Lecture #15: Bayesian Logistic Regression

aka STAT109A, AC209A, CSCIE-109A

CS109A Introduction to Data Science

Pavlos Protopapas, Kevin Rader and Chris Gumb

Outline

- Multiclass Logistic Regression
- Bayes Theorem and Misclassification Rates
- ROC Curves
- Bayes Review
- Beta Distribution
- Beta-Binomial Model
- Hierarchical Modeling: a preview

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).

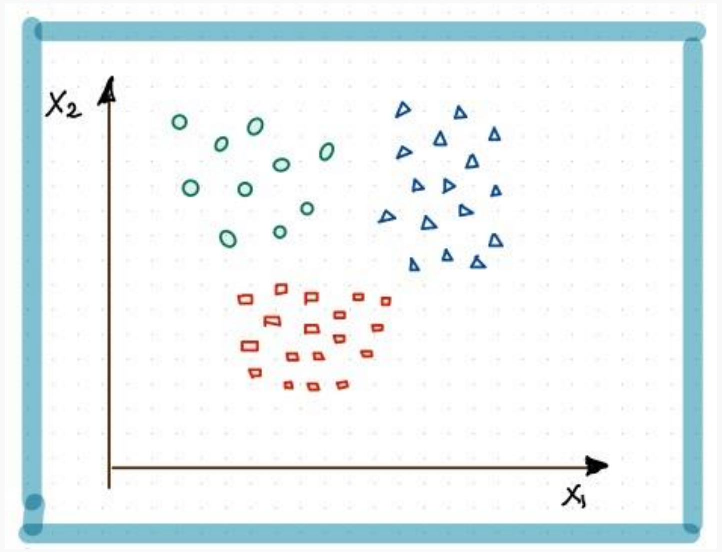
Ignoring interactions, how many parameters would need to be estimated?

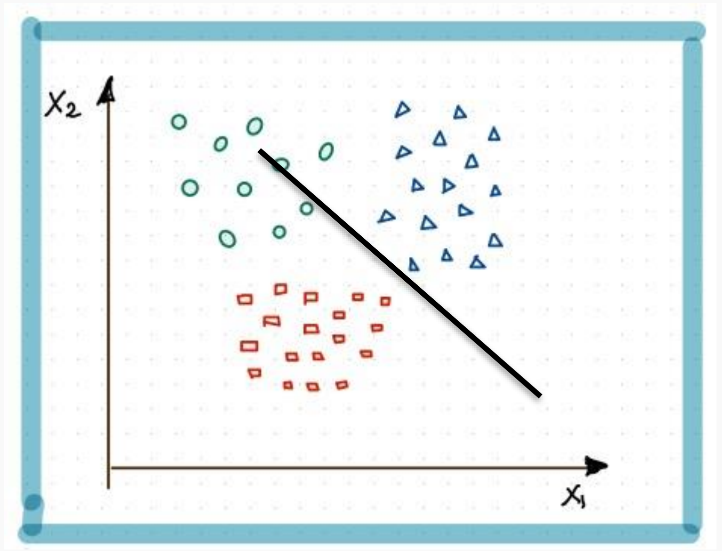
How could these models be used to estimate the probability of an individual falling in each concentration?

Multinomial Logistic

Classifying three classes:

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



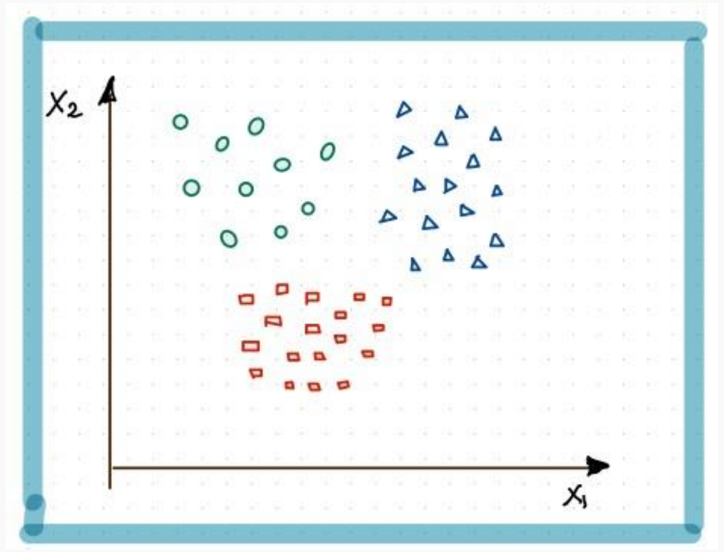


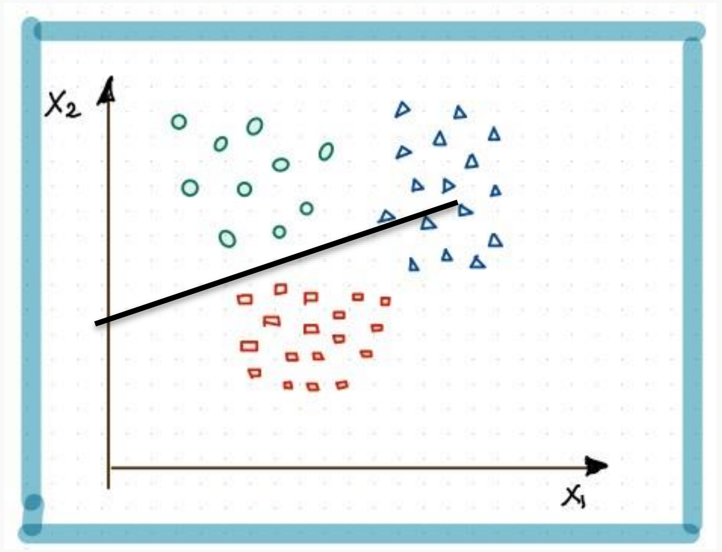
Blue vs Red

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

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

```python
mlogit = LogisticRegression(penalty=None, multi_class = 'multinomial')
mlogit.fit(nfl24[["Down","ToGo"]], nfl24["Play_Type"])


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

```
Estimated beta1:
 [[ 1.67327825  0.09564878]
 [-0.38944861  0.0369145 ]
 [-1.28382963 -0.13256327]]
Estimated beta0:
 [-6.2256748   1.86263164  4.36304316]
```

