wine

Sarah Little

11/8/2020

```r
setwd("~/Library/Mobile Documents/com~apple~CloudDocs/Stat 627 Fall")
wine_review <- read.csv("winemag-data-130k-v2.csv")
```

CLEANING THE DATA

#Step 1: Show a snapshot of some of the variables

```
##   X  country                           designation points price          province
## 1 0    Italy                          Vulkà Bianco     87    NA Sicily & Sardinia
## 2 1 Portugal                              Avidagos     87    15             Douro
## 3 2       US                                           87    14            Oregon
## 4 3       US                 Reserve Late Harvest     87    13          Michigan
## 5 4       US Vintner's Reserve Wild Child Block     87    65            Oregon
## 6 5    Spain                          Ars In Vitro     87    15    Northern Spain
##                                                                                  title
## 1                                                    Nicosia 2013 Vulkà Bianco  (Etna)
## 2                                         Quinta dos Avidagos 2011 Avidagos Red (Douro)
## 3                                         Rainstorm 2013 Pinot Gris (Willamette Valley)
## 4                     St. Julian 2013 Reserve Late Harvest Riesling (Lake Michigan Shore)
## 5 Sweet Cheeks 2012 Vintner's Reserve Wild Child Block Pinot Noir (Willamette Valley)
## 6                                 Tandem 2011 Ars In Vitro Tempranillo-Merlot (Navarra)

##                winery            variety        taster_name
## 1             Nicosia        White Blend      Kerin O'Keefe
## 2 Quinta dos Avidagos     Portuguese Red         Roger Voss
## 3           Rainstorm         Pinot Gris       Paul Gregutt
## 4          St. Julian           Riesling Alexander Peartree
## 5        Sweet Cheeks         Pinot Noir       Paul Gregutt
## 6              Tandem Tempranillo-Merlot  Michael Schachner
##   taster_twitter_handle
## 1          @kerinokeefe
## 2            @vossroger
## 3           @paulgwine
## 4
## 5           @paulgwine
## 6          @wineschach

##
## 1                                                                                          Aromas in
## 2                                 This is ripe and fruity, a wine that is smooth while still struc
## 3                                                                              Tart and snappy, the fl
```

```
## 4                                                                 Pineapple rind, lemon pith and orange
## 5             Much like the regular bottling from 2012, this comes across as rather rough and tannic
## 6 Blackberry and raspberry aromas show a typical Navarran whiff of green herbs and, in this case, ho
```

In the wine titles, we can see that the year the wine was made, or the vintage, is usually included. Vintage is an important factor when it comes to evaluating wines. Therefore we will look into extracting the vintage year data from the wine titles and adding the new variable `vintage` to the dataset.

The wine title pattern goes as follows: Winery name first, theen the year the wine is made.We notice some of the winery names include years also (likely the year of their founding), so we ensure that we exclude years contained in winery names:

#Step 2: Add vintage as a new variable

```r
# Extract vintage year information from wine titles
#we get this formula because the winerary name comes first, then the year the wine is made, so we want
wine_review <- wine_review %>% mutate(vintage = as.numeric(substr(title, str_length(winery)+2,str_lengt
wine_review %>%
  select(title, vintage) %>%
  head()
```

```
##                                                                              title
## 1                                                Nicosia 2013 Vulkà Bianco  (Etna)
## 2                                     Quinta dos Avidagos 2011 Avidagos Red (Douro)
## 3                                     Rainstorm 2013 Pinot Gris (Willamette Valley)
## 4                    St. Julian 2013 Reserve Late Harvest Riesling (Lake Michigan Shore)
## 5 Sweet Cheeks 2012 Vintner's Reserve Wild Child Block Pinot Noir (Willamette Valley)
## 6                               Tandem 2011 Ars In Vitro Tempranillo-Merlot (Navarra)
##   vintage
## 1    2013
## 2    2011
## 3    2013
## 4    2013
## 5    2012
## 6    2011
```

#Step 3: Remove any Nas

The next step in cleaning the data is making sure any Nas that will affect the analysis are taken care of, either by removing the variable entirely or ignoring it from our analysis. The following code will look at the number of NA values by variable and summarise them in a chart.

Variable	Num_NA
country	63
description	0
designation	37465
points	0
price	8996
province	63
region_1	21247
region_2	79460
taster_name	26244
taster_twitter_handle	31213
title	0

Variable	Num_NA
variety	1
winery	0
vintage	4630

After looking at the table above, we can see that over 79,000 observations have missing data for `region_2` and over 21,000 have missing data for `region_1`. The variables `taster_twitter_handle`, 'taster_

After looking at the table above, we can see that over 79,000 observations have missing data for `region_2` and over 21,000 have missing data for `region_1`. The variables `taster_twitter_handle`, `taster_name` and `designation` also have a significant amount of observations missing. Given the other variables available to us as well as the amount of observations missing., we feel that is is appropriate fir us to ignore the variables listed above.

Lastly, the purpose of the analysis will be to predict the wine quality for wine we may not have seen before, so we will be also ignoring the `winery` and `title` variables.

Therefore, we only care about ensuring our dataset has non blank entries for `country`, `price`, `province`, `variety` and our new variable `vintage`.

```r
# Filter out NA or blank entries for the relevant variables in our data set
wine_clean <- wine_review %>% filter(!is.na(price) & price != "" &
                                !is.na(country) & country != "" &
                                !is.na(province) & province != "" &
                                !is.na(variety) & variety != "" &
                                !is.na(vintage) & vintage != "")
wine_clean %>% nrow() # Count the remaining entries in the data set
```

```
## [1] 116761
```

Our dataset has now dropped from 129,971 observations to 116,761 observations after removing necessary blank spaces.

```
##                 Variable Num_NA
## 1                country      0
## 2            description      0
## 3            designation  34474
## 4                 points      0
## 5                  price      0
## 6               province      0
## 7               region_1  19062
## 8               region_2  67221
## 9            taster_name  23562
## 10 taster_twitter_handle  28378
## 11                 title      0
## 12               variety      0
## 13                winery      0
## 14               vintage      0
```

Next, we check to see the number of distinct entries for the relevant variables to be used:

```
##                          Count
```

```
## Distinct Countries        42
## Distinct Descriptions 107671
## Distinct Points           21
## Distinct Prices          389
## Distinct Provinces       415
## Distinct Varieties       682
## Distinct Vintages         56
```

We want to check to make sure there aren't any over laps in location names because we are looking to use this variable in the analysis. To check this we can temporarily unite the country and province variables to create the unique location. Then by looking at the distint entries, if the number of the country_province variable exceeds the number of distinct provinces, there is some overlap we need to address.

```
#create new variable and count entries
wine_clean %>% unite(country_province, country, province, sep = "_") %>%
  summarise(n_distinct(country_province))
```

```
##   n_distinct(country_province)
## 1                          415
```

We see that the result matches the number of distinct province names, 415, and therefore each province name is unique to one country only. This means that during the analysis, we should chose whether to use the more broad variable of country, or to use province. Since the province where wine comes from usually is a huge selling point over country, we will chose to ignore the country variable.

#Step 4: Adjusting the points variable

First, the computing power we are working with will not be able to accomidate a dataset as large as we have right now. We decide to take a random sample of 10% of the dataset to help run our machine learning algorithms in a more managble amount of time.

Next, we are also going to add one more variable called `points_bracket`. It is important to make sort of "buckets" fpor the points variables to live in because the difference between a score such as an 89 might not mean that much to the models, but it is the difference of 'Very Good" and 'Excellent' according to the wine reviewers. The table with the broken down brackets is shown below.

Score	Points_bracket
98-100	Classic
94-97	Superb
90-93	Excellent
87-89	Very Good
83-86	Good
80-82	Acceptable

We then construct a training set containing 75% of the entries and a testing set containing approximately 25%. It is important to rembember to discard entries with provinces and grape varieties that do not appear in the training set, since these both have very large factor levels and the models wont run if all of the factor levels do not match up. We choose an 75:25 split in order to have a substantial proportion of the data entries available in our training set to train the models:

```
library(cleandata)
# Set sample seed to 1 for replicability
set.seed(1, sample.kind="Rounding")
```

```r
# We filter the data set to only include the columns that will be relevant for our study,
# then sample 10% of the cleaned data set entries and add a category variable with the score brackets
wine_clean <- wine_clean %>% select(-X, -designation, -country, -region_1, -region_2,
                                   -taster_name, -taster_twitter_handle, -title, -winery) %>%
  sample_n(11676) %>%
  mutate(point_bracket = as.factor(case_when(points > 97 ~ "Classic",
                              points > 93 ~ "Superb",
                              points > 89 ~ "Excellent",
                              points > 86 ~ "Very Good",
                              points > 82 ~ "Good",
                              points >= 80 ~ "Acceptable")))
# Set sample seed to 1 for replicability
set.seed(1, sample.kind="Rounding")

wine_clean %>% select(-description) %>%
  head() %>% knitr::kable()
```

points	price	province	variety	vintage	point_bracket
90	45	Washington	Cabernet Sauvignon	2014	Excellent
92	64	Alsace	Gewürztraminer	2013	Excellent
90	35	Alentejano	Portuguese White	2012	Excellent
83	10	Central Spain	Tempranillo	2015	Good
86	30	Bordeaux	Bordeaux-style Red Blend	2013	Good
91	55	California	Chardonnay	2011	Excellent

```r
#Create the training and testing sets
# We create a test index using 25% of the entries in the dataset
library(caTools)
set.seed(1, sample.kind="Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```r
wine_sample <- sample.split(wine_clean, SplitRatio = 0.75)
#create train
wine_train <- subset(wine_clean, wine_sample == TRUE) # creates a training dataset named train1 with ro
#create test
wine_test <- subset(wine_clean, wine_sample == FALSE)

# Make sure that provinces and varieties in the test set are also in the training set
wine_test <- wine_test %>%
  semi_join(wine_train, by = "province") %>%
  semi_join(wine_train, by = "variety")
# Remove unused factors for the variety and province variables in the training and test sets
wine_train <- wine_train %>% mutate(variety = droplevels(variety),
                                    province = droplevels(province))
wine_test <- wine_test %>% mutate(variety = droplevels(variety),
                                  province = droplevels(province))
# Match the factor levels in the training and test sets to prepare for use in our models
levels(wine_test$variety) <- levels(wine_train$variety)
levels(wine_test$province) <- levels(wine_train$province)
```

Data_set	No_entries
wine_train	8340
wine_test	3263

We see that the training set and test set contain data entries in approximately an 75:25 proportion.

```
#delete any unused datasets now to save vector memory space
rm(wine_review)
```

#Step 5: Visualizing the data

Now we begin visualizing the data in our training set to further understand the links between the different variables and wine ratings.

