Chapter 4: Classification

The linear model in Ch. 3 assumes the response variable Y is quantitiative. But in many situations, the response is categorical.

e.g. eye color
cancer diagnosis
which movie I will watch next

In this chapter we will look at approaches for predicting categorical responses, a process known as *classification*.

Classification problems occur often, perhaps even more so than regression problems. Some examples include

1. A person arrives at emergency room with a set of symptoms that could be attributed to one of three medical conditions. Which of the three conditions does the person have?

2. An online banking service must be able to determine whether a transaction is fraudulent on the basis of user's IP address, past transaction history, etc.

3. Something is in the street in front of the self-driving car you are riding in. Is it a human or another car?

"fit a model"

As with regression, in the classification setting we have a set of training observations $(x_1, y_1), \dots, (x_n, y_n)$ that we can use to "build a classifier". We want our classifier to perform well on the training data and also on data not used to fit the model (**test data**).

most importantly.

We will use the `Default` data set in the `ISLR` package for illustrative purposes. We are interested in predicting whether a person will default on their credit card payment on the basis of annual income and credit card balance.

Yes or no ⇒ categorical.

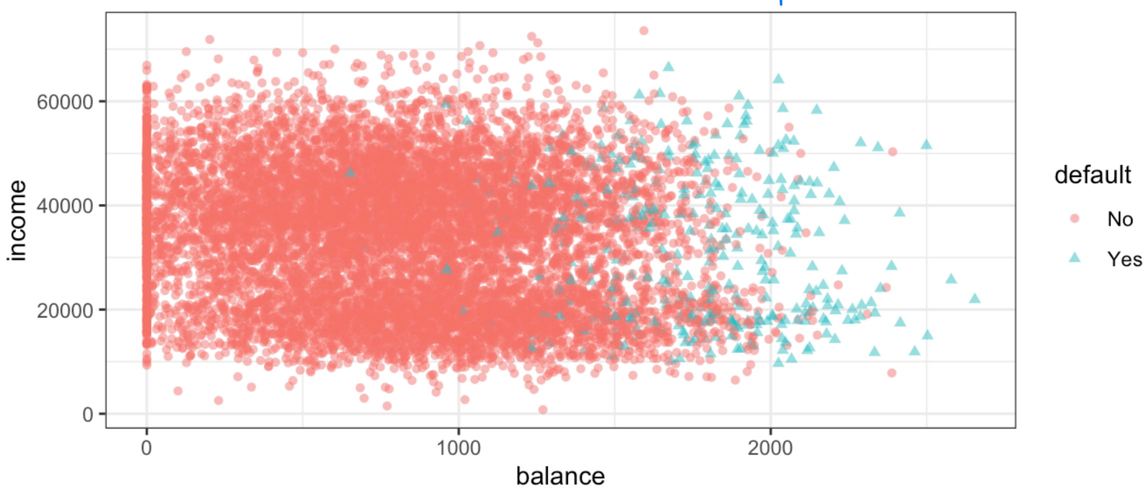
head (Default)

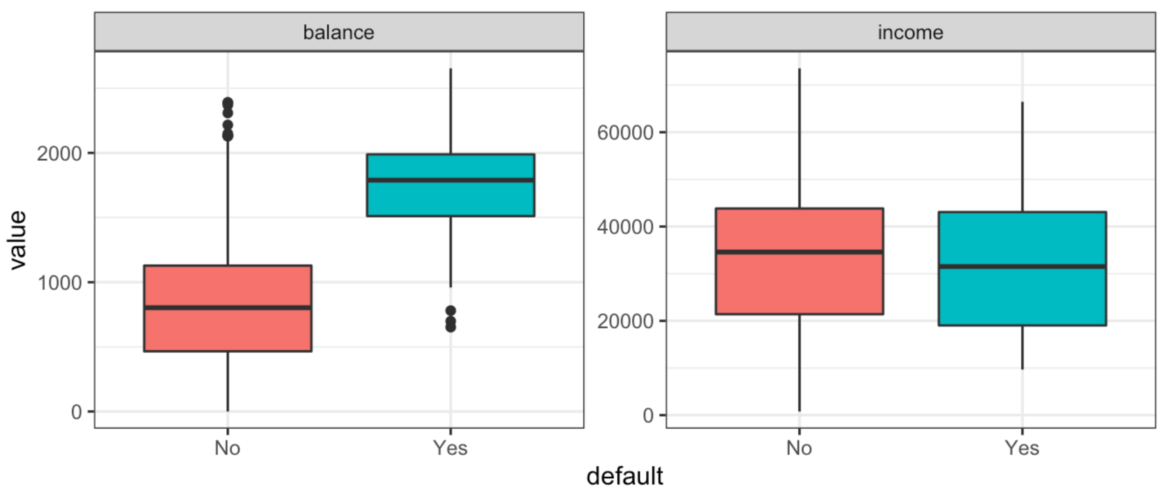
y

features

```
##   default student   balance    income
## 1      No      No  729.5265 44361.625
## 2      No     Yes  817.1804 12106.135
## 3      No      No 1073.5492 31767.139
## 4      No      No  529.2506 35704.494
## 5      No      No  785.6559 38463.496
## 6      No     Yes  919.5885  7491.559
```

decent separation based on balance





pronounced relationship between balance and default

in a lot of cases, relationship is not so pronounced ⇒ classification will be harder.

1 Why not Linear Regression?

I have said that linear regression is not appropriate in the case of a categorical response. Why not?

Let's try it anyways. We could consider encoding the values of `default` in a quantitative repsonse variable Y

$$Y = \begin{cases} 1 & \text{if } \texttt{default} \\ 0 & \text{otherwise} \end{cases}$$

Using this coding, we could then fit a linear regression model to predict Y on the basis of `income` and `balance`. This implies an ① ordering on the outcome, not defaulting comes first before defaulting and ② insists the difference between these two outcomes is 1 unit. In practice, there is no reason for this to be true.

we could let $Y = \begin{cases} 0 & \text{if default} \\ 1 & \text{otherwise} \end{cases}$

OR $Y = \begin{cases} 1 & \text{if default} \\ 10 & \text{otherwise.} \end{cases}$ there is no natural reason why 0/1 encoding, but it has an advantage:

Using the 0/1 dummy encoding, we can get a rough estimate of $P(\texttt{default}|X)$, but it is not guaranteed to be scaled correctly.

doesn't have to be between 0 and 1

but will provide an ordering.

