Classification with KNN and Logistic Regression

Classification with K-Nearest-Neighbors

KNN

We have existing observations

$$(x_1, C_1), \dots (x_n, C_n)$$

where the C_i are **categories**.

Given a new observation x_{new}, how do we predict C_{new}?

1. Find the 5 values in $(x_1, \dots, x_n)$ that are closest to x_{new}
2. Let all the closest neighbors "vote" on the category.
3. Predict $\hat{C}_{new}$ = the category with the most votes.

KNN

To perform **classification** with K-Nearest-Neighbors, we choose the **K** closest observations to our *target*, and we **aggregate** their *response* values.

The Big Questions:

- What is our definition of **closest**?
- What number should we use for **K**?

Example

Example

Let's keep hanging out with the insurance dataset.

Suppose we want to use information about insurance charges to predict whether someone is a smoker or not.

```
ins <- read_csv("https://www.dropbox.com/s/bocjjyo1ehr5auz/insurance.csv?dl=1")
head(ins)
```

```
## # A tibble: 6 x 6
##     age sex      bmi smoker region    charges
##   <dbl> <chr>  <dbl> <chr>  <chr>       <dbl>
## 1    19 female  27.9 yes    southwest  16885.
## 2    33 male    22.7 no     northwest  21984.
## 3    32 male    28.9 no     northwest   3867.
## 4    31 female  25.7 no     southeast   3757.
## 5    60 female  25.8 no     northwest  28923.
## 6    25 male    26.2 no     northeast   2721.
```

Example

Establish our model:

```
knn_mod <- nearest_neighbor(neighbors = 5) %>%
  set_engine("kknn") %>%
  set_mode("classification")
```

- New *mode* - "classification"
- Everything else is the same!

Example

Fit our model:

```
knn_fit_1 <- knn_mod %>%
  fit(smoker ~ charges, data = ins)
```

```
## Error: For classification models, the outcome should be a factor.
```

```
knn_fit_1$fit %>% summary()
```

```
## Error in summary(.): object 'knn_fit_1' not found
```

```
knn_fit_1 <- knn_mod %>%
  fit(smoker ~ charges, data = ins)
```

```
## Error: For classification models, the outcome should be a factor.
```

```
knn_fit_1$fit %>% summary()
```

```
## Error in summary(.): object 'knn_fit_1' not found
```

Example

```r
ins <- ins %>%
  mutate(
    smoker = factor(smoker)
  ) %>%
  drop_na()
```

Example

```
knn_fit_1 <- knn_mod %>%
  fit(smoker ~ charges, data = ins)

knn_fit_1$fit %>% summary()
```

```
##
## Call:
## kknn::train.kknn(formula = smoker ~ charges, data = data, ks = ~5)
##
## Type of response variable: nominal
## Minimal misclassification: 0.06032
## Best kernel: optimal
## Best k: 5
```

Try it!

Open Activity-Classification.Rmd or equivalent

Select the best KNN model for predicting smoker status

(What metrics does the cross-validation process automatically output?)

Logistic Regression

Ordinary linear regression

```r
lm_mod <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("classification")
```

Ordinary linear regression

Consider the following idea:

1. Convert the `smoker` variable to a dummy variable:

```
ins <- ins %>%
  mutate(
    smoker_number = case_when(
      smoker == "yes" ~ 1,
      smoker == "no" ~ 0
    )
  )
```

Ordinary linear regression

Consider the following idea:

1. Fit a linear regression predicting `smoker` dummy var:

```
decorate("reg_fit") %>%
  flair("-smoker") %>%
  flair("lm_mod") %>%
  flair("smoker_number")
```

```
lm_mod <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")

ins_rec <- recipe(smoker_number ~ charges, data = ins) %>%
  step_normalize(all_numeric(), -smoker_number)

ins_wflow <- workflow() %>%
  add_recipe(ins_rec) %>%
  add_model(lm_mod)
```

Ordinary linear regression

Consider the following idea:

1. Predict each observation to be the smoker closest to the number:

```
preds <- ins_fit %>% predict(ins)

ins <- ins %>%
  mutate(
    predicted_num = preds$.pred,
    predicted_smoker = round(predicted_num)
  )
```

