Feature engineering

Data Science in a Box

datasciencebox.org

Modified by Tyler George

Feature engineering

Feature engineering

- We prefer simple models when possible, but **parsimony** does not mean sacrificing accuracy (or predictive performance) in the interest of simplicity

Feature engineering

- We prefer simple models when possible, but **parsimony** does not mean sacrificing accuracy (or predictive performance) in the interest of simplicity
- Variables that go into the model and how they are represented are just as critical to success of the model

Feature engineering

- We prefer simple models when possible, but **parsimony** does not mean sacrificing accuracy (or predictive performance) in the interest of simplicity
- Variables that go into the model and how they are represented are just as critical to success of the model
- **Feature engineering** allows us to get creative with our predictors in an effort to make them more useful for our model (to increase its predictive performance)

Same training and testing sets as before

```r
# Fix random numbers by setting the seed
# Enables analysis to be reproducible when random numbers are used
set.seed(1116)

# Put 80% of the data into the training set
email_split <- initial_split(email, prop = 0.80)

# Create data frames for the two sets:
train_data <- training(email_split)
test_data  <- testing(email_split)
```

A simple approach: mutate()

```r
train_data %>%
  mutate(
    date = lubridate::date(time),
    dow  = wday(time),
    month = month(time)
    ) %>%
  select(time, date, dow, month) %>%
  sample_n(size = 5) # shuffle to show a variety
```

```
## # A tibble: 5 x 4
##   time                date         dow month
##   <dttm>              <date>     <dbl> <dbl>
## 1 2012-03-15 13:51:35 2012-03-15     5     3
## 2 2012-03-03 08:24:02 2012-03-03     7     3
## 3 2012-01-18 10:55:23 2012-01-18     4     1
## 4 2012-02-24 22:08:59 2012-02-24     6     2
## 5 2012-01-11 07:18:51 2012-01-11     4     1
```

Modeling workflow, revisited

- Create a **recipe** for feature engineering steps to be applied to the training data

Modeling workflow, revisited

- Create a **recipe** for feature engineering steps to be applied to the training data
- Fit the model to the training data after these steps have been applied

Modeling workflow, revisited

- Create a **recipe** for feature engineering steps to be applied to the training data
- Fit the model to the training data after these steps have been applied
- Using the model estimates from the training data, predict outcomes for the test data

Modeling workflow, revisited

- Create a **recipe** for feature engineering steps to be applied to the training data
- Fit the model to the training data after these steps have been applied
- Using the model estimates from the training data, predict outcomes for the test data
- Evaluate the performance of the model on the test data

Building recipes

Initiate a recipe

```r
email_rec <- recipe(
  spam ~ .,           # formula
  data = train_data  # data to use for cataloguing names and types of variables
  )

summary(email_rec)
```

```
## # A tibble: 21 x 4
##    variable     type    role      source
##    <chr>        <chr>   <chr>     <chr>
##  1 to_multiple  nominal predictor original
##  2 from         nominal predictor original
##  3 cc           numeric predictor original
##  4 sent_email   nominal predictor original
##  5 time         date    predictor original
##  6 image        numeric predictor original
##  7 attach       numeric predictor original
##  8 dollar       numeric predictor original
##  9 winner       nominal predictor original
## 10 inherit      numeric predictor original
## 11 viagra       numeric predictor original
## 12 password     numeric predictor original
## 13 num_char     numeric predictor original
## 14 line_breaks  numeric predictor original
## 15 format       nominal predictor original
## 16 re_subj      nominal predictor original
## 17 exclaim_subj numeric predictor original
## 18 urgent_subj  nominal predictor original
## 19 exclaim_mess numeric predictor original
## 20 number       nominal predictor original
## 21 spam         nominal outcome   original
```

Remove certain variables

```r
email_rec <- email_rec %>%
  step_rm(from, sent_email)
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor         20
##
## Operations:
##
## Delete terms from, sent_email
```

Feature engineer date

```r
email_rec <- email_rec %>%
  step_date(time, features = c("dow", "month")) %>%
  step_rm(time)
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor         20
##
## Operations:
##
## Delete terms from, sent_email
## Date features from time
## Delete terms time
```

Discretize numeric variables

```
email_rec <- email_rec %>%
  step_cut(cc, attach, dollar, breaks = c(0, 1)) %>%
  step_cut(inherit, password, breaks = c(0, 1, 5, 10, 20))
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor         20
##
## Operations:
##
## Delete terms from, sent_email
## Date features from time
## Delete terms time
## Cut numeric for cc, attach, dollar
## Cut numeric for inherit, password
```

