

CS/ECE 148 –

Data Science Fundamentals

Logistic Regression Extensions
and Evaluating Classification

UCLA Computer Science

Lecture Outline

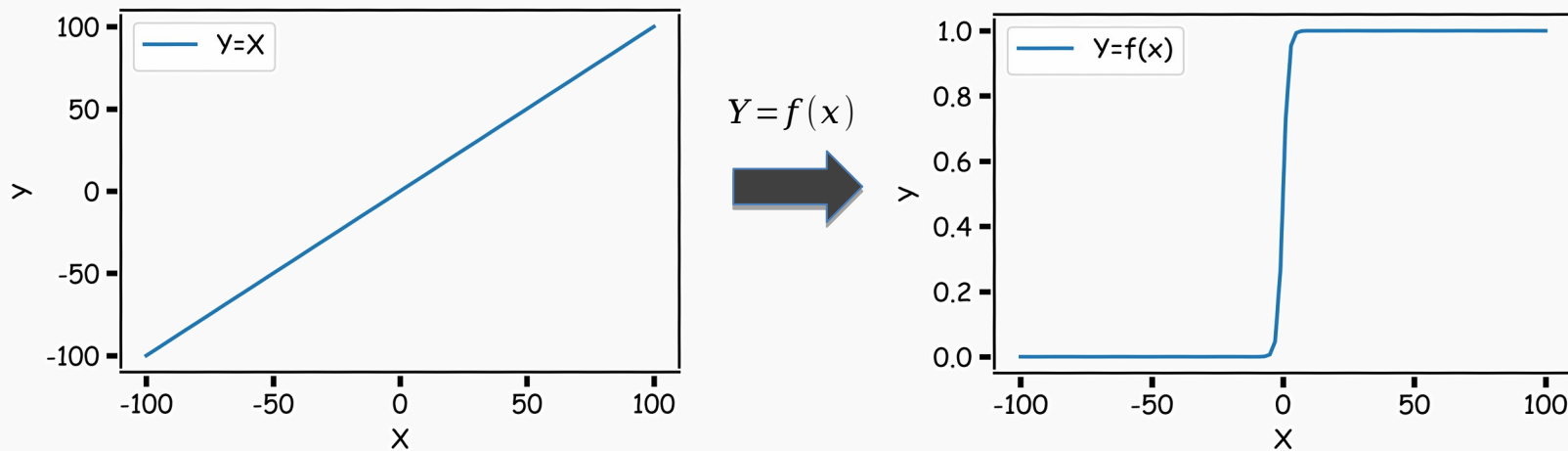
- Logistic Regression: a Brief Review
- Classification Boundaries
- Regularization in Logistic Regression
- Multinomial Logistic Regression
- Bayes Theorem and Misclassification Rates
- ROC Curves

Logistic Regression Review

Logistic Regression

Logistic Regression addresses the problem of The logistic regression model uses a function, called the **logistic** function, to model :

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



Logistic Regression

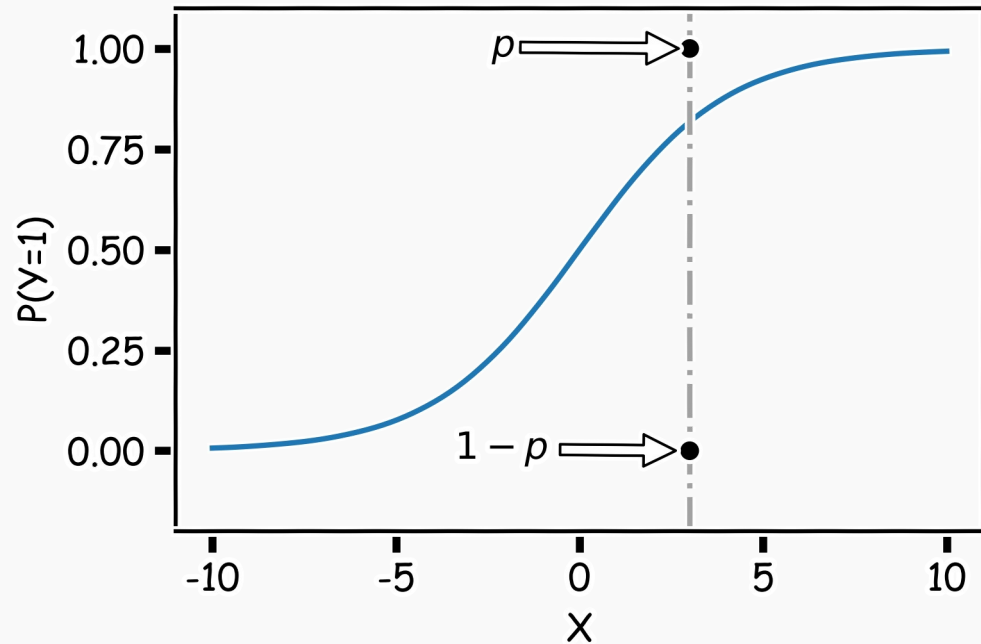
With a little bit of algebraic work, the logistic model can be rewritten as:

$$\ln \left(\frac{P(Y = 1)}{1 - P(Y = 1)} \right) = \beta_0 + \beta_1 X.$$

The value inside the natural log function , is called the

This gives us the natural interpretation of the estimates similar to linear regression: a one unit change in is associated with a change in the log-odds of ; or better yet, a one unit change in is associated with a factor of change in the odds that .

Estimation in Logistic Regression



Probability
Probability

where:

and therefore p depends on X .
Thus not every p is the same for each individual measurement.

