Lecture #14: Logistic Regression II

aka STAT109A, AC209A, CSCIE-109A

CS109A Introduction to Data Science

Pavlos Protopapas, Kevin Rader and Chris Gumb

Outline

- Inference in Logistic Regression
- Multiple Logistic Regression
- Classification Decision Boundaries
- Interpreting interactions in logistic regression
- Regularization in Logistic Regression
- Multiclass Logistic Regression
- Bayes Theorem and Misclassification Rates
- ROC Curves

Estimating the Simple Logistic Model

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

We usually do not like to deal with products, so we can show that finding the $argmax\ L(p|Y)$is INDENTICAL to finding the $\operatorname{argmin}(-\log(L(p|Y)))$:

First, we note that:

$$\operatorname*{argmax}_\beta(L(p|y)) = \operatorname*{argmax}_\beta(\log L(p|y))$$

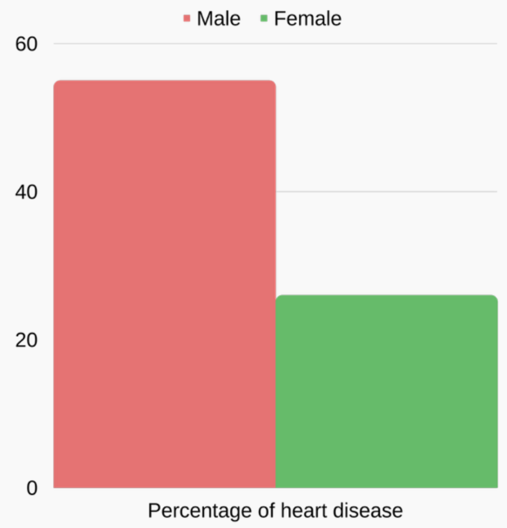
Outline

Statistical Inference in Logistic Regression

Just like in linear regression, when the predictor, X, is binary, the interpretation of the model simplifies.

In this case, what are the interpretations of $\hat{\beta}_0$ and $\hat{\beta}_1$?

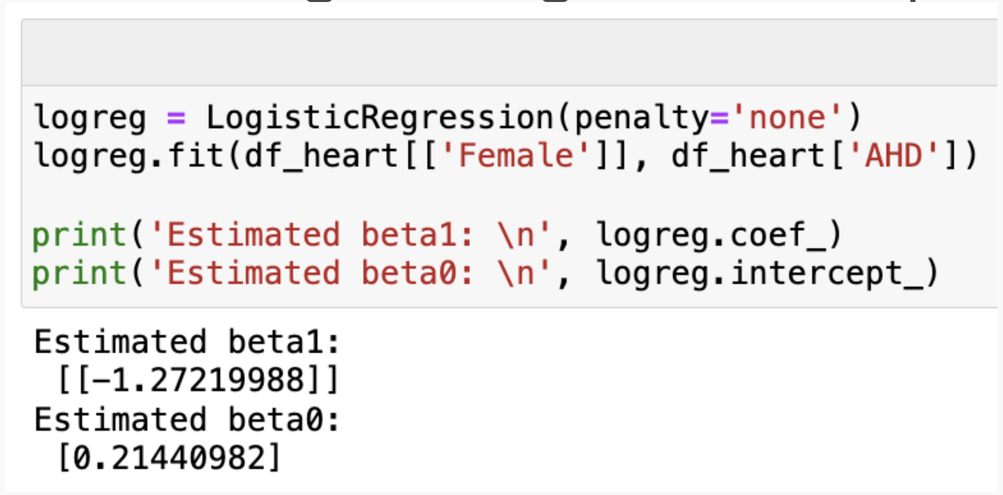
Consider the heart disease dataset represented here:



If we predict heart disease based on biological sex, how would you calculate and interpret the coefficient estimates $\hat{\beta}_0$ and $\hat{\beta}_1$?

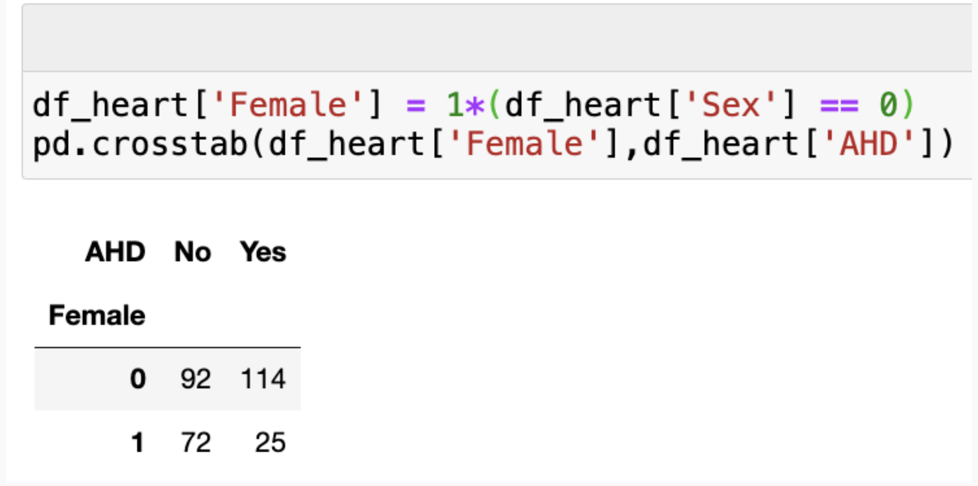
A Second Logistic Regression Model in sklearn

Here is a logistic regression output to predict Y = AHD from X = MaxHR:



```
logreg = LogisticRegression(penalty='none')
logreg.fit(df_heart[['Female']], df_heart['AHD'])

print('Estimated beta1: \n', logreg.coef_)
print('Estimated beta0: \n', logreg.intercept_)
```

```
Estimated beta1: 
 [[-1.27219988]]
Estimated beta0: 
 [0.21440982]
```



```
df_heart['Female'] = 1*(df_heart['Sex'] == 0)
pd.crosstab(df_heart['Female'],df_heart['AHD'])
```

AHD	No	Yes
Female		
0	92	114
1	72	25

What is the estimated model? What are the interpretations of the $\hat{\beta}$s?

$$\ln\left(\frac{\hat{P}(Y=1)}{1-\hat{P}(Y=1)}\right) = 0.2144 - 1.272(Female)$$

What is the estimated log-odd of AHD for Females and for Males? What about estimated probabilities? How does this agree with the table?

Statistical Inference in Logistic Regression

The **uncertainty of the estimates** $\hat{\beta}_0$ and $\hat{\beta}_1$ can be quantified and used to calculate both confidence intervals and hypothesis tests.

Of course, you can use **bootstrapping** (CI) and **permutation testing** (hypothesis testing) to perform these inferences.

Note:

The estimate for the standard errors of these estimates without bootstrap, is based on a quantity called Fisher's Information (beyond the scope of this class), which is related to the curvature of the log-likelihood function (the second derivative). Why does this make sense geometrically?

Due to the nature of the underlying Bernoulli distribution, if you estimate the underlying proportion p_i, you get the variance for free! Because of this, the inferences will be based on the normal approximation (and not t-distribution based).

Outline

- Inference in Logistic Regression
- **Multiple Logistic Regression**
- Interpreting interactions in logistic regression
- Classification Decision Boundaries
- Regularization in Logistic Regression
- Multiclass Logistic Regression
- Bayes Theorem and Misclassification Rates
- ROC Curves

Multiple Logistic Regression

It is simple to illustrate examples in logistic regression when there is just one predictors variable.

But the approach ‘easily’ generalizes to the situation where there are multiple predictors.

A lot of the same details as linear regression apply to logistic regression. Interactions can be considered, multicollinearity is a concern and so is overfitting.

So how do we correct for such problems?

Regularization and checking though train and cross-validation!

We will get into the details of this, along with other extensions of logistic regression, in the next lecture.

Multiple Logistic Regression

Earlier we saw the general form of *simple* logistic regression, meaning when there is just one predictor used in the model:

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

Multiple logistic regression is a generalization to multiple predictors. More specifically we can define a multiple logistic regression model to predict $P(Y=1)$ as such:

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

Fitting Multiple Logistic Regression

The estimation procedure is identical to that as before for simple logistic regression:

- A likelihood approach is taken, and the negative log-likelihood is minimized across all parameters $\beta_0, \beta_1, \ldots, \beta_p$ using an iterative method like Gradient Descent.

The actual fitting of a Multiple Logistic Regression is easy using software (of course there's a python package for that) as the iterative minimization of the -ve log-likelihood has already been hard coded.

In the `sklearn.linear_model` package, you just have to create your multidimensional design matrix X to be used as predictors in the `LogisticRegression` function.

Interpretation of Multiple Logistic Regression

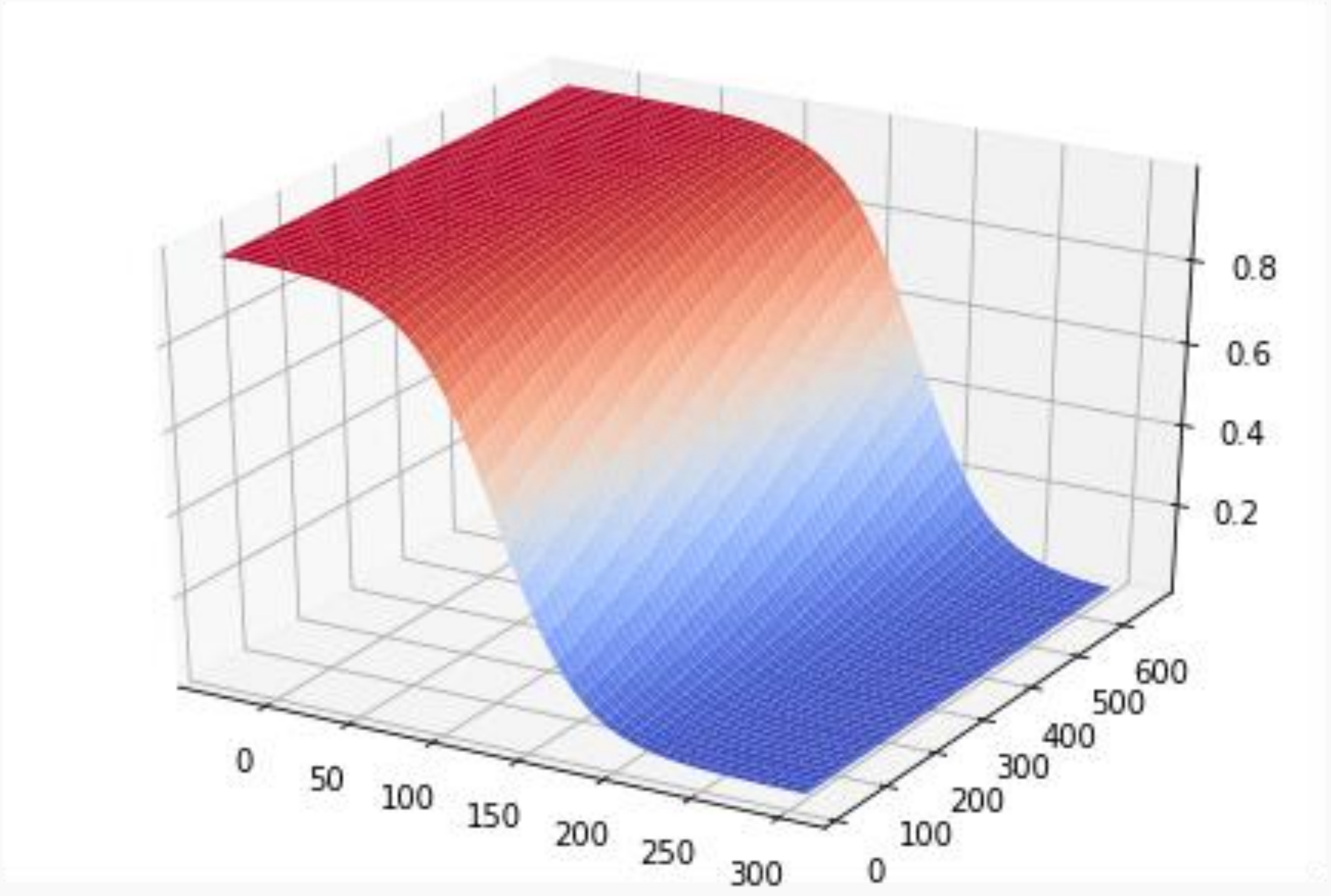
Interpreting the coefficients in a multiple logistic regression is similar to that of linear regression.

Key: since there are other predictors in the model, the coefficient $e^{\widehat{\beta}_j}$ is the multiplicative change in odds between the j^{th} predictor and the response. But do we have to say “Controlling for the other predictors in the model”?

We are trying to attribute the partial *effects* of each model controlling for the others (aka, controlling for possible *confounders*).

Multicollinearity plays a role just like in linear regression.

Visualizing Multiple Logistic Regression



Outline

- Inference in Logistic Regression
- Multiple Logistic Regression
- Classification Decision Boundaries
- Interpreting interactions in logistic regression
- Regularization in Logistic Regression
- Multiclass 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.

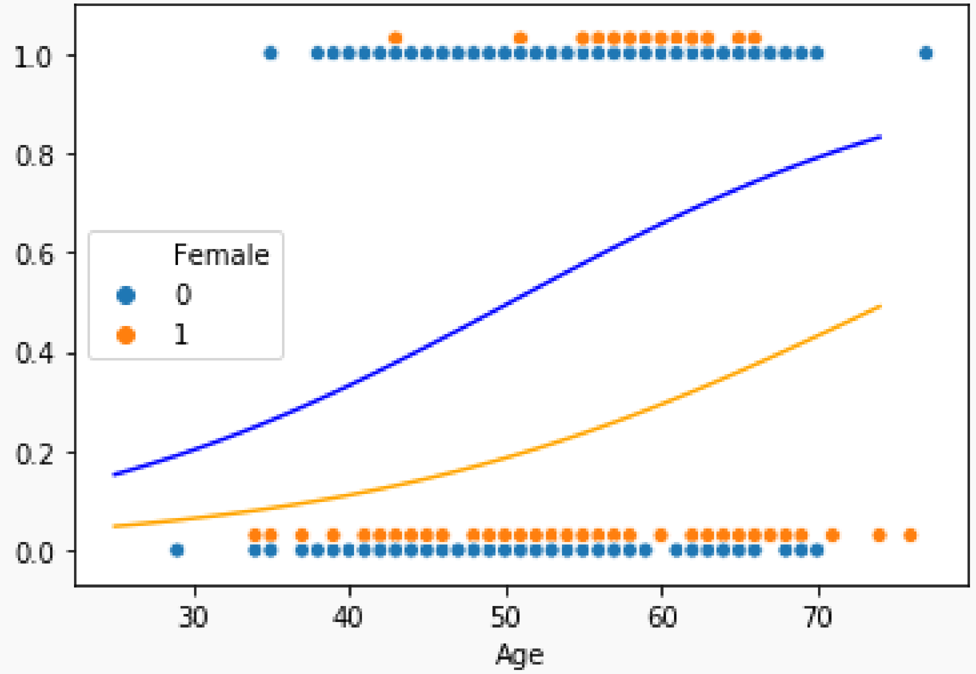
Here, we are predicting AHD from Age, Female, and the interaction between the two.

