Class 5: Classification

BUS 696

Prof. Jonathan Hersh

Class 5: Announcements

1. Will post Problem Set 2 tomorrow,

   Due Oct 9

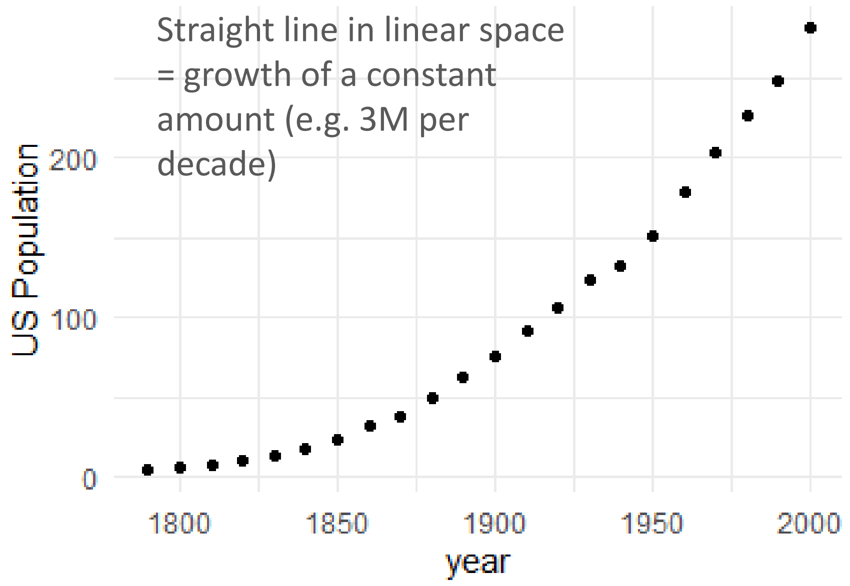
2. Office Hours
   1. TA: Wed: 12-1, Th 5-6
   2. Instructor: M: 4-5, W 5-6

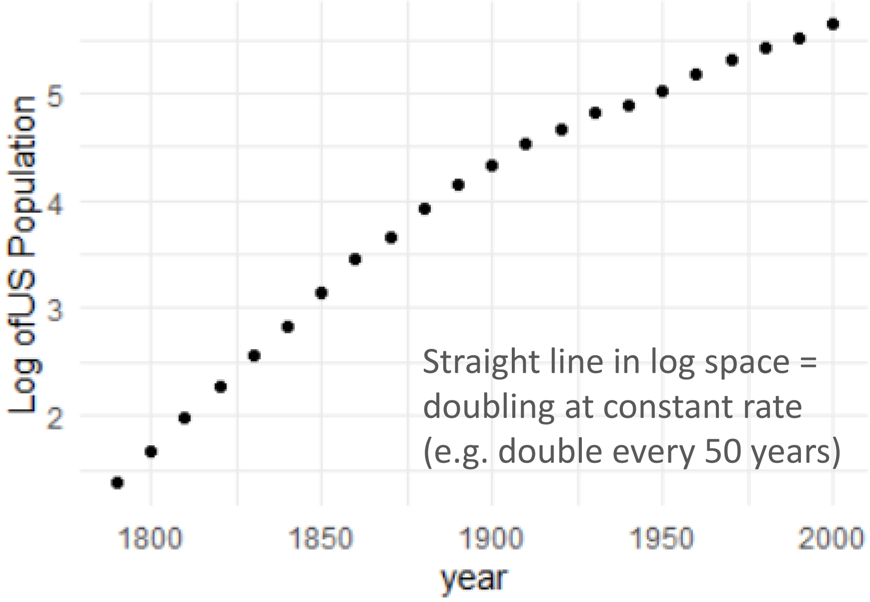
3. Post October 5th In Person

Class 5: Outline

1. Log Transformations and Interpreting log-log Regressions
2. Model Evaluation
   - Testing and Training Sets, RMSE, and R-Squared
3. Why not regression for classification?
4. Logistic Function
5. Log Odds Ratio
6. Estimating Logistic Regressions in R
7. Classification Lab 1
8. False/True Positives False/True Negatives
9. Confusion Matrices
10. ROC Curves and AUC
11. Classification Lab 2

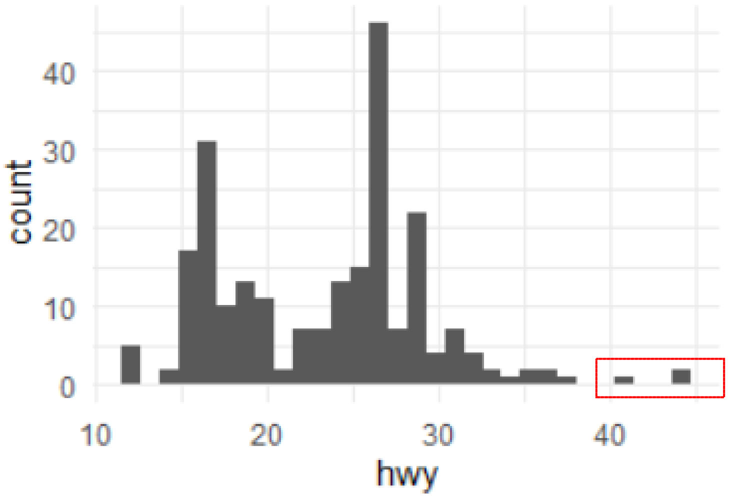
Log Transformations

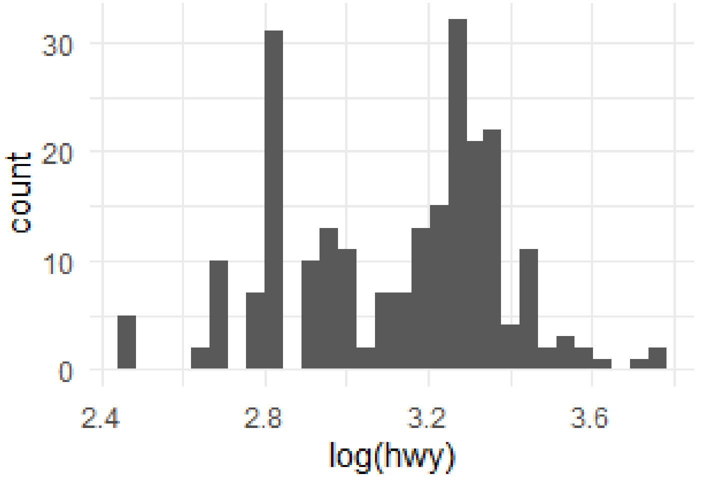


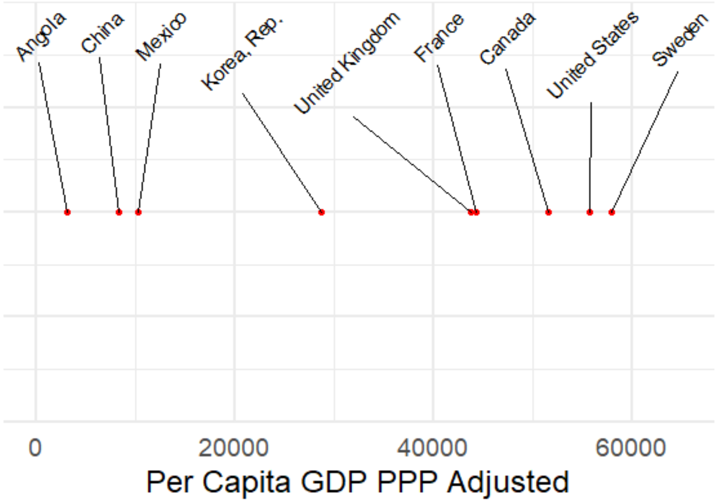


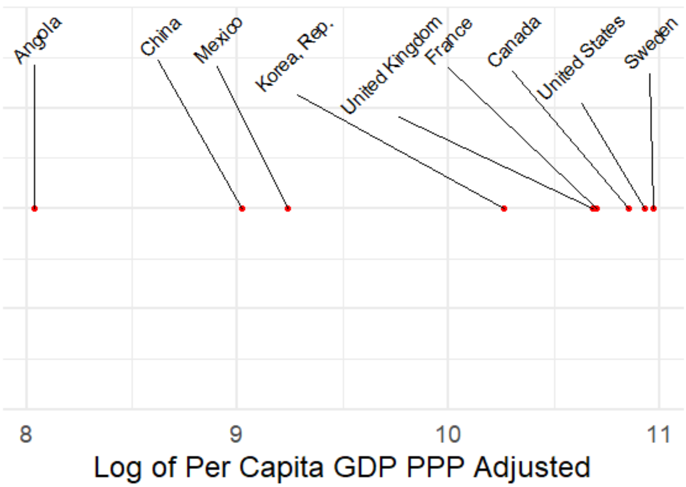
- Recall that log is the inverse of exponentiation
- Logging a number answers the question "what is the exponent I must raise the base of the log to produce this number"
  - $\log_{10} 100 = 2 \rightarrow 10^2 = 100$
  - $\log_{10} 1000 = 3 \rightarrow 10^3 = 1000$
- The natural log is $\log_e x = \ln(x)$ where e=2.718…
- Each unit increase of ln(x) = a doubling of x
- This is a useful data transformation because we often want to incorporate data that increases rapidly in our regression analysis

Why Log Transformations For Regression Data









- Log transforming our data is often very useful if our data is particularly “spread out”
- In particular if there are outlier values this will improve model performance
- Some data like income should usually be logged

Model accuracy often improves when logging dependent variable but interpretation can suffer

Log-Log Regression Model Coefficients = Elasticities!

$$\log(hwy_i) = \beta_0 + \beta_1 \log(displ_i) + \beta_2 year_i + \epsilon_i$$

```
> mod1 <- lm(log(hwy) ~ log(displ) + year,
+            data = mpg)
> summary(mod1)

Call:
lm(formula = log(hwy) ~ log(displ) + year, data = mpg)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.46033 -0.09856 -0.00202  0.08837  0.48681 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -10.874596   4.551216  -2.389  0.01768 *  
log(displ)   -0.560514   0.027052 -20.720  < 2e-16 ***
year          0.007314   0.002274   3.217  0.00148 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1547 on 231 degrees of freedom
Multiple R-squared:  0.6502,    Adjusted R-squared:  0.6471 
F-statistic: 214.7 on 2 and 231 DF,  p-value: < 2.2e-16
```

- Suppose we log our hwy regression model
- How do we now interpret the coefficient $\beta_{\log(displ)}$
  - $y = \exp(\beta_0 + \beta_1 \log(x) + \epsilon)$
  - $\frac{dy}{dx} = \frac{\beta_1}{x} \exp(\beta_0 + \beta_1 \log(x) + \epsilon)$
  - $\Rightarrow \beta_1 = \frac{dy/y}{dx/x}$

Therefore log coefficients can be interpreted as elasticities!

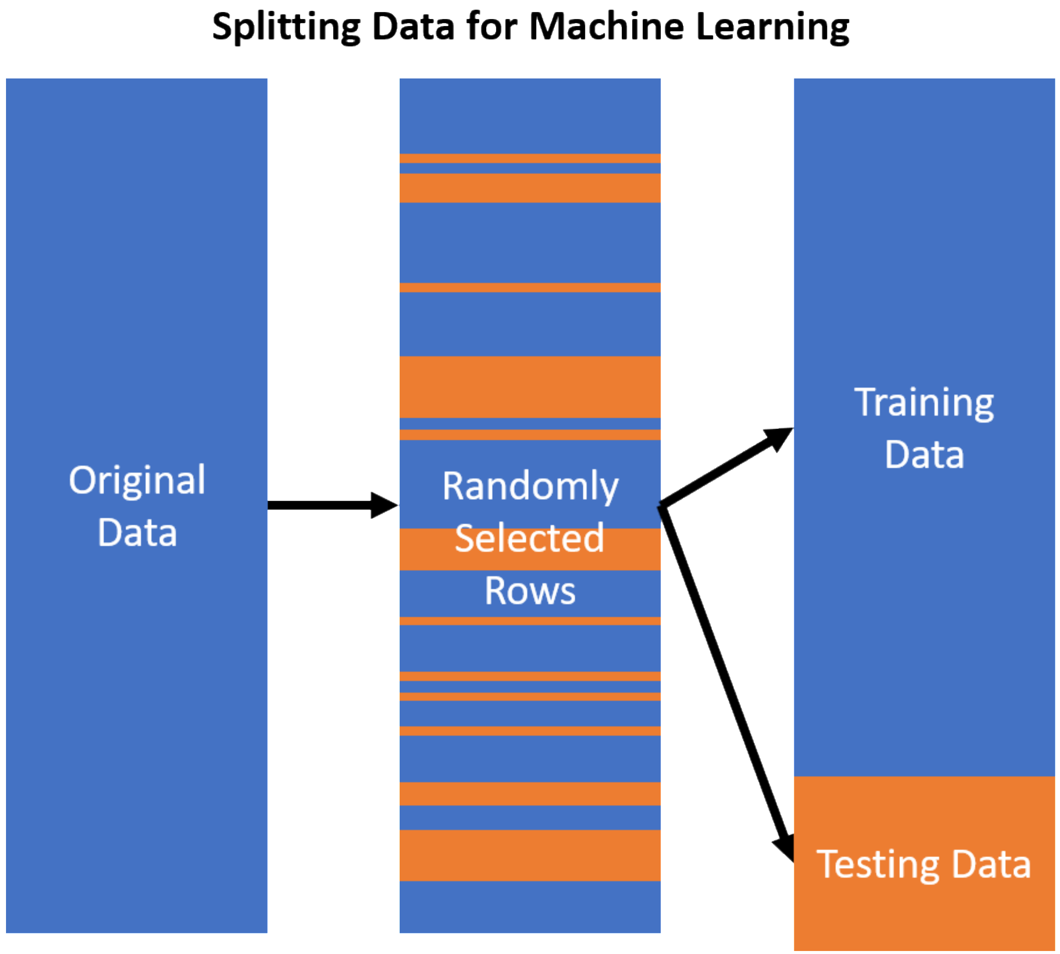
A 1% change in x-variable results in a β_1 % change in the outcome variable

- Here a 1% increase in displacement -> a 0.56% decrease in highway mpg.