Likelihood

The likelihood of a single observation for p given x and y is:

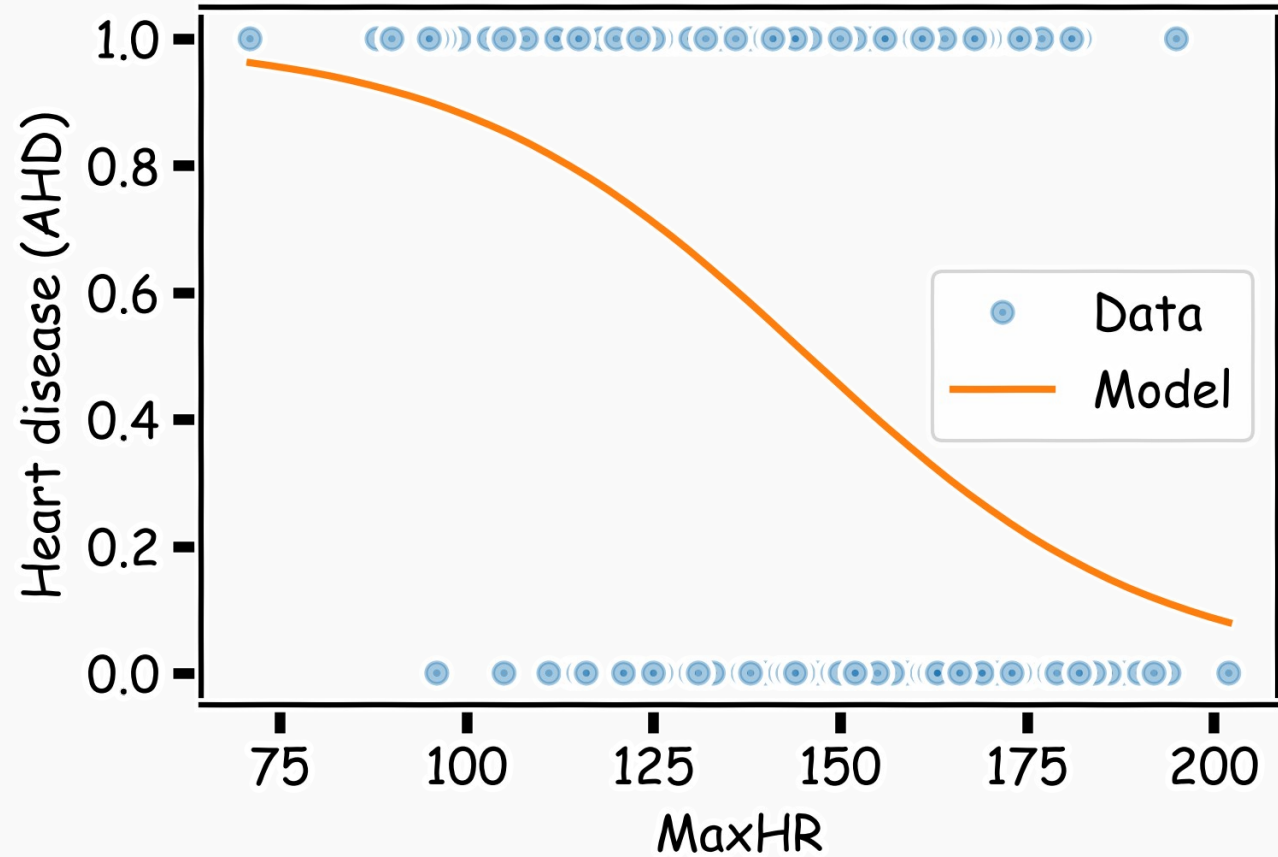
$$L(p_i | Y_i) = P(Y_i = y_i) = p_i^{y_i} (1 - p_i)^{1 - y_i}$$

Given the observations are independent, what is the likelihood function for p ?

$$L(p | Y) = \prod_i P(Y_i = y_i) = \prod_i p_i^{y_i} (1 - p_i)^{1 - y_i}$$

$$l(p | Y) = -\log L(p | Y) = - \sum_i y_i \log p_i + (1 - y_i) \log (1 - p_i)$$

Heart Data: logistic estimation



Multiple Logistic Regression: the model

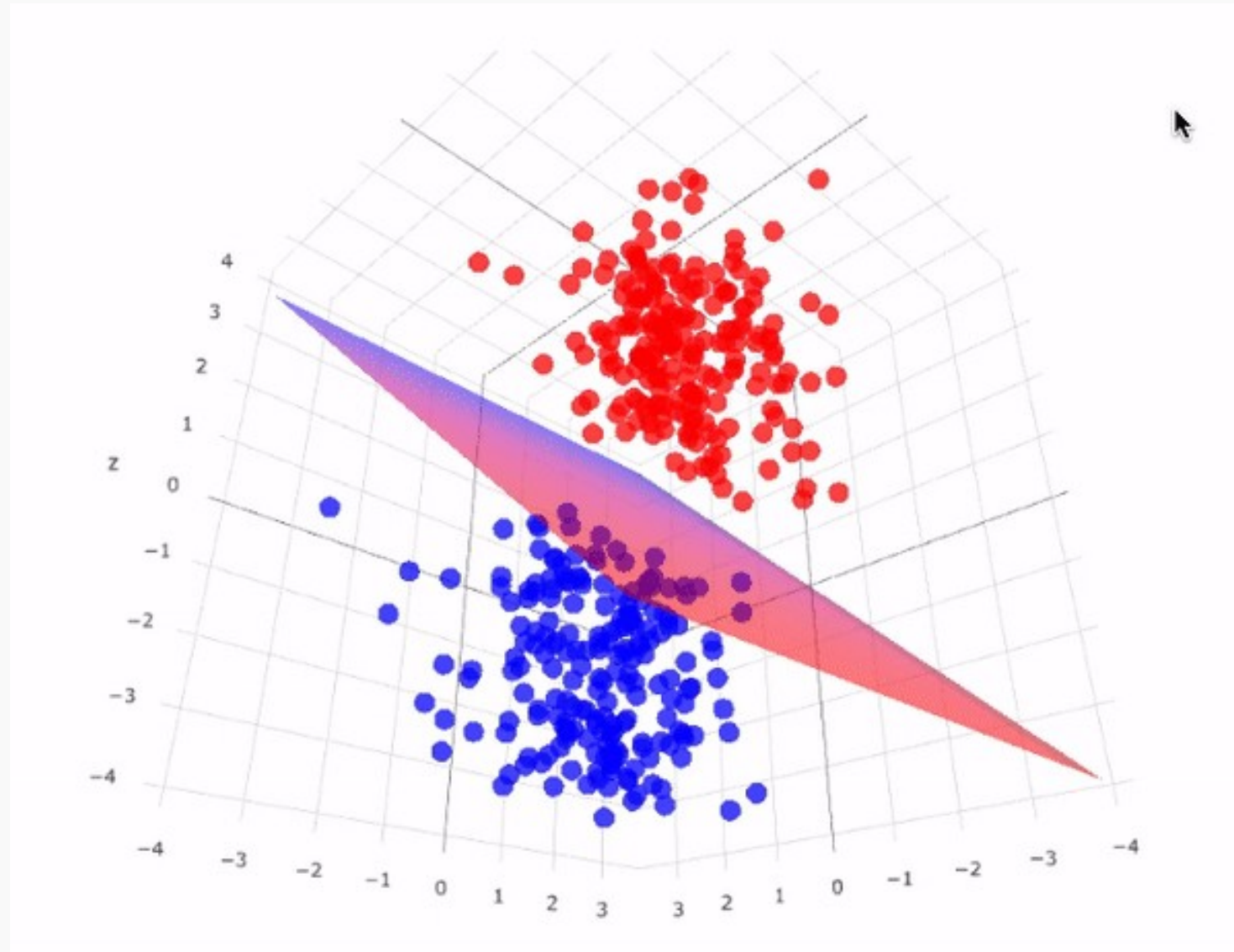
Last time we saw the general form of the multiple logistic regression model. Specifically we can define a multiple logistic regression model to predict $P(Y = 1)$ as such:

$$\log \left(\frac{P(Y = 1)}{1 - P(Y = 1)} \right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p$$

where there are p predictors: $\mathbf{X} = (X_1, X_2, \dots, X_p)$.

