo (Amarru)"         "Criollo (Ocumare 61)"
## [11] "Criollo (Ocumare 67)"     "Criollo (Ocumare 77)"
## [13] "Criollo (Ocumare)"        "Criollo (Porcelana)"
## [15] "Criollo (Wild)"           "Criollo, +"
## [17] "Criollo, Forastero"       "Criollo, Trinitario"
## [19] "EET"                      "Forastero"
## [21] "Forastero (Amelonado)"    "Forastero (Arriba)"
## [23] "Forastero (Arriba) ASS"   "Forastero (Arriba) ASSS"
## [25] "Forastero (Catongo)"      "Forastero (Nacional)"
## [27] "Forastero (Parazinho)"    "Forastero(Arriba, CCN)"
## [29] "Forastero, Trinitario"    "Matina"
## [31] "Nacional"                 "Nacional (Arriba)"
## [33] "Trinitario"               "Trinitario (85% Criollo)"
## [35] "Trinitario (Amelonado)"   "Trinitario (Scavina)"
## [37] "Trinitario, Criollo"      "Trinitario, Forastero"
## [39] "Trinitario, Nacional"     "Trinitario, TCGA"
```

We will also add another main variety called "Blend" because there are values with more than one main variety. For example, "Trinitario, Nacional", "Blend-Forastero, Criollo", "Trinitario (85% Criollo)", etc.

```
# Grouping the values from the BeanType column by main bean variety.
forastero <- c("Forastero",
                    "Forastero (Amelonado)",
                    "Forastero (Arriba)",
                    "Forastero (Arriba) ASS",
                    "Forastero (Arriba) ASSS",
                    "Forastero (Catongo)",
                    "Forastero (Parazinho)")

criollo <- c("Criollo",
             "Criollo (Amarru)",
             "Criollo (Ocumare)",
             "Criollo (Ocumare 61)",
             "Criollo (Ocumare 67)",
             "Criollo (Ocumare 77)",
             "Criollo (Porcelana)",
             "Criollo (Wild)")

trinitario <- c("Trinitario",
                "Trinitario (Amelonado)",
                "Trinitario (Scavina)")

nacional <- c("Forastero (Nacional)",
                   "Nacional",
                   "Nacional (Arriba)")
```

```
blend <- c("Amazon",
                  "Amazon mix",
                  "Amazon, ICS",
                  "Blend",
                  "Blend-Forastero,Criollo",
                  "Criollo, +",
                  "Criollo, Forastero",
                  "Criollo, Trinitario",
                  "Forastero, Trinitario",
                  "Forastero(Arriba, CCN)",
                  "Trinitario (85% Criollo)",
                  "Trinitario, Criollo",
                  "Trinitario, Forastero",
                  "Trinitario, Nacional",
                  "Trinitario, TCGA",
                  "Beniano",
                  "CCN51",
                  "EET",
                  "Matina")

choco$BeanType[which(choco$BeanType %in% forastero)] <- "Forastero"
choco$BeanType[which(choco$BeanType %in% criollo)] <- "Criollo"
choco$BeanType[which(choco$BeanType %in% trinitario)] <- "Trinitario"
choco$BeanType[which(choco$BeanType %in% nacional)] <- "Nacional"
choco$BeanType[which(choco$BeanType %in% blend)] <- "Blend"
```

Additionally, we will insert the value "Blend" into rows that have missing values in the BeanType column only if: there is more than one country in the BroadBeanOrigin column, or the name of the chocolate bar contains "Blend", "blend" or ",".

```
# Inserting the value "Blend" into rows that have n/a values
# in the BeanType column only if
#     - More than one country in the BroadBeanOrigin column
#     - Name of the chocolate bar contains "Blend", "blend" or ",".
choco$BeanType[is.na(choco$BeanType) & str_detect(choco$BroadBeanOrigin, "\\|")] <- "Blend"
choco$BeanType[is.na(choco$BeanType) & str_detect(choco$ChocoName, "blend")] <- "Blend"
choco$BeanType[is.na(choco$BeanType) & str_detect(choco$ChocoName, "Blend")] <- "Blend"
choco$BeanType[is.na(choco$BeanType) & str_detect(choco$ChocoName, "\\,")] <- "Blend"
```

We will create a new variable called RatingClass, as it will be used in the visualization section.

```
# Creating a new variable called RatingClass, as it will be used in the visualization section.
choco <- choco %>%
  mutate(RatingClass = case_when(Rating >= 1.00 & Rating <= 1.75  ~ "1-Unpleasant",
                                 Rating >= 2.00 & Rating <= 2.75  ~ "2-Disappointing",
                                 Rating >= 3.00 & Rating <= 3.75  ~ "3-Satisfactory",
                                 Rating >= 4.00 & Rating <= 4.75  ~ "4-Premium",
                                 Rating > 4.75  ~ "5-Elite"))
```

Looking back at the values from the BroadBeanOrigin column, we used the "|" as the delimiter for the rows with multiple bean origin values. We will now separate the values and place each one in its own row to have a more consistent estimate.

