Feature Engineering I

Jameson Watts, Ph.D.
1/23/2020

Agenda

1. Data ethics review
2. What is feature engineering?
3. Overview of the ‘caret’ package
4. Parameter selection

Data ethics review

Problems

- Demographic data
- Profit optimizing
- Autonomous cars
- Recommendation engines
- Criminal sentencing
- Choice of classification model
- Killer robots

Reasonable people will disagree over subtle matters of right and wrong… thus, the important part of data ethics is committing to *consider* the ethical consequences of your choices.

The difference between “regular” ethics and data ethics is that algorithms scale really easily. Thus, seemingly small decisions can have wide-ranging impact.

What is feature engineering?

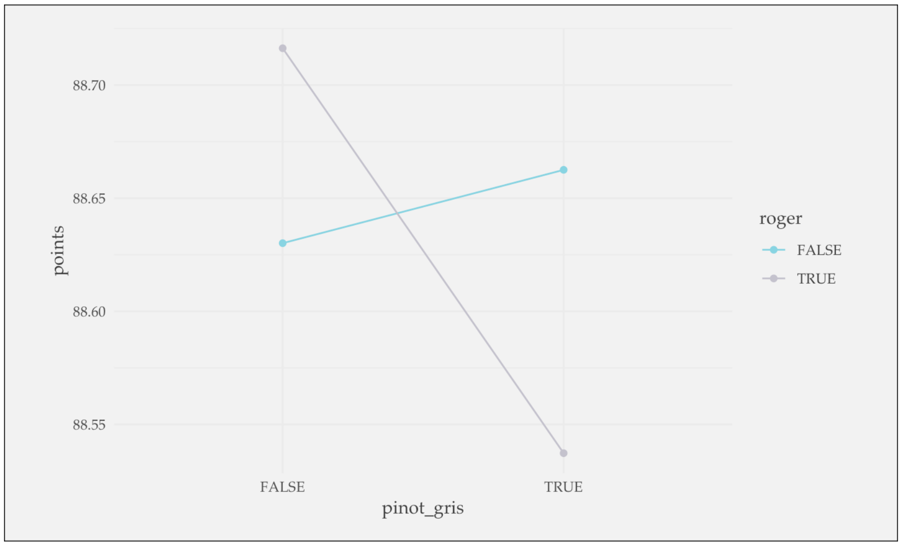
Setup

```
knitr::opts_chunk$set(echo = TRUE, message = FALSE, warning = FALSE)
library(tidyverse)
library(caret)
source('theme.R')
wine = read_rds("../resources/wine.rds")
```

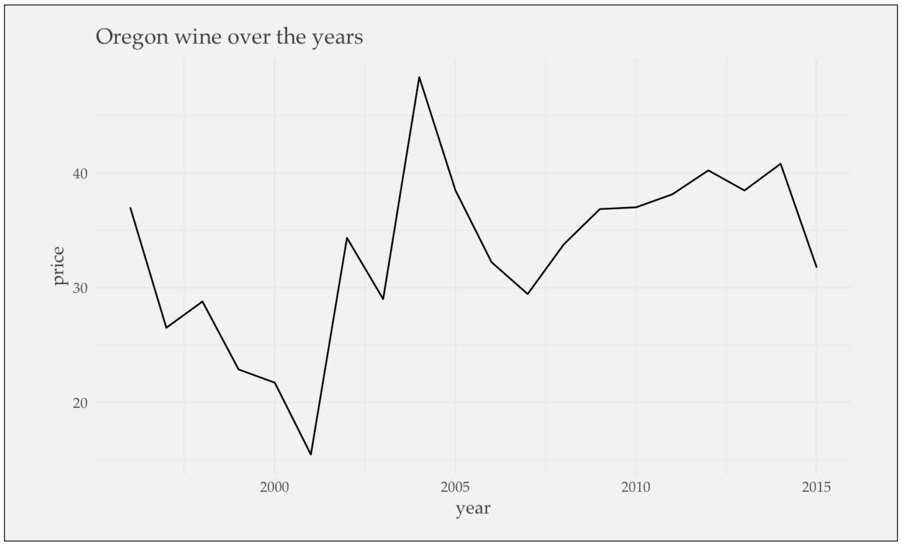
Exploratory visualizations

- Finding interactions
- Looking at correlations
- Assessing distribution of data