Class 9: Outline

1. Log Transformations and Interpreting log-log Regressions
2. **Model Evaluation**
   - Testing and Training Sets, RMSE, and R-Squared
3. Lab Class 9
4. Why Classification Models?
5. Logistic Function
6. Log Odds Ratio

Recall: Training and Testing Sets



Training set: (observation-wise) subset of data used to develop models

Test set: subset of data used to evaluate final model performance

Building Training and Testing Sets in R

```r
#-------------------------------------------------------------
# Testing and Training Sets
#-------------------------------------------------------------

# install rsample if necessary
# install.packages('rsample')
library('rsample')
library('tidyverse')
data(mpg)

# always want to set a seed before doing any randomization
# procedure to make our code reproducible
set.seed(1818)

# set train proportion = % of total data in training set
# common values are 0.9, 0.75, 0.8, 0.5
train_prop <- 0.8
mpg_split <- initial_split(mpg, prop = train_prop)
```

```r
> mpg_train <- training(mpg_split)
> mpg_test <- testing(mpg_split)
> # check the number of rows to ensure training
> # and testing split is correct
> nrow(mpg_train)
[1] 188
> nrow(mpg_test)
[1] 46
>
```

- initial_split() is a helper function to create testing and training sets
- Must specify the data frame to split
- Can also specify the % of data to use for training (defaults to 75%)
- The functions training() and testing() will create separate testing and training sets from the original data set

Always set seed before any randomize procedure to ensure code is reproducible

Generating In-Sample (Training) and Out-of-Sample (Test) Predictions

```r
#-----------------------------------------------------------
# Estimating Models on Training Sets
#-----------------------------------------------------------
# estimate a model using the training set
mod <- lm(hwy ~ year + displ,
          data = mpg_train)

# generate in-sample (training) predictions
preds_train <- predict(mod)
# either method works
preds_train <- predict(mod, newdata = mpg_train)

# generate out-of-sample (test set) predictions
preds_test <- predict(mod, newdata = mpg_test)
```

- Estimate a model on the training set
- Never estimate a model using the test set
- **In-sample predictions** are the predicts using data in the training set
- **Out-of-sample predictions** are the predicts using data in the test set

Quantitative Model Evaluation Using Yardstick

- The package 'yardstick' has several functions to quantitatively evaluate model performance
- We must first compile our model output in a 'results' data frame
- We include the predictions from the model
- True outcome values must exclude any missing values for Xs or Ys
- We must also do this for the test data set

```r
library('yardstick')

# create a
results_train <- data.frame(
  predicted = preds_train,
  actual = mpg_train %>%
    filter(complete.cases(hwy, year, displ)) %>%
    select(hwy),
  type = rep("train", length(preds_train))
) %>%
  rename(`predicted` = 1, `actual` = 2, `type` = 3)
```

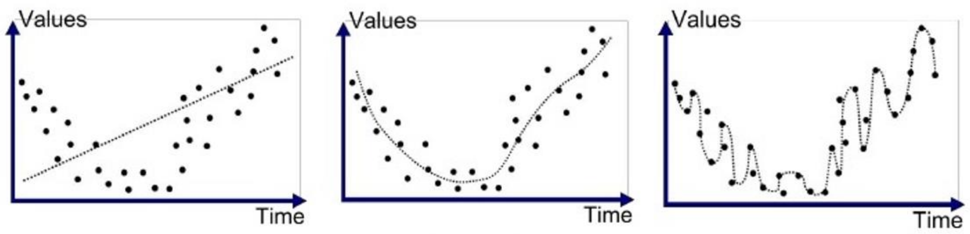
```r
results_test <- data.frame(
  predicted = preds_test,
  actual = mpg_test %>%
    filter(complete.cases(hwy, year, displ)) %>%
    select(hwy),
  type = rep("test", length(preds_test))
) %>%
  rename(`predicted` = 1, `actual` = 2, `type` = 3)
```

Overfit Vs Underfit: Compare the Test and Training Error

- rmse() function computes root mean squared error (i.e. MSE^(1/2))
- Pass the data frame of results
- Also the column names (in the results DF) for the predicted and true values
- We compare across models by examining the same error metric
- Since the error in the test set is lower than error in the training set we conclude the model is underfit, meaning we can increase model complexity

```
> rmse(results_train, predicted, actual)
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard        4.03
```

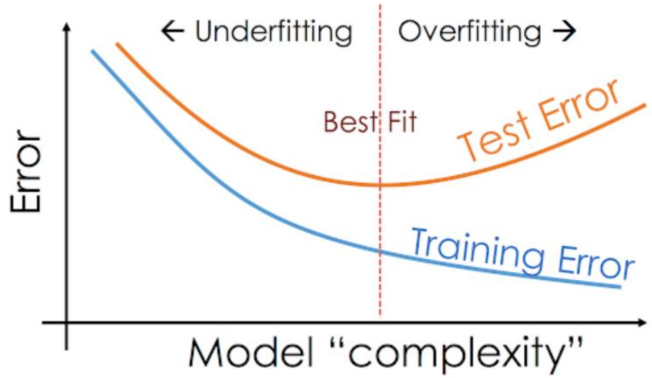
```
> rmse(results_test, predicted, actual)
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard        2.34
```



Underfitted　Good Fit/Robust　Overfitted

Other Functions for Evaluation: metrics() and mae()

```
> metrics(results_train, predicted, actual)
# A tibble: 3 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard       4.03
2 rsq     standard       0.563
3 mae     standard       2.93
> metrics(results_test, predicted, actual)
# A tibble: 3 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard       2.34
2 rsq     standard       0.820
3 mae     standard       1.83
```



- metrics() function estimates a series of evaluation metrics
- mae is "mean absolute error"

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i|$$

- rsq is our friend R^2

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y}_i)^2}$$

$$= 1 - \frac{sum\ of\ squared\ residuals}{sum\ of\ total\ squares}$$

- All tell the same story: model is underfit

Lab Time!

lab_class_9_linear_regression_3_and_Cla...

```
#---------------------------------------------------------
# Exercises
#---------------------------------------------------------

# 1. Load the movie data set and apply the cleaning code below.
# 2. Set seed of 1818 and Split into testing and training sets
#    using a training proportion of 70%.
# 3. Estimate a model of log movie grossM (Y) on log facebook likes
#    Store this as mod1 and summarize the object.
#    Be sure to use training data to estimate the model!
# 4. Interpret the coefficient on log facebook likes
# 5. Estimate a model of log movie grossM (Y) on log budgetM, and store this
#    as mod2 and summarize
# 6. Interpret the coefficient on log budget M
# 7. Estimate a model of log movie grossM (Y) on log budgetM, limdb_score, and
#    log facebook likes and store this as mod3 and summarize.
# 8. Interpret the coefficient on log of imdb score.
# 9. Upload code to canvas!

library('tidyverse')
library('forcats')
set.seed(1818)
movies <- read.csv(here::here("datasets", "IMDB_movies.csv"))
```

Class 9: Outline

1. Log Transformations and Interpreting log-log Regressions
2. Model Evaluation
   - Testing and Training Sets, RMSE, and R-Squared
3. Lab Class 9
4. **Why Classification Models?**
5. Logistic Function
6. Log Odds Ratio

Classification examples

Credit Card Default Dataset

Default {ISLR} R Documentation

Credit Card Default Data

Description

A simulated data set containing information on ten thousand customers. The aim here is to predict which customers will default on their credit card debt.

Usage

`Default`

Format

A data frame with 10000 observations on the following 4 variables.

`default`

A factor with levels `No` and `Yes` indicating whether the customer defaulted on their debt

`student`

A factor with levels `No` and `Yes` indicating whether the customer is a student

`balance`

The average balance that the customer has remaining on their credit card after making their monthly payment

`income`

Income of customer

Source

Simulated data

Why not regression?

```r
library(ISLR)
data(Default)
options(scipen = 3)

library(magrittr)
library(tidyverse)
library(ggExtra)

# create a binary version of default
Default %<>% mutate(default_binary =
                      ifelse(default == "Yes", 1,0))

summary(Default)

# estimate an OLS model using the 0,1
# variable as our dependent variable
mod1 <- lm(default_binary ~ balance,
           data = Default)
summary(mod1)
```

```
> summary(mod1)

Call:
lm(formula = default_binary ~ balance, data = Default)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.23533 -0.06939 -0.02628  0.02004  0.99046 

Coefficients:
               Estimate   Std. Error t value Pr(>|t|)    
(Intercept) -0.075191959  0.003354360  -22.42   <2e-16 ***
balance      0.000129872  0.000003475   37.37   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1681 on 9998 degrees of freedom
Multiple R-squared:  0.1226,	Adjusted R-squared:  0.1225 
F-statistic:  1397 on 1 and 9998 DF,  p-value: < 2.2e-16
```

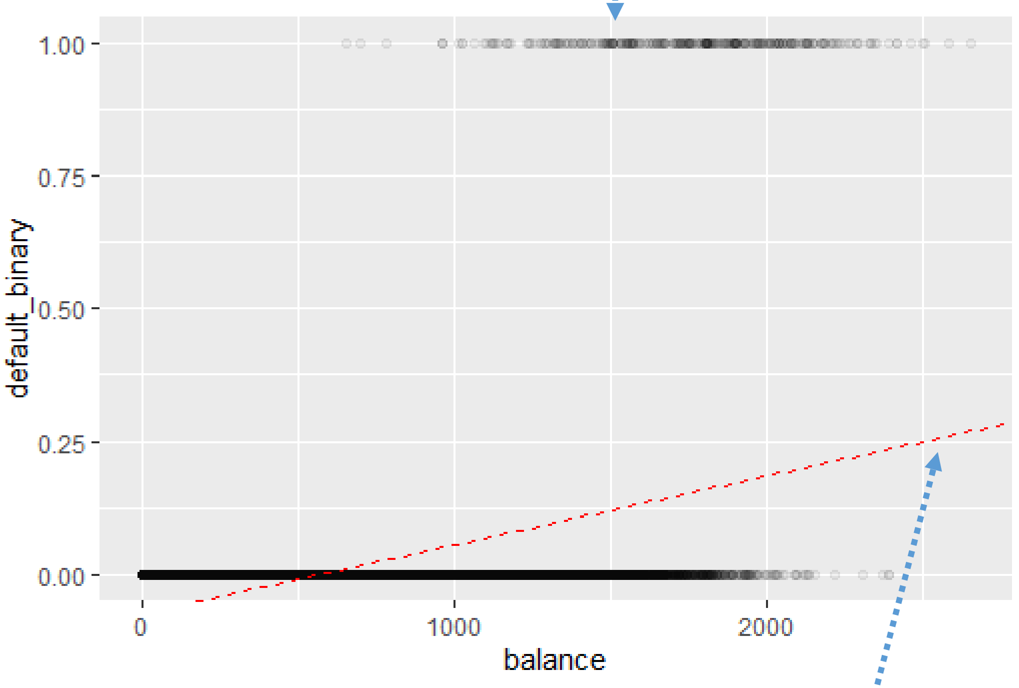
- Let’s estimate a model predicting default based on credit card balance
- R2 looks low but otherwise this model looks perfectly fine