Real problem: this cannot be easily extended to more than 2 classes.

We can instead use methods specifically formulated for categorical responses.

2 Logistic Regression

Let's consider again the `default` variable which takes values `Yes` or `No`. Rather than modeling the response directly, logistic regression models the *probability* that Y belongs to a particular category.

e.g. $P(\text{default} = \text{Yes} \mid \text{balance})$

we can abbreviate this as $p(\text{balance}) \in [0, 1]$.

For any given value of `balance`, a prediction can be made for `default`.

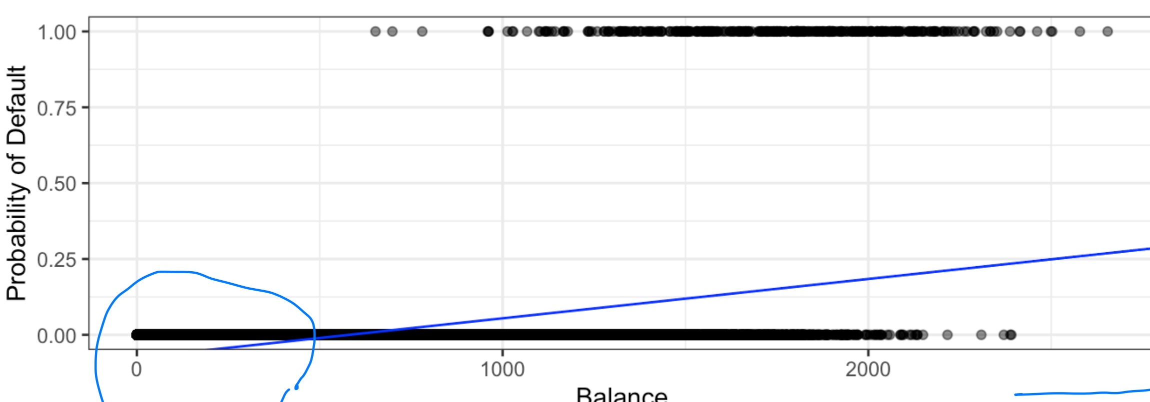
e.g. predict default = Yes if $p(\text{balance}) > 0.5$

or the company could be more conservative predict default = Yes if $p(\text{balance}) > 0.1$ threshold.

2.1 The Model

How should we model the relationship between $p(X) = P(Y = 1|X)$ and X? We could use a linear regression model to represent those probabilities

$$p(X) = \beta_0 + \beta_1 X$$



problem:
for balances close to zero we predict negative probability of default

If we had a large balance would have predicted prob. > 1 of defaulting

neither of these make any sense.

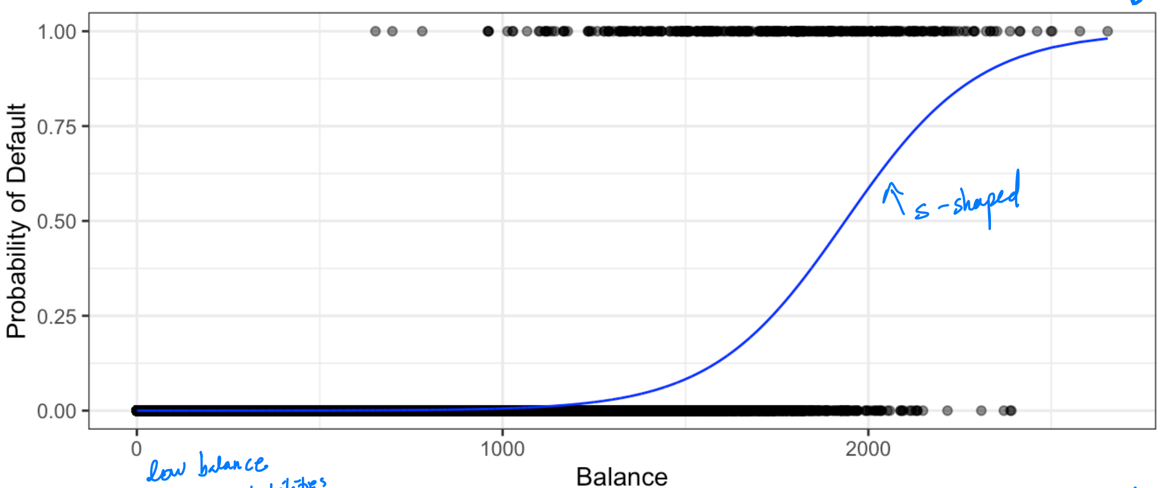
standard logistic function
$f(x) = \frac{e^x}{1+e^x}$

To avoid this, we must model $p(X)$ using a function that gives outputs between 0 and 1 for all values of X. Many functions meet this description, but in *logistic* regression, we use the *logistic* function,

$$p(x) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

we will use maximum likelihood to estimate β's (later).

will never be above 1



s-shaped

low balance predict probabilities close to zero, but never below.

will always get a sensible prediction.

