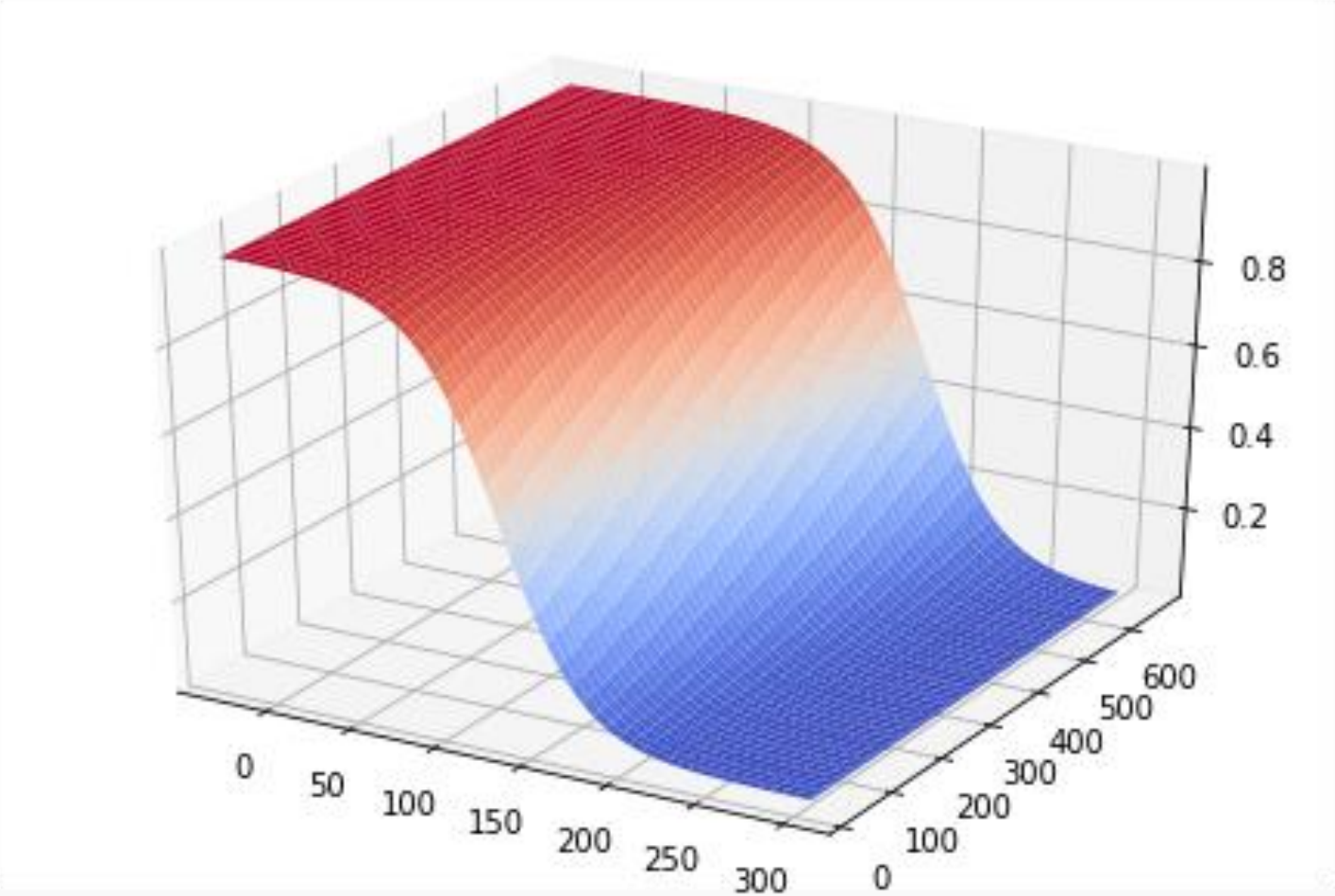
# Lecture #14: Logistic Regression - Part II

CS109OA Introduction to Data Science

Pavlos Protopapas, Natesh Pillai, and Chris Gumb

# Visualizing Multiple Logistic Regression



# Outline

- Interpreting interactions in logistic regression
- Regularization in Logistic Regression
- Multiclass Logistic Regression
    - Multinomial Logistic Regression
    - One-vs-Rest Logistic Regression
- Bayes Theorem and Misclassification Rates
- ROC Curves

# Interactions in Multiple Logistic Regression

Just like in linear regression, interaction terms can be considered in logistic regression. An **interaction terms** is incorporated into the model the same way, and the interpretation is very similar (on the log-odds scale of the response of course).

Write down the model for the Heart data for the 2 predictors plus the interactions term based on the output on the next slide.

Here, we are predicting AHD from Age, Female, and the interaction between the two.

# Interpreting Multiple Logistic Regression: an Example

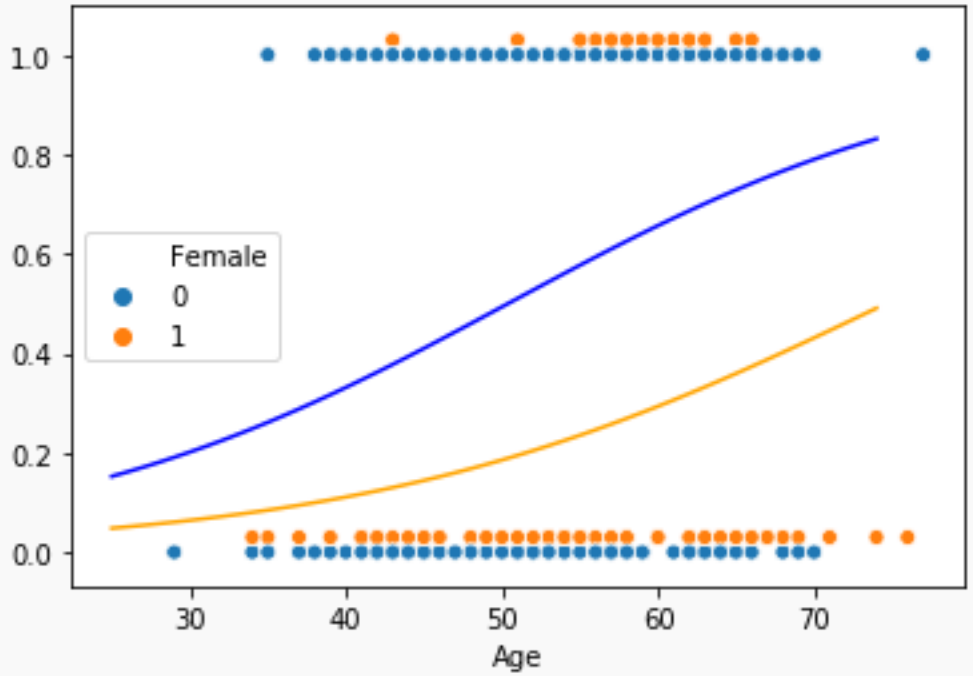
The results for the multiple logistic regression model are:

```
df_heart['Age_Female'] = df_heart['Age']*df_heart['Female']
X = df_heart[['Age','Female','Age_Female']]

logit = LogisticRegression(penalty='none', fit_intercept=True)
logit.fit(X, df_heart['AHD'])

print('Estimated betas (B0,B1,B2,B3):', logit.intercept_,logit.coef_)
```

```
Estimated betas (B0,B1,B2,B3): [-3.40463831] [[ 0.06754528 -1.08200061 -0.00742792]]
```



# Some questions

```
Estimated betas (B0,B1,B2,B3): [-3.40463831] [[ 0.06754528 -1.08200061 -0.00742792]]
```

1. Write down the complete model. Break this down into the model to predict log-odds of heart disease (HD) based on Age for males and the same model for females.

# Some questions

```
Estimated betas (B0,B1,B2,B3): [-3.40463831] [[ 0.06754528 -1.08200061 -0.00742792]]
```

2. Interpret the results of this model. What does the coefficient for the interaction term represent?

# Outline

- Interpreting interactions in logistic regression
- **Regularization in Logistic Regression**
- Multiclass Logistic Regression
  - Multinomial Logistic Regression
  - One-vs-Rest Logistic Regression
- Bayes Theorem and Misclassification Rates
- ROC Curves

# Review: Regularization in Linear Regression

What was the loss function in linear regression (not regularized)?

We saw in linear regression that maximizing the log-likelihood is equivalent to minimizing the sum of squares error:

$$\underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - (\beta_0 + \beta_1 x_{1i} + \cdots + \beta_p x_{pi})\right)^2$$

# Review: Regularization in Linear Regression

A regularization approach was to add a penalty to this loss.

For Ridge Regression this becomes:

$$\underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} \left(y_i - (\beta_0 + \beta_1 x_{1i} + \cdots + \beta_p x_{pi})\right)^2 + \lambda \sum_{j=1}^{J} \beta_j^2$$

This penalty *shrinks* the estimates towards zero.

Note: This it is analogue of using a Normal prior centered at zero in the Bayesian paradigm.

# Recall: 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 (**binary cross-entropy**):

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\operatorname{argmin}} \left[ -\sum_{i=1}^{n} \left( y_i \ln(p_i) + (1 - y_i) \ln(1 - p_i) \right) \right]$$

where $p_i = \frac{1}{1 - e^{-(\beta_0 + \beta_1 X_{1,i} + \cdots + \beta_p x_{p,i})}}$

Why is this a good loss function to minimize? Where does this come from?

The log-likelihood for independent $Y_i \sim \text{Bern}(p_i)$.

# 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:

$$\underset{\beta}{\operatorname{argmin}} \left[ -\sum y_i \log p_i + (1 - y_i) \log(1 - p_i) \quad + \lambda \sum_{j=1}^{J} \beta_j^2 \right]$$

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

**Note:** the `sklearn` package uses a different tuning parameter: instead of $\lambda$ they use a constant that is $C = \frac{1}{\lambda}$.