Interpreting Multiple Logistic Regression: an Example

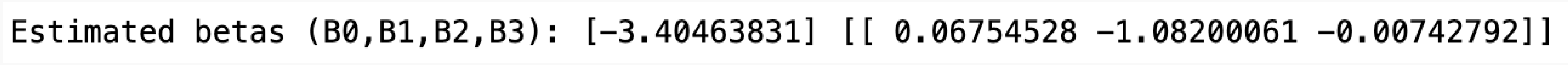
The results for the multiple logistic regression model are:

```
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]]
```



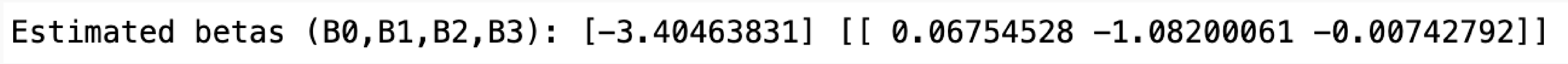
Some questions



```
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?

3. Estimate the odds ratio of AHD comparing men to women using this model [trick question].

Outline

- Inference in Logistic Regression
- Multiple Logistic Regression
- Classification Decision Boundaries
- Interpreting interactions in logistic regression
- Regularization in Logistic Regression
- Multiclass Logistic Regression
- Bayes Theorem and Misclassification Rates
- ROC Curves

Using Logistic Regression for Classification

How can we use logistic regression to perform classification?

That is, how can we predict when $Y = 1$ vs. when $Y = 0$?

We can classify all observations for which:

- Classify all observations with $\hat{P}(Y = 1) \geq 0.5$ to be in the group associated with $Y = 1$.

- Classify all observations with $\hat{P}(Y = 1) < 0.5$ to be in the group associated with $Y = 0$.

How would this extend if Y has 3+ classes?

Decision Boundaries for Classification

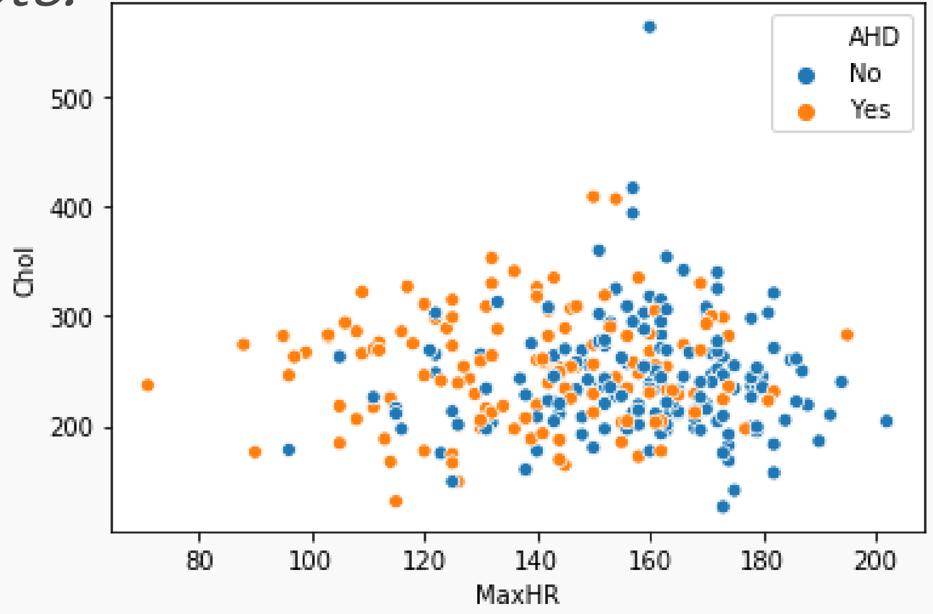
Recall that we could attempt to purely classify each observation based on whether the estimated $P(Y = 1)$ from the model is at least 0.5:

$$\hat{P}(Y = 1) \geq 0.5$$

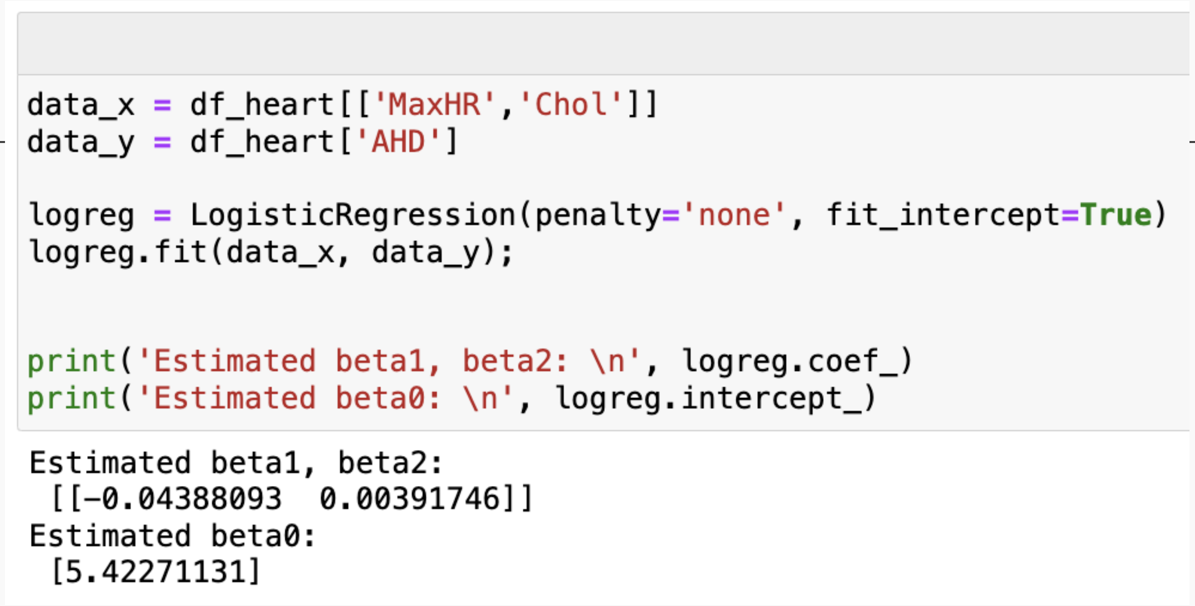
This results in a **decision boundary**: a surface (line, curve, etc. in 2D) that separates the predicted classes into *sets*.

Here's a 2-D plot from our Heart Data Set:

How do you expect logistic regression to draw the decision boundary?



Decision Boundaries Example



```
data_x = df_heart[['MaxHR','Chol']]
data_y = df_heart['AHD']

logreg = LogisticRegression(penalty='none', fit_intercept=True)
logreg.fit(data_x, data_y);


print('Estimated beta1, beta2: \n', logreg.coef_)
print('Estimated beta0: \n', logreg.intercept_)
```

```
Estimated beta1, beta2:
 [[-0.04388093  0.00391746]]
Estimated beta0:
 [5.42271131]
```

Here is the output from a logistic regression model with 2 predictors:

What is the estimated model?

$$\ln\left(\frac{\hat{P}(Y=1)}{1-\hat{P}(Y=1)}\right) = 5.423 - 0.0439(MaxHR) + 0.0039(Chol)$$

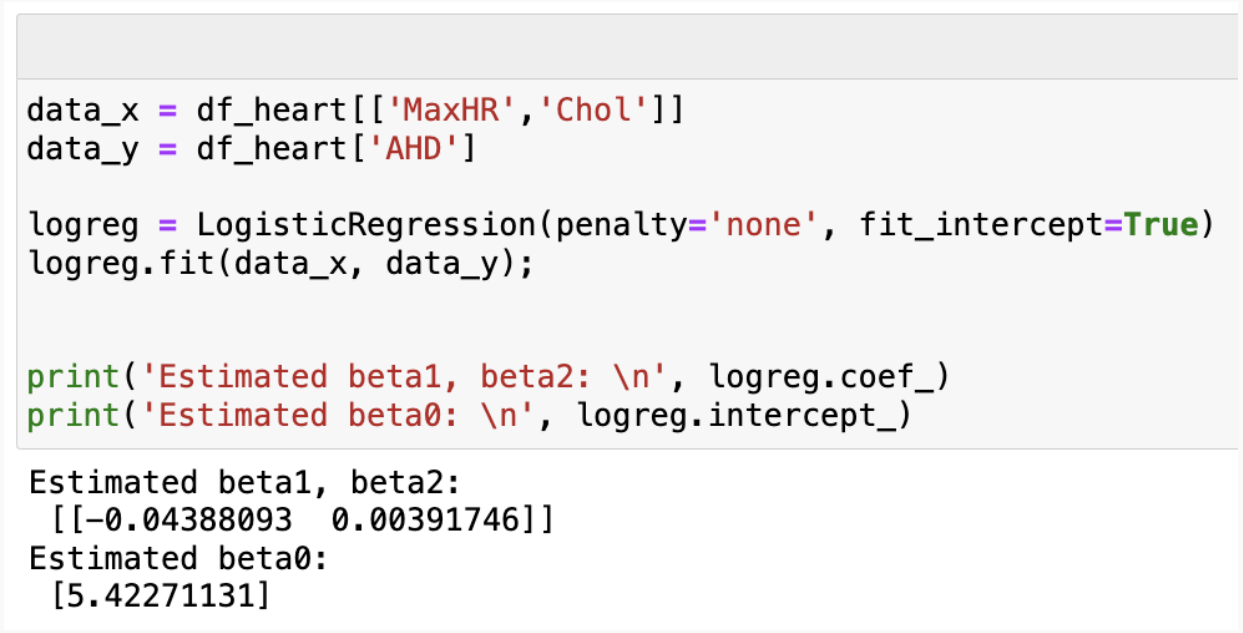
What are the interpretations of the $\hat{\beta}$s?

What will the decision boundary look like? Key: if $\hat{P}(Y=1) = 0.5$, then what are the estimated odds? What are the estimated log-odds?

In logistic regression, the decision boundary is defined when $X\beta = 0$.

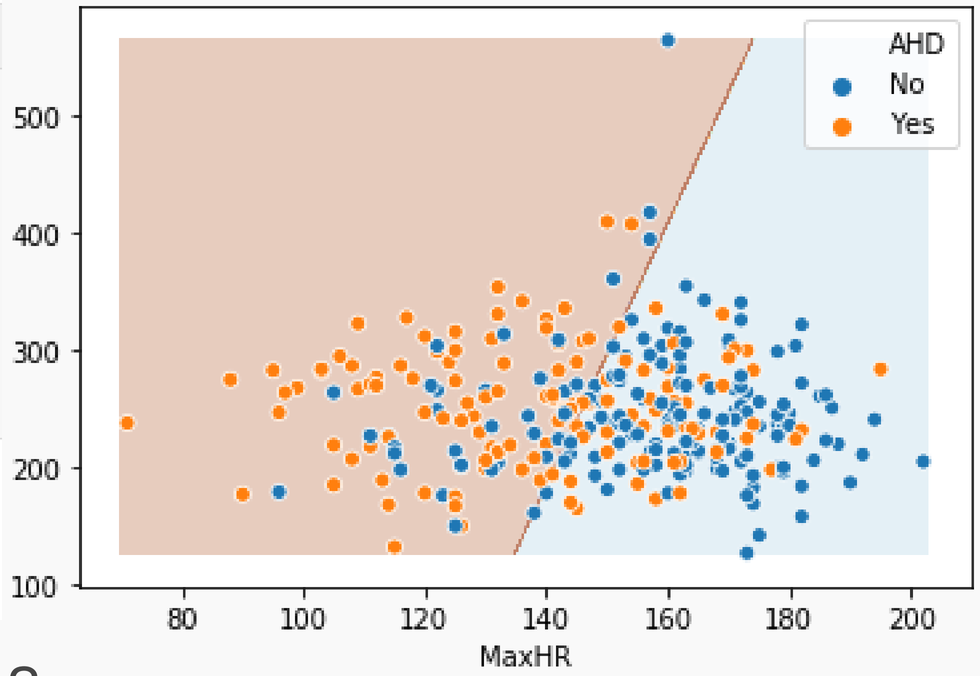
2D Classification in Logistic Regression: Example #1

A logistic regression model was fit to predict Y = AHD from X_1 = MaxHR and X_2 = Chol. Results shown below:



```
data_x = df_heart[['MaxHR','Chol']]
data_y = df_heart['AHD']

logreg = LogisticRegression(penalty='none', fit_intercept=True)
logreg.fit(data_x, data_y);


print('Estimated beta1, beta2: \n', logreg.coef_)
print('Estimated beta0: \n', logreg.intercept_)
```

```
Estimated beta1, beta2:
 [[-0.04388093  0.00391746]]
Estimated beta0:
 [5.42271131]
```

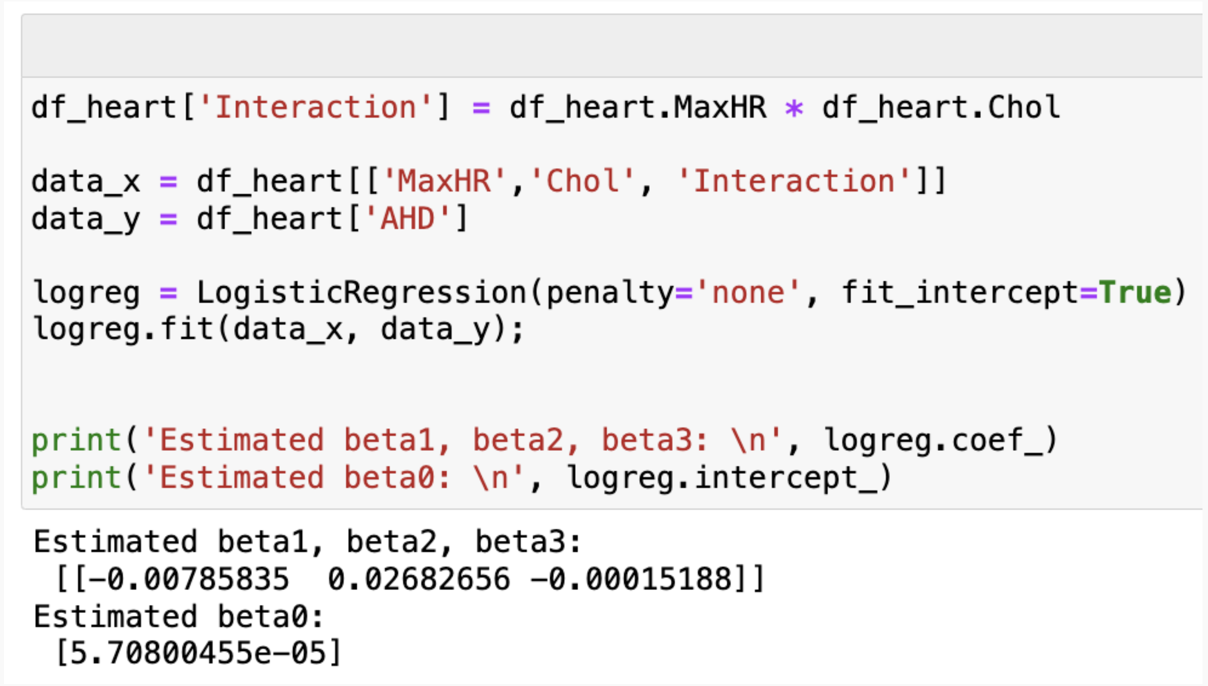


What will the decision boundary look like?

In logistic regression, decision boundaries are structured to be linear!

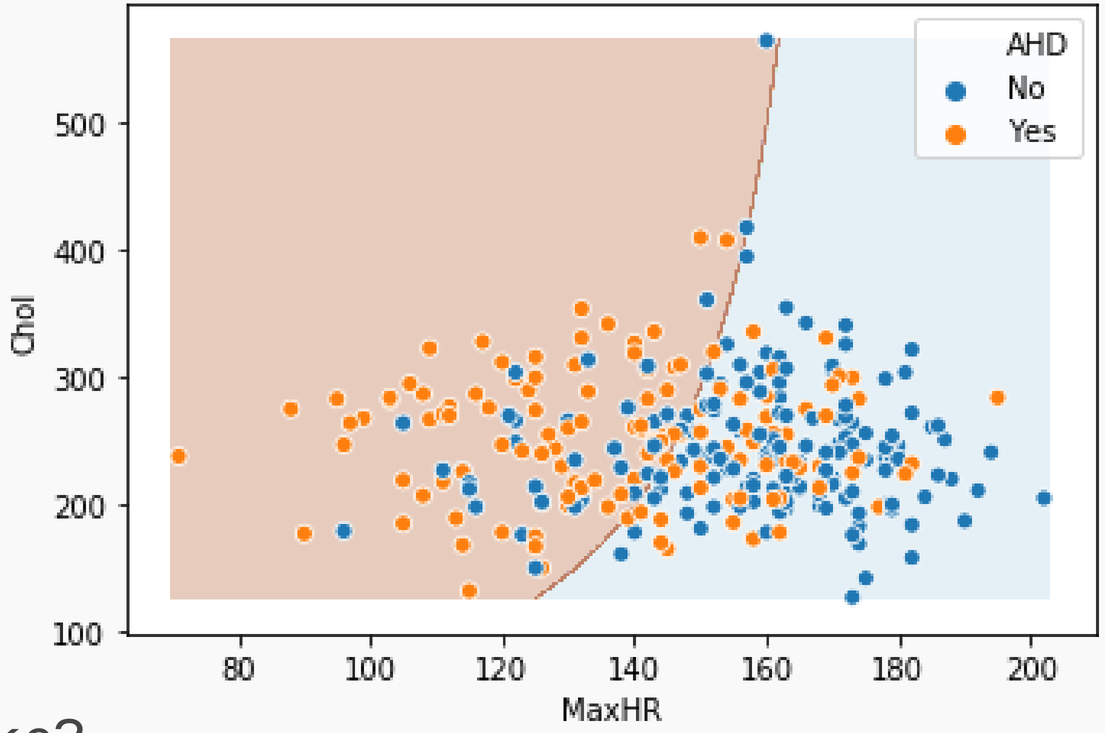
2D Classification in Logistic Regression: Example #2

A logistic regression model was fit to predict Y = AHD from X_1 = MaxHR and X_2 = Chol, and X_3 = their interaction. Results are shown below:



```
df_heart['Interaction'] = df_heart.MaxHR * df_heart.Chol

data_x = df_heart[['MaxHR','Chol', 'Interaction']]
data_y = df_heart['AHD']

logreg = LogisticRegression(penalty='none', fit_intercept=True)
logreg.fit(data_x, data_y);


print('Estimated beta1, beta2, beta3: \n', logreg.coef_)
print('Estimated beta0: \n', logreg.intercept_)
```