Linear Model to Predict Default

```
preds_DF <- data.frame(
  preds = predict(mod1),
  Default
)

# what kind of predictions do we get for this model?
head(preds_DF)

p <- ggplot(preds_DF, aes(x = balance,
                          y = default_binary)) +
  geom_point(alpha = 1/20) +
  geom_abline(intercept = mod1$coefficients[1],
              slope = mod1$coefficients[2],
              color = "red", linetype = "dashed")

plot(p)
```

- Let’s use the model to generate predictions of default (which is binary)

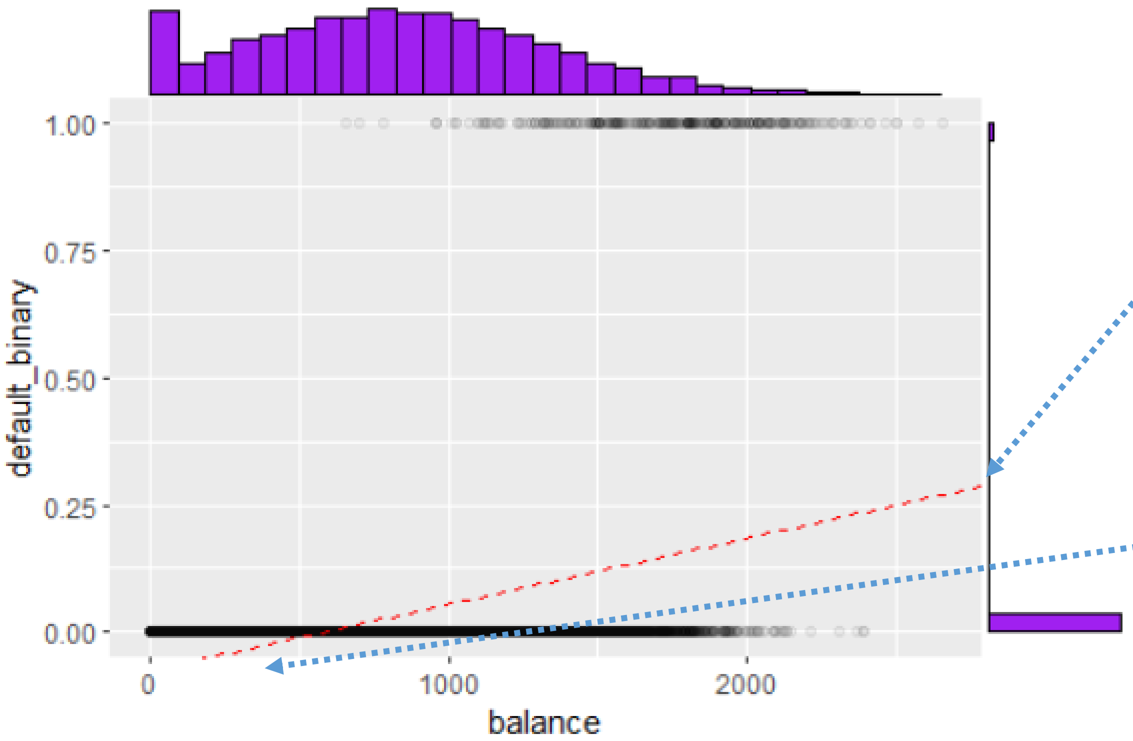
- Black dots show actual default behavior



- Red line shows the predictions from the model

Why is the red line a bad prediction model for default?

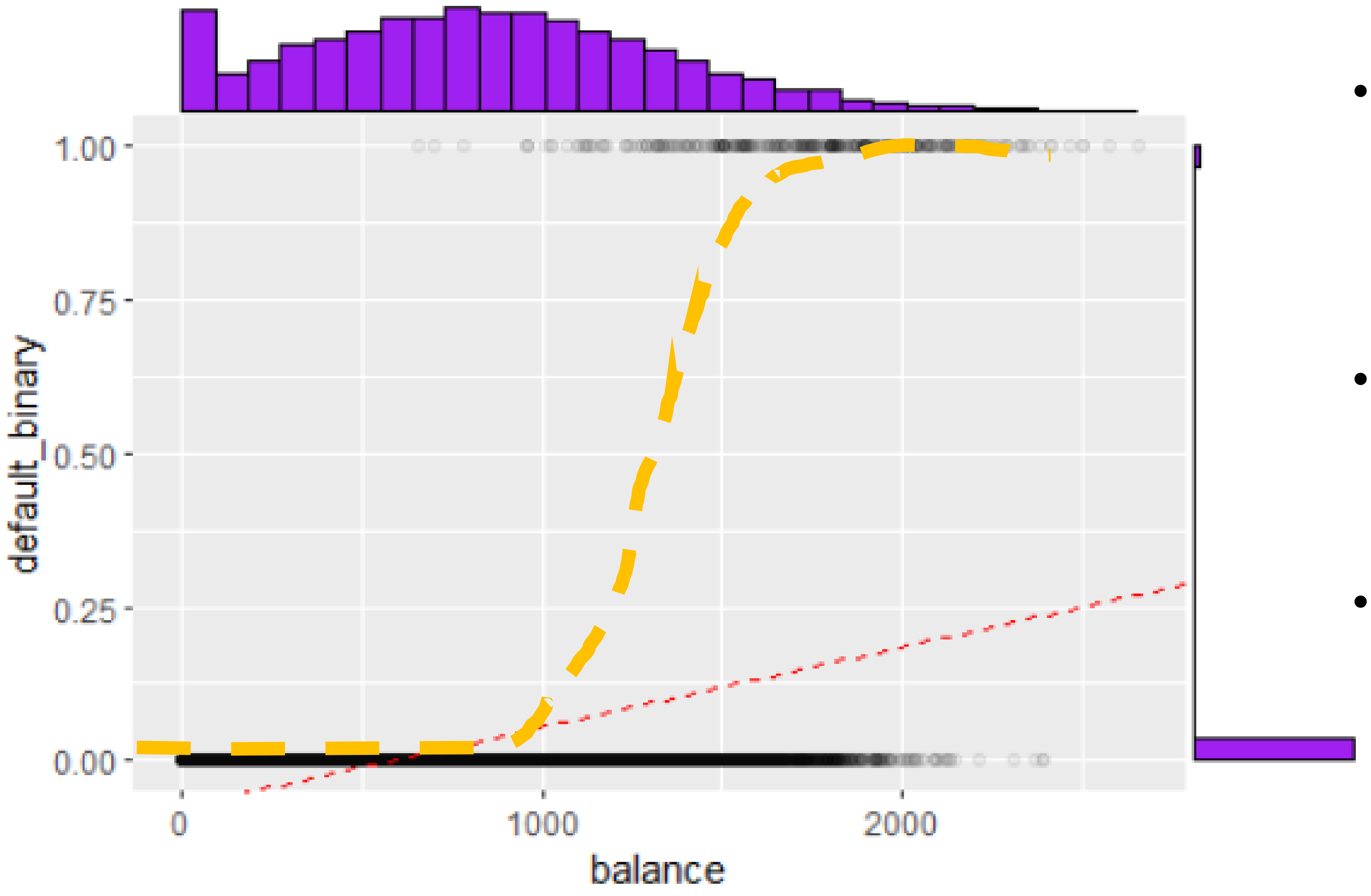
Why Is This a Bad Prediction Model?




```
p <- ggplot(preds_DF, aes(x = balance,
                          y = default_binary)) +
  geom_point(alpha = 1/20) +
  geom_abline(intercept = mod1$coefficients[1],
              slope = mod1$coefficients[2],
              color = "red", linetype = "dashed")

plot(p)
p <- ggMarginal(p, type = "histogram", fill="purple", size=6)
plot(p)
```

1. We never predict more than 30% chance of default!
2. Most observations do not default!
3. We predict negative probability of default!

One Way to Improve the Earlier Model: Squash Predictions



- Because probabilities are between 0 and 1 we want to compress red line to lie within 0 and 1 on the y axis
- i.e let $p(X) = Pr(Y = 1|X)$ be the probability the event occurs
- We want our model to output:

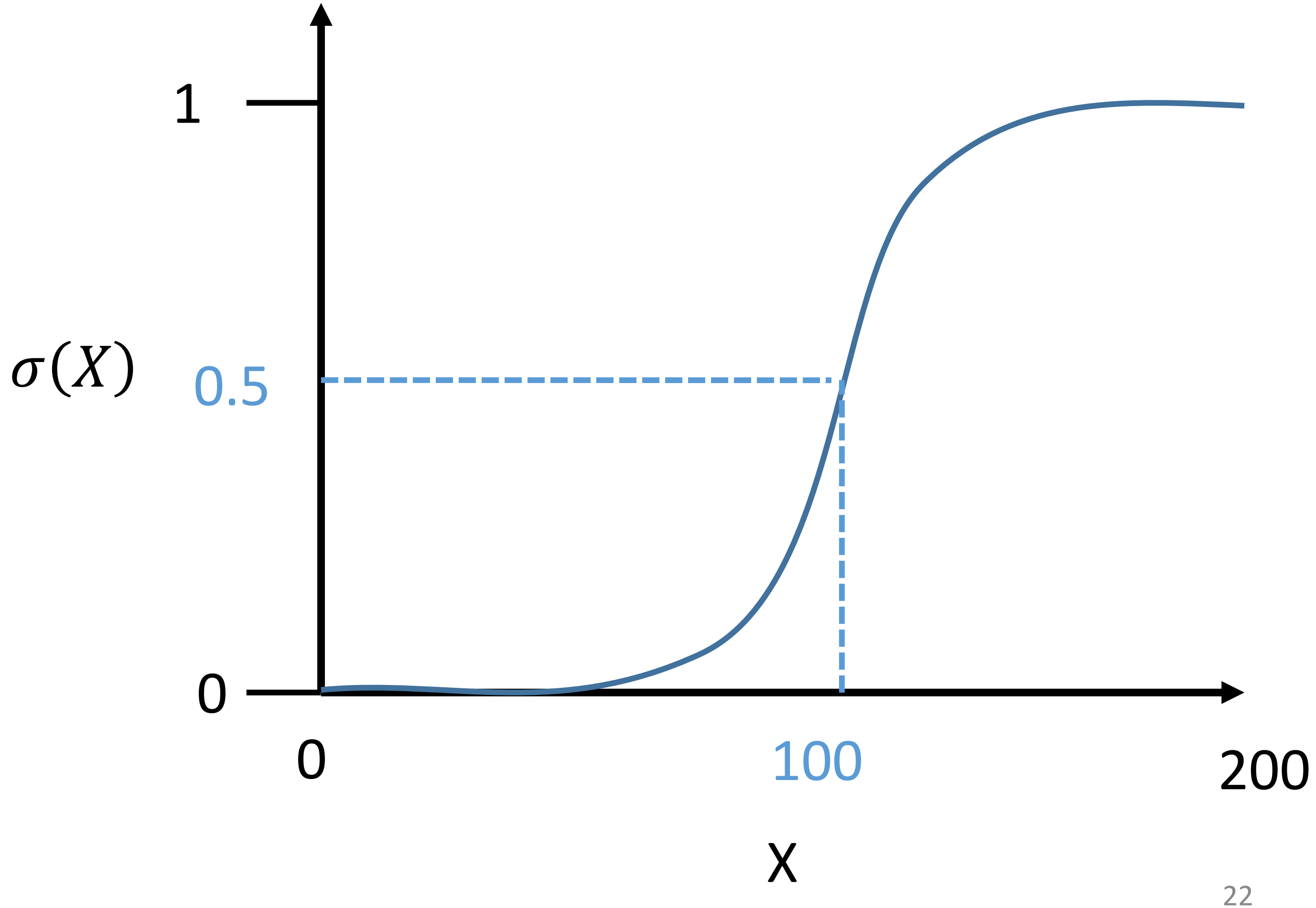
$$Pr(Y = 1|X) \in [0,1]$$

What is The Logistic/Sigmoid Function

- Logisitic is a function that naturally takes inputs X and transforms between 0 and 1