Note: statisticians are often lazy and use the notation “log” to mean “ln” (the text does this). We will write \log_{10} if this is what we mean.

Multiple Logistic Regression

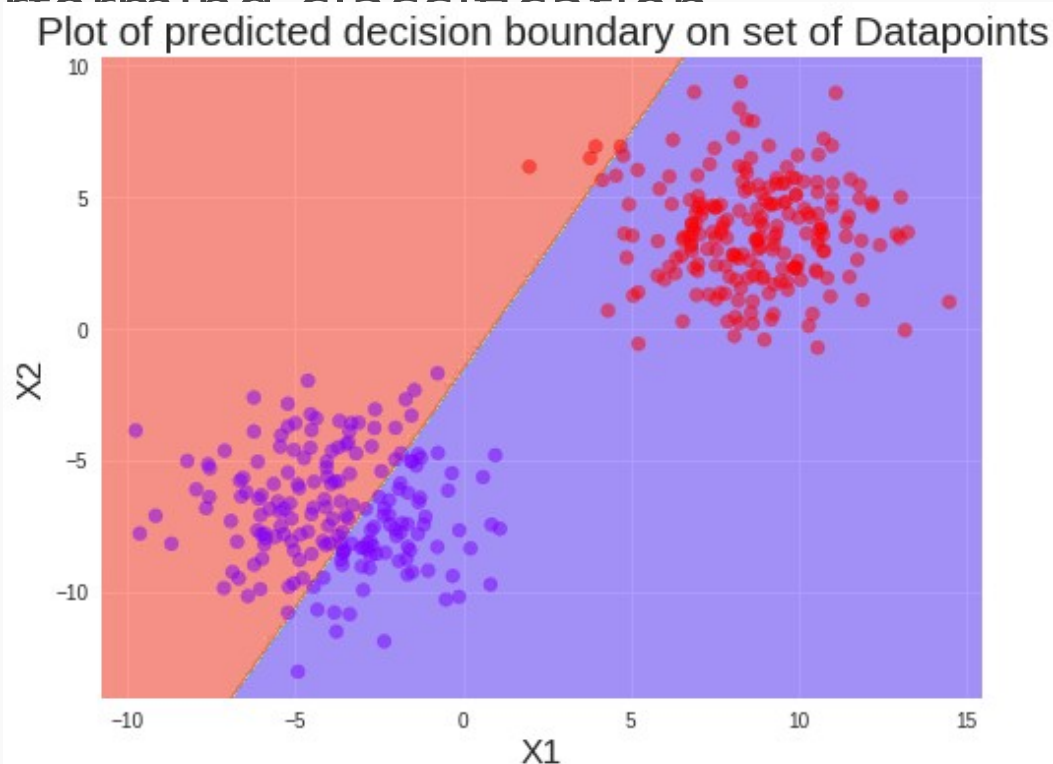


Classification Boundaries

Classification boundaries

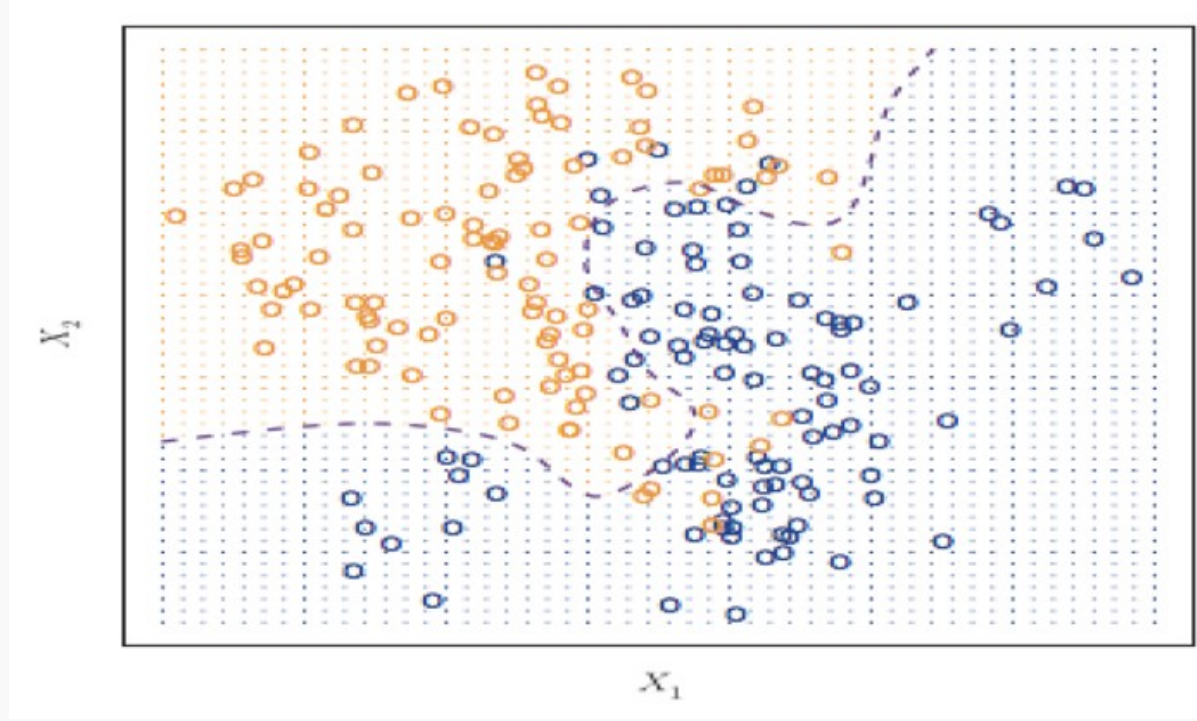
Recall that we could attempt to purely classify each observation based on whether the estimated from the model was greater than 0.5.