First, we will start with some sentiment analysis of the descriptions. If this variable seems to have a strong link to the points scored by the wine, then it should be defintly included in our models. We utilize the `tinytex` oackage in R and the `bing` lexicon, which classifies words as either positive or negative only:

```
bing <- get_sentiments("bing")                            # Load the bing sentiments
set.seed(1, sample.kind = "Rounding")                     # Set seed for replicability
sample_set_bing <- wine_train %>%
  mutate(description = as.character(description)) %>% # Convert descriptions from factors
  sample_n(20) %>%                                        # Sample 20 rows
  unnest_tokens(word, description) %>%                    # Tokenize words
  filter(!word %in% stop_words$word &                     # Filter out stop words
           !str_detect(word, "^\\d+$")) %>%               # Filter out numbers
  inner_join(bing, by="word")                             # Join the bing sentiments by word
sample_set_bing
```

```
##    points price          province             variety vintage
## 1      86    13        California          Chardonnay    2010
## 2      86    13        California          Chardonnay    2010
## 3      86    13        California          Chardonnay    2010
## 4      86    13        California          Chardonnay    2010
## 5      87    18        California            Viognier    2013
## 6      87    18        California            Viognier    2013
## 7      87    18        California            Viognier    2013
## 8      87    18        California            Viognier    2013
## 9      92    23            Oregon          Pinot Gris    2015
## 10     92    23            Oregon          Pinot Gris    2015
## 11     92    23            Oregon          Pinot Gris    2015
## 12     87    23 Sicily & Sardinia             Zibibbo    2013
## 13     87    23 Sicily & Sardinia             Zibibbo    2013
## 14     93    25        Washington            Riesling    2008
## 15     88    40             Douro              Sousão    2013
## 16     88    28        California               Syrah    2012
## 17     88    28        California               Syrah    2012
## 18     88    28        California               Syrah    2012
## 19     88    28        California               Syrah    2012
## 20     88    28        California               Syrah    2012
## 21     85    12        California          Chardonnay    2006
## 22     85    12        California          Chardonnay    2006
## 23     89    15      Loire Valley      Cabernet Franc    2011
```

```
## 24     89    15      Loire Valley          Cabernet Franc    2011
## 25     89    15      Loire Valley          Cabernet Franc    2011
## 26     89    15      Loire Valley          Cabernet Franc    2011
## 27     86    19    Northern Spain                 Godello    2009
## 28     86    19    Northern Spain                 Godello    2009
## 29     86    19    Northern Spain                 Godello    2009
## 30     86    19    Northern Spain                 Godello    2009
## 31     89    10             Pfalz                Riesling    2012
## 32     89    10             Pfalz                Riesling    2012
## 33     89    10             Pfalz                Riesling    2012
## 34     89    10             Pfalz                Riesling    2012
## 35     89    10             Pfalz                Riesling    2012
## 36     89    10             Pfalz                Riesling    2012
## 37     89    10             Pfalz                Riesling    2012
## 38     89    10             Pfalz                Riesling    2012
## 39     87    45        California             Tempranillo    2013
## 40     87    45        California             Tempranillo    2013
## 41     91    34            Veneto                   Glera    2014
## 42     91    34            Veneto                   Glera    2014
## 43     91    34            Veneto                   Glera    2014
## 44     91    34            Veneto                   Glera    2014
## 45     91    34            Veneto                   Glera    2014
## 46     91    34            Veneto                   Glera    2014
## 47     91    34            Veneto                   Glera    2014
## 48     91    34            Veneto                   Glera    2014
## 49     91    34            Veneto                   Glera    2014
## 50     91    55        California                   Syrah    2006
## 51     91    55        California                   Syrah    2006
## 52     91    55        California                   Syrah    2006
## 53     91    55        California                   Syrah    2006
## 54     91    55        California                   Syrah    2006
## 55     94    85        Washington    Rhône-style Red Blend    2014
## 56     94    85        Washington    Rhône-style Red Blend    2014
## 57     94    85        Washington    Rhône-style Red Blend    2014
## 58     94    85        Washington    Rhône-style Red Blend    2014
## 59     94    85        Washington    Rhône-style Red Blend    2014
## 60     94    85        Washington    Rhône-style Red Blend    2014
## 61     94    85        Washington    Rhône-style Red Blend    2014
## 62     94    85        Washington    Rhône-style Red Blend    2014
## 63     88    26        California          Cabernet Franc    2013
## 64     88    26        California          Cabernet Franc    2013
## 65     88    26        California          Cabernet Franc    2013
## 66     88    26        California          Cabernet Franc    2013
## 67     85   110        California               Red Blend    2011
## 68     85   110        California               Red Blend    2011
## 69     85   110        California               Red Blend    2011
## 70     84    20  Mendoza Province               Red Blend    2014
## 71     92    70           Tuscany              Sangiovese    2012
## 72     92    70           Tuscany              Sangiovese    2012
## 73     92    70           Tuscany              Sangiovese    2012
## 74     92    70           Tuscany              Sangiovese    2012
## 75     92    70           Tuscany              Sangiovese    2012
## 76     88    35          Bordeaux Bordeaux-style Red Blend    2014
## 77     88    35          Bordeaux Bordeaux-style Red Blend    2014
```

```
## 78     88    35          Bordeaux Bordeaux-style Red Blend    2014
## 79     88    35          Bordeaux Bordeaux-style Red Blend    2014
## 80     88    35          Bordeaux Bordeaux-style Red Blend    2014
## 81     88    35          Bordeaux Bordeaux-style Red Blend    2014
##    point_bracket            word sentiment
## 1           Good           solid  positive
## 2           Good            rich  positive
## 3           Good           lemon  negative
## 4           Good            nice  positive
## 5      Very Good           crisp  positive
## 6      Very Good        richness  positive
## 7      Very Good         elegant  positive
## 8      Very Good          modern  positive
## 9      Excellent       perfectly  positive
## 10     Excellent       excellent  positive
## 11     Excellent           fresh  positive
## 12     Very Good         intense  negative
## 13     Very Good           peach  positive
## 14     Excellent           dense  negative
## 15     Very Good            dark  negative
## 16     Very Good           sweet  positive
## 17     Very Good straightforward  positive
## 18     Very Good          bright  positive
## 19     Very Good            dark  negative
## 20     Very Good           dense  negative
## 21          Good           brisk  positive
## 22          Good       versatile  positive
## 23     Very Good           solid  positive
## 24     Very Good            fine  positive
## 25     Very Good           smoke  negative
## 26     Very Good           fresh  positive
## 27          Good           peach  positive
## 28          Good           clean  positive
## 29          Good          smooth  positive
## 30          Good         refined  positive
## 31     Very Good            cute  positive
## 32     Very Good      remarkably  positive
## 33     Very Good        balanced  positive
## 34     Very Good           peach  positive
## 35     Very Good           zippy  positive
## 36     Very Good           lemon  negative
## 37     Very Good          thirst  negative
## 38     Very Good           solid  positive
## 39     Very Good           fried  negative
## 40     Very Good      powerfully  positive
## 41     Excellent        enticing  positive
## 42     Excellent            wild  negative
## 43     Excellent           peach  positive
## 44     Excellent            lead  positive
## 45     Excellent         crushed  negative
## 46     Excellent         elegant  positive
## 47     Excellent         elegant  positive
## 48     Excellent         vibrant  positive
## 49     Excellent        richness  positive
```

```
## 50     Excellent          pretty  positive
## 51     Excellent      incredibly  positive
## 52     Excellent            fine  positive
## 53     Excellent           sweet  positive
## 54     Excellent         perfect  positive
## 55        Superb           leads  positive
## 56        Superb        fragrant  positive
## 57        Superb         crushed  negative
## 58        Superb           smoke  negative
## 59        Superb        seamless  positive
## 60        Superb            rich  positive
## 61        Superb       exquisite  positive
## 62        Superb     exceptional  positive
## 63     Very Good           faint  negative
## 64     Very Good           peach  positive
## 65     Very Good         playful  positive
## 66     Very Good      delightful  positive
## 67          Good          strong  positive
## 68          Good            cool  positive
## 69          Good            cool  positive
## 70          Good          smells  negative
## 71     Excellent          mature  positive
## 72     Excellent            dark  negative
## 73     Excellent            wild  negative
## 74     Excellent         compact  positive
## 75     Excellent         support  positive
## 76     Very Good           dense  negative
## 77     Very Good            dark  negative
## 78     Very Good          bitter  negative
## 79     Very Good           crisp  positive
## 80     Very Good           shame  negative
## 81     Very Good            dark  negative
```

By just breifly glancing at the sample of 20 entries, it does not seem that the description variable will be offering us any useful information when trying to predict the wine score. A lot of the words the bing lexicon sees as negative, should not be considered negative in the wine industry. However, we will look further before ruling anything out.

If we group sentiments by review and calculate the percentage of positive words for each entry, we see that all 20 entries now have some sentiment value attached to them:

```
# Group the word sentiment scores by review and calculate the postive word percentage
sample_set_bing %>% unite(review, province, vintage, variety, sep = " ") %>%
  group_by(review) %>%
  summarise(points = points[1],
            Positive_percentage = sum(sentiment == "positive")*100/n())
```

```
## # A tibble: 20 x 3
##    review                                 points Positive_percentage
##    <chr>                                   <int>               <dbl>
##  1 Bordeaux 2014 Bordeaux-style Red Blend     88                16.7
##  2 California 2006 Chardonnay                 85               100
##  3 California 2006 Syrah                      91               100
##  4 California 2010 Chardonnay                 86                75
```

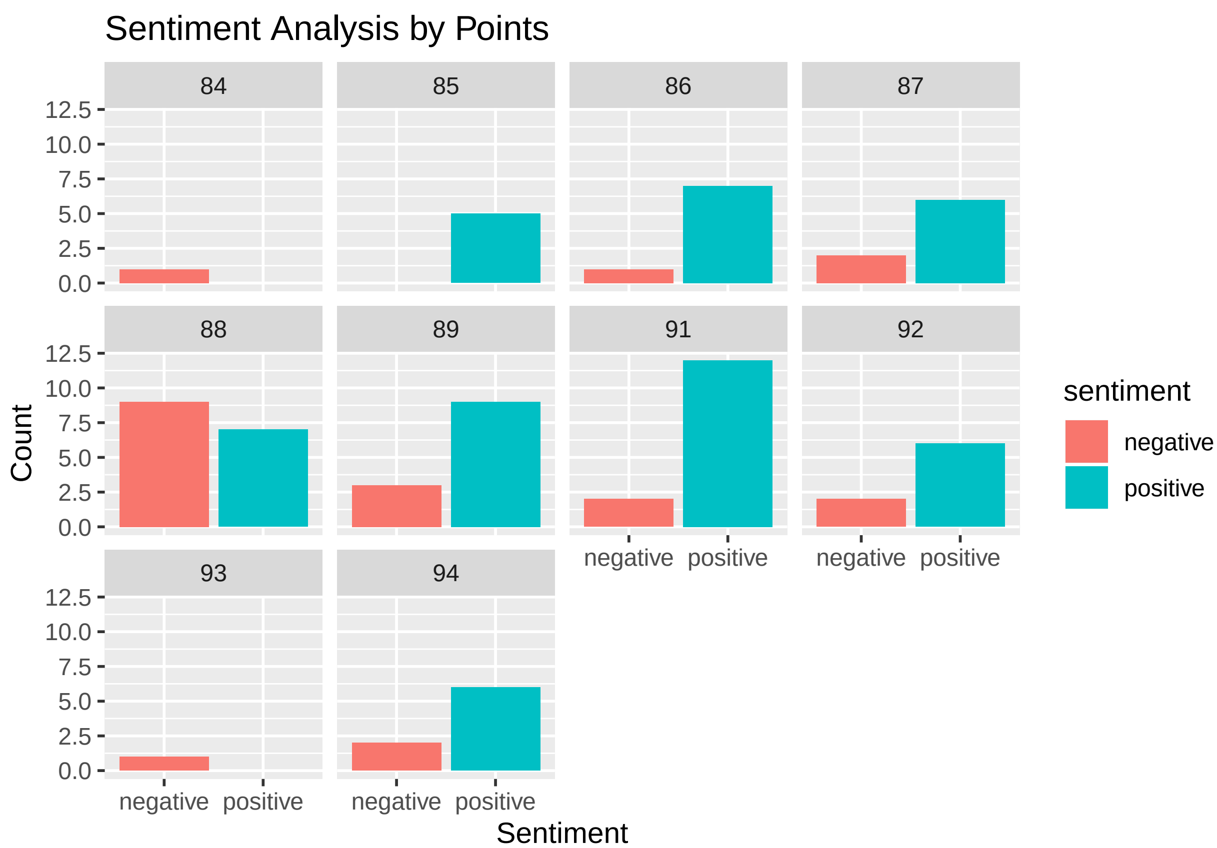
```
##  5 California 2011 Red Blend                   85              100
##  6 California 2012 Syrah                       88               60
##  7 California 2013 Cabernet Franc              88               75
##  8 California 2013 Tempranillo                 87               50
##  9 California 2013 Viognier                    87              100
## 10 Douro 2013 Sousão                           88                0
## 11 Loire Valley 2011 Cabernet Franc            89               75
## 12 Mendoza Province 2014 Red Blend             84                0
## 13 Northern Spain 2009 Godello                 86              100
## 14 Oregon 2015 Pinot Gris                      92              100
## 15 Pfalz 2012 Riesling                         89               75
## 16 Sicily & Sardinia 2013 Zibibbo              87               50
## 17 Tuscany 2012 Sangiovese                     92               60
## 18 Veneto 2014 Glera                           91               77.8
## 19 Washington 2008 Riesling                    93                0
## 20 Washington 2014 Rhône-style Red Blend       94               75
```

This table gives us a brief snapshot of another reason why it does not look like the description will give us any analytic value. Some of the wines that scored in the lower bracket are shown to have a 100% positive sentiment analysis, when we would expect them to have at least some negative values.

We look below at a a plot of sentiment, faceted over the points offered for a visual.