```
Estimated beta1, beta2, beta3:
 [[-0.00785835  0.02682656 -0.00015188]]
Estimated beta0:
 [5.70800455e-05]
```

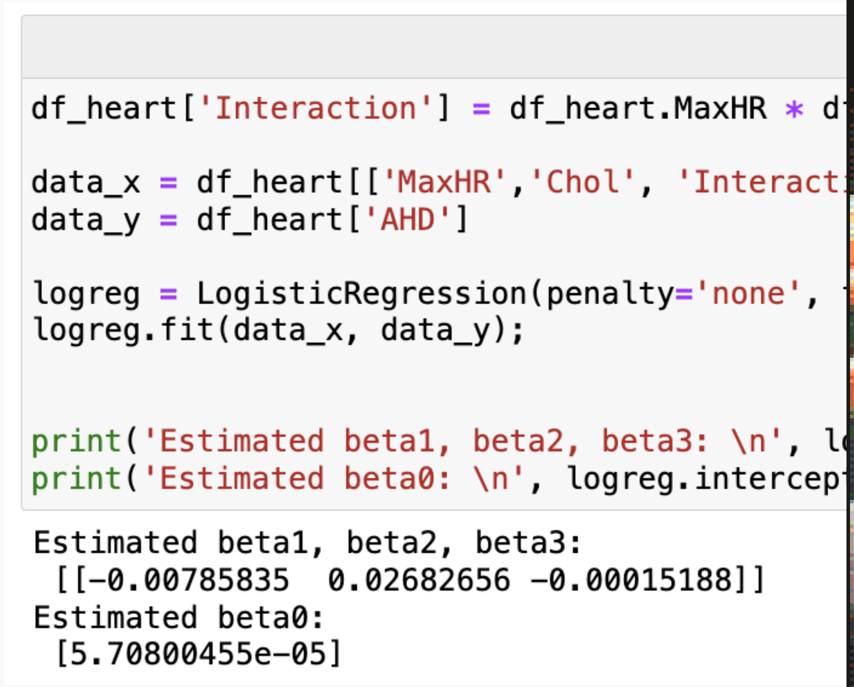


What will the decision boundary look like?

In logistic regression, decision boundaries are not always structured to be linear!

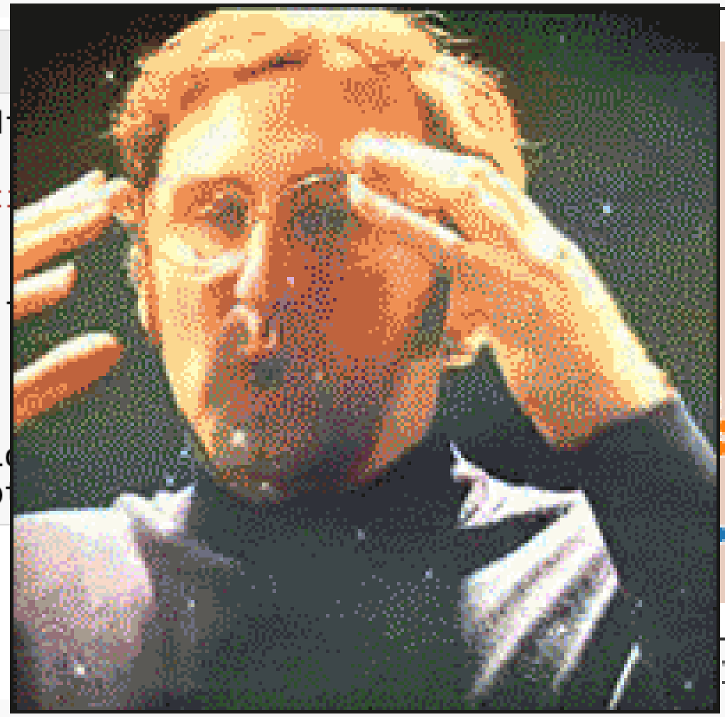
2D Classification in Logistic Regression: Example #2

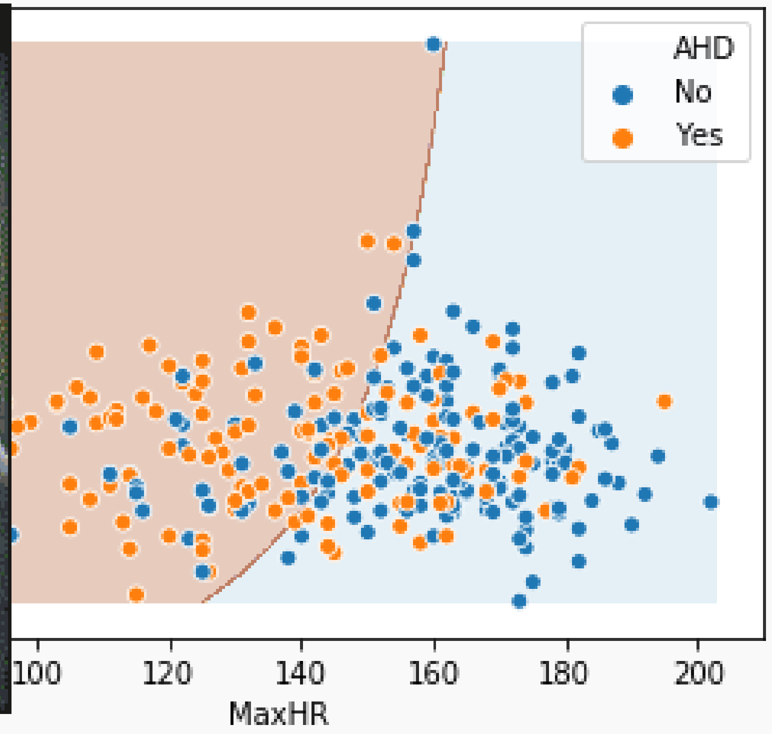
A logistic regression model was fit to predict Y = AHD from X_1 = MaxHR and X_2 = Chol, and X_3 = their interaction. Results are shown below:



```
df_heart['Interaction'] = df_heart.MaxHR * d

data_x = df_heart[['MaxHR','Chol', 'Interact
data_y = df_heart['AHD']

logreg = LogisticRegression(penalty='none',
logreg.fit(data_x, data_y);


print('Estimated beta1, beta2, beta3: \n', l
print('Estimated beta0: \n', logreg.intercep
```