When dealing with 'well-separated' data, logistic regression can work well in performing classification

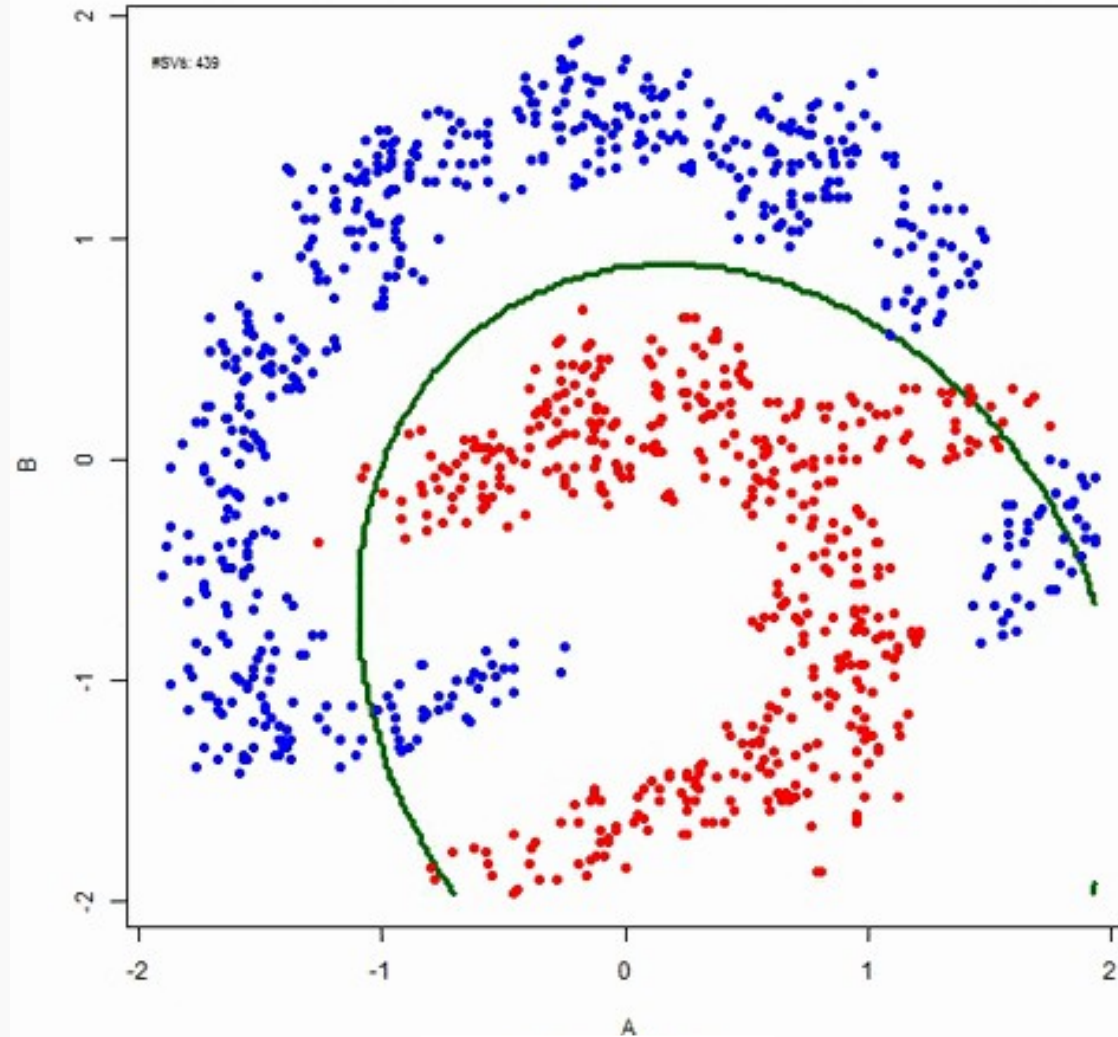


Classification boundaries

We saw a 2-D plot last time which had two predictors, and depicted the classes as different colors. A similar one is shown on the next slide.



Classification boundaries



Regularization in Logistic Regression

Review: Regularization in Linear Regression

Based on the Likelihood framework, a loss function can be determined based on the likelihood function.

In linear regression maximizing the log-likelihood is equivalent to minimizing the sum of squares error:

$$\arg \min \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \arg \min \sum_{i=1}^n (y_i - (\beta_0 + \beta_1 x_{1i} + \dots + \beta_p x_{pi}))^2$$

Review: Regularization in Linear Regression

And a regularization approach was to add a penalty factor to this equation. Which for Ridge Regression becomes:

$$\arg \min \left[\sum_{i=1}^n \left(y_i - \left(\beta_0 + \sum_{j=1}^n \beta_j x_{ji} \right) \right)^2 + \lambda \sum_{j=1}^n \beta_j^2 \right]$$

Note: this penalty *shrinks* the estimates towards zero.

Loss function in Logistic Regression

A similar approach can be used in logistic regression. Here, maximizing the log-likelihood is equivalent to minimizing the following loss function:

$$\arg \min_{\beta_0, \beta_1, \dots, \beta_p} \left[- \sum_{i=1}^n \left(y_i \ln(p_i) + (1 - y_i) \ln(1 - p_i) \right) \right]$$

where

Regularization in Logistic Regression

A penalty factor can then be added to this loss function and results in a new loss function that penalizes large values of the parameters:

$$\arg \min_{\beta_0, \beta_1, \dots, \beta_p} \left[- \sum_{i=1}^n \left(y_i \ln(p_i) + (1 - y_i) \ln(1 - p_i) \right) + \lambda \sum_{j=1}^p \beta_j^2 \right]$$

The result is just like in linear regression: shrink the parameter estimates towards zero.

In practice, the intercept is usually not part of the penalty factor.

Note: the sklearn package uses a different tuning parameter: instead of λ they use a constant that is essentially $\frac{1}{2\lambda}$.