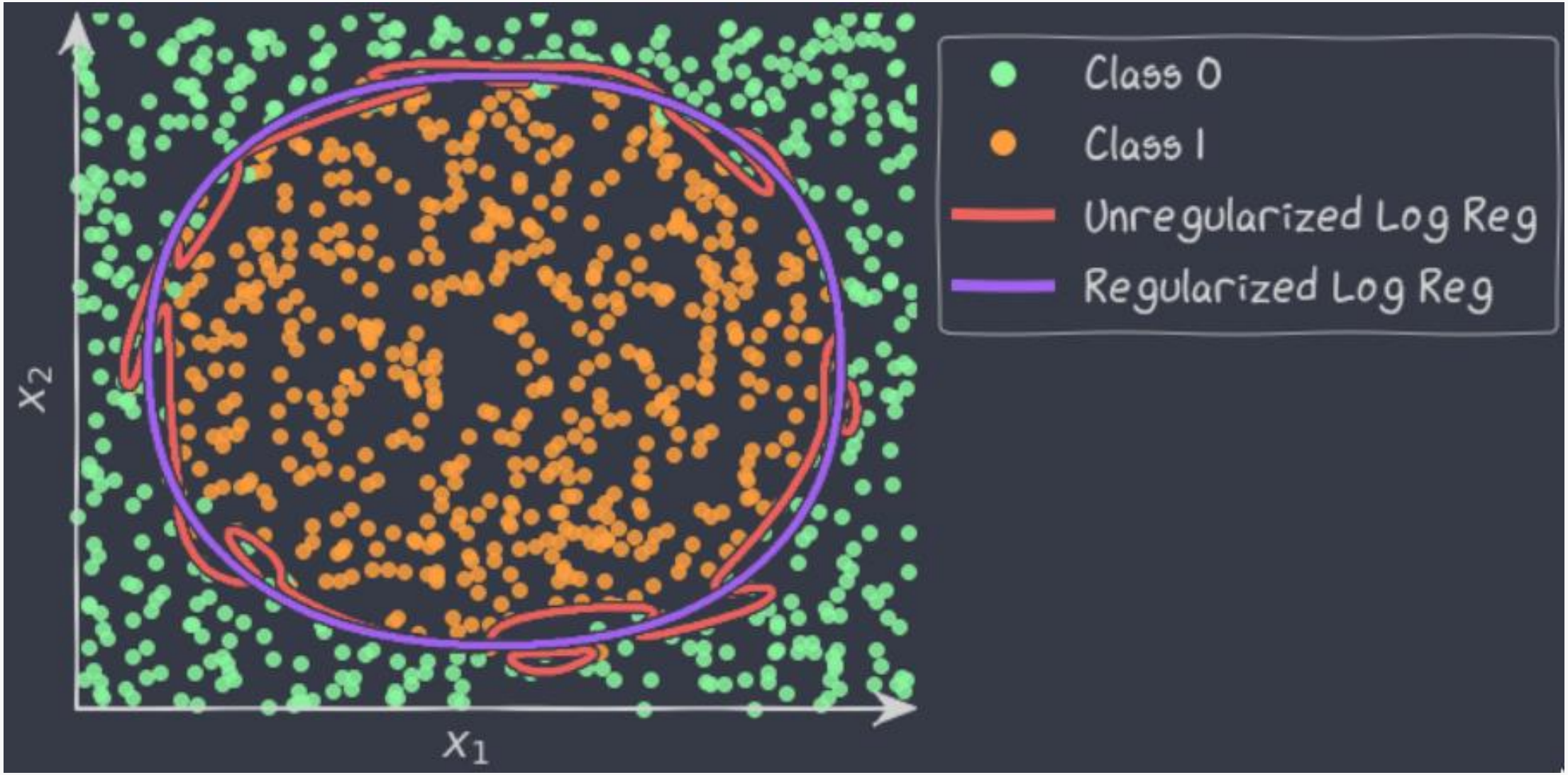
# Regularization in Logistic Regression: tuning lambda

Just like in linear regression, the regularization parameter must be chosen.

How should we go about doing this?

Cross Validation! Through building multiple training and validation set , we can select the best regularization parameter.

# Regularized Decision Boundaries



# Outline

- Interpreting interactions in logistic regression
- Regularization in Logistic Regression
- **Multiclass Logistic Regression**
  - **Multinomial Logistic Regression**
  - One-vs-Rest Logistic Regression
- Bayes Theorem and Misclassification Rates
- ROC Curves

# Logistic Regression for predicting 3+ classes

There are several extensions to standard logistic regression when the response variable $Y$ has more than 2 categories. The two most common are:

Nominal is used when the categories have no inherent order (like eye color: blue, green, brown, hazel, etc).

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

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

The first 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. For example, we could attempt to predict a student's concentration:

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

from predictors $X_1$ number of psets per week, $X_2$ how much time playing video games per week, etc.

# Multinomial Logistic Regression (cont.)

We could select the $y = 3$ case as the reference group (other concentration), and then fit two **separate models**:

1) A model to predict $y = 1$ (CS) from $y = 3$ (others)
2) A model to predict $y = 2$ (Stat) from $y = 3$ (others).

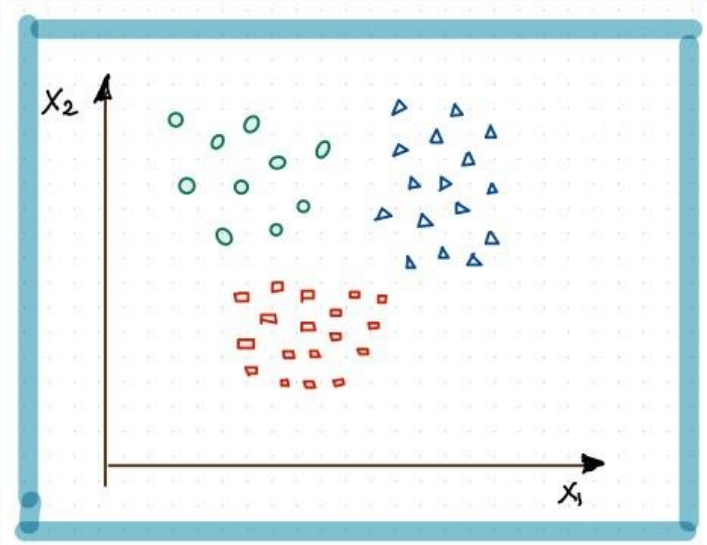
Ignoring interactions, how many parameters would need to be estimated?

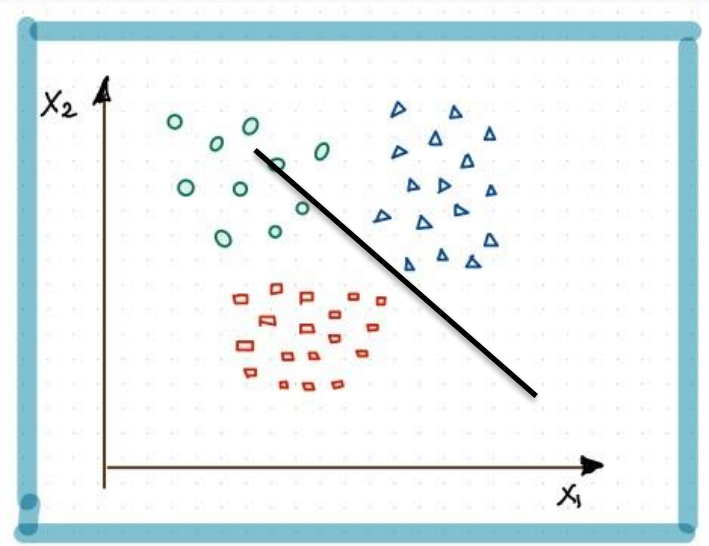
How could these models be used to estimate the probability of an individual falling in each concentration?

# Multinomial Logistic

Classifying three classes:

Red, Blue and Green can be turn into two binary Logistic Regressions



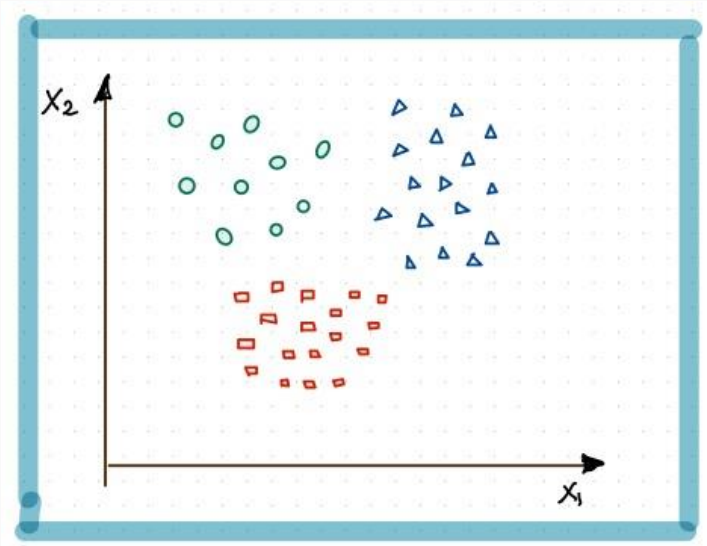



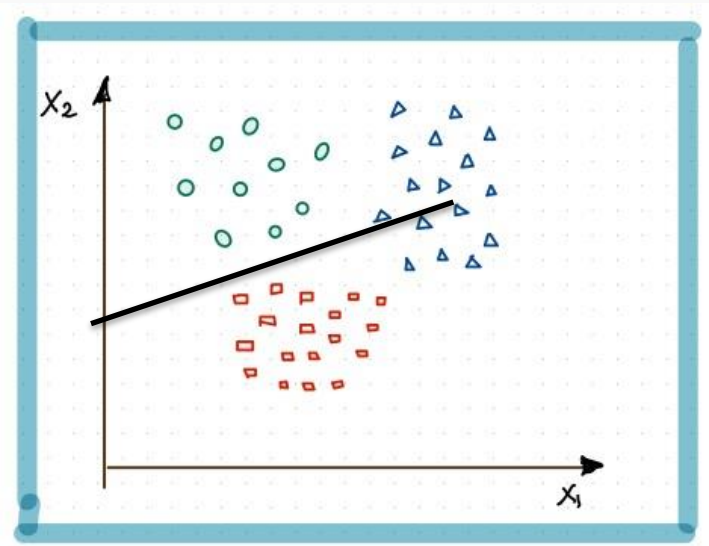

Blue vs Red

$$\ln\left(\frac{P(b)}{P(r)}\right) = \beta_b X$$

# Multinomial Logistic

Classifying three classes,
Red, Blue and Green can be turn into two binary Logistic Regressions




Green vs Red

$$\ln\left(\frac{P(g)}{P(r)}\right) = \beta_g X$$

# Multinomial Logistic Regression: the model

To predict $K$ classes ($K > 2$) from a set of predictors $X$, a multinomial logistic regression can be fit:

$$\ln\left(\frac{P(Y=1)}{P(Y=K)}\right) = \beta_{0,1} + \beta_{1,1}X_1 + \beta_{2,1}X_2 + \cdots + \beta_{p,1}X_p$$

$$\ln\left(\frac{P(Y=2)}{P(Y=K)}\right) = \beta_{0,2} + \beta_{1,2}X_1 + \beta_{2,2}X_2 + \cdots + \beta_{p,2}X_p$$

$$\vdots$$

$$\ln\left(\frac{P(Y=K-1)}{P(Y=K)}\right) = \beta_{0,K-1} + \beta_{1,K-1}X_1 + \beta_{2,K-1}X_2 + \cdots + \beta_{p,K-1}X_p$$

Each separate model can be fit as independent standard logistic regression models!