Example


88.70
88.65
88.60
88.55
points
FALSE
TRUE
pinot_gris
roger
FALSE
TRUE

Example


Oregon wine over the years
price
year
20
30
40
2000
2005
2010
2015

Encoding categorical predictors: few dummies

```
library(fastDummies)
wine %>%
  select(taster_name) %>%
  dummy_cols() %>%
  head()
```

taster_name	taster_name_Roger Voss	taster_name_Paul Gregutt	taster_name_Alexander Peartree	taster_name_Michael Schachner	taster_name_Kerin O'Keefe	taster_name_Anna Lee C. Iijima	taste
Roger Voss	1	0	0	0	0	0	
Paul Gregutt	0	1	0	0	0	0	
Alexander Peartree	0	0	1	0	0	0	
Paul Gregutt	0	1	0	0	0	0	
Michael Schachner	0	0	0	1	0	0	
Kerin O'Keefe	0	0	0	0	1	0	

Encoding categorical predictors: many dummies

```
wine %>%
  select(variety) %>%
  mutate(variety=fct_lump(variety,5)) %>%
  dummy_cols() %>%
  head()
```

variety	variety_Bordeaux-style Red Blend	variety_Cabernet Sauvignon	variety_Chardonnay	variety_Pinot Noir	variety_Red Blend	variety_Other
Other	0	0	0	0	0	1
Other	0	0	0	0	0	1
Other	0	0	0	0	0	1
Pinot Noir	0	0	0	1	0	0
Other	0	0	0	0	0	1
Other	0	0	0	0	0	1

Other types of engineered factors…

- Words or phrases in text
- A given time period
- An arbitrary numerical cut-off
- Etc.

Engineering numeric predictors: Box-Cox

Box-Cox transformations use MLE to estimate λ

$$x^* = \begin{cases} \frac{x^\lambda - 1}{\lambda\ \tilde{x}^{\lambda-1}}, & \lambda \neq 0 \\ \tilde{x}\ \log x, & \lambda = 0 \end{cases}$$

- when $\lambda = 1$, there is no transformation
- when $\lambda = 0$, it is log transformed
- when $\lambda = 0.5$, it is square root
- when $\lambda = -1$, it is an inverse

Calculating Box-Cox

```
x <- as.data.frame(select(wine,price,points))
x_proc <- preProcess(x, method = "BoxCox")
x_proc
```

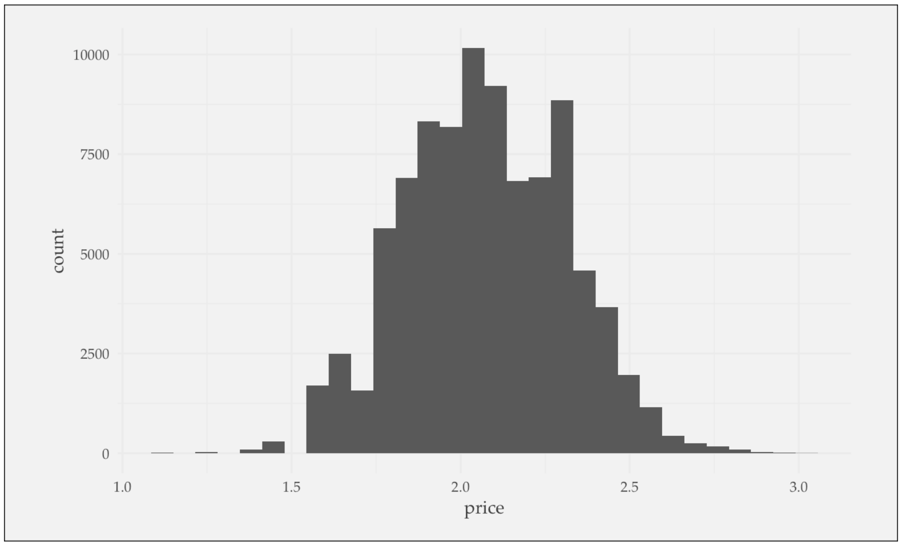
```
## Created from 89556 samples and 2 variables
##
## Pre-processing:
##   - Box-Cox transformation (2)
##   - ignored (0)
##
## Lambda estimates for Box-Cox transformation:
## -0.3, 1.1
```

```
x_new <- predict(x_proc, x)
head(x_new)
```