```
Estimated beta1, beta2, beta3:
 [[-0.00785835  0.02682656 -0.00015188]]
Estimated beta0:
 [5.70800455e-05]
```



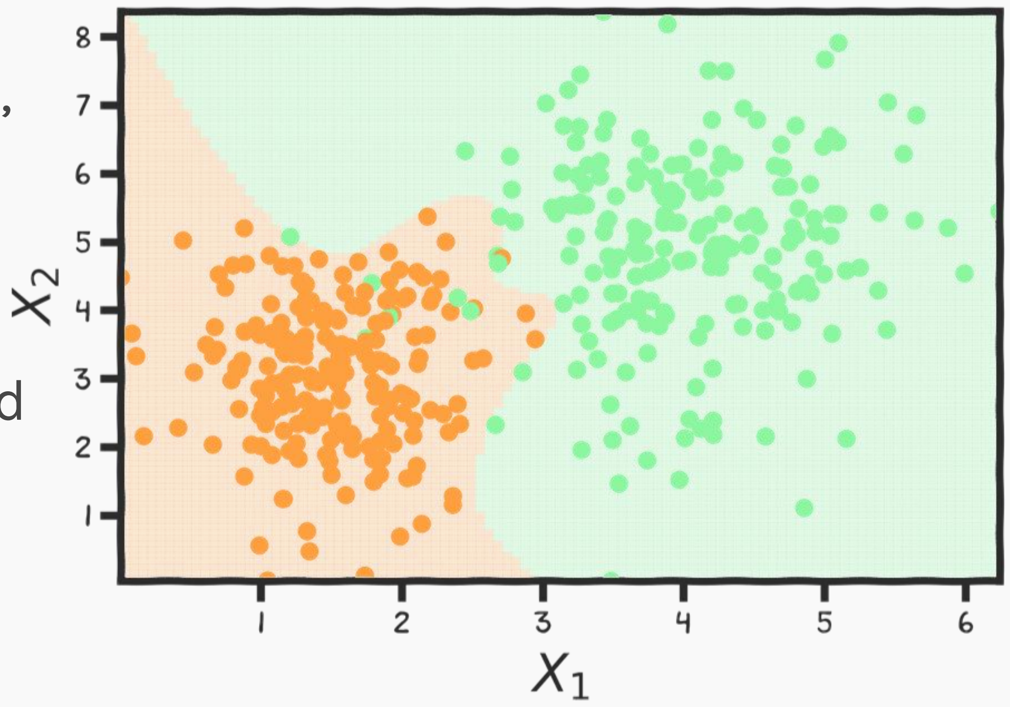


What will the decision boundary look like?

In logistic regression, decision boundaries are not always structured to be linear!

Polynomial Logistic Regression

We saw a 2-D plot last time which had two predictors, X_1, X_2. A similar one is shown here but the decision boundary is again not linear.



We can extend multiple Logistic Regression as we did with polynomial regression:

We transform the data by adding new predictors:

$$\tilde{x} = [1, \tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_M]$$

where $\tilde{x}_k = x^k$.

The polynomial Logistic Regression can be expressed as:

$$\boxed{\log\left(\frac{P(Y=1)}{1-P(Y=1)}\right) = \tilde{X}\beta}$$

Geometry of Decision Boundaries (Logistic Regression)

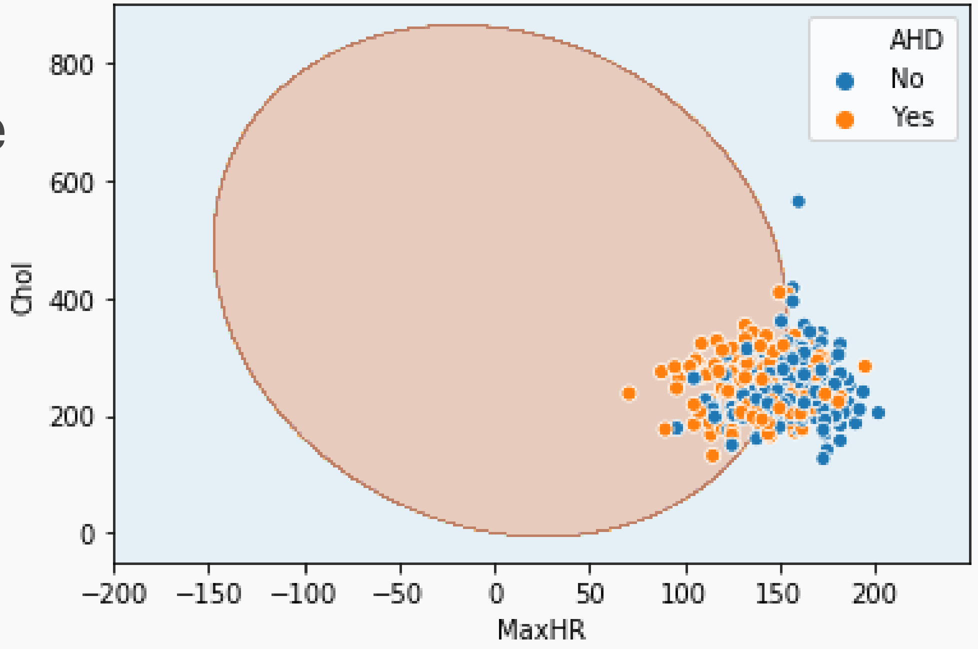
$$\boxed{\log\left(\frac{P(Y=1)}{1-P(Y=1)}\right) = \tilde{X}\beta}$$

Thus we can define our logistic regression model to achieve a desired geometry.

For example, what set for $X = f(X_1, X_2)$ should we choose if we want a *circular* decision boundary?

$$X = \{X_1, X_1^2, X_2, X_2^2, X_1 X_2\}$$

What could be an alternative modeling approach (think outside of logistic regression)?



Outline

- Inference in Logistic Regression
- Multiple Logistic Regression
- Classification Decision Boundaries
- Interpreting interactions in logistic regression
- **Regularization in Logistic Regression**
- Multiclass 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:

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

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**):

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

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 λ they use a constant that is $C = \frac{1}{\lambda}$.

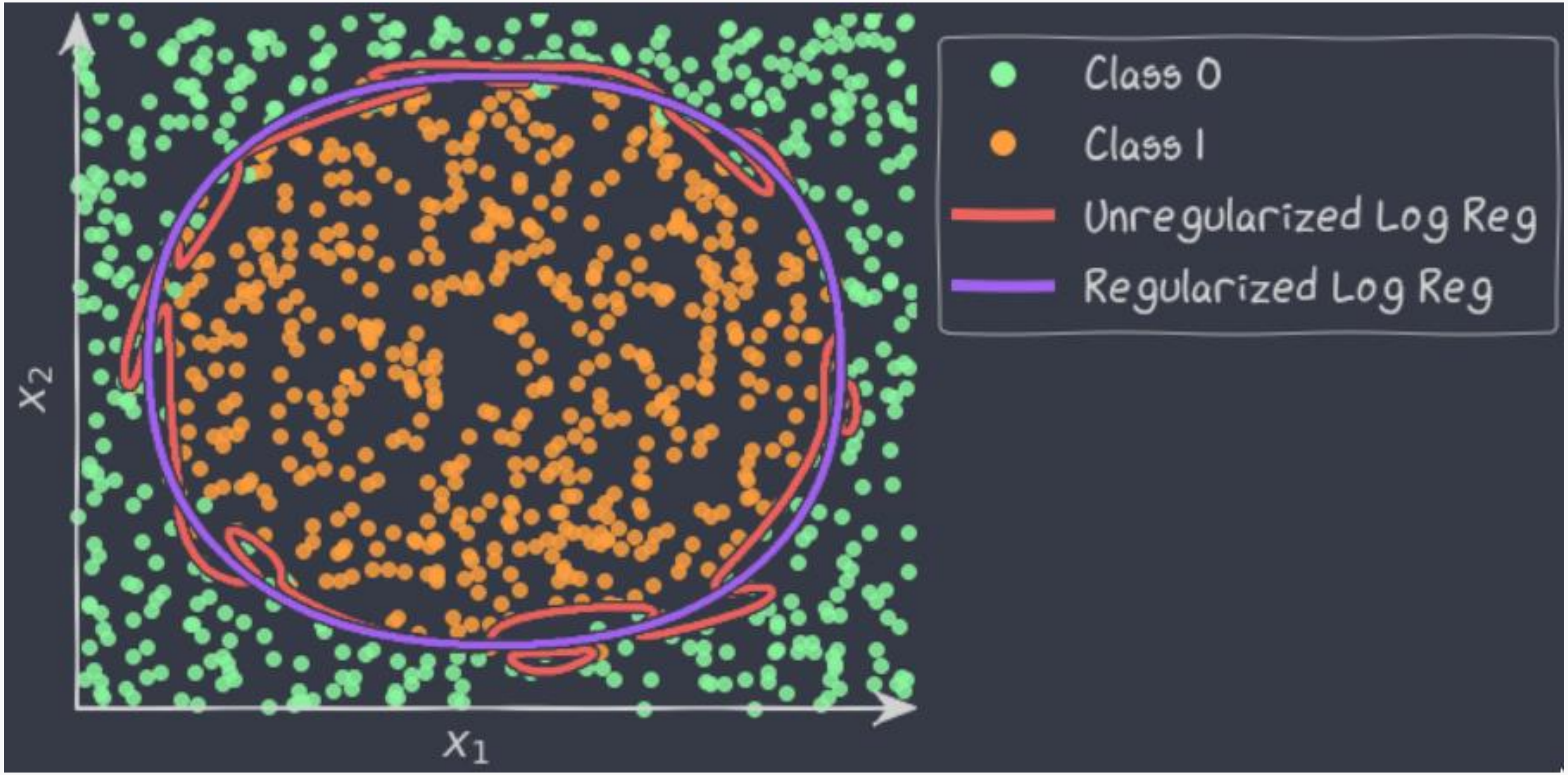
Regularization in Logistic Regression: tuning λ

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

- Inference in Logistic Regression
- Multiple Logistic Regression
- Classification Decision Boundaries
- Interpreting interactions in logistic regression
- Regularization in Logistic Regression
- **Multiclass 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).

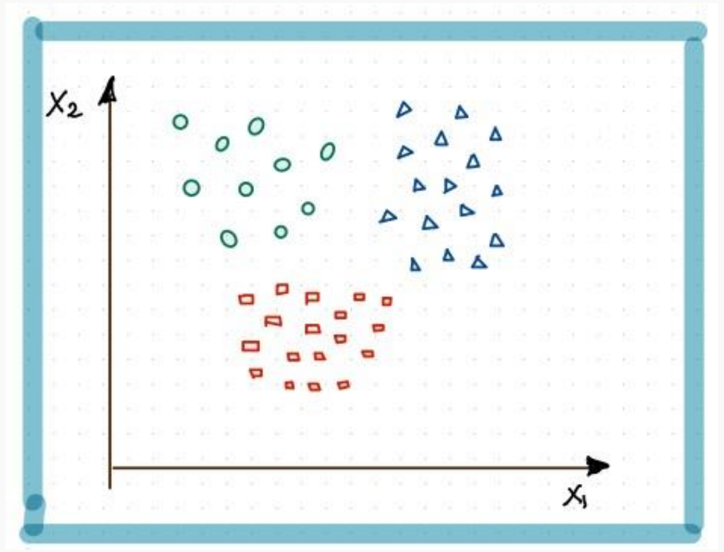
Ignoring interactions, how many parameters would need to be estimated?

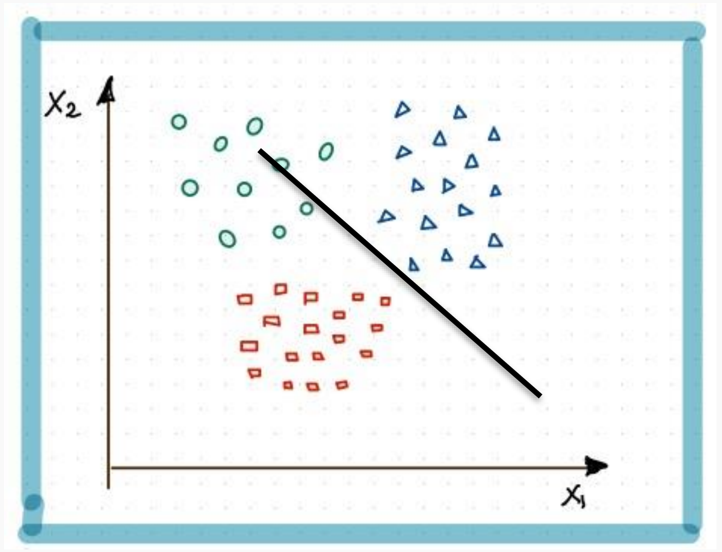
How could these models be used to estimate the probability of an individual falling in each concentration?

Multinomial Logistic

Classifying three classes:

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





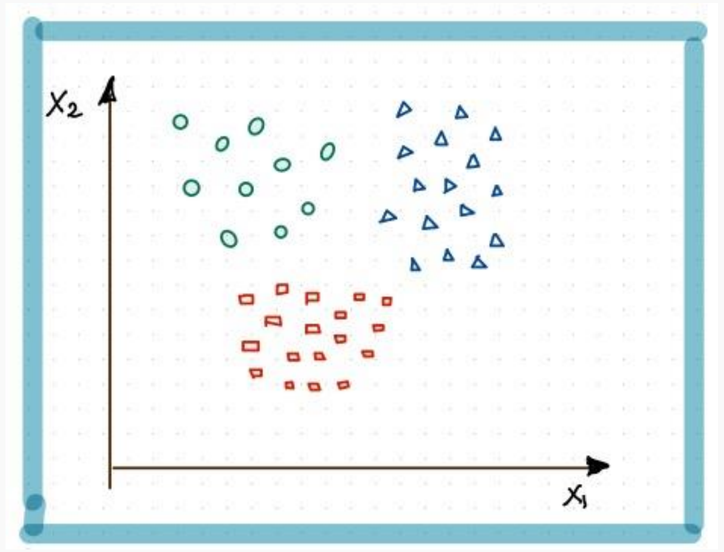
Blue vs Red

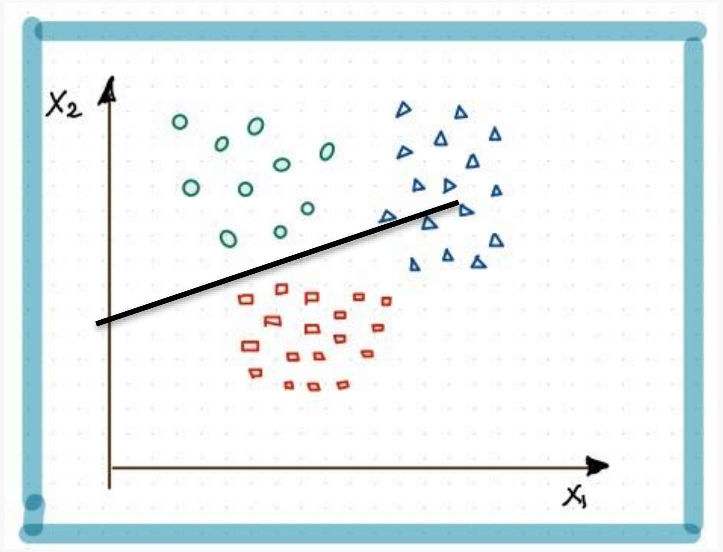
$$\ln\left(\frac{P(b)}{P(r)}\right) = \beta_b X$$

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:

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

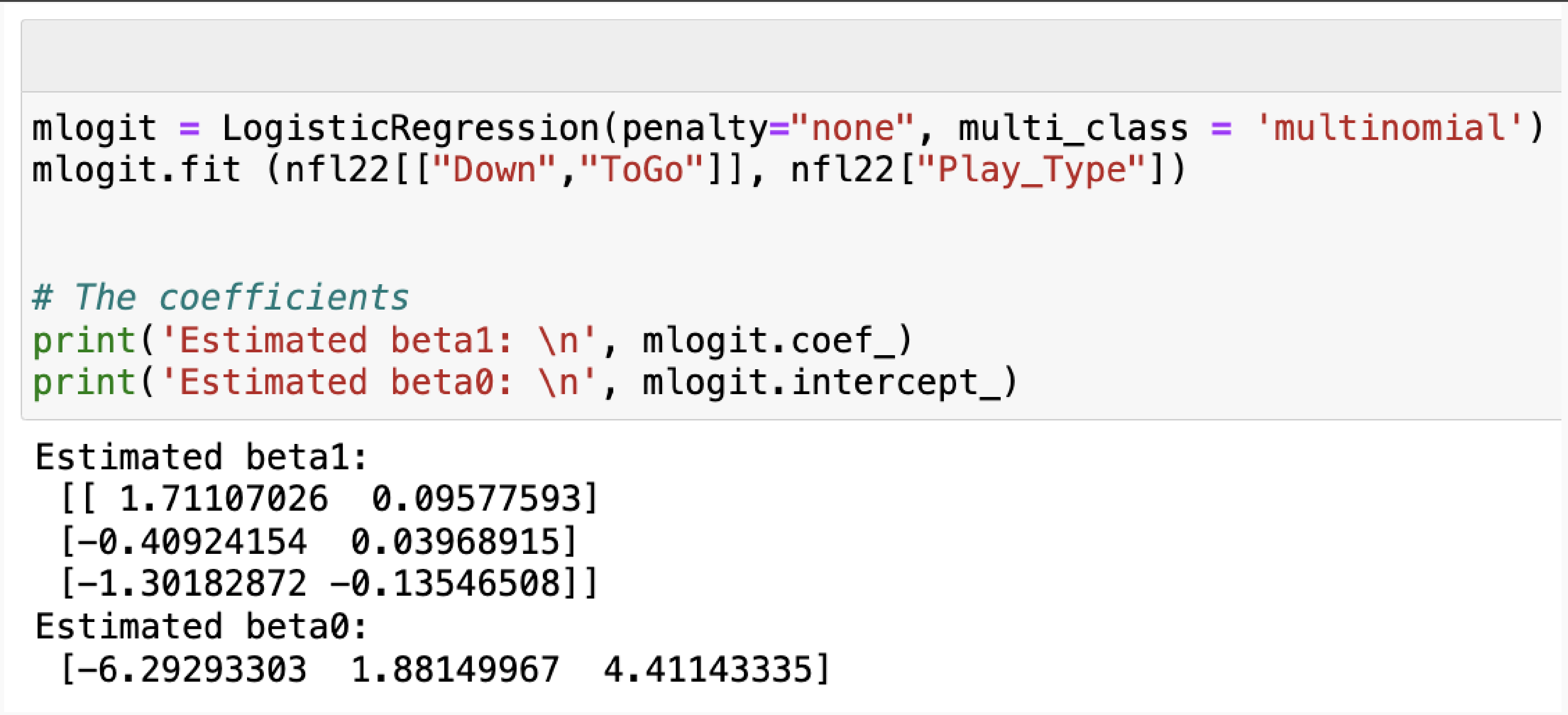
$$\ln\left(\frac{P(Y=2)}{P(Y=K)}\right) = \beta_{0,2} + \beta_{1,2}X_1 + \beta_{2,2}X_2 + \cdots + \beta_{p,2}X_p$$

$$\vdots$$

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

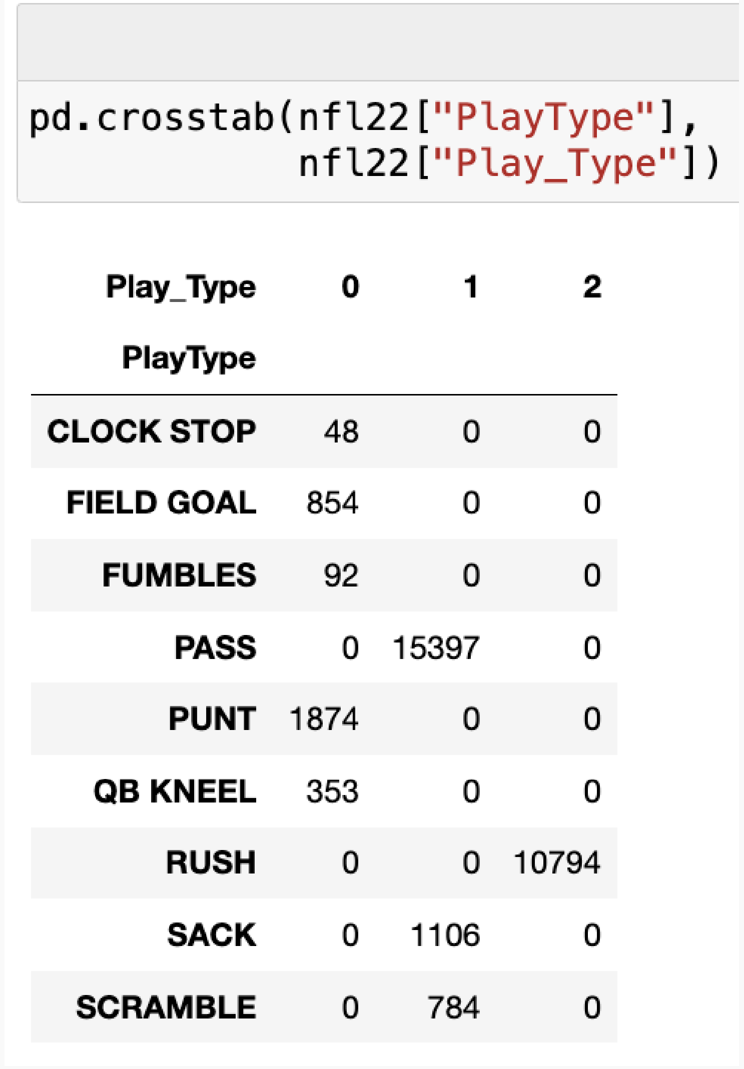
Each separate model can be fit as independent standard logistic regression models!

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]
```



```
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

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

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):

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

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.

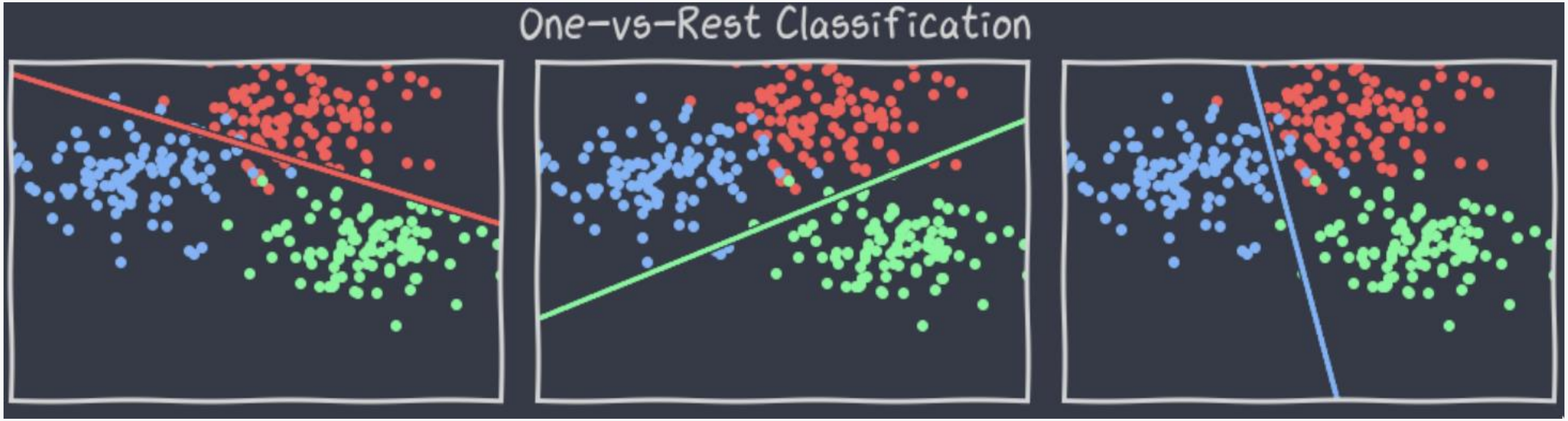
If there are 3 classes, then 3 separate logistic regressions are fit, where the probability of each category is predicted over the rest of the categories combined. So for the concentration example, 3 models would be fit:

- 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.

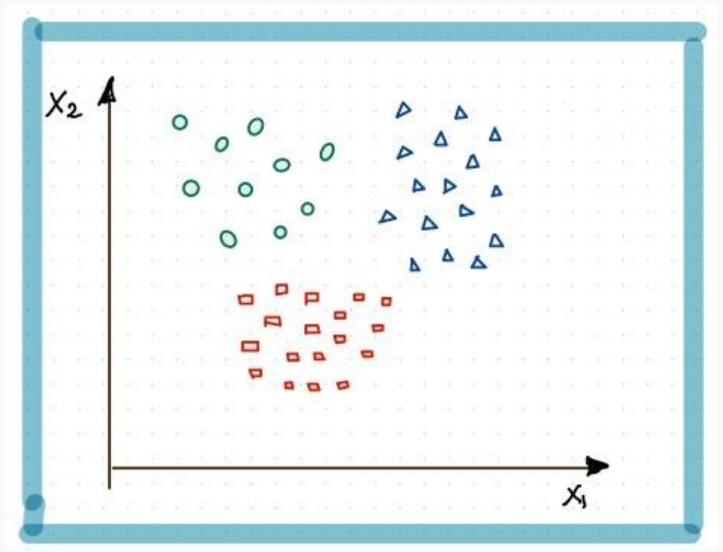
One vs. Rest (ovr)

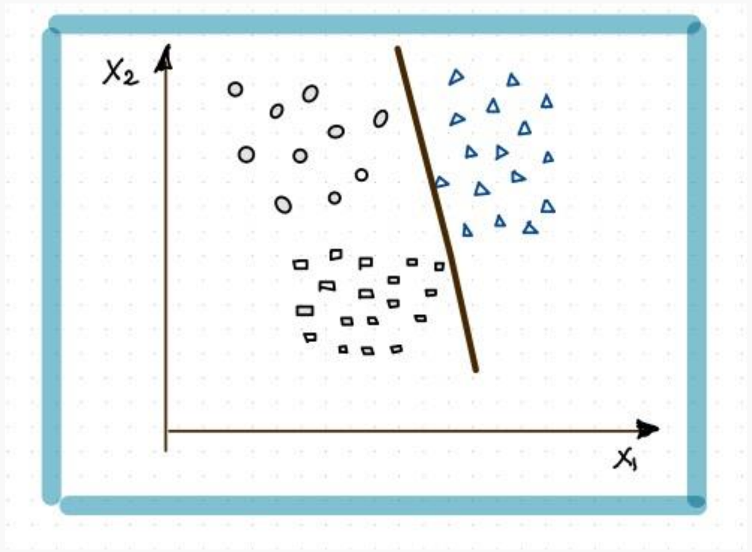
Classifying three classes,
Red, Blue and Green can be turn into three binary Logistic Regressions



One vs. Rest (ovr)

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



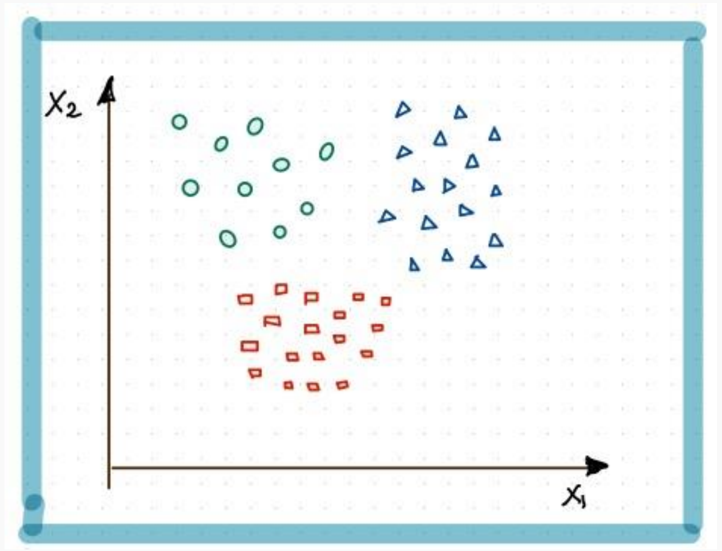


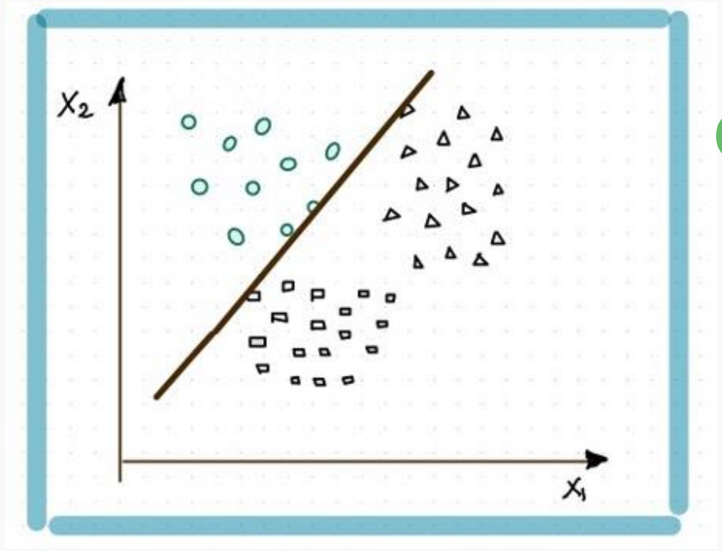
Blue vs others

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

One vs. Rest (ovr)

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



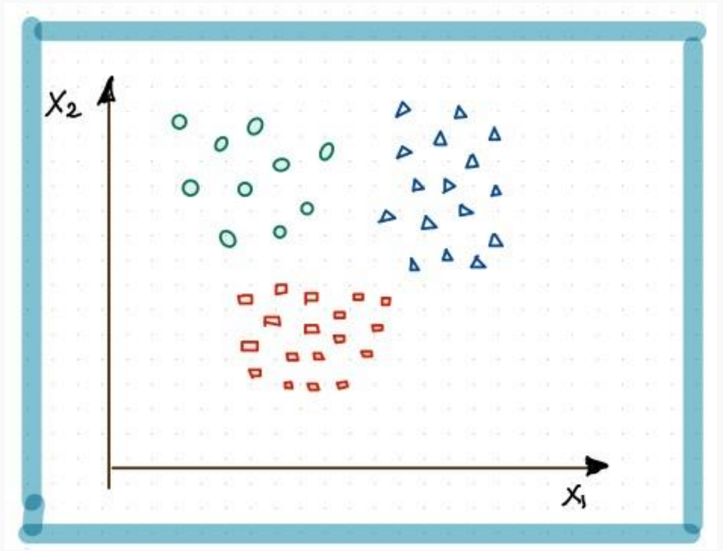


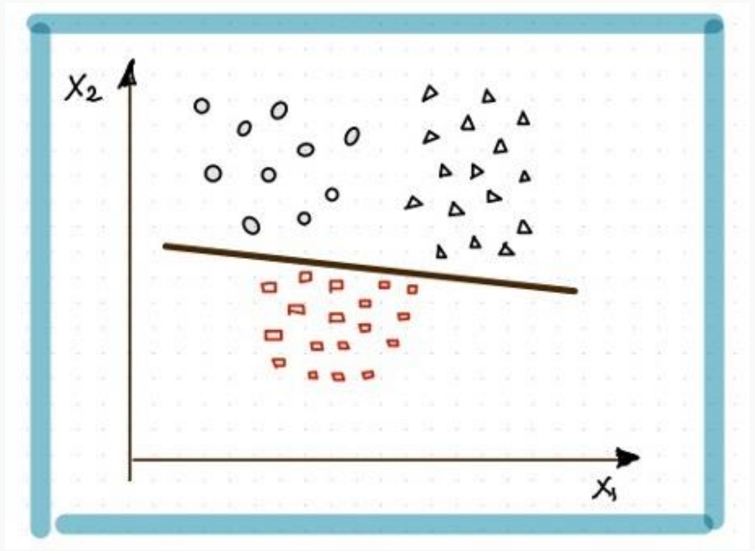
Green vs others

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

One vs. Rest (ovr)

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





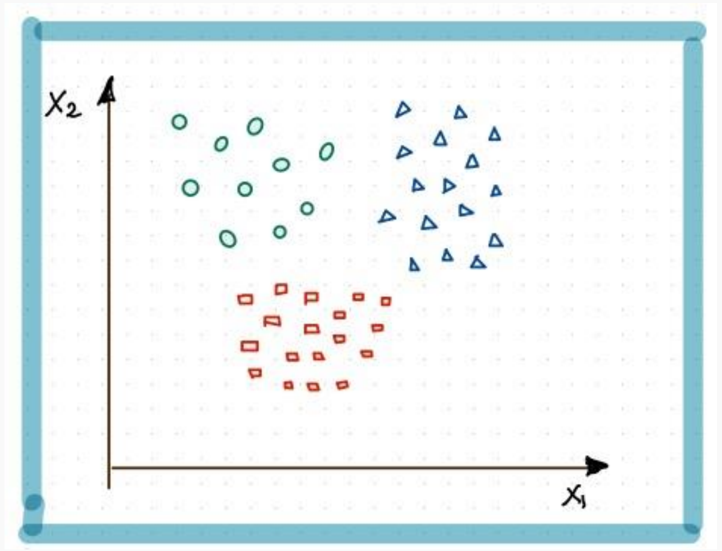
Red vs others

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

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:

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

One vs. Rest (OvR) Logistic Regression: the model

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

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

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

$$\vdots$$

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

Again, each separate model can be fit as independent standard logistic regression models!

Softmax

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

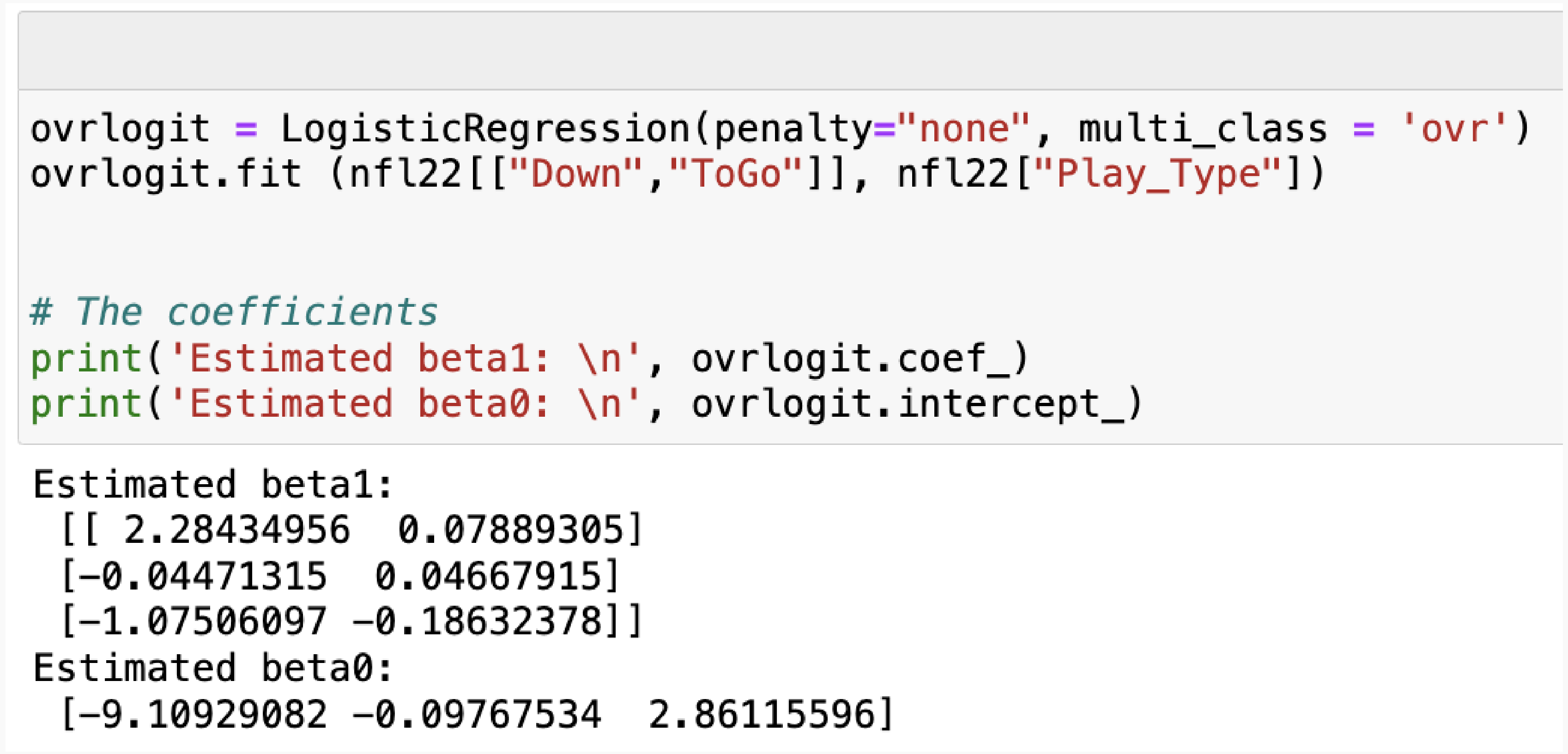
The ***softmax*** function is used. That is, the weights are just normalized for each predicted probability. AKA, predict the 3 class probabilities from each of the 3 models, and just rescale so they add up to 1.

Mathematically that is:

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

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

OVR Logistic Regression in Python

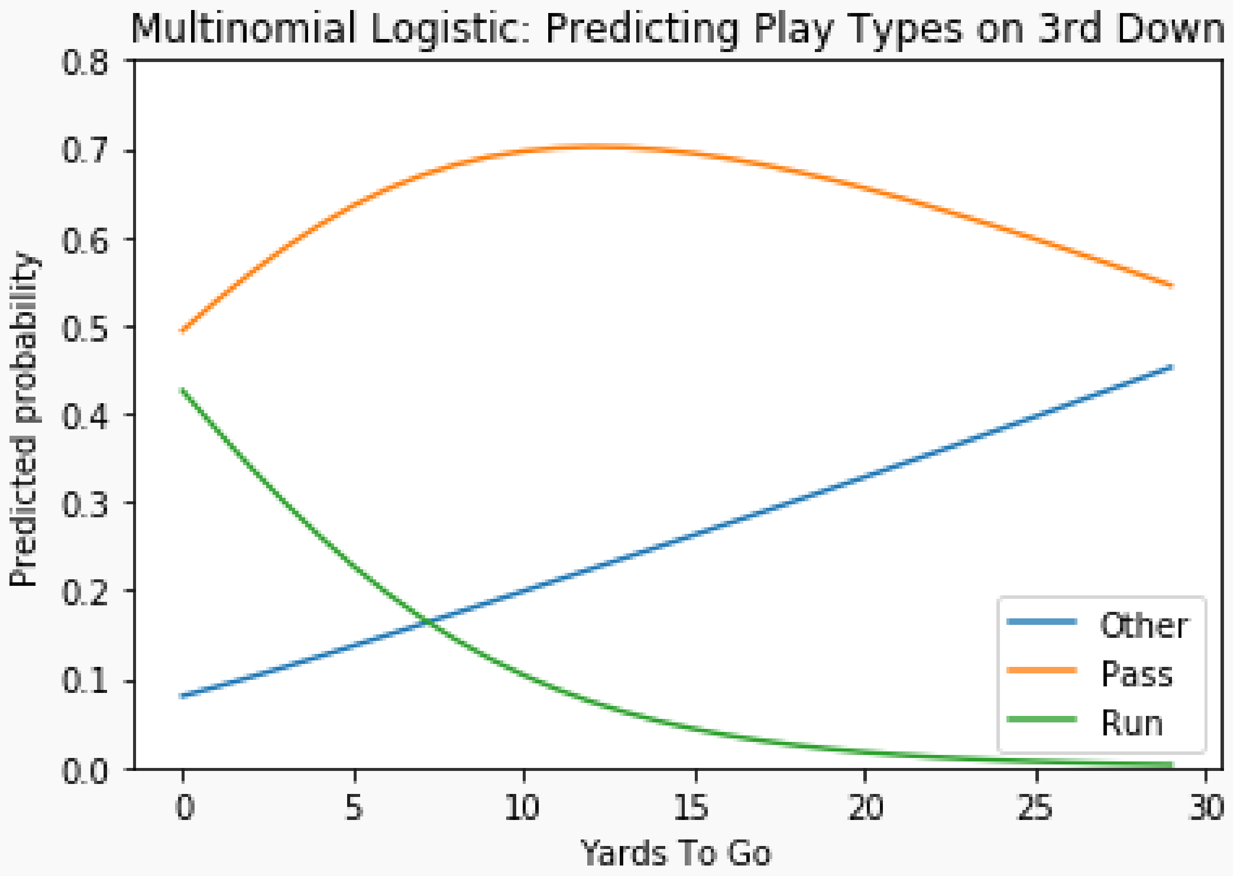


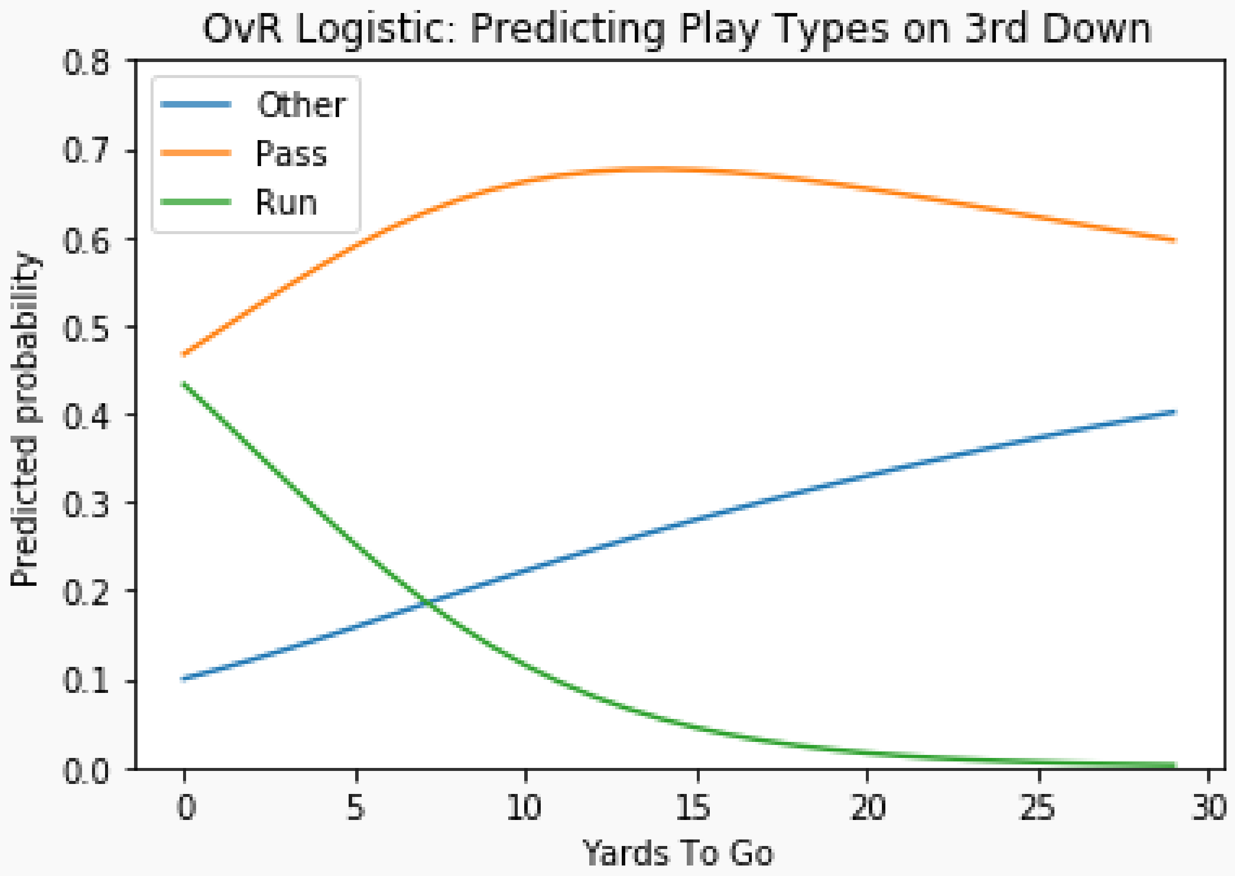
```
ovrlogit = LogisticRegression(penalty="none", multi_class = 'ovr')
ovrlogit.fit (nfl22[["Down","ToGo"]], nfl22["Play_Type"])