```
# Separating multiple bean origin values using the separate_rows function.
choco <- choco %>%
  separate_rows(BroadBeanOrigin,
                sep = "\\|",
                convert = FALSE)
```

Taking a look at the values from the CocoaPercent column, all of them have a percent sign. It will be difficult to calculate numeric values with strings so we will remove the percent signs and convert the column to numeric type. Also, the percentages will be rounded to the nearest integer.

```
# Removing the percent (%) sign from the CocoaPercent column
# and converting the column to numeric type.
# Percentages will be rounded to the nearest integer.
choco$CocoaPercent <- as.numeric(sub("%", "", choco$CocoaPercent, fixed = TRUE))
choco$CocoaPercent <- round(choco$CocoaPercent, digits = 0)
```

The Rating column will be converted to numeric type, and the rest of the columns will be converted to factor type.

```
# Converting the Rating column to numeric type.
choco$Rating <- as.numeric(choco$Rating)

# Converting the rest of the columns to factor type.
choco$CompanyMaker <- as.factor(choco$CompanyMaker)
choco$CompanyLocation <- as.factor(choco$CompanyLocation)
choco$BeanType <- as.factor(choco$BeanType)
choco$BroadBeanOrigin <- as.factor(choco$BroadBeanOrigin)
choco$ReviewDate <- as.factor(choco$ReviewDate)
choco$RatingClass <- as.factor(choco$RatingClass)
```