```python
pd.crosstab(nfl24["PlayType"],
            nfl24["Play_Type"])
```

Play_Type	0	1	2
PlayType			
CLOCK STOP	75	0	0
FIELD GOAL	1115	0	0
FUMBLES	119	0	0
PASS	0	18806	0
PUNT	2051	0	0
QB KNEEL	405	0	0
RUSH	0	0	13420
SACK	0	1401	0
SCRAMBLE	0	1133	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.

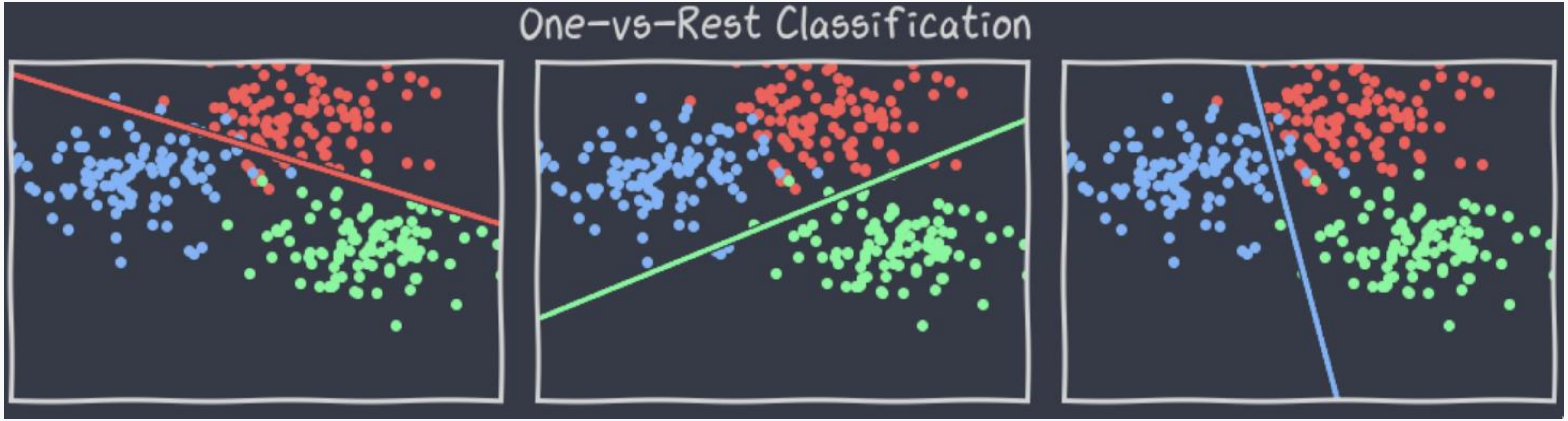
If there are 3 classes, then 3 separate logistic regressions are fit, where the probability of each category is predicted over the rest of the categories combined. So for the concentration example, 3 models would be fit:

- 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.

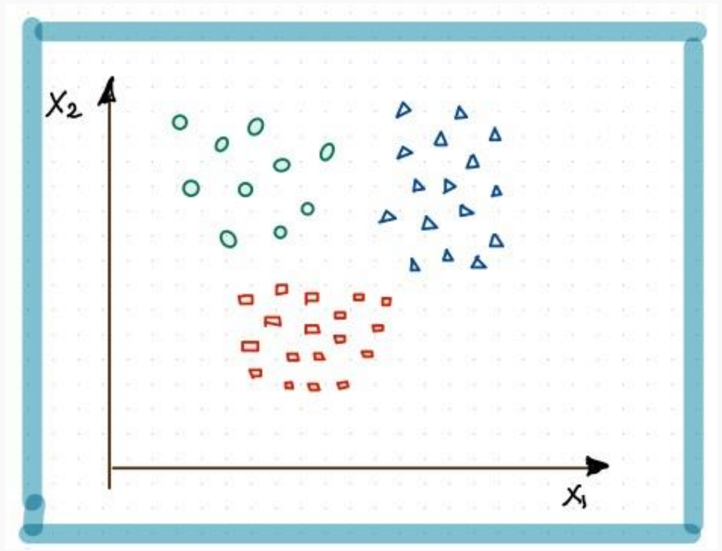
One vs. Rest (ovr)

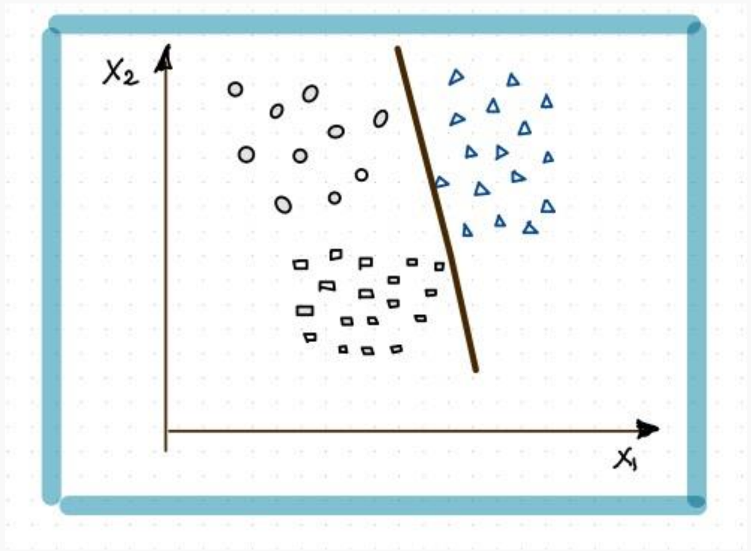
Classifying three classes,
Red, Blue and Green can be turn into three binary Logistic Regressions



One vs. Rest (ovr)

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



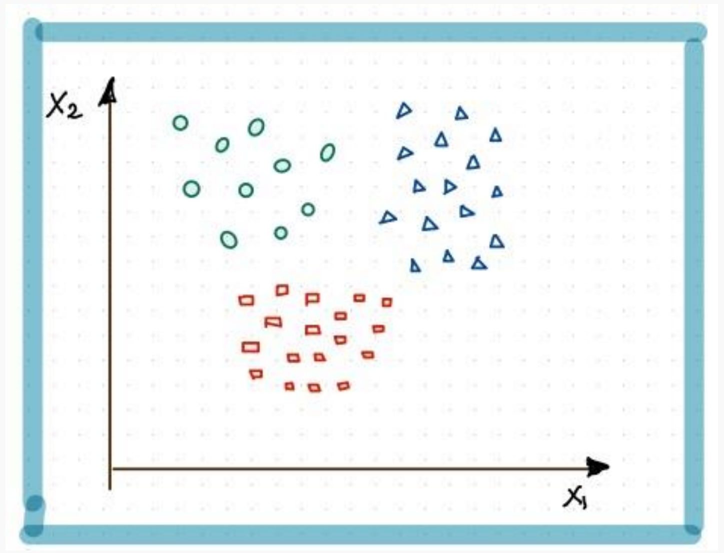


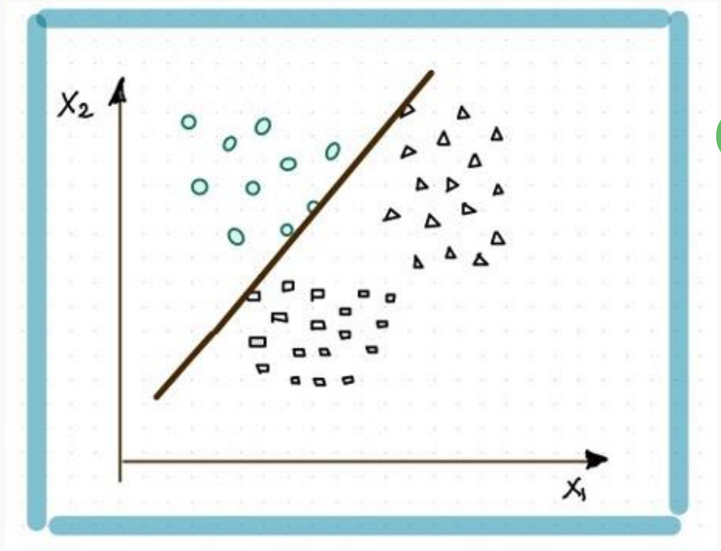
Blue vs others

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

One vs. Rest (ovr)

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



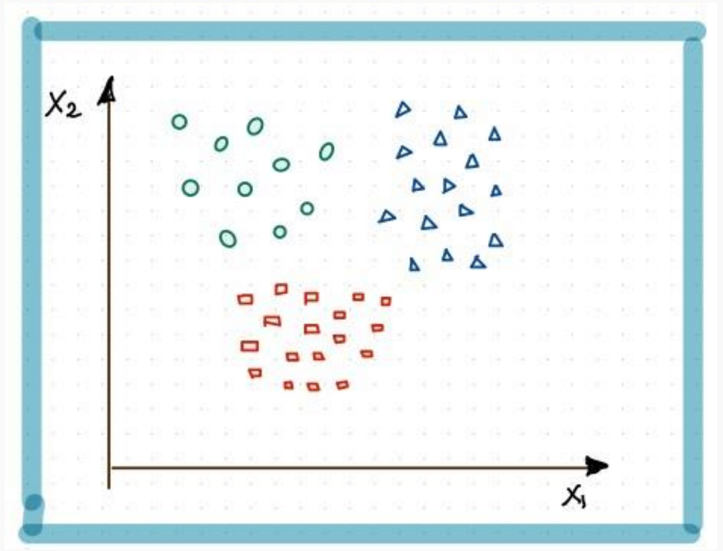


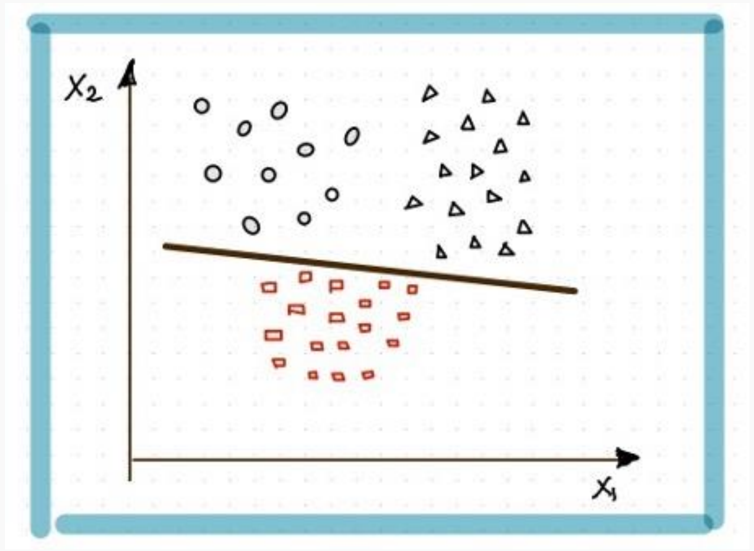
Green vs others

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

One vs. Rest (ovr)

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



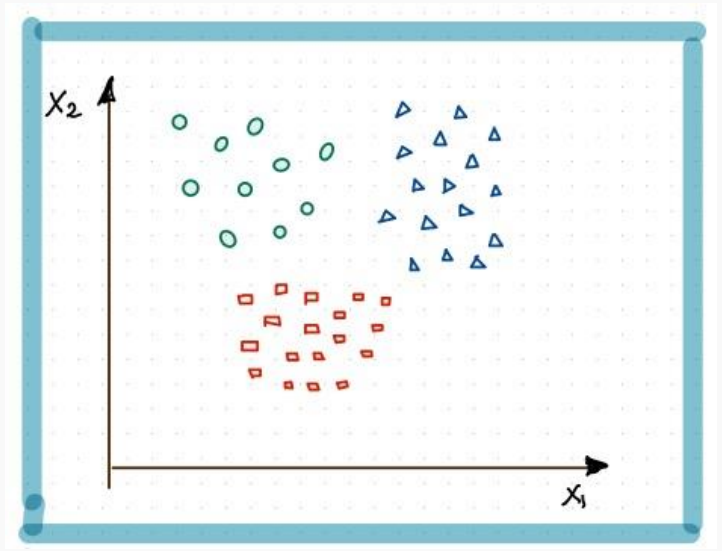


Red vs others

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

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?

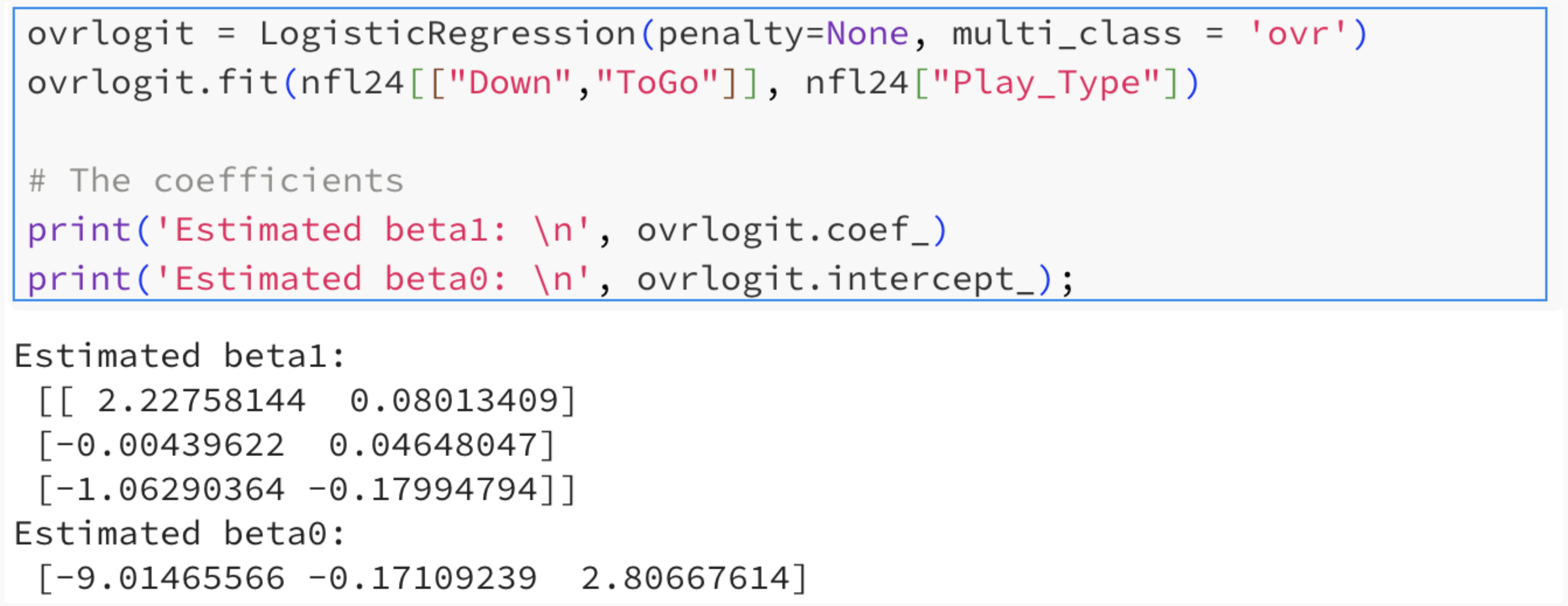
The ***softmax*** function is used. That is, the weights are just normalized for each predicted probability. AKA, predict the 3 class probabilities from each of the 3 models, and just rescale so they add up to 1.

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

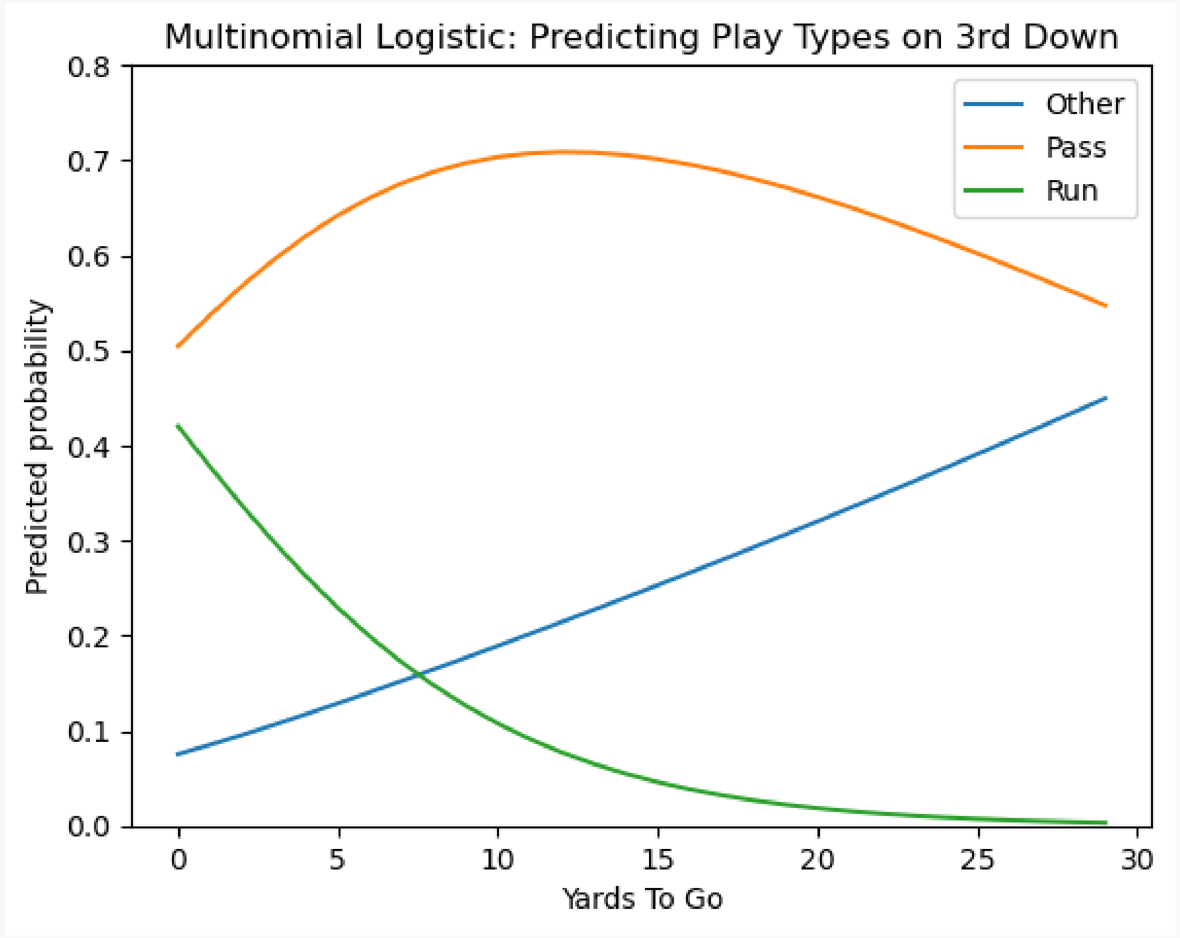


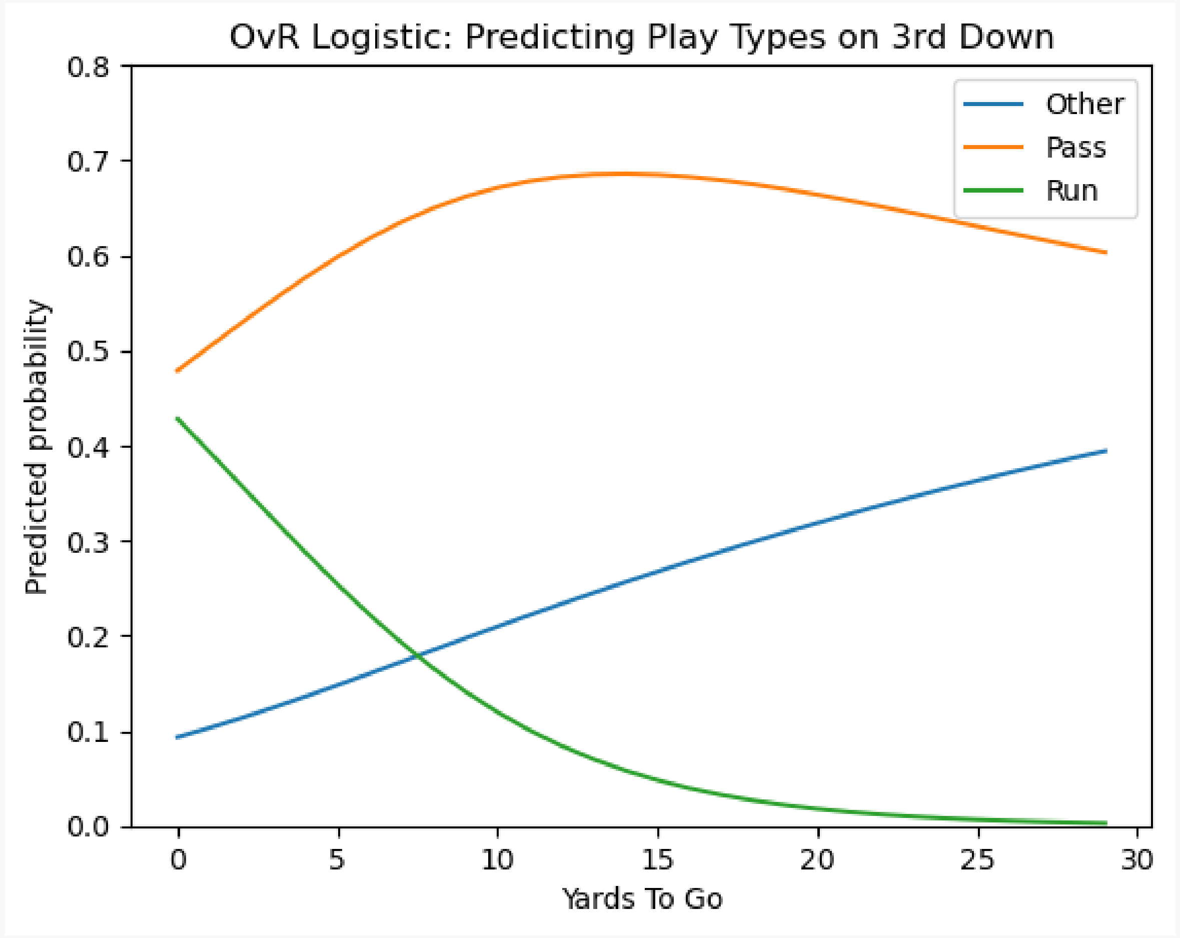
```
ovrlogit = LogisticRegression(penalty=None, multi_class = 'ovr')
ovrlogit.fit(nfl24[["Down","ToGo"]], nfl24["Play_Type"])