Ordinary linear regression

How did we do?

```
ins %>%
  count(predicted_smoker, smoker_number)
```

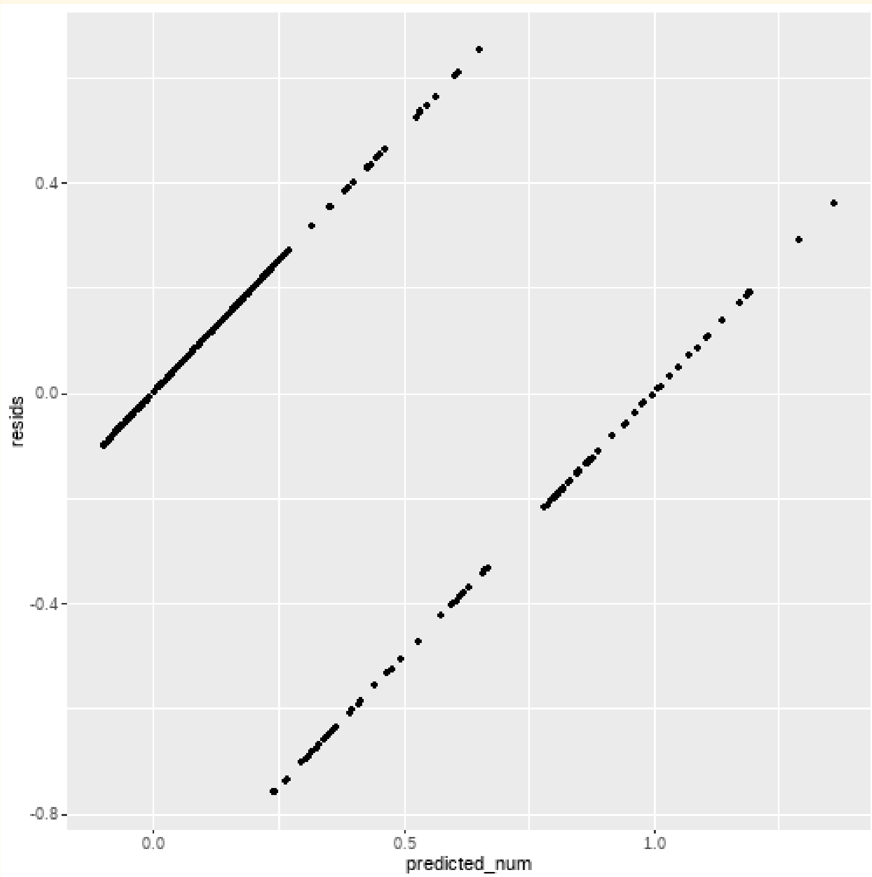
```
## # A tibble: 4 x 3
##   predicted_smoker smoker_number     n
##              <dbl>         <dbl> <int>
## 1                0             0   336
## 2                0             1    28
## 3                1             0     8
## 4                1             1    59
```

Ordinary linear regression

What's wrong with this?

Linear regression assumes that the residuals are **Normally distributed**. Obviously, they are not here.

Residuals



Logistic Regression

Solution: How about the same approach, Y is a function of X plus noise, but we let the noise be non-Normal?

Y = f(X) + `(\epsilon)`

$$Y = g^{-1}(\beta_0 + \beta_1 X + \epsilon)$$

for some function g.

Logistic Regression

Easier way to think of it:

Before:

$$\mu_Y = \beta_0 + \beta_1 X$$

Now:

$$g(\mu_Y) = \beta_0 + \beta_1 X$$

(g is called the **link function**)

Logistic Regression

A common transformation is the **logistic**:

$$g(u) = \log(u) / \log(1 - u)$$

In this case, u represents the *probability* of someone being a smoker.

Our observations Y have probability 0 or 1, since we observe them.

Future observations are unknown, so we predict them.

