Introduction to Data Science: Logistic Regression

Héctor Corrada Bravo

University of Maryland, College Park, USA

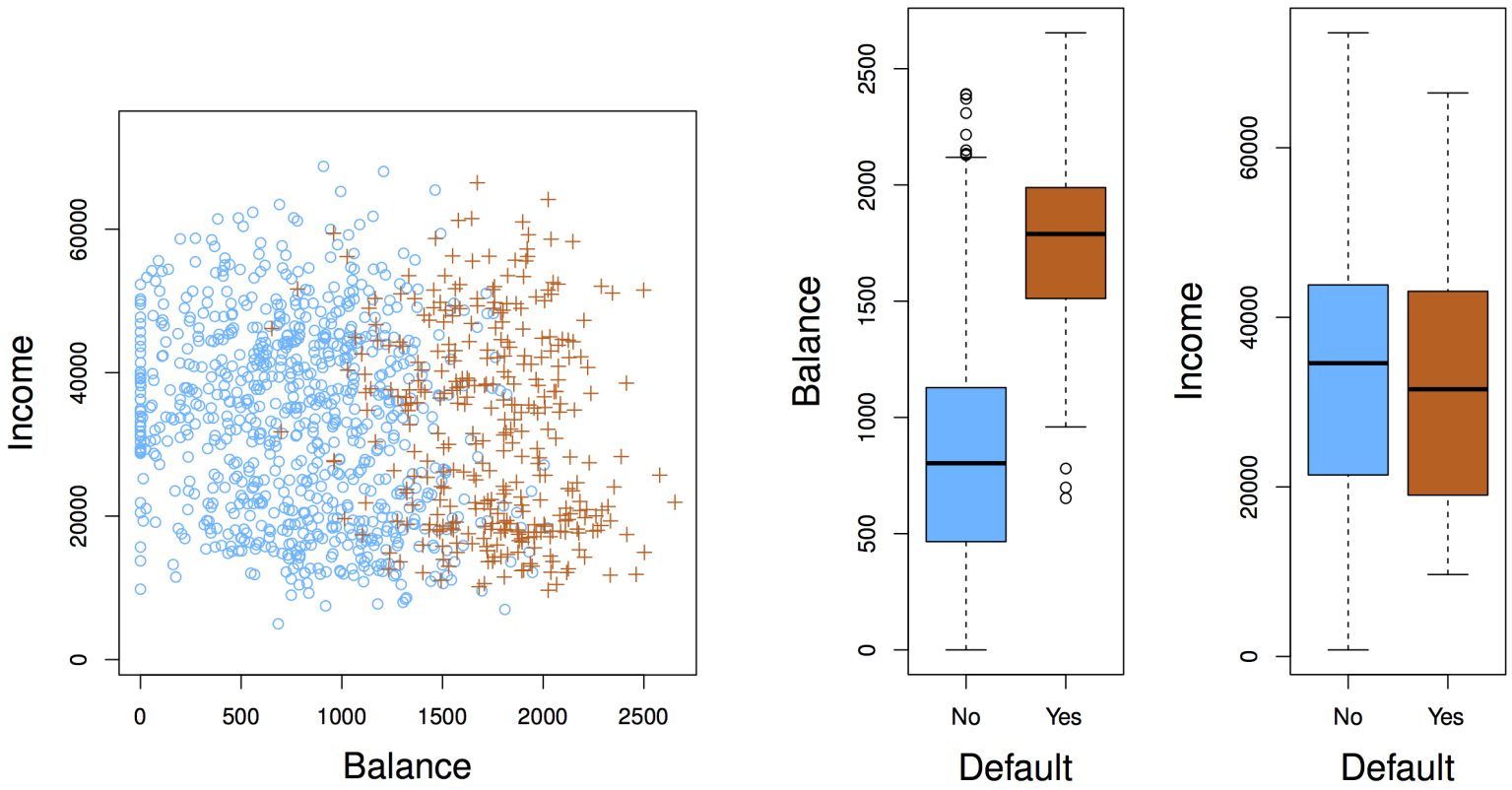
2020-04-05

Linear models for classification

The general classification setting is: can we predict categorical response/output Y, from set of predictors $X_1, X_2, \ldots, X_p$?

As in the regression case, we assume training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$. In this case, however, responses y_i are categorical and take one of a fixed set of values.

