Wine Ratings Predictions Project

Joseph Wellman

04 June 2020

1. INTRODUCTION

In this project, we will study a data set of wine ratings available on Kaggle (https://www.kaggle.com/zynicide/wine-reviews/data). The data set comprises information on wine reviews scraped from the Wine Enthusiast Magazine website (https://www.winemag.com/ratings/) on November 22nd, 2017.

Our data set contains nearly 130,000 reviews written by wine critics for the magazine with information on wine grape variety, price, place of origin, and several others. The full text of the review is included for each entry, as well as a points score between 80-100, which can be categorized in the following six categories (see https://www.winemag.com/wine-vintage-chart/):

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

Wines scoring below 80 are not reviewed or recommended by the magazine.

We will look to build a machine learning algorithm which will predict one of the six score categories for wines with unknown titles or wineries by utilizing other available variables in the data set. We will first explore the data set in some detail to undertake necessary data cleaning and to determine which variables are useable based on the available data.

Then we will construct training and testing subsets from the full data set with points scores stratified by category, and perform some data visualization to evaluate each variable's usefulness in predicting wine score categories.

Several machine learning algorithms will then be trained on the training set and we will assess them based on their accuracy rate in determining correct points categories.

Finally, we will select one final machine learning model to run on our testing set, review its accuracy performance, and provide suggestions for improvements or next steps for further study.

2. DATA ANALYSIS AND ALGORITHM METHODS

2.1 DATA ANALYSIS

Examining the structure of the raw data set below, we see that it includes 129,971 entries across 14 variables:

- `X`: integer referencing the row number;
- `country`: factor providing the country where the wine was produced;
- `description`: factor containing the text of the review written by the reviewer as originally posted on https://www.winemag.com/ratings/;
- `designation`: factor of any special designations given by the winery in the wine's name;
- `points`: integer giving the wine score of the review;
- `price`: numeric providing the price per bottle of the wine in US Dollars;
- `province`: factor providing the province of the country where the wine was produced;
- `region_1`: factor for regional detail where the wine was produced;
- `region_2`: factor for further sub-regional detail where the wine was produced;
- `taster_name`: factor giving the name of the reviewer;
- `taster_twitter_handle`: factor giving the twitter handle of the reviewer;
- `title`: factor giving the title of the wine beginning with the name of the winery, vintage year, and then the name of the wine;
- `variety`: factor giving the grape variety of the wine; and
- `winery`: factor giving the name of the winery that produced the wine.

```r
str(dataset)
```

```
## 'data.frame':    129971 obs. of  14 variables:
##  $ X                    : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ country              : Factor w/ 44 levels "","Argentina",..: 24 33 44 44 44 39 24 17 19 17 ...
##  $ description          : Factor w/ 119955 levels "\"Chremisa,\" the ancient name of Krems, is commen
##  $ designation          : Factor w/ 37980 levels "","'61 Rosé",..: 36977 2353 1 28124 36714 1997 305
##  $ points               : int  87 87 87 87 87 87 87 87 87 87 ...
##  $ price                : num  NA 15 14 13 65 15 16 24 12 27 ...
##  $ province             : Factor w/ 426 levels "","Achaia","Aconcagua Costa",..: 335 111 270 221 270
##  $ region_1             : Factor w/ 1230 levels "","Abruzzo","Adelaida District",..: 426 1 1219 552 
##  $ region_2             : Factor w/ 18 levels "","California Other",..: 1 1 18 1 18 1 1 1 1 1 ...
##  $ taster_name          : Factor w/ 20 levels "","Alexander Peartree",..: 11 17 16 2 16 14 11 17 3 1
##  $ taster_twitter_handle: Factor w/ 16 levels "","@AnneInVino",..: 6 12 9 1 9 14 6 12 1 12 ...
##  $ title                : Factor w/ 118840 levels ":Nota Bene 2005 Una Notte Red (Washington)",..: 7
##  $ variety              : Factor w/ 708 levels "","Abouriou",..: 693 453 440 482 444 592 189 212 212
##  $ winery               : Factor w/ 16757 levels ":Nota Bene","1+1=3",..: 11641 12988 13054 14432 14
```

We show the first six entries of the data set below for illustration:

X	country	designation	points	price	province	region_1
0	Italy	Vulkà Bianco	87	NA	Sicily & Sardinia	Etna
1	Portugal	Avidagos	87	15	Douro	
2	US		87	14	Oregon	Willamette Valley
3	US	Reserve Late Harvest	87	13	Michigan	Lake Michigan Shore
4	US	Vintner's Reserve Wild Child Block	87	65	Oregon	Willamette Valley
5	Spain	Ars In Vitro	87	15	Northern Spain	Navarra

region_2	title
	Nicosia 2013 Vulkà Bianco (Etna)
	Quinta dos Avidagos 2011 Avidagos Red (Douro)
Willamette Valley	Rainstorm 2013 Pinot Gris (Willamette Valley)
	St. Julian 2013 Reserve Late Harvest Riesling (Lake Michigan Shore)
Willamette Valley	Sweet Cheeks 2012 Vintner's Reserve Wild Child Block Pinot Noir (Willamette Valley)

region_2	title
	Tandem 2011 Ars In Vitro Tempranillo-Merlot (Navarra)

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

description
Aromas include tropical fruit, broom, brimstone and dried herb. The palate isn't overly expressive, offering unripened apple, citrus and dried sage alongside brisk acidity.
This is ripe and fruity, a wine that is smooth while still structured. Firm tannins are filled out with juicy red berry fruits and freshened with acidity. It's already drinkable, although it will certainly be better from 2016.
Tart and snappy, the flavors of lime flesh and rind dominate. Some green pineapple pokes through, with crisp acidity underscoring the flavors. The wine was all stainless-steel fermented.
Pineapple rind, lemon pith and orange blossom start off the aromas. The palate is a bit more opulent, with notes of honey-drizzled guava and mango giving way to a slightly astringent, semidry finish.
Much like the regular bottling from 2012, this comes across as rather rough and tannic, with rustic, earthy, herbal characteristics. Nonetheless, if you think of it as a pleasantly unfussy country wine, it's a good companion to a hearty winter stew.
Blackberry and raspberry aromas show a typical Navarran whiff of green herbs and, in this case, horseradish. In the mouth, this is fairly full bodied, with tomatoey acidity. Spicy, herbal flavors complement dark plum fruit, while the finish is fresh but grabby.

Noting that the wine vintage year is included in the title labels, and that this is a particularly important factor when it comes to evaluating wines, we extract the vintage year data from the wine titles and save this in a new variable named `vintage`. 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:

```
# Extract vintage year information from wine titles
dataset <- dataset %>% mutate(vintage = as.numeric(substr(title, str_length(winery)+2,
                                                          str_length(winery)+5)))
dataset %>% select(title, vintage) %>% head() %>% knitr::kable()
```

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

We also notice from examining the structure of the data set above that it appears there are a number of blank or missing entries in several of the variables. We check the number for each variable and summarise this in the table below:

Variable	Blank_or_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
variety	1
winery	0
vintage	4630

In evaluating the results above, we see that more than half of the entries (79,460) have missing data for `region_2`, 21,247 entries have missing `region_1` data, 37,465 have missing `designation` data, 8,996 have missing `price` data, and 4,630 have missing `vintage` data. Only 63 entries have missing `country` and `province` data, while just one entry has missing `variety` data. We will not focus on the `taster_name` or `taster_twitter_handle` in this study, and so we can ignore these missing entries.

Based on the available data in the data set, it looks appropriate to ignore the variables `region_1`, `region_2`, `designation`, `taster_name`, and `taster_twitter_handle` in predicting our wine ratings. We will also ignore the `title` and `winery` variables, since our models will be designed to predict wine quality for wines with previously unseen titles and wineries through utilizing other factors.

We filter our data set to include only non-blank entries for `price`, `country`, `province`, `variety`, and `vintage`:

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

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

We see there are now 116,761 entries remaining, and now re-check the number of blank or NA entries for each variable:

Variable	Blank_or_NA
country	0
description	0
designation	34474
points	0
price	0
province	0

Variable	Blank_or_NA
region_1	19062
region_2	67221
taster_name	23562
taster_twitter_handle	28378
title	0
variety	0
winery	0
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

Looking at province names, we want to check that each province name applies only to one country (for example there is not a case of a province name "Southern", say, which is used in more than one country). We can check this by using the `unite()` function to create a temporary new variable, `country_province`, and then counting the number of distinct entries. If the number is greater than the number of distinct provinces, then there is a province name used in more than one country:

```
dataset %>% unite(country_province, country, province, sep = "_") %>%
  summarise(n_distinct(country_province)) %>%
  knitr::kable()
```

n_distinct(country_province)
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 country information is implied in the `province` variable and may make the `country` variable unnecessary in our algortithms.

Now, we proceed to build training set and a test set from our original full data set.

We first reduce the size of our data set to 10,000 entries in order to provide a more practical sample size to run our machine learning algorithms in a manageable amount of time. We also add a `category` variable which contains a stratified score by category A-F according to the table below:

Score	Category
98-100	A
94-97	B
90-93	C
87-89	D

Score	Category
83-86	E
80-82	F

We then construct a training set containing 80% of the entries and a testing set containing approximately 20%, discarding entries with provinces and grape varieties that do not appear in the training set. We choose an 80:20 split in order to have a substantial proportion of the data entries available in our training set to train the models:

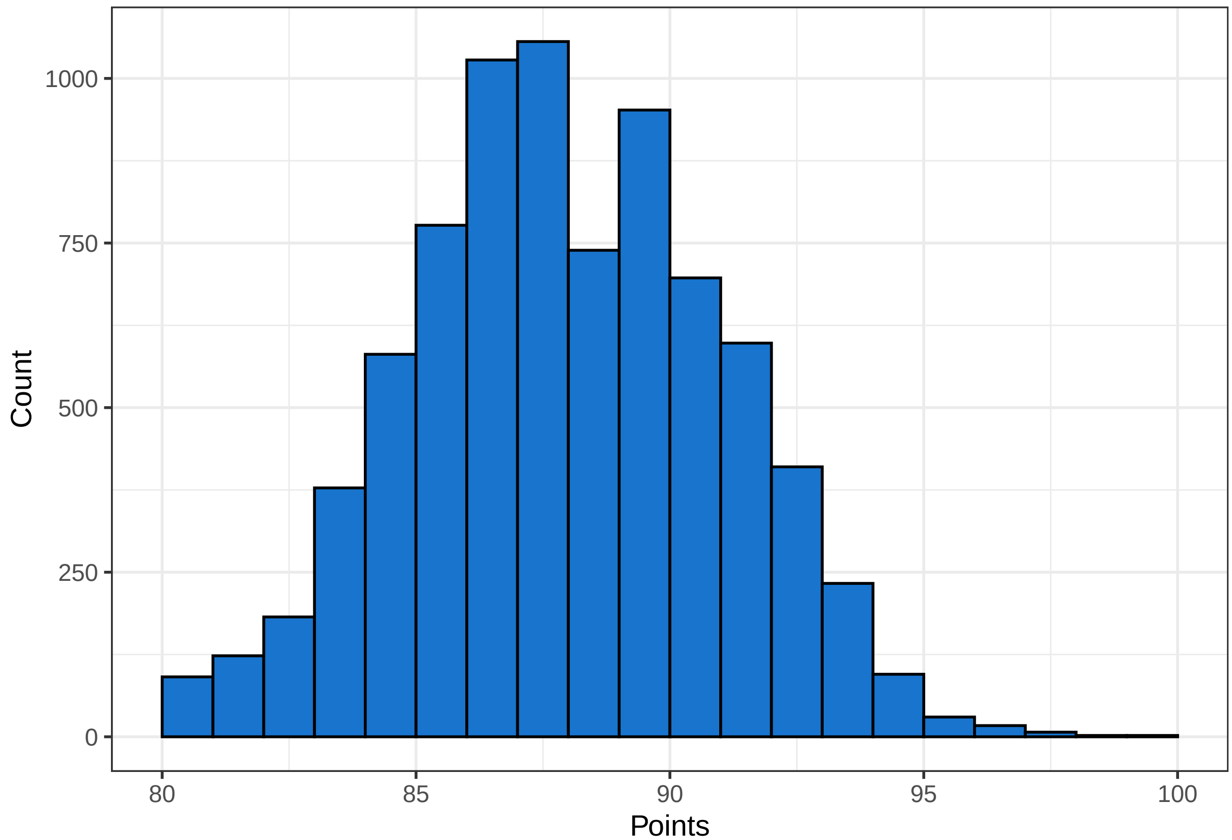
```
# Set sample seed to 1 for replicability
set.seed(1, sample.kind="Rounding")

# We filter the data set to only include the columns that will be relevant for our study,
# then sample 10,000 entries and add a category variable with the stratified scores
dataset <- dataset %>% select(-X, -designation, -country, -region_1, -region_2,
                              -taster_name, -taster_twitter_handle, -title, -winery) %>%
  sample_n(10000) %>%
  mutate(category = as.factor(case_when(points > 97 ~ "A",
                              points > 93 ~ "B",
                              points > 89 ~ "C",
                              points > 86 ~ "D",
                              points > 82 ~ "E",
                              points >= 80 ~ "F")))

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

# We create a test index using 20% of the entries in the dataset
test_index <- createDataPartition(y = dataset$points, times = 1, p = 0.2, list = FALSE)

# We then create the training and testing sets using the test index
train_set <- dataset[-test_index,]
test_set <- dataset[test_index,]

# Make sure that provinces and varieties in the test set are also in the training set
test_set <- test_set %>%
  semi_join(train_set, by = "province") %>%
  semi_join(train_set, by = "variety")

# Remove unused factors for the variety and province variables in the training and test sets
train_set <- train_set %>% mutate(variety = droplevels(variety),
                                  province = droplevels(province))
test_set <- test_set %>% 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(test_set$variety) <- levels(train_set$variety)
levels(test_set$province) <- levels(train_set$province)

# Removing the dataset and test_index objects to clean up since they are no longer needed
rm(dataset, test_index)
```

Data_set	No_entries
train_set	7998
test_set	1961

We see that the training set and test set contain data entries in approximately an 80:20 proportion.

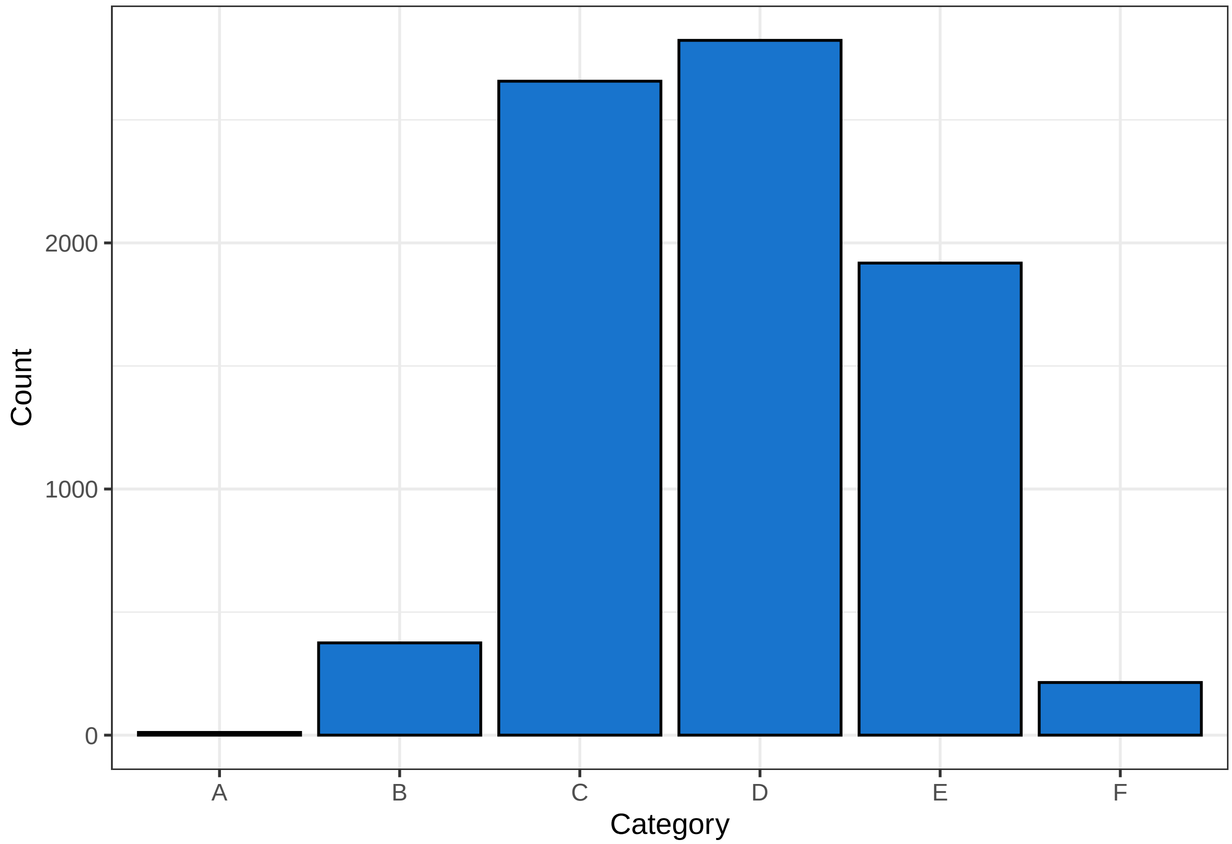
Now we begin visualizing the data in our training set to further understand the links between the different variables and wine ratings.

Firstly, looking at the distribution of wine review scores (points), we plot a histogram below:



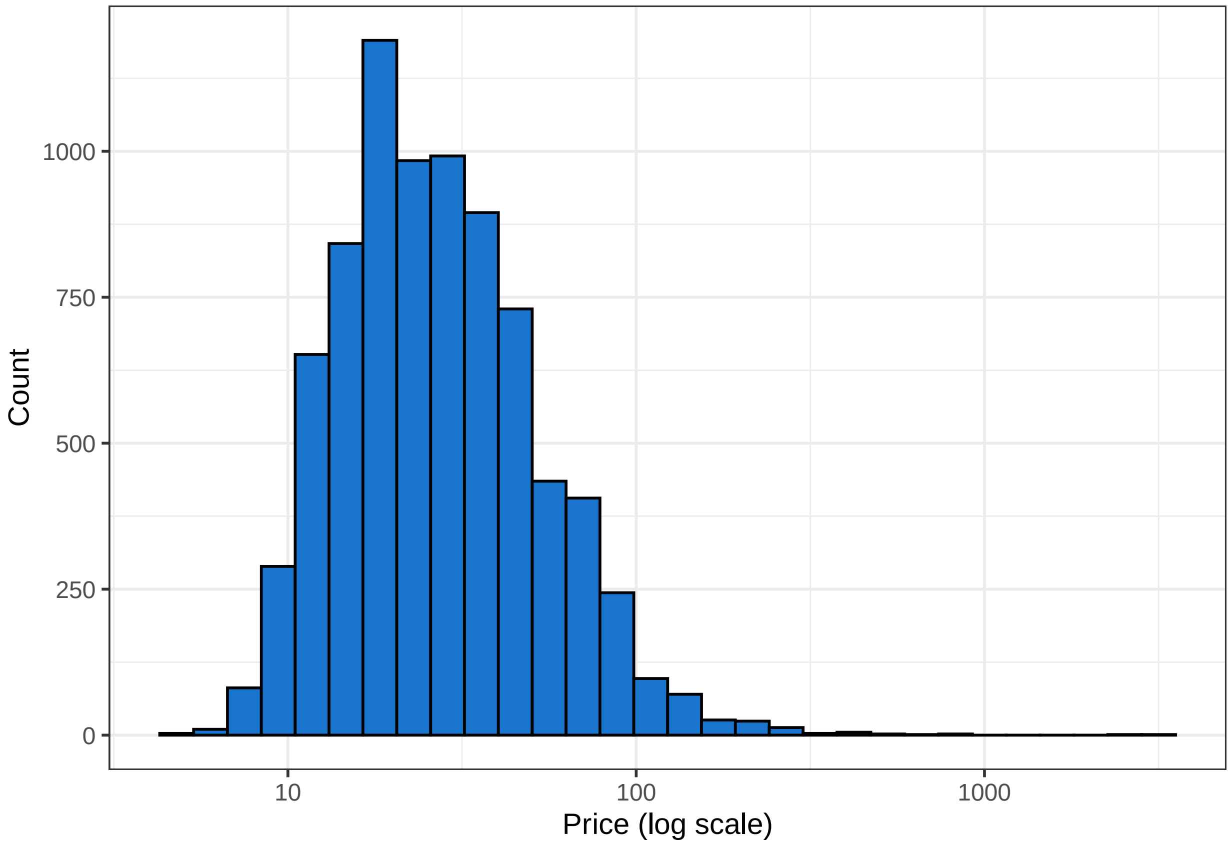
Mean points	SD points	Min points	Max points
88.45	3.0802	80	100

We can see that the distribution of points somewhat resembles a normal distribution, albeit with two peaks, with a mean of 88.5 and standard deviation of 3.08.