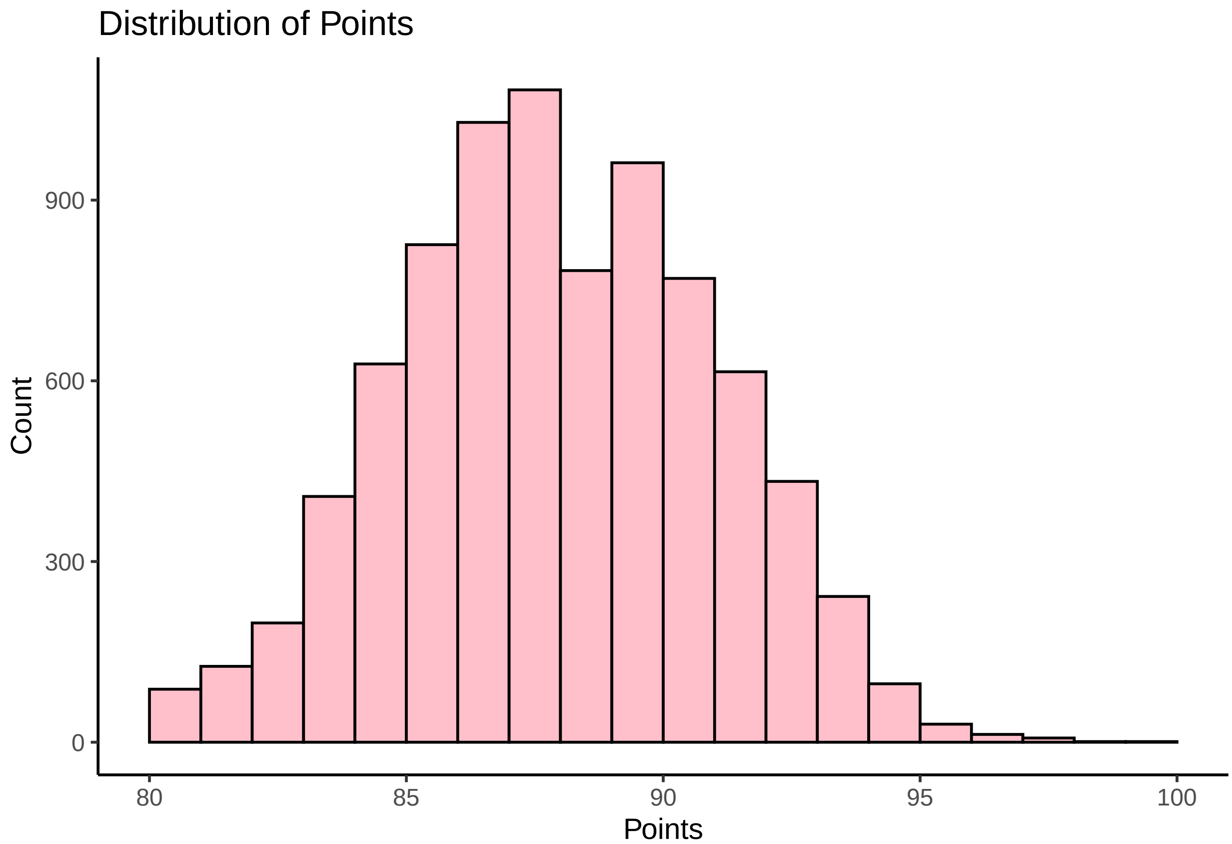
```
#graph the sentiment
sample_set_bing %>%
  ggplot(aes(fill = sentiment)) +geom_bar(mapping = aes(x = factor(sentiment),position = "fill")) +face
```

```
## Warning: Ignoring unknown aesthetics: position
```



We can see from the plot that sentiment of the description has no real pattern or effect on the points given. It seems that the description is more of a description than an oppinionist review of the wine.

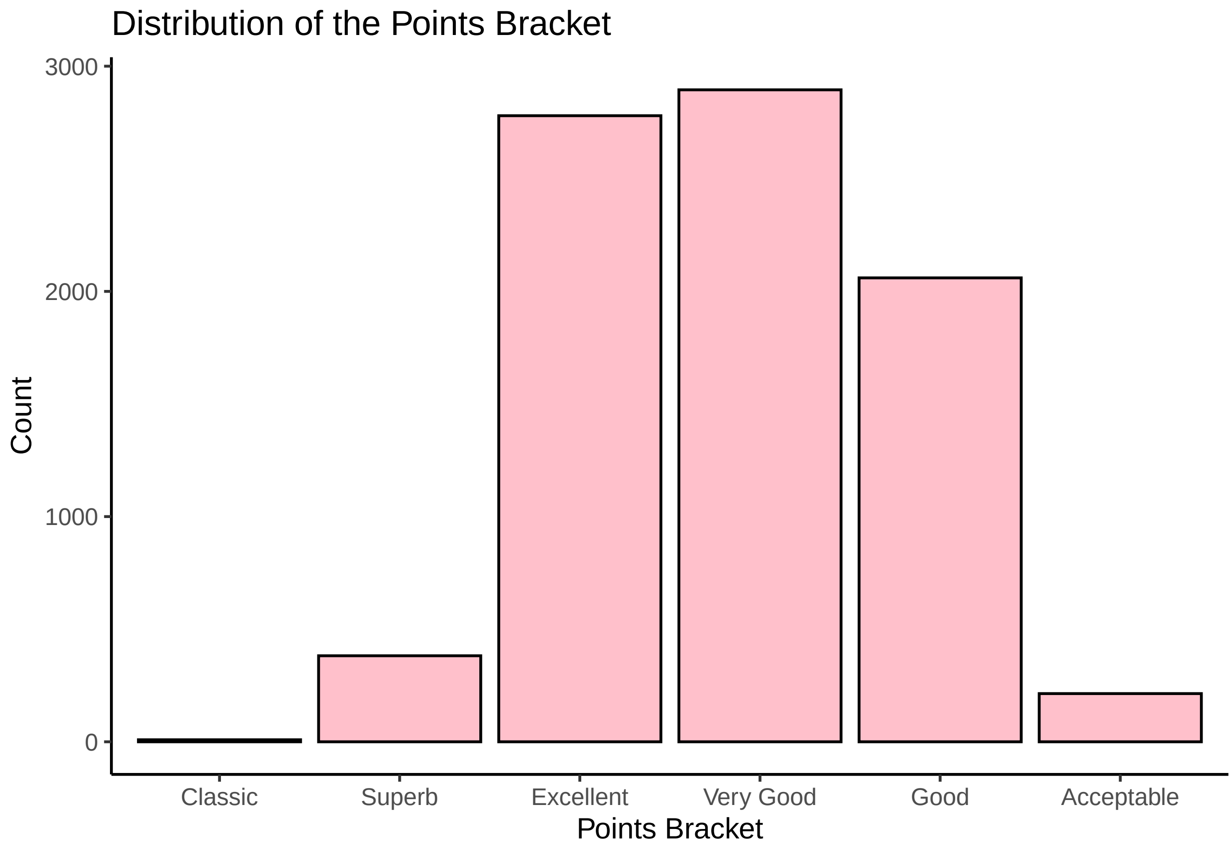
Next, we look into the distribution of the wine review scores and plot them on a histogram.




```
##   Mean points SD points Min points Max points
## 1        88.4      3.08         80        100
```

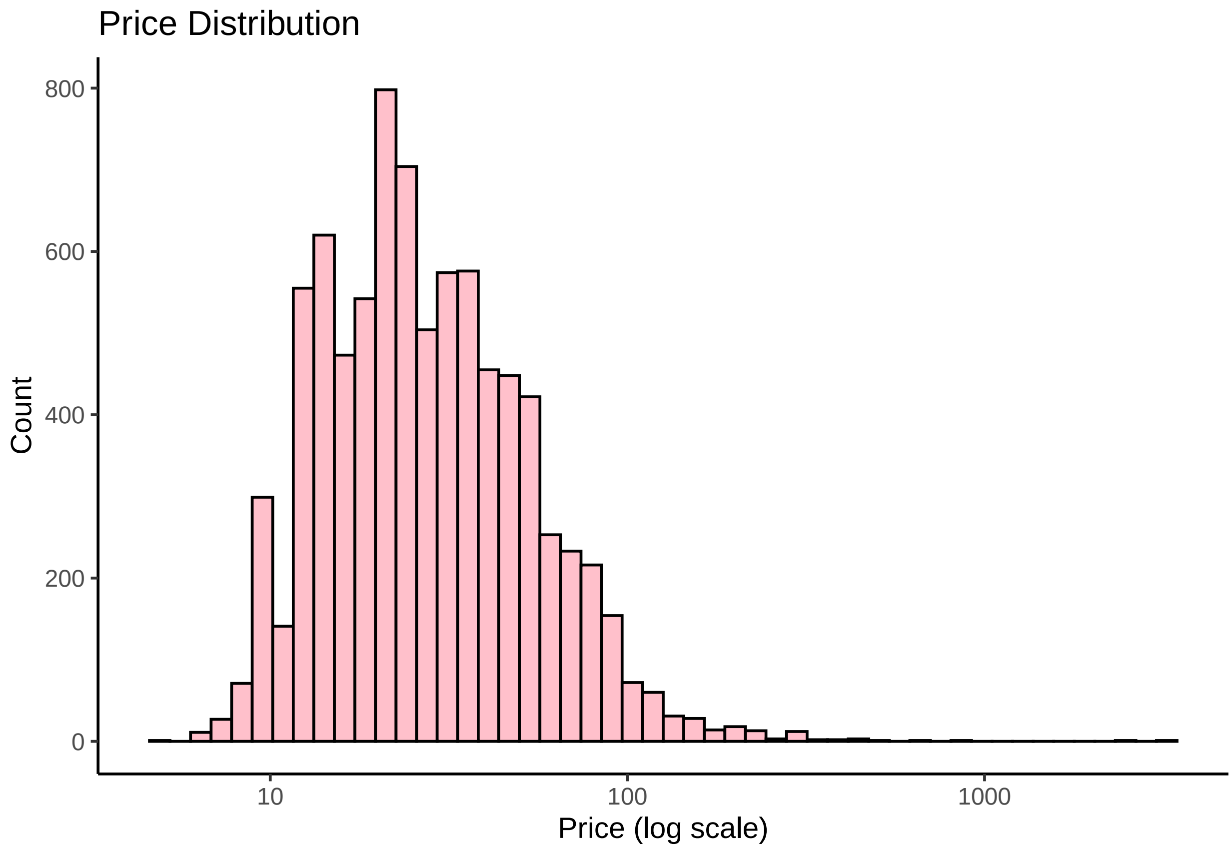
We can see that the distribution of points somewhat resembles a normal distribution, albeit with two peaks, with a mean of 88.4 and standard deviation of 3.08.

Next, we look into points by the score bracket.



Plotting a histogram by points score category shows a similar distribution.

Now looking at wine prices, we plot the distribution of wine prices in our data set. We have to adjust by log scale because there are a few very highly priced bottles of wine that are very far away from the average prices.

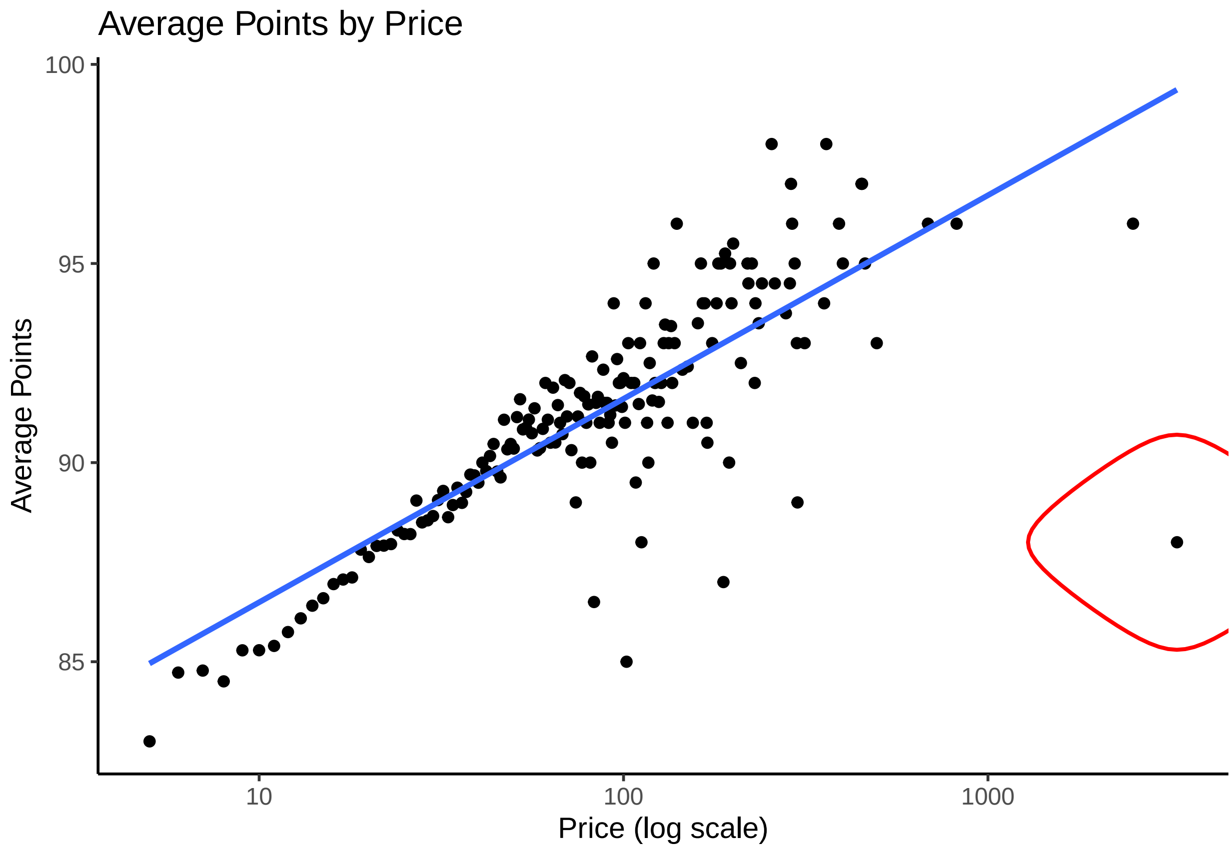



```
##   Mean price SD price Min price Max price
## 1       35.3     55.4         5      3300
```

We see that there is a very large range in wine prices, from $5 to $3,300 per bottle in our training set.But the mean price is around $35, so it seems that there are only a couple bottles of wine on the higher end.

Looking further into price, we will examine the relationship between price and points scored.