Linear models for classification



Linear models for classification

An example classification problem

An individual's choice of transportation mode to commute to work.

Predictors: income, cost and time required for each of the alternatives: driving/carpooling, biking, taking a bus, taking the train.

Response: whether the individual makes their commute by car, bike, bus or train.

Linear models for classification

Why not linear regression?

Why can't we use linear regression in the classification setting.

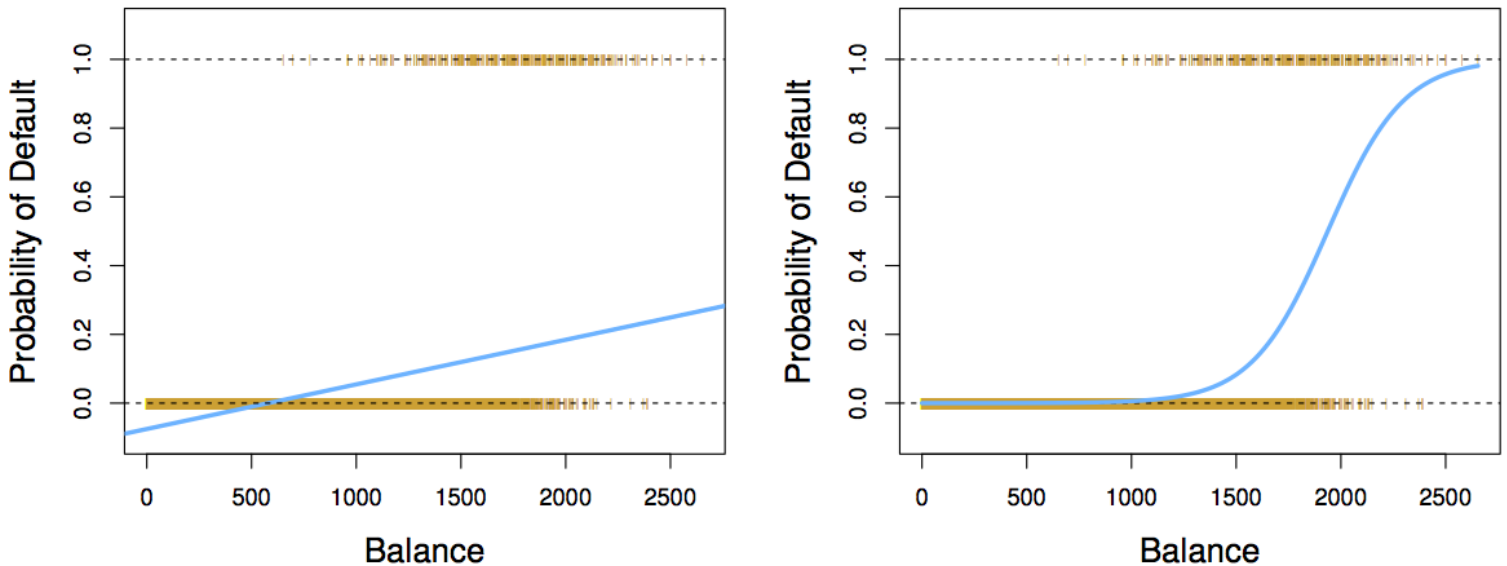
For categorical responses with more than two values, if order and scale (units) don't make sense, then it's not a regression problem

Linear models for classification

For **binary** (0/1) responses, it's a little better.

We could use linear regression in this setting and *interpret* response Y as a probability (e.g, if $\hat{y} > 0.5$ predict **`drugoverdose`**)