# Multinomial Logistic Regression in sklearn

```
mlogit = LogisticRegression(penalty="none", multi_class = 'multinomial')
mlogit.fit (nfl22[["Down","ToGo"]], nfl22["Play_Type"])


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

```
Estimated beta1:
 [[ 1.71107026  0.09577593]
 [-0.40924154  0.03968915]
 [-1.30182872 -0.13546508]]
Estimated beta0:
 [-6.29293303  1.88149967  4.41143335]
```

But wait, I thought you said we only fit $K-1$ logistic regression models!?!? Why are there $K$ intercepts and $K$ sets of coefficients????

```
pd.crosstab(nfl22["PlayType"],
            nfl22["Play_Type"])
```

| Play_Type | 0 | 1 | 2 |
|---|---|---|---|
| PlayType | | | |
| CLOCK STOP | 48 | 0 | 0 |
| FIELD GOAL | 854 | 0 | 0 |
| FUMBLES | 92 | 0 | 0 |
| PASS | 0 | 15397 | 0 |
| PUNT | 1874 | 0 | 0 |
| QB KNEEL | 353 | 0 | 0 |
| RUSH | 0 | 0 | 10794 |
| SACK | 0 | 1106 | 0 |
| SCRAMBLE | 0 | 784 | 0 |

# What is sklearn doing?

The $K-1$ models in multinomial regression lead to the following probability predictions:

$$\ln\left(\frac{P(Y=k)}{P(Y=K)}\right) = \beta_{0,k} + \beta_{1,k}X_1 + \beta_{2,k}X_k + \cdots + \beta_{p,k}X_p$$

$$\vdots$$

$$P(Y=k) = P(Y=K)e^{\beta_{0,k}+\beta_{1,k}X_1+\beta_{2,k}X_k+\cdots+\beta_{p,k}X_p}$$

Note:
the different denominators

This give us $K-1$ equations to estimate $K$ probabilities for everyone. But probabilities add up to 1 ☺, so we are all set.

sklearn then converts the above probabilities back into new betas (just like logistic regression, but the betas won't match):

$$\ln\left(\frac{P(Y=k)}{P(Y\neq k)}\right) = \beta'_{0,k} + \beta'_{1,k}X_1 + \beta'_{2,k}X_k + \cdots + \beta'_{p,k}X_p$$

# Outline

- Interpreting interactions in logistic regression
- Regularization in Logistic Regression
- Multiclass Logistic Regression
    - Multinomial Logistic Regression