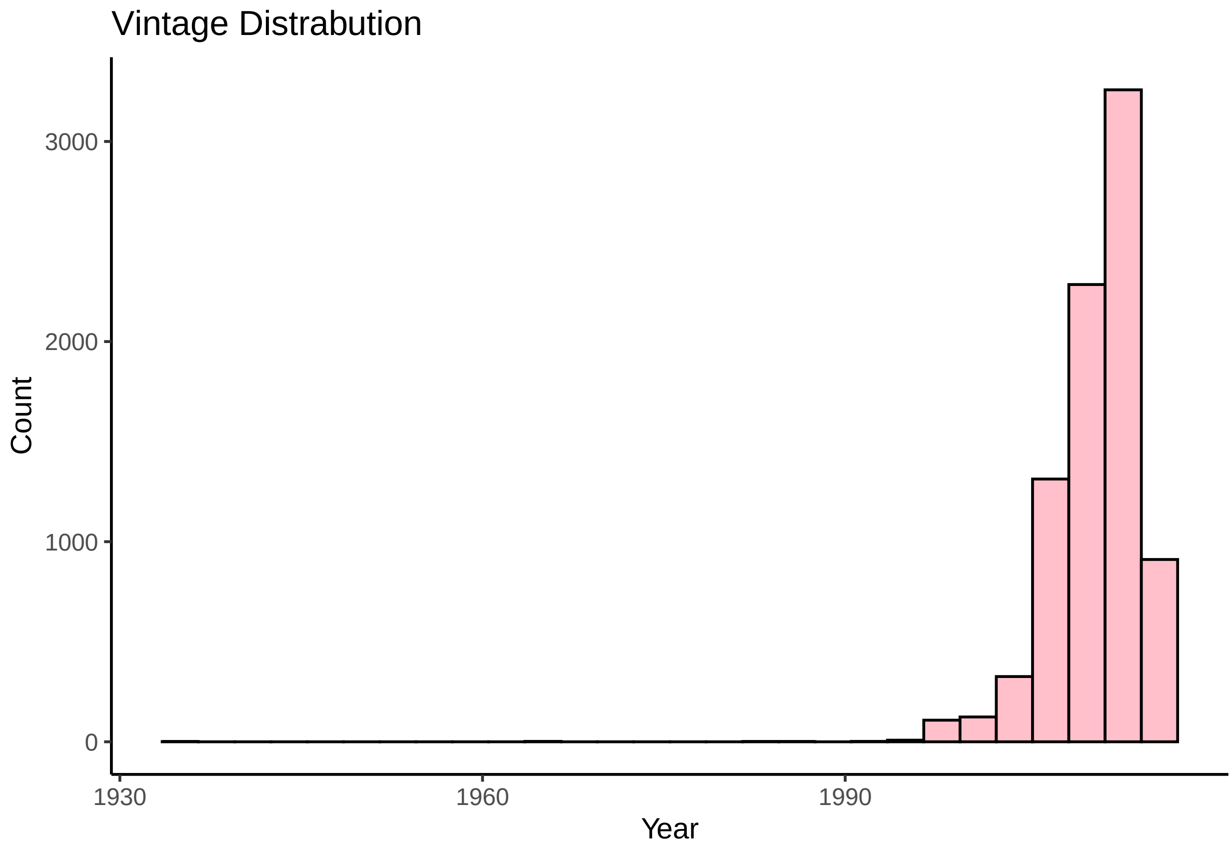
```
## Registered S3 methods overwritten by 'ggalt':
##   method                  from
##   grid.draw.absoluteGrob  ggplot2
##   grobHeight.absoluteGrob ggplot2
##   grobWidth.absoluteGrob  ggplot2
##   grobX.absoluteGrob      ggplot2
##   grobY.absoluteGrob      ggplot2
```



We see from the plot above that there is a strong positive trend between wine price and average points. However, from the graph and the summary below, we see for the wine priced at the maximum $3,300 price is actually below the average of all ratings (88.4):

```
## # A tibble: 1 x 3
##   price `Average points` count
##   <dbl>            <dbl> <int>
## 1  3300               88     1
```

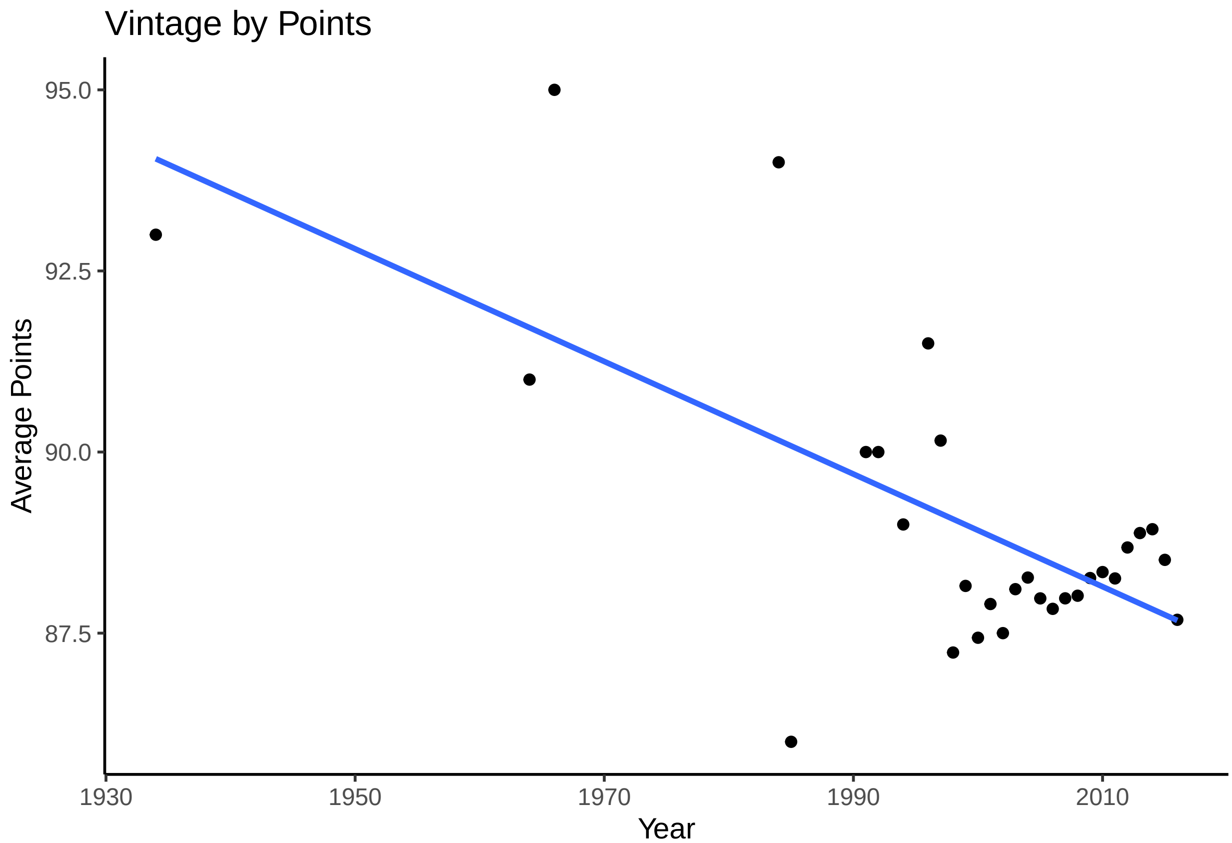
Next, we look at the distribution of vintage years:




```
##   Mean year SD year Min year Max year
## 1      2011    3.71     1934     2016
```

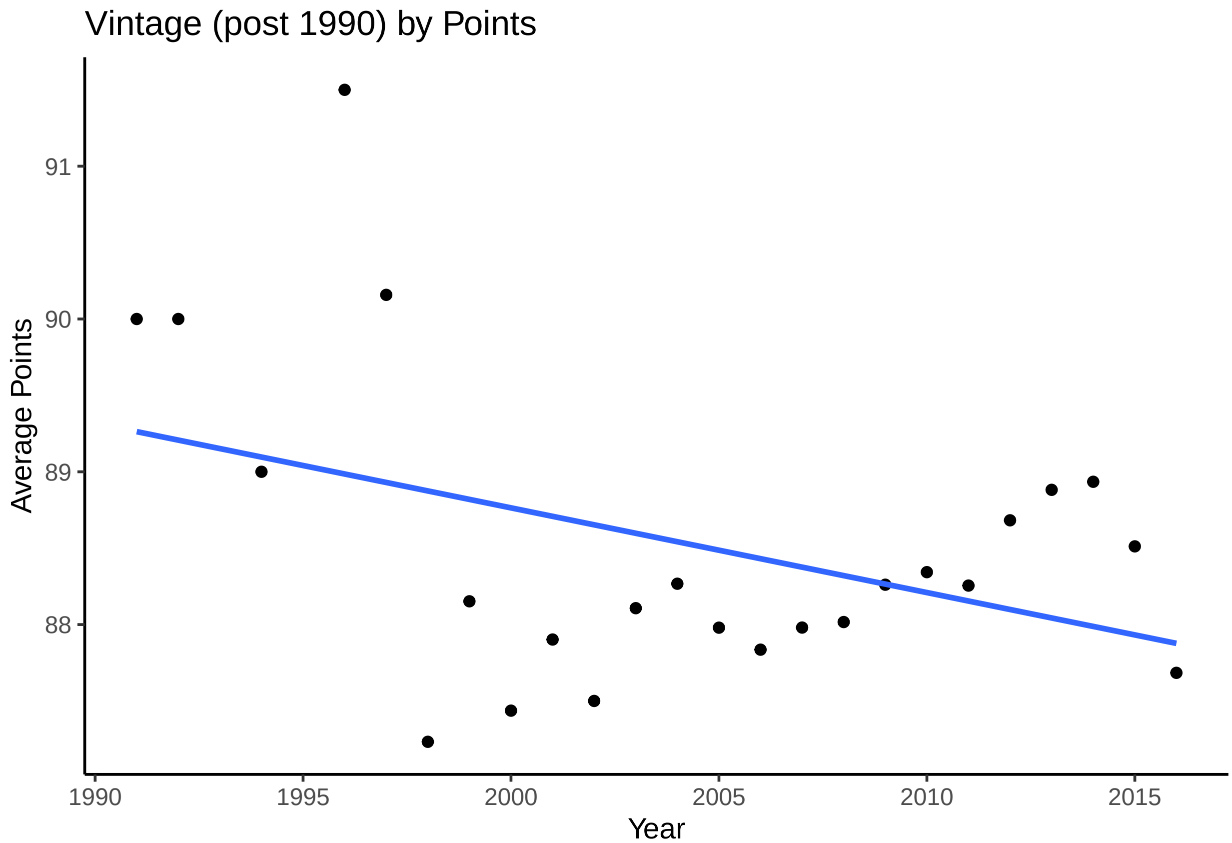
We see that the vintages for wines in our training set are heavily clustered in more recent years, with an average vintage of 2010, but with an earliest vintage of 1934.

We look at a plot of the relationship of vintage against points below:



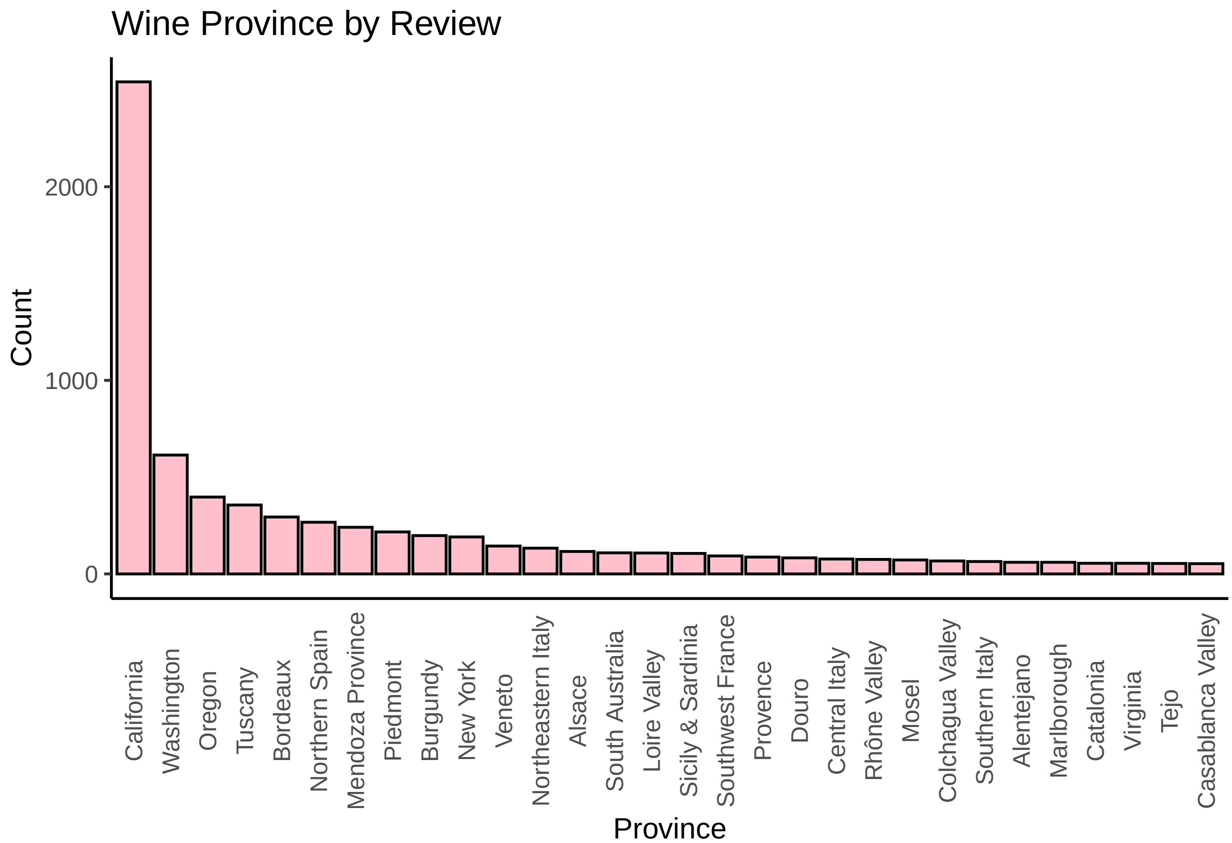
There is a very strong negative trend for wine points versus vintage, that is the newer wines score way lower than the older wines did. However, we can also see that the newer wines have way more data points available than the older vintages, so this mught be one reason for the affect we are seeing.