Linear models for classification



Linear models for classification

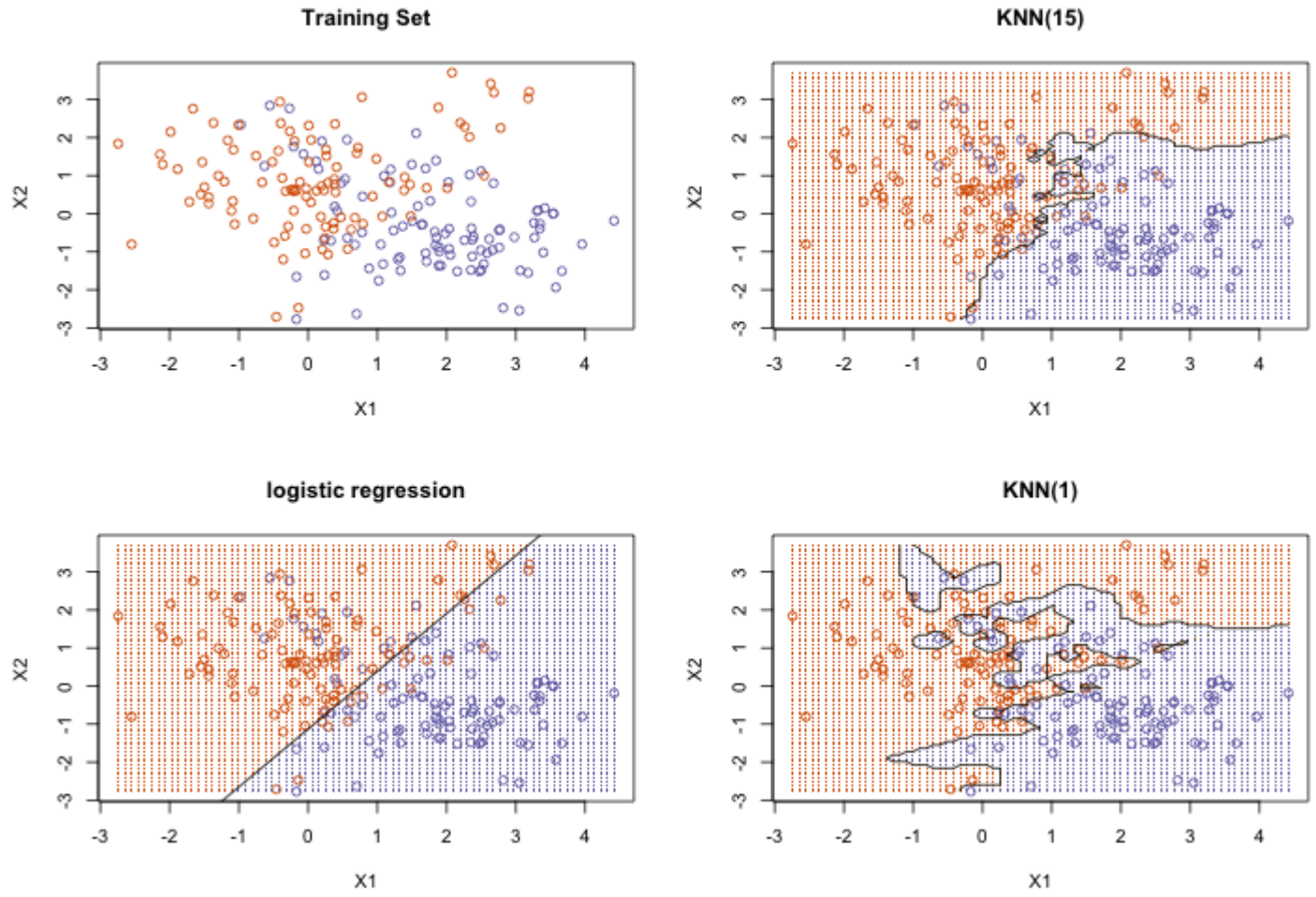
Classification as probability estimation problem

Instead of modeling classes 0 or 1 directly, we will model the conditional class probability $p(Y = 1|X = x)$, and classify based on this probability.

In general, classification approaches use *discriminant* (think of *scoring*) functions to do classification.

Logistic regression is **one** way of estimating the class probability $p(Y = 1|X = x)$ (also denoted $p(x)$)