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

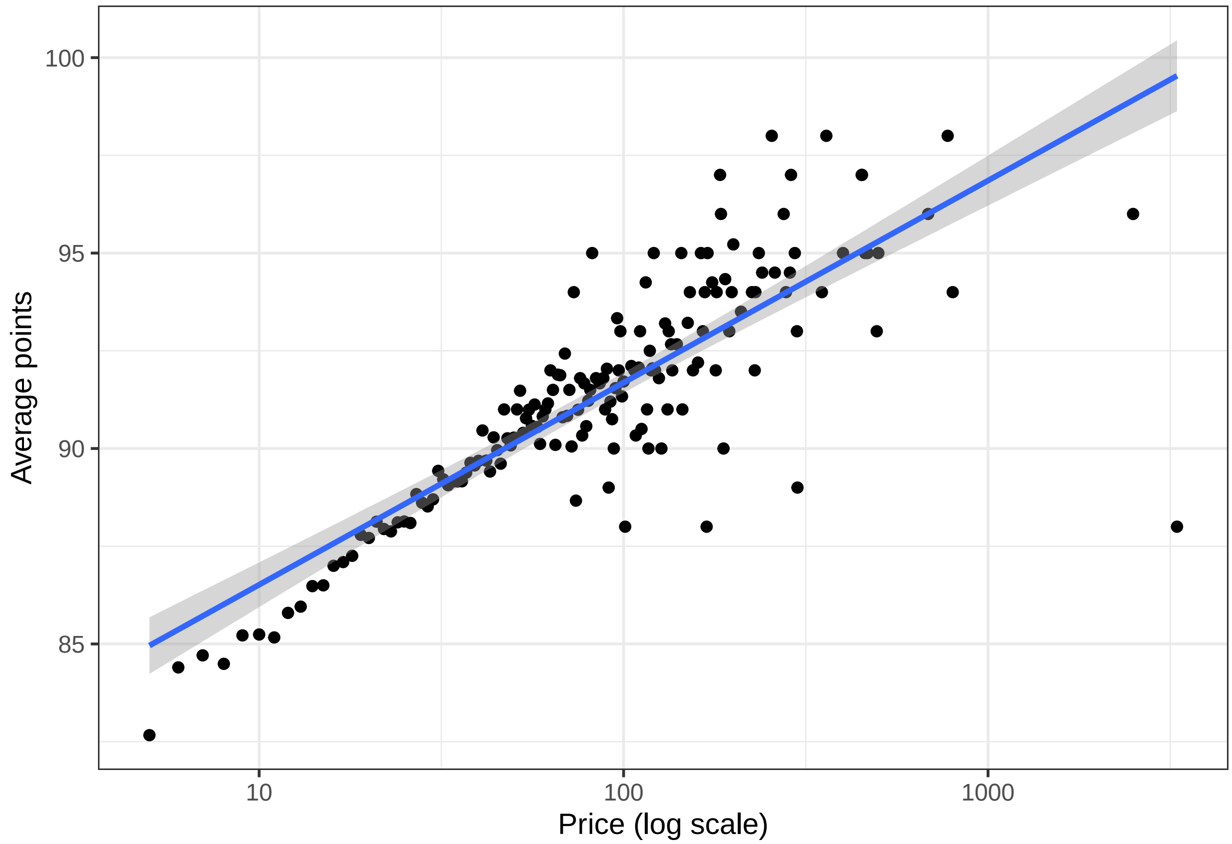
Now looking at wine prices, we plot the distribution of wine prices in our data set:



Mean price	SD price	Min price	Max price
35.706	57.073	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 with a heavy skew towards the lower end with a mean of 35.7 and a standard deviation of 57.

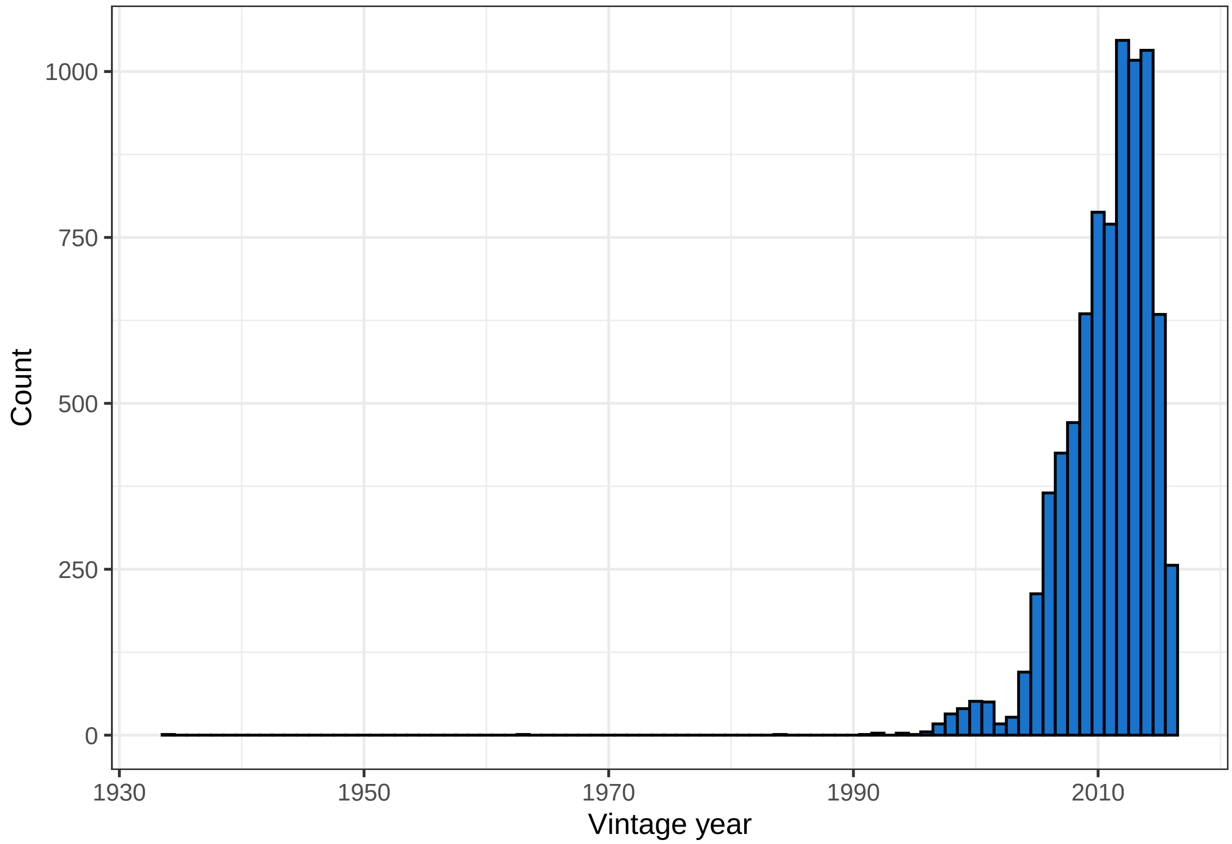
We next examine the relationship between wine price and points in the training set:



We see from the plot above that there is a strong positive trend between wine price and average points, although we see the points for the wine priced at the maximum $3,300 price is actually below the average of all ratings (88.5):

price	Average points	count
3300	88	1

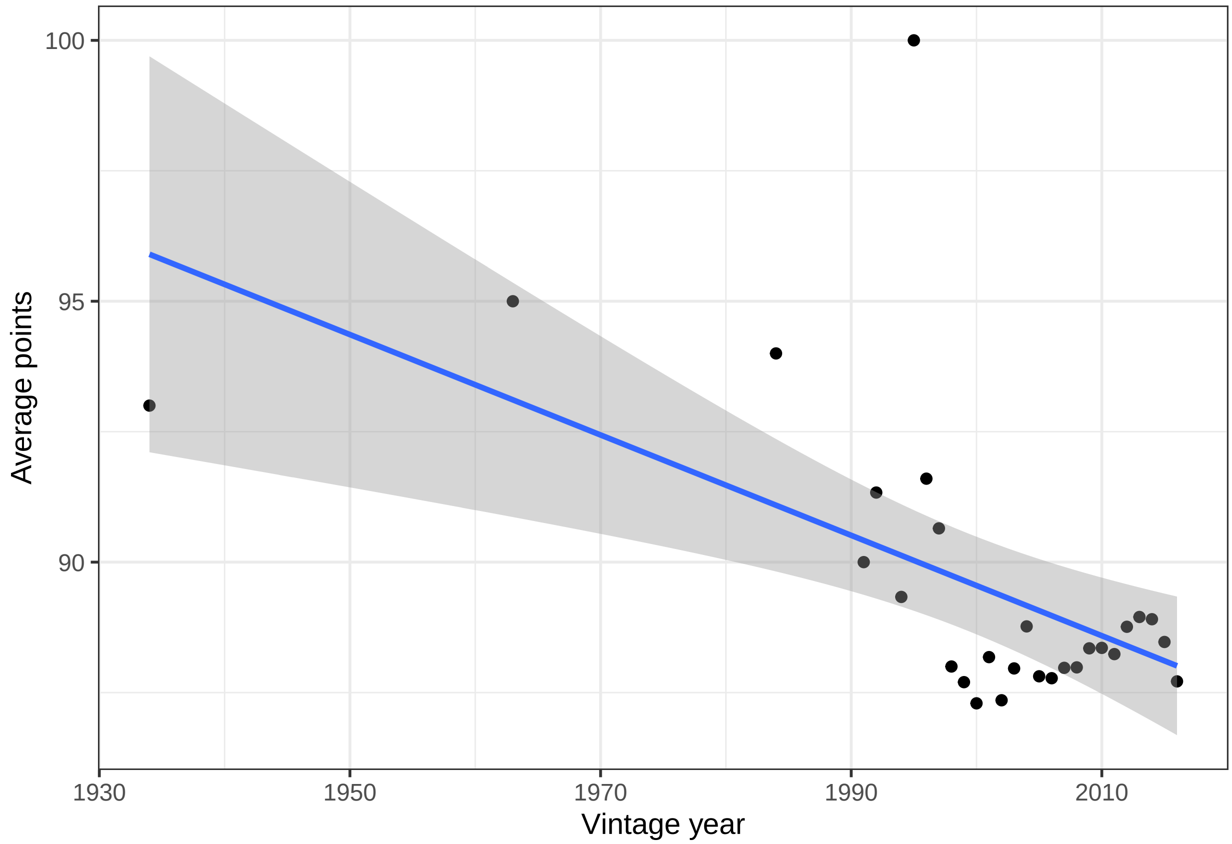
Next we look at the distribution of wine vintage years in our training set:



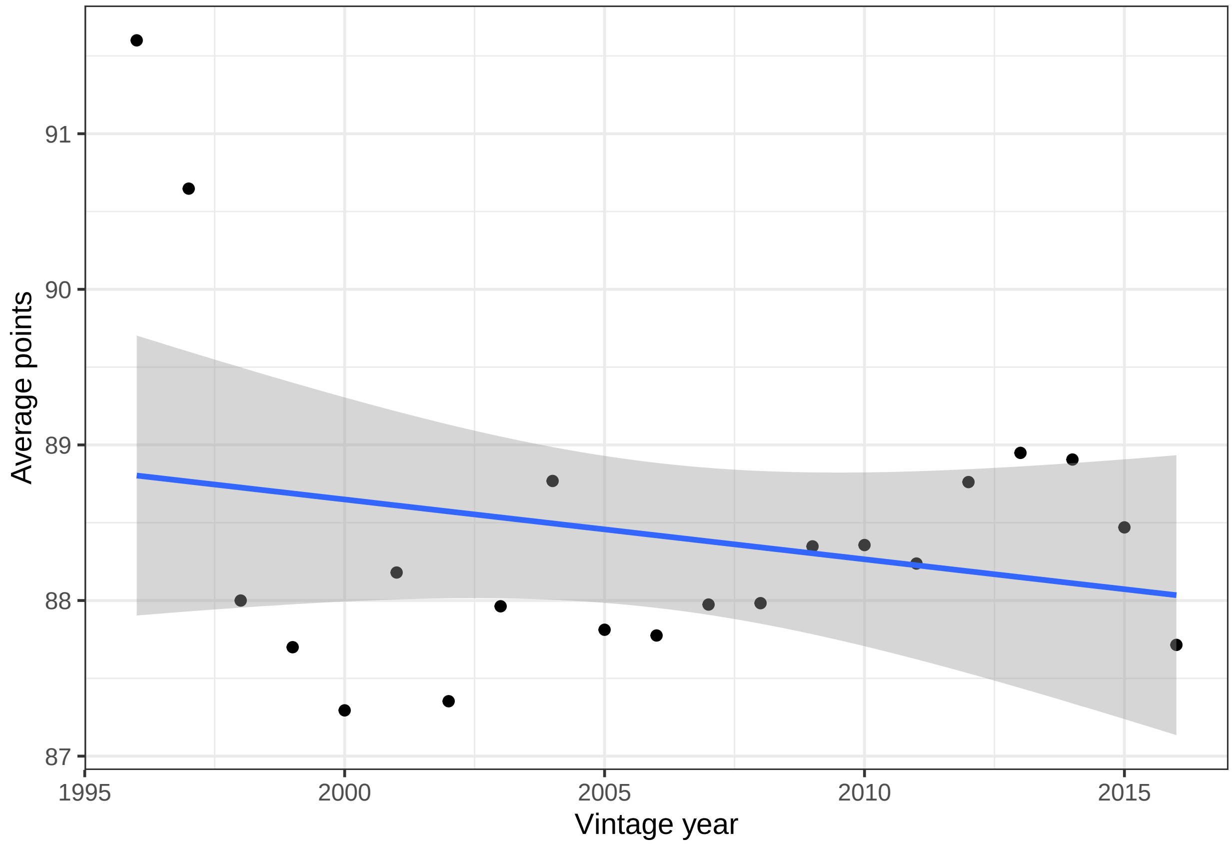
Mean year	SD year	Min year	Max year
2010.8	3.6713	1934	2016

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:

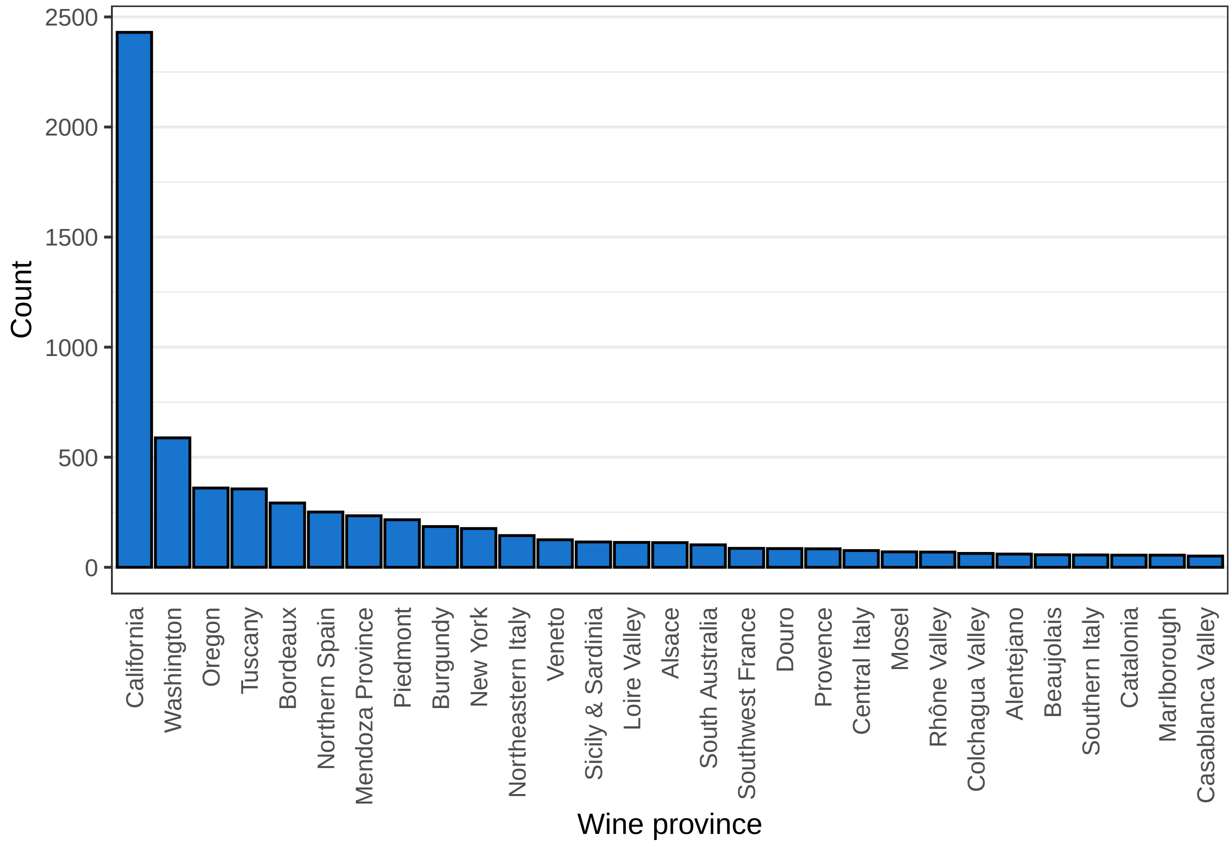


We see there is a strong negative trend for wine points versus vintage year; newer wines appear to be rated lower than older ones. However, since there are far fewer data points available for older wines we re-plot the data using only vintages after 1995:

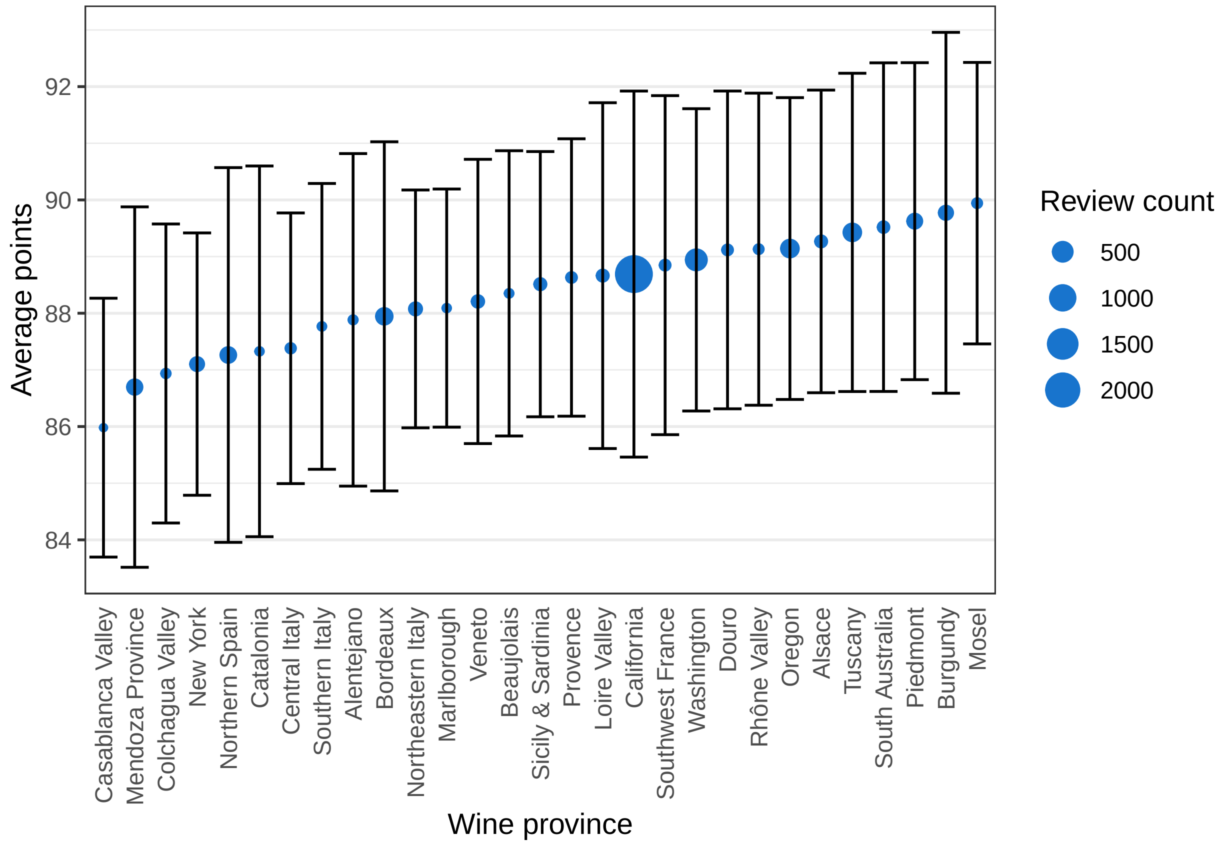


Clearly the negative trend for wine points versus vintage year is much less noticeable when focusing only on more recent years. Nevertheless, vintage year looks to provide useful information when it comes to predicting ratings.