    - **One-vs-Rest Logistic Regression**
- Bayes Theorem and Misclassification Rates
- ROC Curves

# One vs. Rest (OvR) Logistic Regression

An alternative multiclass logistic regression model in sklearn is called the 'One vs. Rest' (OvR) approach, which is our second method.

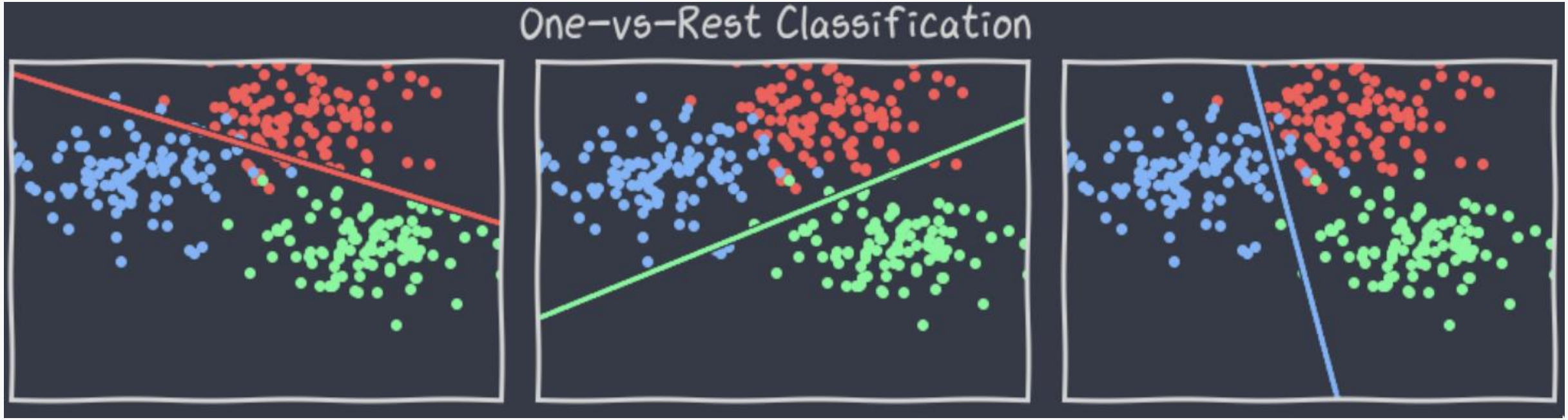
If there are 3 classes, then 3 separate logistic regressions are fit, where the probability of each category is predicted over the rest of the categories combined. So for the concentration example, 3 models would be fit:

- 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.

A picture is worth 1000 words.

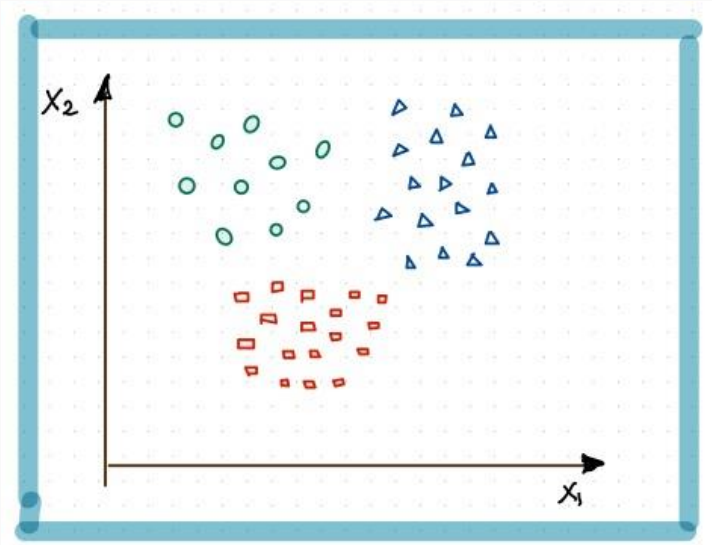
# One vs. Rest (ovr)

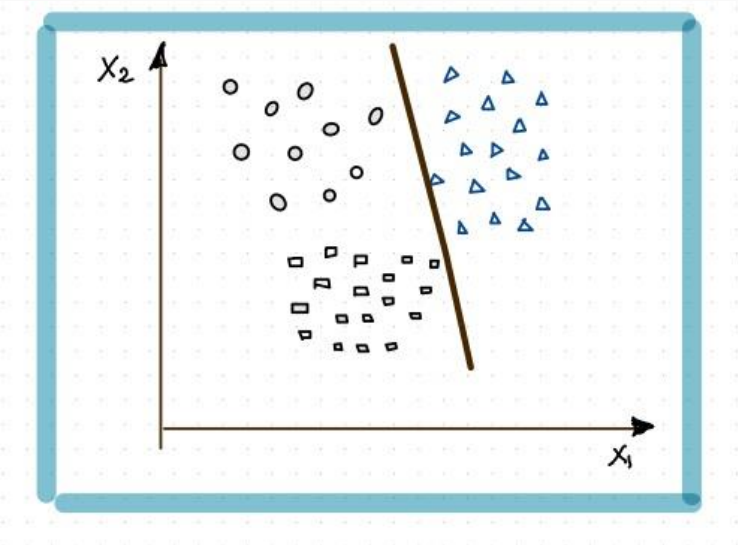
Classifying three classes,
Red, Blue and Green can be turn into three binary Logistic Regressions



# One vs. Rest (ovr)

Classifying three classes,
Red, Blue and Green can be turn into three binary Logistic Regressions



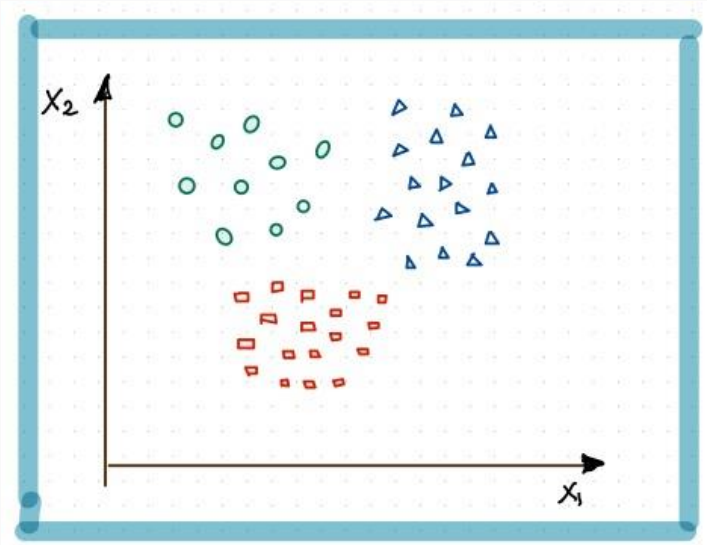

Blue vs others

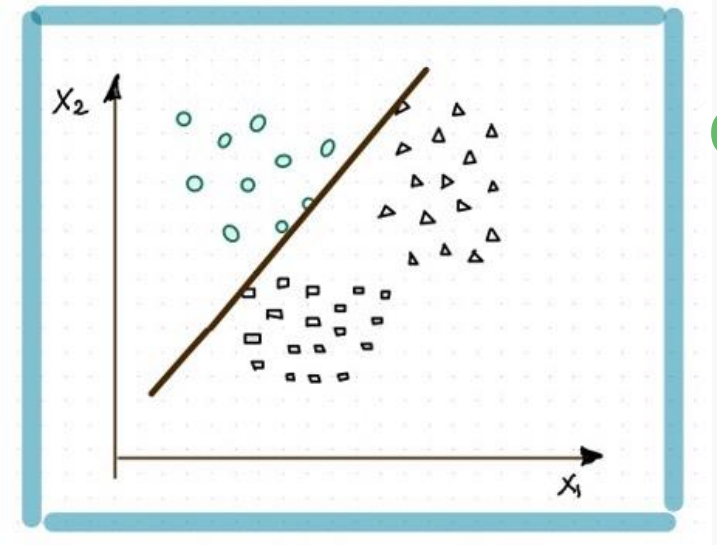
$$\ln\left(\frac{P(b)}{1-P(b)}\right) = \beta_b X$$

# One vs. Rest (ovr)

Classifying three classes,

Red, Blue and Green can be turn into three binary Logistic Regressions



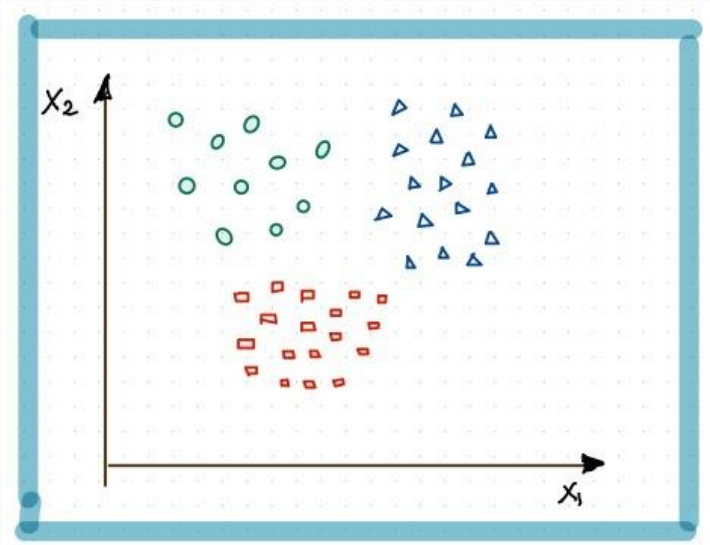

Green vs others

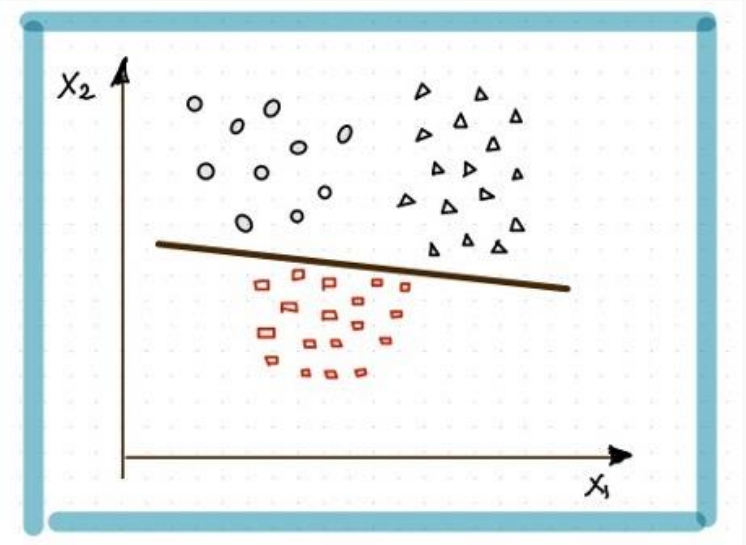
$$\ln\left(\frac{P(g)}{1-P(g)}\right) = \beta_g X$$

# One vs. Rest (ovr)

Classifying three classes,

Red, Blue and Green can be turn into three binary Logistic Regressions



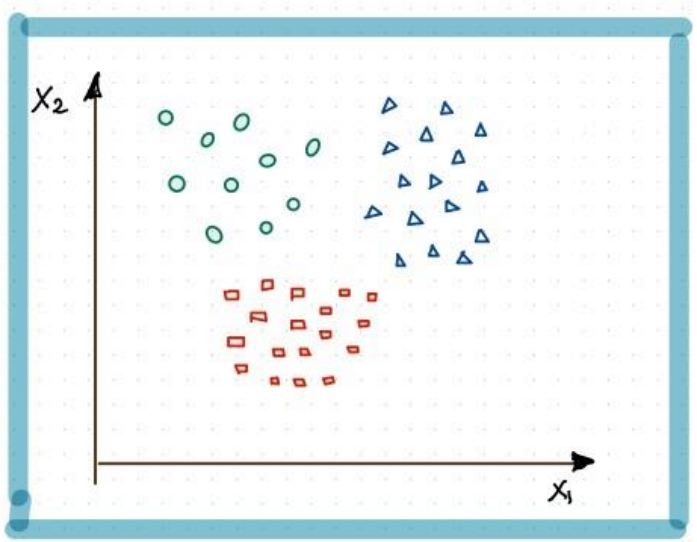

Red vs others

$$\ln\left(\frac{P(r)}{1-P(r)}\right) = \beta_r X$$

# One vs. Rest (ovr)

Classifying three classes,

Red, Blue and Green can be turn into three binary Logistic Regressions




Green vs others:

$$\ln\left(\frac{P(g)}{1-P(g)}\right) = \beta_g X$$

Blue vs others:

$$\ln\left(\frac{P(b)}{1-P(b)}\right) = \beta_b X$$

Red vs others:

$$\ln\left(\frac{P(r)}{1-P(r)}\right) = \beta_r X$$

`sklearn` normalizes the output of each of the three models when predicting probabilities:

$$\tilde{P}(b) = \frac{P(b)}{P(g)+P(b)+P(r)} \qquad \tilde{P}(g) = \frac{P(g)}{P(g)+P(b)+P(r)} \qquad \tilde{P}(r) = \frac{P(r)}{P(g)+P(b)+P(r)}$$

# Estimation and Regularization in multiclass settings

There is no difference in the approach to estimating the coefficients in the multiclass setting: we maximize the log-likelihood (or minimize negative log-likelihood).

This combined negative log-likelihood of all $K$ classes is sometimes called the **cross-entropy or multinomial logistic loss**:

$$\ell = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{1}(y_i = k)\ln(\hat{P}\,(y_i = k)) + \mathbb{1}(y_i \neq k)\ln(1 - \hat{P}\,(y_i = k))$$

And regularization can be done like always: add on a penalty term to this loss function based on L1 (sum of the absolute values) or L2 (sum of squares) norms.

# Outline

- Interpreting interactions in logistic regression
- Regularization in Logistic Regression
- Multiclass Logistic Regression
    - Multinomial Logistic Regression
    - One-vs-Rest Logistic Regression
- **Bayes Theorem and Misclassification Rates**
- ROC Curves

# Probability Review: Bayes' Theorem

What is conditional probability?

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

And using the fact that $P(A \; and \; B) = \; P(A|B)P(B)$ we get the simplest form of Bayes' Theorem:

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

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

$$P(B|A) = \frac{P(A|B)P(B)}{P(A|B)P(B) + P(A|B^C)P(B^C)}$$

# 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 predictive probability of the tests.

# Diagnostic Testing (cont.)

Bayes' theorem can be rewritten for diagnostic tests:

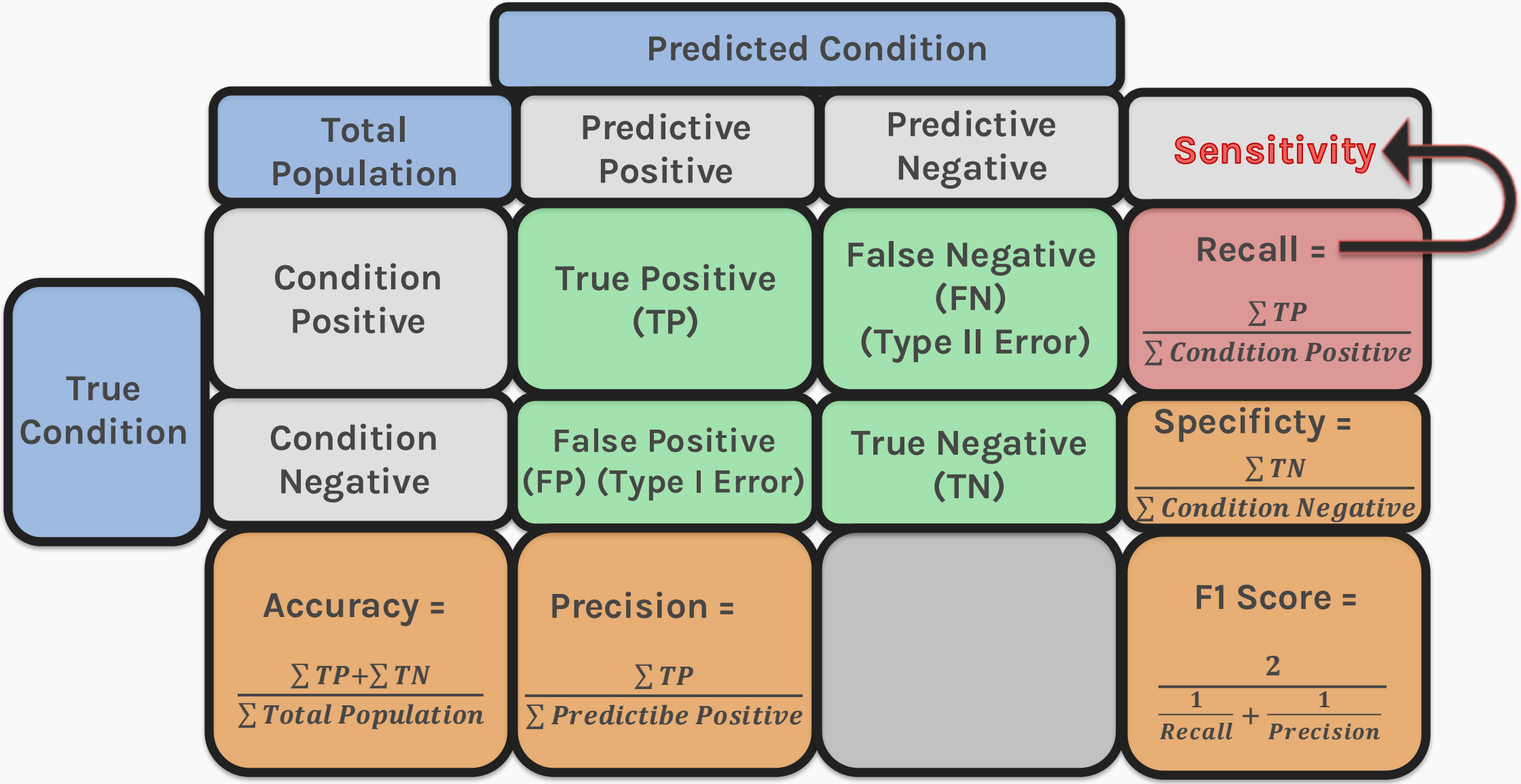
$$P(D+|T+) = \frac{P(T+|D+)P(D+)}{P(T+|D+)P(D+) + P(T+|D-)P(D-)}$$

These probability quantities can then be defined as:

- *Sensitivity*: $P(T+|D+)$
- *Specificity*: $P(T-|D-)$
- *Prevalence*: $P(D+)$
- *Positive Predictive Value*: $P(D+|T+)$
- *Negative Predictive Value*: $P(D-|T-)$

*1 – Specificity*

How do positive and negative predictive values relate? Be careful…


Predicted Condition
Total Population
Predictive Positive
Predictive Negative
Sensitivity
True Condition
Condition Positive
True Positive (TP)
False Negative (FN) (Type II Error)
Recall = $\frac{\sum TP}{\sum Condition\ Positive}$
Condition Negative
False Positive (FP) (Type I Error)
True Negative (TN)
Specificty = $\frac{\sum TN}{\sum Condition\ Negative}$
Accuracy = $\frac{\sum TP+\sum TN}{\sum Total\ Population}$
Precision = $\frac{\sum TP}{\sum Predictibe\ Positive}$
F1 Score = $\frac{2}{\frac{1}{Recall}+\frac{1}{Precision}}$

# 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?

# Error in Classification

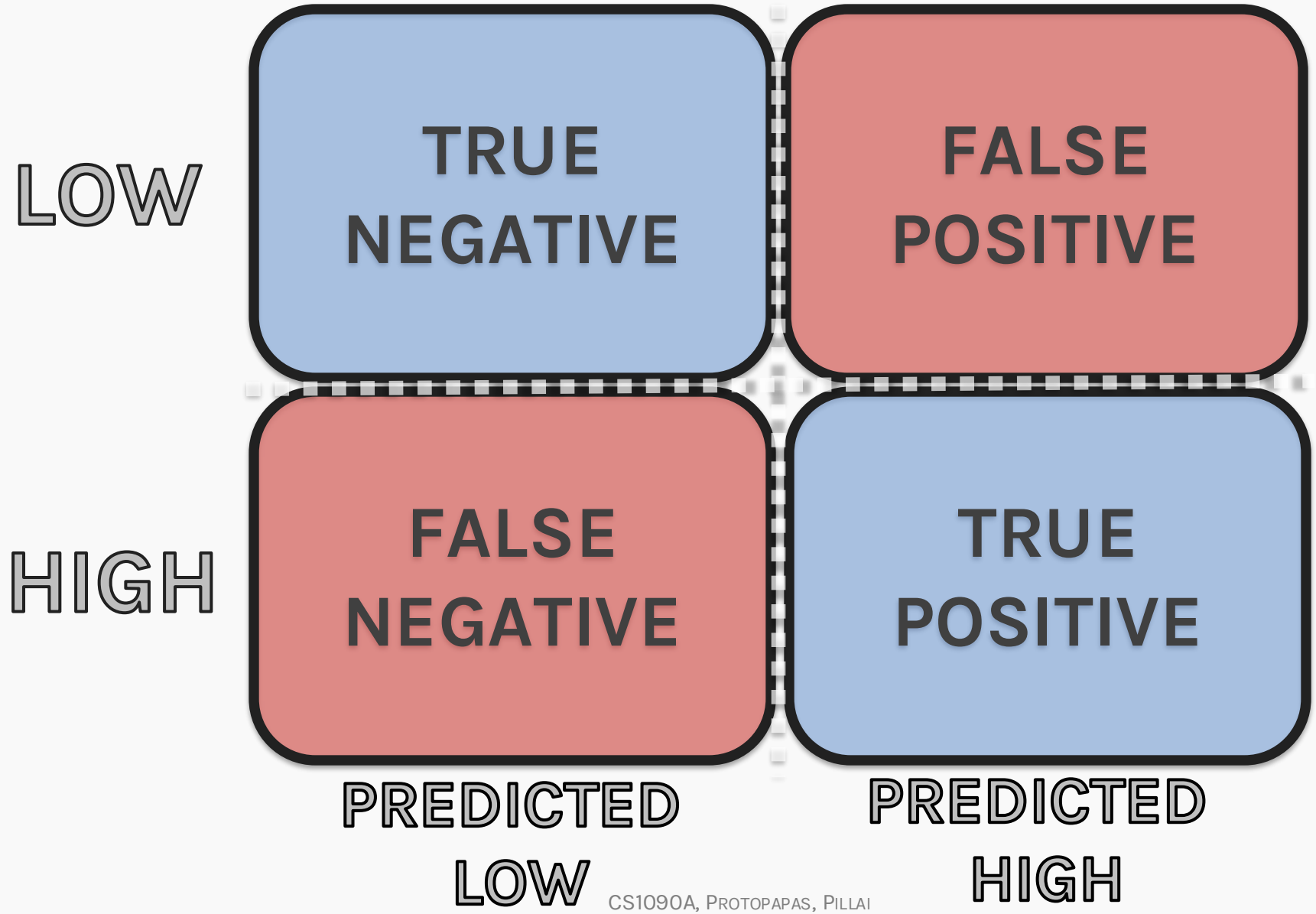
There are 2 major types of error in classification problems based on a binary outcome. They are:

False positives: incorrectly predicting $\hat{Y} = 1$ when it truly is in $Y = 0$.

False negatives: incorrectly predicting $\hat{Y} = 0$ when it truly is in $Y = 1$.

The results of a classification algorithm are often summarized in two ways: (1) a **confusion matrix**, sometimes called a **contingency table**, or a 2x2 table (more generally ($k$ x $k$) table) and (2) a receiver operating characteristics (ROC) curve.

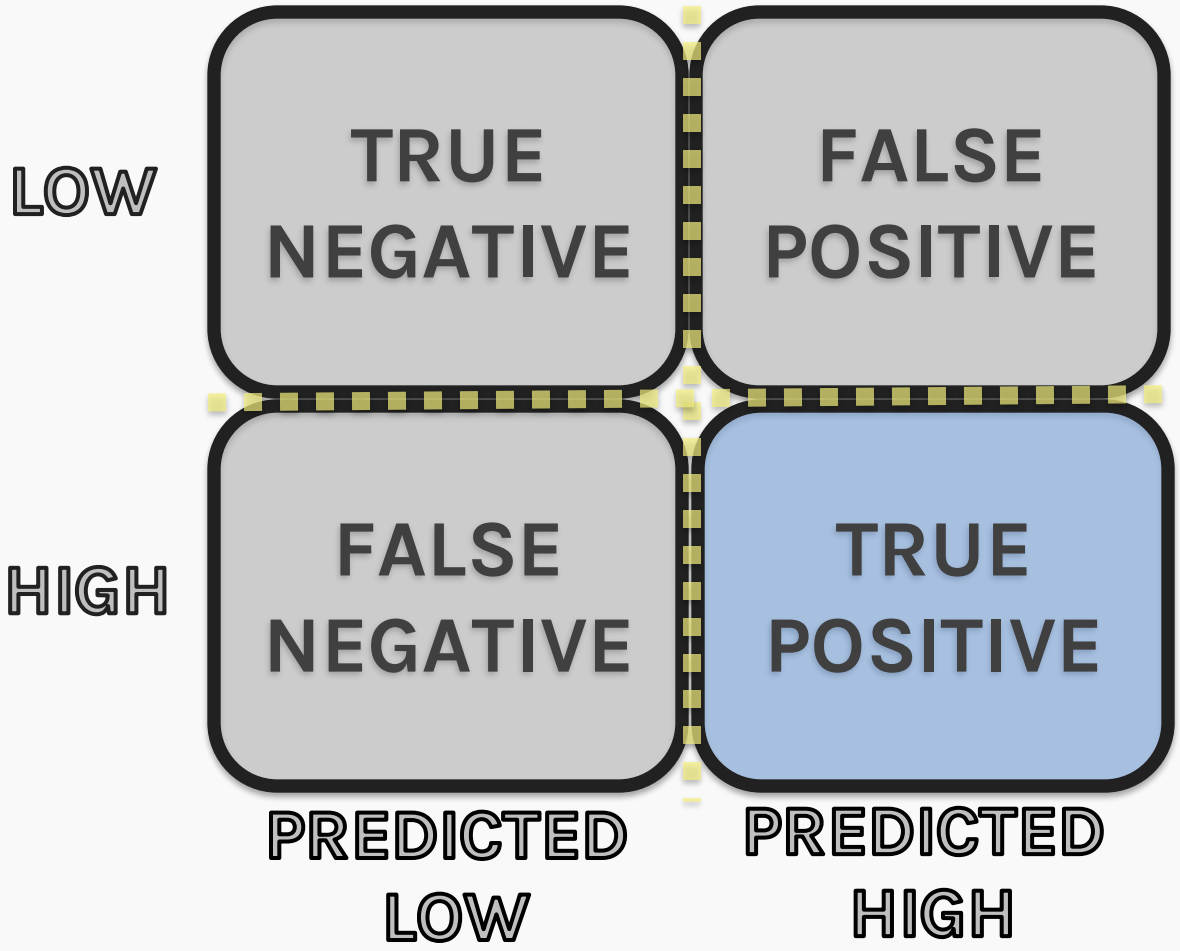
# The 'Confusion' Matrix



# The 'Confusion' Matrix

## TRUE POSITIVE (TP)

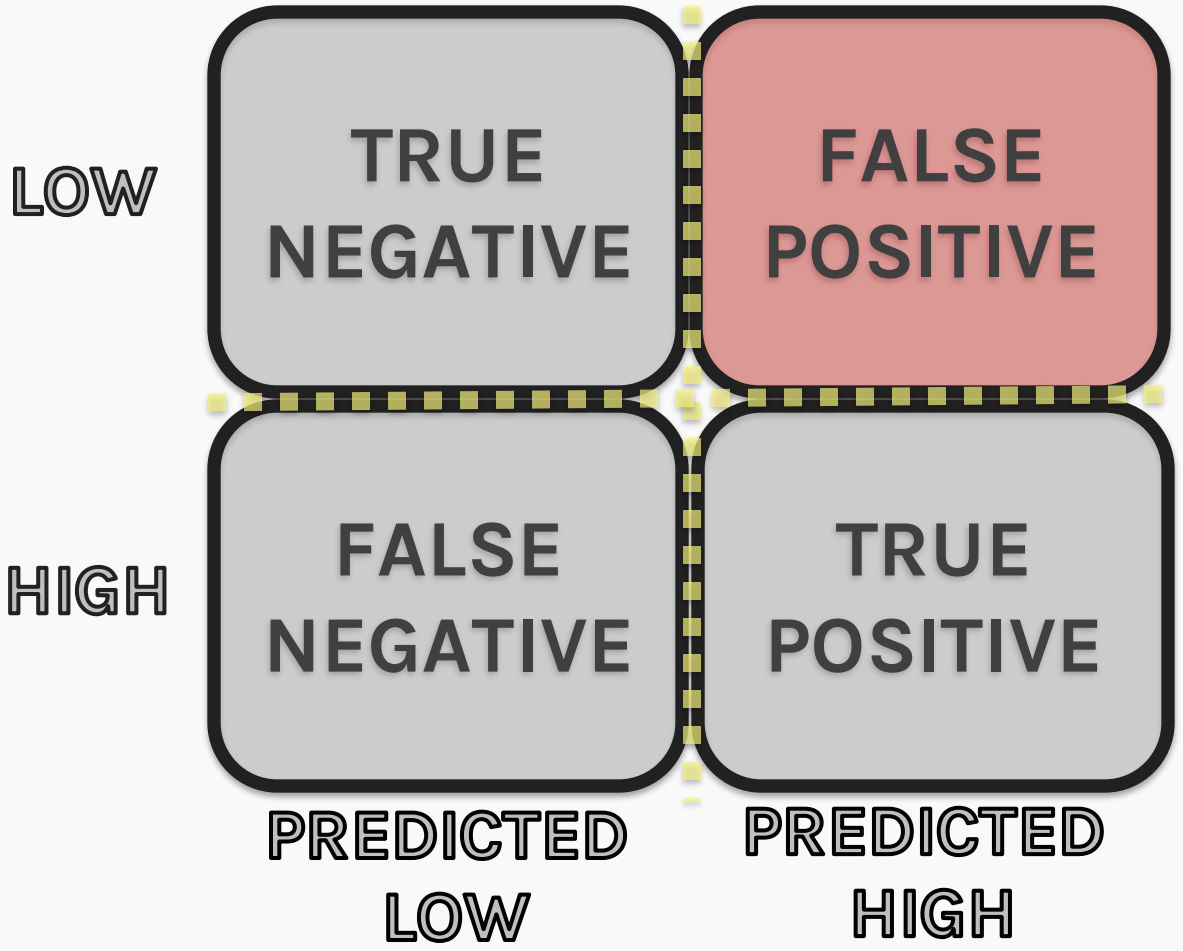