We will replot the wines for vintages after 1990:

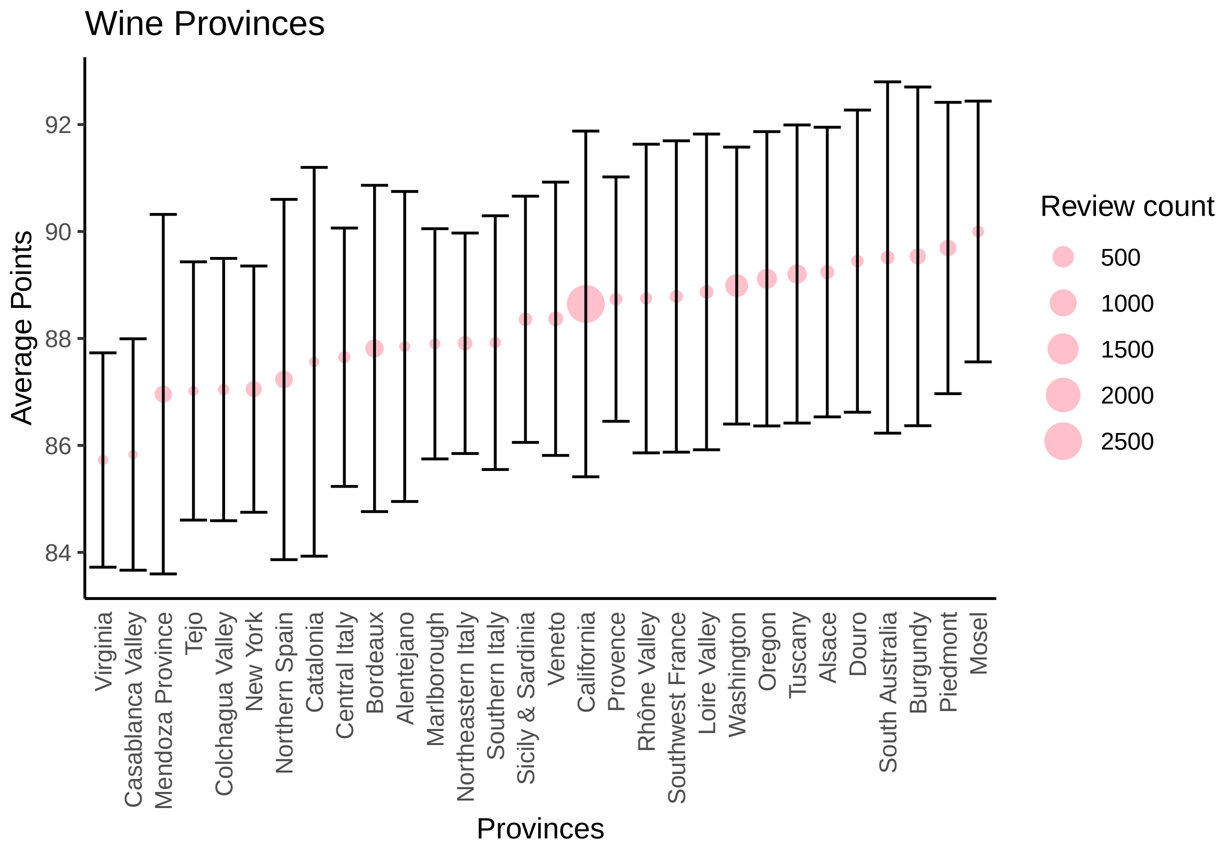


The steep negative trend we saw before is not as noticable without the older vintages.

Now, we examine the link between the province of wine origin and points. First, since we have over 230 provinces, we need to subset that group to make the visualization a little better. We will start with subsetting by number of reviews given by region and only include regions that recieved over 50 reviews to be plotted



Frpm the graph, we can see that Calirfonia has an overwhelming amount of reviews compared to the other provinces. In terms of countries however, France and Italy are featured far more often than the other countries, even if the regions did not recueve nearly as many reviews as just California.



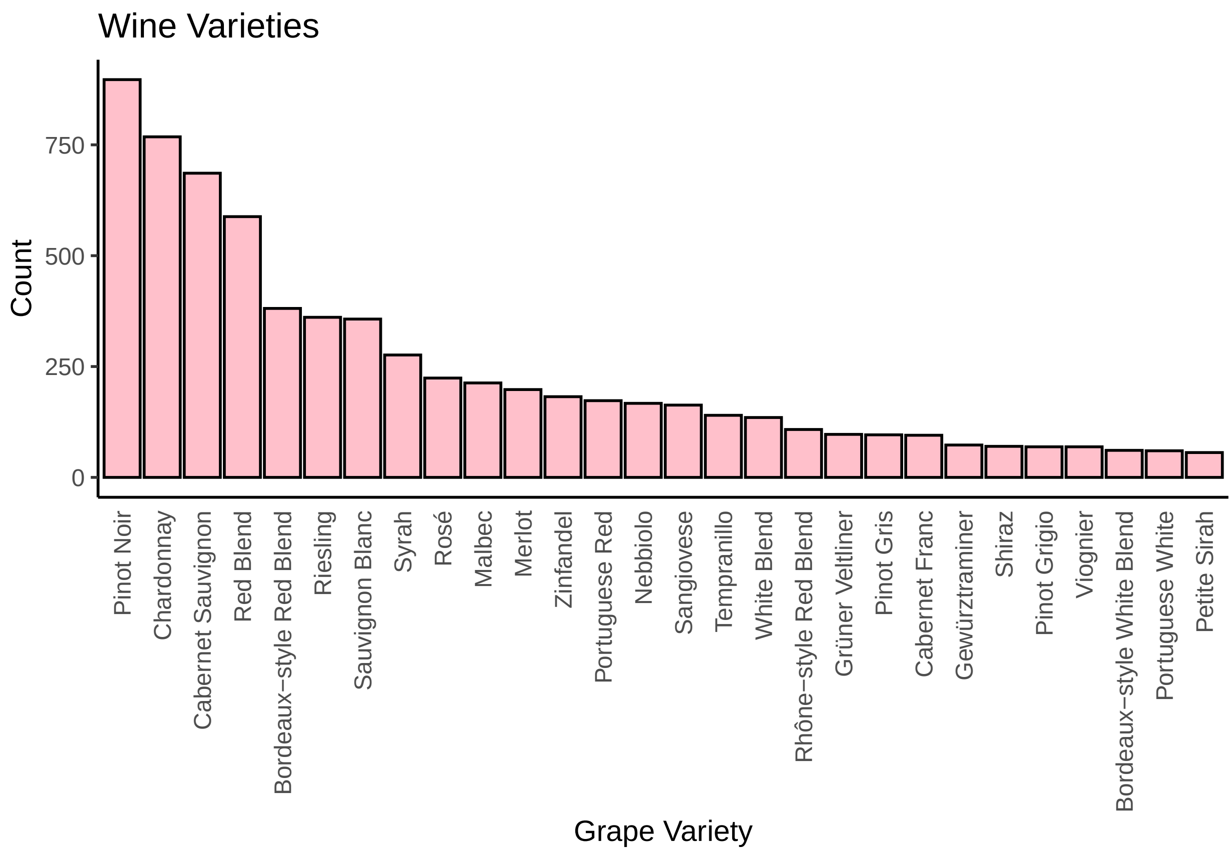
Looking at an ordered plot of average points with error bars at one standard deviation for provinces with more than 50 reviews we see there is some disparity between provinces, although provinces with the highest number of reviews tend to be clustered together around the overall training set average of 88.5. Standard deviation of points for each province is approximately 3 points. The outliers at the high and low ends generally tend to be provinces with very few reviews:

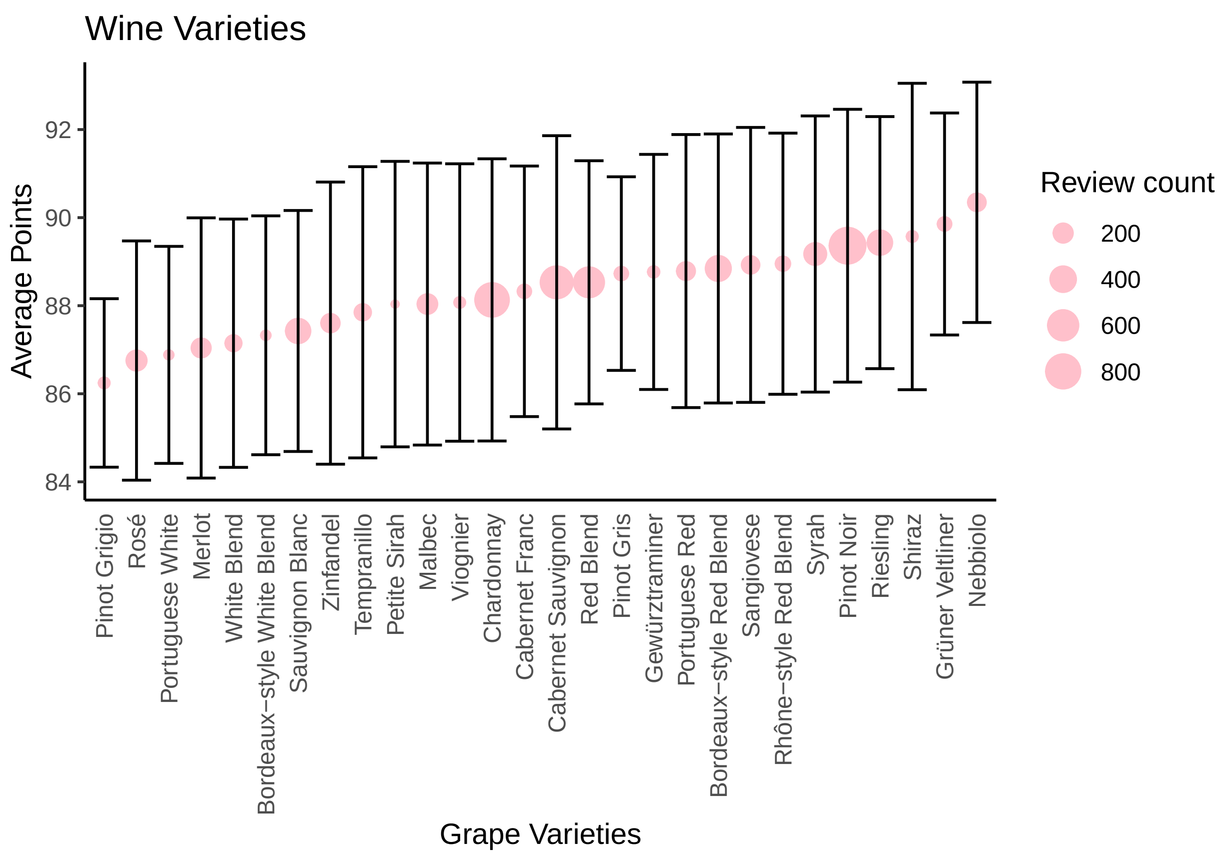
```
## # A tibble: 5 x 4
##   province    Mean_points SD_points Count
##   <fct>             <dbl>     <dbl> <int>
## 1 Eisenberg          94.5     0.707     2
## 2 Colares            93      NA         1
## 3 England            93      NA         1
## 4 Mittelrhein        93      NA         1
## 5 Tulbagh            93      NA         1
```

```
## # A tibble: 8 x 4
##   province              Mean_points SD_points Count
##   <fct>                       <dbl>     <dbl> <int>
## 1 Vale dos Vinhedos            80       NA        1
## 2 Ica                          82.5      2.12     2
## 3 North Carolina               82.5      2.12     2
## 4 Valle de Guadalupe           82.6      1.52     5
## 5 Lolol Valley                 83       NA        1
## 6 San Jose                     83       NA        1
## 7 Tasmania                     83       NA        1
## 8 Ukraine                      83       NA        1
```

Based on this preliminary analysis, wine provinces also look useful in training our algorithms.