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

```
Estimated beta1:
 [[ 2.22758144  0.08013409]
 [-0.00439622  0.04648047]
 [-1.06290364 -0.17994794]]
Estimated beta0:
 [-9.01465566 -0.17109239  2.80667614]
```

- Phew! This one is as expected ☺
- sklearn's `LogisticRegression` model documentation. Note: fitting `ovr` with the above code will be deprecated soon and will require a different model class.

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



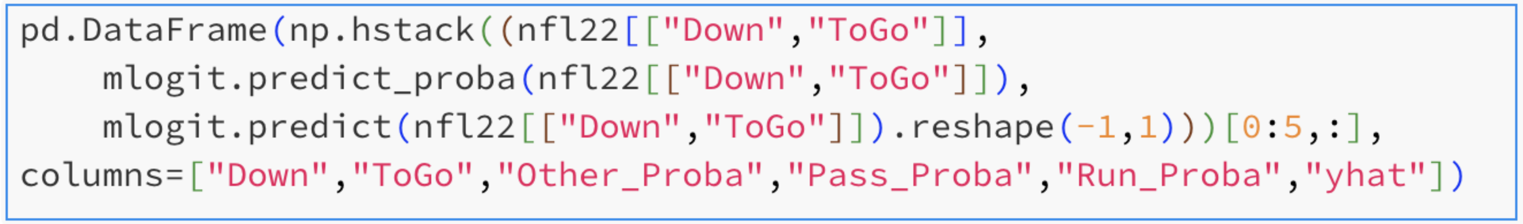


Classification for more than 2 Categories

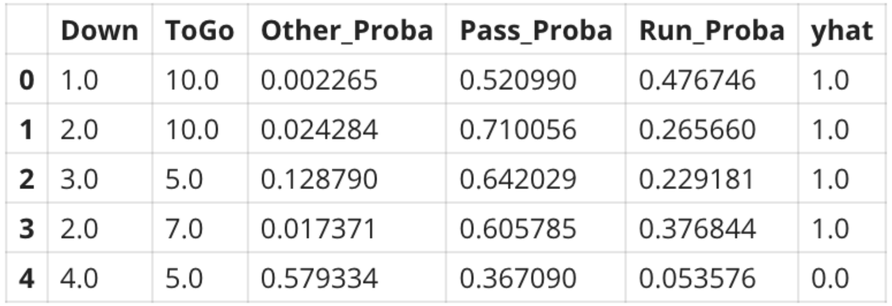
When there are more than 2 categories in the response variable, then there is no guarantee that $P(Y = k) \geq 0.5$ for any one category. So any classifier based on logistic regression (or other classification model) will instead have to select the group with **the largest estimated probability**.

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`!

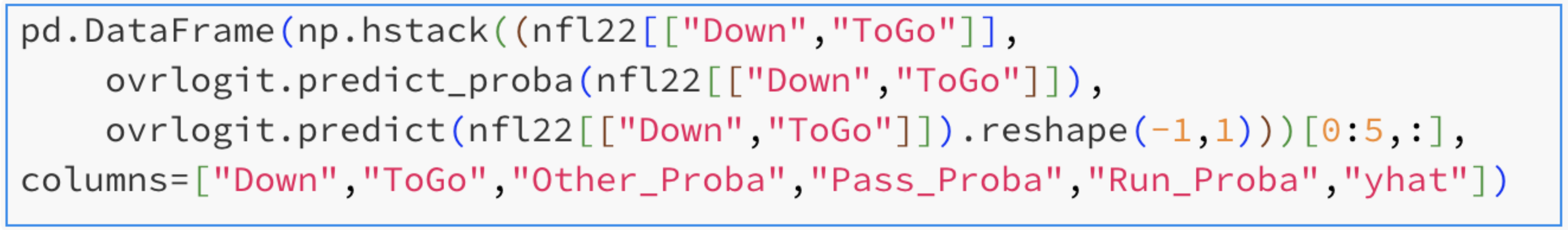
Prediction using Multiclass Logistic Regression



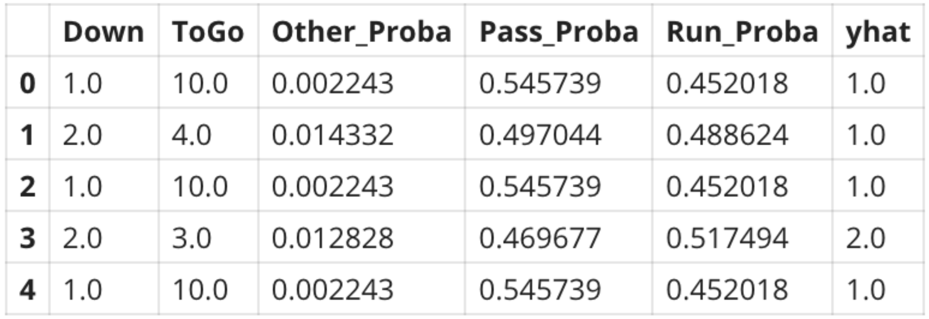
```
pd.DataFrame(np.hstack((nfl22[["Down","ToGo"]],
    mlogit.predict_proba(nfl22[["Down","ToGo"]]),
    mlogit.predict(nfl22[["Down","ToGo"]]).reshape(-1,1)))[0:5,:],
columns=["Down","ToGo","Other_Proba","Pass_Proba","Run_Proba","yhat"])
```



	Down	ToGo	Other_Proba	Pass_Proba	Run_Proba	yhat
0	1.0	10.0	0.002265	0.520990	0.476746	1.0
1	2.0	10.0	0.024284	0.710056	0.265660	1.0
2	3.0	5.0	0.128790	0.642029	0.229181	1.0
3	2.0	7.0	0.017371	0.605785	0.376844	1.0
4	4.0	5.0	0.579334	0.367090	0.053576	0.0