Linear models for classification



Linear models for classification

Logistic regression

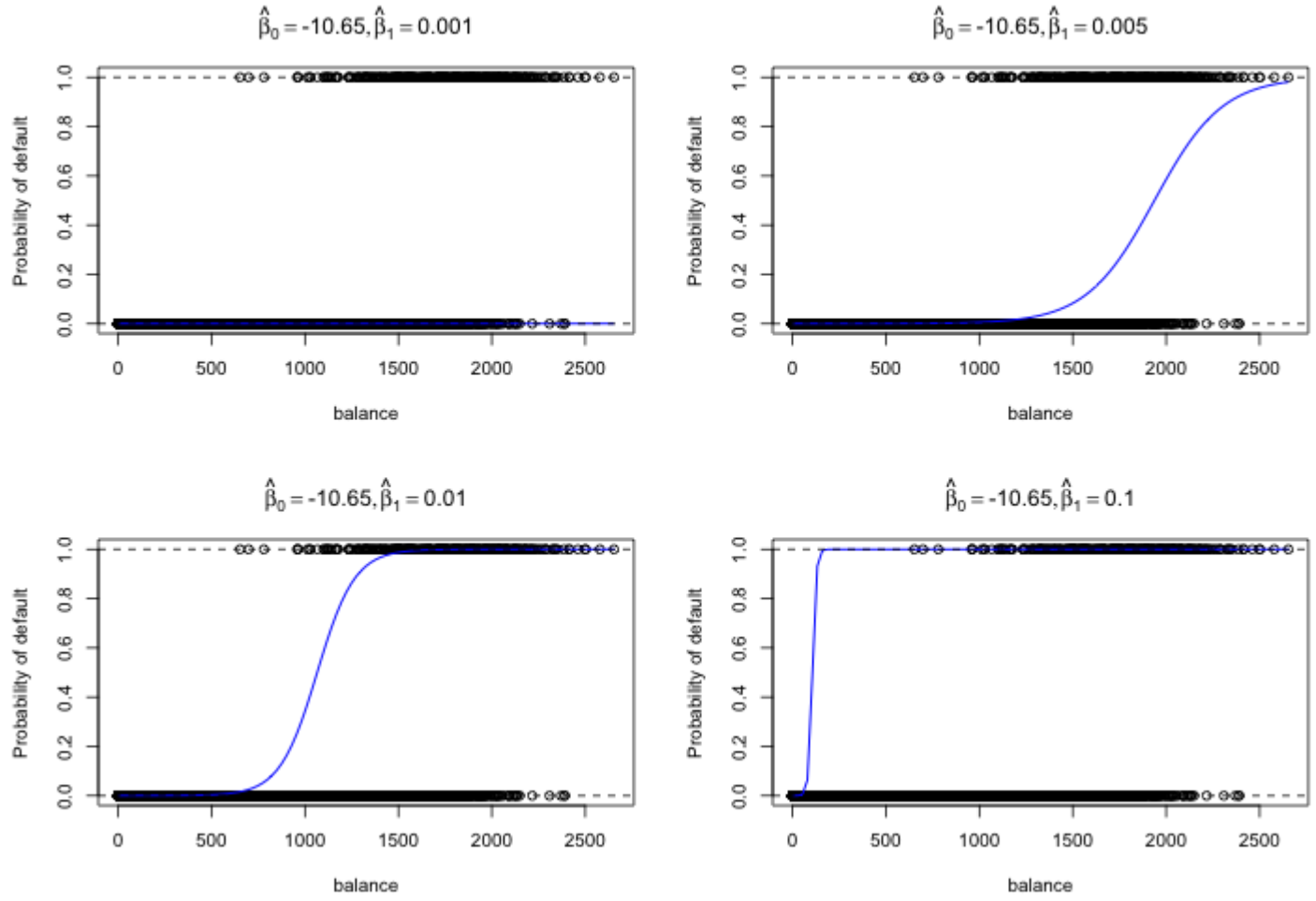
The basic idea behind *logistic regression* is to build a **linear** model *related* to $p(x)$, since linear regression directly (i.e. $p(x) = \beta_0 + \beta_1 x$) doesn't work.

Linear models for classification

Instead we build a linear model of *log-odds*:

$$\log \frac{p(x)}{1-p(x)} = \beta_0 + \beta_1 x$$

Linear models for classification



Linear models for classification

Here is how we compute a logistic regression model in R

```
default_fit <- glm(default ~ balance, data=Default, family=binomial)

default_fit %>%

  tidy()
```

```
## # A tibble: 2 x 5

##   term  estimate std.error statistic

##   <chr>    <dbl>     <dbl>     <dbl>

## 1 (Int… -1.07e+1  0.361        -29.5

## 2 bala…  5.50e-3  0.000220      25.0

## # … with 1 more variable: p.value <dbl>
```

Linear models for classification

Interpretation of logistic regression models is slightly different than the linear regression model we looked at.

In this case, the **odds** that a person defaults increase by $e^{0.05} \approx 1.051$ for every dollar in their account balance.

Linear models for classification

As before, the **accuracy** of $\hat{\beta}_1$ as an estimate of the **population** parameter is given its standard error.

