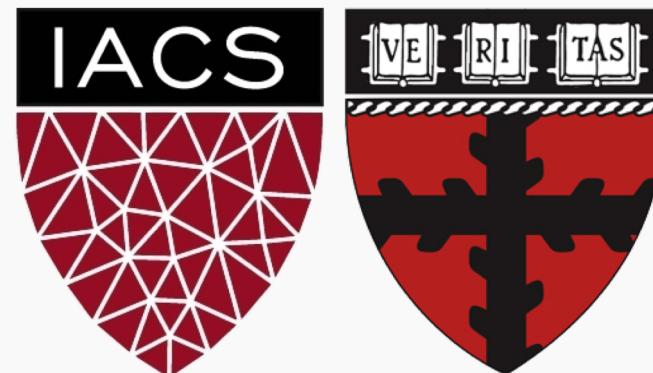


# Lecture 18: Multiclass Logistic Regression

CS109A Introduction to Data Science  
Pavlos Protopapas, Kevin Rader and Chris Tanner



# Lecture Outline: Multiclass Classification

---

Multinomial Logistic Regression

OvR Logistic Regression

$k$ -NN for multiclass

# 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:

1. ordinal logistic regression
2. multinomial logistic regression.

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

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

# Multinomial Logistic Regression



# 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:

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

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: a model to predict  $y = 1$  (CS) from  $y = 3$  (others) and a separate model to predict  $y = 2$  (Stat) from  $y = 3$  (others).

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

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

# Multinomial Logistic Regression: the model

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

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

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

⋮

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

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(nfl_19[['ydstogo','down']],nfl_19['playtype'])
print(mlogit.intercept_)
print(mlogit.coef_)

[-6.78443446  4.46327069  2.32116377]
[[ 0.08130727  1.86026457]
 [-0.10687815 -1.33128781]
 [ 0.02557089 -0.52897676]]
```