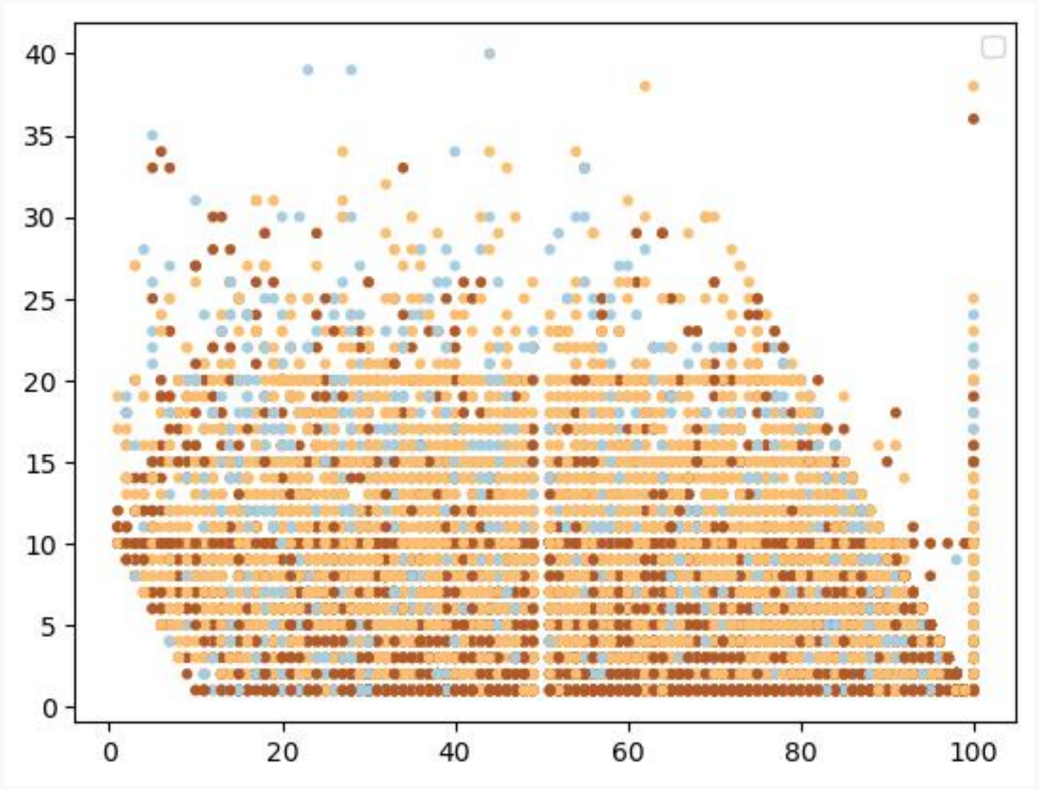


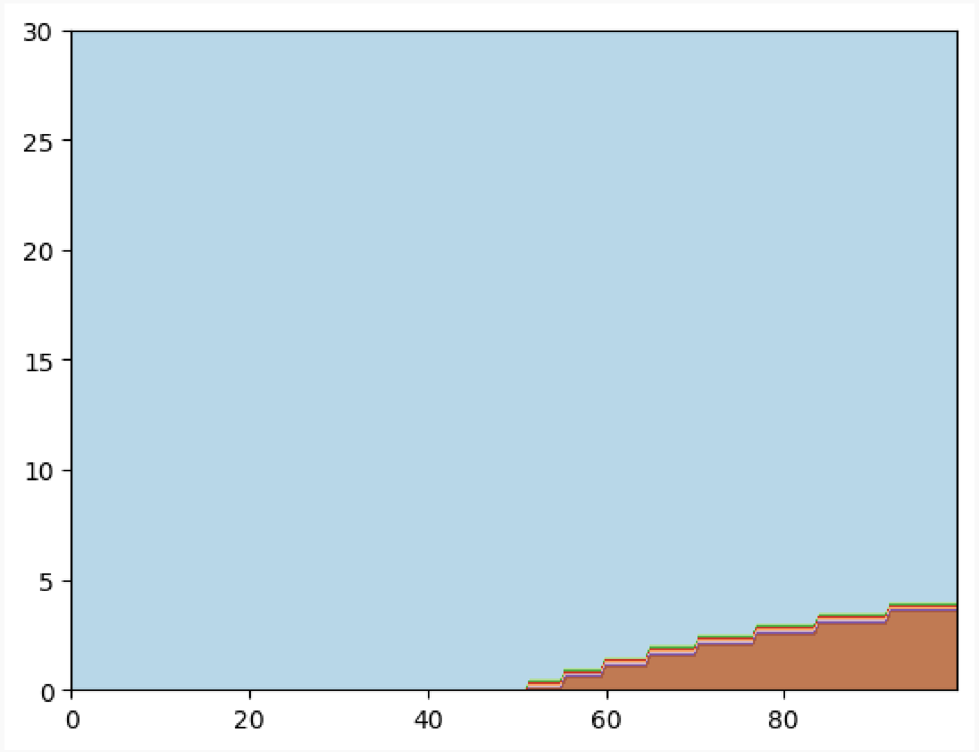
```
pd.DataFrame(np.hstack((nfl22[["Down","ToGo"]],
    ovrlogit.predict_proba(nfl22[["Down","ToGo"]]),
    ovrlogit.predict(nfl22[["Down","ToGo"]]).reshape(-1,1)))[0:5,:],
columns=["Down","ToGo","Other_Proba","Pass_Proba","Run_Proba","yhat"])
```



	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

- Multiclass Logistic Regression
- Bayes Theorem and Misclassification Rates
- ROC Curves
- Bayes Review
- Beta Distribution
- Beta-Binomial Model
- Hierarchical Modeling: a preview

Probability Review: Bayes' Theorem

What is conditional probability?

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

And using the fact that $P(A \text{ and } B) = P(A|B)P(B)$ 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:

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

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)	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.

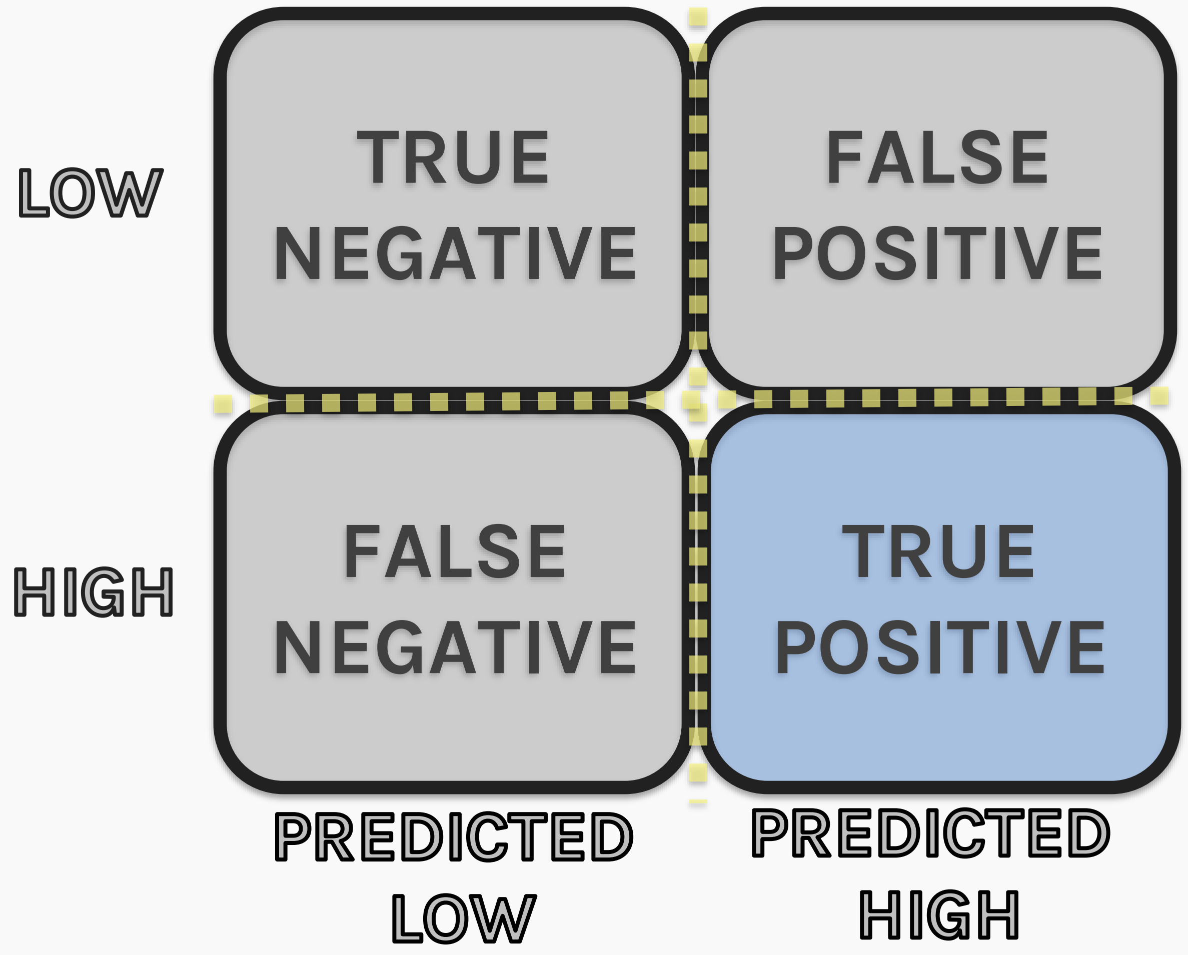
The 'Confusion' Matrix

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

The 'Confusion' Matrix

TRUE POSITIVE (TP)

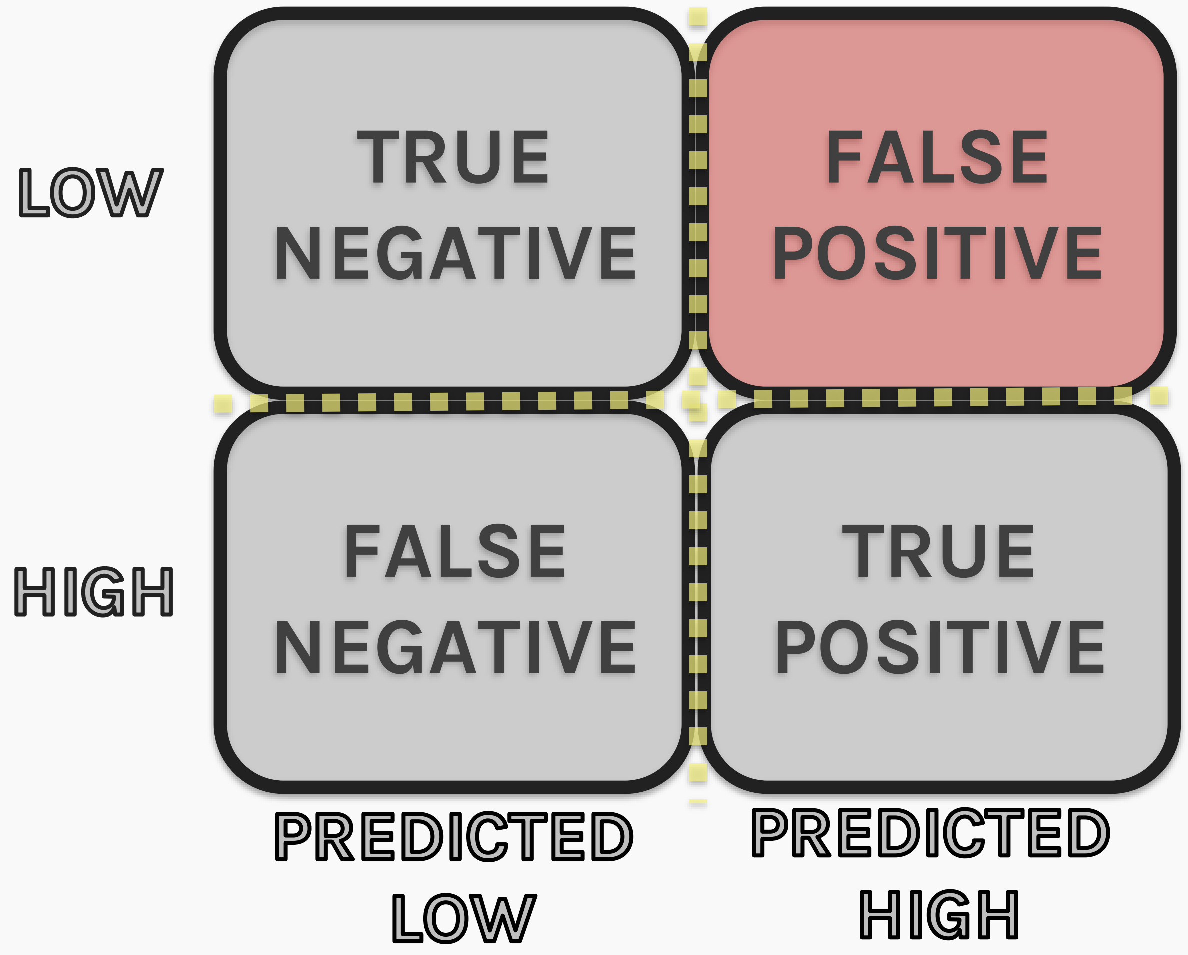
- Samples that are positive that the classifier predicts as positive are called True Positives.
- Example: a positive Covid test result would be a TRUE POSITIVE if you actually have Covid.