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

```
Estimated beta1:
 [[ 2.28434956  0.07889305]
 [-0.04471315  0.04667915]
 [-1.07506097 -0.18632378]]
Estimated beta0:
 [-9.10929082 -0.09767534  2.86115596]
```

Phew! This one is as expected ☺

Predicting Type of Play in the NFL: Multinomial vs. OvR



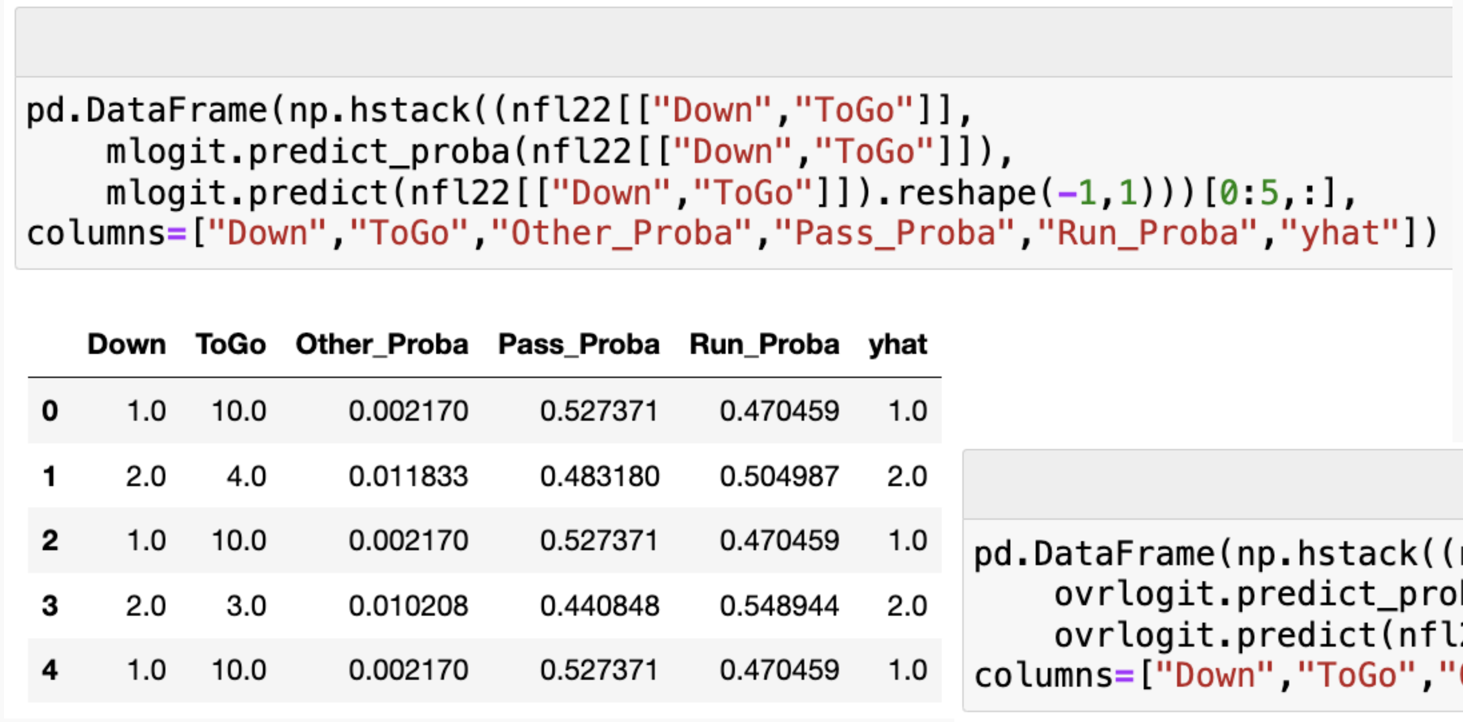


Classification for more than 2 Categories

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 (or other classification model) will instead have to select the group with **the largest estimated probability**.