After cleaning the data, let's look at our improved dataset.

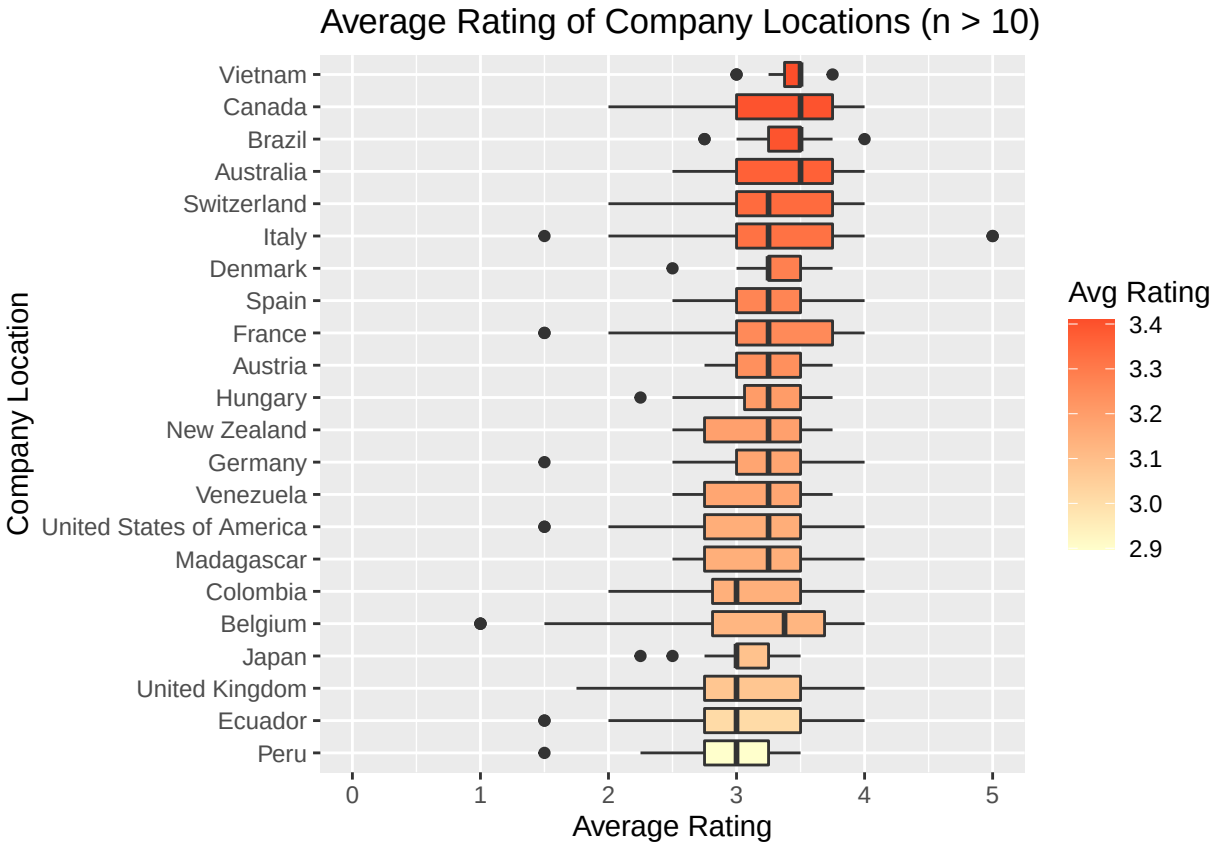
CompanyMaker	ChocoName	Reference	ReviewDate	CocoaPercent	CompanyLocation	Rating	BeanType	BroadBeanOrigin	RatingClass
A. Morin	Agua Grande	1876	2016	63	France	3.75	NA	Sao Tome and Principe	3-Satisfactory
A. Morin	Kpime	1676	2015	70	France	2.75	NA	Togo	2-Disappointing
A. Morin	Atsane	1676	2015	70	France	3.00	NA	Togo	3-Satisfactory
A. Morin	Akata	1680	2015	70	France	3.50	NA	Togo	3-Satisfactory
A. Morin	Quilla	1704	2015	70	France	3.50	NA	Peru	3-Satisfactory
A. Morin	Carenero	1315	2014	70	France	2.75	Criollo	Venezuela	2-Disappointing

2.3 Data Visualization

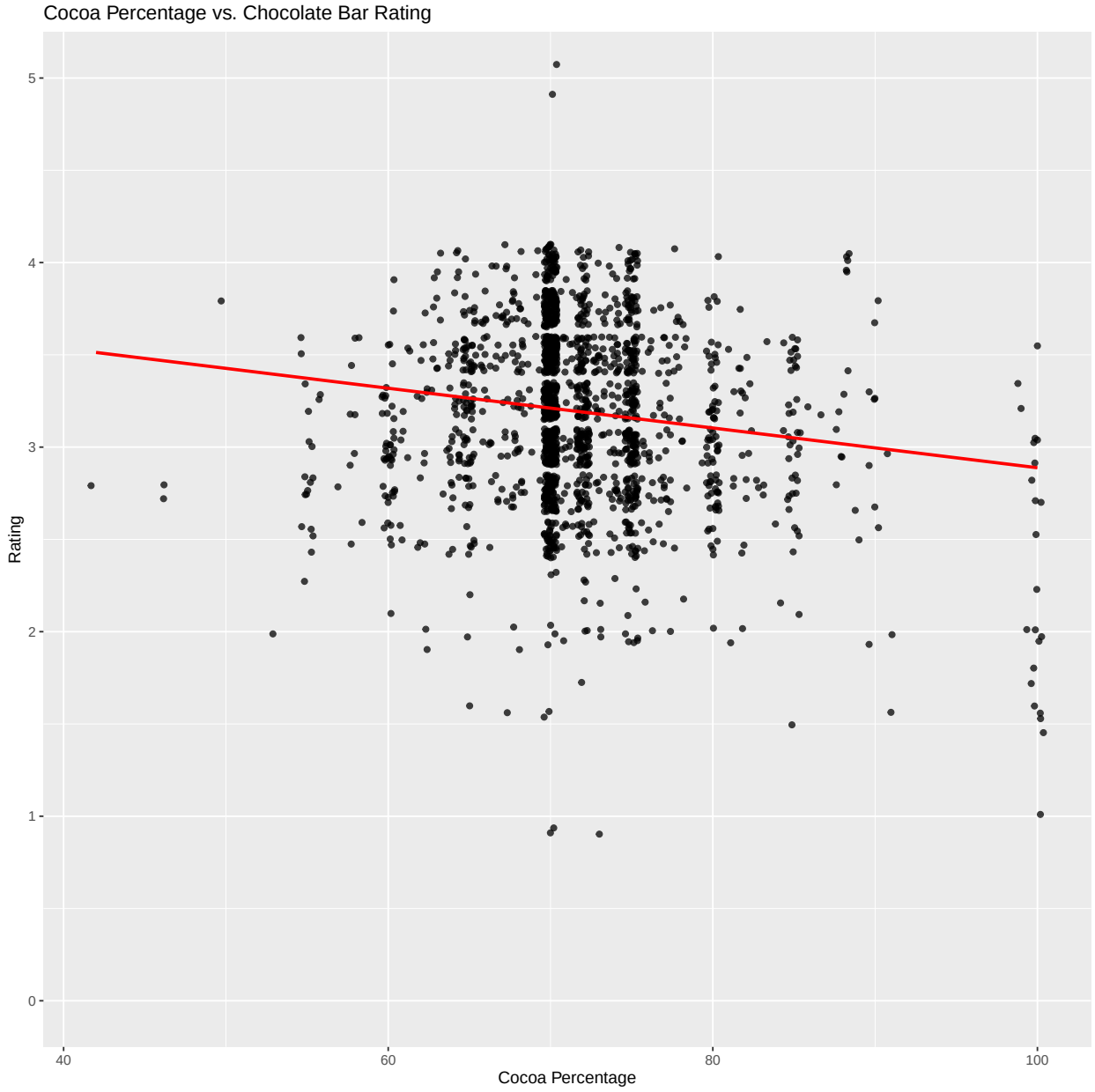
The table shows that the company, Amedei, produces the highest average rating of chocolate bars with 3.85. It is then followed by Idilio (felchlin) with an average rating of 3.78. Companies that have at least 10 ratings are only included in the table.

CompanyMaker	NumRating	AvgRating
Amedei	13	3.846154
Idilio (Felchlin)	10	3.775000
Soma	67	3.664179
Arete	22	3.534091
Smooth Chocolator, The	16	3.515625

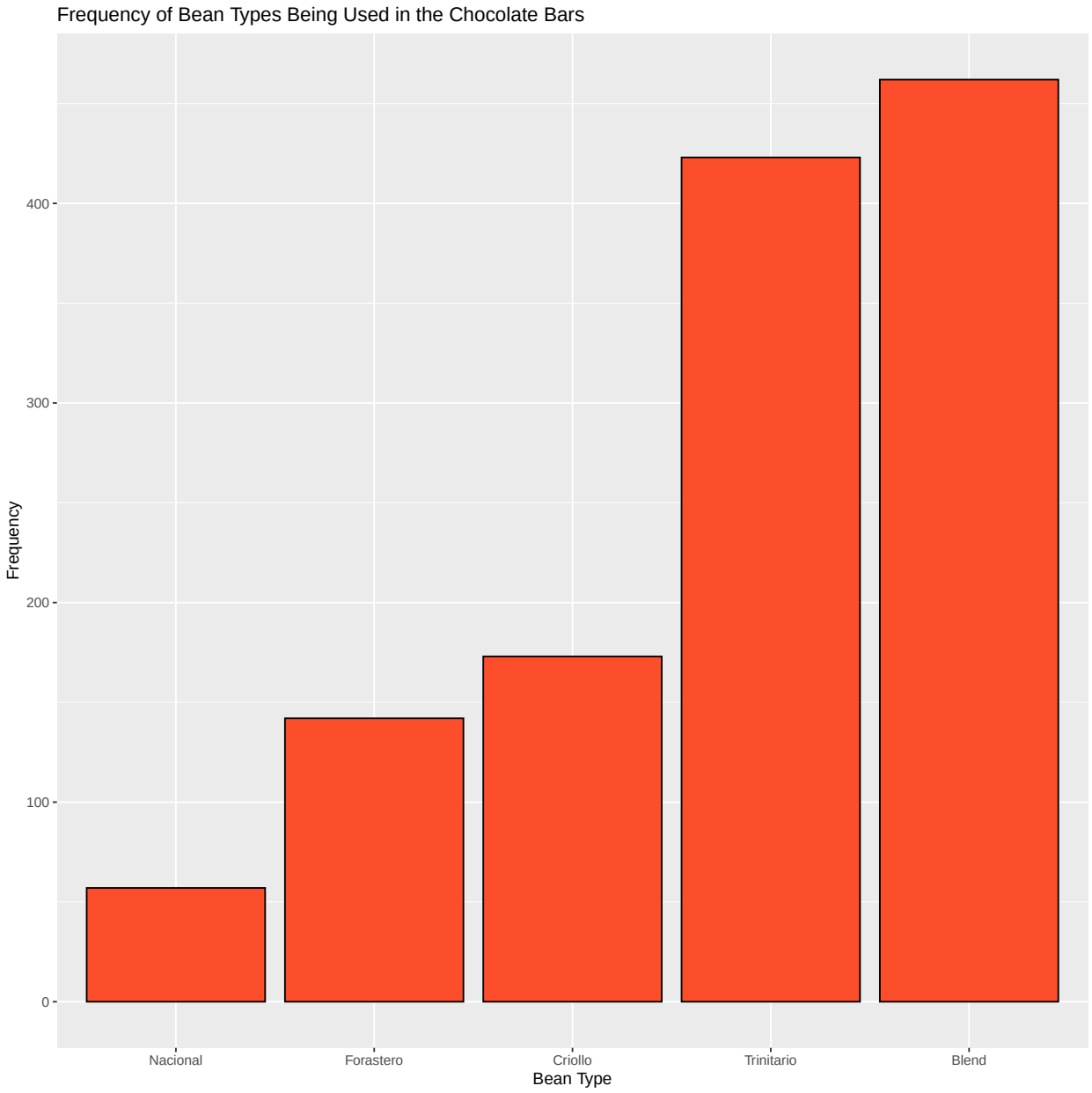
The graph below shows that companies from Vietnam, Canada, Brazil, and Australia tend to produce the highest average rating of chocolate bars with 3.40. Locations that have appeared at least 10 times in the data are only included in the graph.