- Samples that are positive that the classifier predicts as positive are called True Positives.
- Example: a positive Covid test result would be a TRUE POSITIVE if you actually have Covid.



# The 'Confusion' Matrix

## FALSE POSITIVE (FP)

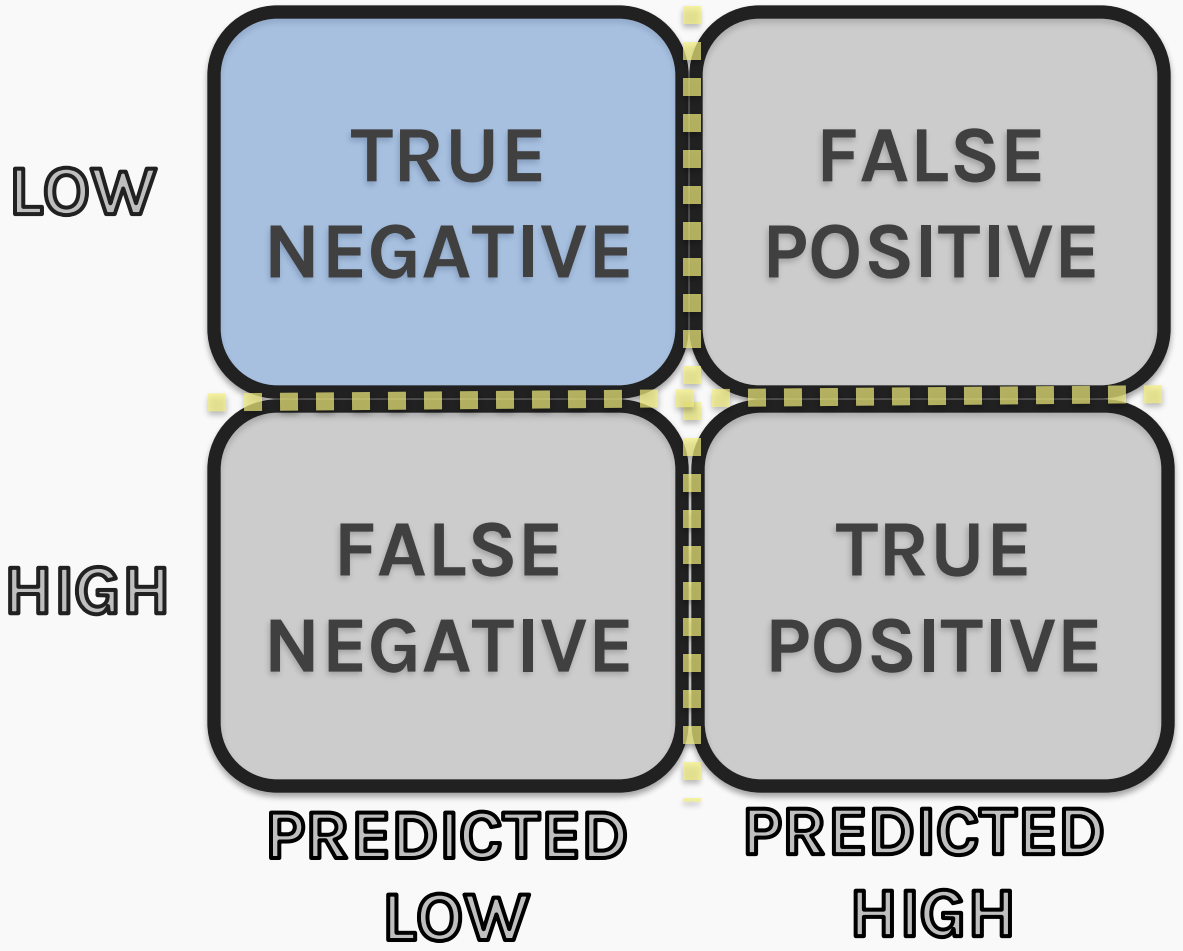
- Samples that are negative that the classifier predicts as positive are called False Positives.
- Example: a positive Covid test result would be a FALSE POSITIVE if you actually don't have Covid.



# The 'Confusion' Matrix

## TRUE NEGATIVE (TN)

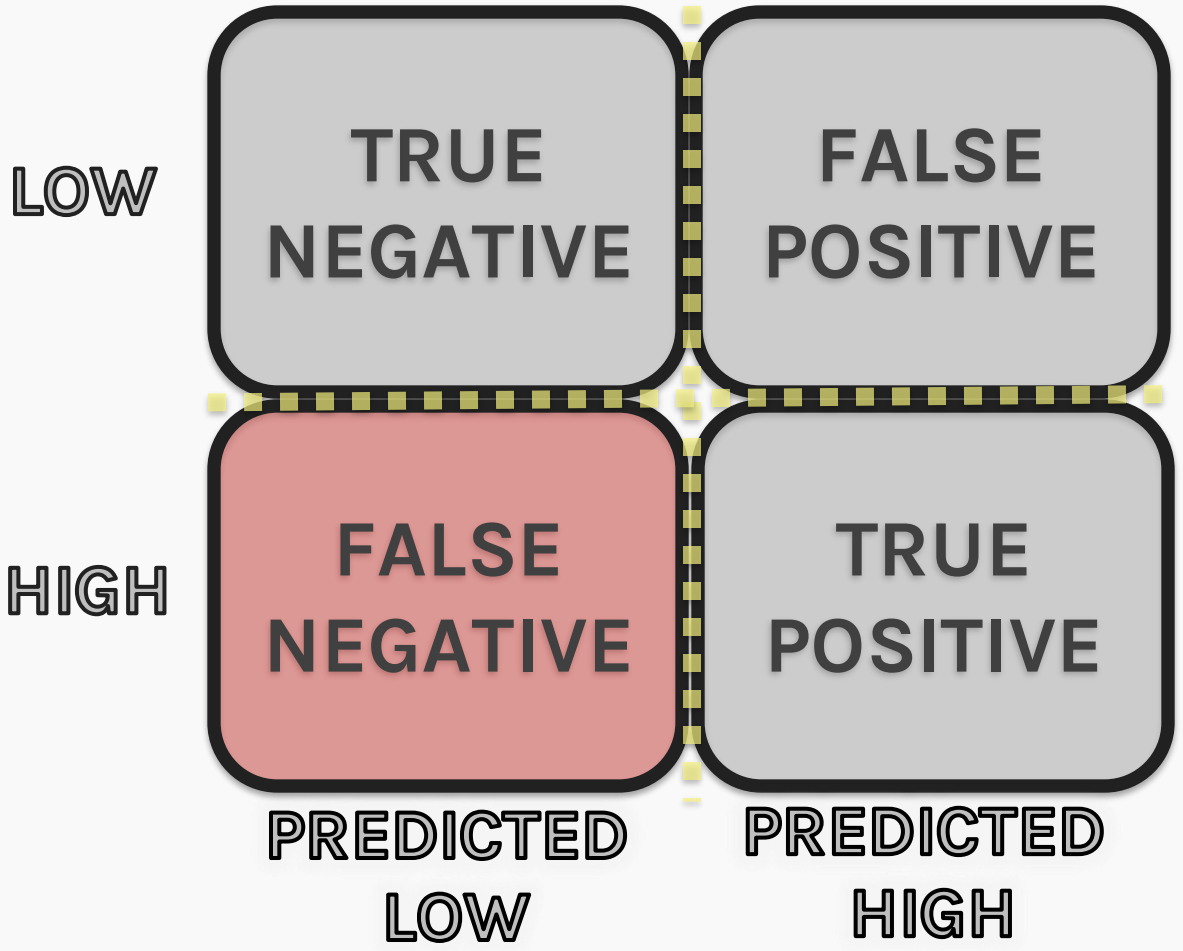
- Samples that are negative that the classifier predicts as negative are called True Negatives.
- Example: a negative Covid test result would be a TRUE NEGATIVE if you actually don't have Covid.



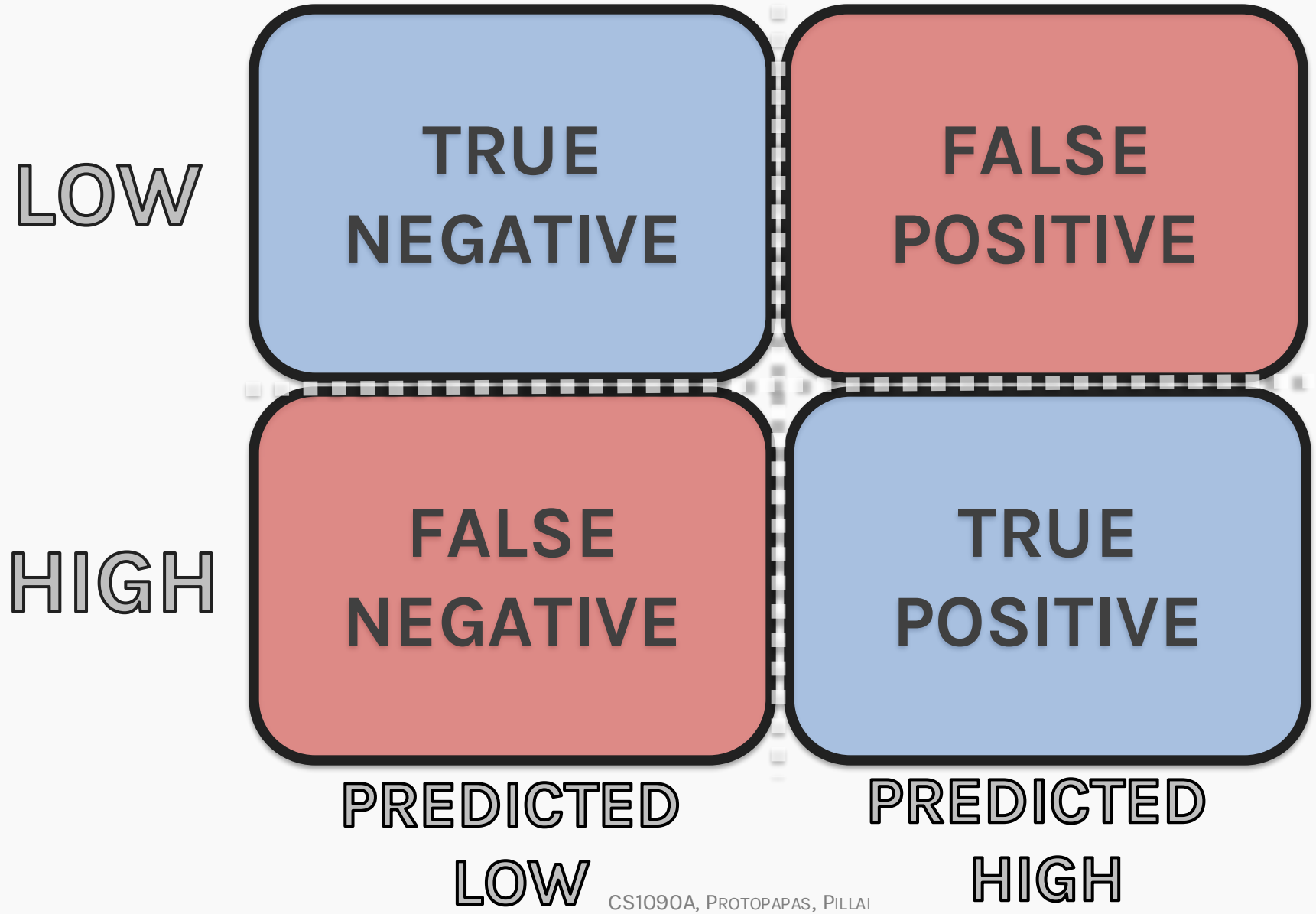
# The 'Confusion' Matrix

## FALSE NEGATIVE (FN)

- Samples that are negative that the classifier predicts as positive are called False Negatives.
- Example: a negative Covid test result would be a FALSE NEGATIVE if you actually have Covid.



# The 'Confusion' Matrix



# Confusion matrix

When a classification algorithm (like logistic regression) is used, the results can be summarize in a ($k$ x $k$) table as such:

| | Predicted no AHD ($\hat{Y} = 0$) | Predicted AHD ($\hat{Y} = 1$) |
|---|---|---|
| Truly no AHD ($Y = 0$) | 110 | 54 |
| Truly AHD ($Y = 1$) | 53 | 86 |

The table above was a classification based on a logistic regression model to predict AHD based on "3" predictors: $X_1$ = Age, $X_2$ = Sex, and $X_3$ = interaction between Age and Sex.

# 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 is was more important to predict AHD patients correctly (fewer false negatives), what could we do for our Bayes' classifier level?

We could classify instead based on:

$$\hat{P}(Y = 1) > \pi$$

and we could choose $\pi$ to be some level other than 0.50.