Now, we examine the link between the province of wine origin and points. First we plot the distribution among provinces with over 50 reviews in our training set below:



We can see that almost one third of the wines in our training set originate from California, with mainly other US, Italian, and French provinces featuring prominently.



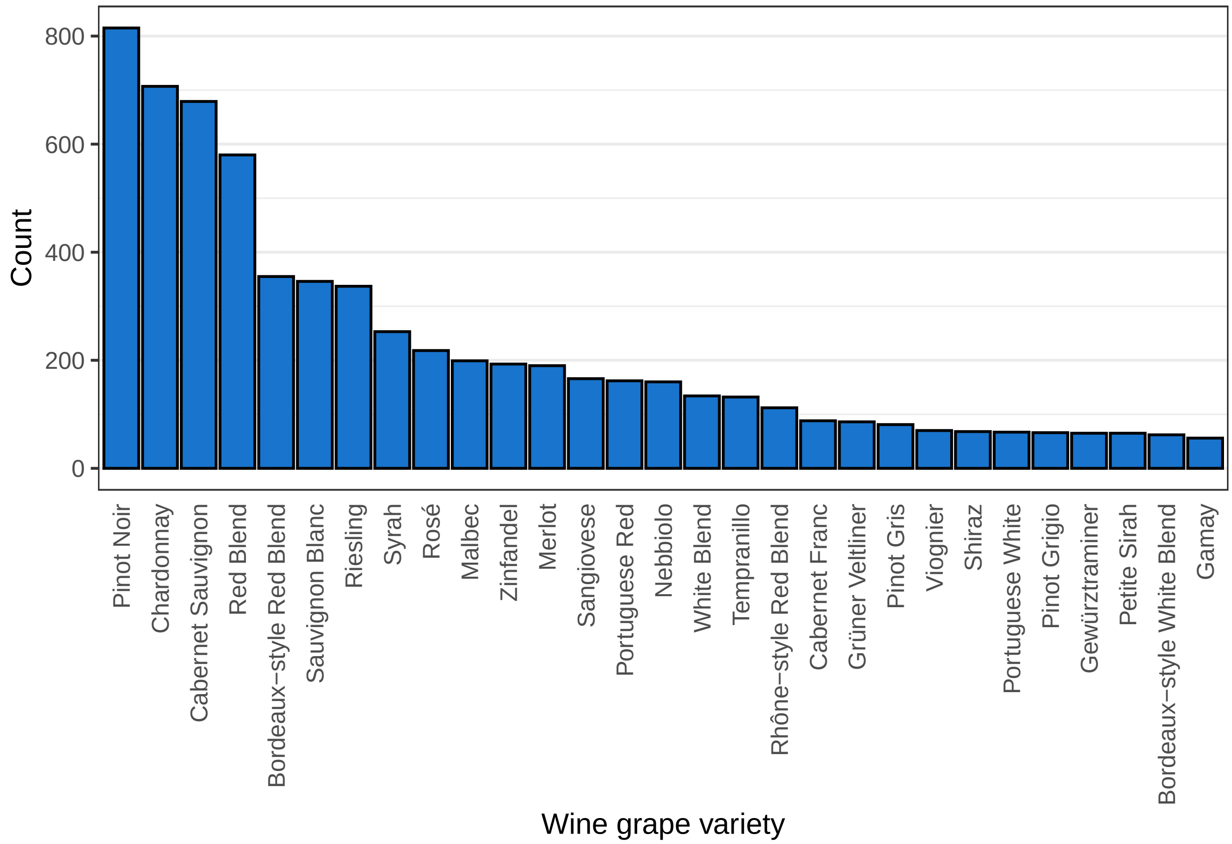
Looking at an ordered plot of average points with error bars at one standard deviation for provinces with more than 50 reviews we see there is some disparity between provinces, although provinces with the highest number of reviews tend to be clustered together around the overall training set average 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:

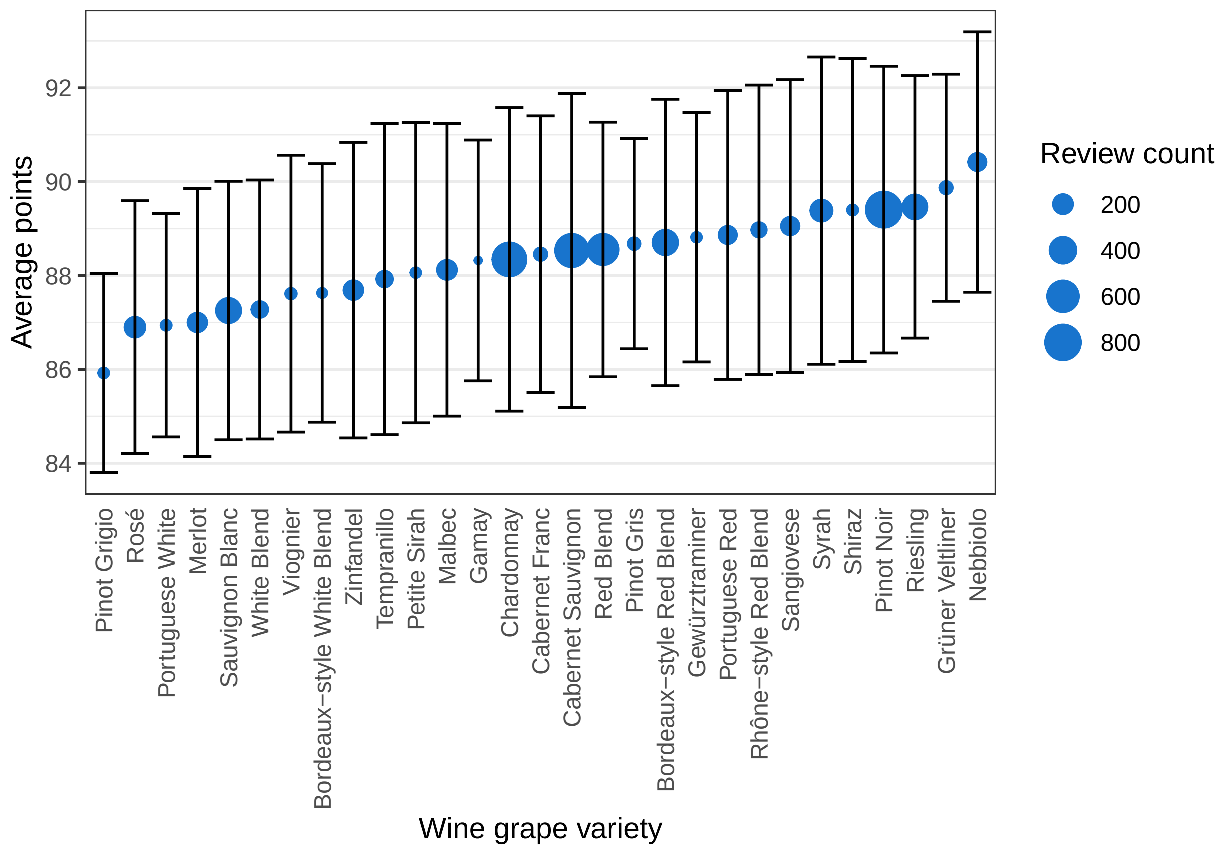
province	Mean_points	SD_points	Count
Eisenberg	94	1	3
Beira Atlantico	93	0	2
Colares	93	NaN	1
England	93	NaN	1
Mittelrhein	93	NaN	1

province	Mean_points	SD_points	Count
Missouri	80.0	NaN	1
Vale dos Vinhedos	80.0	NaN	1
Juanico	82.0	NaN	1
Ica	82.5	2.3805	4
Lolol Valley	83.0	NaN	1
North Carolina	83.0	1.7321	3
Tasmania	83.0	NaN	1
Ticino	83.0	NaN	1
Ukraine	83.0	NaN	1

Overall, wine province looks to be a useful variable for use 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:





We see that the most popular grape varieties are Pinot Noir, Chardonnay, and Cabernet Sauvignon, along with Red Blend, which have average ratings between 88 and 89.5 - around the overall average. Standard deviation of points for each variety is again approximately 3 points. Below, we again see that the outliers at the high and low end of average points have only one or two reviews each:

variety	Mean_points	SD_points	Count
Prugnolo Gentile	95	7.0711	2
Nosiola	94	NaN	1
Baga	93	NaN	1
Jaen	93	NaN	1
Ramisco	93	NaN	1
Roviello	93	NaN	1

variety	Mean_points	SD_points	Count
Chambourcin	80	NaN	1
Shiraz-Tempranillo	80	NaN	1
Malvar	81	NaN	1
Cabernet Merlot	82	NaN	1
Garnacha Blanca	82	NaN	1
Tempranillo-Syrah	82	NaN	1

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

Finally, we can evaluate the usefulness of wine descriptions, the text of the reviews themselves, in predicting wine ratings. The descriptions are in the form of free-form text, so we can use the `tidytext` package to tokenize the words of each review, remove stop words, and then perform a sentiment analysis on the remaining words.