Regularization in Logistic Regression: an Example

Let's see how this plays out in an example in logistic regression

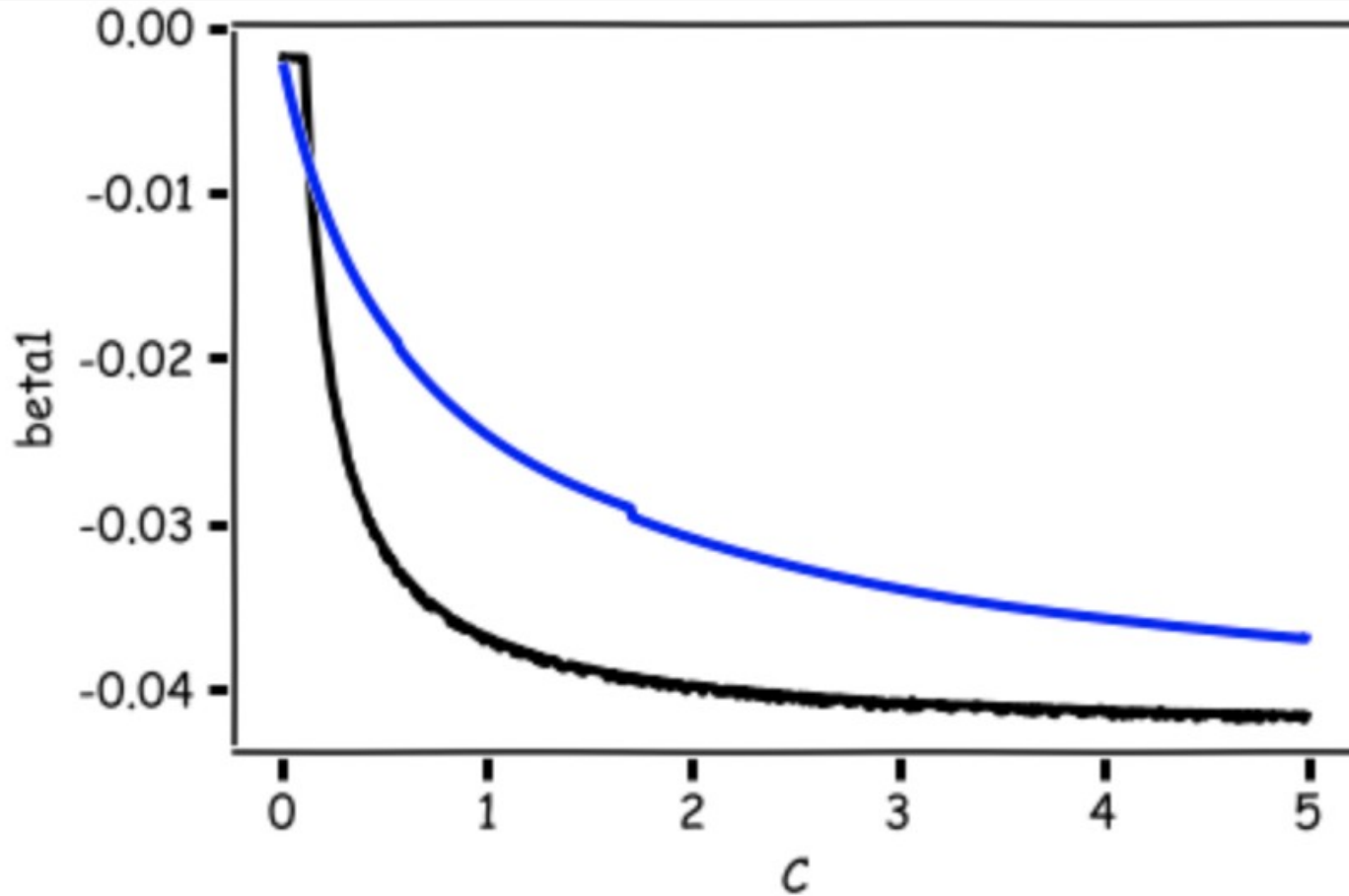
```
beta1_l1 = []
beta1_l2 = []
Cs = []
data_x = df_heart[['MaxHR']]
data_y = df_heart['AHD']

for i in range(1, 500):
    C = i/100
    logitm_l1 = sk.LogisticRegression(C = C, penalty = "l1")
    logitm_l1.fit(data_x, data_y)
    logitm_l2 = sk.LogisticRegression(C = C, penalty = "l2")
    logitm_l2.fit(data_x, data_y)
    beta1_l1.append(logitm_l1.coef_[0])
    beta1_l2.append(logitm_l2.coef_[0])
    Cs.append(C)

plt.plot(Cs, beta1_l1, color='black', lw=3)
plt.plot(Cs, beta1_l2, color='blue', lw=3)
plt.xlabel("C")
plt.ylabel("beta1")
plt.show()
```