Create dummy variables

```r
email_rec <- email_rec %>%
  step_dummy(all_nominal(), -all_outcomes())
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor         20
##
## Operations:
##
## Delete terms from, sent_email
## Date features from time
## Delete terms time
## Cut numeric for cc, attach, dollar
## Cut numeric for inherit, password
## Dummy variables from all_nominal(), -all_outcomes()
```

Remove zero variance variables

Variables that contain only a single value

```
email_rec <- email_rec %>%
  step_zv(all_predictors())
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor         20
##
## Operations:
##
## Delete terms from, sent_email
## Date features from time
## Delete terms time
## Cut numeric for cc, attach, dollar
## Cut numeric for inherit, password
## Dummy variables from all_nominal(), -all_outcomes()
## Zero variance filter on all_predictors()
```

All in one place

```
email_rec <- recipe(spam ~ ., data = email) %>%
  step_rm(from, sent_email) %>%
  step_date(time, features = c("dow", "month")) %>%
  step_rm(time) %>%
  step_cut(cc, attach, dollar, breaks = c(0, 1)) %>%
  step_cut(inherit, password, breaks = c(0, 1, 5, 10, 20)) %>%
  step_dummy(all_nominal(), -all_outcomes()) %>%
  step_zv(all_predictors())
```

Building workflows

Define model

```r
email_mod <- logistic_reg() %>%
  set_engine("glm")

email_mod
```

```
## Logistic Regression Model Specification (classification)
##
## Computational engine: glm
```

Define workflow

Workflows bring together models and recipes so that they can be easily applied to both the training and test data.

```
email_wflow <- workflow() %>%
  add_model(email_mod) %>%
  add_recipe(email_rec)
```

```
## == Workflow ==========================================================================
## Preprocessor: Recipe
## Model: logistic_reg()
##
## -- Preprocessor ----------------------------------------------------------------------
## 7 Recipe Steps
##
## * step_rm()
## * step_date()
## * step_rm()
## * step_cut()
## * step_cut()
## * step_dummy()
## * step_zv()
##
## -- Model -----------------------------------------------------------------------------
## Logistic Regression Model Specification (classification)
##
## Computational engine: glm
```

Fit model to training data

```
email_fit <- email_wflow %>%
  fit(data = train_data)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
tidy(email_fit) %>% print(n = 31)
```

```
## # A tibble: 31 x 5
##    term               estimate  std.error statistic  p.value
##    <chr>                 <dbl>      <dbl>     <dbl>    <dbl>
##  1 (Intercept)        -0.914      0.251     -3.65   2.62e- 4
##  2 image              -1.65       0.933     -1.76   7.78e- 2
##  3 viagra              2.27     182.         0.0125 9.90e- 1
##  4 num_char            0.0469     0.0243     1.93   5.40e- 2
##  5 line_breaks        -0.00509    0.00138   -3.68   2.32e- 4
##  6 exclaim_subj       -0.202      0.277     -0.729  4.66e- 1
##  7 exclaim_mess        0.00882    0.00186    4.74   2.17e- 6
##  8 to_multiple_X1     -2.61       0.354     -7.37   1.69e-13
##  9 cc_X.1.68.         -0.312      0.489     -0.638  5.24e- 1
## 10 attach_X.1.21.      2.05       0.368      5.57   2.55e- 8
## 11 dollar_X.1.64.      0.218      0.217      1.00   3.16e- 1
## 12 winner_yes          2.18       0.428      5.08   3.71e- 7
## 13 inherit_X.1.5.     -9.25     764.        -0.0121 9.90e- 1
## 14 inherit_X.5.10.     2.52       1.44       1.75   7.97e- 2
## 15 password_X.1.5.    -1.71       0.749     -2.29   2.22e- 2
## 16 password_X.5.10.  -12.5      475.        -0.0263 9.79e- 1
## 17 password_X.10.20. -13.7      813.        -0.0168 9.87e- 1
## 18 password_X.20.22. -13.9     1029.        -0.0135 9.89e- 1
## 19 format_X1          -0.920      0.159     -5.79   6.95e- 9
## 20 re_subj_X1         -2.91       0.437     -6.65   2.88e-11
## 21 urgent_subj_X1      3.52       1.08       3.25   1.16e- 3
## 22 number_small       -0.902      0.168     -5.38   7.43e- 8
## 23 number_big         -0.209      0.250     -0.838  4.02e- 1
## 24 time_dow_Mon        0.134      0.297      0.453  6.51e- 1
## 25 time_dow_Tue        0.441      0.268      1.65   9.99e- 2
## 26 time_dow_Wed       -0.131      0.275     -0.478  6.33e- 1
## 27 time_dow_Thu        0.123      0.279      0.442  6.58e- 1
## 28 time_dow_Fri        0.0896     0.283      0.316  7.52e- 1
## 29 time_dow_Sat        0.277      0.300      0.923  3.56e- 1
## 30 time_month_Feb      0.760      0.180      4.22   2.41e- 5
## 31 time_month_Mar      0.519      0.180      2.88   4.01e- 3
```