We tokenize the words in the descriptions of a subset of our training set containing 10 entries for experimentation. We remove stop words and numbers and then run sentiment analysis on the remaining words using the 'Afinn' lexicon, which assess the positivity or negativity of words on a scale of -5 to 5:

```
afinn <- get_sentiments("afinn")                           # Load the afinn sentiments

set.seed(1, sample.kind = "Rounding")                      # Set seed for replicability
sample_set <- train_set %>%
  mutate(description = as.character(description)) %>% # Convert descriptions from factors
  sample_n(10) %>%                                         # Sample 10 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(afinn, by="word")                             # Join the afinn sentiments by word
sample_set %>% knitr::kable()
```

points	price	province	variety	vintage	category	word	value
89	28	Mosel	Riesling	2013	D	sweet	2
97	69	Loire Valley	Chenin Blanc	2010	B	wonderful	4
97	69	Loire Valley	Chenin Blanc	2010	B	fine	2
97	69	Loire Valley	Chenin Blanc	2010	B	lovely	3
97	69	Loire Valley	Chenin Blanc	2010	B	bright	1
97	69	Loire Valley	Chenin Blanc	2010	B	sweet	2
97	69	Loire Valley	Chenin Blanc	2010	B	intense	1
90	28	California	Chardonnay	2013	C	rich	2
90	28	California	Chardonnay	2013	C	bright	1
86	20	Alsace	Pinot Gris	2016	E	cut	-1
86	20	Alsace	Pinot Gris	2016	E	bitter	-2
86	20	Alsace	Pinot Gris	2016	E	helps	2
88	26	Alsace	Gewürztraminer	2013	D	friendly	2
88	26	Alsace	Gewürztraminer	2013	D	cuts	-1
80	25	California	Viognier	2006	F	sweet	2
80	25	California	Viognier	2006	F	helps	2
91	70	Tuscany	Red Blend	2010	C	crushed	-2
91	70	Tuscany	Red Blend	2010	C	supporting	1
91	70	Tuscany	Red Blend	2010	C	bright	1
87	22	New York	Merlot	2011	D	elegantly	2

We can see from the above that only a very small subset of words in the descriptions have a sentiment value attached to them, and if we group the sentiments by review, we see that only eight out of the ten reviews have any sentiment value attached at all:

```
# Group the word sentiment scores by review and calculate the average sentiment
sample_set %>% unite(review, province, vintage, variety, sep = " ") %>%
  group_by(review) %>%
  summarise(points = points[1], Avg_sentiment = mean(value)) %>%
  knitr::kable()
```

review	points	Avg_sentiment
Alsace 2013 Gewürztraminer	88	0.50000
Alsace 2016 Pinot Gris	86	-0.33333
California 2006 Viognier	80	2.00000
California 2013 Chardonnay	90	1.50000
Loire Valley 2010 Chenin Blanc	97	2.16667
Mosel 2013 Riesling	89	2.00000
New York 2011 Merlot	87	2.00000
Tuscany 2010 Red Blend	91	0.00000

We look also at a sentiment analysis using 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 <- train_set %>%
  mutate(description = as.character(description)) %>% # Convert descriptions from factors
  sample_n(10) %>%                                      # Sample 10 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 %>% knitr::kable()
```

points	price	province	variety	vintage	category	word	sentiment
89	28	Mosel	Riesling	2013	D	wild	negative
89	28	Mosel	Riesling	2013	D	thirst	negative
89	28	Mosel	Riesling	2013	D	peach	positive
89	28	Mosel	Riesling	2013	D	lemon	negative
89	28	Mosel	Riesling	2013	D	sweet	positive
97	69	Loire Valley	Chenin Blanc	2010	B	wonderful	positive
97	69	Loire Valley	Chenin Blanc	2010	B	fine	positive
97	69	Loire Valley	Chenin Blanc	2010	B	lovely	positive
97	69	Loire Valley	Chenin Blanc	2010	B	bright	positive
97	69	Loire Valley	Chenin Blanc	2010	B	richly	positive
97	69	Loire Valley	Chenin Blanc	2010	B	sweet	positive
97	69	Loire Valley	Chenin Blanc	2010	B	intense	negative
97	69	Loire Valley	Chenin Blanc	2010	B	sweetness	positive
90	28	California	Chardonnay	2013	C	rich	positive
90	28	California	Chardonnay	2013	C	bright	positive
90	28	California	Chardonnay	2013	C	lemon	negative
90	28	California	Chardonnay	2013	C	blossom	positive
90	28	California	Chardonnay	2013	C	proving	positive
90	28	California	Chardonnay	2013	C	balanced	positive
90	28	California	Chardonnay	2013	C	ample	positive
86	20	Alsace	Pinot Gris	2016	E	bitter	negative
88	26	Alsace	Gewürztraminer	2013	D	richness	positive
88	26	Alsace	Gewürztraminer	2013	D	friendly	positive
86	80	California	Red Blend	2010	E	fragrant	positive
80	25	California	Viognier	2006	F	sweet	positive

points	price	province	variety	vintage	category	word	sentiment
80	25	California	Viognier	2006	F	crisp	positive
80	25	California	Viognier	2006	F	sweetness	positive
91	70	Tuscany	Red Blend	2010	C	lead	positive
91	70	Tuscany	Red Blend	2010	C	crushed	negative
91	70	Tuscany	Red Blend	2010	C	supporting	positive
91	70	Tuscany	Red Blend	2010	C	balanced	positive
91	70	Tuscany	Red Blend	2010	C	bright	positive
87	22	New York	Merlot	2011	D	elegantly	positive
87	22	New York	Merlot	2011	D	appeal	positive
84	19	Rhône Valley	Rhône-style Red Blend	2012	E	dominated	positive
84	19	Rhône Valley	Rhône-style Red Blend	2012	E	decent	positive

Slightly more words are picked up in the Bing lexicon than Afinn, however it is still a small subset of all words. If we group sentiments by review and calculate the percentage of positive words for each entry, we see that all ten 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()) %>%
  knitr::kable()
```

review	points	Positive_percentage
Alsace 2013 Gewürztraminer	88	100.000
Alsace 2016 Pinot Gris	86	0.000
California 2006 Viognier	80	100.000
California 2010 Red Blend	86	100.000
California 2013 Chardonnay	90	85.714
Loire Valley 2010 Chenin Blanc	97	87.500
Mosel 2013 Riesling	89	40.000
New York 2011 Merlot	87	100.000
Rhône Valley 2012 Rhône-style Red Blend	84	100.000
Tuscany 2010 Red Blend	91	80.000

The sentiments here clearly have little relation to the points scores, however, with five of the entries having the maximum 100% positive sentiment but points scores in the 80's as low as the minimum score of 80, and the wines scoring above 90 having lower proportions of positive word sentiments.

Overall, it appears that the wine descriptions may provide limited use in our machine learning algorithms for predicting wine quality compared to our other variables. For the purposes of this study, we will leave out further analysis of wine descriptions.

In summary then, in our machine learning algorithms we will use the **price**, **vintage**, **province**, and **variety** variables to predict wine points categories.

We now remove the `description` and `points` variables from our training and testing sets, as well as remove remaining objects from the text sentiment analysis that are no longer required.

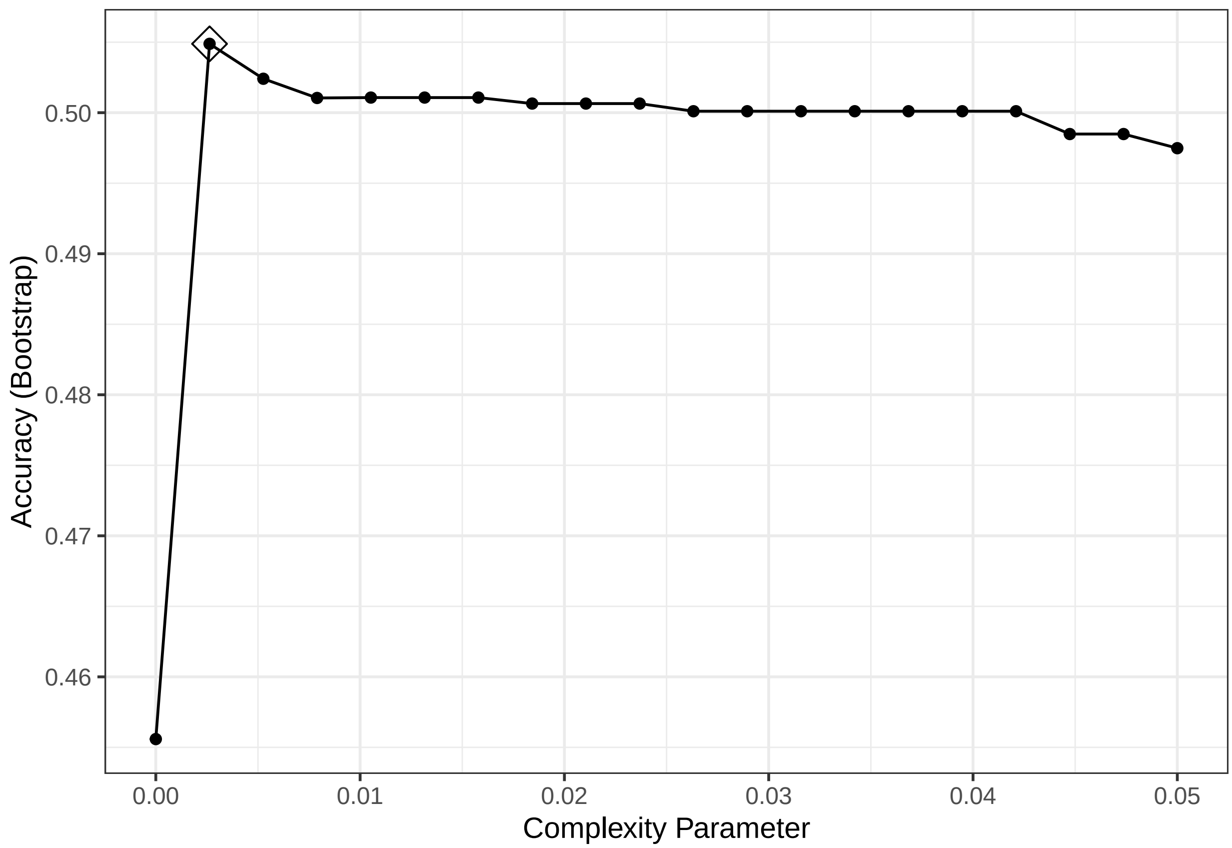
```r
# Remove unrequired variables from training and testing sets and unused objects
train_set <- train_set %>% select(-description, -points)
test_set <- test_set %>% select(-description, -points)
rm(afinn, bing, sample_set, sample_set_bing)
```

2.2 ALGORITHM METHODS

CLASSIFICATION TREE

We will first look at a Classification Tree model. We run the `rpart` algorithm using the `train()` function in the `caret` package, which by default performs a 25-fold cross validation. We set the algorithm to use the tuning parameter, complexity parameter, over a range of 0 to 0.05:

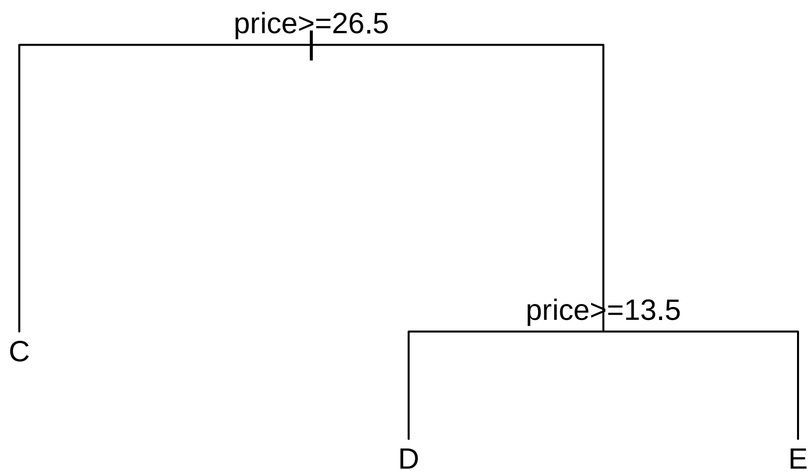
```r
# Train a classification tree model on train_set
train_rpart <- train(category ~ .,
                     method = "rpart",
                     data = train_set,
                     tuneGrid = data.frame(cp = seq(0.0, 0.05, len = 20)))
```




```r
train_rpart$results %>% filter(Accuracy == max(Accuracy))
```

```
##          cp Accuracy   Kappa AccuracySD  KappaSD
## 1 0.0026316  0.50488 0.26961  0.0088885 0.014721
```

We see that utilizing the optimal complexity parameter for our model, the accuracy is only 0.50488.