We can again construct a confidence interval for this estimate as we've done before.

Linear models for classification

As before, we can do hypothesis testing of a relationship between account balance and the probability of default.

In this case, we use a Z-statistic $\frac{\hat{\beta}_1}{\text{SE}(\hat{\beta}_1)}$ which plays the role of the t-statistic in linear regression: a scaled measure of our estimate (signal / noise).

Linear models for classification

As before, the P-value is the probability of seeing a Z-value as large (e.g., 24.95) under the null hypothesis that **there is no relationship between balance and the probability of defaulting**, i.e., $\beta_1 = 0$ in the population.

Linear models for classification

We require an algorithm required to *estimate* parameters β_0 and β_1 according to a data fit criterion.

In logistic regression we use the **Bernoulli** probability model we saw previously (think of flipping a coin weighted by $p(x)$), and *estimate* parameters to **maximize** the *likelihood* of the observed training data under this coin flipping (binomial) model.

Linear models for classification

Usually, we do this by *minimizing* the negative of the log likelihood of the model. I.e.: solve the following optimization problem

$$\min_{\beta_0,\beta_1} \sum_{i:\, y_i=1} -y_i f(x_i) + \log(1 + e^{f(x_i)})$$

where $f(x_i) = \beta_0 + \beta_1 x_i$. This is a non-linear (but convex) optimization problem.

Linear models for classification

Making predictions

We can use a learned logistic regression model to make predictions.
E.g., "on average, the probability that a person with a balance of $1,000 defaults is":

$$\hat{p}(1000) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1\times 1000}}{1+e^{\beta_0+\beta_1\times 1000}} \approx \frac{e^{-10.6514+0.0055\times 1000}}{1+e^{-10.6514+0.0055\times 1000}} \approx 0.00576$$

Linear models for classification

Multiple logistic regression

This is a classification analog to linear regression:

$$\log \frac{p(\mathbf{x})}{1-p(\mathbf{x})} = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

Linear models for classification

```
fit <- glm(default ~ balance + income + student, data=Default, family="binomial")

fit %>%

  tidy()
```

```
## # A tibble: 4 x 5

##   term  estimate std.error statistic

##   <chr>    <dbl>     <dbl>     <dbl>

## 1 (Int… -1.09e+1   4.92e-1   -22.1

## 2 bala…  5.74e-3   2.32e-4    24.7

## 3 inco…  3.03e-6   8.20e-6     0.370

## 4 stud… -6.47e-1   2.36e-1    -2.74

## # … with 1 more variable: p.value <dbl>
```

Linear models for classification

As in multiple linear regression it is essential to avoid **confounding!**.

Linear models for classification

Consider an example of single logistic regression of default vs. student status:

```
fit1 <- glm(default ~ student, data=Default, family="binomial")

fit1 %>% tidy()
```

```
## # A tibble: 2 x 5

##   term  estimate std.error statistic p.value

##   <chr>    <dbl>     <dbl>     <dbl>   <dbl>

## 1 (Int…   -3.50     0.0707    -49.6  0.

## 2 stud…    0.405    0.115       3.52 4.31e-4
```

