Naive Bayes

Hendrik Orem, Ph.D., with thanks to Jameson Watts

01/26/2023

Agenda

1. Review of Homework 3
2. The Naive Bayes algorithm
3. Dinner break
4. Tidy text and bag of words
5. Group work
6. Vocabulary

The Naive Bayes Algorithm

Bayes' Theorem

$$P(L \mid \text{features}) = \frac{P(\text{features} \mid L)P(L)}{P(\text{features})}$$

More generally…

$$P(\text{Thing1} \mid \text{Thing2}) = \frac{P(\text{Thing2} \mid \text{Thing1})\text{P}(\text{Thing1})}{P(\text{Thing2})}$$

Bayes' Theorem Example

Suppose half of all emails are spam, and you've just purchased some software (hurray) that filters spam emails, claiming to detect 99% of spam and that the probability of a false positive (marking non-spam as spam) is 5%.

Now suppose an incoming email is marked as spam. What is the probability that it's a non-spam email?

Thing 1 = email is non-spam email

Thing 2 = emaill is marked as spam

P(thing2 | thing1) =

P(thing 1) =

P(thing 2) =

Bayes' Theorem Example Solution

Solution:

P(thing2 | thing1) = 5%

P(thing1) = 50%

P(thing 2) = 99% * 50% + 5% * 50%

$$0.05 * 0.5/(0.99 * 0.5 + 0.05 * 0.5) = 4.81\%$$

Bayes' Theorem Exercises!

1. You have three cards: one is red on both sides, one is black on both sides, and one has one red side and one black side. You pick a card at random, and put it on the table on a random side, and the color showing is red. What is the probability that the other side is black?
2. Let's imagine half of all rainy days start off cloudy in the morning. However, we live in a cloudy place, and about 40% of days start off cloudy, and you know that 90% of days this time of year do not have rain. What are the odds it will rain today?

Solutions

1. Solution:

Thing 1 = card is red-black

thing 2 = side up is red

P(side up is red | card is red-black) = 1/2

P(thing 1) = 1/3

P(thing 2) = 100% * 1/3 + 50% * 1/3 + 0% * 1/3

so P(card is red-black | side up is red) = 1/3 * 1/2 / (1/3 + 1/6) = 1/3

2. Solution:

thing 1 = rain during the day

thing 2 = cloudy in the morning

P (thing 2 | thing 1) = 50%

P(thing 1) = 10%

P(thing 2) = 40%

so we get P(thing 1 | thing 2) = 0.1*0.5 / 0.4 = 0.125

Algorithm

$$P(L \mid \text{features}) = \frac{P(\text{features} \mid L)P(L)}{P(\text{features})}$$

If we only care about choosing between two labels L1 and L2, then we only need the ratio:

$$\frac{P(L_1 \mid \text{features})}{P(L_2 \mid \text{features})} = \frac{P(\text{features} \mid L_1)}{P(\text{features} \mid L_2)} \frac{P(L_1)}{P(L_2)}$$

But how on earth can we get P(features | L)? Well, we have to make an assumption. "Naive" in Naive Bayes means we keep it simple.

Really we would need P(Cherry, Fruit, Bordeaux | Chardonnay), "Naive" assumption is independence so the algorithm calculates P(Cherry | Chardonnay) * P(Fruit | Chardonnay) * P(Bordeaux | Chardonnay).

Setup

```
knitr::opts_chunk$set(echo = TRUE, message = FALSE, warning = FALSE)
library(tidyverse)
library(caret)
library(naivebayes)
library(fastDummies)
#source('theme.R')

wine = read_rds("../resources/pinot.rds")
names(wine)[names(wine) == 'id'] = 'ID'
```