Logistic Regression

In summary:

- Given *predictors*, we try to predict the **log-odds** of a person being a smoker.
- We assume random noise on the relationship between the predictors and the **log-odds** of the response
- From these **log-odds**, we calculate the **probabilities**.
- We compare the **probabilities** (between 0 and 1) to the **observed truths** (0 or 1 exactly).

Logistic Regression

New model:

```
logit_mod <- logistic_reg() %>%
  set_mode("classification") %>%
  set_engine("glm")
```

Same recipe (but sticking with the original `smoker` variable now):

```
ins_rec <- recipe(smoker ~ charges, data = ins) %>%
  step_normalize(all_numeric())
```

Logistic Regression

New workflow:

```
ins_wflow_logit <- workflow() %>%
  add_recipe(ins_rec) %>%
  add_model(logit_mod)

ins_fit <- ins_wflow_logit %>%
  fit(ins)

ins_fit %>% pull_workflow_fit()
```

```
## parsnip model object
##
## Fit time:  10ms
##
## Call:  stats::glm(formula = ..y ~ ., family = stats::binomial, data = data)
##
## Coefficients:
## (Intercept)      charges
##       -2.62         3.62
##
## Degrees of Freedom: 430 Total (i.e. Null);  429 Residual
## Null Deviance:        434
## Residual Deviance: 139     AIC: 143
```

Logistic Regression

Notice: Now our predictions are of the type **pred_class**!

R did the hard part for us.

```
preds <- ins_fit %>% predict(ins)
preds
```

```
## # A tibble: 431 x 1
##    .pred_class
##    <fct>
##  1 no
##  2 yes
##  3 no
##  4 no
##  5 yes
##  6 no
##  7 yes
##  8 no
##  9 yes
## 10 no
## # ... with 421 more rows
```

Logistic Regression

Suppose we wanted to see the predicted log-odds values:

```
ins_fit %>% predict(ins, type = "raw")
```

```
##         1         2         3         4         5         6         7         8
## -1.225256  0.328346 -5.191275 -5.224858  2.442245 -5.540267  2.102734 -2.990489
##         9        10        11        12        13        14        15        16
##  5.698586 -5.639630  4.853385 -2.339098  4.471990 -5.699642 -5.667924  8.306921
##        17        18        19        20        21        22        23        24
## -5.441333 -0.084214 -5.285697  5.129131 -1.887326 -5.838252 -1.318829 -2.909468
##        25        26        27        28        29        30        31        32
## -5.902457 -5.530000 -4.367436 -3.951805  6.907207 -2.995833 -2.992969 -5.751803
##        33        34        35        36        37        38        39        40
## -3.253843  0.458793 -1.549477 -4.484697 -5.258837  0.133482 -5.495849 -2.901954
##        41        42        43        44        45        46        47        48
## -5.849704 -5.681580 -2.465043 -5.712592 -5.985026 -5.746103 -5.709225 -3.996180
##        49        50        51        52        53        54        55        56
## -0.286973 -4.252375  0.096566  4.887747  4.643787 -2.285313 -5.853003 -5.499408
##        57        58        59        60        61        62        63        64
##  8.505278 -4.881437 -5.718084 -6.022795 -5.869830 -4.648117 -3.431162 -1.829511
##        65        66        67        68        69        70        71        72
## -5.720210 -2.405098  4.943694 -4.191930 -2.791677 -2.140432 -3.214434  1.271953
##        73        74        75        76        77        78        79        80
## -3.150687  4.226450 -0.423825 -5.549963  1.011647 -5.842953 -5.877080  2.625047
```