- The logisitic is defined as

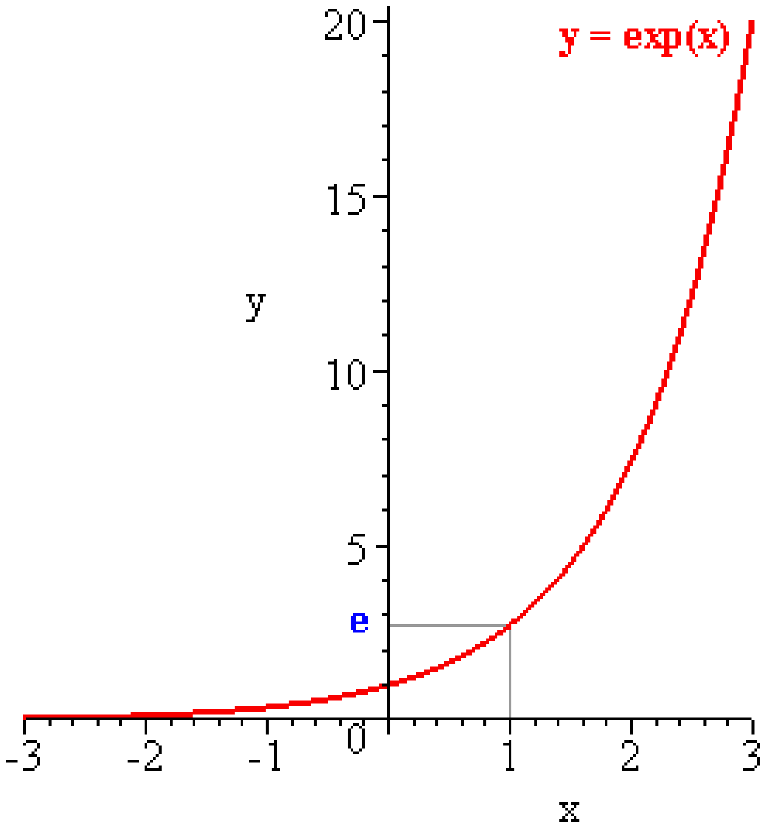
$$\sigma(X) = \frac{1}{1+e^{-X}} = \frac{e^X}{e^X+1}$$



A note on $e^X = \exp(X)$

- Super spooky mathematical function
- $e = 2.718281828459045\ldots$
- $\frac{d}{dx} e^{\mathrm{X}} = \mathrm{e}^{\mathrm{X}}$ and $e^0 = 1$
  - e.g. rate of increase in function at X is equal to the function at X

Many other ways to characterize function



Using the Logistic/Sigmoid Function to Generate Probabilities

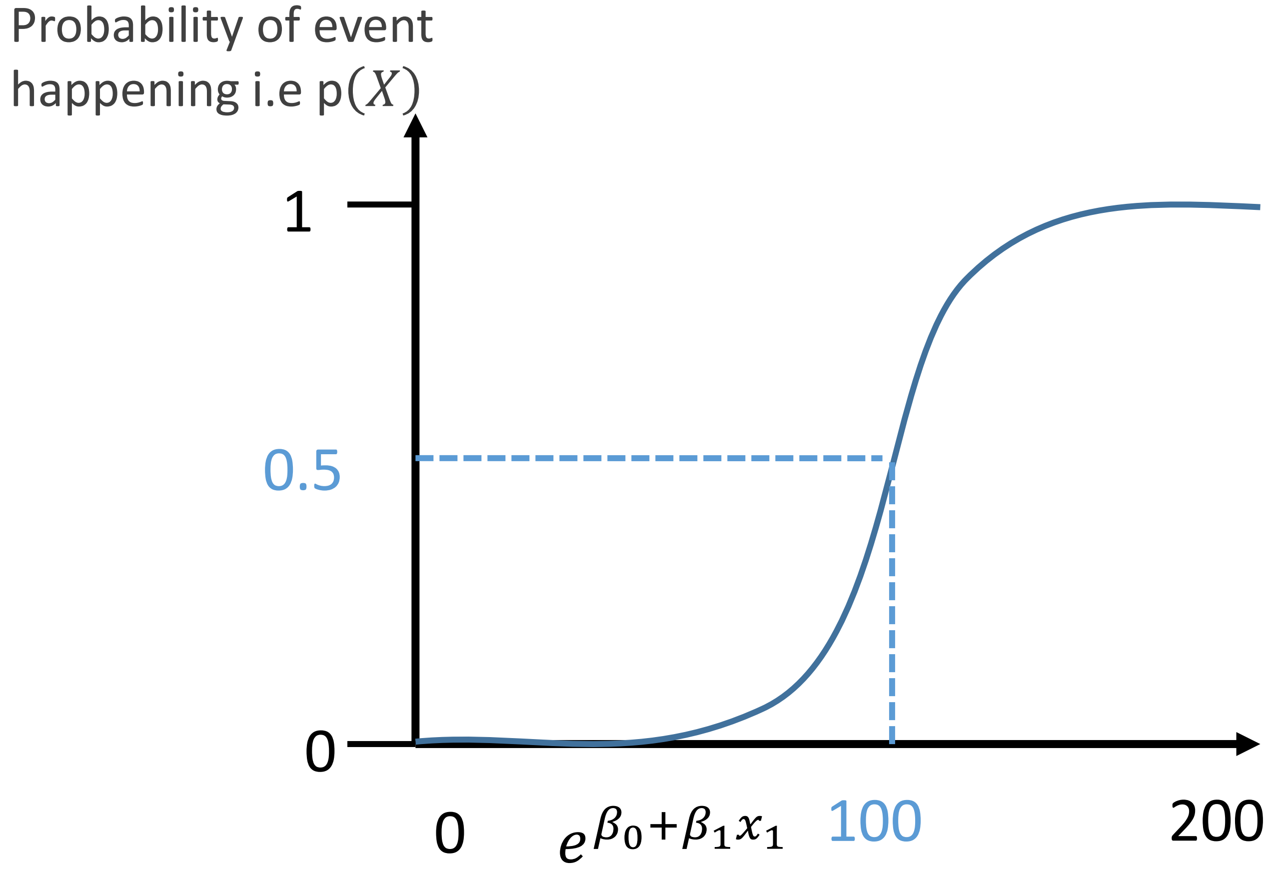
- How do we generate probabilities from this function?

- We let X = $\beta_0 + \beta_1 \cdot x_1 + \cdots + \beta_k \cdot x_k$ and plug this into the logistic function

$$\sigma(X) = \frac{1}{1+e^{-X}} = \frac{e^X}{e^X + 1}$$

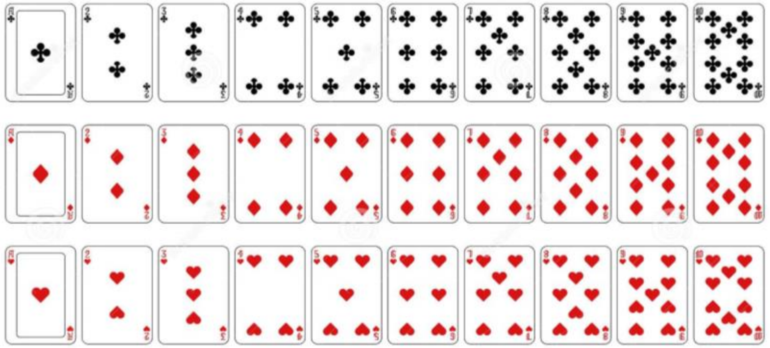
$$Pr(Y = 1|X) = \frac{e^{\beta_0+\beta_1 \cdot x_1}}{e^{\beta_0+\beta_1 \cdot X} + 1}$$

$$Pr(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0+\beta_1 x_1)}}$$



This is equivalent mathematically! I promise. Work it out on pen and paper if you don't believe me

What is the "Odds Ratio"



- The outcome variable in a logit regression is the "odds ratio" (OR)
- In a deck of 52 cards there are 13 spades
- The probability of randomly drawing a spade is 13/52 = 25%
- The probability of not drawing a spade is 39/52 = 75%
- Therefore the ratio of odds of drawing a spade vs not drawing a spade is

$$\frac{ratio\ of\ drawing\ a\ spade}{ratio\ of\ \text{not}\ drawing\ a\ spade} = \frac{13/52}{39/52} = \frac{13}{39} = 1:3 = 0.333$$

- Log odds ratio is just log(0.333) = -0.4771…

Logit Models Model the Outcome As a Log Odds Ratio

$$\frac{p(Y=1|X)}{p(Y=0|\ X)} = e^{\beta_0+\beta_1 X}$$

$$log\left(\frac{p(Y=1|X)}{p(Y=0|\ X)}\right) = \log\left(e^{\beta_0+\beta_1 X}\right)$$

$$log\left(\frac{p(Y=1|X)}{p(Y=0|\ X)}\right) = \beta_0 + \beta_1 X$$

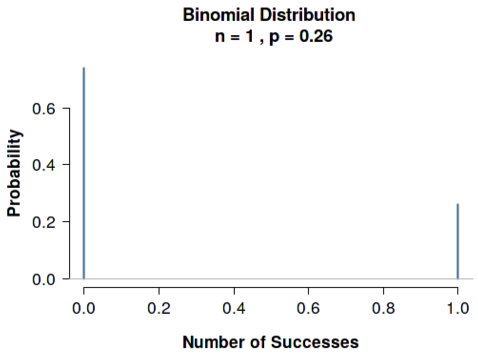
The outcome variable (Y) for a logistic regression is the log odds ratio

Log odds ratio is a linear expression of constants and coefficients of a nonlinear process!

All logistic coefficients can be interpreted as impact on log odds ratio

Estimating Logit Models Using glm()

```
#-----------------------------------------------------
# Estimating Logistic Regression in R
#-----------------------------------------------------
library('ISLR')
# load data which has credit card default behavior
data(Default)
head(Default)

# make sure to use glm() function!
# set family = binomial to set logistic function
logit_fit1 <- glm(default ~ student,
                  family = binomial,
                  data = Default)
```



- Estimate logistic regression using the function glm() in R
- We still specify a formula in the usual manner
- glm() estimate a variety of "generalized linear models"
- To specific logit we must use the option "family = binomial"
- Binomial is a binary distribution aka the "link" function

If curious see more here: https://shiny.rit.albany.edu/stat/binomial/

Estimating Impact of Being a Student on Default Probability using glm()

$$log\left(\frac{p(Y = default|X)}{p(Y = not\ default|\ X)}\right) = \beta_0 + \beta_1 \cdot student_i + \epsilon_i$$

$$\exp\left(log\left(\frac{p(Y = default|X)}{p(Y = not\ default|\ X)}\right)\right) = \exp(\beta_0 + \beta_1 \cdot student_i + \epsilon_i)$$

$$\frac{p(Y = default|X)}{p(Y = not\ default|\ X)} = \exp(\beta_0 + \beta_1 \cdot student_i + \epsilon_i)$$

```
glm(formula = default ~ student, family = binomial, data = Default)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.2970  -0.2970  -0.2434  -0.2434   2.6585

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.50413    0.07071  -49.55  < 2e-16 ***
studentYes   0.40489    0.11502    3.52 0.000431 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> exp(logit_fit1$coefficients)
(Intercept)  studentYes
 0.03007299  1.49913321
```

- Remember the outcome variable in a logistic regression is the log odds ratio
- If we exponentiate the coefficients this tells us the impact of the variable on the unlogged odds ratio
- If we take our estimated logistic model we see $\beta_1 = 0.40489$
- This means students have a 0.40489 higher log odd of defaulting
- Exponentiating the coefficients returns the impact of the X-variable on the odds ratio directly.
- Therefore the ratio of odds of default for student vs non-student is 1.49, or students have a 49% higher probability of default

Lab Time!

```
#-----------------------------------------------------
# Lab 1
#-----------------------------------------------------
# 1. Estimate a logistic regression model predicting
#    default as a function of student, balance, and income
#    and store this as 'logit_mod2'
# 2. Exponentiate the coefficient vector of logit_mod2
# 3. Interpret the impact of being a student on the probabiliy of default
# 4. Do students face a higher or lower risk of credit card default?
```

Generating Predicted Probabilities from a Logit Model

- To generate predictions, we use the estimated coefficients in the logit equation

$$\hat{p}(X = 1000) = \frac{e^{\widehat{\beta}_0 + \widehat{\beta}_1 x_1}}{1 + e^{\widehat{\beta}_0 + \widehat{\beta}_1 x_1}} =$$

- The estimated probability of default with a balance of $1,000 is given by
- The estimated probability of default with a balance of $2,000 is given by
- To generate predicted probability for all observations in a dataset we use the predict function, **but note type = "response"!**
- This is also called "scoring" a dataset