```
pd.crosstab(nfl_19["play_type"],
            nfl_19["playtype"])
```

playtype	0	1	2
play_type			
field_goal	978	0	0
pass	0	0	18981
punt	2150	0	0
qb_kneel	393	0	0
qb_spike	72	0	0
run	0	12994	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:

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

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



# One vs. Rest (OvR) Logistic Regression

---

An alternative multiclass logistic regression model in sklearn is called the 'One vs. Rest' 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.

An example to predict play call from the NFL data follows...



# 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:

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

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

⋮

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

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:

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

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



# OVR Logistic Regression in Python

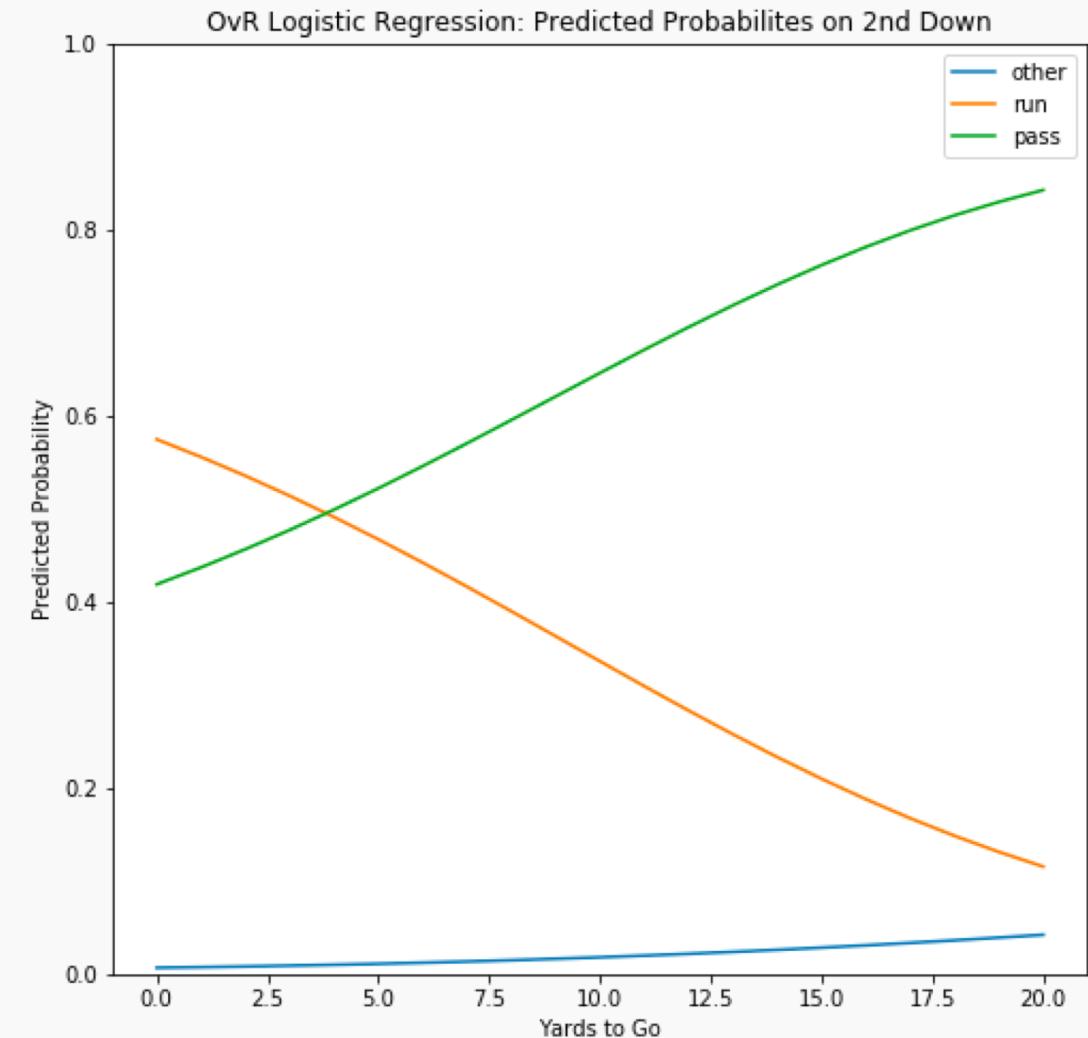
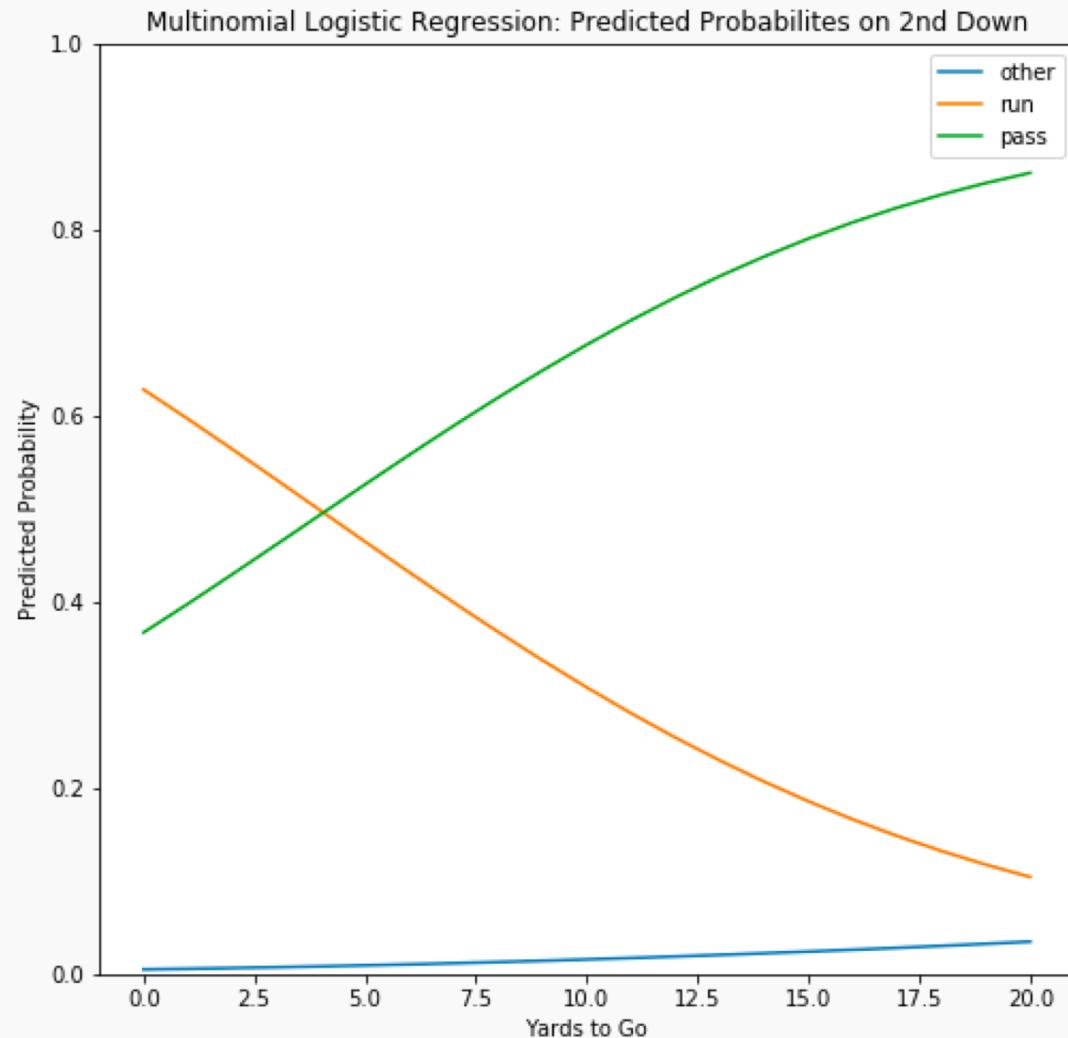
```
ovrlogit = LogisticRegression(penalty='none', multi_class='ovr')