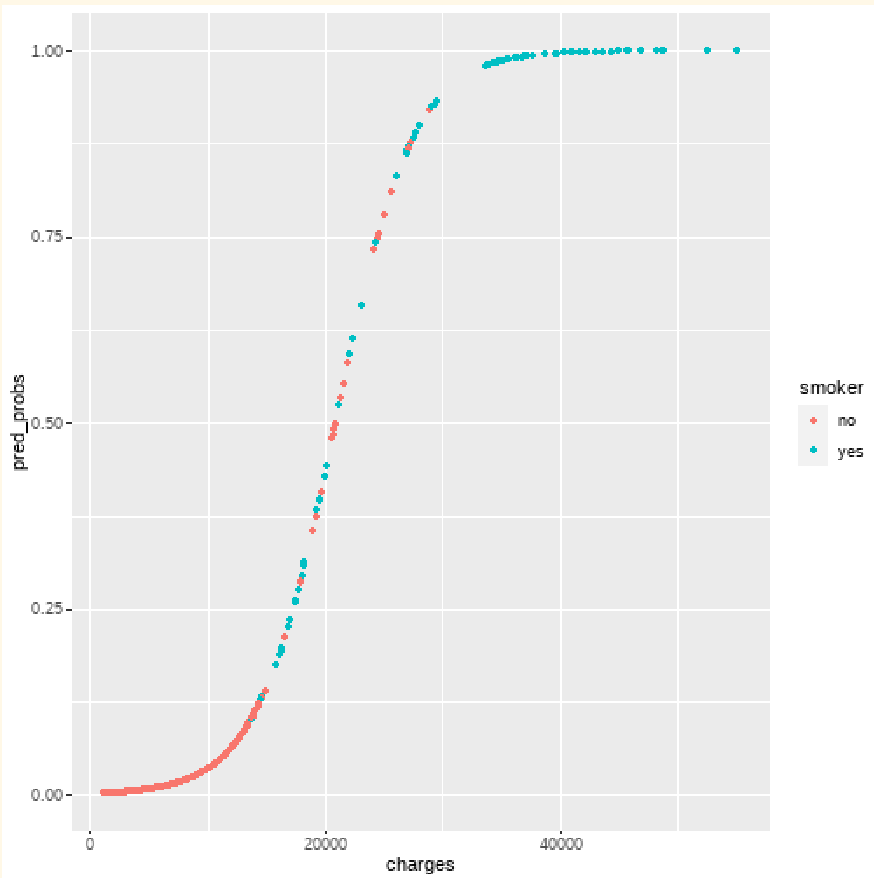
Logistic Regression

Suppose we wanted to see the predicted **probabilities**:

```r
ins_fit %>% predict(ins, type = "prob")
```

```
## # A tibble: 431 x 2
##    .pred_no .pred_yes
##       <dbl>     <dbl>
##  1  0.773     0.227
##  2  0.419     0.581
##  3  0.994     0.00553
##  4  0.995     0.00535
##  5  0.0800    0.920
##  6  0.996     0.00391
##  7  0.109     0.891
##  8  0.952     0.0479
##  9  0.00334   0.997
## 10  0.996     0.00354
## # ... with 421 more rows
```

Logistic Regression



Logistic Regression

How many did we get correct?

```
preds <- ins_fit %>% predict(ins)

ins <- ins %>%
  mutate(
    predicted_smoker = preds$.pred_class
  )

ins %>% count(predicted_smoker, smoker)
```

```
## # A tibble: 4 x 3
##   predicted_smoker smoker     n
##   <fct>            <fct>  <int>
## 1 no               no       333
## 2 no               yes       24
## 3 yes              no        11
## 4 yes              yes       63
```

Logistic Regression

What percentage did we get correct?

```
ins %>%
  mutate(
    correct = (predicted_smoker == smoker)
  ) %>%
  count(correct) %>%
  mutate(
    pct = n/sum(n)
  )
```

```
## # A tibble: 2 x 3
##   correct     n    pct
##   <lgl>   <int>  <dbl>
## 1 FALSE      35 0.0812
## 2 TRUE      396 0.919
```

Logistic Regression

What percentage did we get correct?

```
ins %>%
  accuracy(truth = smoker,
           estimate = predicted_smoker)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.919
```

Questions to ponder

- What if we have a categorical variable where 99% of our values are Category A?
- What if we have a categorical variable with more than 2 categories?
- What if we want to do a transformation besides logistic?
- Are there other ways to do classification besides these **logistic regression** and **KNN**?

Try it!

Open Activity-Classification.Rmd again

Select the best logistic regression model for predicting smoker status

Report the cross-validated metrics - how do they compare to KNN?