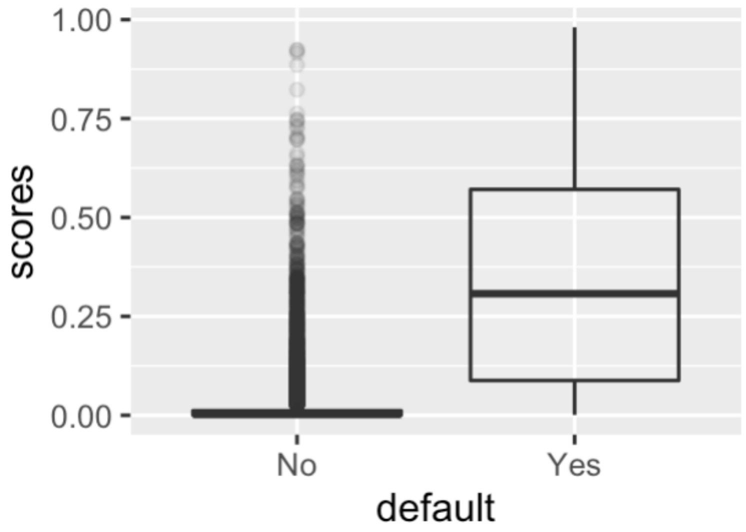
```
glm(formula = default ~ balance, family = binomial, data = Default)

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) -10.6513306   0.3611574  -29.49   <2e-16 ***
balance       0.0054989   0.0002204   24.95   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

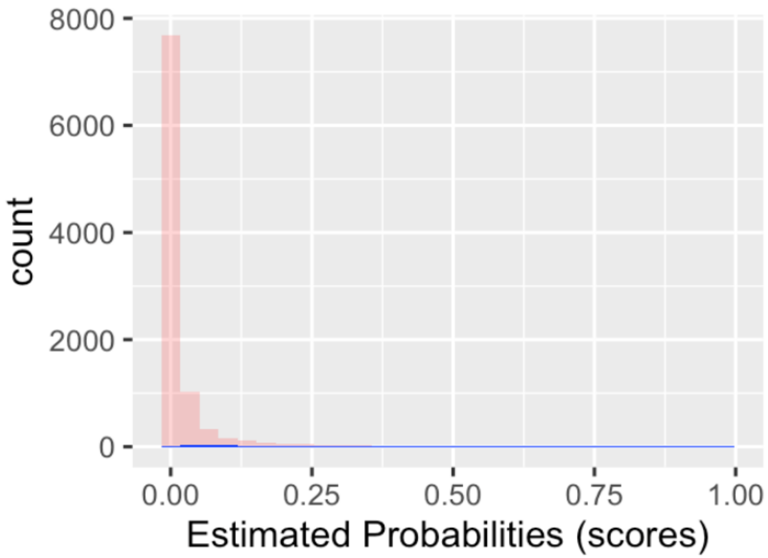
$$\hat{p}(X = 1000) = \frac{e^{-10.6513 + 0.0055 \cdot \mathbf{1000}}}{1 + e^{-10.6513 + 0.0055 \cdot \mathbf{1000}}} = 0.00575$$

$$\hat{p}(X = 2000) = \frac{e^{-10.6513 + 0.0055 \cdot \mathbf{2000}}}{1 + e^{-10.6513 + 0.0055 \cdot \mathbf{2000}}} = 0.05857$$

```
scores <- predict(logit_fit3,
                  type = "response")
```

What Do We Do With Scores or Estimated Probabilities?



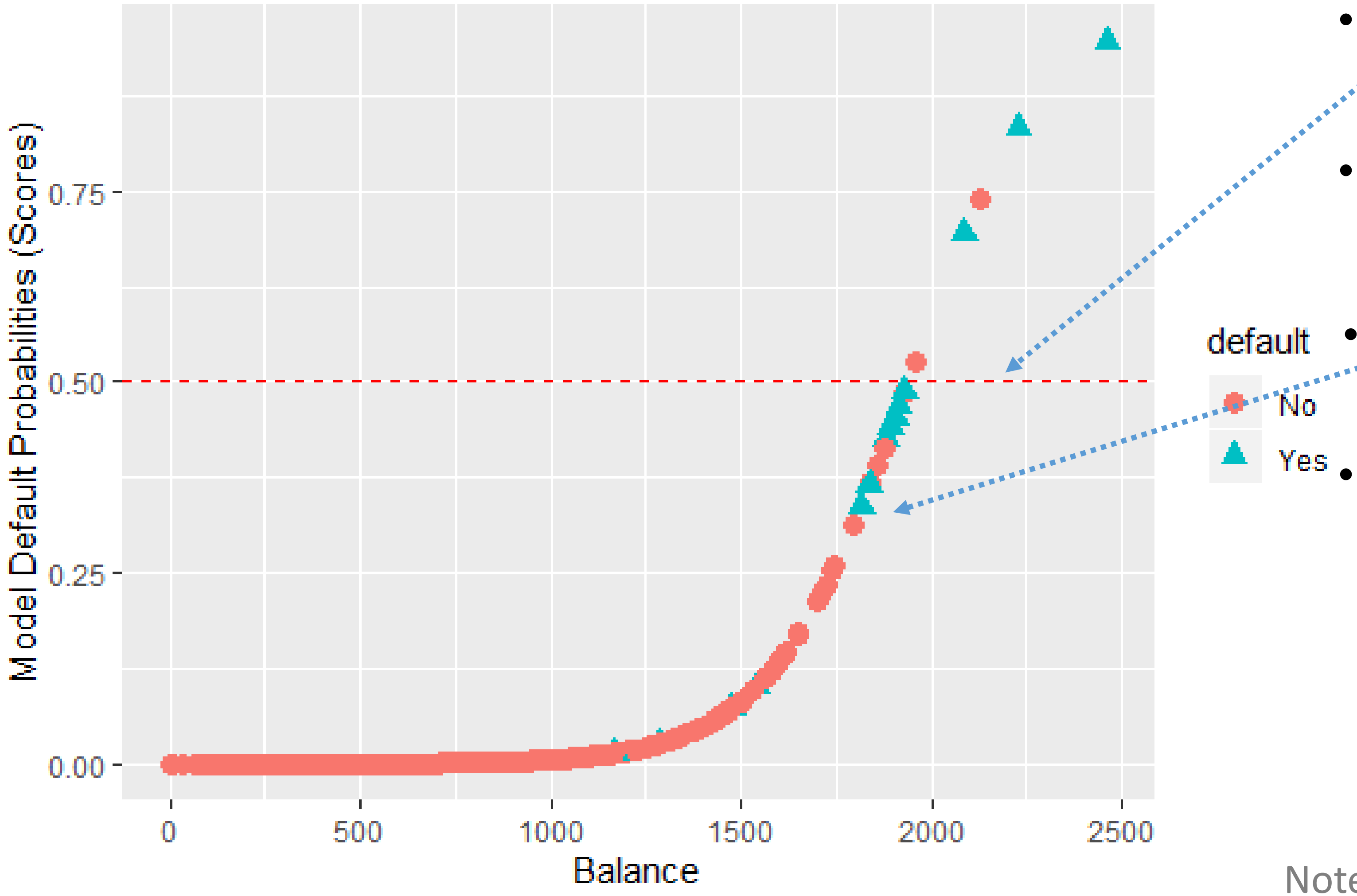


- Okay, so we have probabilities, what then?
- Note there is almost always some overlap between the probabilities of the classes
- We can't choose a probability such that above this all actual defaulters are correctly identified, and below this all non-defaulter are identified
- So we will always have some **false positives** and **false negatives**

Confusion Matrix: Table of False/True Positives and False/True Negatives

		True default status	
		No	**Yes**
Predicted default status	**No**	True negative (TN)	False Negative (FN)
	Yes	False Positive (FP)	True Positive (TP)

Assigning Class=Default to $\hat{p} > 0.5$

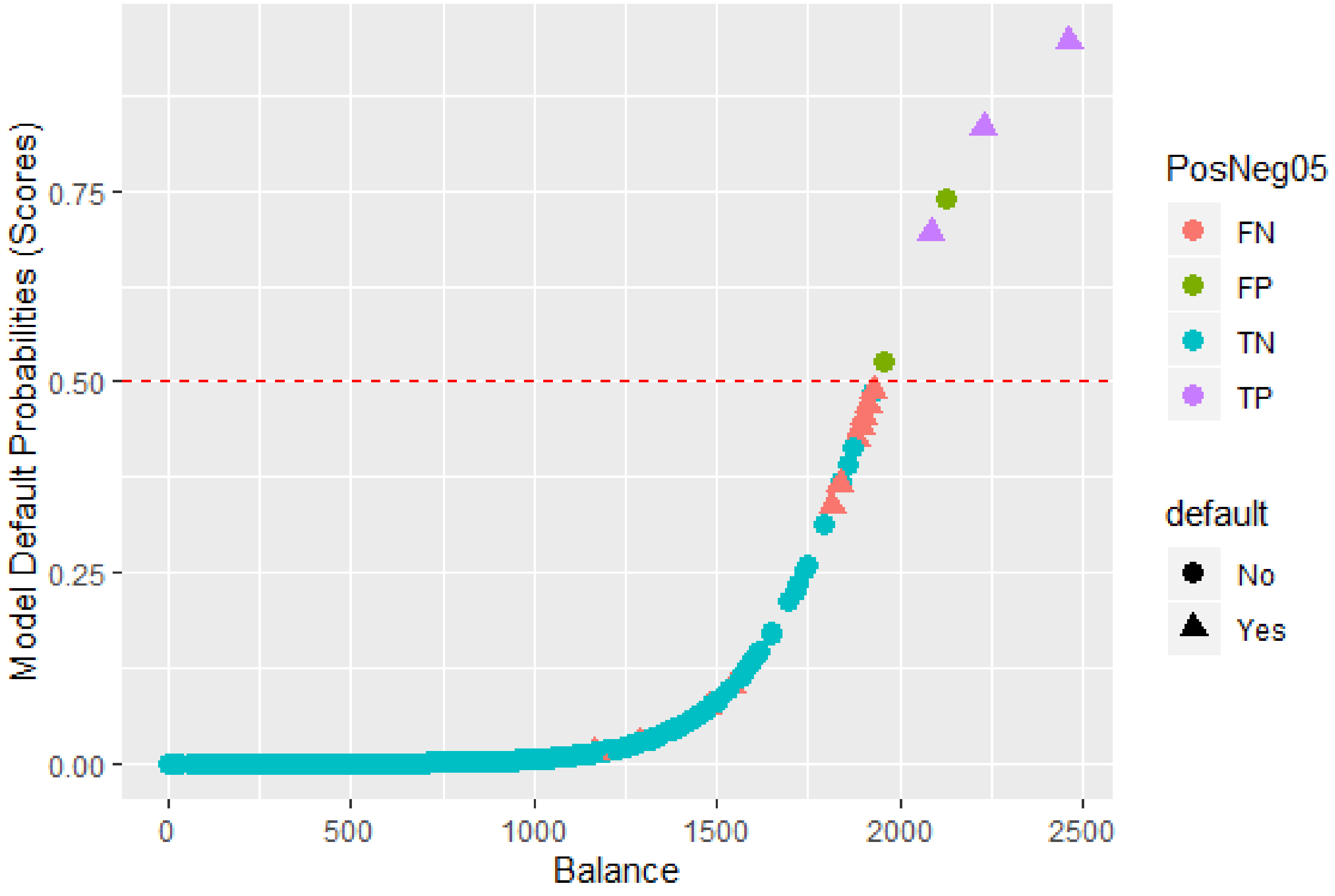


- Above this line, observations are classified as defaulting
- Below this line, observations classified as **not** defaulting
- Actual default = teal triangles
- If we choose a probability cutoff of 0.5, then we see we have 2 false positives and 11 FN

```
> table(preds_sample$PosNeg05)