The 'Confusion' Matrix

FALSE POSITIVE (FP)

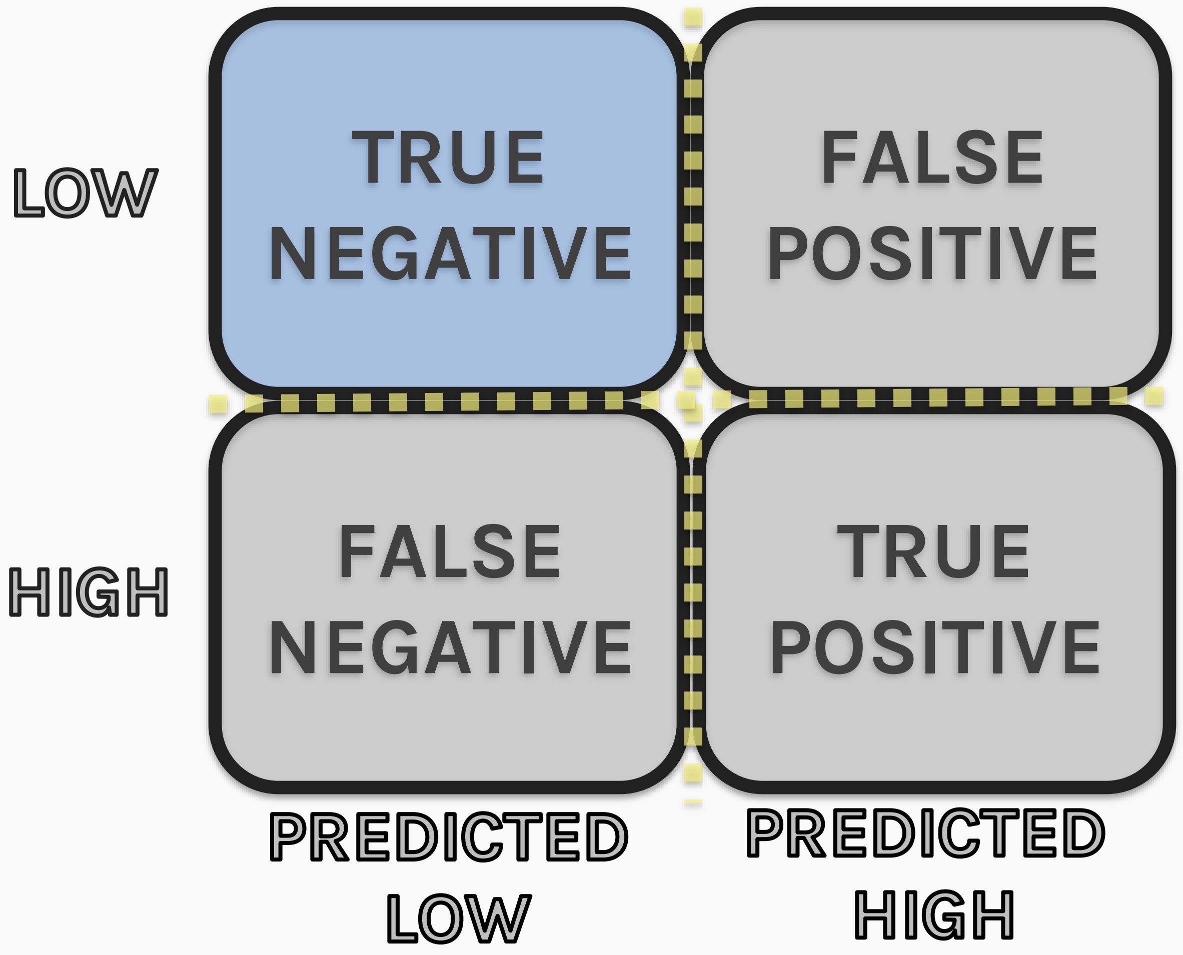
- Samples that are negative that the classifier predicts as positive are called False Positives.
- Example: a positive Covid test result would be a FALSE POSITIVE if you actually don't have Covid.



The 'Confusion' Matrix

TRUE NEGATIVE (TN)

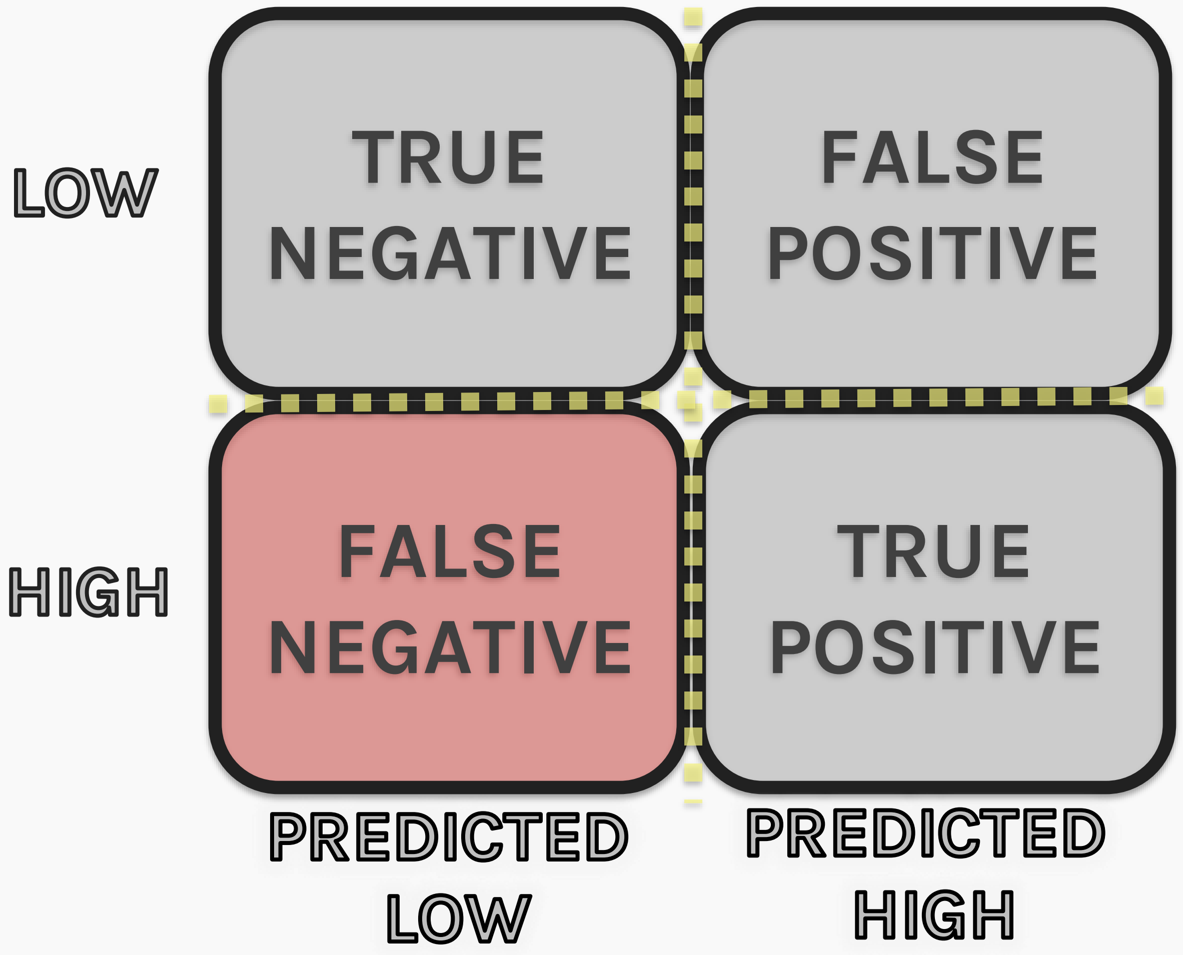
- Samples that are negative that the classifier predicts as negative are called True Negatives.
- Example: a negative Covid test result would be a TRUE NEGATIVE if you actually don't have Covid.



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).

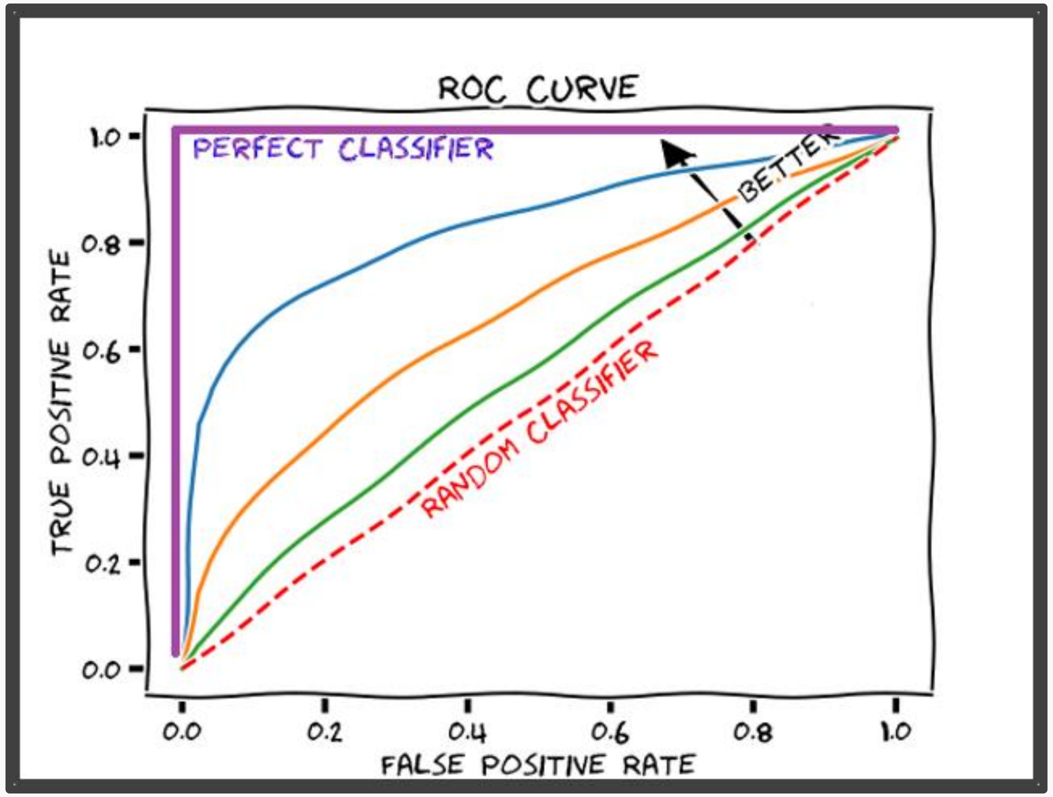
The vertical axis displays the true positive predictive value and the horizontal axis depicts the true negative predictive value.