price	points
1.854050	87
1.823113	87
1.789161	87
2.380527	87
1.854050	87
1.882416	87

Histogram after Box-Cox

```
ggplot(x_new,aes(price))+geom_histogram()
```



Engineering numeric predictors: Standardizing

- mean-centering $x - \bar{x}$
- scaling: $x/std(x)$

…allows for common scale across variables. Also helps reduce bias when interactions are included (i.e. eliminates variance inflation).

And there are many other transformations that you can read about.

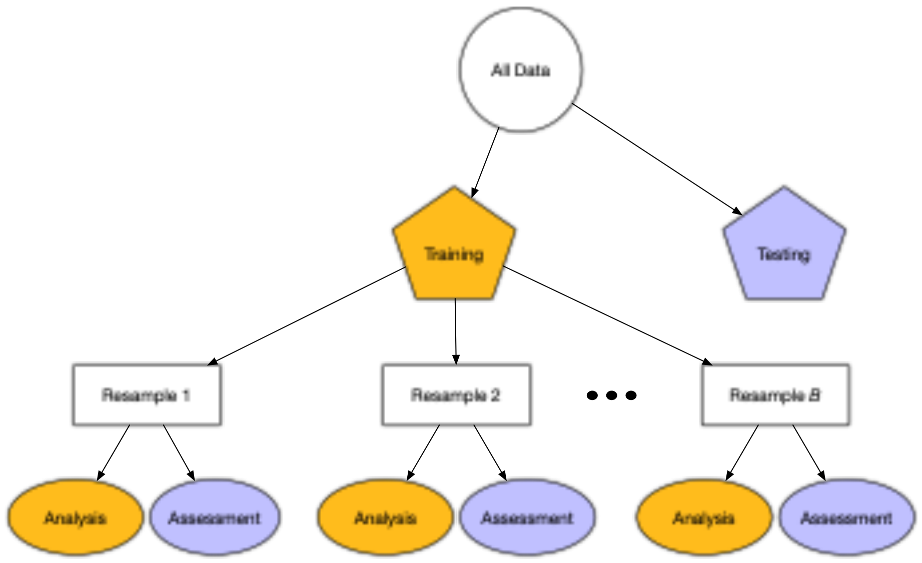
Interaction effects

This chapter has a good overview of interactions.

- start with domain knowledge
- use visualizations
- 3-way interactions exist, but are rare
- brute force is a last resort

The ‘caret’ package

Philosophy


All Data
Training
Testing
Resample 1
Resample 2
Resample B
Analysis
Assessment
Analysis
Assessment
Analysis
Assessment

Types of resampling

- V-fold Cross-Validation
- Monte Carlo Cross-Validation
- The Bootstrap

Typical setup

```r
wino <- wine %>% ## look ma, engineered features!
  mutate(fr=(country=="France")) %>%
  mutate(cab=str_detect(variety,"Cabernet")) %>%
  mutate(lprice=log(price)) %>%
  drop_na(fr, cab) %>%
  select(lprice, points, fr, cab)

wine_index <- createDataPartition(wino$lprice, p = 0.8, list = FALSE)
wino_tr <- wino[ wine_index, ]
wino_te <- wino[-wine_index, ]

set.seed(5004)
lm_fit <- train(lprice ~ .,
                data = wino_tr,
                method = "lm",
                trControl = trainControl(number = 1))
```

Follow this link for the full documentation on caret.