We now look at wine grape variety in a similar manner, plotting review counts and average ratings for varieties with more than 50 reviews (given the overwhelming amount of data)





We see that the most popular grape varieties for the reviewers to review are Pinot Noir, Chardonnay, and Cabernet Sauvignon, along with Red Blend, which have average ratings between 88 and 89.5 - around the overall average. The highest average points can be seen in the last 3 grape varieties, Shiraz, Grunwe- Veltilner and Nebbiolo, although these also have the number of reviews on the lower scale. Shiraz alo has one of the longer ranges of average points.

```
## # A tibble: 5 x 4
##   variety             Mean_points SD_points Count
##   <fct>                     <dbl>     <dbl> <int>
## 1 Alsace white blend           94        NA     1
## 2 Syrah-Petite Sirah           94        NA     1
## 3 Jaen                         93        NA     1
## 4 Ramisco                      93        NA     1
## 5 Roviello                     93        NA     1
```

```
## # A tibble: 7 x 4
##   variety                         Mean_points SD_points Count
##   <fct>                                 <dbl>     <dbl> <int>
## 1 Shiraz-Tempranillo                       80     NA        1
## 2 Cabernet Sauvignon-Tempranillo           81     NA        1
## 3 Malvar                                   81     NA        1
## 4 Cabernet Merlot                          82     NA        1
## 5 Garnacha Blanca                          82     NA        1
## 6 Portuguese Rosé                          82      1.41     2
## 7 Tempranillo-Syrah                        82     NA        1
```

Wine grape variety again in general looks to provide useful information for use in our algorithms.

Preliminary analysis conclusion: The wine descriptions will not provide much use in this projects machine learning purpose for predicting wine quality. Therefore, we remove the description as well as the points variables because we are fovusing on the points brackets instead. The variables we do decide to procede further with are `price`, `vintage`, `variety` and `province`.

```
# Remove unrequired variables from training and testing sets and unused objects
wine_train <- wine_train %>% select(-description, -points)
wine_test <- wine_test %>% select(-description, -points)
#definitly remove unused stuff
rm(bing, sample_set_bing,wine_clean)
```

#Step 6: Machine Learning Algorithms

Since categorical variables cannot be used with some of the algorithms, we first need to create dummy variables for each level of the categorical variables `variety` and `province`. We do this by using the `dummyVars()` function in the `caret` package, which creates variables that are either 1 or 0 for each factor level.

```
# Set our training set outcomes in a separate vector
y_train <- wine_train$point_bracket
# Create a dummy variable matrix of predictors for factor variable levels
dummyvars <- dummyVars( ~ price + vintage + variety + province, data = wine_train)
train_dummyvars <- predict(dummyvars, newdata = wine_train)
dim(train_dummyvars)
```

```
## [1] 8340  557
```

	Count
wine_train distinct provinces	230
wine_train distinct varieties	325

Since we are trying to make predicitions on a discrete variable, with 6 classes, we will be looking into classification machine learning methods. For most of the methods, we will be using the `train()` function in the `caret` package.

Since we are unsure about if our classes have distinctly different covariances, but it does seem that our data follows a normal distribution, as we saw in the plots above, we will start with LDA and QDA methods and compare them for accuracy. LDA and QDA are preffered over logistical regression because we have more than 2 classes. However, both these methods are poor at handling categorical predictor variables, so we focus on price and vintage only. Also, since o is small (p = 2) because of the restriction to have continous variables, we should expect both methods to perform well if p also has equal covariances.

QDA MODEL

First the QDA Model. The model fails to run including the `variety` and `province` variables due to insufficient individual factor level datapoints available in our sample, so we utilize only the `price` and `vintage` variables. Both these variables look fairly normal, with one outlier in vintage.

```
# Train QDA model on train_set
set.seed(1)
train_qda <- train(point_bracket ~ price + vintage,
```

```r
                  data = wine_train,
                  method = "qda")
```

```
## Quadratic Discriminant Analysis
##
## 8340 samples
##    2 predictor
##    6 classes: 'Acceptable', 'Classic', 'Excellent', 'Good', 'Superb', 'Very Good'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 8340, 8340, 8340, 8340, 8340, 8340, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.427     0.192
```

We see the accuracy is only .427

Model	Accuracy
QDA	0.427

LDA Model

We get the same warning running the LDA model with discrete variables, so we only include price and vintage.

```r
# Train QDA model on train_set
set.seed(1)
train_lda <- train(point_bracket ~ price + vintage,
                  data = wine_train,
                  method = "lda")
```

```
## Linear Discriminant Analysis
##
## 8340 samples
##    2 predictor
##    6 classes: 'Acceptable', 'Classic', 'Excellent', 'Good', 'Superb', 'Very Good'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 8340, 8340, 8340, 8340, 8340, 8340, ...
## Resampling results:
##
##   Accuracy  Kappa
##   0.435     0.151
```

The accuracy is slightly better at .435

Model	Accuracy
QDA	0.427
LDA	0.435

We see †hat LDA accuracy is slightly higher than QDA. This could indicate that the assumption of common covariance is suitable for this data set.

However,LDA is a much less flexible classifier than QDA, and so has substantially lower variance. This can potentially lead to improved prediction performance. But there is a trade-off: if LDA's assumption that the the predictor variable share a common variance across each Y response class is badly off, then LDA can suffer from high bias.

This is why we could have seen a slight improvement with LDA over QDA. Still, both of these prediction accuracies are fairly low, so we will move onto other, stronger classificantion methods.

Since we only focused on 2 predictors out of the 4 available, the other classification models we chose will be better at handling categorical predictors and hopefully have higher accuracy ratees.

Therefore we move onto the classification tree method.

CLASSIFICATION TREE

The classification and regression tree methods were found while researchers were seeking to solve the problems of other methods such as not handling large data well, too many iterations(not knowing when to stop) and not being able to process categorical predictors (AID,ID3 and CHAID for example.)

We run the `rpart` algorithm , which by default performs a 25-fold cross validation. We can use all the predictors for the model. We set the algorithm to use the tuning parameter, complexity parameter to the 0.05 level, that is if the cost of adding another variable to the decision tree from the current node is above the value of cp, then tree building does not continue.

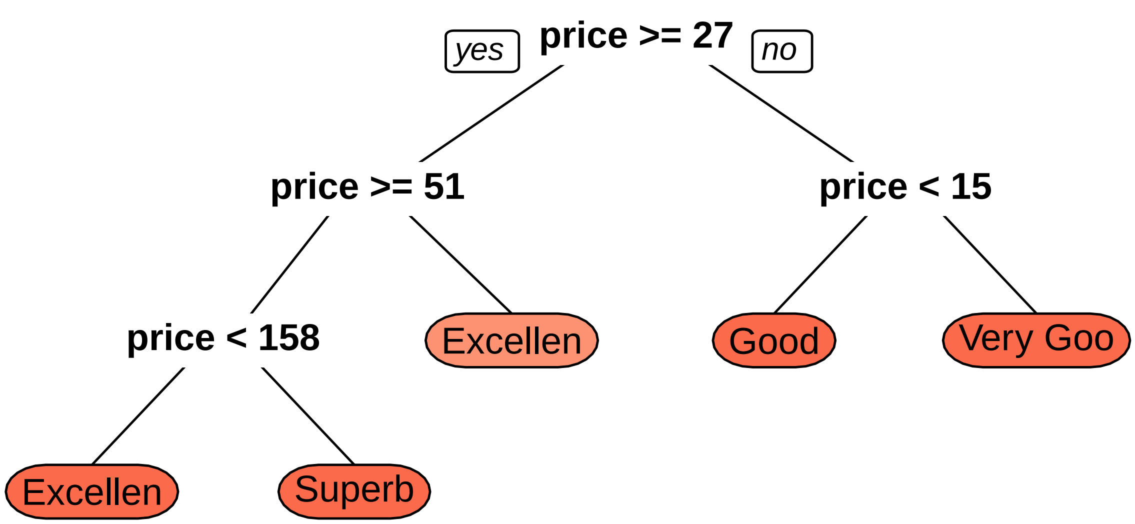
```
# Train a classification tree model on train_set
set.seed(1)
train_class <- train(point_bracket ~ price + province + variety + vintage,
                     method = "rpart",
                     data = wine_train,
                     trControl = trainControl(method = "cv"),
                     tuneGrid = data.frame(cp = seq(0.0, 0.05, len = 20))) #cp parmeter
train_class
```

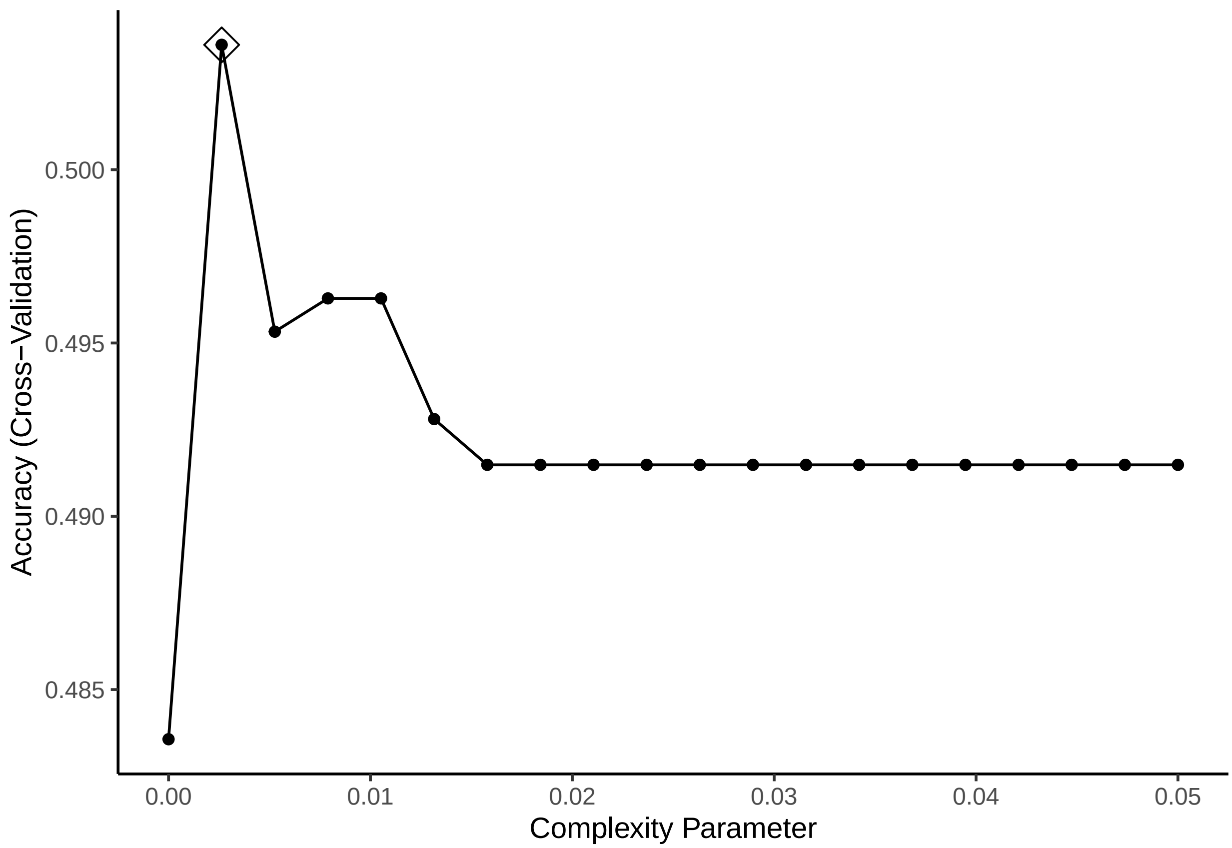
```
## CART
##
## 8340 samples
##    4 predictor
##    6 classes: 'Acceptable', 'Classic', 'Excellent', 'Good', 'Superb', 'Very Good'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 7505, 7505, 7507, 7505, 7506, 7507, ...
## Resampling results across tuning parameters:
##
##   cp       Accuracy  Kappa
##   0.00000  0.484     0.251
##   0.00263  0.504     0.274
##   0.00526  0.495     0.256
##   0.00789  0.496     0.256
```

```
##   0.01053  0.496     0.256
##   0.01316  0.493     0.251
##   0.01579  0.491     0.249
##   0.01842  0.491     0.249
##   0.02105  0.491     0.249
##   0.02368  0.491     0.249
##   0.02632  0.491     0.249
##   0.02895  0.491     0.249
##   0.03158  0.491     0.249
##   0.03421  0.491     0.249
##   0.03684  0.491     0.249
##   0.03947  0.491     0.249
##   0.04211  0.491     0.249
##   0.04474  0.491     0.249
##   0.04737  0.491     0.249
##   0.05000  0.491     0.249
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.00263.
```

The final model uses the lowest cp and the most accuracy.

```
## Loading required package: rpart
```





Price >= 27 is the first split.