What is the shape of an ideal ROC curve?

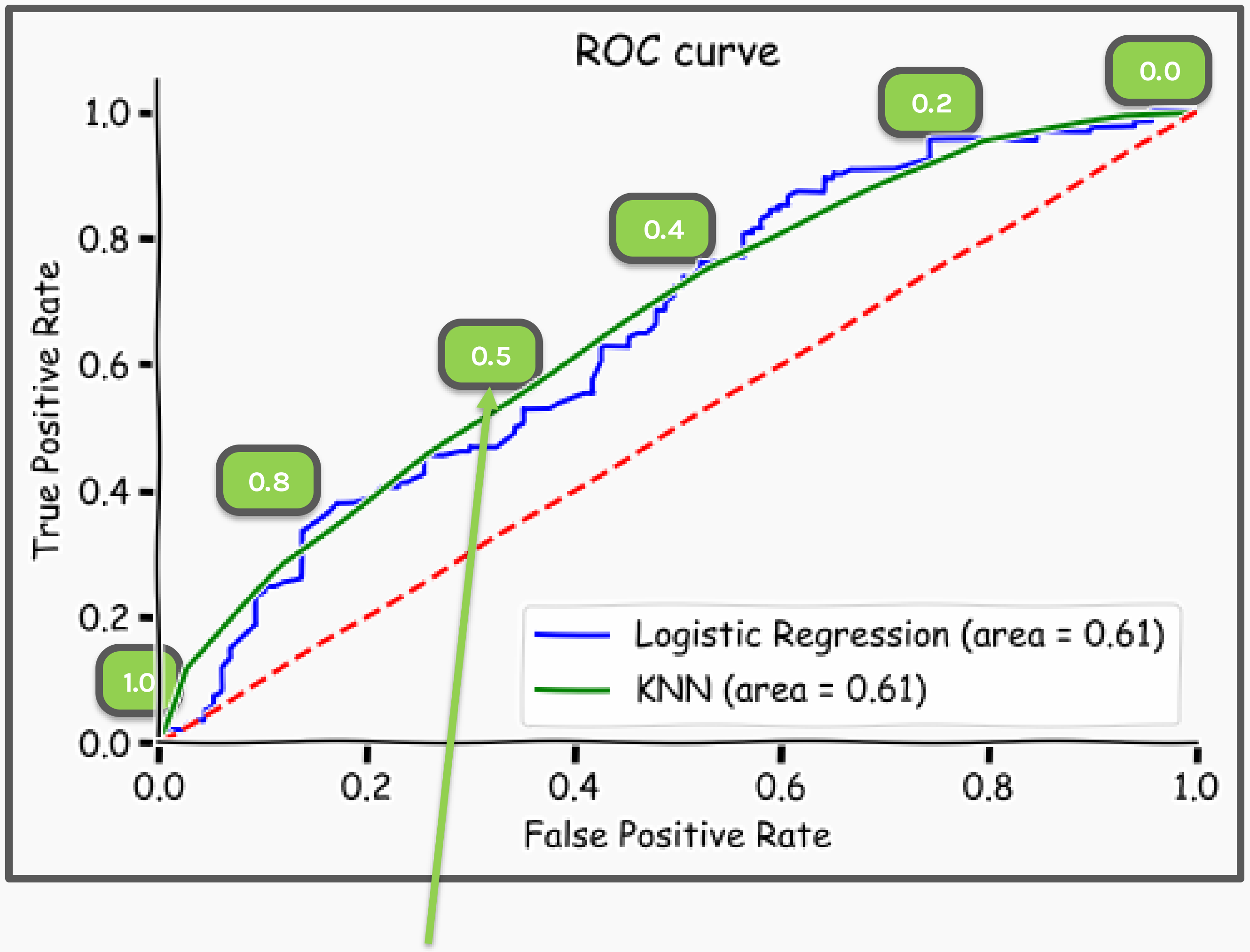
See next slide for an example.

Receiver Operating Characteristic curve (ROC)

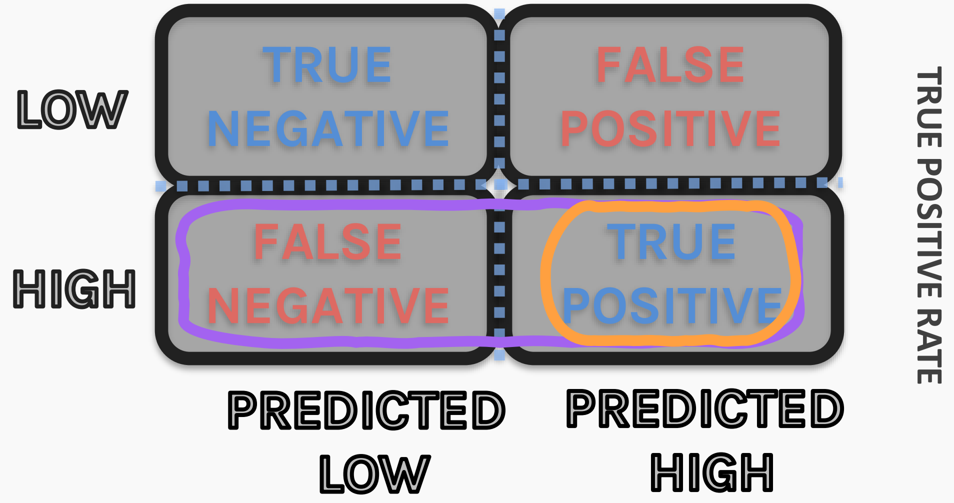
- The ROC curve was first developed by radar engineers during World War II for detecting enemy objects in battlefields.
- The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

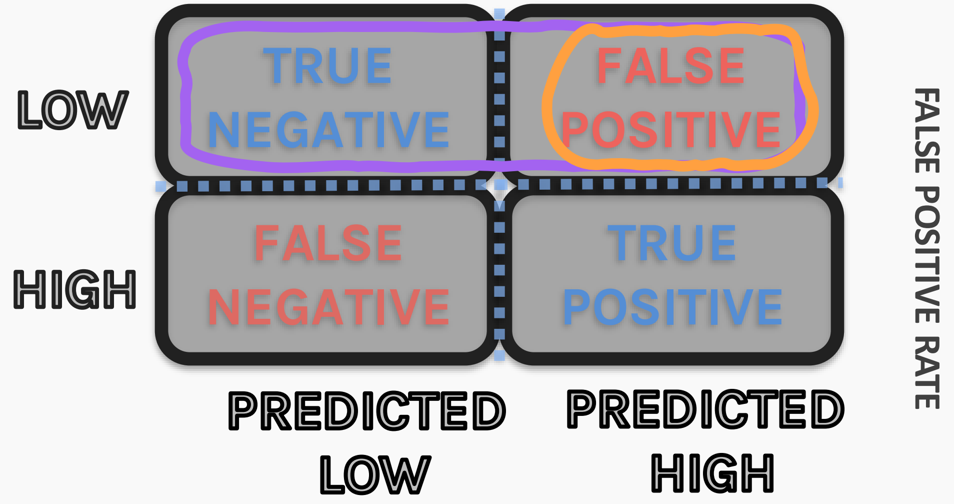


ROC curve for various thresholds

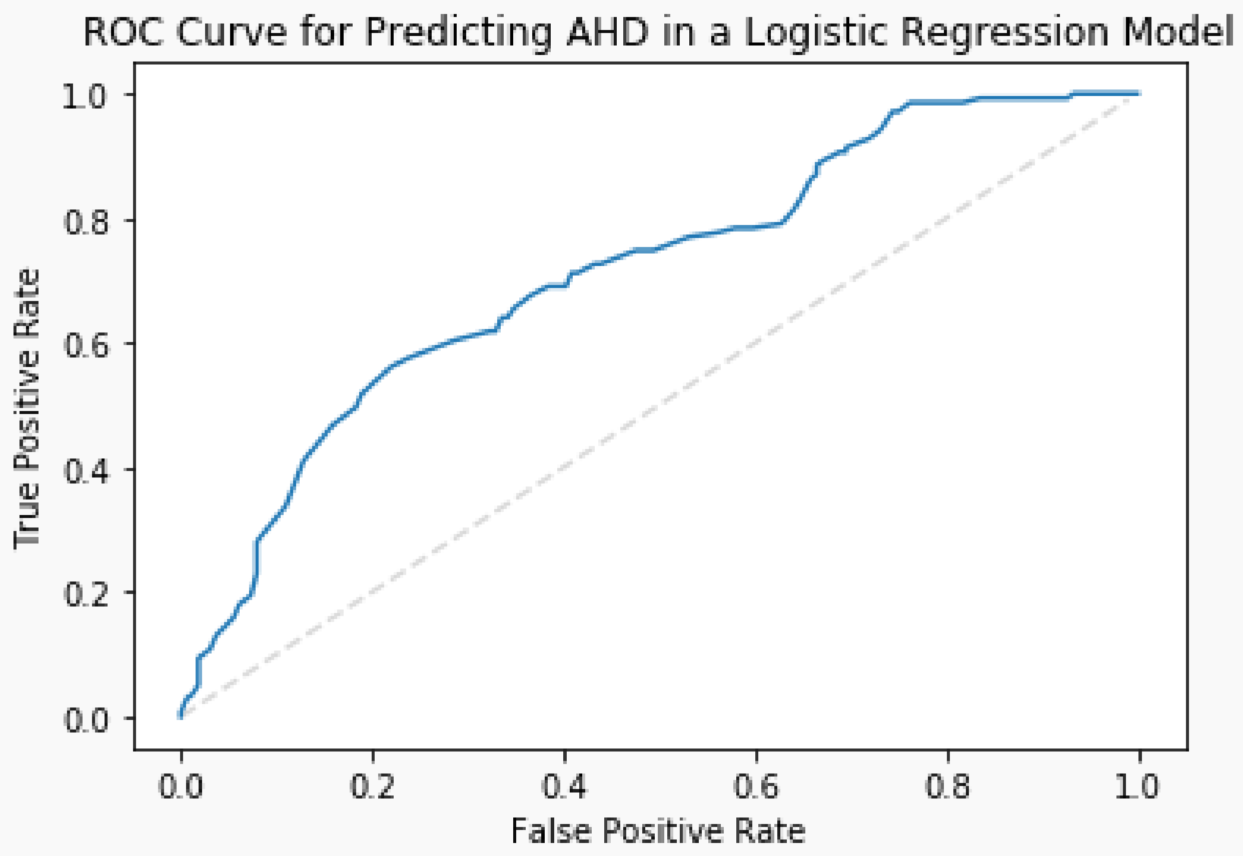


THRESHOLD





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.

Outline

- Multiclass Logistic Regression
- Bayes Theorem and Misclassification Rates
- ROC Curves
- **Bayes Review**
- Beta Distribution
- Beta-Binomial Model
- Hierarchical Modeling: a preview

Bayes Rule, for distributions!

- We just saw the simplest form of Bayes' Rule:

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

- What is Bayes' Rule effectively doing?

- How would this be useful for statistical inference?
  *Think: parameters (θ) and data (X).

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)}$$

Bayes Rule/Inference, for continuous RVs

- This can be rewritten for a set of parameters, θ, treating it as a continuous random variable, in terms of PDFs:

$$f(\theta|X) = \frac{f(X|\theta)f(\theta)}{f(X)}$$

- Let's break this down:

  X : the vector (or matrix) of data: $X_1, X_2, \ldots, X_n$

  θ : the vector of parameters (or just a scalar).