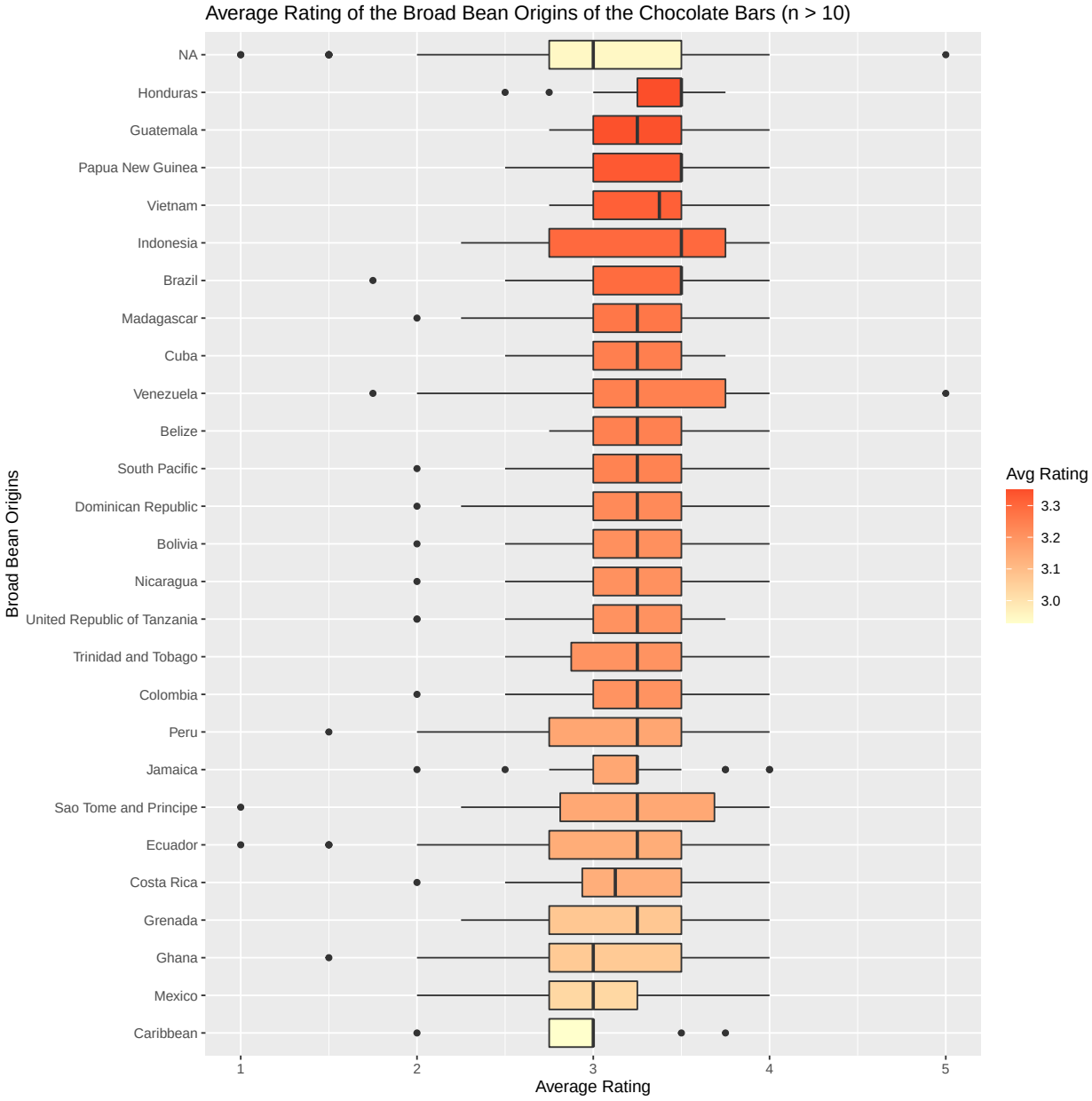
The graph below shows that there isn't a strong relationship between cocoa percentage and chocolate bar rating. However looking at the red line that was created from the geom_smooth function, it appears that as the percentage of cocoa increases, the rating of chocolate decreases.



The graph below shows that Blend and Trinitario are the most common bean types in the data. Combining these two bean types, they account for almost all of the chocolate bars. However, it is worth noting that there are about 600 chocolate bars with no bean type values, so we don't know if this graph is truly the representation of the data.

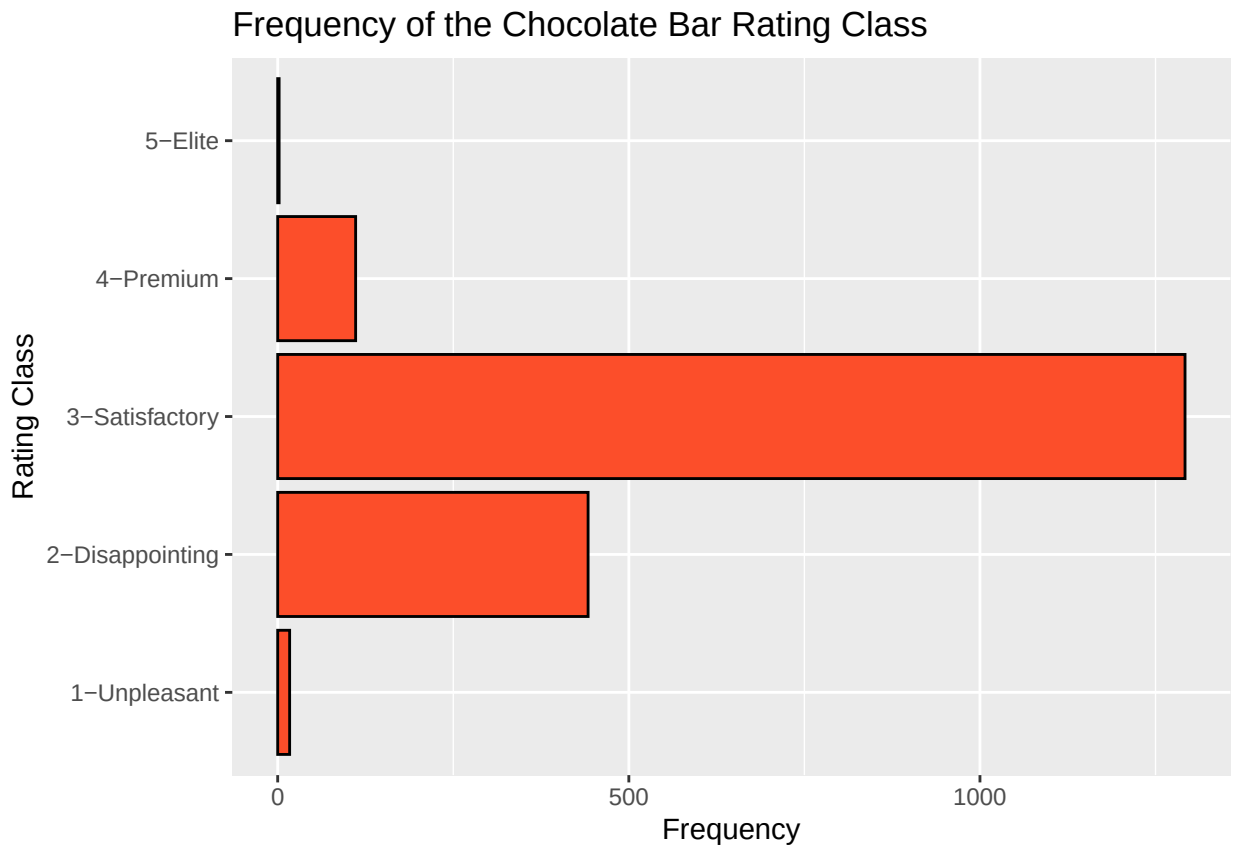


The graph below shows that chocolate bars with cocoa beans that come from the Caribbean or Southeast Asia regions are rated higher on average, although not significantly higher than the rest. Origins that have appeared at least 10 times in the data are only included in the graph.



The graph below shows that there is a trend of rising number of chocolate bar reviews until the year 2016 when it starts to decline. Also, there is less variation in the chocolate bar ratings in the recent years.

The graph below shows that the most common rating class of chocolate bars in the data is Satisfactory. Also, premium and elite chocolate bars are very rare in the data.



2.4 Modeling Approach

We will now start developing different machine learning models to reach our objective, which is to obtain the model with the highest accuracy score after predicting the Chocolate Bar Rating Class.

The features or variables that we will use for our machine learning models to predict the RatingClass of the chocolate bars are:

- CompanyLocation
- CocoaPercent
- BeanType
- BroadBeanOrigin
- ReviewDate

These are the key steps to achieve the goal of the project:

Step 1: Split the dataset into training and testing (validation) sets. The training set will make up about 80 percent of the data, while the testing set will make up about 20 percent of the data.

Step 2: Define and develop the machine learning models.

Step 3: Train the models by feeding them with training data.

Step 4: Test the models by running their predictions in the testing (validation) set.

Step 5: Compare the performance results of the models and determine the model with the highest accuracy score. This model will be considered the best at predicting the RatingClass of the chocolate bars.

These are the machine learning models that we will use:

- Model 1: Support Vector Machine (SVM)
- Model 2: K-Nearest Neighbors (KNN)
- Model 3: Random Forest (RF)

3. Results

Before developing the machine learning models, we will select the features to feed into the models.