Some basic features

```
wino <- wine %>%
  mutate(year_f = as.factor(year)) %>%
  mutate(cherry = str_detect(description,"cherry")) %>%
  mutate(chocolate = str_detect(description,"chocolate")) %>%
  mutate(earth = str_detect(description,"earth")) %>%
  select(-description, year)

glimpse(wino)
```

```
## Rows: 8,380
## Columns: 9
## $ ID        <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 1~
## $ province  <chr> "Oregon", "Oregon", "California", "Oregon", "Oregon", "Orego~
## $ price     <dbl> 65, 20, 69, 50, 22, 25, 64, 55, 44, 38, 28, 45, 22, 55, 40, ~
## $ points    <dbl> 87, 87, 87, 86, 86, 86, 91, 91, 91, 91, 85, 85, 85, 89, 89, ~
## $ year      <dbl> 2012, 2013, 2011, 2010, 2009, 2015, 2013, 2012, 2014, 2014, ~
## $ year_f    <fct> 2012, 2013, 2011, 2010, 2009, 2015, 2013, 2012, 2014, 2014, ~
## $ cherry    <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, TRUE, FALSE~
## $ chocolate <lgl> FALSE, TRUE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE~
## $ earth     <lgl> TRUE, FALSE, TRUE, FALSE, TRUE, FALSE, FALSE, FALSE, FALSE, ~
```

A basic model

```
set.seed(504)
wine_index <- createDataPartition(wino$province, p = 0.80, list = FALSE)
train <- wino[ wine_index, ]
test <- wino[-wine_index, ]

control <- trainControl(method = "cv")

fit <- train(province ~ .,
             data = train,
             method = "naive_bayes",
             metric = "Kappa",
             trControl = control)
fit
```

```
## Naive Bayes
##
## 6707 samples
```

```
##    8 predictor
##    6 classes: 'Burgundy', 'California', 'Casablanca_Valley', 'Marlborough', 'New_York', 'Oregon'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 6035, 6038, 6037, 6038, 6036, 6036, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.4635340  0.18240715
##    TRUE      0.4122263  0.09141955
##
## Tuning parameter 'laplace' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Kappa was used to select the optimal model using the largest value.
## The final values used for the model were laplace = 0, usekernel = FALSE
##  and adjust = 1.
```

What's going on here?

Maybe bin the data?

```
wino <- wino %>%
  select(-starts_with("year_")) %>%
  mutate(points_f = case_when(
    points < 90 ~ "low",
    points >= 90 & points < 96 ~ "med",
    points >= 96 ~ "high"
  )
         )  %>%
  mutate(price_f = case_when(
    price < 16 ~ "low",
    price >= 16 & price < 41 ~ "med",
    price >= 41 ~ "high"
  )
         )  %>%
  mutate(year_f = case_when(
    year < 2005 ~ "old",
    year >= 2005 & year < 2011 ~ "recent",
    year >= 2011 ~ "current"
  )
         ) %>%
  select(-price,-points,-year)

  head(wino)
```

ID	province	cherry	chocolate	earth	points_f	price_f	year_f
1	Oregon	FALSE	FALSE	TRUE	low	high	current
2	Oregon	FALSE	TRUE	FALSE	low	med	current
3	California	FALSE	FALSE	TRUE	low	high	current
4	Oregon	FALSE	FALSE	FALSE	low	high	recent
5	Oregon	FALSE	FALSE	TRUE	low	med	recent

ID	province	cherry	chocolate	earth	points_f	price_f	year_f
6	Oregon	FALSE	FALSE	FALSE	low	med	current

Binned model

```
set.seed(504)
train <- wino[ wine_index, ]
test <- wino[-wine_index, ]

fit <- train(province ~ .,
             data = train,
             method = "naive_bayes",
             metric = "Kappa",
             trControl = control)
fit
```

```
## Naive Bayes
##
## 6707 samples
##    7 predictor
##    6 classes: 'Burgundy', 'California', 'Casablanca_Valley', 'Marlborough', 'New_York', 'Oregon'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 6037, 6035, 6037, 6038, 6035, 6036, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.4571336  0.2539893
##    TRUE      0.5182601  0.1470779
##
## Tuning parameter 'laplace' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## Kappa was used to select the optimal model using the largest value.
## The final values used for the model were laplace = 0, usekernel = FALSE
##  and adjust = 1.
```

Little better, but let’s look at the confusion matrix to see what might be going on.