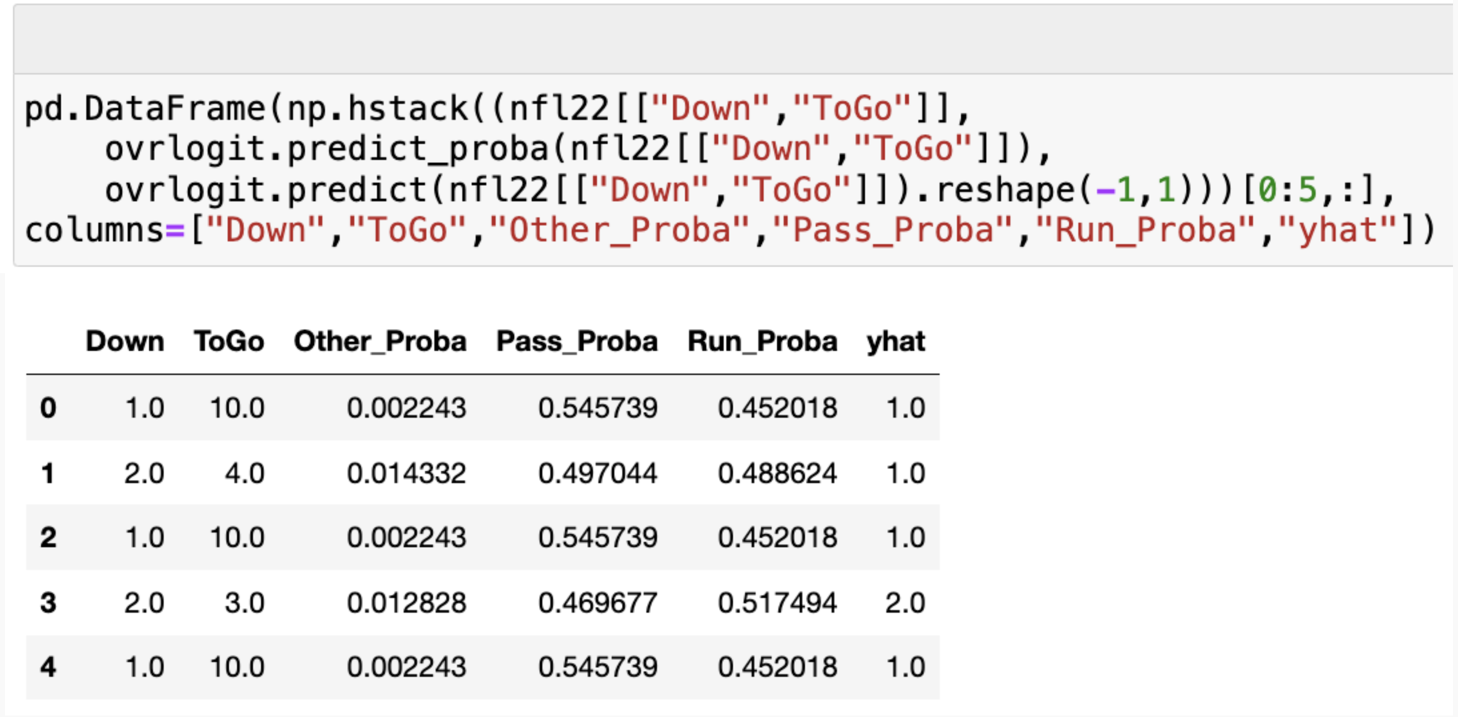
The classification boundaries are then much more difficult to determine mathematically. We will not get into the algorithm for determining these in this class, but we can rely on `predict` and `predict_proba`!

Prediction using Multiclass Logistic Regression



```
pd.DataFrame(np.hstack((nfl22[["Down","ToGo"]],
    mlogit.predict_proba(nfl22[["Down","ToGo"]]),
    mlogit.predict(nfl22[["Down","ToGo"]]).reshape(-1,1)))[0:5,:],
columns=["Down","ToGo","Other_Proba","Pass_Proba","Run_Proba","yhat"])
```

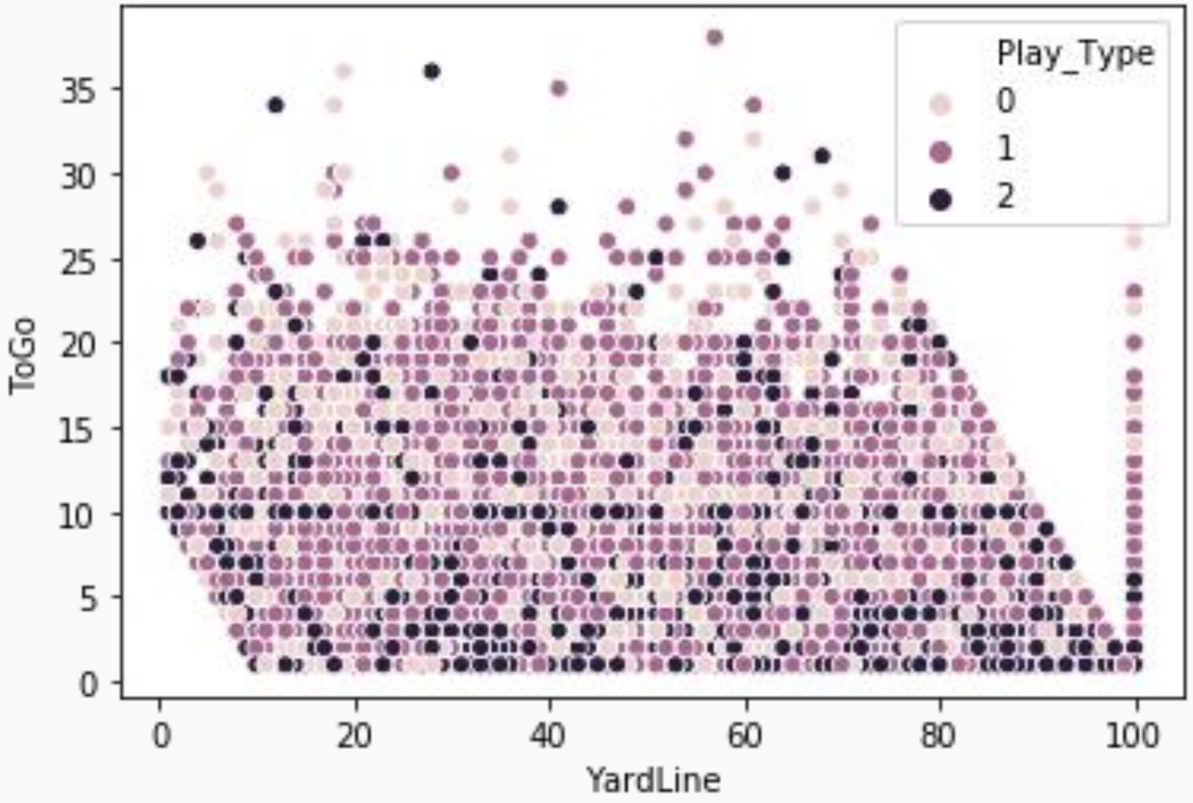
	Down	ToGo	Other_Proba	Pass_Proba	Run_Proba	yhat
0	1.0	10.0	0.002170	0.527371	0.470459	1.0
1	2.0	4.0	0.011833	0.483180	0.504987	2.0
2	1.0	10.0	0.002170	0.527371	0.470459	1.0
3	2.0	3.0	0.010208	0.440848	0.548944	2.0
4	1.0	10.0	0.002170	0.527371	0.470459	1.0

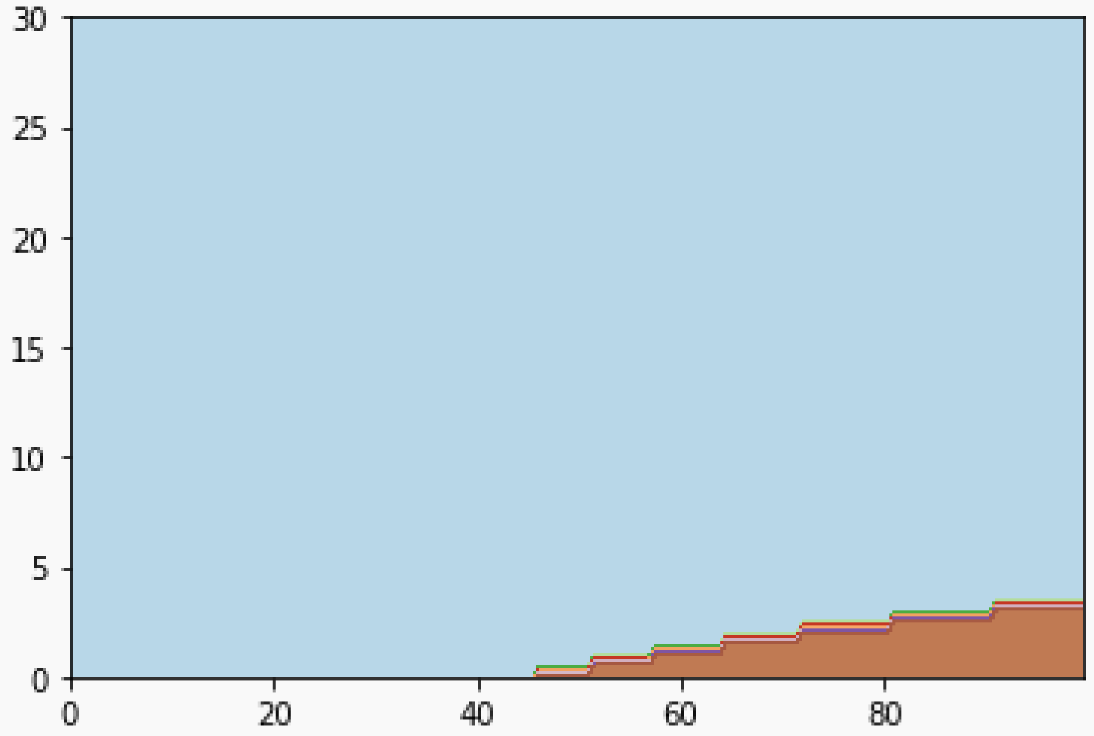


```
pd.DataFrame(np.hstack((nfl22[["Down","ToGo"]],
    ovrlogit.predict_proba(nfl22[["Down","ToGo"]]),
    ovrlogit.predict(nfl22[["Down","ToGo"]]).reshape(-1,1)))[0:5,:],
columns=["Down","ToGo","Other_Proba","Pass_Proba","Run_Proba","yhat"])
```

	Down	ToGo	Other_Proba	Pass_Proba	Run_Proba	yhat
0	1.0	10.0	0.002243	0.545739	0.452018	1.0
1	2.0	4.0	0.014332	0.497044	0.488624	1.0
2	1.0	10.0	0.002243	0.545739	0.452018	1.0
3	2.0	3.0	0.012828	0.469677	0.517494	2.0
4	1.0	10.0	0.002243	0.545739	0.452018	1.0

Classification Boundary for 3+ Classes in sklearn





Multinomial vs. ovr

Multinomial is slightly more efficient in estimation since there are technically 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.

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.

Staying Very Classy

Outline

- Inference in Logistic Regression
- Multiple Logistic Regression
- Classification Decision Boundaries
- Interpreting interactions in logistic regression
- Regularization in Logistic Regression
- Multiclass 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.)

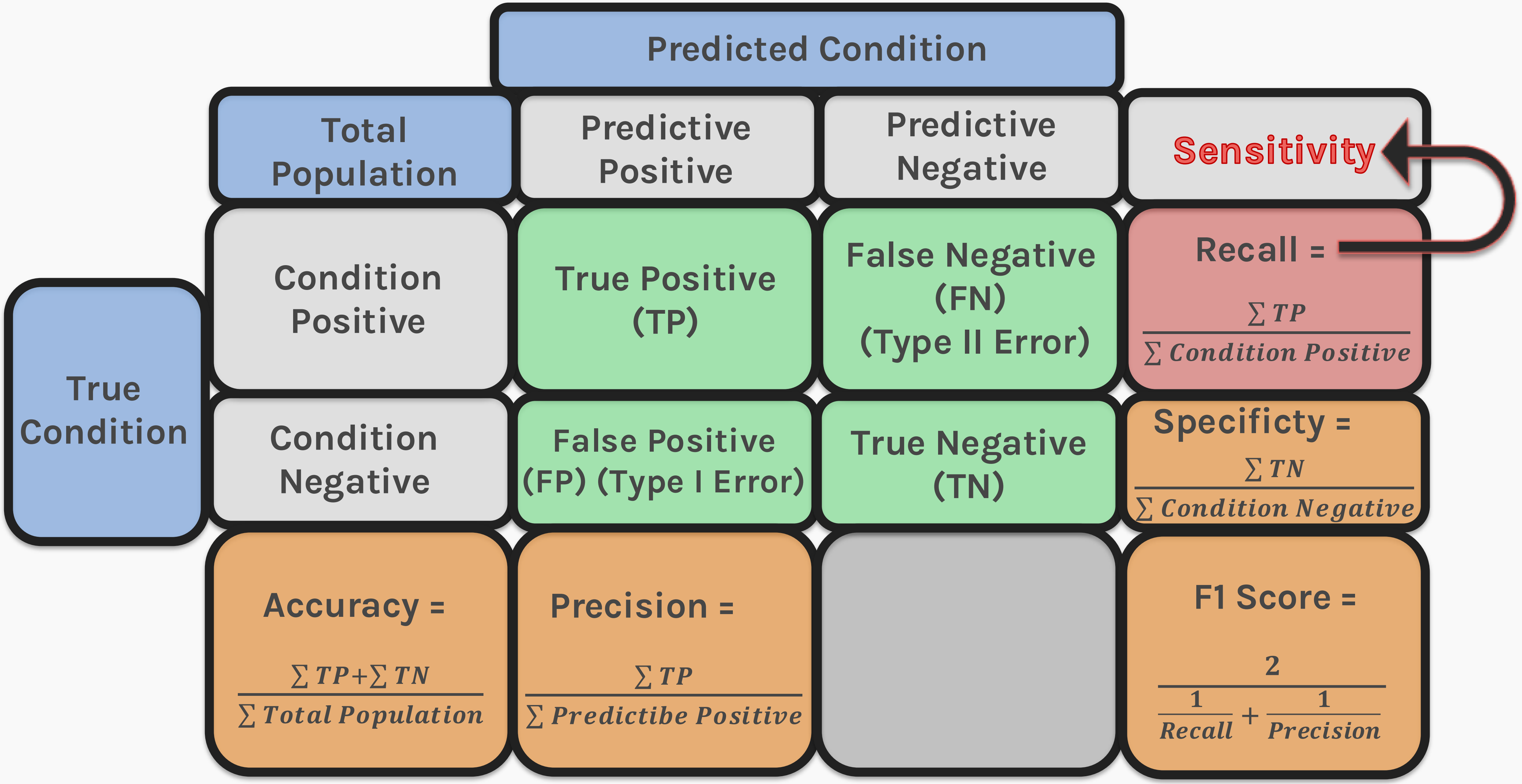
Bayes' theorem can be rewritten for diagnostic tests:

$$\boxed{P(D+|T+)} = \frac{\boxed{P(T+|D+)}\boxed{P(D+)}}{\boxed{P(T+|D+)}\boxed{P(D+)} + \underbrace{\boxed{P(T+|D-)}}_{\boxed{1 - \textit{Specificity}}}P(D-)}$$

These probability quantities can then be defined as:

- $\boxed{\textit{Sensitivity}: P(T+|D+)}$
- $\boxed{\textit{Specificity}: P(T-|D-)}$
- $\boxed{\textit{Prevalence}: P(D+)}$
- $\boxed{\textit{Positive Predictive Value}: P(D+|T+)}$
- *Negative Predictive Value*: $P(D-|T-)$

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)	Specificity = $\frac{\sum TN}{\sum Condition\ Negative}$
	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?