```
# Selecting the features to use for the machine learning models and
# Removing rows with NAs.
choco_feat <- choco %>%
  select(CompanyLocation,
         CocoaPercent,
         BeanType,
         BroadBeanOrigin,
         ReviewDate,
         RatingClass) %>%
  drop_na()
```

Let's split the data into training and testing sets. The training set will make up about 80 percent of the data, while the testing set will make up about 20 percent of the data.

```
# Splitting the data into training and testing sets.
# The training set will make up about 80% of the data, while the testing set will make up about 20% of

set.seed(1111)
train_index <- createDataPartition(y = choco_feat$RatingClass,
                                   times = 1,
                                   p = 0.8,
                                   list = FALSE)

# Creating the training set.
train <- choco_feat[train_index, ]

# Creating the testing set.
test <- choco_feat[-train_index, ]
```

We will also modify the resampling method to "repeatedcv", which is a repeated K-fold cross-validation with 10 folds and 3 repetitions.

```
ctr <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
```

Model 1: Support Vector Machine (SVM)

The goal of the Support Vector Machine algorithm is to identify a hyperplane that clearly separates the data points of different classes. The hyperplane that SVM is looking for is the one that has the highest margin between the two classes. The reason for this is that if a new data point appears, then it can be classified with more confidence.

We will apply the Support Vector Machine model to our dataset and test the accuracy of its predictions on the test set.

Let's train the SVM model on the training set and check the accuracy score of its predictions.