After a bit of manipulation,

$$\underbrace{\frac{p(x)}{1-p(x)}}_{\text{"odds"}} = \frac{e^{\beta_0+\beta_1 X} \Big/ \cancel{1+e^{\beta_0+\beta_1 X}}}{1 \Big/ \cancel{1+e^{\beta_0+\beta_1 X}}} = e^{\beta_0+\beta_1 X}$$

"odds" $\rightarrow$ can take any value between 0 and ∞

low prob of default

high prob. of default.

eg. $p(x) = 0.2 \Rightarrow \text{odds} = \frac{0.2}{1-0.2} = \frac{1}{4}$ "one in 5 people default"

By taking the logarithm of both sides we see,

$$\underbrace{\log\left(\frac{p(X)}{1-p(X)}\right)}_{\text{"log-odds" "logit"}} = \beta_0 + \beta_1 X$$

logit is linear in X

Recall from Ch. 3 that β_1 gives the "average change in Y associated with a one unit increase in X." In contrast, in a logistic model,

increasing X by one unit corresponds to changing the log-odds by β_1

$\Longleftrightarrow$

Increasing X by one unit corresponds to multiplying the odds by e^{β_1}

However, because the relationship between $p(X)$ and X is not linear, β_1 does **not** correspond to the change in $p(X)$ associated with a one unit increase in X. The amount that $p(X)$ changes due to a 1 unit increase in X depends on the current value of X.

Regardless of the value of X,

if β_1 is positive $\Rightarrow$ increasing X increases $p(X)$

if β_1 is negative $\Rightarrow$ increasing X decreases $p(X)$.

2.2 Estimating the Coefficients

The coefficients β_0 and β_1 are unknown and must be estimated based on the available training data. To find estimates, we will use the method of *maximum likelihood*.

The basic intuition is that we seek estimates for β_0 and β_1 such that the predicted probability $\hat{p}(x_i)$ of default for each individual corresponds as closely as possible to the individual's observed default status.

in fact, least squares gives the same answer as MLE w/ our assumptions from before.

to do this, use the likelihood function $\ell(\beta_0, \beta_1) = \prod_{i: y_i = 1} p(x_i) \prod_{i': y_{i'} = 0} (1 - p(x_{i'}))$

$\hat{\beta}_0$ and $\hat{\beta}_1$ chosen to maximize $\ell(\beta_0, \beta_1)$

```
logistic_spec <- logistic_reg()

logistic_fit <- logistic_spec |>
  fit(default ~ balance, family = "binomial", data = Default)

logistic_fit |>
  pluck("fit") |>
  summary()
```

← model specification

$y \sim x$

y takes values in $\{0, 1\}$.

training data

```
##
## Call:
## stats::glm(formula = default ~ balance, family = stats::binomial,
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2697  -0.1465  -0.0589  -0.0221   3.7589
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.065e+01  3.612e-01  -29.49   <2e-16 ***
## balance      5.499e-03  2.204e-04   24.95   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1596.5  on 9998  degrees of freedom
## AIC: 1600.5
##
## Number of Fisher Scoring iterations: 8
```

$\hat{\beta}_0$

$\hat{\beta}_1$

accuracy of estimates

Hypothesis test $H_0: \beta_i = 0$, $H_a: \beta_i \neq 0$ for $i = 0, 1$

for $i = 1$ implies $p(x) = \frac{e^{\beta_0}}{1 + e^{\beta_0}}$ ⇒ doesn't depend on X ⇒ there is no significant relationship.

$p < .05$ ⇒ reject H_0 ⇒ there is a significant relationship btw/ balance & default.

$\hat{\beta}_1 = 0.0055$ ⇒ increase in balance is associated w/ an increase in prob. of default.

A \$1 increase in balance is associated w/ an expected increase in log-odds of default by 0.0055 units.

multiplicative increase in odds of default by $e^{0.0055}$

2.3 Predictions

Once the coefficients have been estimated, it is a simple matter to compute the probability of `default` for any given credit card balance. For example, we predict that the default probability for an individual with `balance` of $1,000 is

$$\hat{p}(x) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 x}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 x}}$$

$$\hat{p}(1000) = \frac{e^{-10.6513 + 0.0055 \times 1000}}{1 + e^{-10.6513 + 0.0055 \times 1000}} = 0.00576$$

In contrast, the predicted probability of default for an individual with a balance of $2,000 is

$$\hat{p}(2000) = \frac{e^{-10.6513 + 0.0055 \times 2000}}{1 + e^{-10.6513 + 0.0055 \times 2000}} \approx 0.586$$

58.6% > 50% ⇒ might predict default = YES if threshold = 0.5.

in R:
predict function
augment function

2.4 Multiple Logistic Regression

We now consider the problem of predicting a binary response using multiple predictors. By analogy with the extension from simple to multiple linear regression,

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p$$

$$\Downarrow$$

$$p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \dots + \beta_p X_p}}$$

Just as before, we can use maximum likelihood to estimate $\beta_0, \beta_1, \dots, \beta_p$.

```
logistic_fit2 <- logistic_spec |>
  fit(default ~ ., family = "binomial", data = Default)

logistic_fit2 |>
  pluck("fit") |>
  summary()
```

$Y \sim$ every other column in data frame.

$Y \sim X_1 + X_2 + \dots + X_p$

$\hat{\beta}$ $se(\hat{\beta})$ $H_0: \beta_i = 0$ $H_a: \beta_i \neq 0$

dummy variable.

```
##
## Call:
## stats::glm(formula = default ~ ., family = stats::binomial, data =
data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4691  -0.1418  -0.0557  -0.0203   3.7383
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.087e+01  4.923e-01 -22.080  < 2e-16 ***
## studentYes  -6.468e-01  2.363e-01  -2.738  0.00619 **
## balance      5.737e-03  2.319e-04  24.738  < 2e-16 ***
## income       3.033e-06  8.203e-06   0.370  0.71152
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1571.5  on 9996  degrees of freedom
## AIC: 1579.5
##
## Number of Fisher Scoring iterations: 8
```

no significant relationship btw/ income & default

$\hat{\beta}_{student(Yes)} < 0 \Rightarrow$ if you are a student, LESS likely to default holding balance & income constant.

student confounded w/ balance (if you are a student you more likely to have higher balance)

but if you have the same balance (and income) as non-student, less likely to default.