Confusion Matrix

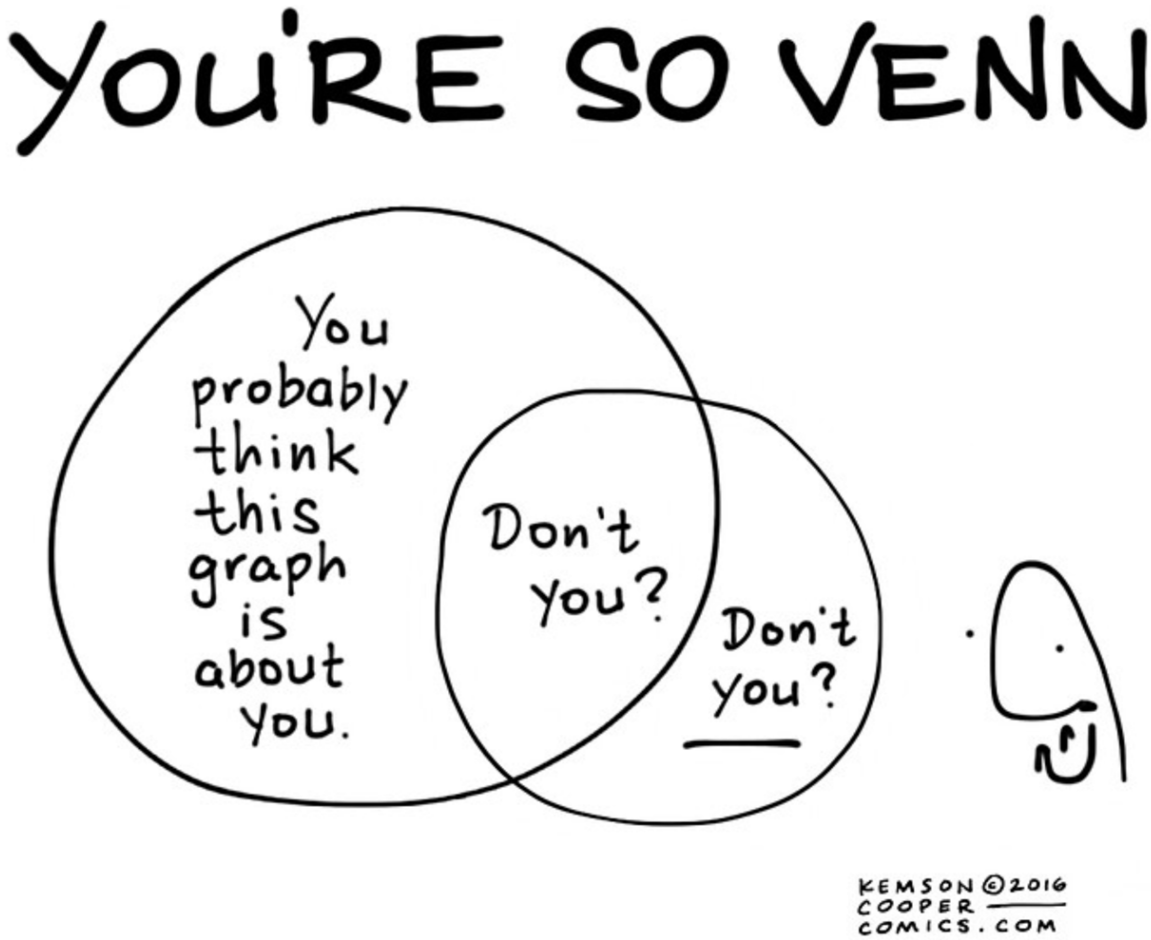
```
confusionMatrix(predict(fit, test),factor(test$province))
```

```
## Confusion Matrix and Statistics
##
##                      Reference
## Prediction            Burgundy California Casablanca_Valley Marlborough New_York
##   Burgundy                 211        311                13          17        5
##   California                17        427                 1          12       11
##   Casablanca_Valley          2         27                 9           6        4
##   Marlborough                1          1                 1           2        1
##   New_York                   2          4                 0           4        5
```

```
##   Oregon                  5        21                 2           4        0
##                  Reference
## Prediction          Oregon
##   Burgundy             249
##   California           162
##   Casablanca_Valley      4
##   Marlborough           16
##   New_York              10
##   Oregon               106
##
## Overall Statistics
##
##                Accuracy : 0.4543
##                  95% CI : (0.4302, 0.4785)
##     No Information Rate : 0.4728
##     P-Value [Acc > NIR] : 0.9386
##
##                   Kappa : 0.2477
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: Burgundy Class: California Class: Casablanca_Valley
## Sensitivity                   0.8866            0.5398                  0.34615
## Specificity                   0.5854            0.7698                  0.97389
## Pos Pred Value                0.2618            0.6778                  0.17308
## Neg Pred Value                0.9689            0.6510                  0.98951
## Prevalence                    0.1423            0.4728                  0.01554
## Detection Rate                0.1261            0.2552                  0.00538
## Detection Prevalence          0.4818            0.3766                  0.03108
## Balanced Accuracy             0.7360            0.6548                  0.66002
##                      Class: Marlborough Class: New_York Class: Oregon
## Sensitivity                    0.044444        0.192308       0.19378
## Specificity                    0.987715        0.987857       0.97158
## Pos Pred Value                 0.090909        0.200000       0.76812
## Neg Pred Value                 0.973955        0.987257       0.71270
## Prevalence                     0.026898        0.015541       0.32696
## Detection Rate                 0.001195        0.002989       0.06336
## Detection Prevalence           0.013150        0.014943       0.08249
## Balanced Accuracy              0.516080        0.590082       0.58268
```