 FN  FP  TN  TP
 11   2 484   3
```

Note I'm working with a 5% sample of the dataset to make the numbers easier

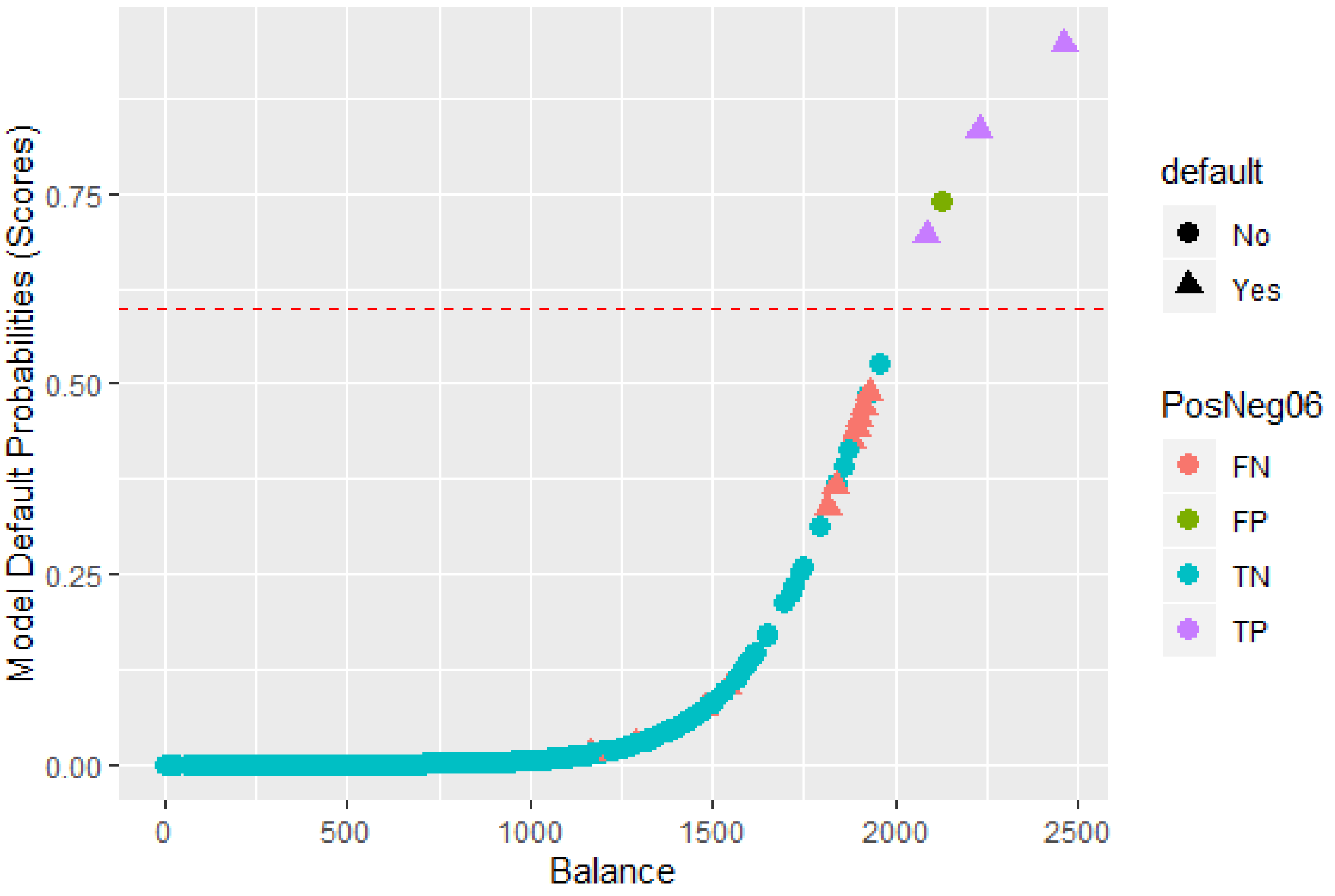
Assigning Class=Default to $\hat{p} > 0.5$



- If we choose a probability cutoff of 0.5, then we see we have 2 false positives and 11 FN

```
> table(preds_sample$PosNeg05)

 FN  FP  TN  TP
 11   2 484   3
```

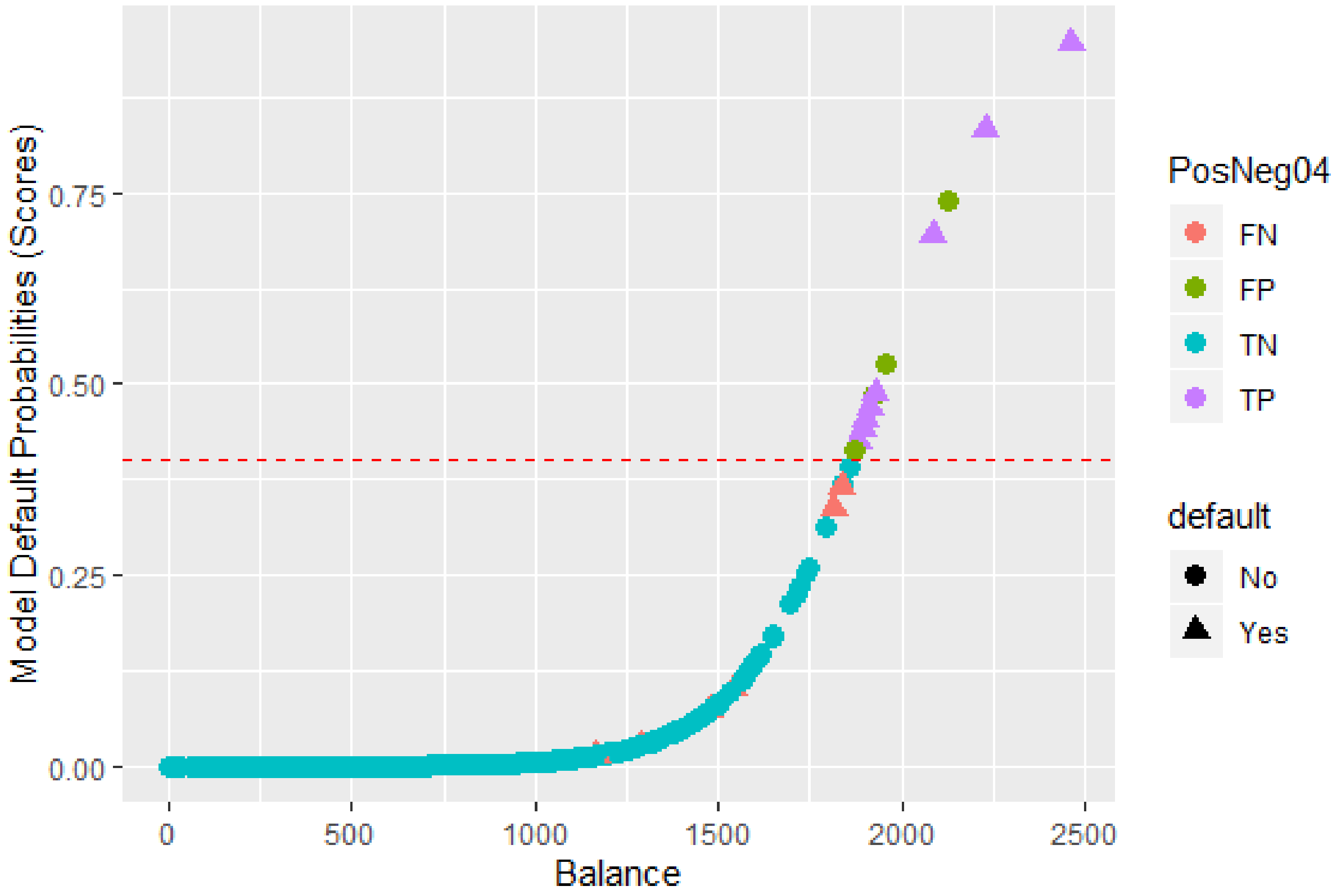
Assigning Class=Default to $\hat{p} > 0.6$



- Raising the cutoff to 0.6, then we see we have 1 false positives and 11 FN

```
> table(preds_sample$PosNeg06)

 FN  FP  TN  TP
 11   1 485   3
```

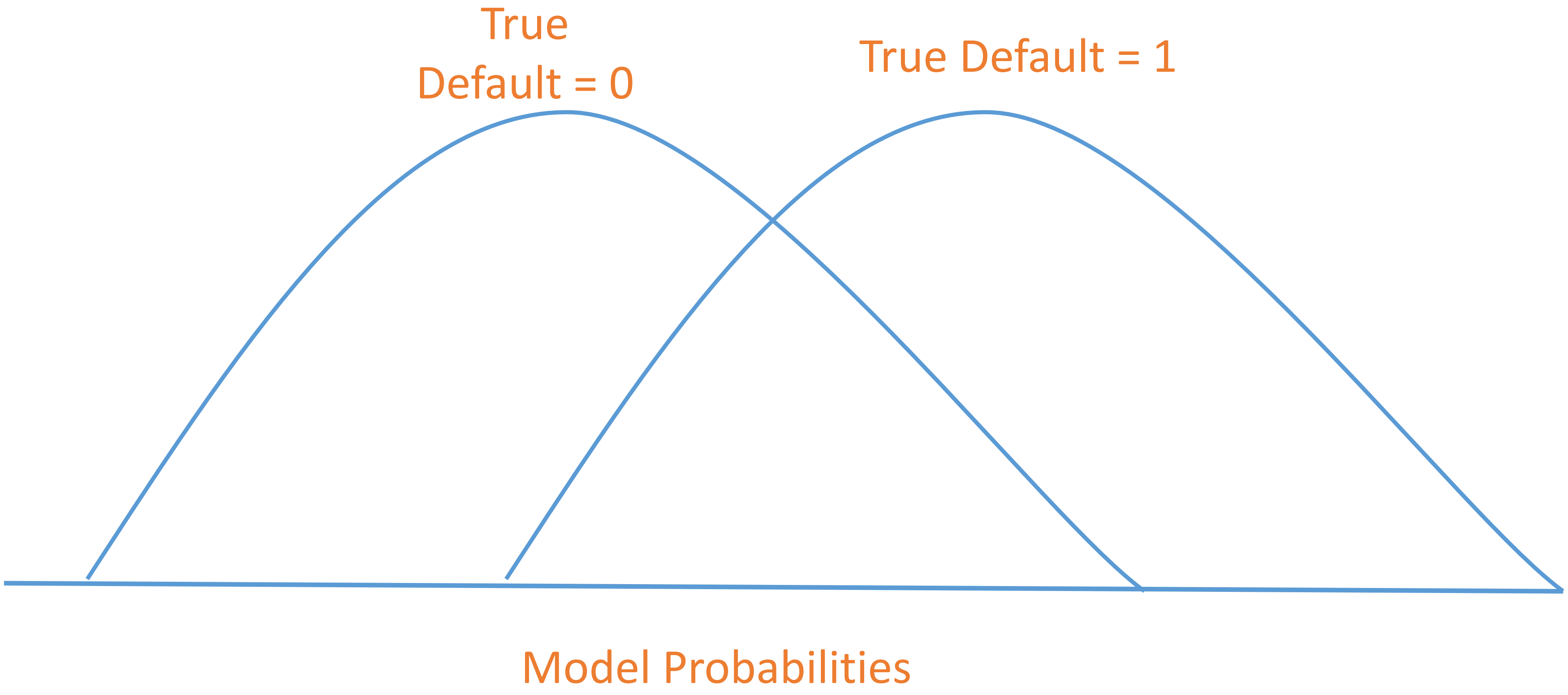
Assigning Class=Default to $\hat{p} > 0.4$



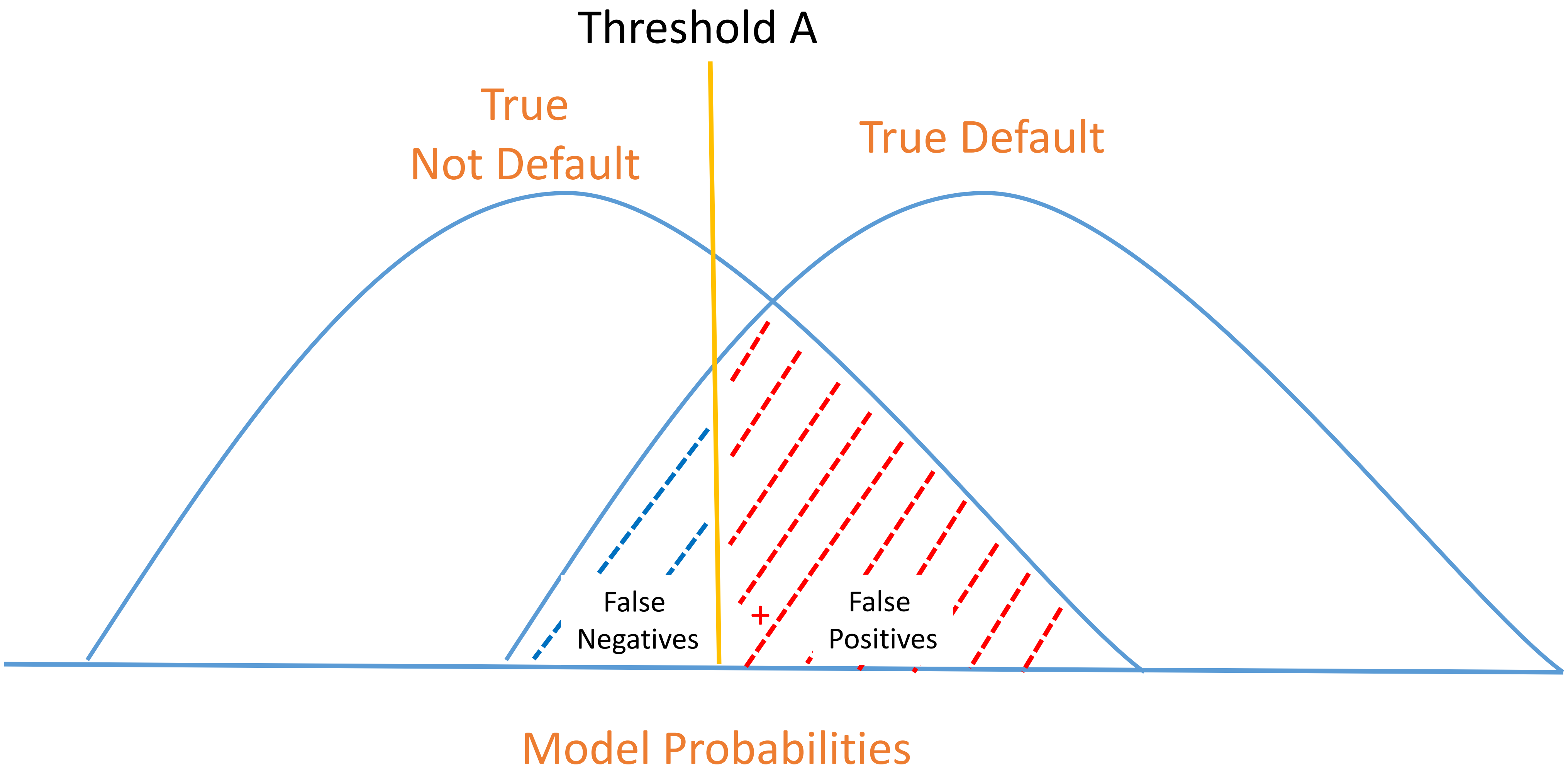
- Lowering the cutoff to 0.4 results in more FPs (4) but fewer FNs (6)