By substituting estimates for the regression coefficients from the model summary, we can make predictions. For example, a student with a credit card balance of $1,500 and an income of $40,000 has an estimated probability of default of

$$\hat{p}(x) = \frac{e^{-10.869 + 0.00574 \times 1500 + .000003 \times 40000 - 0.6468 \times 1}}{1 + e^{-10.869 + 0.00574 \times 1500 + .000003 \times 40000 - 0.6468 \times 1}}$$

$$= 0.058$$

A non-student with the same balance and income has an estimated probability of default of

$$\hat{p}(x) = \frac{e^{-10.869 + 0.00574 \times 1500 + 0.000003 \times 40000 - 0.\cancel{6468} \times 0}}{1 + e^{-10.869 + 0.00574 \times 1500 + 0.000003 \times 40000 - 0.\cancel{6468} \times 0}}$$

$$= 0.105$$

predict/augment w/ new.data

balance	income	student
1500	40000	Yes
1500	40000	No

2.5 Logistic Regression for > 2 Classes

We sometimes which to classify a response variable that has more than two classes. There are multi-class extensions to logistic regression ("multinomial regression"), but there are far more popular methods of performing this.

3 LDA "linear discriminant analysis"

Logistic regression involves ~~direction~~ directly modeling $P(Y = k|X = x)$ using the logistic function for the case of two response classes. We now consider a less direct approach.

Idea:

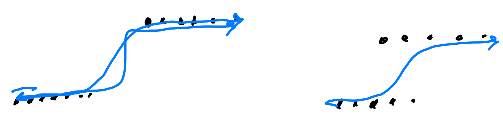
Model the distribution of the predictors separately in each response class (given Y) and then use Bayes theorem to flip these and get estimates $P(Y=k|X=x)$

$\hookrightarrow P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

Why do we need another method when we have logistic regression?

1. We might have more than 2 response classes.

2. When the classes are well-separated, the parameter estimates for logistic regression are suprisingly unstable.



3. If n is small and the distributions of the predictors X approximately $\sim N$ in each class, LDA is more stable than logistic regression.

3.1 Bayes' Theorem for Classification

Suppose we wish to classify an observation into one of K classes, where $K \geq 2$.

Categorical Y can take K possible distinct and unordered values.

Notation

π_k = overall probability that a randomly chosen observation comes from k^{th} class.

"prior probability"

$f_k(x)$ $= P(X = x \mid Y = k)$ (discrete X)

= "probability that X falls in a small region around x given $Y = k$ (continuous X).

"density function" of X for an observation that comes from class k

"likelihood"

$$\overset{A}{P(Y = k}|\overset{B}{X = x)} = \frac{\overset{P(X=x|Y=k)\ P(Y=k)}{f_k(x)\,\pi_k}}{\underset{P(X=x)}{\sum_{\ell=1}^{K} f_\ell(x)\,\pi_\ell}}$$

(Bayes theorem).

We will use the same abbreviation as before $p_k(x)$ ← "posterior probability" that an obs w/ $X = x$ comes from class k.

In general, estimating π_k is easy if we have a random sample of Y's from the population.

Compute the fraction of training observations that come from k^{th} class.

Estimating $f_k(x)$ is more difficult unless we assume some particular forms.

If we can estimate $f_k(x)$ we can develop a classifier that is close to the "best" classifier (more later).

3.2 p = 1

Let's (for now) assume we only have 1 predictor. We would like to obtain an estimate for $f_k(x)$ that we can plug into our formula to estimate $p_k(x)$. We will then classify an observation to the class for which $\hat{p}_k(x)$ is greatest.

$p=1$

$\frac{\pi_k f_k(x)}{\sum_{j=1}^K \pi_j f_j(x)}$

estimating the Bayes classifier!

assignment to class w/ highest $p_k(x)$ is called "Bayes classifier" and is known to be optimal, i.e. we can do no better.

Suppose we assume that $f_k(x)$ is normal. In the one-dimensional setting, the normal density takes the form

Gaussian

$$f_k(x) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2\right)$$

μ_k and σ_k^2 mean and variance parameters for k^{th} class.

Let's also (for now) assume $\sigma_1^2 = \dots = \sigma_K^2 = \sigma^2$. (shared variance term).

Plugging this into our formula to estimate $p_k(x)$,

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)}{\sum_{\ell=1}^K \pi_\ell \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_\ell)^2\right)}$$

actually $\pi = 3.14159\dots$

prior prob that observation falls into ℓ^{th} class.

Bayes classifier

We then assign an observation $X = x$ to the class which makes $p_k(x)$ tthe largest. This is equivalent to

(log + rearranging)

assign obs. to class for which

$$\delta_k(x) = x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

is maximized.

Example 3.1 Let $K = 2$ and $\pi_1 = \pi_2$. When does the Bayes classifier assign an observation to class 1?

When $\delta_1(x) > \delta_2(x)$

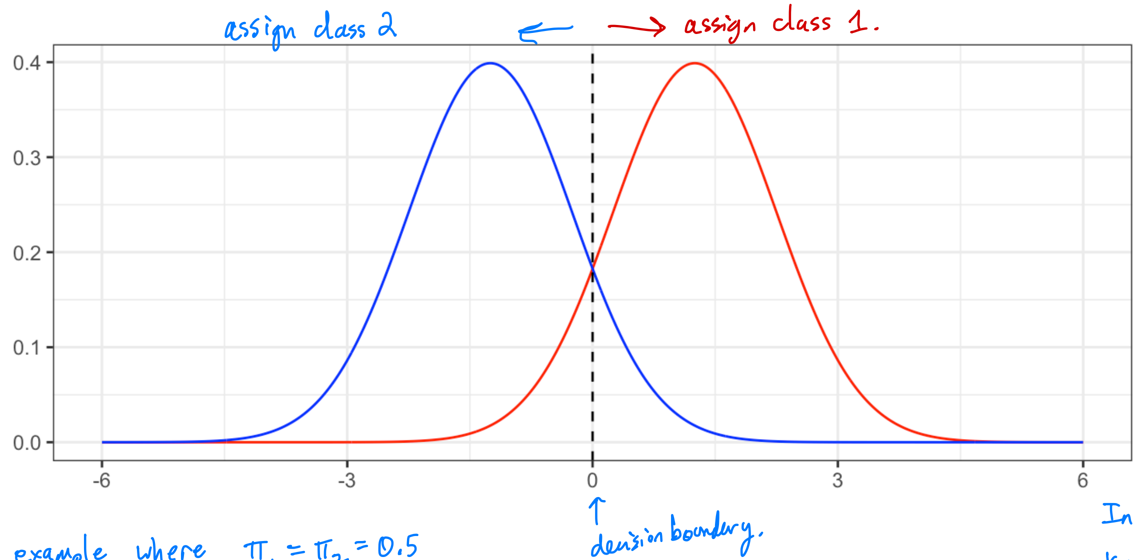
$$\Leftrightarrow \quad x\frac{\mu_1}{\cancel{\sigma^2}} - \frac{\mu_1^2}{2\cancel{\sigma^2}} + \cancel{\log(\pi_1)} > x\frac{\mu_2}{\cancel{\sigma^2}} - \frac{\mu_2^2}{2\cancel{\sigma^2}} + \cancel{\log(\pi_2)}$$