Naive bayes is best when you want to consider a bunch of predictors simultaneously to get a 'holistic' view.

Dinner (and virtual high fives)



Tidytext and frequency distributions

Tidytext

```
library(tidytext)
data(stop_words)
head(stop_words, 25)$word
```

```
##  [1] "a"            "a's"          "able"         "about"        "above"
##  [6] "according"    "accordingly"  "across"       "actually"     "after"
## [11] "afterwards"   "again"        "against"      "ain't"        "all"
## [16] "allow"        "allows"       "almost"       "alone"        "along"
## [21] "already"      "also"         "although"     "always"       "am"
```

Create document term matrix

```
df <- wine %>%
  unnest_tokens(word, description) %>%
  anti_join(stop_words) %>% # get rid of stop words
```

```
  filter(word != "wine") %>%
  filter(word != "pinot") %>%
  count(ID, word) %>%
  group_by(ID) %>%
  mutate(freq = n/sum(n)) %>%
  mutate(exists = (n>0)) %>%
  ungroup %>%
  group_by(word) %>%
  mutate(total = sum(n))

head(df, 10)
```

ID	word	n	freq	exists	total
1	2012	1	0.0588235	TRUE	71
1	bottling	1	0.0588235	TRUE	849
1	characteristics	1	0.0588235	TRUE	61
1	companion	1	0.0588235	TRUE	22
1	country	1	0.0588235	TRUE	11
1	earthy	1	0.0588235	TRUE	804
1	hearty	1	0.0588235	TRUE	99
1	herbal	1	0.0588235	TRUE	438
1	nonetheless	1	0.0588235	TRUE	33
1	pleasantly	1	0.0588235	TRUE	28

Top words in database

```
df %>%
  count(word) %>%
  arrange(desc(n)) %>%
  head(25)
```

word	n
fruit	3724
cherry	3423
flavors	3048
black	2029
palate	2025
red	1976
finish	1973
tannins	1937
acidity	1856
aromas	1431
light	1410
nose	1399
drink	1371
ripe	1341
raspberry	1335
vineyard	1285
cranberry	1185
oak	1140
strawberry	1130

word	n
bodied	1016
spice	1015
dark	1000
plum	973
fruits	945
texture	920

Pivot wide and rejoin with wine

```
wino <- df %>%
  filter(total > 1000) %>%
  pivot_wider(id_cols = ID, names_from = word, values_from = exists, values_fill = list(exists=0)) %>%
  merge(select(wine,ID, province), all.y=TRUE) #%>%
  #drop_na()

#wino <- merge(select(wine,ID, province), wino, by="ID", all.x=TRUE) %>%
#  arrange(ID)
#View(wino)
wino <- replace(wino, is.na(wino), FALSE)

head(wino, 10) %>%
  select(1:5,province)
```