Regularization in Logistic Regression: an Example

Let's see how this plays out in an example in logistic regression



Regularization in Logistic Regression: tuning

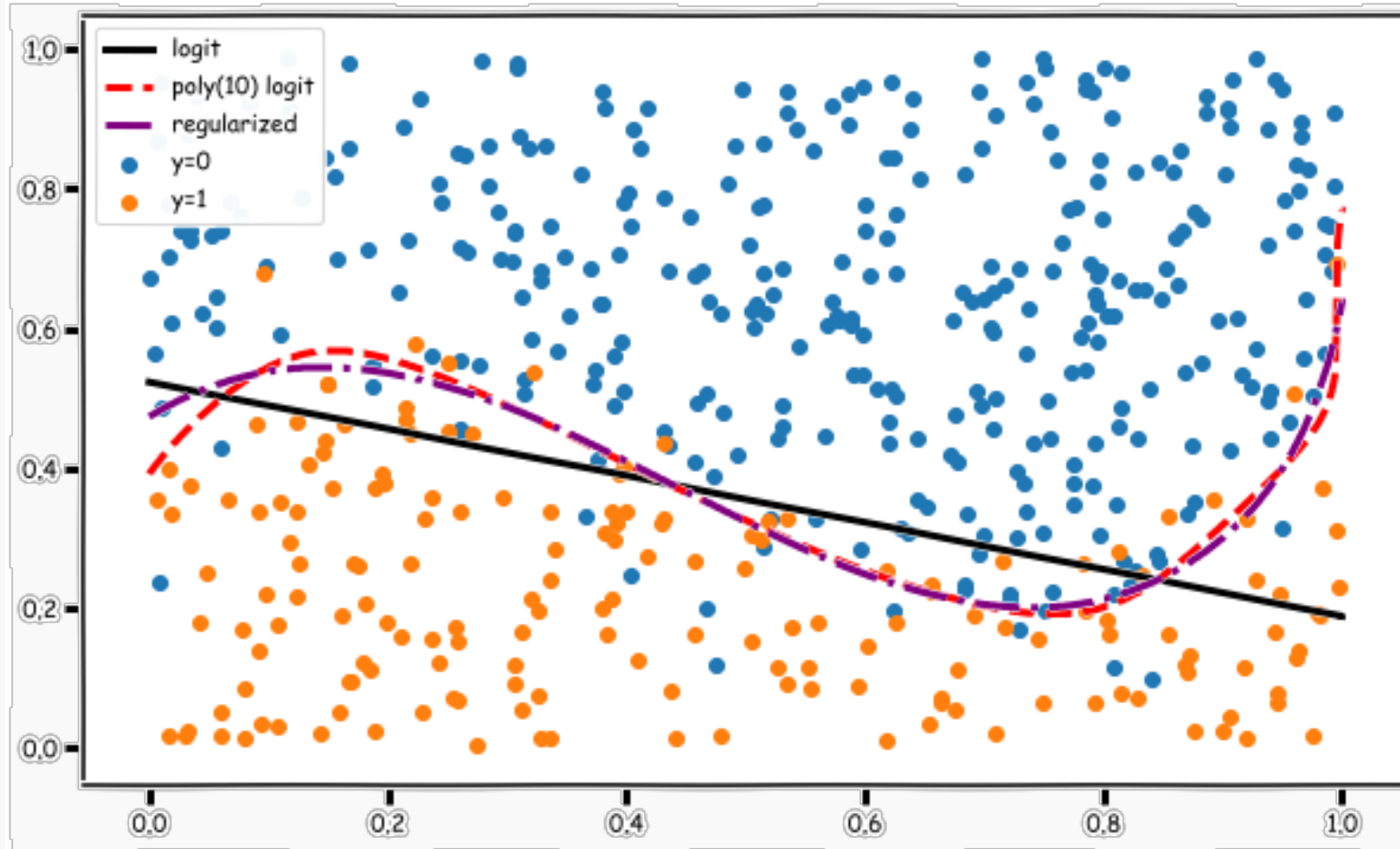
Just like in linear regression, the shrinkage factor must be chosen. How should we go about doing this?

Through building multiple training and test sets (through k -fold or random subsets), we can select the best shrinkage factor to mimic out-of-sample prediction.

How could we measure how well each model fits the test sets?

We should measure this based on some loss function!

Regularized Decision Boundaries



Multinomial Logistic Regression

Logistic Regression for predicting more than 2 Classes

There are several extensions to standard logistic regression when the response variable Y has more than 2 categories.

The two most common are:

1. ordinal logistic regression
2. multinomial logistic regression.

Ordinal logistic regression is used when the categories have a specific **hierarchy** (like class year: Freshman, Sophomore, Junior, Senior; or a 7-point rating scale from strongly disagree to strongly agree).

Multinomial logistic regression is used when the categories have **no inherent order** (like eye color: blue, green, brown, hazel, et...).

Multiclass Logistic Regression

There are two common approaches to estimating a nominal (not-ordinal) categorical variable that has more than 2 classes.

For example we could attempt to predict a student's concentration:

$$y = \begin{cases} 1 & \text{if Computer Science (CS)} \\ 2 & \text{if Statistics} \\ 3 & \text{otherwise} \end{cases}$$

from predictors x_1 number of psets per week and x_2 how much time spent in Library.

One vs. Rest (ovr) Logistic Regression

The default multiclass logistic regression model is called the '**One vs. Rest**' approach.

If there are 3 classes, then 3 separate logistic regressions are fit, where the probability of each category is predicted over the rest of the categories combined. So for the concentration example, 3 models would be fit:

- a first model would be fit to predict CS from (Stat and Others) combined.
- a second model would be fit to predict Stat from (CS and Others) combined.
- a third model would be fit to predict Others from (CS and Stat) combined.

OVR Logistic Regression in Python

In [293]:

Slide Type



```
# 'Multinomial' Logistic Regression Example

data_x = df_heart[['Sex']]

# 0 = normal; 1 = having ST-T; 2 = hypertrophy
data_y = df_heart['RestECG']

logitm = LogisticRegression(C = 10000000,solver='lbfgs',multi_class='ovr')
logitm.fit(data_x, data_y)

# The coefficients
print('Estimated beta1: \n', logitm.coef_)
print('Estimated beta0: \n', logitm.intercept_)
```

```
Estimated beta1:
[[-0.04003728]
 [-1.87833213]
 [ 0.1445807 ]]
Estimated beta0:
[ 0.02061932 -3.44468483 -0.14458062]
```

Multiclass Logistic Regression (cont.)

The second approach sets one of the categories in the response variable as the *reference* group, and then fits separate logistic regression models to predict the other cases based off of the reference group.

We could select the $y = 3$ case as the reference group (other concentration), and then fit two separate models: a model to predict $y = 1$ (CS) from $y = 3$ (others) and a separate model to predict $y = 2$ (Stat) from $y = 3$ (others).

Ignoring interactions, how many parameters would need to be estimated?

How could these models be used to estimate the probability of an individual falling in each concentration?

'True' Multinomial Logistic Regression

This second approach for multiclass logistic regression model is the *true* 'multinomial' logistic regression model.

One of the classes is chosen as the *baseline group* (think $Y = 0$ group in typical logistic regression), and the other $K - 1$ classes are compared to it. Thus a sequence of models are built to predict being in class k from class K :

Note: sklearn normalizes the calculations by adding a K^{th} set of estimates in the output.

'True' Multinomial Logistic Regression

If outcome K (the last outcome) is chosen as the pivot:

$$\begin{aligned}\ln \frac{\Pr(Y_i = 1)}{\Pr(Y_i = K)} &= \beta_1 \cdot \mathbf{X}_i \\ \ln \frac{\Pr(Y_i = 2)}{\Pr(Y_i = K)} &= \beta_2 \cdot \mathbf{X}_i \\ &\dots\dots\dots \\ \ln \frac{\Pr(Y_i = K-1)}{\Pr(Y_i = K)} &= \beta_{K-1} \cdot \mathbf{X}_i\end{aligned}$$

If we exponentiate both sides, and solve for the probabilities, we get:

$$\begin{aligned}\Pr(Y_i = 1) &= \Pr(Y_i = K)e^{\beta_1 \cdot \mathbf{X}_i} \\ \Pr(Y_i = 2) &= \Pr(Y_i = K)e^{\beta_2 \cdot \mathbf{X}_i} \\ &\dots\dots\dots \\ \Pr(Y_i = K-1) &= \Pr(Y_i = K)e^{\beta_{K-1} \cdot \mathbf{X}_i}\end{aligned}$$