$$x\mu_1 - \frac{\mu_1^2}{2} > x\mu_2 - \frac{\mu_2^2}{2}$$

$$2x(\mu_1 - \mu_2) > \mu_1^2 - \mu_2^2$$

$$x > \frac{\mu_1 + \mu_2}{2}$$

decision boundary.



example where $\pi_1 = \pi_2 = 0.5$

$\sigma^2 = 1$

$\mu_1 = 1.25$

$\mu_2 = -1.25$ $\Rightarrow$ decision boundary is $\dfrac{1.25 - 1.25}{2} = 0.$

In this case, we know $f_k(x) \sim N(\mu_k, \sigma^2)$ $\Rightarrow$ we can create the Bayes classifier!

In practice, even if we are certain of our assumption that X is drawn from a Gaussian distribution within each class, we still have to estimate the parameters $\mu_1, \ldots, \mu_K, \pi_1, \ldots, \pi_K, \sigma^2$. to estimate the Bayes classifier.

The *linear discriminant analysis* (LDA) method approximated the Bayes classifier by plugging estimates in for π_k, μ_k, σ^2.

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i \quad \leftarrow \text{average of training obs in } k^{th} \text{ class}$$

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2 \quad \leftarrow \text{weighted average of class variances.}$$

n = # training obs.

n_k = # training obs in class k

from scientific knowledge or data sampling scheme.

Sometimes we have knowledge of class membership probabilities $\pi_1, \ldots, \pi_K$ that can be used directly. If we do not, LDA estimates π_k using the proportion of training observations that belong to the kth class.

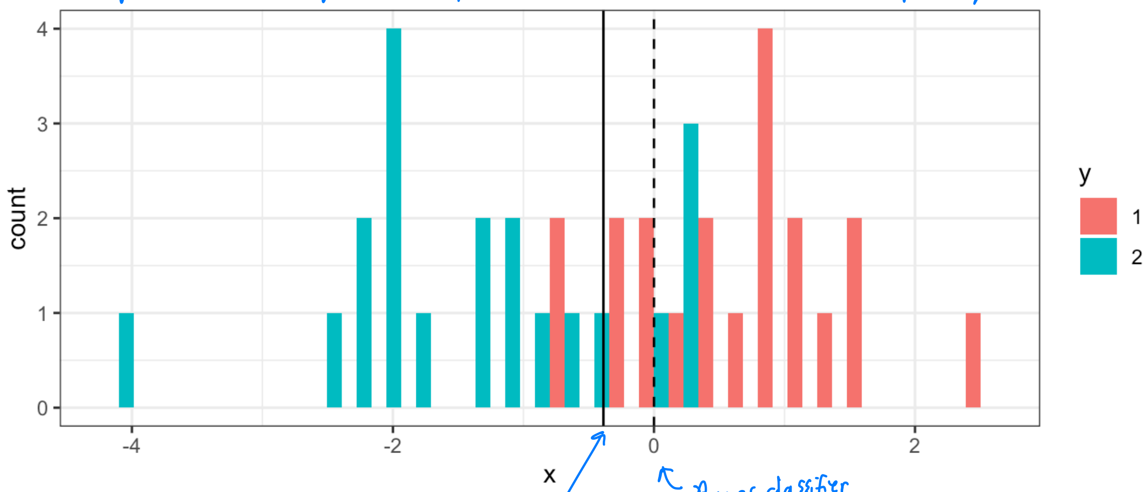
$$\hat{\pi}_k = \frac{n_k}{n}$$

The LDA classifier assignes an observation $X = x$ to the class with the highest value of

$$\hat{\delta}_k(x) = x \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k).$$

histogram of randomly sampled points from class 1 and class 2 from prev. plot.

$n_1 = n_2 = 20$



LDA boundary (based on data) $\frac{\hat{\mu}_1 + \hat{\mu}_2}{2}$

Bayes classifier boundary

```
##     pred
## y       1     2
##   1 18966  1034
##   2  3855 16145
```

pred ← prediction based on LDA boundary.

y ↓ true value

1034 ← got wrong.

16145 got right

← "confusion matrix"

simulated many test points (20k from each class).

The LDA test error rate is approximately 12.22% while the Bayes classifier error rate is approximately 10.52%.

"$\frac{\text{\# got wrong}}{\text{\# test data points}} = \frac{1034 + 3855}{40000} = 12.22\%$

The Bayes error rate is the best we can possibly do!
(we can only estimate it because this is a simulated example).

The LDA approach did almost as well!

The LDA classifier results from assuming that the (1) observations within each class come from a normal distribution with a (2) class-specific mean vector and a (3) common variance σ^2 and plugging estimates for these parameters into the Bayes classifier.

we will relax this later.