Train vs. test

```
lm_fit
```

```
## Linear Regression
##
## 71603 samples
##     3 predictor
##
## No pre-processing
## Resampling: Bootstrapped (1 reps)
## Summary of sample sizes: 71603
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.5114613  0.3859124  0.4033647
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
wine_pred <- predict(lm_fit, wino_te)
postResample(pred=wine_pred, obs = wino_te$lprice)
```

```
##      RMSE  Rsquared       MAE
## 0.5124344 0.3931137 0.4028655
```

Exercise (30-40 minutes)

1. Gather in your modeling teams
2. Create 5 new features (in addition to points)
3. Create training and test data
4. Use your new predictors to train a linear regression model
5. Report RMSE on test set

Parameter selection

Stepwise selection is bad

Harrell (2015) provides a comprehensive indictment of the method that can be encapsulated by the statement:

"… if this procedure had just been proposed as a statistical method, it would most likely be rejected because it violates every principle of statistical estimation and hypothesis testing."

Reference: Harrell, F. 2015. Regression Modeling Strategies. Springer.

Basic model with 11 parameters

```
wino <- wine %>%
  mutate(country=fct_lump(country,5)) %>%
  mutate(variety=fct_lump(variety,5)) %>%
  mutate(lprice=log(price)) %>%
  select(lprice, points, country, variety) %>%
  drop_na(.)

library(fastDummies)
wino <- dummy_cols(wino, remove_selected_columns = T) %>%
  select(-country_Other, -variety_Other) %>%
  rename_all(funs(tolower(.))) %>%
  rename_all(funs(str_replace_all(., "-", "_"))) %>%
  rename_all(funs(str_replace_all(., " ", "_")))

wine_index <- createDataPartition(wino$lprice, p = 0.8, list = FALSE)
wino_tr <- wino[ wine_index, ]
wino_te <- wino[-wine_index, ]

set.seed(5004)
lm_fit <- train(lprice ~ .,
                data = wino_tr,
                method = "lm",
                trControl = trainControl(number = 1))
```

Results

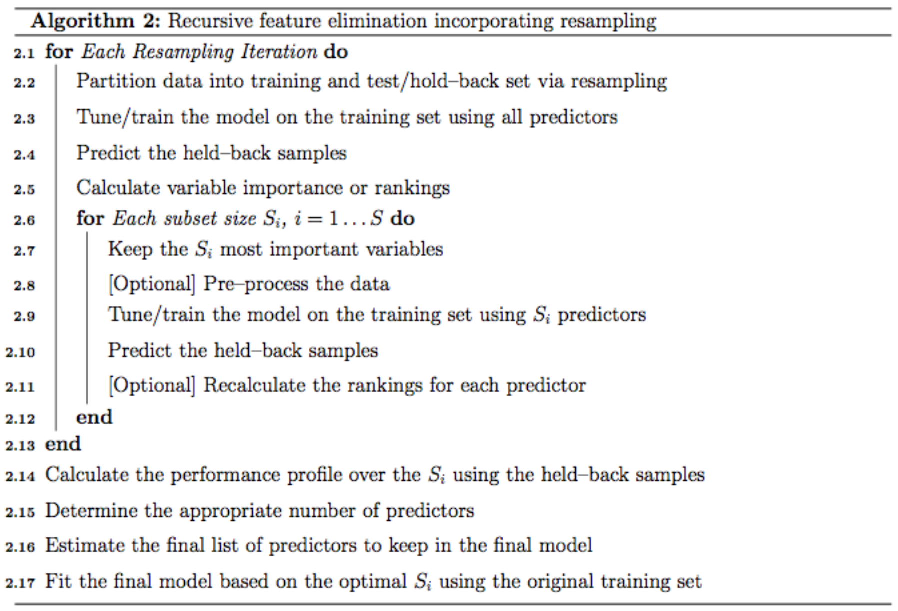
```
lm_fit
```

```
## Linear Regression
##
## 71603 samples
##    11 predictor
##
## No pre-processing
## Resampling: Bootstrapped (1 reps)
## Summary of sample sizes: 71603
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.4891515  0.4488954  0.3806915
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
wine_pred <- predict(lm_fit, wino_te)
postResample(pred=wine_pred, obs = wino_te$lprice)
```

```
##      RMSE  Rsquared       MAE
## 0.4880510 0.4468570 0.3804876
```