'True' Multinomial Logistic Regression

Using the fact that all K of the probabilities must sum to one, we find:

$$\Pr(Y_i = K) = 1 - \sum_{k=1}^{K-1} \Pr(Y_i = k) = 1 - \sum_{k=1}^{K-1} \Pr(Y_i = K) e^{\beta_k \cdot \mathbf{X}_i} \Rightarrow \Pr(Y_i = K) = \frac{1}{1 + \sum_{k=1}^{K-1} e^{\beta_k \cdot \mathbf{X}_i}}$$

We can use this to find the other probabilities:

$$\begin{aligned}\Pr(Y_i = 1) &= \frac{e^{\beta_1 \cdot \mathbf{X}_i}}{1 + \sum_{k=1}^{K-1} e^{\beta_k \cdot \mathbf{X}_i}} \\ \Pr(Y_i = 2) &= \frac{e^{\beta_2 \cdot \mathbf{X}_i}}{1 + \sum_{k=1}^{K-1} e^{\beta_k \cdot \mathbf{X}_i}} \\ &\dots\dots\dots \\ \Pr(Y_i = K - 1) &= \frac{e^{\beta_{K-1} \cdot \mathbf{X}_i}}{1 + \sum_{k=1}^{K-1} e^{\beta_k \cdot \mathbf{X}_i}}\end{aligned}$$

Multinomial Logistic Regression in Python

In [294]:

Slide Type



```
logitm2 = LogisticRegression(C = 10000000,solver='lbfgs',multi_class='multinomial')
logitm2.fit (data_x, data_y)
```

```
# The coefficients
```

```
print('Estimated beta1: \n', logitm2.coef_)
```

```
print('Estimated beta0: \n', logitm2.intercept_)
```

```
Estimated beta1:
```

```
[[ 0.57896717]
```

```
[-1.25284168]
```

```
[ 0.67387452]]
```

```
Estimated beta0:
```

```
[ 0.95944398 -1.83373675  0.87429277]
```

‘multinomial’ vs. ‘ovr’

There are 2 options in sklearn, so which should we use?

‘multinomial’ is slightly more efficient in estimation since there technically are fewer parameters (though sklearn reports extra ones to normalize the calculations to 1) and is more suitable for inferences/group comparisons.

‘ovr’ is often preferred for determining classification: you simply just predict from all 3 separate models (for each individual) and choose the highest probability.

They give VERY similar results in estimated probabilities and classifications.

Classification for more than 2 Categories

When there are more than 2 categories in the response variable, then there is no guarantee that $P(Y = k) \geq 0.5$ for any one category. So any classifier based on logistic regression will instead have to select the group with the largest estimated probability.

The classification boundaries are then much more difficult to determine. We will not get into the algorithm for drawing these in this class.

Softmax

So how do we convert a set of probability estimates from separate models to one set of probability estimates?

The ***softmax*** function is used. That is, the weights are just normalized for each predicted probability. AKA, predict the 3 class probabilities from each model of the 3 models, and just rescale so they add up to 1.

Mathematically that is:

$$P(\mathbf{y} = k | \vec{\mathbf{x}}) = \frac{e^{\vec{\mathbf{x}}^T \hat{\beta}_k}}{\sum_{j=1}^K e^{\vec{\mathbf{x}}^T \hat{\beta}_j}}$$

where $\vec{\mathbf{x}}$ is the vector of covariates for that observation and $\hat{\beta}_k$ are the associated logistic regression coefficient estimates.

Classification for more than 2 Categories in sklearn

In [307]:

Slide Type



```
X=np.arange(0,2)
print("For OVR Logistic Regression:")
print(logitm.predict_proba(X.reshape(-1,1)))
print(logitm.predict(X.reshape(-1,1)))

print("For Multinomial Logistic Regression:")
print(logitm2.predict_proba(X.reshape(-1,1)))
print(logitm2.predict(X.reshape(-1,1)))
```

For OVR Logistic Regression:

```
[[0.5051546  0.03092776 0.46391764]
 [0.49514565 0.00485434 0.50000001]]
[0 2]
```

For Multinomial Logistic Regression:

```
[[0.50515267 0.03092856 0.46391877]
 [0.49514568 0.00485429 0.50000003]]
[0 2]
```

Bayes Theorem and Misclassification Rates

Bayes' Theorem

We define conditional probability as:

And using the fact that we get the simplest form of **Bayes' Theorem**:

Another version of Bayes' Theorem is found by substituting in the Law of Total Probability (LOTP) into the denominator:

Diagnostic Testing

In the diagnostic testing paradigm, one cares about whether the results of a test (like a classification test) matches truth (the true class that observation belongs to). The simplest version of this is trying to detect disease ($D+$ vs. $D-$) based on a diagnostic test ($T+$ vs. $T-$).

Medical examples of this include various screening tests: breast cancer screening through (i) self-examination and (ii) mammography, prostate cancer screening through (iii) PSA tests, and Colo-rectal cancer through (iv) colonoscopies.

These tests are a little controversial because of poor

Diagnostic Testing (cont.)

Bayes' theorem can be rewritten for diagnostic tests:

These probability quantities can then be defined as:

- *Sensitivity:*
- *Specificity:*
- *Prevalence:*
- *Positive Predictive Value:*
- *Negative Predictive Value:*

Diagnostic Testing

We mentioned that these tests are a little controversial because of their poor predictive probability. When will these tests have poor positive predictive probability?

When the disease is not very prevalent, then the number of 'false positives' will overwhelm the number of true positive. For example, PSA screening for prostate cancer has sensitivity of about 90% and specificity of about 97% for some age groups (men in their fifties), but prevalence is about 0.1%.

What is positive predictive probability for this diagnostic test?

Why do we care?

As data scientists, why do we care about diagnostic testing from the medical world?

Because classification can be thought of as a diagnostic test. Let $Y_i = k$ be the event that observation i truly belongs to category k , and let $\hat{Y}_i = k$ be the event that we correctly predict it to be in class k . Then Bayes' rule

$$P(Y_i = k | \hat{Y}_i = k) = \frac{P(\hat{Y}_i = k | Y_i = k)P(Y_i = k)}{P(\hat{Y}_i = k | Y_i = k)P(Y_i = k) + P(\hat{Y}_i = k | Y_i \neq k)P(Y_i \neq k)}$$

Thus the probability of a predicted outcome truly being in a specific group depends on what? The proportion of observations in that class!