3.3 p > 1

We now extend the LDA classifier to the case of multiple predictors. We will assume

$\underline{X} = (X_1, \dots, X_p)$ drawn from a multivariate Gaussian dsn w/ class specific mean vector & common covariance matrix

↳ each individual component follows Gaussian and some covariance between components.

$$\sim N_p(\overset{p\times 1 \text{ vector}}{\underline{\mu}}, \overset{p\times p \text{ matrix}}{\Sigma}) \Rightarrow E\underline{X} = \underline{\mu}$$

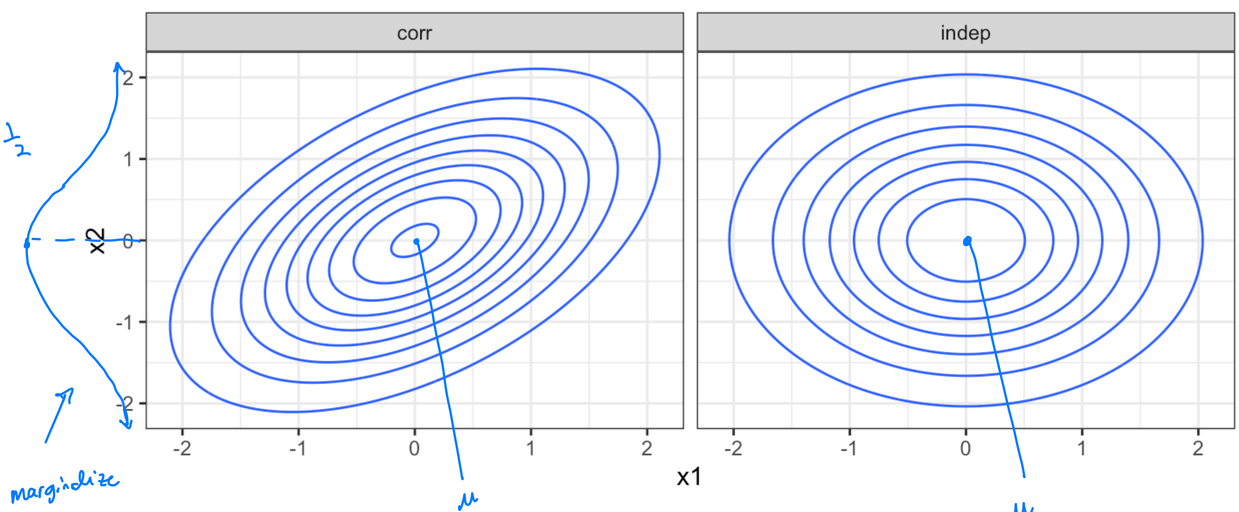
$$cov(\underline{X}) = \Sigma$$

Formally the multivariate Gaussian density is defined as

$$f(\underline{x}) = \frac{1}{(2\pi)^{p/2}\underset{\text{“trace” = sum of diag. elements.}}{|\Sigma|^{1/2}}} \exp\left(-\frac{1}{2}(\underline{x}-\underline{\mu})^{\overset{\text{transpose}}{T}} \underset{\text{matrix inverse}}{\Sigma^{-1}}(\underline{x}-\underline{\mu})\right)$$

$cov(X_1, X_2) = \frac{1}{2}$

results in ovals.



$cov(X_1, X_2) = 0$

⇒ independence

results in circles

If you marginalize out X_1 or X_2 ⇒ Gaussian dsn "bell shape"

$p = 2$ Gaussian density w/ $\mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ and 2 Σs.

In the case of $p > 1$ predictors, the LDA classifier assumes the observations in the kth class are drawn from a multivariate Gaussian distribution $N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$. common covariance matrix.
class-specific mean

Plugging in the density function for the kth class, results in a Bayes classifier
assign an observation $\underset{\sim}{X} = \underset{\sim}{x}$ the class for which

discriminant function → $\delta_k(\underset{\sim}{x}) = \underset{\sim}{x}^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$ is maximized.

linear in $\underset{\sim}{x}$ (hence the name LDA)

Once again, we need to estimate the unknown parameters $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K, \pi_1, \dots, \pi_K, \boldsymbol{\Sigma}$.

Use similar ideas as p=1. to estimate.

To classify a new value $X = x$, LDA plugs in estimates into $\delta_k(\underset{\sim}{x})$ and chooses the class which maximized this value.

⇒ $\hat{\delta}_k(\underset{\sim}{x})$ and choose k which maximizes it (i.e. estimating Bayes classifier)

Let's perform LDA on the `Default` data set to predict if an individual will default on their CC payment based on balance and student status.

model specification →

```
lda_spec <- discrim_linear(engine = "MASS")

lda_fit <- lda_spec |>
  fit(default ~ student + balance, data = Default)

lda_fit |>
  pluck("fit")
```

specify formula for y ~ x's
Same as linear, logistic regression.

look at the fit.

no good "summary" function for LDA →

```
## Call:
## lda(default ~ student + balance, data = data)
##
## Prior probabilities of groups:
##      No    Yes
## 0.9667 0.0333
##
## Group means:
##      studentYes   balance
## No    0.2914037  803.9438
## Yes   0.3813814 1747.8217
##
## Coefficients of linear discriminants:
##                       LD1
## studentYes -0.249059498
## balance     0.002244397
```

estimates of π_k based on class membership in training data.

$\hat{\mu}_k$ = average of each predictor w/in each class from training data

linear combinations of student and balance used to form the LDA decision rule.

```r
# training data confusion matrix
lda_fit |>
  augment(new_data = Default) |>
  conf_mat(truth = default, estimate = .pred_class)
```

gets predictions on new_data

confusion matrix

column name of predictions results from augment().

```
##            Truth
## Prediction   No  Yes
##        No  9644  252
##        Yes   23   81
```

overall training error rate = 2.75%

For Default = Yes, only got $\frac{81}{252+81}$ = 24% right!

Why does the LDA classifier do such a poor job of classifying the customers who default?

Only 3.33% of individuals in training data set defaulted.

A simple (but useless) classifier could just predict default = No and only get 3.33% wrong!

LDA is trying to approximate the Bayes classifier ⇒ yield smallest possible overall error rate.

A CC company may want misclassifying default = Yes people, can adjust how we classify.