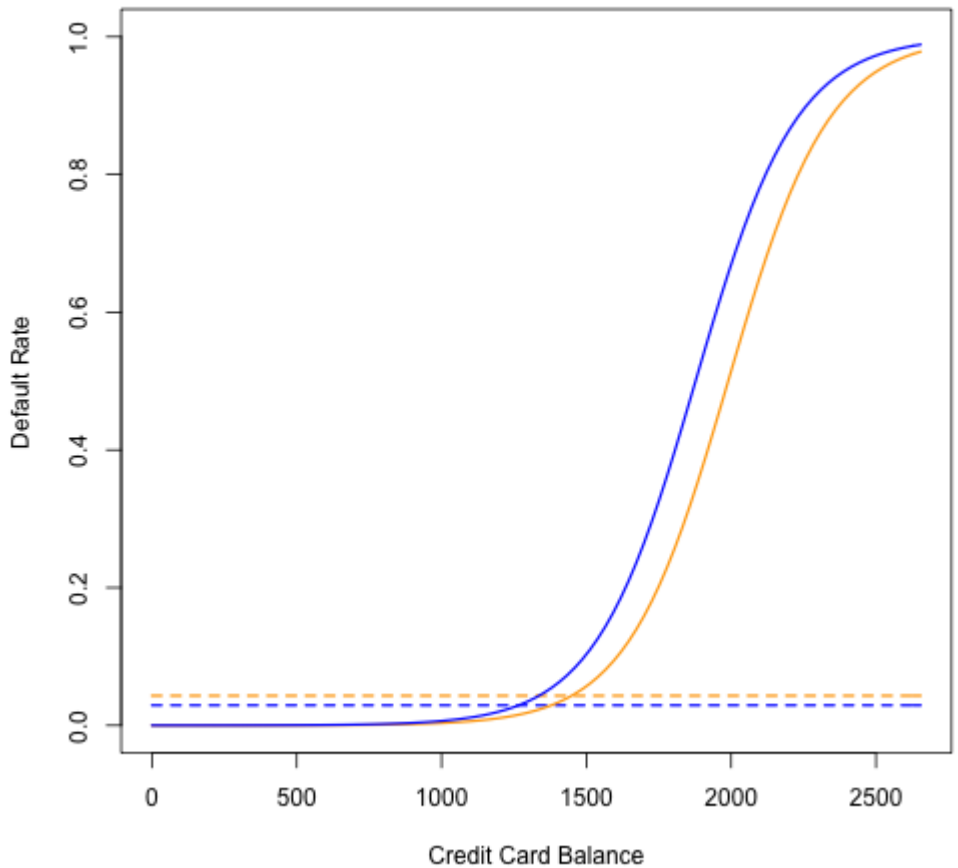
Linear models for classification

and a multiple logistic regression:

```
fit2 <- glm(default ~ balance + income + student, data=Default, family="binomial")

fit2 %>% tidy()
```

```
## # A tibble: 4 x 5

##   term  estimate std.error statistic

##   <chr>    <dbl>     <dbl>     <dbl>

## 1 (Int… -1.09e+1   4.92e-1   -22.1

## 2 bala…  5.74e-3   2.32e-4    24.7

## 3 inco…  3.03e-6   8.20e-6     0.370

## 4 stud… -6.47e-1   2.36e-1    -2.74
```

Linear models for classification



Classifier evaluation

How do we determine how well classifiers are performing?

One way is to compute the *error rate* of the classifier, the percent of mistakes it makes when predicting class

Classifier evaluation

We need a more precise language to describe classification mistakes:

	True Class +	**True Class -**	**Total**
Predicted Class +	True Positive (TP)	False Positive (FP)	P*
Predicted Class -	False Negative (FN)	True Negative (TN)	N*
Total	P	N	

Classifier evaluation

Using these we can define statistics that describe classifier performance

Name	Definition	Synonyms
False Positive Rate (FPR)	FP / N	Type-I error, 1-Specificity
True Positive Rate (TPR)	TP / P	1 - Type-II error, power, sensitivity, **recall**
Positive Predictive Value (PPV)	TP / P*	**precision**, 1-false discovery proportion
Negative Predicitve Value (NPV)	FN / N*	

Classifier evaluation

In the credit default case we may want to increase **TPR** (recall, make sure we catch all defaults) at the expense of **FPR** (1-Specificity, clients we lose because we think they will default)

Classifier evaluation

This leads to a natural question: Can we adjust our classifiers TPR and FPR?

Remember we are classifying `Yes` if

$$\log \frac{P(Y = \texttt{Yes}|X)}{P(Y = \texttt{No}|X)} > 0 \Rightarrow \ P(Y = \texttt{Yes}|X) > 0.5$$

What would happen if we use $P(Y = \texttt{Yes}|X) > 0.2$?

Classifier evaluation

A way of describing the TPR and FPR tradeoff is by using the **ROC curve** (Receiver Operating Characteristic) and the **AUROC** (area under the ROC)

Another metric that is frequently used to understand classification errors and tradeoffs is the precision-recall curve:

Summary

We approach classification as a class probability estimation problem.

Logistic regression partition predictor space with linear functions.

Logistic regression learns parameter using Maximum Likelihood (numerical optimization)

Summary

Error and accuracy statistics are not enough to understand classifier performance.

Classifications can be done using probability cutoffs to trade, e.g., TPR-FPR (ROC curve), or precision-recall (PR curve).

Area under ROC or PR curve summarize classifier performance across different cutoffs.