Let's see what the table looks like if $\pi$ were 0.40 or 0.60 instead. What should happen to the False Positive and False Negative frequencies?

# Other Confusion tables

Based on $\pi$ = 0.4:

| | Predicted no AHD ($\hat{Y} = 0$) | Predicted AHD ($\hat{Y} = 1$) |
|---|---|---|
| Truly no AHD ($Y = 0$) | 93 | 71 |
| Truly AHD ($Y = 1$) | 38 | 101 |

What has improved? What has worsened?

Based on $\pi$ = 0.6:

| | Predicted no AHD ($\hat{Y} = 0$) | Predicted AHD ($\hat{Y} = 1$) |
|---|---|---|
| Truly no AHD ($Y = 0$) | 138 | 26 |
| Truly AHD ($Y = 1$) | 74 | 65 |

Which should we choose? Why?

# Outline

- Interpreting interactions in logistic regression
- Regularization in Logistic Regression
- Multiclass Logistic Regression
  - Multinomial Logistic Regression
  - One-vs-Rest Logistic Regression
- Bayes Theorem and Misclassification Rates
- **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).

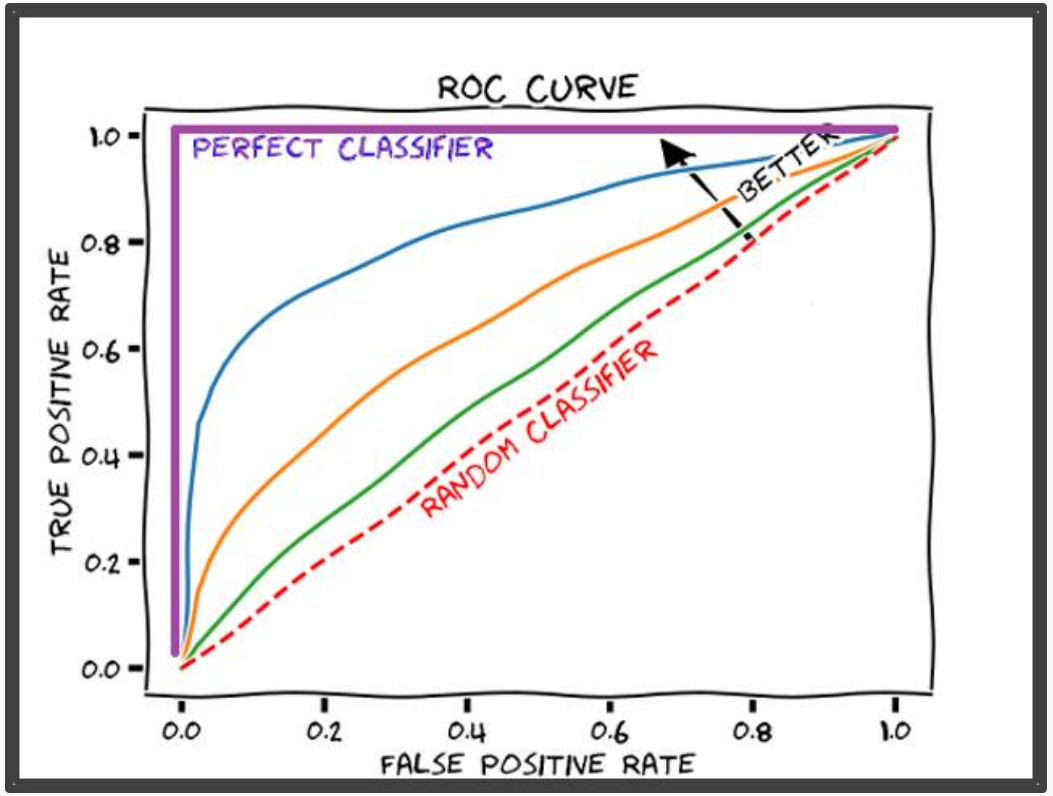
The vertical axis displays the true positive predictive value and the horizontal axis depicts the true negative predictive value.

What is the shape of an ideal ROC curve?

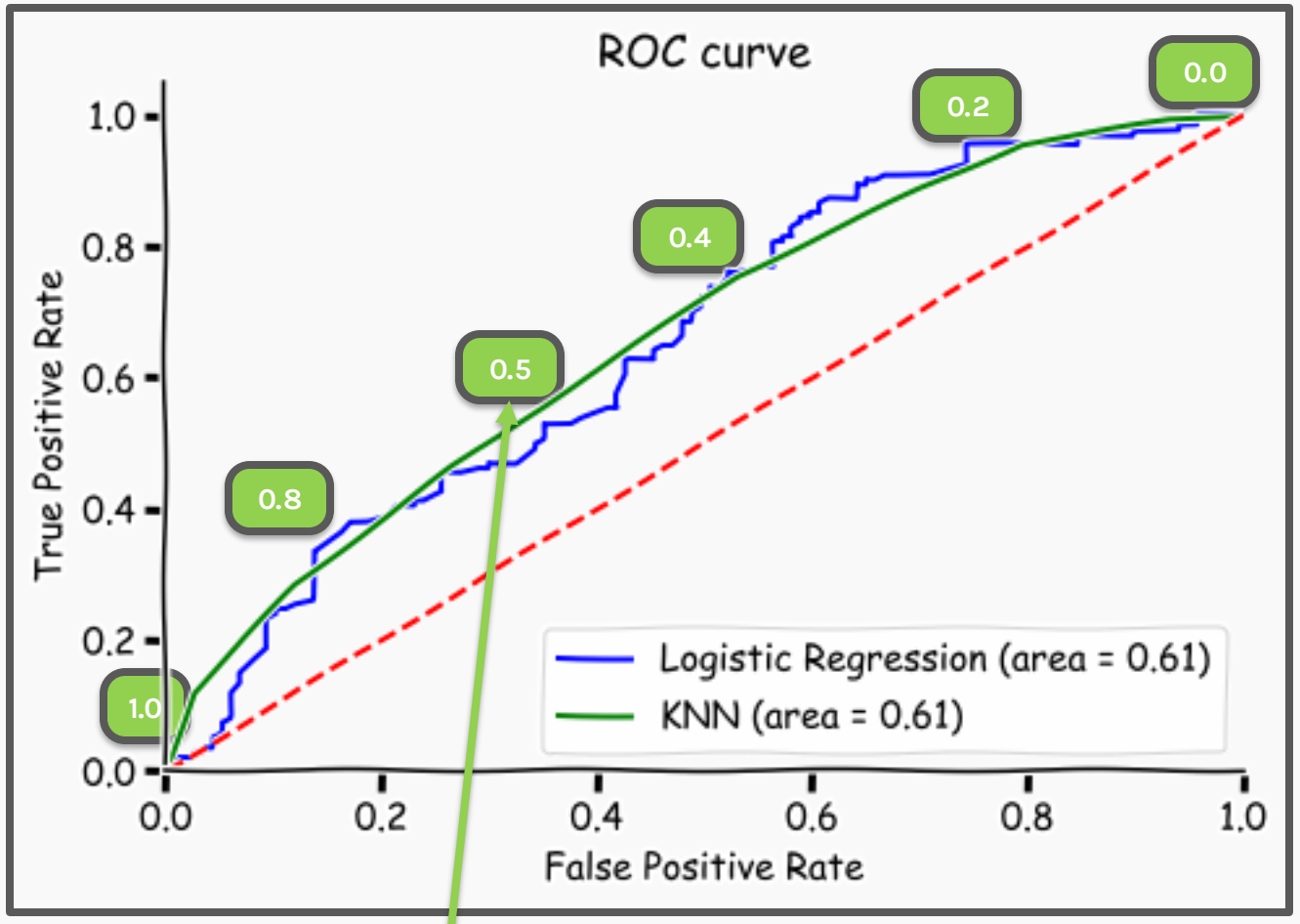
See next slide for an example.

# Receiver Operating Characteristic curve (ROC)

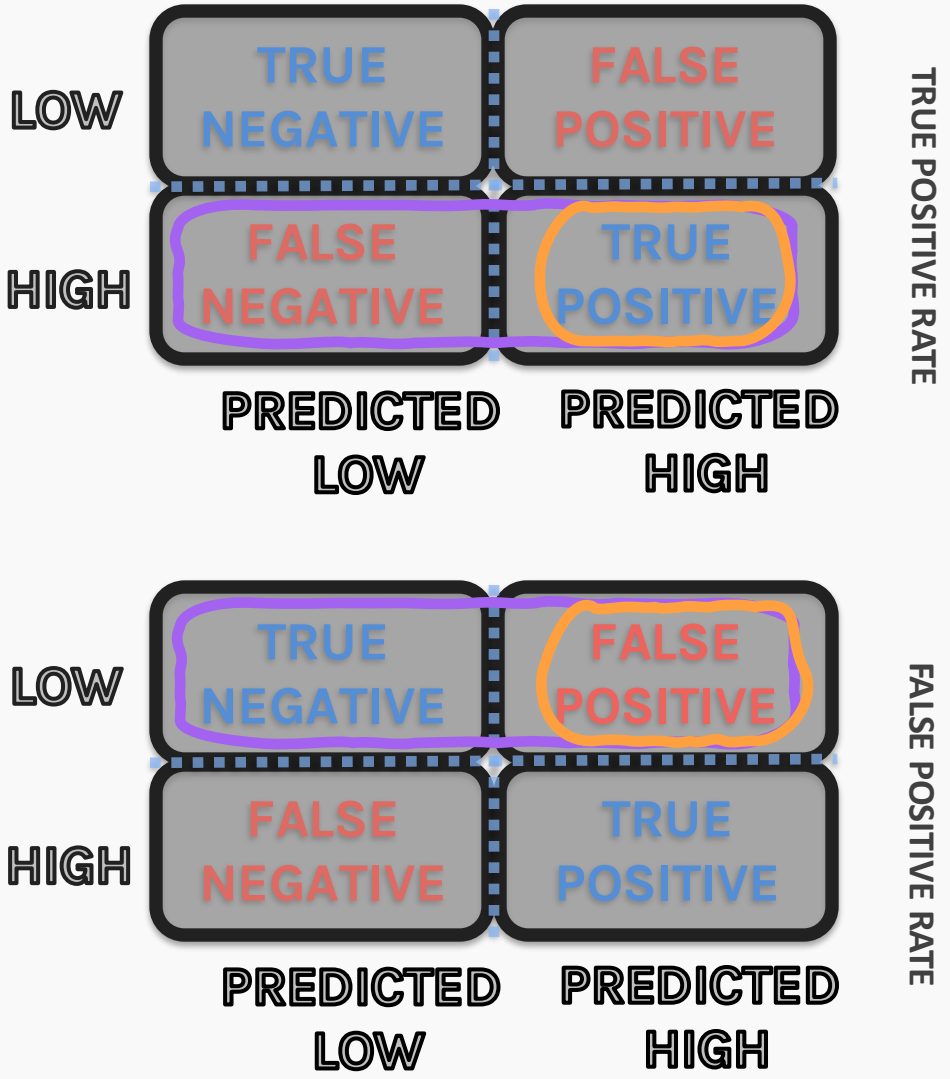
- The ROC curve was first developed by radar engineers during World War II for detecting enemy objects in battlefields.
- The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.



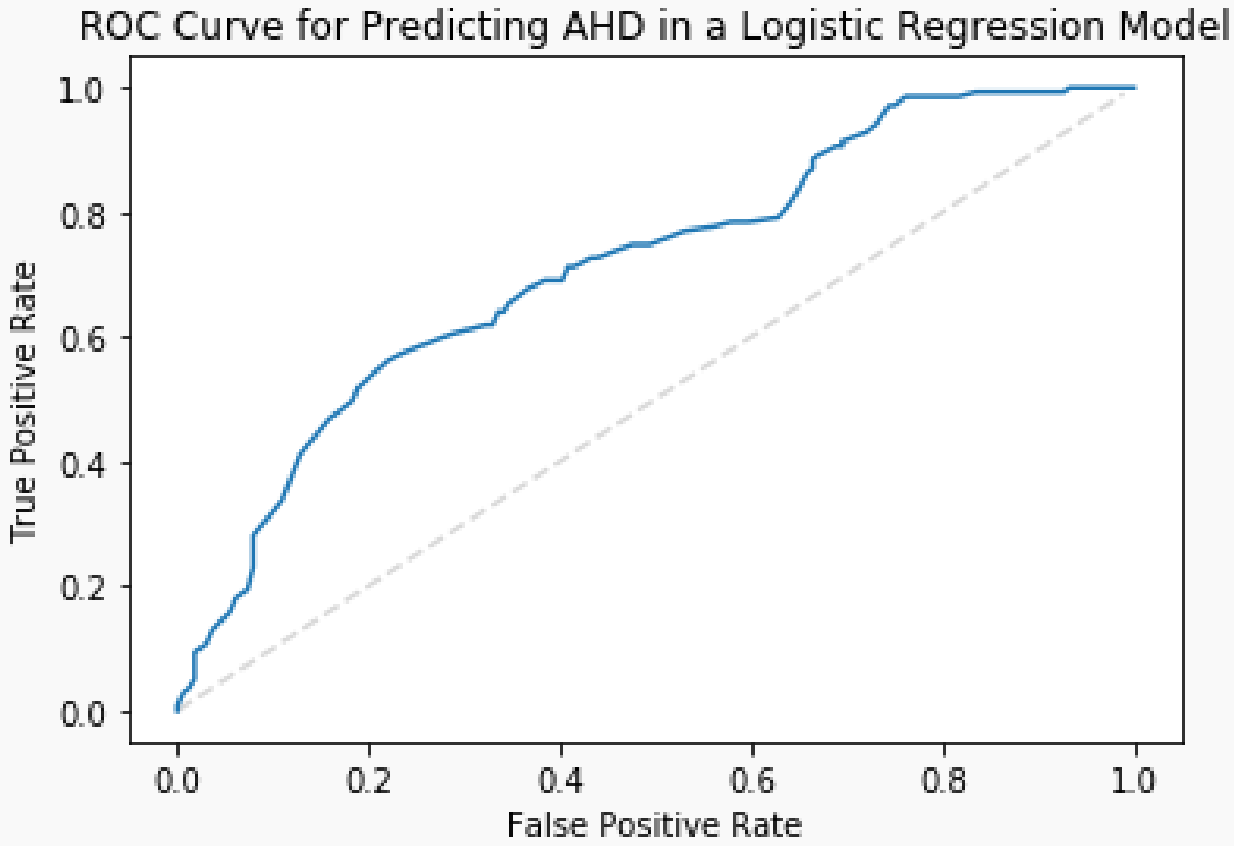
# ROC curve for various thresholds




THRESHOLD



# ROC Curve Example



# AUC 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 [biased] coin to perform classification?

AUC can be used to compare various approaches to classification: Logistic regression, *k*-NN, Decision Trees (to come), etc.