```
> table(preds_sample$PosNeg04)

 FN  FP  TN  TP 
  6   4 482   8 
>
```

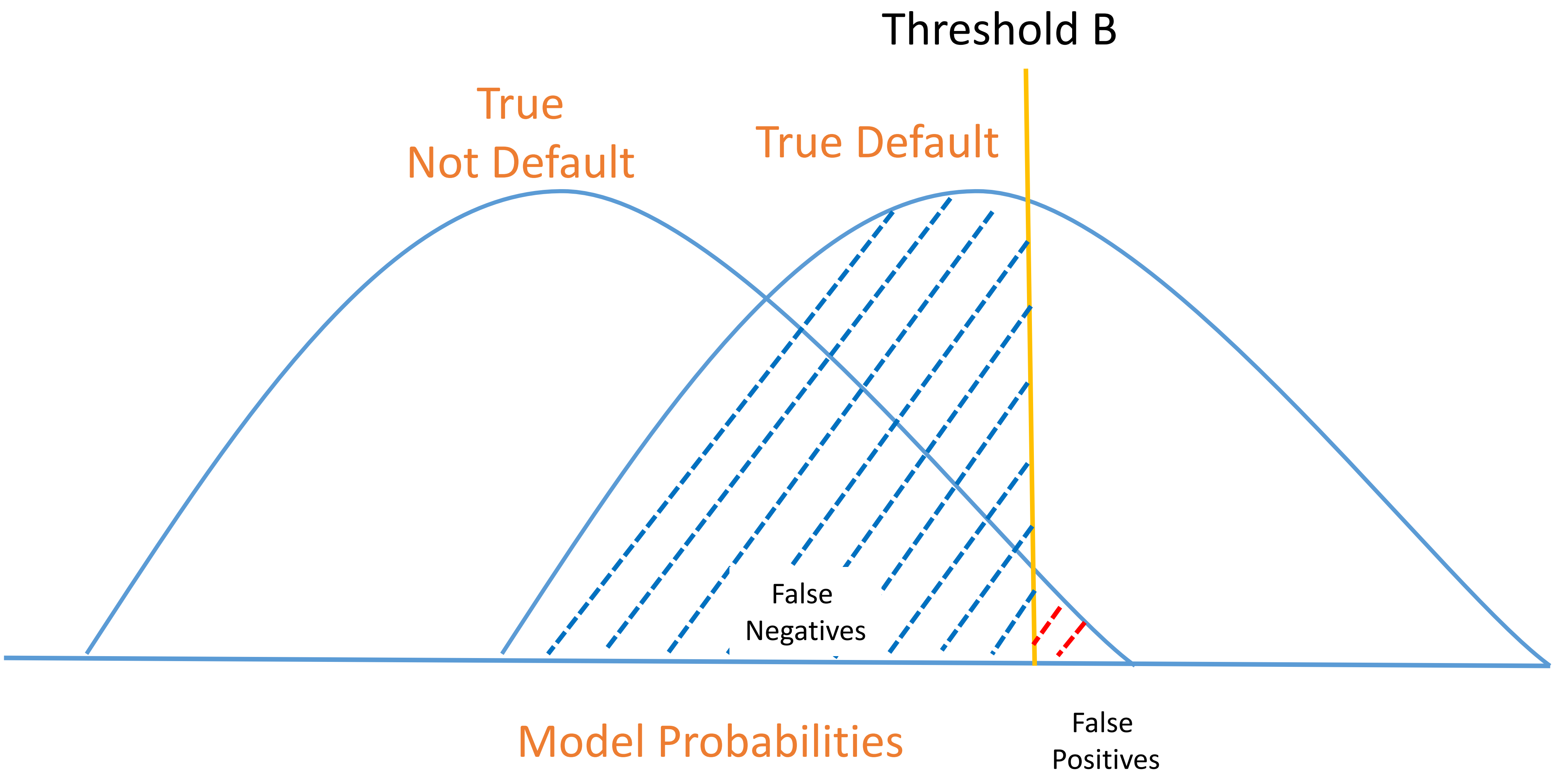
Choosing Probability Cutoff to Assign Class



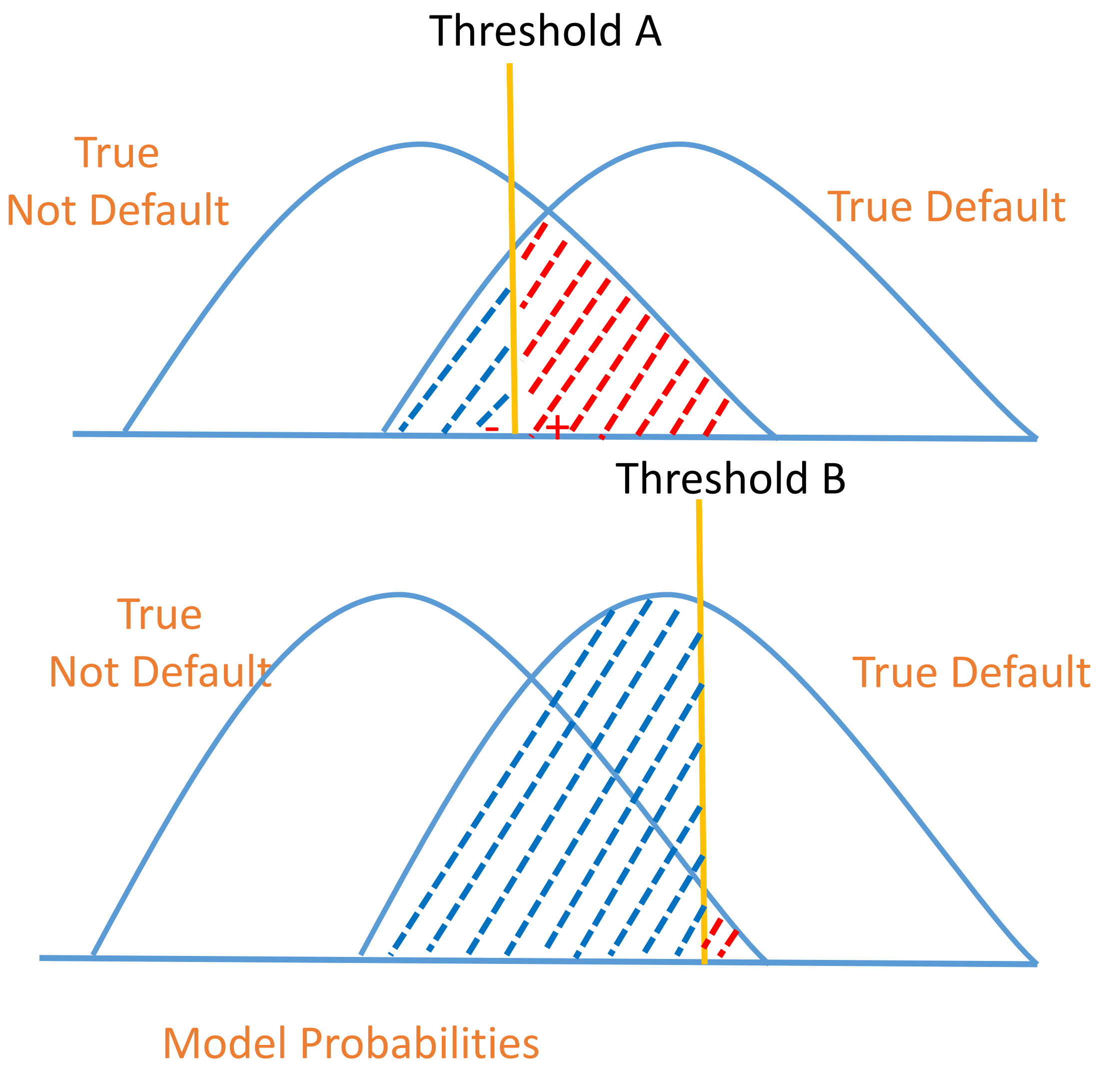
Threshold A: Moderate Threshold



Threshold B: Higher Threshold



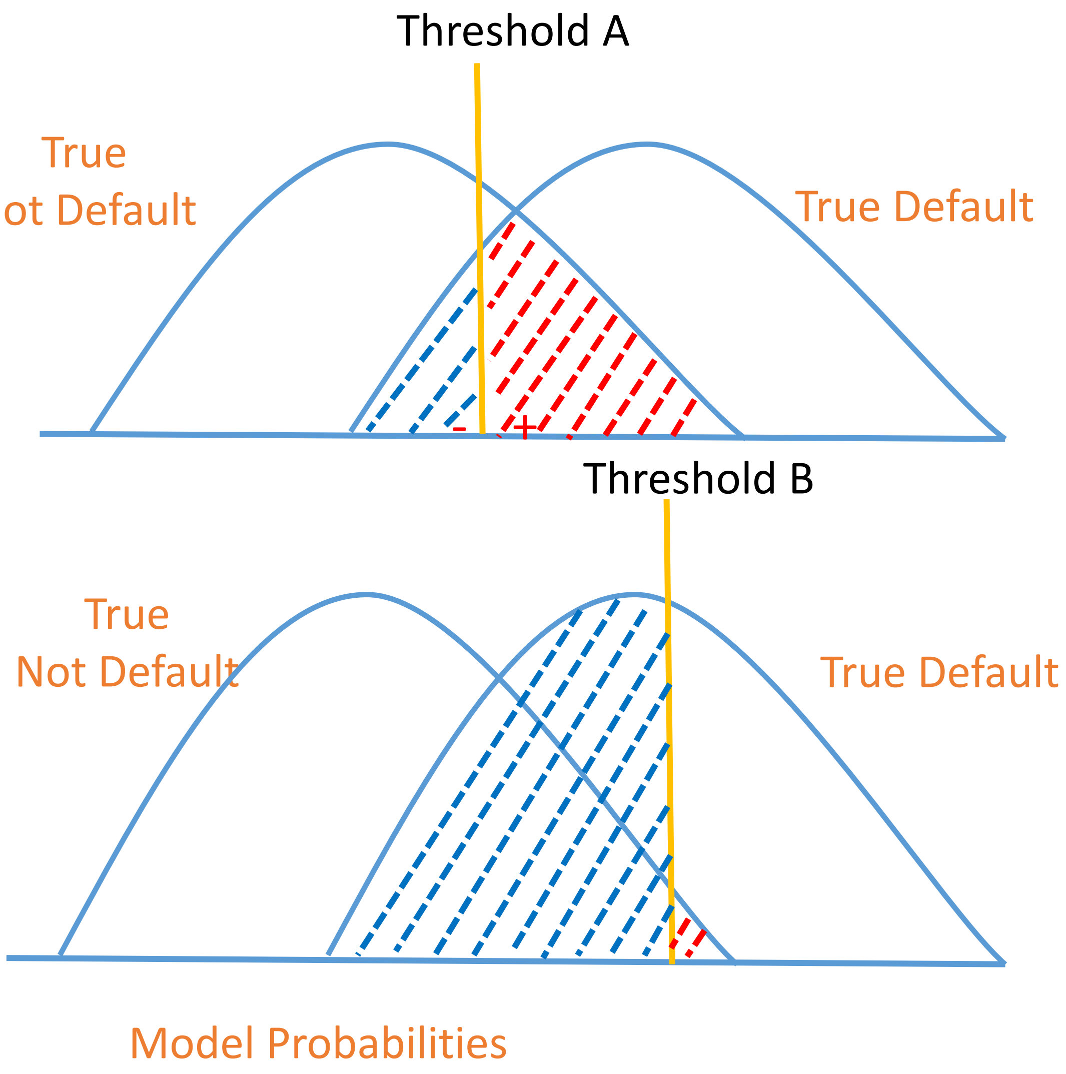
Comparing cutoffs:



Many false positives

Few false positives

Which Probability Cutoff To Use?



- Threshold you choose should depend on relative costs of FPs and FN
  - e.g. screening at airport (cost of false neg high)
  - e.g. direct mail advertisement (cost of false positive low)
- Some common choices
  - **Maximize Accuracy** (equal weighting of FPs and FNs)
  - **Threshold p_hat** > 0.5
  - **Minimize cost**: TC = costFP *FPs + cost FN * TNs

Sensitivity and Specificity, Confusion Matrix at P Cutoff > 0.5

		True default status		
		No	**Yes**	
Predicted default status (cutoff p>0.5)	**No**	TN = 484	FN = 11	N^* = 495
	Yes	FP = 2	TP = 3	P^*= 5
		N = 486	P = 14	

- **Sensitivity:** True positive rate (aka 1 – power or recall)
  - TP/P = 3 / 14 = 21.4%
- **Specificity:** True negative rate
  - TN/N = 484 / 486= 99.5%

- **False positive rate** (aka Type I error, 1 - Specificity)
  - FP/N = 2/486= 0.004%

Generating Confusion Matrices in R

- To produce a confusion matrix in R we will use the yardstick package
- The function conf_mat() produces confusion matrices but we must format our data correctly
- We need to specify a data frame with
- Actual event (Y = 1) values
- Our estimated probabilities (scores)
- This example data frame shows how we need to structure our results data frame

```
Usage

conf_mat(data, ...)