```
###-1st Model: Support Vector Machine (SVM)-###
set.seed(1111)

# Training the SVM model on the train set.
svm_fit <- train(RatingClass ~ .,
                 data = train,
                 trControl = ctr,
                 method = "svmRadial")
```

The table shows the accuracy scores of the SVM model in different regularization parameter values (C). We will obtain the highest accuracy score to reference later.

sigma	C	Accuracy	Kappa	AccuracySD	KappaSD
0.0854885	0.25	0.7130343	0.0000000	0.0069929	0.0000000
0.0854885	0.50	0.7116597	-0.0022467	0.0105072	0.0123056
0.0854885	1.00	0.7136697	0.0259222	0.0125810	0.0408303

Let's test the accuracy of the SVM's predictions on the test set.

```
# Predicting the SVM model on the test set
svm_pred <- predict(svm_fit,
                            newdata = test)

# Defining Confusion Matrix
svm_cmatrix <- confusionMatrix(svm_pred, test$RatingClass)
```

As we can see, the accuracy score of the SVM model's predictions on the testing set is about 72.4%.

ML_Model	Train_Acc_Score	Test_Acc_Score
Support Vector Machine (SVM)	0.7136697	0.7242798

Model 2: K-Nearest Neighbors (KNN)

The K-Nearest Neighbor algorithm depends on the concepts of closeness and similarity. Basically, the algorithm tries to identify which class the unknown data point belongs to by looking at its nearest neighboring data points, as the name (K-Nearest Neighbor) suggests. KNN assumes that data with similar traits are near to each other.

We will apply the K-Nearest Neighbors model to our dataset and test the accuracy of its predictions on the test set.

Let's train the KNN model on the training set and check the accuracy score of its predictions.

```
###-2nd Model: K-Nearest Neighbors (KNN)-###
set.seed(1111)

# Training the KNN model on the train set.
knn_fit <- train(RatingClass ~ .,
                 data = train,
                 trControl = ctr,
                 method = "knn")
```

The table shows the accuracy scores of the KNN model in different number of neighbors parameter values (k). We will obtain the highest accuracy score to reference later.

k	Accuracy	Kappa	AccuracySD	KappaSD
5	0.6983099	0.1129094	0.0339674	0.1044734
7	0.7010589	0.0737065	0.0269517	0.0833249
9	0.7003783	0.0397292	0.0198297	0.0638502

Let's test the accuracy of the KNN's predictions on the test set.

```
# Predicting the KNN model on the test set
knn_pred <- predict(knn_fit,
                    newdata = test)

# Defining Confusion Matrix
knn_cmatrix <- confusionMatrix(knn_pred, test$RatingClass)
```

As we can see, the accuracy score of the KNN model's predictions on the testing set is about 67.9%.

ML_Model	Train_Acc_Score	Test_Acc_Score
Support Vector Machine (SVM)	0.7136697	0.7242798
K-Nearest Neighbors (KNN)	0.7010589	0.6790123

Model 3: Random Forest (RF)

The Random Forest algorithm is a classifier that combines multiple decision trees on various subsets of a dataset and obtains the average to improve the dataset's predicted accuracy. The way the algorithm works is based on ensemble learning, which is a technique of uniting several classifiers to solve complex problems and improve the overall result.

We will apply the Random Forest model to our dataset and test the accuracy of its predictions on the test set.

Let's train the RF model on the training set and check the accuracy score of its predictions.

```
###-3rd Model: Random Forest (RF)-###
set.seed(1111)

# Training the RF model on the train set.
rf_fit <- train(RatingClass ~ .,
                data = train,
                trControl = ctr,
                method = "rf")
```

The table shows the accuracy scores of the RF model in different mtry parameter values. The mtry parameter is the number of variables randomly sampled at each tree. We will obtain the highest accuracy score to reference later.

mtry	Accuracy	Kappa	AccuracySD	KappaSD
2	0.7144825	0.0000000	0.0057237	0.0000000
61	0.6830193	0.1589682	0.0292479	0.0790776
121	0.6738860	0.1555690	0.0339051	0.0768803

Let's test the accuracy of the RF's predictions on the test set.

```
# Predicting the RF model on the test set
rf_pred <- predict(rf_fit,
                   newdata = test)