Examining the optimal tree structure, we find that only points categories C, D, and E are being assigned and `price` is the only variable being used to determine the category.

```
## rpart variable importance
##
##   only 20 most important variables shown (out of 542)
##
##                              Overall
## price                         100.00
## vintage                         7.59
## provinceCalifornia              4.58
## varietyPinot Noir               3.47
## varietyPinot Grigio             2.36
## varietyRiesling                 1.52
## provinceCentral Spain           1.49
## provinceKremstal                1.36
## `varietyPinot Auxerrois`        0.00
## provinceMartinborough           0.00
## `provinceCentral Otago`         0.00
## `varietyPrugnolo Gentile`       0.00
## varietySciaccerellu             0.00
## provinceNiederösterreich        0.00
## `varietyBlauer Portugieser`     0.00
## provincePort                    0.00
## `varietyG-S-M`                  0.00
```

```
## provincePeloponnese           0.00
## `varietyMonastrell-Syrah`     0.00
## provinceHungary               0.00
```

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

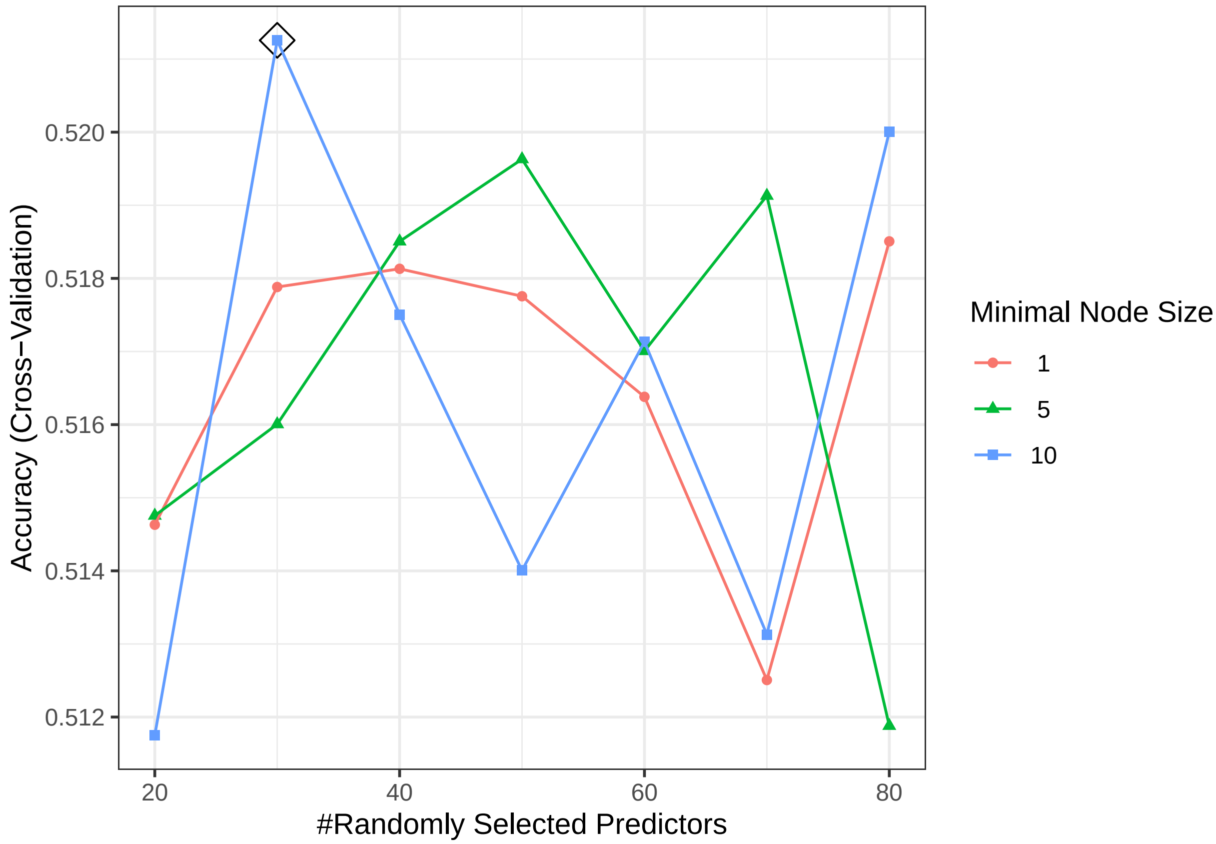
Model	Accuracy
Classification Tree	0.50488

RANDOM FOREST

Next we look at a Random Forest model using the `Rborist` package. 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 1, 5, and 10 under the `minNode` parameter:

```
# Set to 3-fold cross validation and our tuning parameter values to test
control <- trainControl(method="cv", number = 3)
grid <- expand.grid(minNode = c(1,5, 10) , predFixed = seq(20,80,10))

# Train random forest model on train_set with 50 trees sampling 500 rows each
train_rf <- train(category ~ .,
                     method = "Rborist",
                     data = train_set,
                     nTree = 50,
                     tuneGrid = grid,
                     trControl = control,
                     nSamp = 500)
```




```
## Random Forest
##
## 7998 samples
##    4 predictor
##    6 classes: 'A', 'B', 'C', 'D', 'E', 'F'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 5333, 5331, 5332
## Resampling results across tuning parameters:
##
##   minNode  predFixed  Accuracy  Kappa
##    1       20         0.51463   0.28747
##    1       30         0.51788   0.29376
##    1       40         0.51813   0.29424
##    1       50         0.51776   0.29349
##    1       60         0.51638   0.29192
##    1       70         0.51251   0.28578
##    1       80         0.51851   0.29461
##    5       20         0.51475   0.28654
##    5       30         0.51601   0.29076
##    5       40         0.51851   0.29329
##    5       50         0.51963   0.29651
##    5       60         0.51701   0.29183
##    5       70         0.51913   0.29468
##    5       80         0.51188   0.28547
```

```
##   10       20         0.51175   0.28365
##   10       30         0.52126   0.29789
##   10       40         0.51751   0.29387
##   10       50         0.51401   0.28860
##   10       60         0.51713   0.29202
##   10       70         0.51313   0.28784
##   10       80         0.52001   0.29670
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were predFixed = 30 and minNode = 10.
```

We see that our optimal model is using 30 predictors and a minimum node size of 10, resulting in an accuracy of 0.52126.

```
## Rborist variable importance
##
##   only 20 most important variables shown (out of 539)
##
##                                 Overall
## price                            100.00
## vintage                           33.57
## provinceBordeaux                   6.17
## provinceCalifornia                 5.87
## varietyChardonnay                  5.84
## varietyPinot Noir                  5.60
## varietyRosé                        5.53
## varietyRed Blend                   5.38
## varietySangiovese                  4.32
## varietyRiesling                    4.22
## varietyCabernet Sauvignon          3.89
## provinceBurgundy                   3.87
## varietyZinfandel                   3.73
## varietyRhône-style Red Blend       3.53
## provinceNorthern Spain             3.34
## provinceMendoza Province           3.25
## varietyPetite Sirah                3.20
## varietyBordeaux-style Red Blend    3.14
## varietySyrah                       3.14
## provinceWashington                 3.04
```

Looking at the variable importance for the random forest model, we again see that price is the most important variable, followed by vintage.

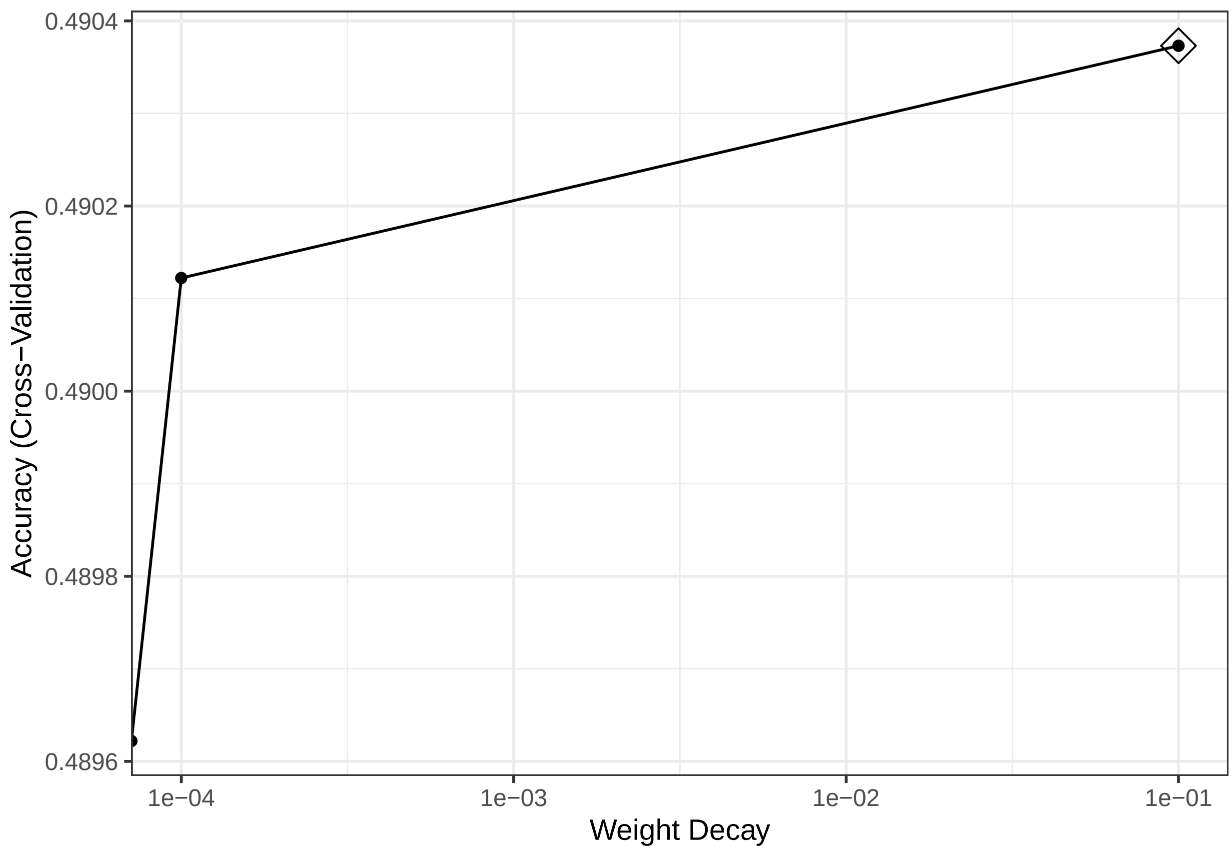
Model	Accuracy
Classification Tree	0.50488
Random Forest	0.52126