## S3 method for class 'data.frame'
conf_mat(data, truth, estimate, dnn = c("Prediction", "Truth"), ...)

## S3 method for class 'conf_mat'
tidy(x, ...)

autoplot.conf_mat(object, type = "mosaic", ...)
```

```
> head(two_class_example)
   truth      Class1       Class2 predicted
1 Class2 0.003589243 0.9964107574    Class2
2 Class1 0.678621054 0.3213789460    Class1
3 Class2 0.110893522 0.8891064779    Class2
4 Class1 0.735161703 0.2648382969    Class1
5 Class2 0.016239960 0.9837600397    Class2
6 Class1 0.999275071 0.0007249286    Class1
```

Formatting Results Matrix for Confusion Matrix

- Let’s store the model results in a data frame
- We must specify the actual default behavior
- And the probability of class1 (default) as well as probability of class2 (not default)
- We *must* specify a cutoff above which probabilities are classified as “class1” (or having the event) and below which they are not
- The cutoff probability is determined by the relative cost of false positives and false negatives! Do not use rules of thumb!

```
results_logit <- data.frame(
  `truth` = Default$default,
  `Class1` =  scores,
  `Class2` = 1 - scores,
  `predicted` = as.factor(ifelse(scores > 0.4,
                         "Yes","No"))
)
```

Why Do So Many Practicing Data Scientists Not Understand Logistic Regression?

Posted on June 27, 2020 by W.D.

Logistic Regression is Not Fundamentally a Classification Algorithm

Classification is when you make a concrete determination of what category something is a part of. Binary classification involves two categories, and by the law of the excluded middle, that means binary classification is for determining whether something “is” or “is not” part of a single category. There either are children playing in the park today (1), or there are not (0).

https://ryxcommar.com/2020/06/27/why-do-so-many-practicing-data-scientists-not-understand-logistic-regression/

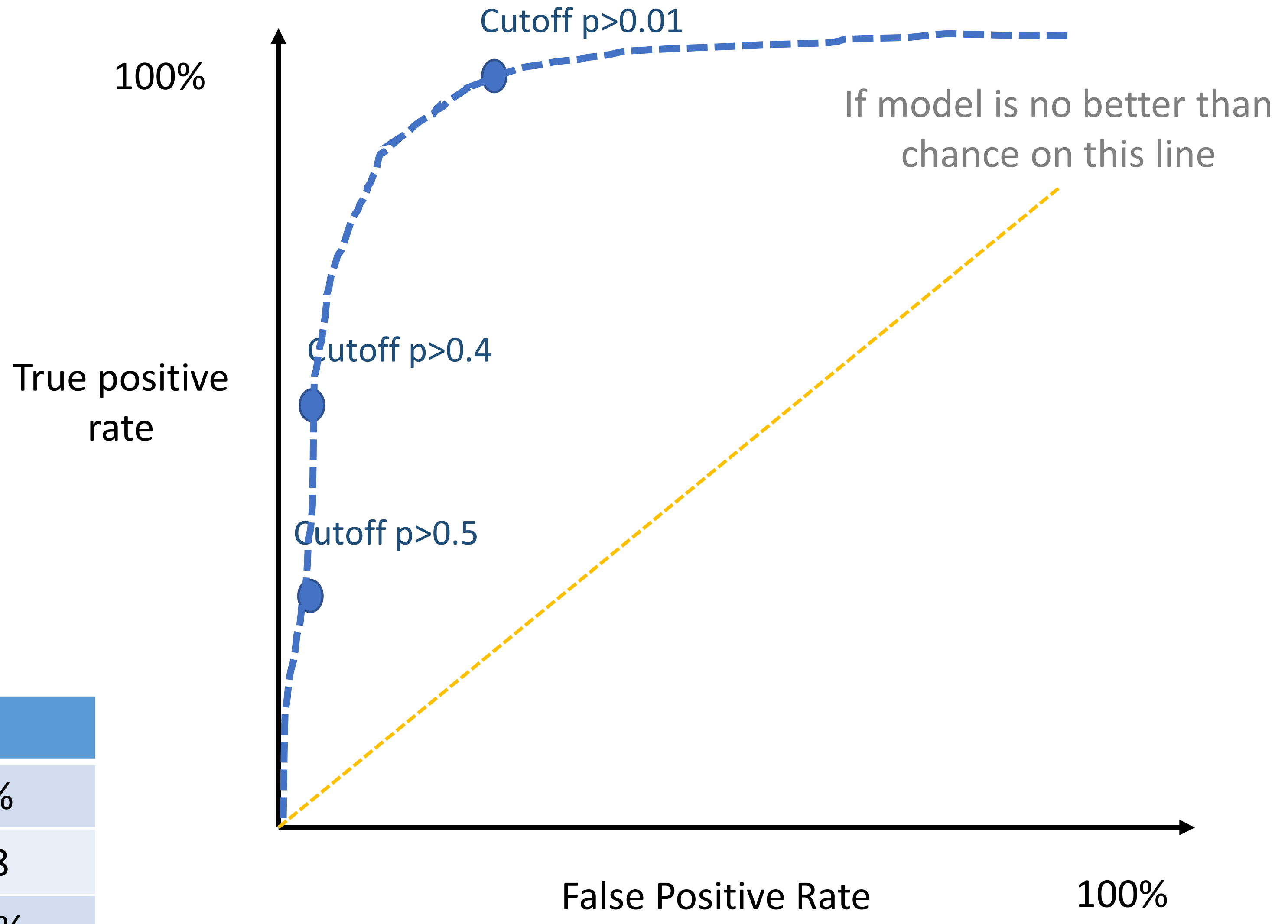
Producing Confusion Matrix Using Formatted Results Data

- The conf_mat() function shows the confusion matrix
- If we summarize the conf_mat() object we see more binary metrics of classification (don't need to know all of these)
- **Sensitivity** is the true positive rate (TP/P) and here we identify of the true positives = 131/333 = 39.3%
- We may need to lower our threshold of cutoff probability

```
> cm <- conf_mat(results_logit,
+                truth = truth,
+                estimate = predicted)
> print(cm)
          Truth
Prediction   No  Yes
       No  9594  202
       Yes   73  131
```

Continuous Cutoff: ROC Curve

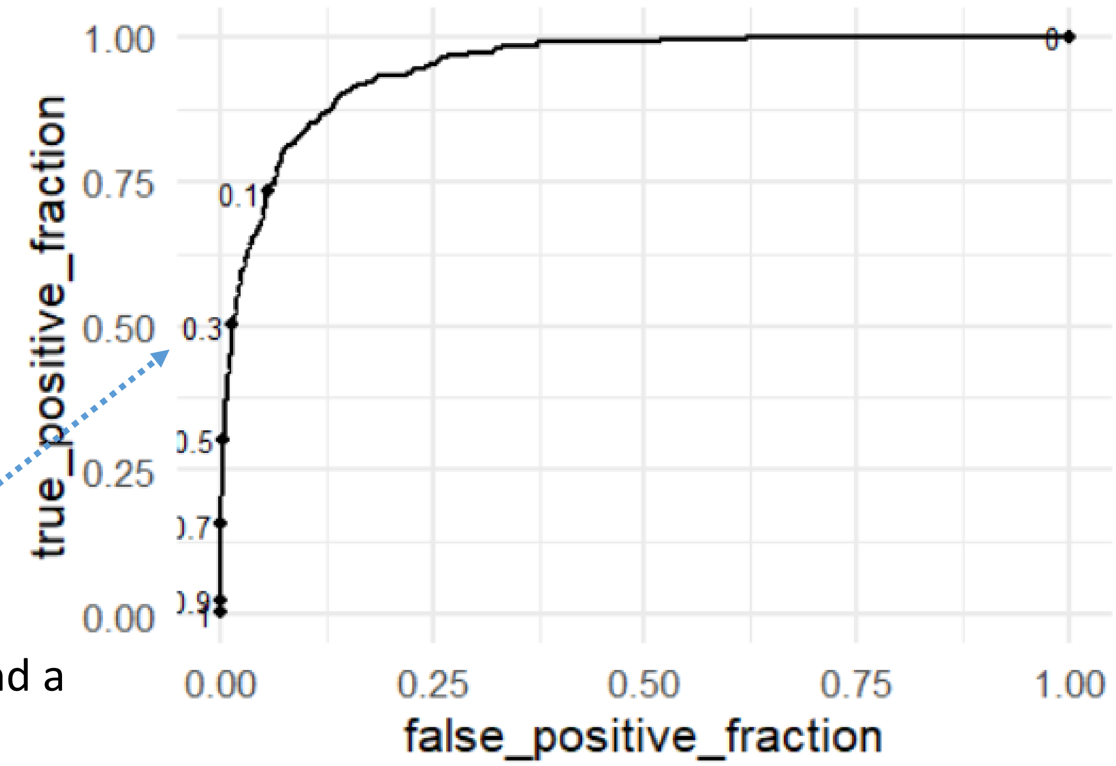
- Can we show consequences of FPs and FNs as we vary the cutoff probability to assign classes?
- Idea of a ROC (Receiver Operator Curve) plot



Cutoff	TPR	FPR
0.01	100%	22.6%
0.4	57%	0.008
0.5	21.4%	0.004%
0.6	21.4%	0.002%

ROC Curves in R

```r
#-------------------------------------------------------
# ROC plots
#-------------------------------------------------------
library('ggplot2')
library('plotROC')

p <- ggplot(results_logit,
       aes(m = Class1, d = truth)) +
  geom_roc(labelsize = 3.5,
           cutoffs.at =
             c(0.99,0.9,0.7,0.5,0.3,0.1,0)) +
  theme_minimal(base_size = 16)
print(p)
```



- At a cutoff of 0.3, we get a true positive fraction of 0.5 and a false positive fraction of a very low number
- Better models lie up and to the left in the ROC plot
- AUC calculates how much total area is under a particular curve
- AUC of 0.947 is pretty good

```
> calc_auc(p)
  PANEL group         AUC
1     1    -1 0.9479842
```

Lab (time permitting)

```
#-----------------------------------------------------------------
# Exercises
#-----------------------------------------------------------------
# 1.  Generate predictions using your logit_mod2 model
#     that predicts default as a function of
#     student, balance, and income
# 2.  Generate predicted probabilities (score the model)
# 3.  Create a results data frame and print a confusion
#     matrix using the results data
# 4.  Plot a ROC curve using the results data
# 5.  How well does the model perform?
```

Class 5 Summary

- Log transformations of Y or X variables is useful when the data are "spread out"
- We interpret log-log regression coefficients as elasticities: a 1% change in the X variable leads to a coefficient % change in the Y variable
- We split data into testing and training sets, estimate a model on the training set and evaluate on the test set
- Logit functions compress predictions to lie between 0 and 1, which are valid probabilities
- The logistic model models the outcome (Y) as the log odds ratio!
- Confusion matrices show the true/false positives/negatives.
- ROC plots measure the consequence on true positive fraction and false positive fraction for different cutoff probabilities
- Higher AUC scores mean a better ROC plot indicating a better model