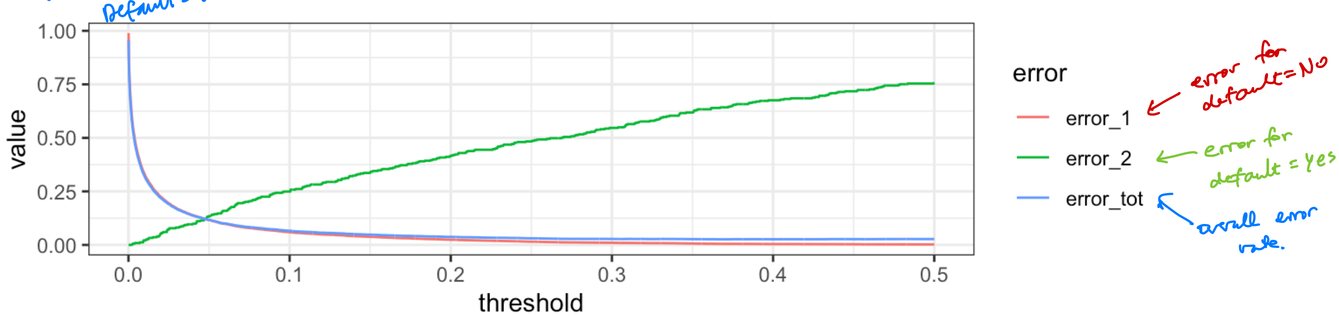
```r
lda_fit |>
  augment(new_data = Default) |>
  mutate(pred_lower_cutoff = factor(ifelse(.pred_Yes > 0.2, "Yes",
         "No"))) |>
  conf_mat(truth = default, estimate = pred_lower_cutoff)
```

new threshold.

```
##            Truth
## Prediction   No  Yes
##        No  9432  138
##        Yes  235  195
```

do worse w/ Default = No.

do better at Default = Yes



as threshold ↓, error for default = NO ↓

error for default = YES ↑

How to choose the threshold?

Domain knowledge

or pick 0.5 because theoretical justification

or another option next chapter...

3.4 QDA

LDA assumes that the observations within each class are drawn from a multivariate Gaussian distribution with a class-specific mean vector and a common covariance matrix across all K classes.

Quadratic Discriminant Analysis (QDA) also assumes the observations within each class are drawn from a multivariate Gaussian distribution with a class-specific mean vector but now each class has its own covariance matrix.

an obs. from k^{th} class $\underline{X} \sim N(\underline{\mu}_k, \Sigma_k)$

Under this assumption, the Bayes classifier assigns observation $X = x$ to class k for whichever k maximizes

$$\delta_k(\underline{x}) = -\frac{1}{2}(\underline{x}-\underline{\mu}_k)^T \Sigma_k^{-1} (\underline{x}-\underline{\mu}_k) - \frac{1}{2}\log|\Sigma_k| + \log \pi_k$$

$$= \underbrace{-\frac{1}{2} x^T \Sigma_k^{-1} x}_{} + x^T \Sigma_k^{-1} \mu_k - \frac{1}{2}\mu_k^T \Sigma_k^{-1} \mu_k - \frac{1}{2}\log|\Sigma_k| + \log \pi_k$$

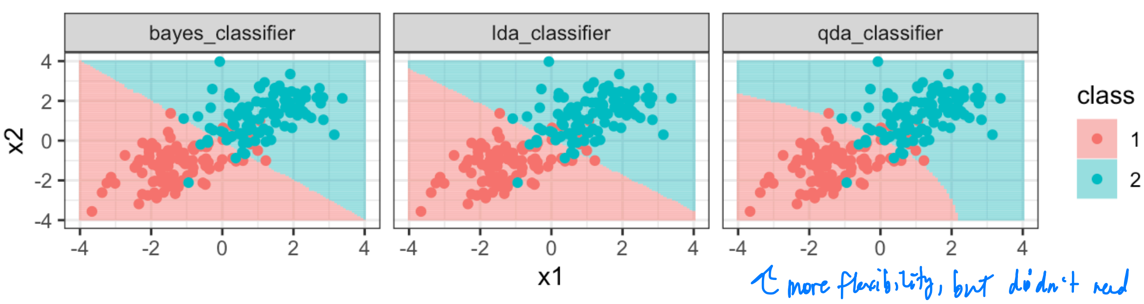
← quadratic in x ⇒ QDA

plugin estimates for μ_k, Σ_k, π_k and choose $\underset{k}{\operatorname{argmax}}\ \hat{\delta}_k(\underline{x})$.

When would we prefer QDA over LDA?

common covariance of 0.5

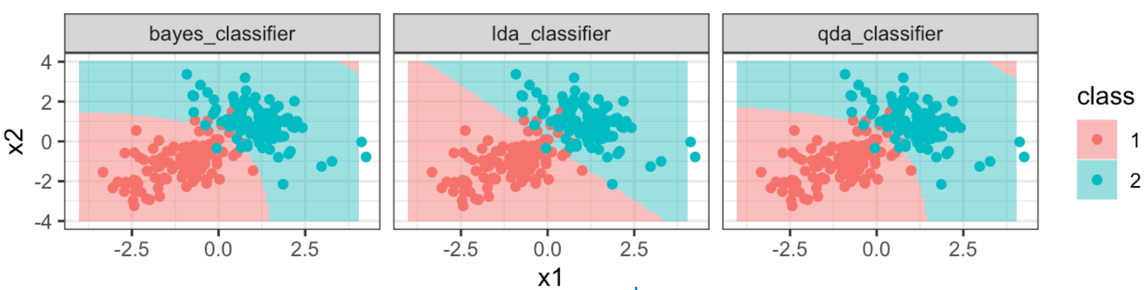
⇒ LDA similar to Bayes classifier



← more flexibility, but didn't need it.

different Σ_k

⇒ QDA similar to Bayes classifier



← might not be flexible enough.