$f(\mathbf{X}|\theta)$: the likelihood of X_i's

$f(\mathbf{X})$: the marginal pdf of X_i's (just a normalizing constant)

$f(\theta)$: the *prior distribution* of θ

$f(\theta|\mathbf{X})$: the *posterior distribution* of θ

Bayesian Inference: from prior to posterior

- The prior distribution, $f(\theta)$, often is based on a known distribution with it's own set of parameters. These are called *hyperparameters*.
- The marginal PDF of X is the distribution of X ignoring θ. How do you solve for a marginal PDF based on a joint distribution?

$$f(X) = \int_{\theta} f(x,\theta)d\theta = \int_{\theta} f(X|\theta)f(\theta)d\theta$$

- By definition, this marginal PDF of X will not involve θ. Thus, it can be though of as a multiplicative normalizing constant with respect to θ.
- So we can write the posterior dist. as proportional to:

$$f(\theta|X) = \frac{f(X|\theta)f(\theta)}{f(X)} \propto f(X|\theta)f(\theta)$$

Outline

- Multiclass Logistic Regression
- Bayes Theorem and Misclassification Rates
- ROC Curves
- Bayes Review
- Beta Distribution
- Beta-Binomial Model
- Hierarchical Modeling: a preview

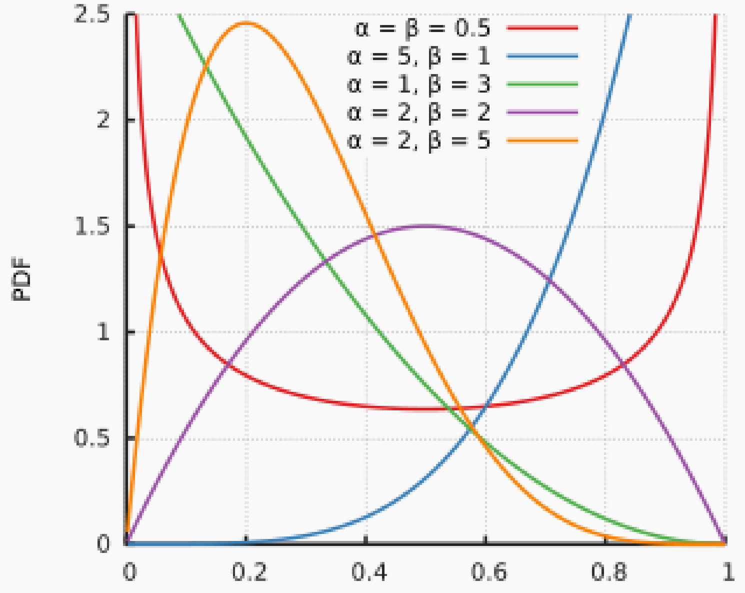
Beta Distribution

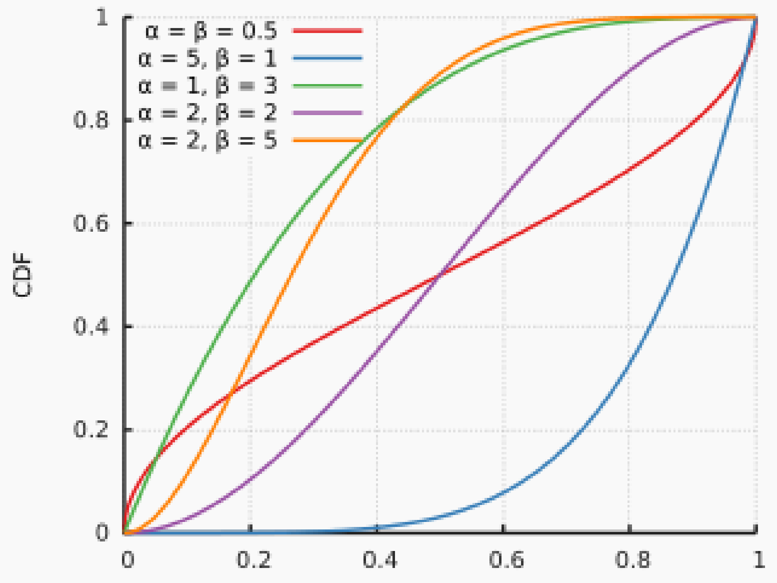
- Let's define a novel distribution for us, the Beta distribution!
- Let $X \sim Beta(\alpha, \beta)$. Then the PDF is defined as ($x \in (0,1)$):

$$f(x|\alpha, \beta) = c \cdot x^{\alpha-1}(1-x)^{\beta-1}$$

where the normalizing function is $c = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}$.

- Let's visualize the distribution





Beta Distribution

- Let $X \sim Beta(\alpha, \beta)$:

$$f(x|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \cdot x^{\alpha-1}(1-x)^{\beta-1}$$

- What is $\Gamma(\alpha)$?
    - If α is an integer, then $\Gamma(\alpha) = (\alpha-1)!$
- What is the mean of X?

$$\boxed{E(X) = \frac{\alpha}{\alpha+\beta}}$$

- What is the variance of X?

$$\boxed{\mathrm{Var}(X) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}}$$

Beta Distribution: the mean and variance derivations

Let's calculate the mean of X:

$$\mu = E(X) = \int xf(x)dx = \int_0^1 x\,\tfrac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha-1}(1-x)^{\beta-1}dx$$

$$= \int_0^1 \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{(\alpha+1)-1}(1-x)^{\beta-1}dx$$

$$= \left(\frac{\Gamma(\alpha+1)}{\Gamma(\alpha)}\right)\left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha+\beta+1)}\right)\int_0^1 \frac{\Gamma(\alpha+\beta+1)}{\Gamma(\alpha+1)\Gamma(\beta)}x^{(\alpha+1)-1}(1-x)^{\beta-1}dx$$

$$= \left(\frac{\Gamma(\alpha+1)}{\Gamma(\alpha)}\right)\left(\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha+\beta+1)}\right)\cdot 1 = \alpha\cdot\frac{1}{\alpha+\beta} = \boxed{\frac{\alpha}{\alpha+\beta}}$$

To calculate the variance, we can take a similar process for $E(X^2)$ and use the fact that $\mathrm{Var}(X) = E(X^2) - \mu^2$.

Beta Distribution: so why do we care?

- Let's take a closer look at the PDF for a beta:

$$f(x|\alpha, \beta) = c \cdot x^{\alpha-1}(1-x)^{\beta-1}$$

- Why is this a useful *functional form*? Hint: replace x with p.
  - It is the same form as the Bernoulli!
- Thus, the Beta distribution is the conjugate prior for the parameter p when our data are $Y \sim Bern(p)$ (like in a logistic regression model).
- And we can tweak our choices of α and β in order to control where the mean and variance of the prior.

Outline

- Multiclass Logistic Regression
- Bayes Theorem and Misclassification Rates
- ROC Curves
- Bayes Review
- Beta Distribution
- **Beta-Binomial Model**
- Hierarchical Modeling: a preview

Bayesian Beta-Binomial Model

- Let $X_1, X_2, \dots, X_n \sim \text{Bern}(p)$.
- Let's put a prior on $p \sim Beta(a_0, b_0)$.
- What are the parameter(s) and the hyperparameters?
- Write down the prior:

$$f(p|a_0, b_0) = \left(\frac{\Gamma(a_o)\Gamma(b_0)}{\Gamma(a_0 + b_0 + 1)} p^{a_0 - 1}(1-p)^{b_0 - 1}\right)$$

- Write down the likelihood:

$$f(X_1, \dots, X_n|p) = \prod_{i=1}^{n} (p^{x_i}(1-p)^{1-x_i}) = p^{\Sigma x_i}(1-p)^{n-\Sigma x_i}$$

- Let's ignore the normalizing constant and look at the *functional form* of the posterior (what it is proportional to).

Bayesian Beta-Binomial Model

- Thus the posterior distribution is proportional to:

$$f(p|X) \propto f(X|p) \cdot f(p)$$

$$= \left(\frac{\Gamma(a_o)\Gamma(b_0)}{\Gamma(a_0 + b_0 + 1)} p^{a_0 - 1}(1-p)^{b_0 - 1} \right) \cdot \left(p^{\sum x_i}(1-p)^{n - \sum x_i} \right)$$

$$\propto p^{(a_0 + \sum x_i) - 1}(1-p)^{(b_0 + n - \sum x_i) - 1}$$

- Since the posterior's RV is *p*, any multiplicative constant not involving it can be "absorbed" by the normalizing constant. So that's why the gamma terms drop out, and we can just write the posterior as proportional to just the terms involving *p*.
- What distribution (of *p*) does this have the general form of (without the normalizing constant)?

Beta-Binomial Model: Posterior Result

- So the posterior distribution is:

$$(p|Y) \sim Beta(\alpha + \sum y_i, \beta + (n - \sum y_i))$$

- So what?
- The posterior distribution for the mean of a normal distribution, given the data, only depends on the sample data in terms of the sample mean. The posterior of μ is normally dist. (if we start with a prior that is normally dist.).
- What is the posterior mean estimator (the mean of this distribution)?
- The posterior mean of μ is a weighted average of the prior mean, μ_0, and n-times the sample mean. So what happens to the effect of the prior on the posterior (and the estimator) as n increases?
    - The variance of the posterior decreases as n increases.

Hierarchical Modeling

- How do we extend this Beta-Binomial model to Logistic Regression?
- What are the unknown parameters in this model?

- What is the likelihood for this model? How do they link to the response?

- Recall the Logistic Regression likelihood:

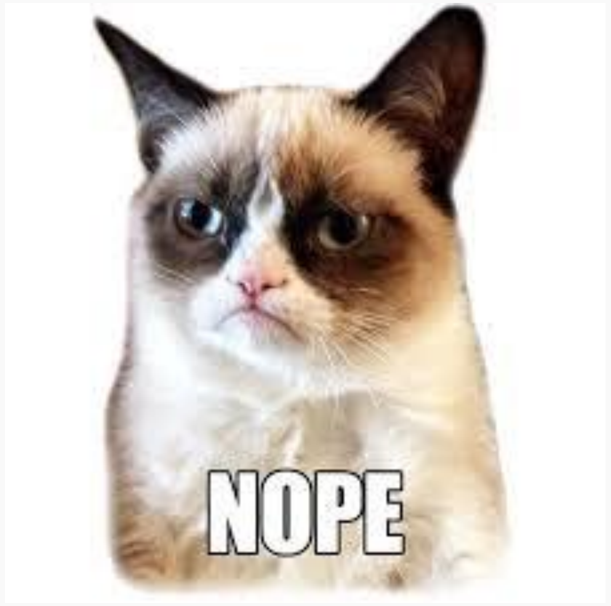
$$\left(Y_i|\vec{\beta},\vec{X}_i\right) \sim Bern(p_i)$$

$$p_i = \frac{e^{\beta_0+\beta_1 x_{1,i}+\cdots+\beta_p x_{p,i}}}{1+e^{\beta_0+\beta_1 x_{1,i}+\cdots+\beta_p x_{p,i}}}$$

- Recall: we often write this as $\ln\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_p x_{p,i}$ for interpretive reasons.

Beta-Binomial Model: Logistic Regression Result

- Can we simply put Beta priors on our unknown parameters?



- Conventionally, we assume $\beta \sim N(\mu_0 = 0, \sigma^2)$ priors. This shrinks the coefficients towards zero (think Ridge).
- We'll come back to this on Wednesday when we learn how to sample from posterior distributions (think MCMC)!