ovrlogit.fit(nfl_19[['ydstogo', 'down']],nfl_19['playtype'])
print(ovrlogit.intercept_)
print(ovrlogit.coef_)

[-10.03308293    2.47802369   -0.10976034]
[[ 0.07547391   2.55126265]
 [-0.14272983  -0.98841346]
 [ 0.03672441  -0.03755844]]
```

Phew! This one is as expected ☺

# Predicting Type of Play in the NFL



# 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

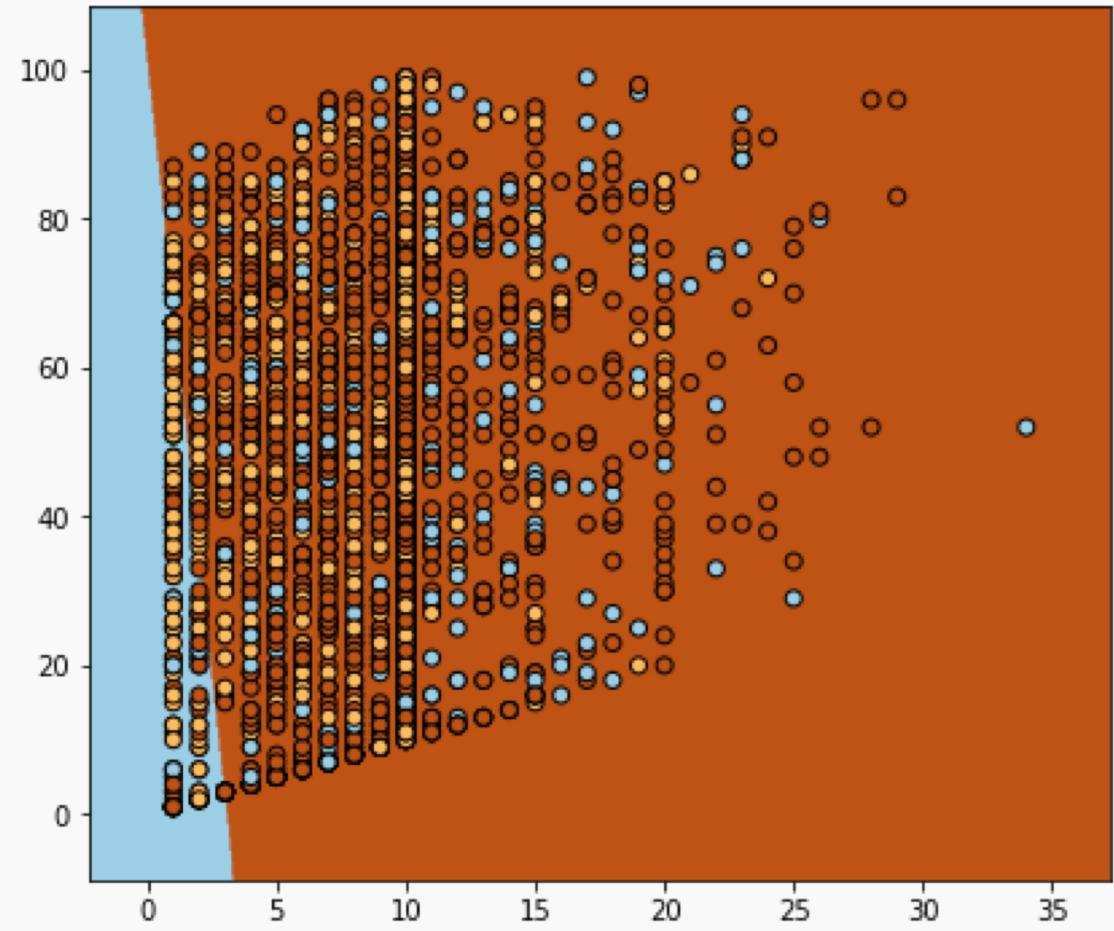
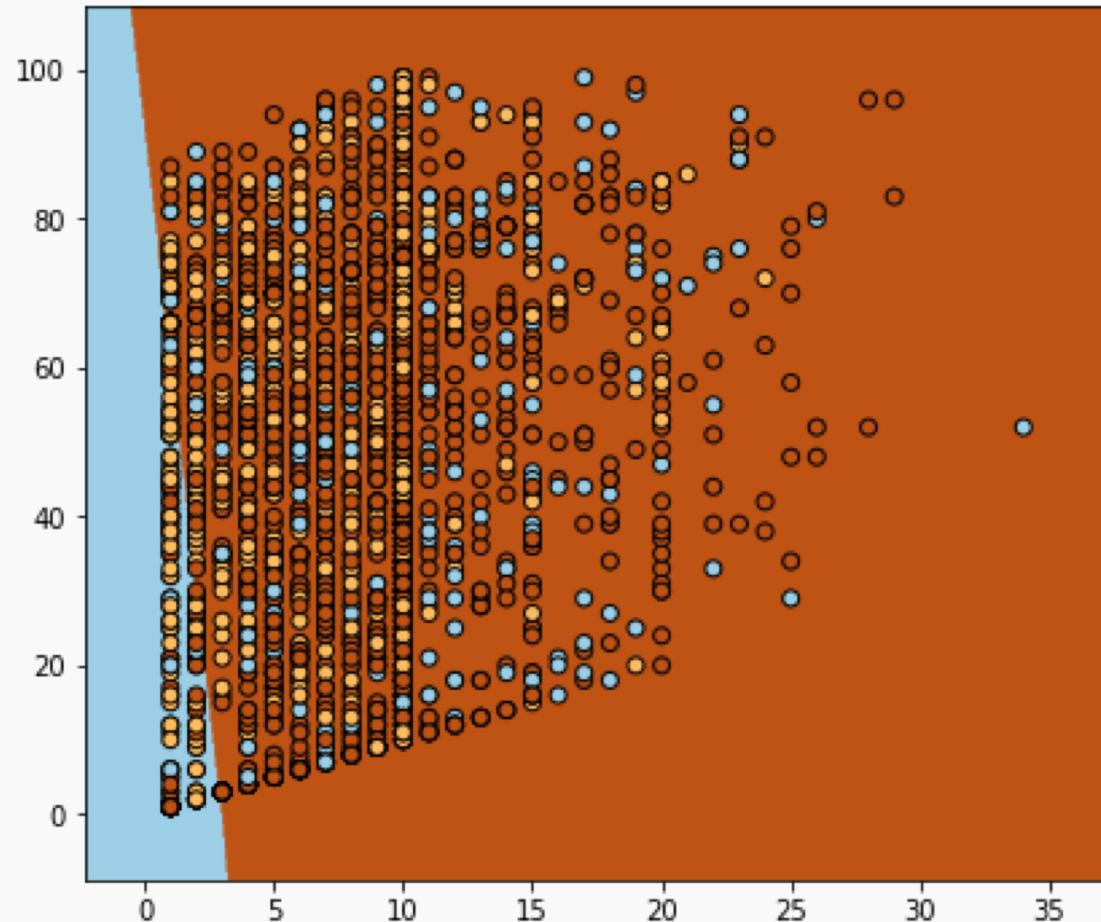
```
print(mlogit.predict_proba(nfl_19[['ydstogo', 'down']])[0:5,:])
print(mlogit.predict(nfl_19[['ydstogo', 'down']])[0:5])
```