Recursive feature elimination



Algorithm 2: Recursive feature elimination incorporating resampling

```
2.1  for Each Resampling Iteration do
2.2      Partition data into training and test/hold–back set via resampling
2.3      Tune/train the model on the training set using all predictors
2.4      Predict the held–back samples
2.5      Calculate variable importance or rankings
2.6      for Each subset size $S_i$, $i = 1 \ldots S$ do
2.7          Keep the $S_i$ most important variables
2.8          [Optional] Pre–process the data
2.9          Tune/train the model on the training set using $S_i$ predictors
2.10         Predict the held–back samples
2.11         [Optional] Recalculate the rankings for each predictor
2.12     end
2.13 end
2.14 Calculate the performance profile over the $S_i$ using the held–back samples
2.15 Determine the appropriate number of predictors
2.16 Estimate the final list of predictors to keep in the final model
2.17 Fit the final model based on the optimal $S_i$ using the original training set
```

Using recursive feature elimination in caret

```
x <- select(wino_tr,-lprice)
y <- wino_tr$lprice
subsets <- c(1:11)
lmProfile <- rfe(x, y,
                 sizes = subsets,
                 rfeControl = rfeControl(functions = lmFuncs, returnResamp = "all"))
```

Results

```
lmProfile
```

```
##
## Recursive feature selection
##
## Outer resampling method: Bootstrapped (25 reps)
##
## Resampling performance over subset size:
##
##  Variables   RMSE Rsquared    MAE   RMSESD RsquaredSD    MAESD Selected
##          1 0.6510  0.01867 0.5201 0.005113   0.013625 0.007404
##          2 0.6368  0.06103 0.5025 0.002298   0.002180 0.001374
##          3 0.6328  0.07283 0.4990 0.002306   0.002321 0.001361
##          4 0.6303  0.08022 0.4978 0.002299   0.002365 0.001436
##          5 0.6255  0.09408 0.4907 0.003552   0.007245 0.005045
##          6 0.6213  0.10623 0.4846 0.002505   0.002763 0.001473
##          7 0.4884  0.44783 0.3793 0.001705   0.002880 0.001258
##          8 0.4876  0.44947 0.3787 0.001678   0.002951 0.001222
##          9 0.4871  0.45065 0.3784 0.001727   0.002903 0.001225
##         10 0.4865  0.45199 0.3778 0.001738   0.002965 0.001266
##         11 0.4859  0.45332 0.3773 0.001728   0.002970 0.001246        *
##
## The top 5 variables (out of 11):
##    country_italy, variety_pinot_noir, variety_cabernet_sauvignon, variety_bordeaux_style_red_blend, country_portu
```