Outline

- Multiclass Logistic Regression
- Bayes Theorem and Misclassification Rates
- ROC Curves
- Bayes Review
- Beta Distribution
- Beta-Binomial Model
- **Hierarchical Modeling: a preview**

Hierarchical Modeling: the idea

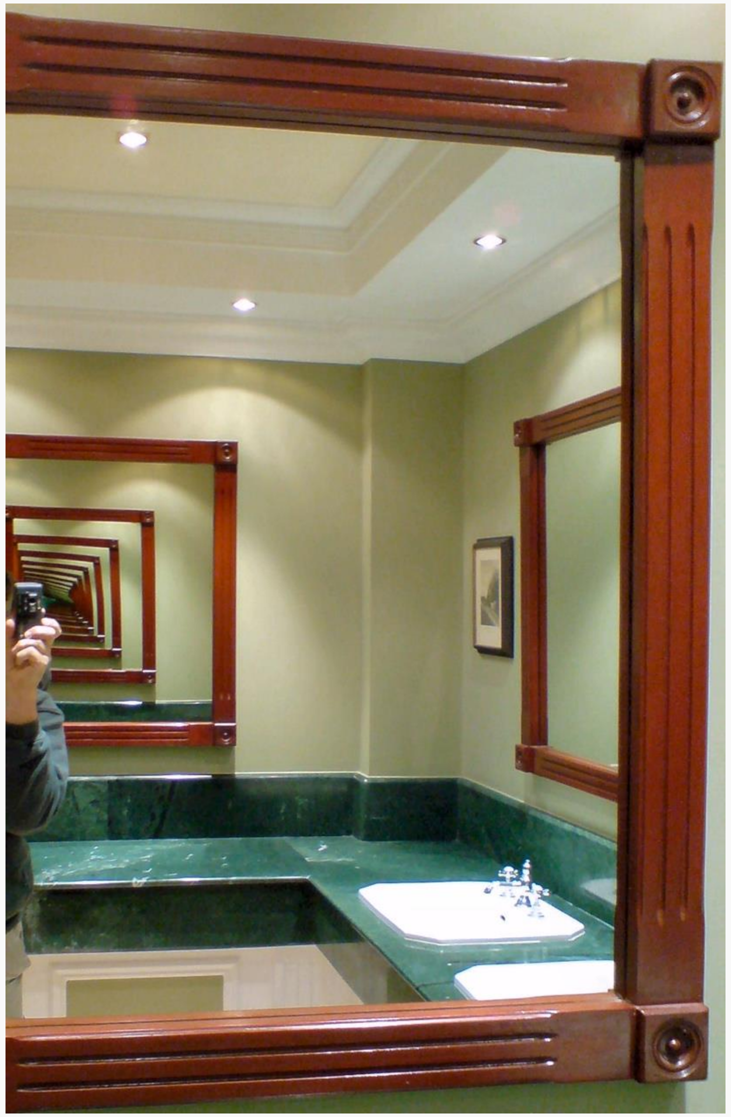
- Recall the Bayesian Beta-Binomial model:

$$(Y_i|p_i) \sim Bern(p_i)$$

$$p_i \sim Beta(a_0, b_0)$$

- What do a_0 and b_0 represent? What do we call them?
- How can we complicate things even more?
  - Let's put priors on the hyperparameters!
- This is called a hyperprior distribution.
  - What would be reasonable hyperprior distributions?
    - Hint: think support.
- What's stopping us from putting priors on our hyperprior's parameters?
  - Shall we call this a hyper-hyperprior distribution?

Hierarchical Modeling: the Bayesian perspective



Hierarchical Modeling: the use case

- So why would we bother with putting hyperpriors on our priors?
- 2 main ideas:
  1. We may be uncertain about which hyperparameter(s) to use in our prior.
  2. More commonly: the data's structure. What if we measure data at several *levels*?
- Examples:
  - **Governments:** data could be measured on individuals, that reside within counties (that have their own measurements), which reside within states, that reside withing regions, etc.
  - **Education:** students within schools within districts within states.
  - **Medicine:** patients from doctors within hospitals
  - **Biology:** cells from tissues from organs from individuals
  - And so many more…

Hierarchical Modeling: examples

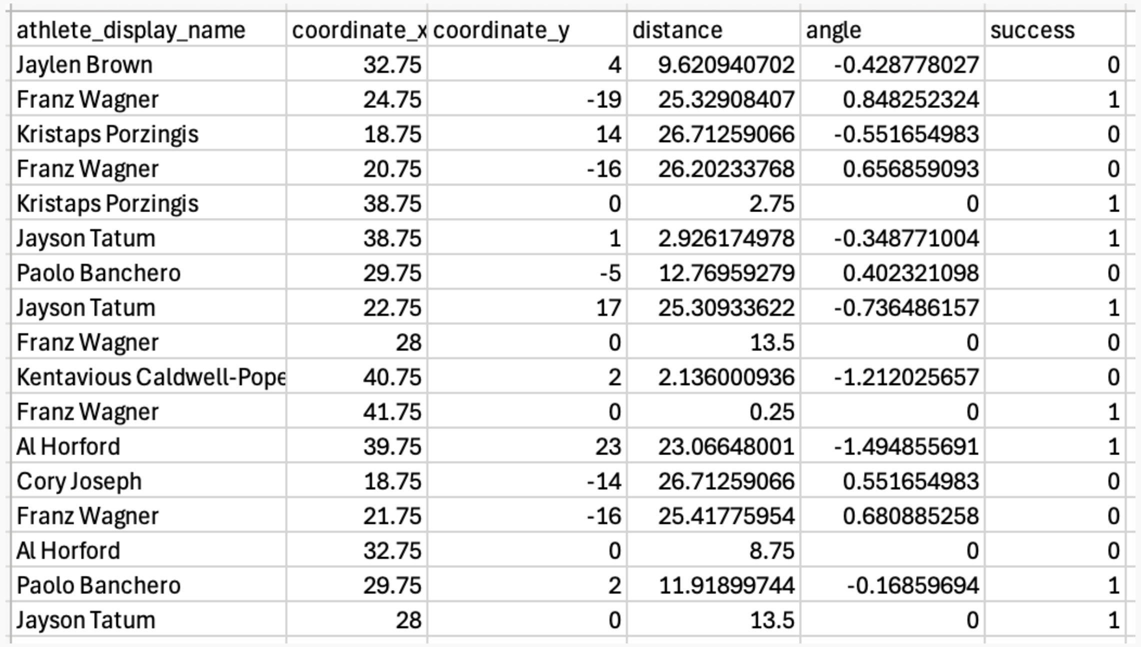
- So why would we bother with putting hyperpriors on our priors?
- 2 main ideas:
  1. We may be uncertain about which hyperparameter(s) to use in our prior.
  2. More commonly: the data's structure. What if we measure data at several *levels*?
- Examples:
  - **Governments:** data could be measured on individuals, that reside within counties (that have their own measurements), which reside within states, that reside withing regions, etc.
  - **Education:** students within schools within districts within states.
  - **Medicine:** patients from doctors within hospitals
  - **Biology:** cells from tissues from organs from individuals
  - And in sports (one last time)…

Hierarchical Modeling: an example

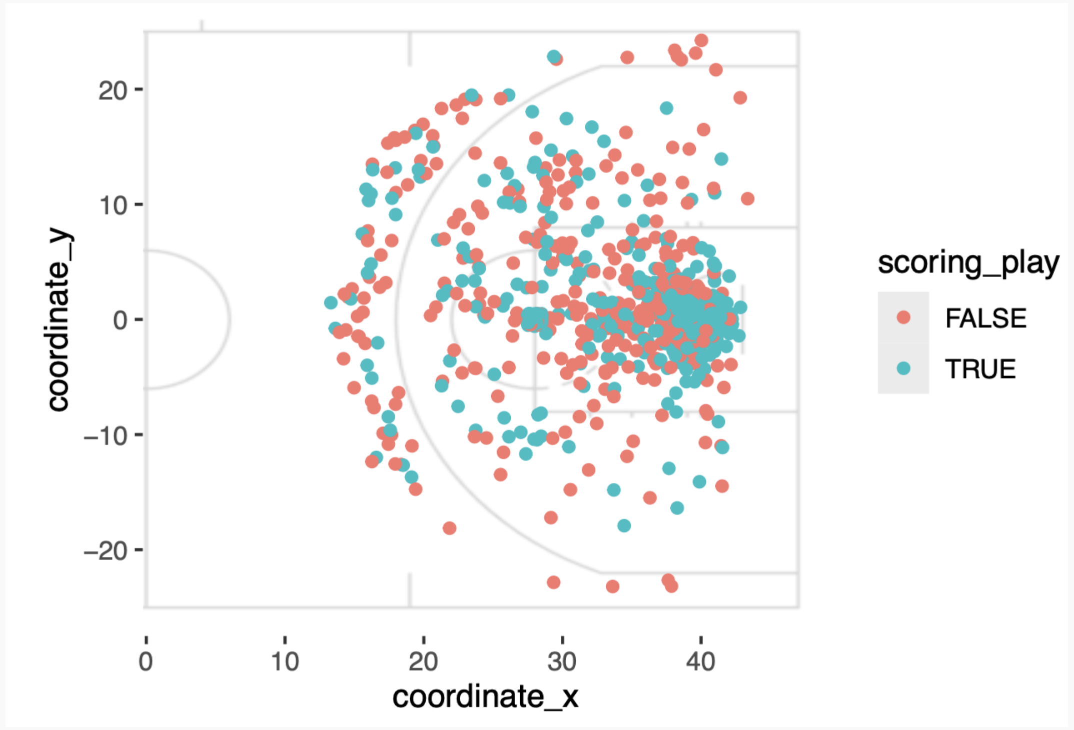
- Who are the best [offensive] players in the NBA?

- We’d like to model the chances of a field goal attempt being a success given the location (distance and possibly angle) and the player who is taking the shot.
- What will the data look like? Why is a hierarchical model a reasonable one?

Basketball Shot data



athlete_display_name	coordinate_x	coordinate_y	distance	angle	success
Jaylen Brown	32.75	4	9.620940702	-0.428778027	0
Franz Wagner	24.75	-19	25.32908407	0.848252324	1
Kristaps Porzingis	18.75	14	26.71259066	-0.551654983	0
Franz Wagner	20.75	-16	26.20233768	0.656859093	0
Kristaps Porzingis	38.75	0	2.75	0	1
Jayson Tatum	38.75	1	2.926174978	-0.348771004	1
Paolo Banchero	29.75	-5	12.76959279	0.402321098	0
Jayson Tatum	22.75	17	25.30933622	-0.736486157	1
Franz Wagner	28	0	13.5	0	0
Kentavious Caldwell-Pope	40.75	2	2.136000936	-1.212025657	0
Franz Wagner	41.75	0	0.25	0	1
Al Horford	39.75	23	23.06648001	-1.494855691	1
Cory Joseph	18.75	-14	26.71259066	0.551654983	0
Franz Wagner	21.75	-16	25.41775954	0.680885258	0
Al Horford	32.75	0	8.75	0	0
Paolo Banchero	29.75	2	11.91899744	-0.16859694	1
Jayson Tatum	28	0	13.5	0	1



Hierarchical Modeling: examples

- Let Y_{ij} be an indicator variable for whether the i^{th} shot from the j^{th} player is a success.
- We are going to predict this response based on distance, X_1, and the player taking the shot.
- This lends itself naturally to a hierarchical logistic regression model:

$$\left(Y_{ij} \mid \alpha_j, \beta_1, X_{1,ij}\right) \sim Bern\left(\ln\left(\frac{p_{ij}}{1-p_{ij}}\right) = \alpha_j + \beta_1 X_{1,ij}\right)$$

$$\alpha_j \sim N(\alpha_0, \sigma_\alpha^2)$$

- What are the interpretations of these parameters?
- Wait, how is this a Bayesian model?