Make predictions for test data

```r
email_pred <- predict(email_fit, test_data, type = "prob") %>%
  bind_cols(test_data)
```

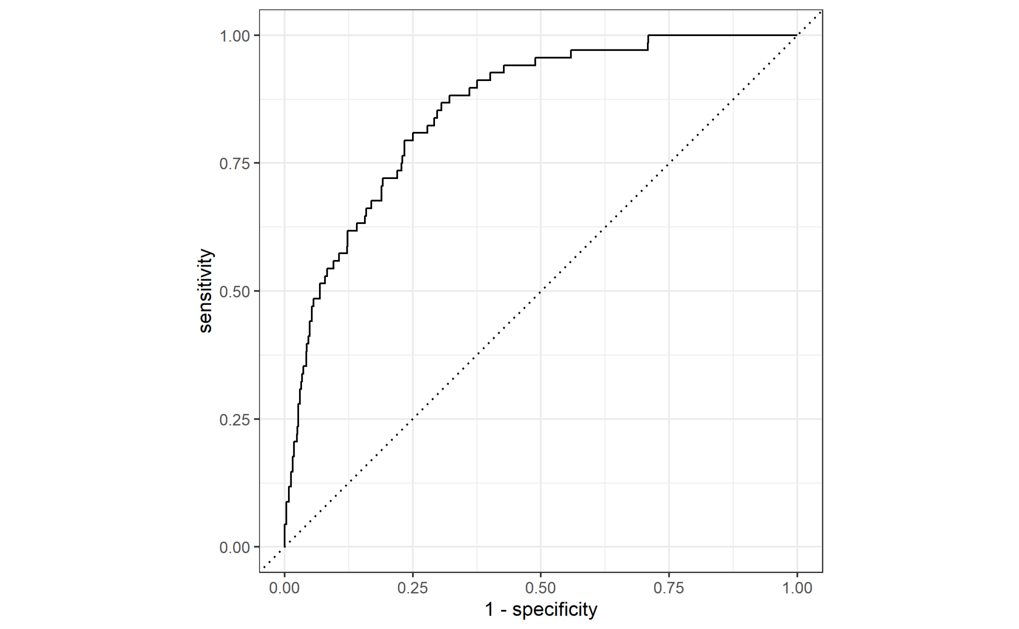
```
## Warning: There are new levels in a factor: NA
```

```r
email_pred
```

```
## # A tibble: 785 x 23
##    .pred_0  .pred_1 spam  to_multiple from     cc sent_email
##      <dbl>    <dbl> <fct> <fct>       <fct> <int> <fct>
## 1    0.995 0.00451  0     1           1         0 1
## 2    0.999 0.00129  0     0           1         1 1
## 3    0.969 0.0306   0     0           1         0 0
## 4    0.999 0.000816 0     0           1         1 0
## 5    0.993 0.00680  0     0           1         4 0
## 6    0.852 0.148    0     0           1         0 0
## # ... with 779 more rows, and 16 more variables: time <dttm>,
## #   image <dbl>, attach <dbl>, dollar <dbl>, winner <fct>,
## #   inherit <dbl>, viagra <dbl>, password <dbl>, num_char <dbl>,
## #   line_breaks <int>, format <fct>, re_subj <fct>,
## #   exclaim_subj <dbl>, urgent_subj <fct>, exclaim_mess <dbl>,
## #   number <fct>
```