30/33

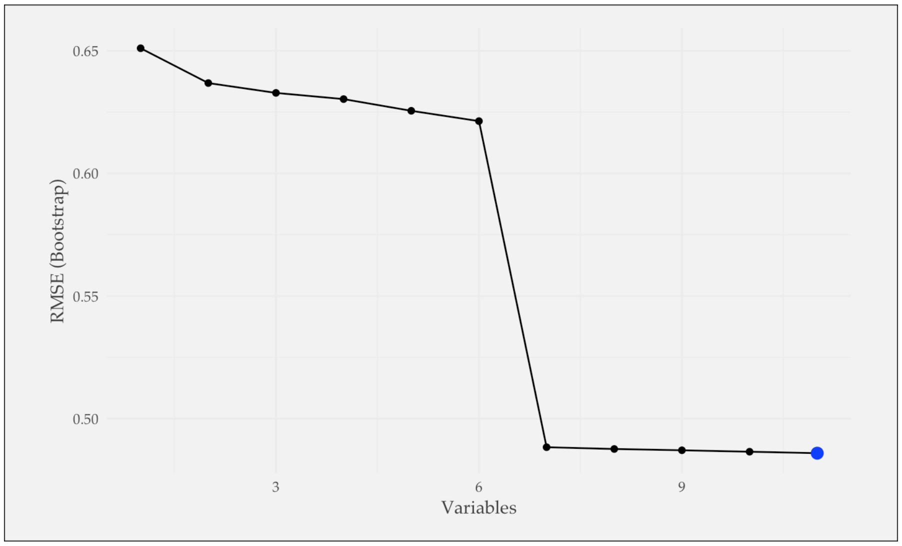
Coefficients

```
lmProfile$fit
```

```
##
## Call:
## lm(formula = y ~ ., data = tmp)
##
## Coefficients:
##                      (Intercept)                     country_italy
##                         -8.46192                           0.33462
##               variety_pinot_noir        variety_cabernet_sauvignon
##                          0.32773                           0.27115
## variety_bordeaux_style_red_blend                  country_portugal
##                          0.19336                          -0.16159
##                       country_us                            points
##                          0.15723                           0.13064
##                   country_france                     country_spain
##                          0.11276                           0.10281
##               variety_chardonnay                 variety_red_blend
##                          0.09754                           0.09101
```

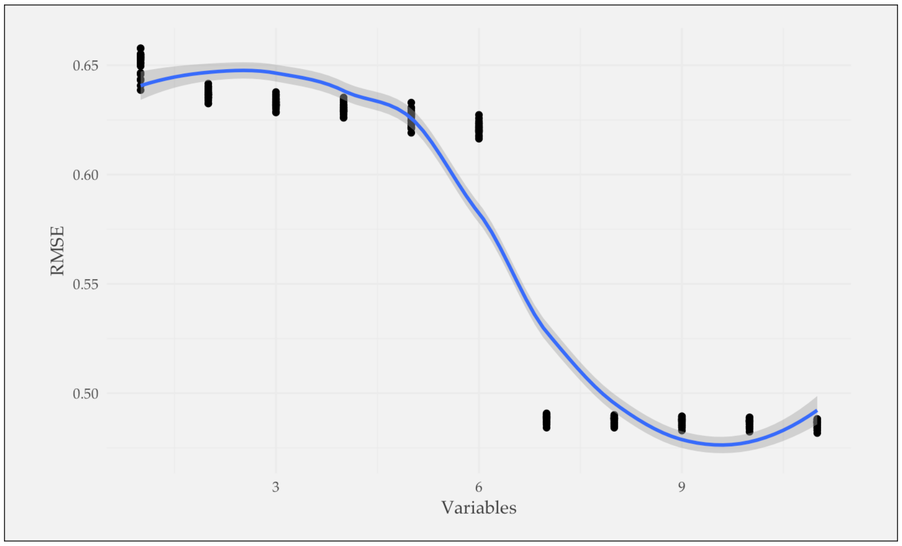
Graphing performance gains

```
ggplot(lmProfile)
```



Visualizing the resampling

```
lmProfile[["resample"]] %>%
  ggplot(aes(Variables, RMSE))+
  geom_point()+
  geom_smooth()
```


RMSE
Variables