ID	drink	oak	aromas	bodied	province
1	FALSE	FALSE	FALSE	FALSE	Oregon
2	TRUE	TRUE	FALSE	FALSE	Oregon
3	FALSE	TRUE	TRUE	TRUE	California
4	FALSE	FALSE	FALSE	FALSE	Oregon
5	FALSE	FALSE	FALSE	FALSE	Oregon
6	TRUE	FALSE	FALSE	FALSE	Oregon
7	FALSE	FALSE	FALSE	FALSE	California
8	FALSE	FALSE	FALSE	FALSE	California
9	FALSE	TRUE	FALSE	FALSE	California
10	FALSE	FALSE	FALSE	FALSE	Oregon

A new model

```
set.seed(504)
wine_index <- createDataPartition(wino$province, p = 0.80, list = FALSE)
train <- wino[ wine_index, ]
test <- wino[-wine_index, ]

fit <- train(province ~ .,
             data = train,
             method = "naive_bayes",
             tuneGrid = expand.grid(usekernel = c(T,F), laplace = T, adjust = T),
             metric = "Kappa",
             trControl = trainControl(method = "cv"))
fit
```

```
## Naive Bayes
##
## 6707 samples
##   24 predictor
##    6 classes: 'Burgundy', 'California', 'Casablanca_Valley', 'Marlborough', 'New_York', 'Oregon'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 6035, 6038, 6037, 6038, 6036, 6036, ...
## Resampling results across tuning parameters:
##
##   usekernel  Accuracy   Kappa
##   FALSE      0.5485336  0.4068340
##    TRUE      0.5671734  0.3751128
##
## Tuning parameter 'laplace' was held constant at a value of TRUE
##
## Tuning parameter 'adjust' was held constant at a value of TRUE
## Kappa was used to select the optimal model using the largest value.
## The final values used for the model were laplace = TRUE, usekernel = FALSE
##  and adjust = TRUE.
```

…now things are getting better.

Confusion Matrix

```
confusionMatrix(predict(fit, test),factor(test$province))
```

```
## Confusion Matrix and Statistics
##
##                    Reference
## Prediction          Burgundy California Casablanca_Valley Marlborough New_York
##   Burgundy               212         73                 0           5        6
##   California               6        405                 3           5        8
##   Casablanca_Valley        3        126                20           4        6
##   Marlborough              1         57                 1          22        2
##   New_York                 9         34                 2           2        4
##   Oregon                   7         96                 0           7        0
##                    Reference
## Prediction          Oregon
##   Burgundy             163
##   California            40
##   Casablanca_Valley     49
##   Marlborough           47
##   New_York               8
##   Oregon               240
##
## Overall Statistics
##
##                Accuracy : 0.5397
##                  95% CI : (0.5155, 0.5638)
##     No Information Rate : 0.4728
##     P-Value [Acc > NIR] : 2.447e-08
##
```

```
##                   Kappa : 0.3912
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: Burgundy Class: California Class: Casablanca_Valley
## Sensitivity                   0.8908            0.5120                  0.76923
## Specificity                   0.8279            0.9297                  0.88585
## Pos Pred Value                0.4619            0.8672                  0.09615
## Neg Pred Value                0.9786            0.6799                  0.99590
## Prevalence                    0.1423            0.4728                  0.01554
## Detection Rate                0.1267            0.2421                  0.01195
## Detection Prevalence          0.2744            0.2791                  0.12433
## Balanced Accuracy             0.8593            0.7209                  0.82754
##                      Class: Marlborough Class: New_York Class: Oregon
## Sensitivity                     0.48889        0.153846        0.4388
## Specificity                     0.93366        0.966606        0.9023
## Pos Pred Value                  0.16923        0.067797        0.6857
## Neg Pred Value                  0.98509        0.986369        0.7680
## Prevalence                      0.02690        0.015541        0.3270
## Detection Rate                  0.01315        0.002391        0.1435
## Detection Prevalence            0.07770        0.035266        0.2092
## Balanced Accuracy               0.71127        0.560226        0.6705
```