# Defining Confusion Matrix
rf_cmatrix <- confusionMatrix(rf_pred, test$RatingClass)
```

As we can see, the accuracy score of the RF model's predictions on the testing set is about 71.6%.

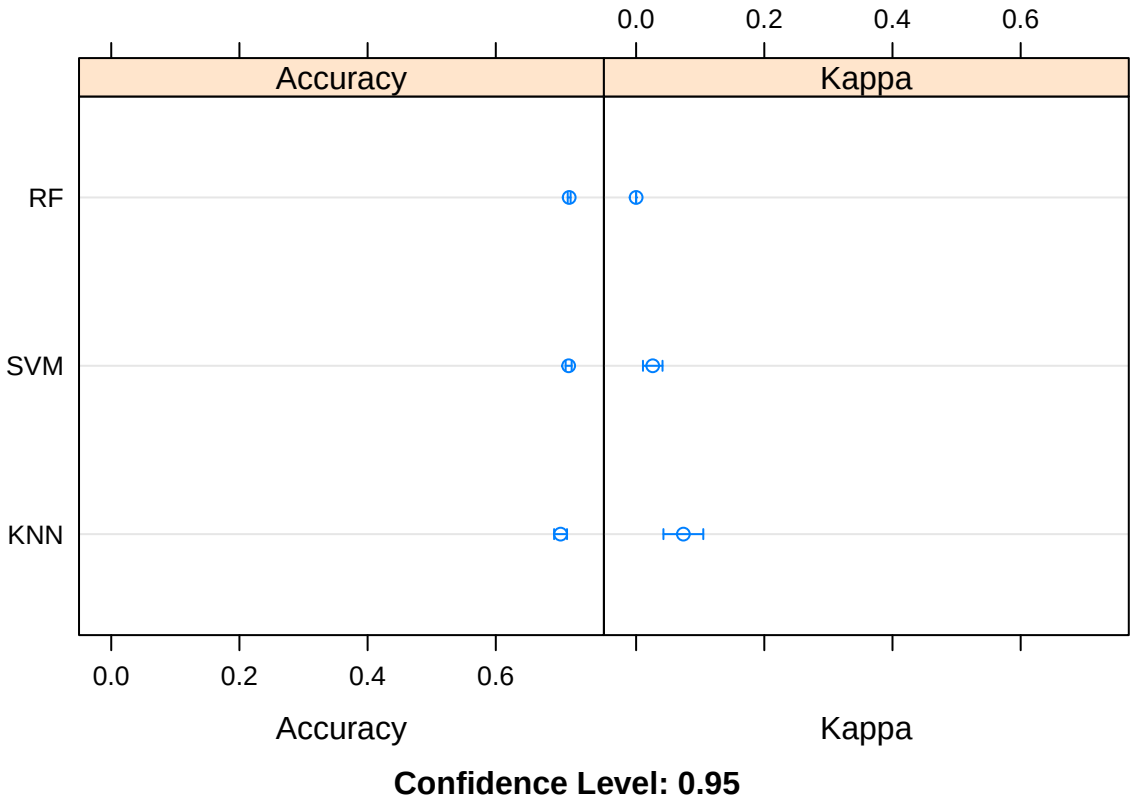
ML_Model	Train_Acc_Score	Test_Acc_Score
Support Vector Machine (SVM)	0.7136697	0.7242798
K-Nearest Neighbors (KNN)	0.7010589	0.6790123
Random Forest (RF)	0.7144825	0.7160494

Compare Results

Just by looking at the accuracy values, it's clear that the model with the highest accuracy score is Support Vector Machine with 72.4%. However, the SVM model's score isn't significantly higher from the model with the second highest accuracy score, which is Random Forest with 71.6%. Therefore, we will need to compare both models in terms of kappa statistics, accuracy, and confidence interval (95%) using a dot plot.

```
# Comparing the results of the models.
clearResults <- resamples(list(SVM = svm_fit,
                               KNN = knn_fit,
                               RF  = rf_fit
))
```

Looking at the plot with the confidence interval set at 95%, it shows that Random Forest is the one that predicts the RatingClass of the chocolate bars most accurately. We will consider Random Forest the best machine learning model to predict the Chocolate Bar Rating Class.



4. Conclusion

In this project, we used the Chocolate Bar Ratings dataset to develop machine learning models that could accurately predict the Chocolate Bar Rating Class. The Random Forest model was the best out of all the models we tested considering it provided the highest percentage of accurate predictions at a 95% confidence interval as shown in the dot plot.

Not only did we build and test machine learning models, but we explored the dataset and created informative visualizations about the features of different chocolate bars such as cocoa percentage, bean type, broad bean origin, company location, and review date. These features were important, as we used them for our machine learning models to predict the Chocolate Bar Rating Class.

4.1 Potential Impact

The potential impact this project brings may benefit chocolate makers. For example, one of the visualizations we created showed that as the cocoa percentage of a chocolate bar increases, the bar's rating decreases. With this information, chocolate makers may limit the cocoa of their chocolates to a certain percentage so they can sell more of their chocolates. Another example that may benefit chocolate makers is when we created the visualization about bean origins. We learned that chocolate bars with cocoa beans that come from the Caribbean or Southeast Asia regions are rated higher on average. This may make chocolate makers consider buying only cocoa beans from the Caribbean or Southeast Asia regions to maximize their profit.

4.2 Limitations

Regarding limitations, I felt like I could have developed more machine learning models into my report, as they could probably produce even better results. However, there is a good chance that running more advanced models would overwhelm the memory of my computer and crash R.

4.3 Future Work

As for future work, we can definitely apply our machine learning approach to different recommendation or rating datasets like the Chocolate Bar Ratings dataset where there are user assessments on items such as products and services.