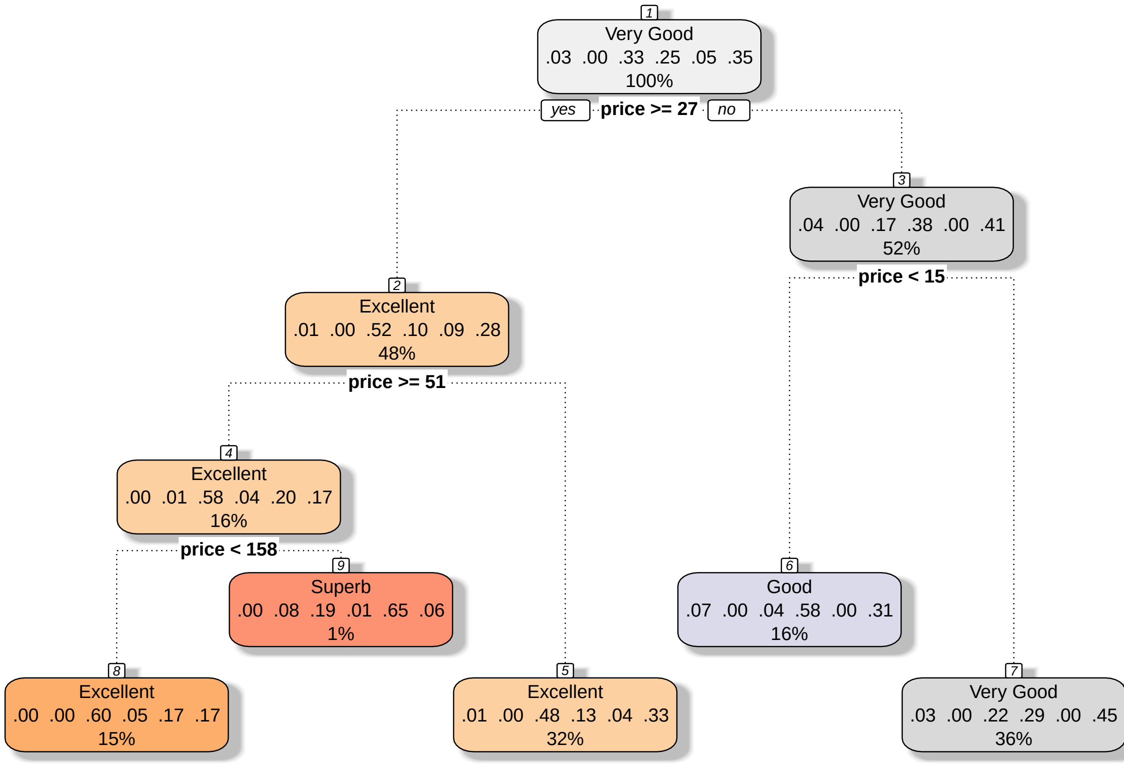
Examining the optimal tree structure, we find that only 4/6 points categories Excellent, Superb, Good and Very Good an are being assigned and `price` is the only variable being used to determine the category.

```
train_class$results %>% filter(Accuracy == max(Accuracy))
```

```
##        cp Accuracy Kappa AccuracySD KappaSD
## 1 0.00263    0.504 0.274     0.0128  0.0201
```

We see that utilizing the optimal complexity parameter for our model, the accuracy is only .504

```
#Another visual representation of the classification tree
suppressMessages(library(rattle))
fancyRpartPlot(train_class$finalModel)
```



Rattle 2020−Nov−29 17:32:17 SarahLittle

Next, we can look at the order of importance of the predictor variables in the model. However it should come at no suprise that price is the first and at 100% importance since it was the only predictor used in the model. Perhaps if we lowered the complexity parameter we would see other variables used in the model.

```
## rpart variable importance
##
##   only 20 most important variables shown (out of 559)
##
##                                  Overall
## price                            100.000
## vintage                           10.197
## provinceCalifornia                 3.252
## varietyPinot Noir                  3.011
## provinceNew York                   2.160
## provinceVirginia                   1.547
## varietyTempranillo                 1.289
## varietyRiesling                    1.241
## provinceWashington                 1.224
## varietySangiovese                  1.143
## varietyPortuguese Red              1.029
## provinceLoire Valley               0.391
## `varietyChenin Blanc-Viognier`     0.000
## provinceTokaji                     0.000
## varietyVespaiolo                   0.000
## provinceGisborne                   0.000
## `varietyGrenache-Mourvèdre`        0.000
```

```
## `provinceNorth Carolina`          0.000
## provinceCephalonia                0.000
## varietyTimorasso                  0.000
```

Examining the variable importance confirms that `price` is overwhelmingly the most important variable in this model.

```
model_results <- bind_rows(model_results,
                           data_frame(Model="Classification Tree",
                                      Accuracy = max(train_class$results$Accuracy)))
model_results %>% knitr::kable()
```

Model	Accuracy
QDA	0.427
LDA	0.435
Classification Tree	0.504

RANDOM FOREST

Next, we run a random forest supervised machine learning algorithm. The forest it builds is a combination of decision trees through bagging methods. This means that the combination of learning models hopefully increases overall results

Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Even though this multiple decision process takes longer than the classification tree, it often results in a more accuracte model.

We look at a Random Forest model using the `Rborist` package. Rborsit handles large n with a smaller p. Unforutuatly, due to cost and time restraints, we will limit the algorithm to a 3-fold cross validation, reduce the number of trees to 50, and take a random subset of 500 observations when constructing each tree in order to save on computation time.

We run the algorithm with the number of predictors tuning parameter `predFixed` over a range of 20 to 80, and minimum node sizes of 2, 6, and 10 under the `minNode` parameter:

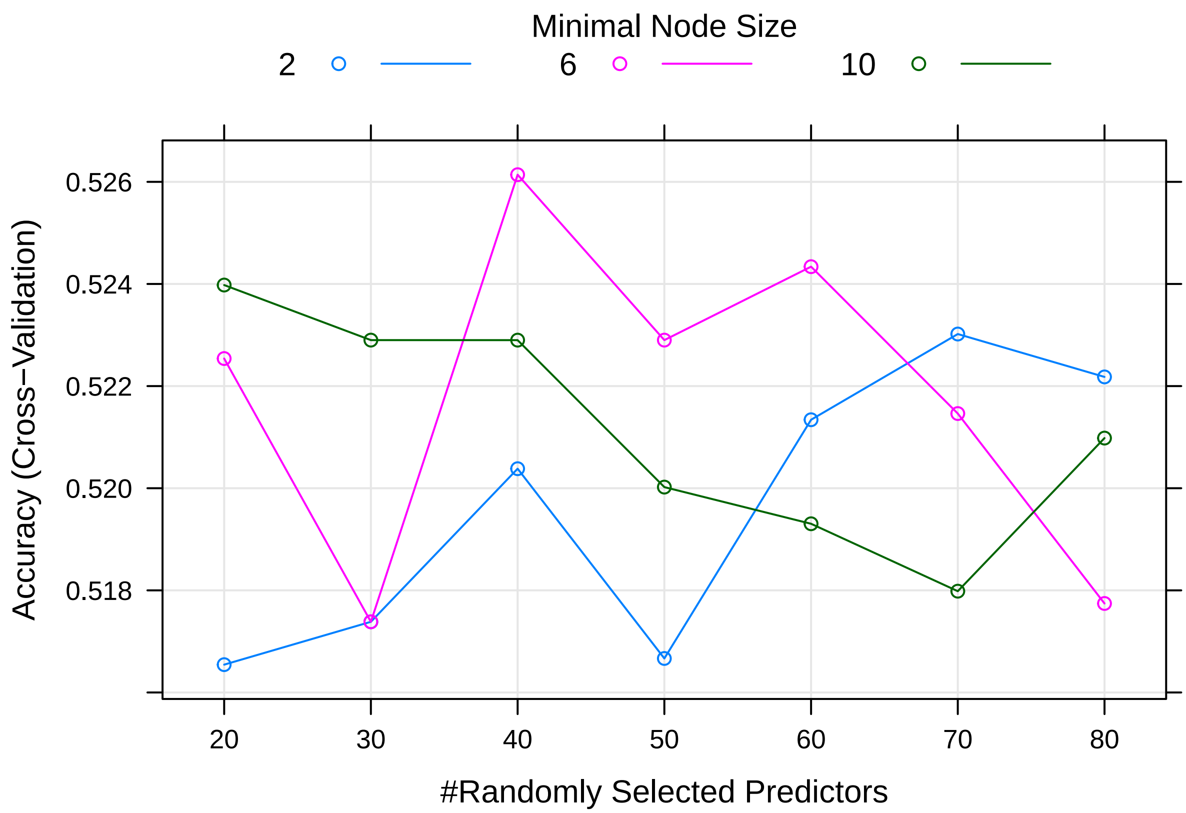
```
set.seed(1)
# Set to 3-fold cross validation and our tuning parameter values to test
trcontrol <- trainControl(method="cv", number = 3)
grid <- expand.grid(minNode = c(2,6,10) , predFixed = seq(20,80,10))
# Train random forest model on train_set with 50 trees sampling 500 rows each


train_rf <- train(point_bracket ~ price + variety + province + vintage,
                    method = "Rborist",
                    data = wine_train,
                 tuneGrid = grid,
                    nTree = 50, #usually is 500 but we restrict to 50 to save on computation time
                    trControl = trcontrol,
                    nSamp = 500)
```

```
## Random Forest
##
## 8340 samples
##    4 predictor
##    6 classes: 'Acceptable', 'Classic', 'Excellent', 'Good', 'Superb', 'Very Good'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 5560, 5561, 5559
## Resampling results across tuning parameters:
##
##   minNode  predFixed  Accuracy  Kappa
##    2       20         0.517     0.292
##    2       30         0.517     0.296
##    2       40         0.520     0.299
##    2       50         0.517     0.293
##    2       60         0.521     0.302
##    2       70         0.523     0.304
##    2       80         0.522     0.302
##    6       20         0.523     0.302
##    6       30         0.517     0.294
##    6       40         0.526     0.308
##    6       50         0.523     0.304
##    6       60         0.524     0.305
##    6       70         0.521     0.301
##    6       80         0.518     0.295
##   10       20         0.524     0.303
##   10       30         0.523     0.303
##   10       40         0.523     0.304
##   10       50         0.520     0.300
##   10       60         0.519     0.298
##   10       70         0.518     0.297
##   10       80         0.521     0.301
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were predFixed = 40 and minNode = 6.
```

We see that our optimal model is using 40 predictors and a minimum node size of 6, resulting in an accuracy of 0.526.

```
## Rborist variable importance
##
##   only 20 most important variables shown (out of 555)
##
##                               Overall
## price                          100.00
## vintage                         34.84
## provinceCalifornia               9.51
## varietyChardonnay                9.21
## varietyCabernet Sauvignon        5.39
## varietyPinot Noir                5.18
## provinceWashington               4.80
## varietyRed Blend                 3.99
## provinceBordeaux                 3.97
```

```
## provinceMendoza Province          3.97
## varietyTempranillo                3.56
## provinceTuscany                   3.30
## varietyMerlot                     3.14
## provinceSouthwest France          3.02
## provinceVirginia                  2.98
## provinceOregon                    2.97
## varietyRosé                       2.83
## provinceNew York                  2.51
## varietyBordeaux-style Red Blend   2.45
## provinceAlsace                    2.43
```



Looking at the variable importance for the random forest model, we again see that price is the most important variable, followed by vintage. We se that the highest accuracy is achieved with a minNode size = 6 and predFix = 40. This is our highest accuracy yet at .526