Error in Classification

There are 2 major types of error in classification problems based on a binary outcome. They are:

- False positives: incorrectly predicting when it truly is in .
- False negative: incorrectly predicting when it truly is in .

The results of a classification algorithm are often summarized in two ways:

1. a confusion matrix, sometimes called a contingency table
2. a receiver operating characteristics (ROC) curve

Confusion table

When a classification algorithm (like logistic regression) is used, the results can be summarized in a (2 x 2) table as

such:	Predicted Healthy ()	Predicted AHD ()
Truly Healthy ()	132	32
Truly AHD ()	56	83

The table above was a classification based on a logistic regression model to predict heart disease based on 3 predictors: $X_1 = \text{Sex}$, $X_2 = \text{Resting HR}$, and $X_3 = \text{the interaction between the two}$.

What are the false positive and false negative rates for this classifier?

Bayes' Classifier Choice

A classifier's error rates can be tuned to modify this table.
How?

The choice of the Bayes' classifier level will modify the characteristics of this table.

If we thought it was more important to predict the disease correctly (lower false positive rate), what could we do for our Bayes' classifier level?

We could classify instead based on:

and we could choose to be some level other than 0.5. Let's see what the table looks like if were 0.4587 or 0.75.

Other Confusion tables

Based on $\text{AUC} = 0.4587$:

	Predicted Healthy (0)	Predicted AHD (1)
Truly Healthy (0)	122	42
Truly AHD (1)	47	92

What has improved? What has worsened?

Based on $\text{AUC} = 0.75$:

	Predicted Healthy (0)	Predicted AHD (1)
Truly Healthy (0)	156	8
Truly AHD (1)	89	50

Which should we choose? Why?

ROC Curves

ROC Curves

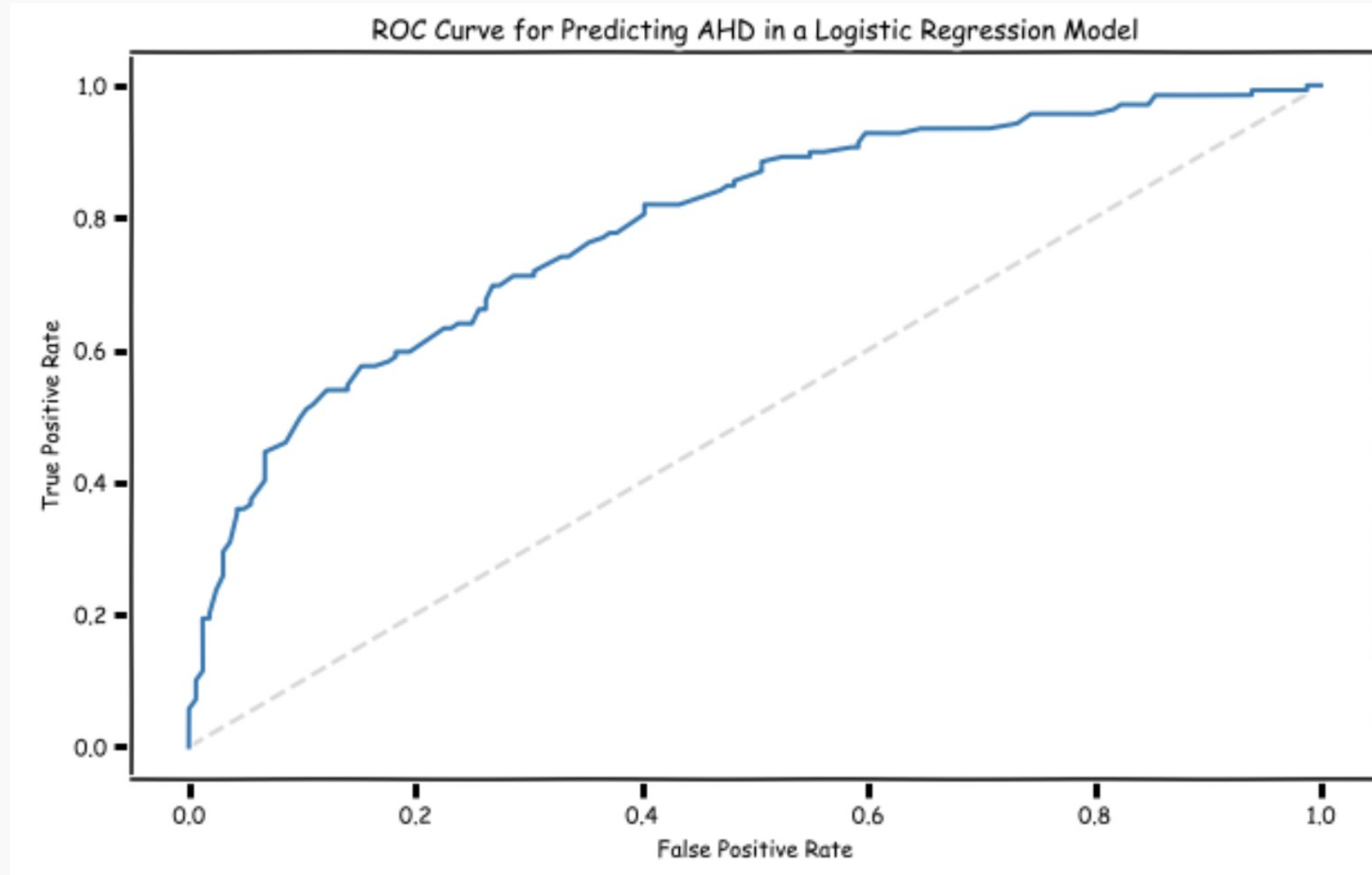
The Radio Operator Characteristics (ROC) curve illustrates the trade-off for all possible thresholds chosen for the two types of error (or correct classification).

The vertical axis displays the true positive rate and the horizontal axis depicts the false positive rate.

What is the shape of an ideal ROC curve?

See next slide for an example.

ROC Curve Example



ROC Curve for measuring classifier performance

The overall performance of a classifier, calculated over all possible thresholds, is given by the area under the ROC curve ('AUC').

An ideal ROC curve will hug the top left corner, so the larger the AUC the better the classifier.

What is the worst case scenario for AUC? What is the best case? What is AUC if we independently just flip a coin to perform classification?

This AUC then can be use to compare various approaches to classification: Logistic regression, LDA (to come), k -NN, etc...