Why do we care?

As data scientists, why do we care about diagnostic testing from the medical world? (hint: it's not just because Kevin is a trained biostatistician!)

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 $\hat{Y}_i = k$ the event that we correctly predict it to be in class *k*. Then Bayes' rule states that our *Positive Predictive Value* for classification is:

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

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

Error in Classification

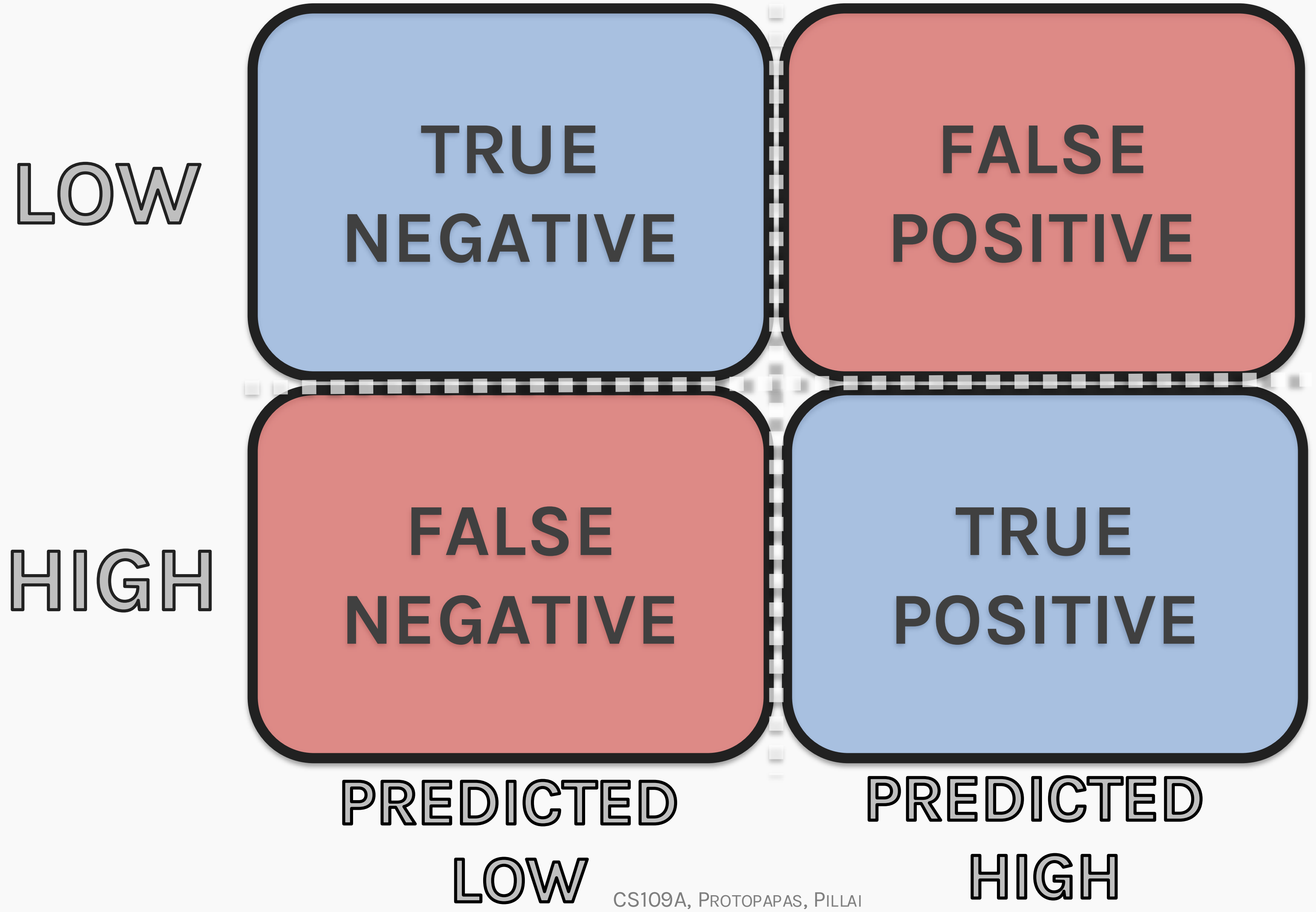
There are 2 major types of error in classification problems based on a binary outcome. They are:

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

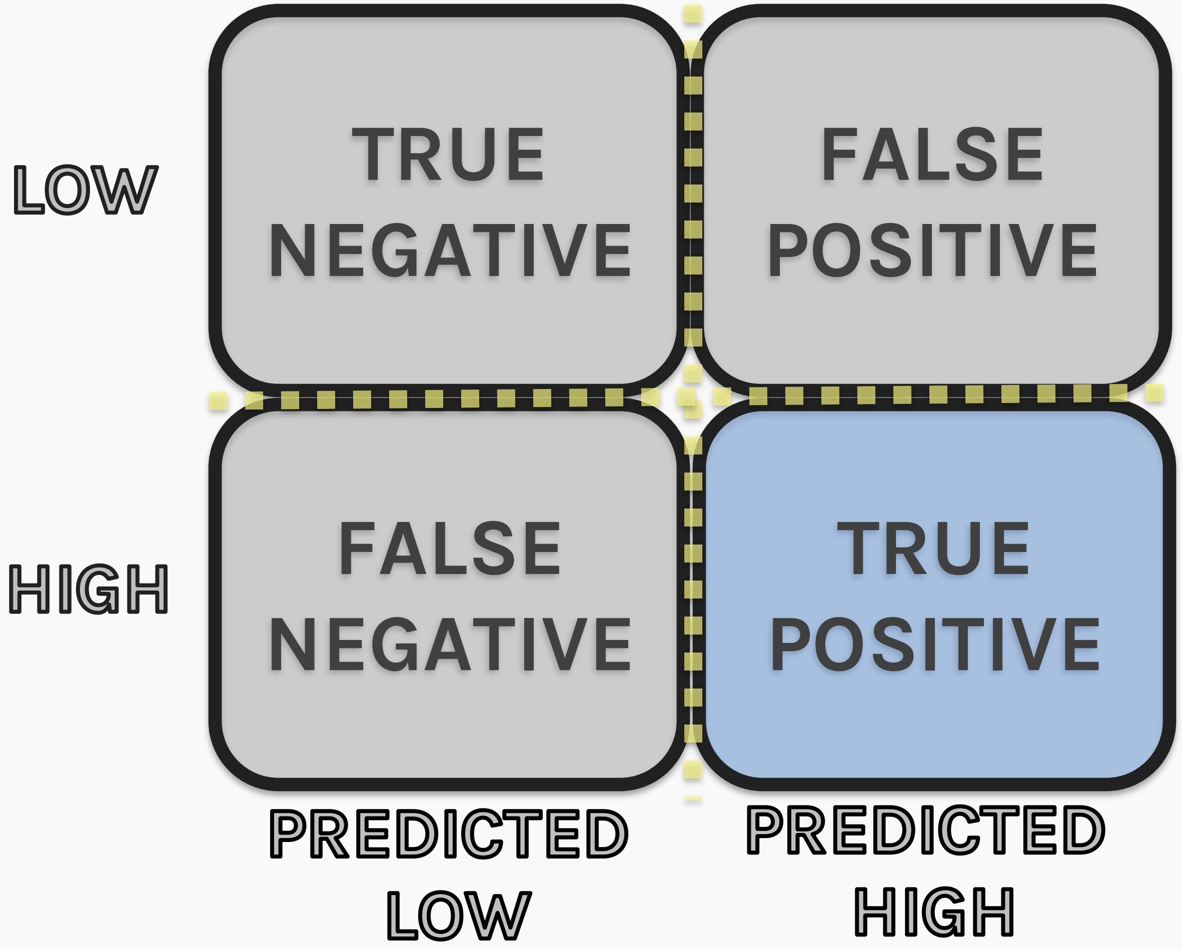


	PREDICTED LOW	PREDICTED HIGH
LOW	TRUE NEGATIVE	FALSE POSITIVE
HIGH	FALSE NEGATIVE	TRUE POSITIVE

The 'Confusion' Matrix

TRUE POSITIVE (TP)

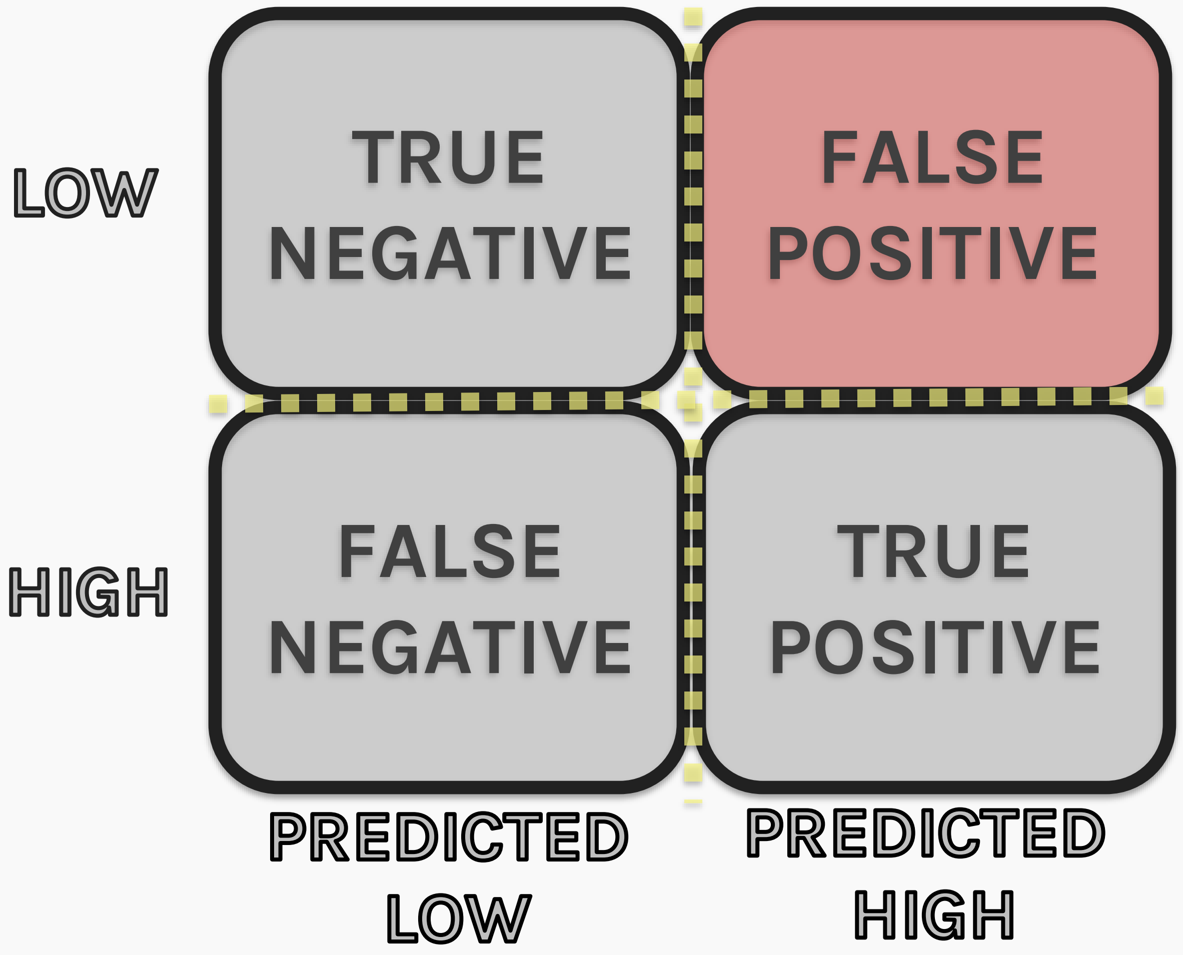
- Samples that are positive that the classifier predicts as positive are called True Positives.
- Example: a positive Covid test result would be a TRUE POSITIVE if you actually have Covid.



The 'Confusion' Matrix

FALSE POSITIVE (FP)

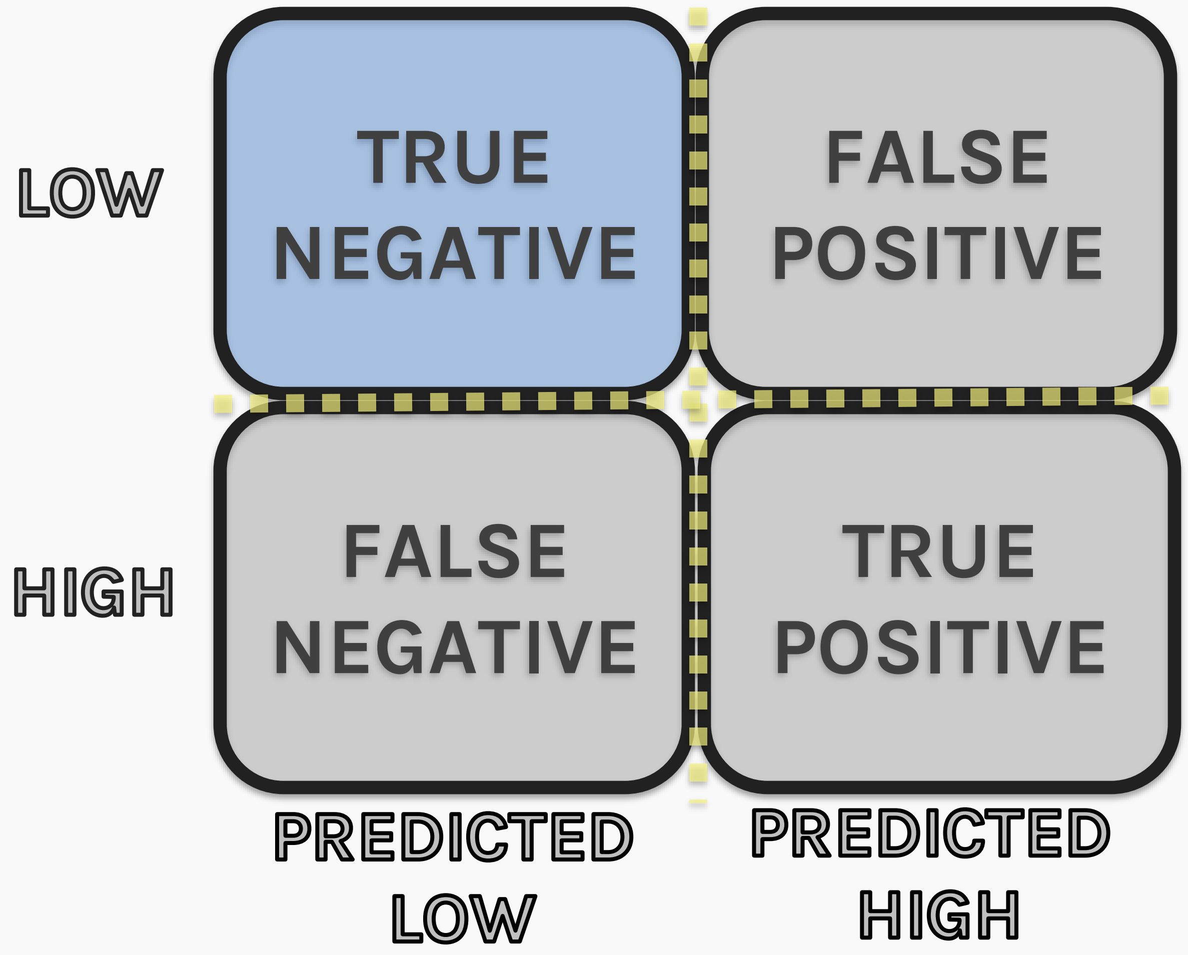
- Samples that are negative that the classifier predicts as positive are called False Positives.
- Example: a positive Covid test result would be a FALSE POSITIVE if you actually don't have Covid.