Evaluate the performance

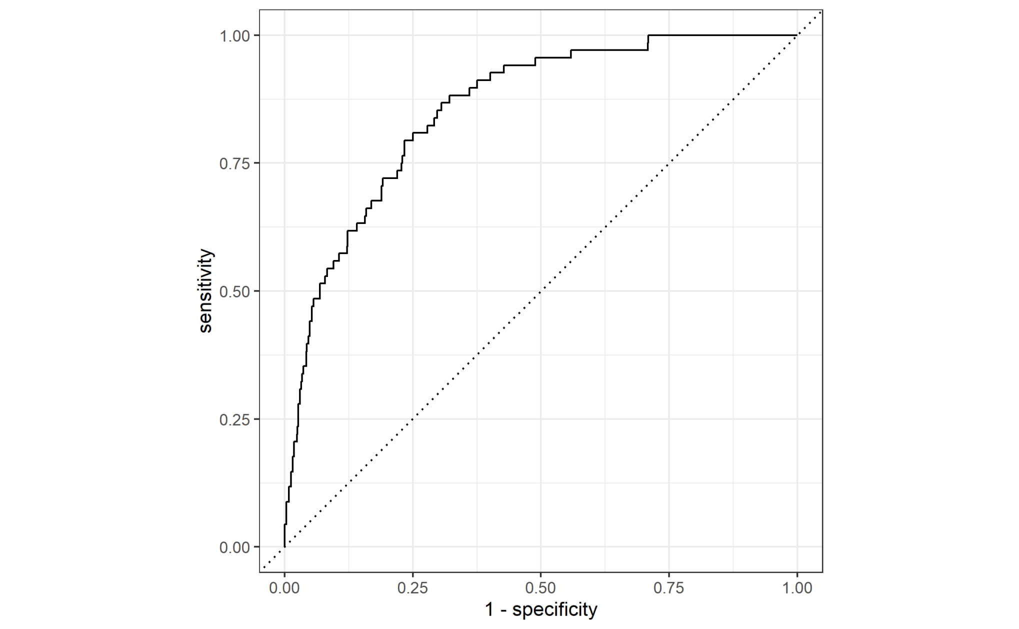
```
email_pred %>%
  roc_curve(
    truth = spam,
    .pred_1,
    event_level = "second"
  ) %>%
  autoplot()
```



Evaluate the performance

```
email_pred %>%
  roc_auc(
    truth = spam,
    .pred_1,
    event_level = "second"
  )
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.856
```



Making decisions

Cutoff probability: 0.5

Output Code

Suppose we decide to label an email as spam if the model predicts the probability of spam to be **more than 0.5**.

	Email is not spam	Email is spam
Email labelled not spam	708	60
Email labelled spam	8	8
NA	1	NA

Cutoff probability: 0.5

Output Code

```r
cutoff_prob <- 0.5
email_pred %>%
  mutate(
    spam      = if_else(spam == 1, "Email is spam", "Email is not spam"),
    spam_pred = if_else(.pred_1 > cutoff_prob, "Email labelled spam", "Email labelled not spam")
    ) %>%
  count(spam_pred, spam) %>%
  pivot_wider(names_from = spam, values_from = n) %>%
  kable(col.names = c("", "Email is not spam", "Email is spam"))
```

Cutoff probability: 0.25

Output Code

Suppose we decide to label an email as spam if the model predicts the probability of spam to be **more than 0.25**.

	Email is not spam	Email is spam
Email labelled not spam	665	33
Email labelled spam	51	35
NA	1	NA

Cutoff probability: 0.25

Output Code

```r
cutoff_prob <- 0.25
email_pred %>%
  mutate(
    spam      = if_else(spam == 1, "Email is spam", "Email is not spam"),
    spam_pred = if_else(.pred_1 > cutoff_prob, "Email labelled spam", "Email labelled not spam")
    ) %>%
  count(spam_pred, spam) %>%
  pivot_wider(names_from = spam, values_from = n) %>%
  kable(col.names = c("", "Email is not spam", "Email is spam"))
```

Cutoff probability: 0.75

Output Code

Suppose we decide to label an email as spam if the model predicts the probability of spam to be **more than 0.75**.

	Email is not spam	Email is spam
Email labelled not spam	714	65
Email labelled spam	2	3
NA	1	NA

Cutoff probability: 0.75

Output Code

```
cutoff_prob <- 0.75
email_pred %>%
  mutate(
    spam      = if_else(spam == 1, "Email is spam", "Email is not spam"),
    spam_pred = if_else(.pred_1 > cutoff_prob, "Email labelled spam", "Email labelled not spam")
    ) %>%
  count(spam_pred, spam) %>%
  pivot_wider(names_from = spam, values_from = n) %>%
  kable(col.names = c("", "Email is not spam", "Email is spam"))
```