When there are p predictors, estimating Σ_k requires estimating $\frac{p(p+1)}{2}$ parameters ⇒ $K\frac{p(p+1)}{2}$ parameters for QDA.

LDA is linear in x ⇒ only $K \cdot p$ parameters to estimate.

⇒ LDA is much less flexible than QDA, but if the assumption of global variance is bad, then LDA predictions can be bad.

⇒ LDA > QDA if not many training obs.

4 KNN — non parametric approach.

Another method we can use to estimate $P(Y = k|X = x)$ (and thus estimate the Bayes classifier) is through the use of K-nearest neighbors.

The KNN classifier first identifies the K points in the training data that are closest to the test data point $X = x$, called $\mathcal{N}(x)$.

neighborhood

Then we can estimate $P(Y=k|\underline{X}=\underline{x})$ as

class k

$$\hat{P}(Y=k|\underline{X}=\underline{x}) = \frac{1}{K}\sum_{i\in \mathcal{N}(\underline{x})} \mathbb{I}(y_i = k)$$

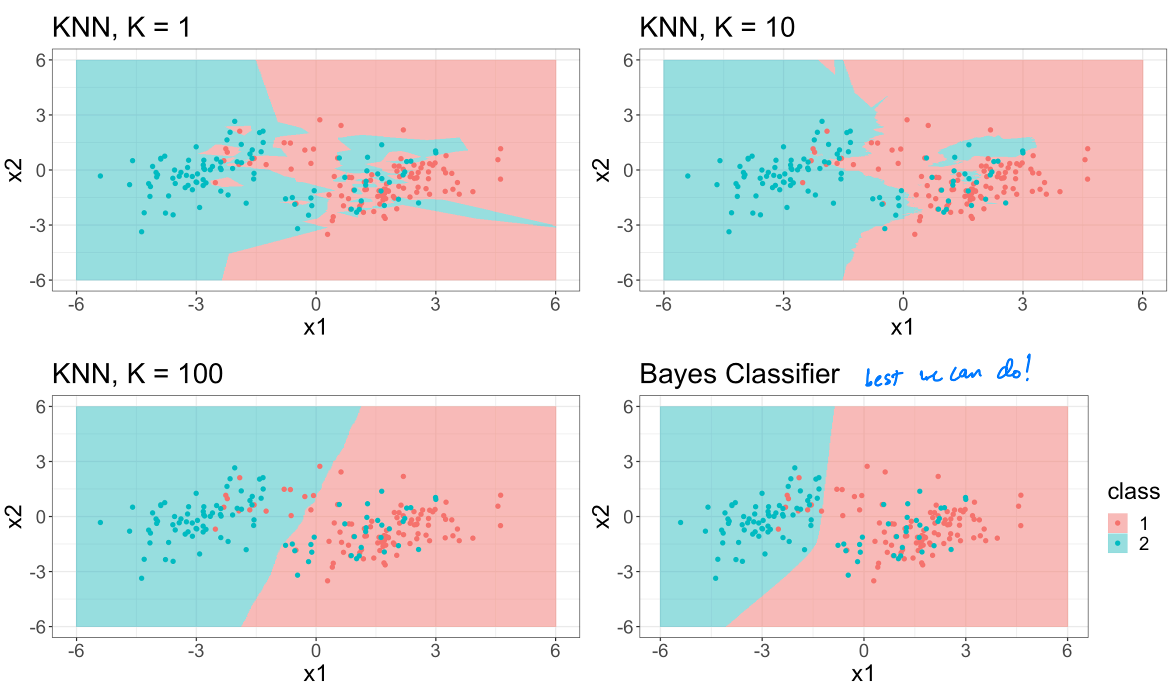
\# points in neighborhood

and then classify $\underline{x}$ to the class k w/ highest $\hat{P}(Y=k|\underline{X}=\underline{x})$.

Just as with regression tasks, the choice of K (neighborhood size) has a drastic effect on the KNN classifier obtained.

overly flexible boundary

less flexible, close to linear boundary.



best we can do!

- Choosing the correct level of flexibility (K) is critical to success of any method (KNN or LDA vs QDA).
- How to choose? Ch. 5 (coming up next).

5 Comparison

LDA vs. Logistic Regression ← closely related!

Consider $K=2$, $p=1$ and $p_1(x)$, $p_2(x) = 1 - p_1(x)$ are probabilities $X=x$ belongs to class 1 or 2, respectively.

logistic regression

linear function of x

$$\log\left(\frac{p_1(x)}{1-p_1(x)}\right) = \underbrace{\beta_0 + \beta_1 x}_{\text{fit using MLE}}$$

LDA

plug in def'n of $p_1(x), p_2(x)$ using LDA and simplified.

$$\log\left(\frac{p_1(x)}{1-p_1(x)}\right) = \log\left(\frac{p_1(x)}{p_2(x)}\right) = \log\left[\frac{\pi_1}{\pi_2}\exp\left(-\frac{1}{2\sigma^2}\left[(x-\mu_1)^2 - (x-\mu_2)^2\right]\right)\right]$$

expanding squared terms and applied log.

linear function of x

$$= \log\pi_1 - \log\pi_2 - \frac{1}{2\sigma^2}\left[\cancel{x^2} - 2x\mu_1 + \mu_1 - \cancel{x^2} + 2x\mu_2 - \mu_2^2\right] = c_0 + c_1 x$$

fit using plugin estimates.

Should give similar results, but LDA assumes Gaussian dsn w/ common variance and logistic does not ⇒ whichever assumption holds will be better (+ consider numerical issues mentioned earlier).

(LDA & Logistic Regression) vs. KNN

KNN non-parametric, no assumptions made about shape of decision boundary.

⇒ should outperform LDA/logistic regression when the decision boundary is highly non-linear.

Needs more data to work well.

KNN → doesn't tell us which predictors are important

QDA

↳ compromise between KNN + linear approaches (LDA, logistic regression).

Quadratic decision boundary ⇒ can accurately model more problems than linear methods.

Not as flexible as KNN ⇒ if not enough training data for KNN, can work better.