Maybe we can find words associated with our sparse provinces?

```
df %>%
  left_join(select(wine, ID, province), by = "ID") %>%
  count(province, word) %>%
  group_by(province) %>%
  top_n(5,n) %>%
  arrange(province, desc(n))
```

word	province	n
tannins	Burgundy	763
drink	Burgundy	673
acidity	Burgundy	652
red	Burgundy	630
fruits	Burgundy	575
cherry	California	1917
palate	California	1587
black	California	1336
flavors	California	1332
fruit	California	1289
flavors	Casablanca_Valley	114
aromas	Casablanca_Valley	101
finish	Casablanca_Valley	93
plum	Casablanca_Valley	69
palate	Casablanca_Valley	65
drink	Marlborough	140
cherry	Marlborough	124
fruit	Marlborough	119

word	province	n
finish	Marlborough	107
noir	Marlborough	84
notes	Marlborough	84
cherry	New_York	105
noir	New_York	83
tannins	New_York	76
palate	New_York	71
finish	New_York	64
fruit	Oregon	1730
flavors	Oregon	1187
cherry	Oregon	1092
finish	Oregon	787
tannins	Oregon	506

Group exercise

Use the top words by province to...

1. Engineer more features that capture the essence of Casablanca, Marlborough and New York
2. Look for difference between California and Oregon
3. Use what you find to run naive Bayes models that achieve a Kappa that approaches 0.5

Vocabulary

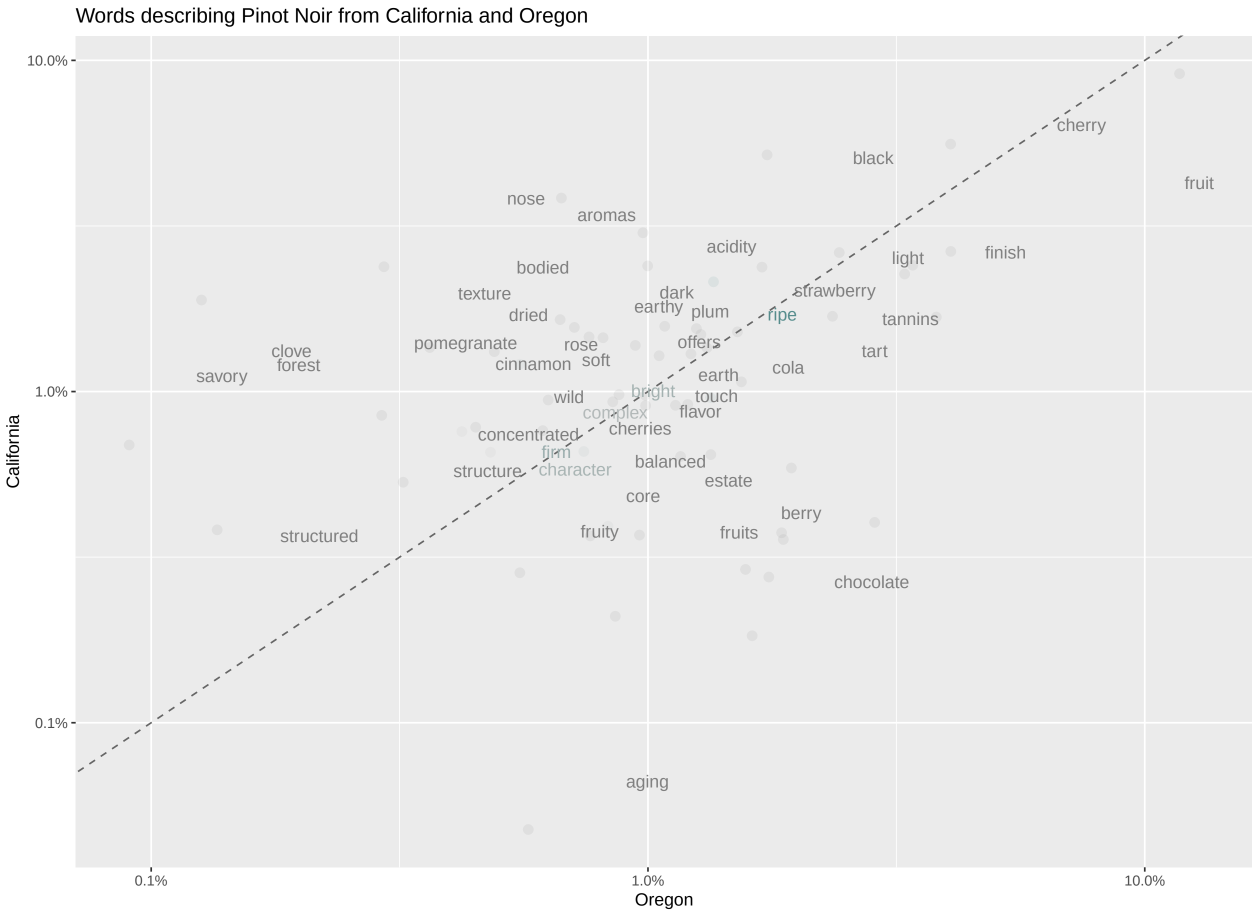
- Naive Bayes
- Correlation
- Residual
- Kappa
- Parameter Tuning
- Conditional Probability