MULTINOMIAL LOGISTIC REGRESSION

Next, we look at a Multinomial Logistic Regression model `multinom` from the `nnet` package. This model is able to fit a logistic regression for multiple category outcomes, as is required in this case. We limit to a

5-fold cross validation to save computing time:

```
# Set to 5-fold cross validation
control <- trainControl(method = "cv", number = 5)

# Train multinomial logistic regression model on train_set
train_mn <- train(category ~ .,
                  data = train_set,
                  method = "multinom",
                  trControl = control,
                  MaxNWts = 3400)
```




```
## Penalized Multinomial Regression
##
## 7998 samples
##    4 predictor
##    6 classes: 'A', 'B', 'C', 'D', 'E', 'F'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 6397, 6398, 6399, 6398, 6400
## Resampling results across tuning parameters:
##
##   decay  Accuracy  Kappa
##   0e+00  0.48962   0.25675
```

```
##   1e-04  0.49012   0.25742
##   1e-01  0.49037   0.25655
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was decay = 0.1.
```

We see using the optimal tuning parameter that we have a model accuracy of only 0.49037.

Model	Accuracy
Classification Tree	0.50488
Random Forest	0.52126
Multinomial Logistic Regression	0.49037

QDA MODEL

Next we look at a 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:

```
# Train QDA model on train_set
train_qda <- train(category ~ price + vintage,
                  data = train_set,
                  method = "qda")
```

```
## Quadratic Discriminant Analysis
##
## 7998 samples
##    2 predictor
##    6 classes: 'A', 'B', 'C', 'D', 'E', 'F'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 7998, 7998, 7998, 7998, 7998, 7998, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.42044   0.18487
```

We see that the accuracy is only 0.42044, the lowest yet.

Model	Accuracy
Classification Tree	0.50488
Random Forest	0.52126
Multinomial Logistic Regression	0.49037
QDA	0.42044

K-NEAREST NEIGHBORS

Finally we will look at training a k-Nearest Neighbors (kNN) algorithm on our training set.

Since categorical variables cannot be used with the kNN algorithm, 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 <- train_set$category

# Create a dummy variable matrix of predictors for factor variable levels
dummyvars <- dummyVars( ~ price + vintage + variety + province, data = train_set)
train_dummyvars <- predict(dummyvars, newdata = train_set)
dim(train_dummyvars)
```

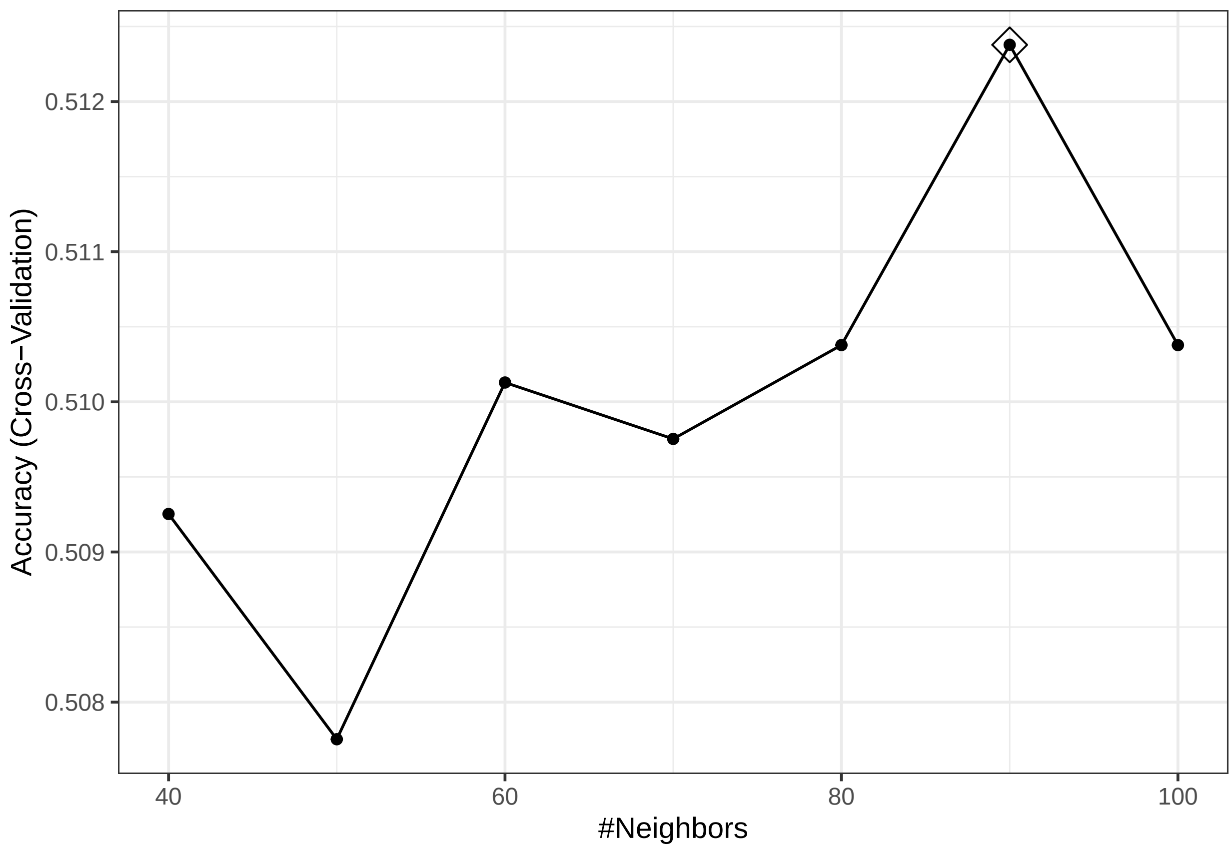
```
## [1] 7998  541
```

We see that we have created a matrix of predictors with 541 variables. We see this is correct since we have 217 distinct province factor levels and 322 distinct variety factor levels in our training set, plus the price and vintage variables:

	Count
train_set distinct provinces	217
train_set distinct varieties	322

We run a kNN algorithm and again limit to 5-fold cross validation to save on computation time. We run the algorithm with values between 40 and 100 for our tuning parameter, the number of neighbors k:

```
# Set to 5-fold cross validation
control <- trainControl(method = "cv", number = 5)

# Train kNN model using the dummy variables and outcomes for train_set
train_knn <- train(train_dummyvars,
                   y_train,
                   method = "knn",
                   tuneGrid = data.frame(k = seq(40, 100, 10)),
                   trControl = control)
```




```
## k-Nearest Neighbors
##
## 7998 samples
##  541 predictor
##    6 classes: 'A', 'B', 'C', 'D', 'E', 'F'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 6398, 6399, 6399, 6398, 6398
## Resampling results across tuning parameters:
##
##   k    Accuracy  Kappa
##    40  0.50925   0.27967
##    50  0.50775   0.27673
##    60  0.51013   0.27920
##    70  0.50975   0.27822
##    80  0.51038   0.27861
##    90  0.51238   0.28098
##   100  0.51038   0.27756
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 90.
```

We see that the optimal value for the number of neighbors k is 90, with an accuracy level of 0.51238 - a close second best.

Model	Accuracy
Classification Tree	0.50488
Random Forest	0.52126
Multinomial Logistic Regression	0.49037
QDA	0.42044
kNN	0.51238

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 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
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 = test_set)
test_dummyvars <- predict(dummyvars, newdata = test_set)

# Create our model predictions for the testing set using the final model
y_hat <- as.factor(predict(final_model, test_dummyvars)$yPred)
cm <- confusionMatrix(y_hat, test_set$category)
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E   F
##          A   0   0   0   0   0   0
##          B   0  13  11   3   0   0
##          C   1  69 466 237  71   6
##          D   0   2 185 362 232  19
##          E   0   0  13  84 171  16
##          F   0   0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.516
##                  95% CI : (0.494, 0.538)
```

```
##     No Information Rate : 0.35
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.28
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E Class: F
## Sensitivity           0.00000  0.15476    0.690    0.528   0.3608   0.0000
## Specificity           1.00000  0.99254    0.701    0.656   0.9240   1.0000
## Pos Pred Value            NaN  0.48148    0.548    0.453   0.6021      NaN
## Neg Pred Value        0.99949  0.96329    0.812    0.721   0.8193   0.9791
## Prevalence            0.00051  0.04284    0.344    0.350   0.2417   0.0209
## Detection Rate        0.00000  0.00663    0.238    0.185   0.0872   0.0000
## Detection Prevalence  0.00000  0.01377    0.433    0.408   0.1448   0.0000
## Balanced Accuracy     0.50000  0.57365    0.696    0.592   0.6424   0.5000
```

Looking at the confusion matrix above, we see that the accuracy of our final model on the test set is 0.516. This is in line with our earlier expected accuracy level from the training process.

We see that our sensitivity for the different categories ranges from as low as 0 for classes A and F with very few entries, up to around 0.5-0.7 for classes C and D - the most populated categories. Specificity is better across all classes, although dropping below 0.7 for the most populated class D.

4. 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 10,000 reviews in order to have a more practical sample size for model fitting purposes and built training and testing sets in a 80:20 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 a classification tree model, random forest, multinomial logistic regression, QDA model, and a k-Nearest Neighbors model, 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 0.516. 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 kNN 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.