The 'Confusion' Matrix

TRUE NEGATIVE (TN)

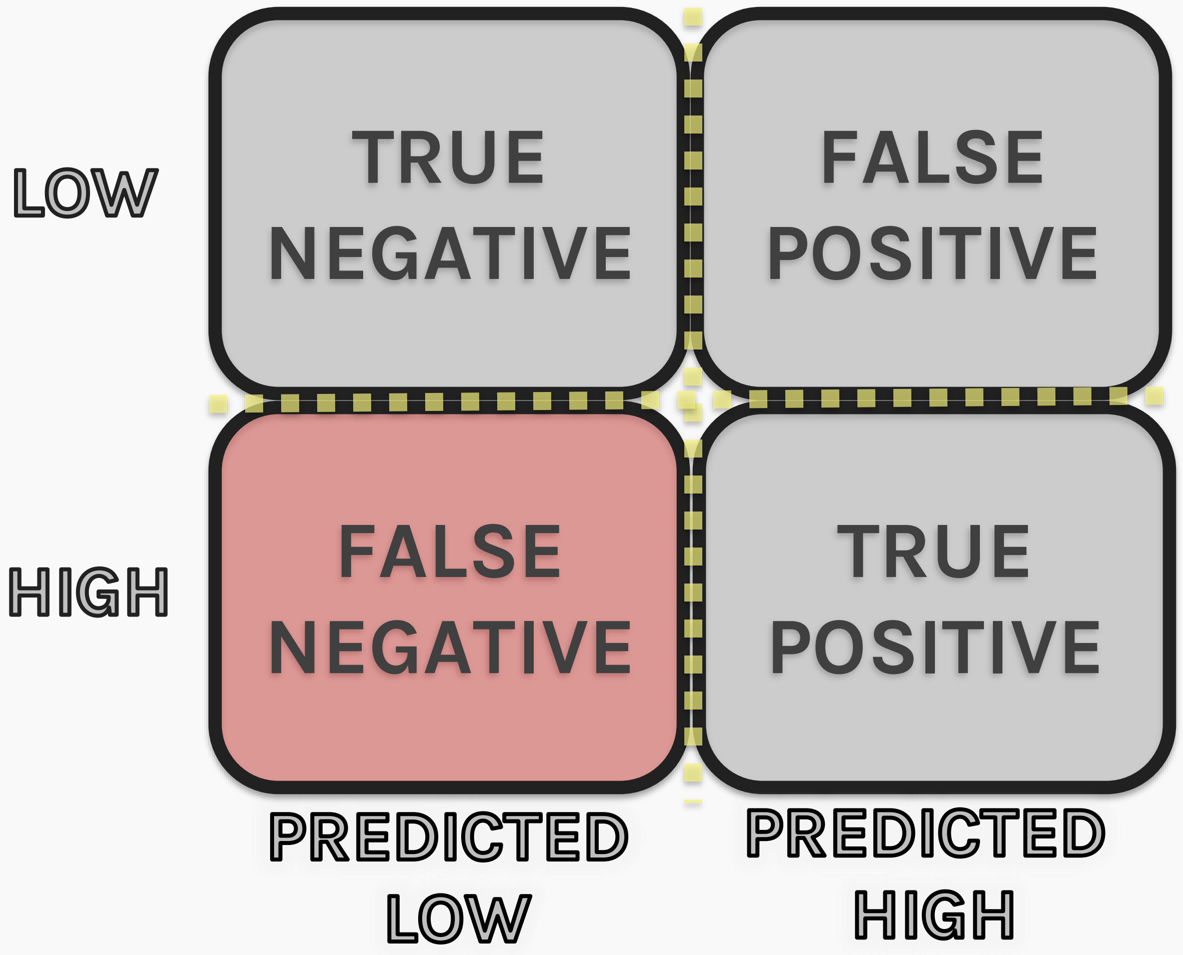
- Samples that are negative that the classifier predicts as negative are called True Negatives.
- Example: a negative Covid test result would be a TRUE NEGATIVE if you actually don't have Covid.



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

	PREDICTED LOW	PREDICTED HIGH
LOW	TRUE NEGATIVE	FALSE POSITIVE
HIGH	FALSE NEGATIVE	TRUE POSITIVE

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.

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 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 π to be some level other than 0.50.

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

Other Confusion tables

Based on π = 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 π = 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

- Inference in Logistic Regression
- Multiple Logistic Regression
- Classification Decision Boundaries
- Interpreting interactions in logistic regression
- Regularization in Logistic Regression
- Multiclass 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).

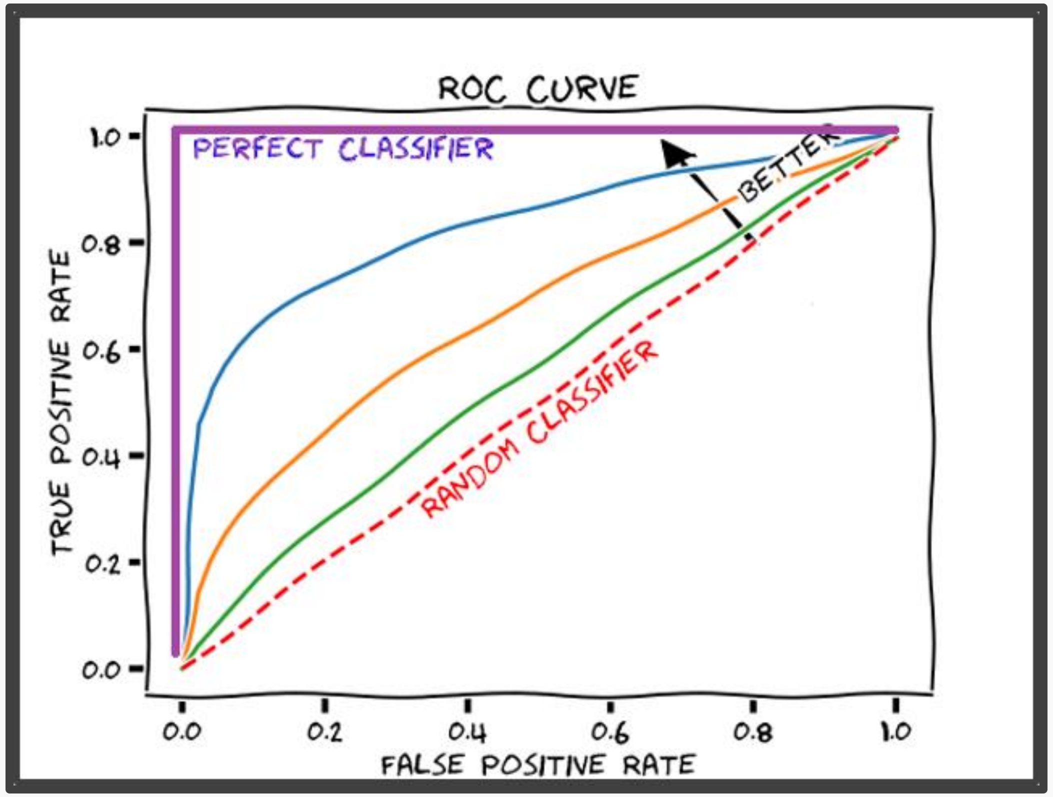
The vertical axis displays the true positive predictive value and the horizontal axis depicts the true negative predictive value.

What is the shape of an ideal ROC curve?

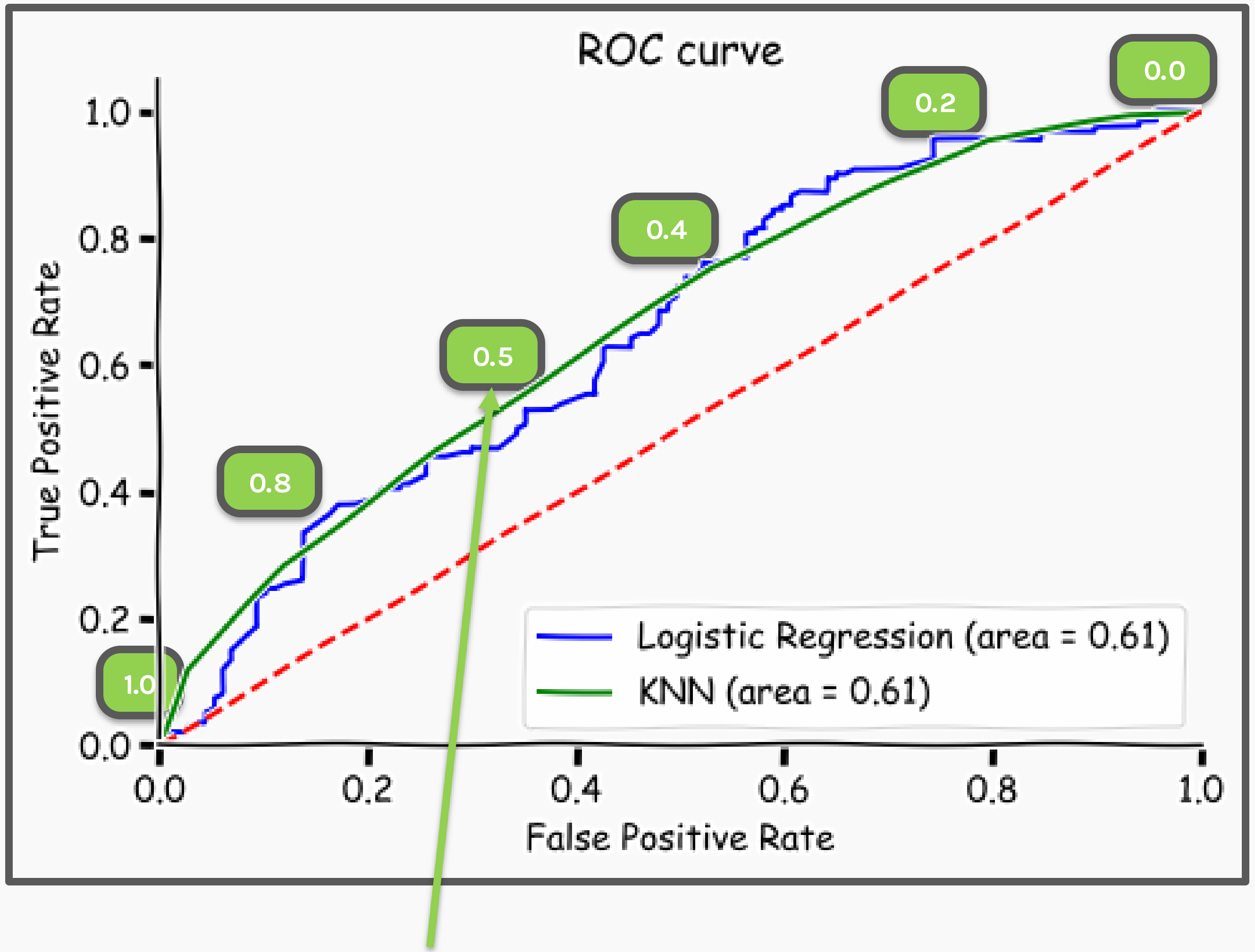
See next slide for an example.

Receiver Operating Characteristic curve (ROC)

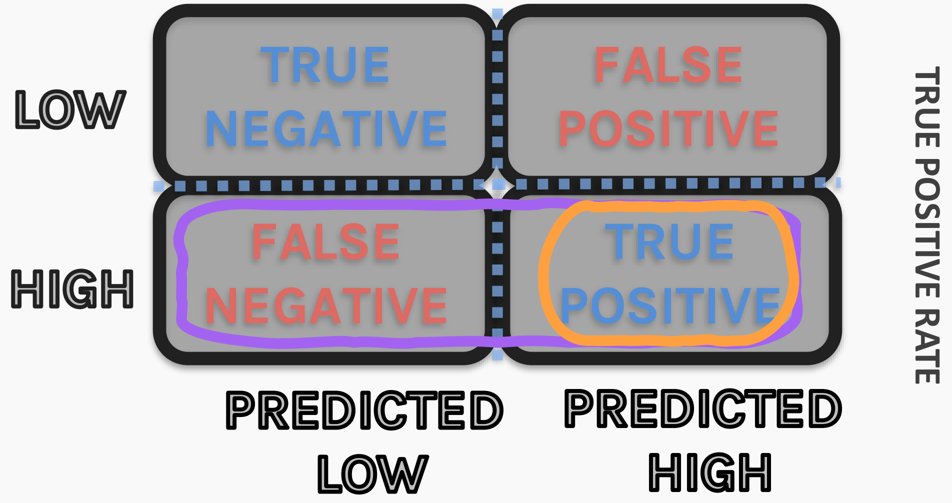
- The ROC curve was first developed by radar engineers during World War II for detecting enemy objects in battlefields.
- The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

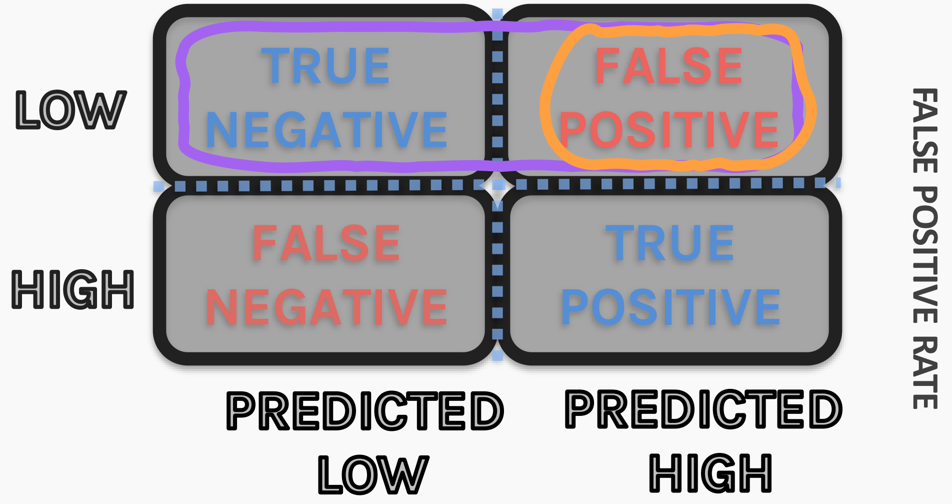


ROC curve for various thresholds

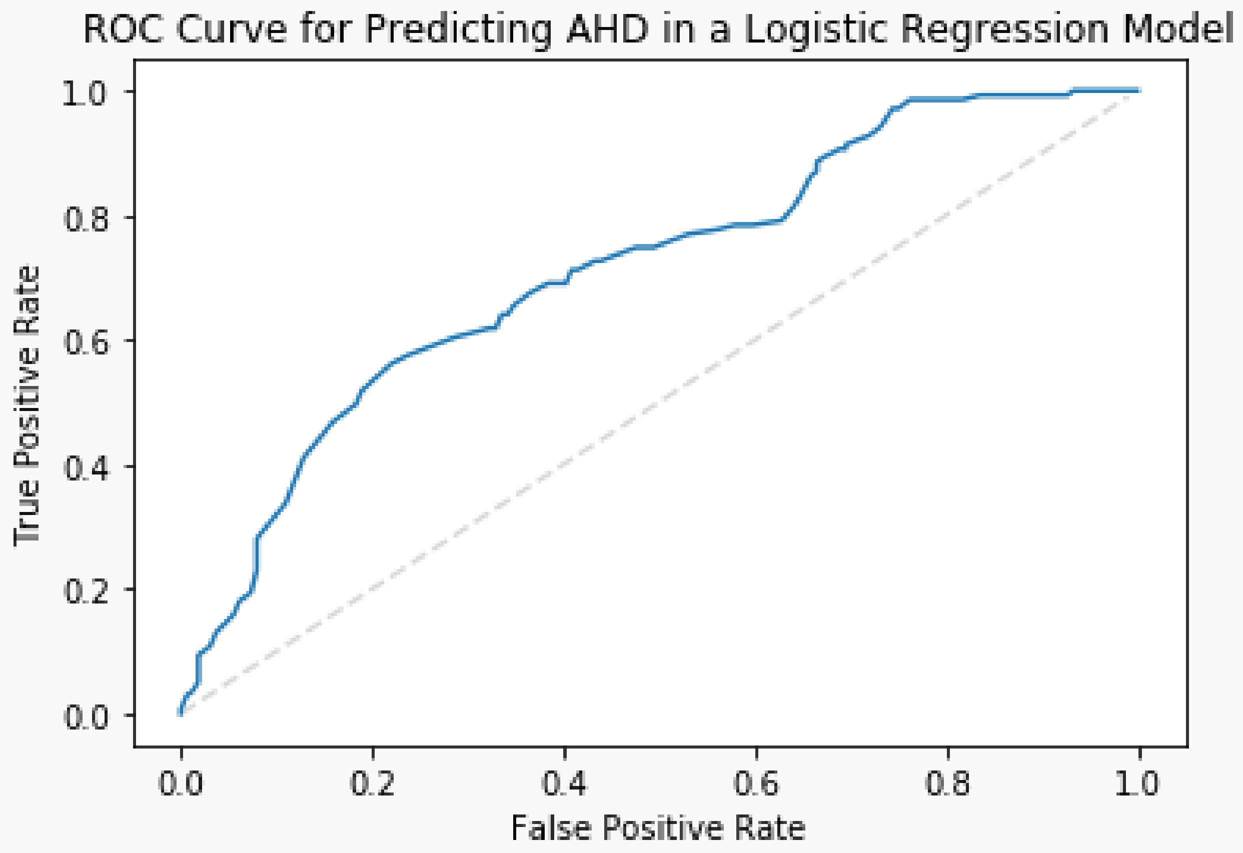


THRESHOLD





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.