Bonus Code

```
library(scales)
wtxt <- wine %>%
  unnest_tokens(word, description) %>%
  anti_join(stop_words) %>%
  filter(str_detect(string = word, pattern = "[a-z+]")) %>%  # get rid weird non alphas
  filter(str_length(word)>3) %>%  # get rid of strings shorter than 3 characters
  group_by(word) %>%
  mutate(total=n()) %>%
  ungroup()

wtxt %>%
    filter(province=="Oregon" | province=="California") %>%
    filter(!(word %in% c("wine","pinot","drink","noir","vineyard","palate","notes","flavors","bottling")
    filter(total > 400) %>%
    group_by(province, word) %>%
    count() %>%
    group_by(province) %>%
    mutate(proportion = n / sum(n)) %>%
```

```
pivot_wider(id_cols = word, names_from = province, values_from = proportion) %>%
ggplot(aes(x = Oregon, y = California, color = abs(Oregon - California))) +
geom_abline(color = "gray40", lty = 2) +
geom_jitter(alpha = 0.1, size = 2.5, width = 0.3, height = 0.3) +
geom_text(aes(label = word), check_overlap = TRUE, vjust = 1.5) +
scale_x_log10(labels = percent_format()) +
scale_y_log10(labels = percent_format()) +
scale_color_gradient(limits = c(0, 0.001), low = "darkslategray4", high = "gray75") +
theme(legend.position="none") +
labs(x = "Oregon", y = "California", title = "Words describing Pinot Noir from California and Orego
```




```
dtxt <- wtxt %>%
  filter(province=="Oregon" | province=="California") %>%
  filter(!(word %in% c("wine","pinot","drink","noir","vineyard","palate","notes","flavors","bottling","
  filter(total > 400) %>%
  group_by(province, word) %>%
  count() %>%
  group_by(province) %>%
  mutate(proportion = n / sum(n)) %>%
  pivot_wider(id_cols = word, names_from = province, values_from = proportion) %>%
  mutate(diff=Oregon-California)

dtxt %>%
  top_n(25, diff) %>%
  mutate(word = reorder(word, diff)) %>%
```

```
  ggplot(aes(word, diff)) +
  geom_col() +
  xlab(NULL) +
  coord_flip()
```