Model	Accuracy
QDA	0.427
LDA	0.435
Classification Tree	0.504
Random Forest	0.526

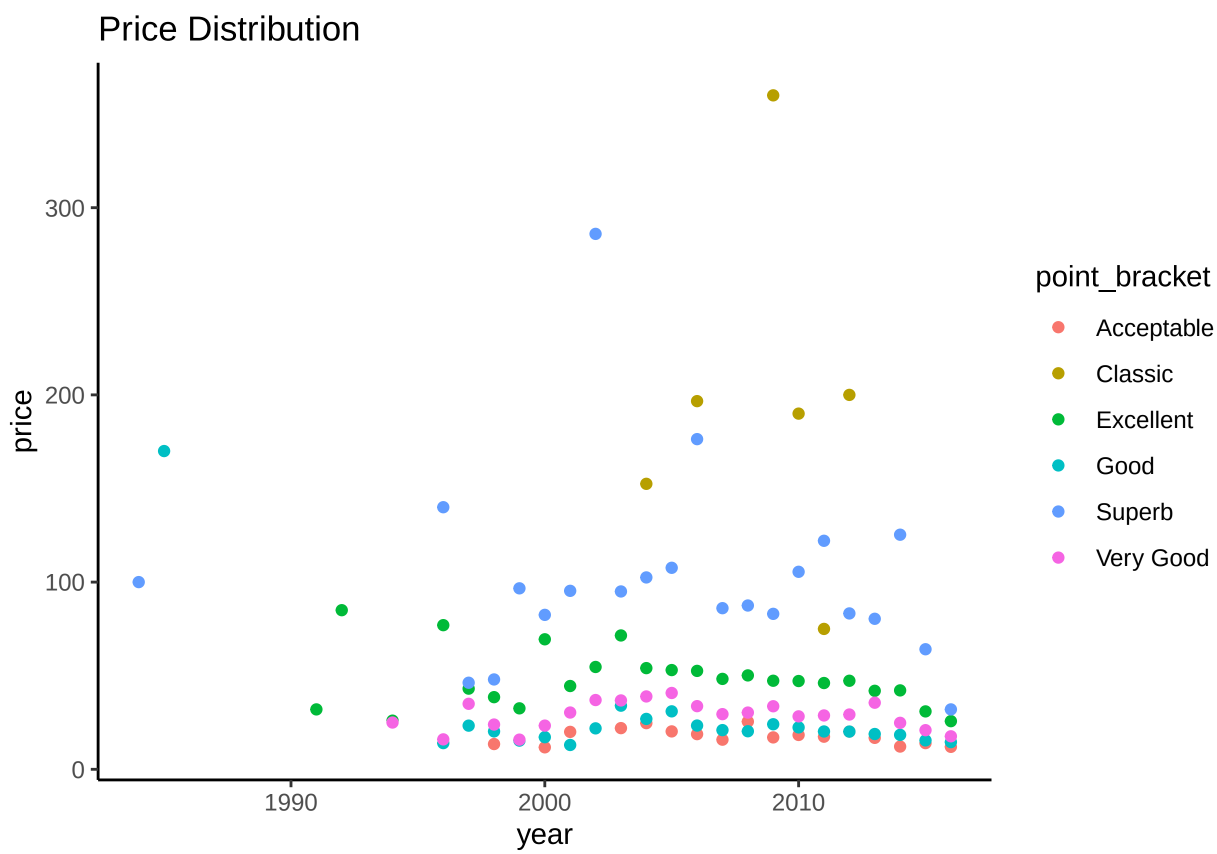
This is a classic example where collective decision making outperformed a single decision-making process.

SVM Models

Lastly, we will try the supported vector machine method.

First we visualize the 2 numeric predictors we have because they seem to be the most important by far compared to the other predictors.

```
#visulaize the predictors
wine_train %>% group_by(point_bracket, vintage) %>%
  filter(vintage >= 1975) %>% #focus on where the large amount of data is
  summarise(price = mean(price)) %>%
  ggplot(aes(vintage,price)) +
  geom_point(aes(color = point_bracket)) +labs(title = "Price Distribution", x= "year", y = "price")
```



As we can see from the graph, we clearly cannot seperate the classes by a hyperplane, so we go with the svm method.

First we try ti find the optimal kernal. We look at costs .1,1 and 10 to save time and only focus on the radial kernal (since the data does not appear linear at all we need to go with a non linear method) to also save computational time. To save time, we change the k fold cross validation to 5 instead of the automatic 10.

```
# install.packages('e1071')
set.seed(1)
library(e1071)
d = wine_train[c(1,4:5)] #data set that only comprises of the variables we need for training
d2 = wine_test[c(1,4:5)] #subset of the testing dataset
tc <- tune.control(cross =5)
Stuned = tune( svm, point_bracket ~ price + vintage, data=d, ranges=list(cost=10^seq(-1,1)),
```

```r
kernel="radial", tunecontrol = tc)
summary(Stuned)
```

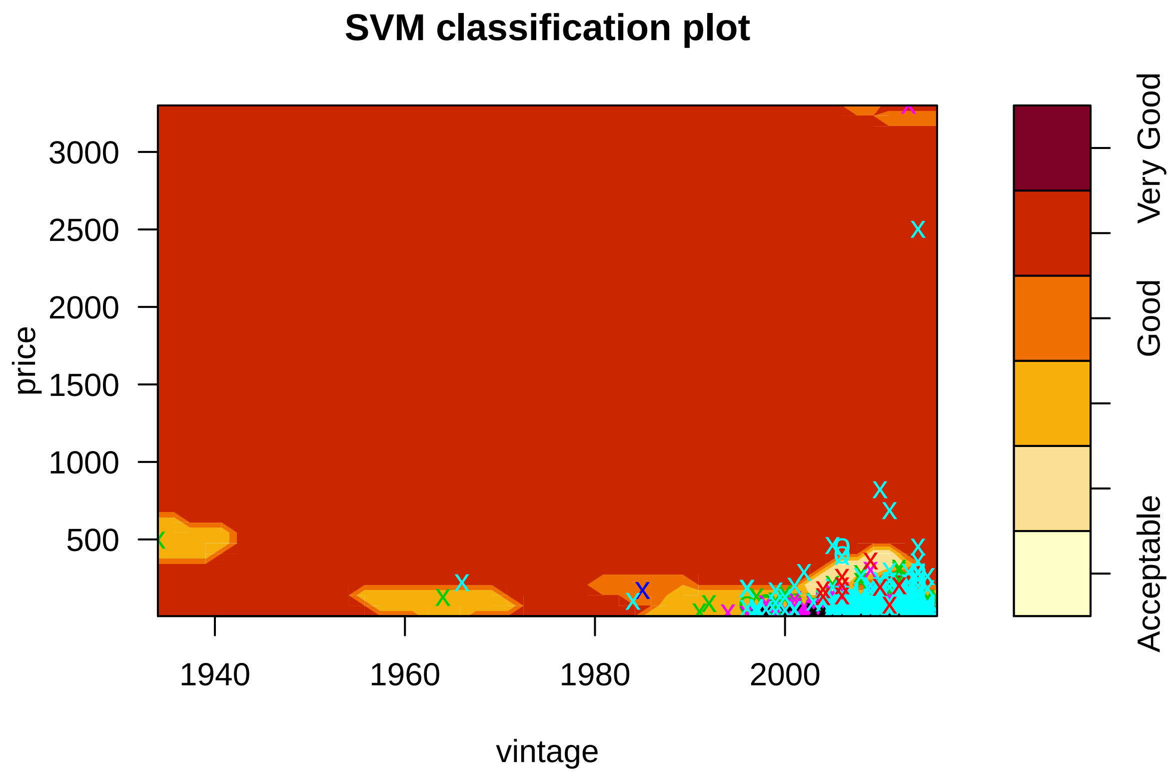
```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.495
##
## - Detailed performance results:
##   cost error dispersion
## 1  0.1 0.520     0.0134
## 2  1.0 0.501     0.0127
## 3 10.0 0.495     0.0133
```

The optimal cost is 10 and the optimal kernal is radial (given that it is the only one we tested).

Next we graph the SVM optimal plot. We see the number of support vectors is 7655

```r
Soptimal = svm( point_bracket ~ price + vintage, data=d, cost=10, kernel="radial" )
summary(Soptimal); plot(Soptimal,data=d)
```

```
##
## Call:
## svm(formula = point_bracket ~ price + vintage, data = d, cost = 10,
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  7655
##
##  ( 2238 1925 2895 214 374 9 )
##
##
## Number of Classes:  6
##
## Levels:
##  Acceptable Classic Excellent Good Superb Very Good
```



Now we look at the accuracy

```
attach(wine_train)
Yhat = predict(Soptimal, data=wine_test[d2,])
table( Yhat, point_bracket)
```

```
##              point_bracket
## Yhat          Acceptable Classic Excellent Good Superb Very Good
##   Acceptable           0       0         0    0      0         0
##   Classic              0       4         1    0      1         0
##   Excellent           22       3      1942  326    322       977
##   Good               126       0       105  864      1       525
##   Superb               0       2        10    0     47         3
##   Very Good           66       0       722  870     11      1390
```

```
svm_accuracy <- mean( Yhat==point_bracket )
```

The accuracy is about .509.

Model	Accuracy
QDA	0.427
LDA	0.435
Classification Tree	0.504
Random Forest	0.526
SVM	0.509

SVM accuracy falls at a close second with .509 accuracy

Overall, our model with the highest accuracy is the Random Forest model and we will proceed to test this model on our testing set.

3. RESULTS

We chose this package because it allows for unlimited factor levels

We first optimize a final Random Forest model on the training set utilizing the `Rborist()` function, setting our optimal parameters from the previous cross validation and the number of trees to be 500. The `Rborist()` function again needs us to use the dummy variables for each of the factor levels.

```
# Train final random forest model using train_set dummy variables and outcomes with 500 trees
library(Rborist)
```

```
## Rborist 0.2-3
```

```
## Type RboristNews() to see new features/changes/bug fixes.
```

```
final_model <- Rborist(x = train_dummyvars,
                       y = y_train,
                       nTree = 500,
                       predFixed = train_rf$bestTune$predFixed,
                       minNode = train_rf$bestTune$minNode)
```

Now we create a matrix of predictors including dummy variables for our testing set, `test_dummyvars`, and predict our wine points categories for the test set using our final model with the `predict()` function:

```
# Create a dummy variable matrix of predictors for factor variable levels in the testing set
dummyvars <- dummyVars( ~ price + vintage + variety + province, data = wine_test)
test_dummyvars <- predict(dummyvars, newdata = wine_test)
# Create our model predictions for the testing set using the final model
y_hat <- as.factor(predict(final_model, test_dummyvars)$yPred)
cm <- confusionMatrix(y_hat, wine_test$point_bracket)
cm
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Acceptable Classic Excellent Good Superb Very Good
##   Acceptable          0       0         0    0      0         0
##   Classic             0       0         0    0      0         0
##   Excellent          10       2       744  151    124       462
##   Good               46       0        19  300      0       172
##   Superb              0       3        13    0     18         2
##   Very Good          18       0       274  330      5       570
##
## Overall Statistics
##
##                Accuracy : 0.5
##                  95% CI : (0.483, 0.517)
```

```
##     No Information Rate : 0.37
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.262
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Acceptable Class: Classic Class: Excellent
## Sensitivity                     0.0000        0.00000            0.709
## Specificity                     1.0000        1.00000            0.662
## Pos Pred Value                     NaN            NaN            0.498
## Neg Pred Value                  0.9773        0.99847            0.827
## Prevalence                      0.0227        0.00153            0.322
## Detection Rate                  0.0000        0.00000            0.228
## Detection Prevalence            0.0000        0.00000            0.458
## Balanced Accuracy               0.5000        0.50000            0.685
##                      Class: Good Class: Superb Class: Very Good
## Sensitivity               0.3841       0.12245            0.473
## Specificity               0.9045       0.99422            0.695
## Pos Pred Value            0.5587       0.50000            0.476
## Neg Pred Value            0.8236       0.96002            0.692
## Prevalence                0.2394       0.04505            0.370
## Detection Rate            0.0919       0.00552            0.175
## Detection Prevalence      0.1646       0.01103            0.367
## Balanced Accuracy         0.6443       0.55834            0.584
```

```r
final_results <- data_frame(Model="Random Forests",
                            Accuracy = cm$overall['Accuracy'])
final_results %>% knitr::kable()
```

Model	Accuracy
Random Forests	0.5

Our final results show a 50% accuraccy on the testing set. The sensitivity (true positive rate) or the proportion of positive results out of the number of samples which were actually positive is 0 for the Accebtable and Classic classes (the ones with the least amount of data). But it is over 70% for the excellent class, 38% for Good, 12% for Superb and 47% for very good. The specificity (true negative rate) is the proportion of truly negative cases that were classified as negative; thus, it is a measure of how well your classifier identifies negative cases. This was better across all classes, with the lowest for Excellent at 66%, the most populated class. The detection rate, he number of correct positive class predictions made as a proportion of all of the predictions made.

#Conclusion

In this project we constructed a machine learning algorithm to predict wine points score categories for wines with unknown titles and wineries in a data set of ratings sourced from Kaggle with nearly 130,000 wine reviews from the Wine Enthusiast Magazine website.

After initially inspecting the data and performing some data cleaning, we took a subset of over 11,000 reviews in order to have a more practical sample size for model fitting purposes and built training and testing sets in a 75:25 proportion. We then analyzed and visualized the different variables in some detail to explore their link to wine points scores and determined that price, vintage year, province of origin, and grape variety all looked to have an impact on score.

We then trained various machine learning algorithms on the training set, including QDA, LDA, a classification tree model, random forest, SVM, and found that the random forest model indicated the best accuracy performance on our training set.

Finally, we ran a random forest model trained on our training set on the testing set and found a final accuracy value of 50%. This final value is lower than we might have hoped for, where just over half of the time the correct points category is predicted. Ideally, we would want to have an accuracy value of greater than 0.8 or 0.9 in order to have a more useful model.

To improve our model results, we could use a larger sample size for training, up to say over 100,000 reviews. This would, of course, greatly increase the computing time needed to train the models, in particular the computationally intense random forest and SVM models. We could also include more variables in our models, such as country, sub-regional detail where available, and winery if we wanted our models to use that information. The text descriptions may also be able to be used if we explored the text sentiment analysis further and found it to be helpful at scale.

In addition, further machine learning models could be explored which were not tried out in this study, including k-means clustering, neural networks, and a matrix factorization model using singular value decomposition and principal component analysis. We would again, however, need to find a sample size to strike a balance between a practical amount of computing time and model accuracy.