```
[[0.001048  0.50329827  0.49565372]
 [0.01559789 0.30793278  0.67646934]
 [0.17275682 0.14020122  0.68704196]
 [0.82376365 0.00418569  0.17205066]
 [0.001048  0.50329827  0.49565372]]
[1 2 2 0 1]
```

```
print(ovrlogit.predict_proba(nfl_19[['ydstogo', 'down']])[0:5,:])
print(ovrlogit.predict(nfl_19[['ydstogo', 'down']])[0:5])
```

```
[[0.00111671 0.48115571  0.51772757]
 [0.01792012 0.33603242  0.64604746]
 [0.19847963 0.15493954  0.64658083]
 [0.57254676 0.0088239   0.41862935]
 [0.00111671 0.48115571  0.51772757]]
[2 2 2 0 2]
```

# Classification Boundary for 3+ Classes in sklearn



# 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 **binary crossentropy**:

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

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.

# Multiclass $k$ -NN



# $k$ -NN for 3+ Classes

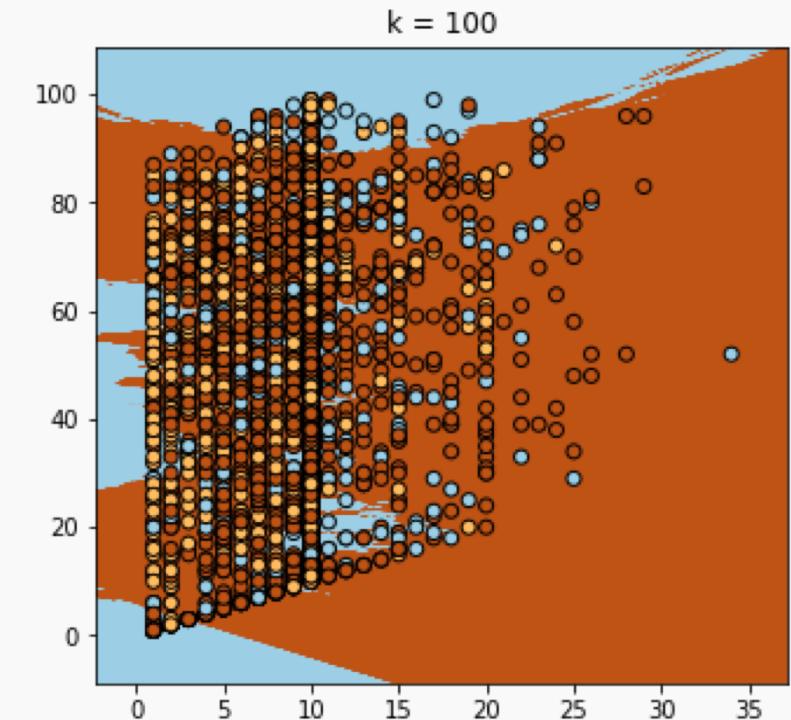
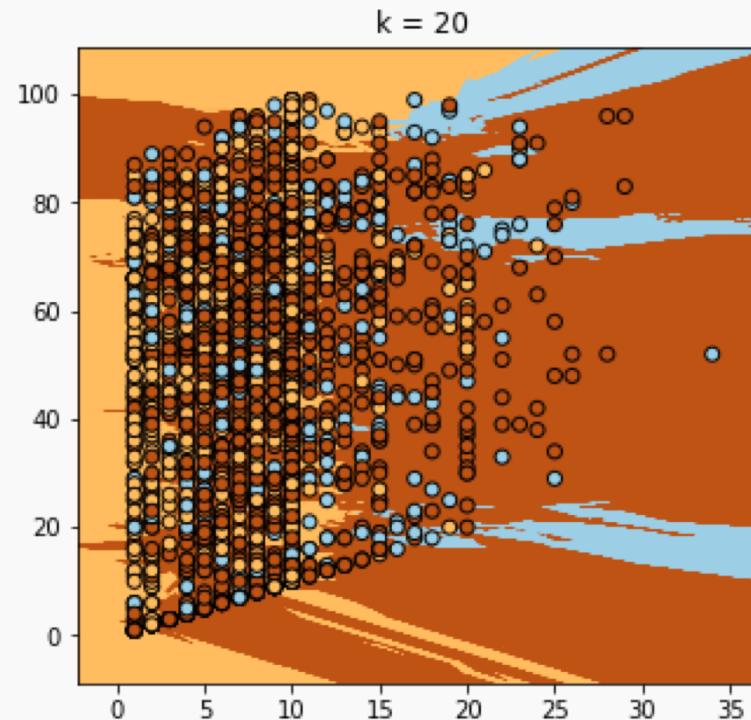
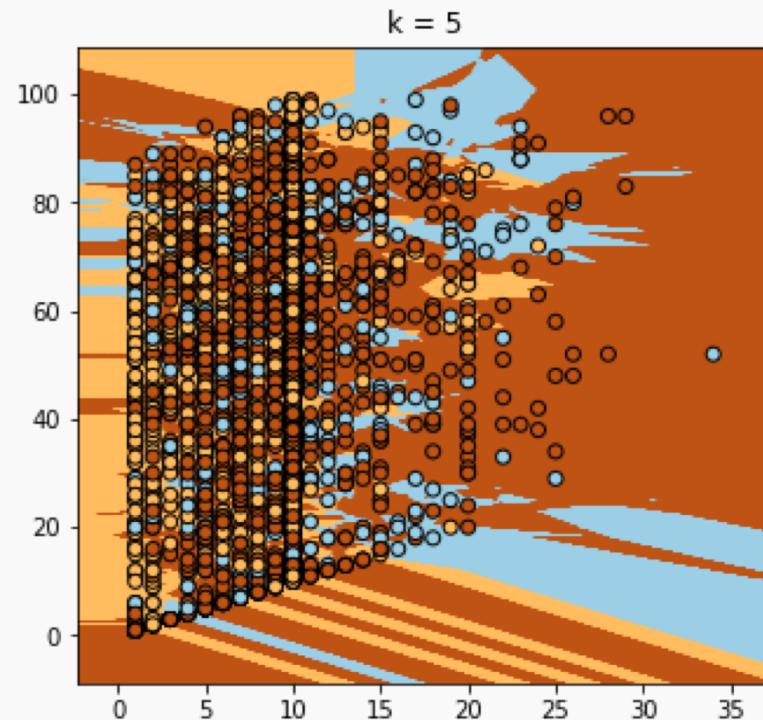
---

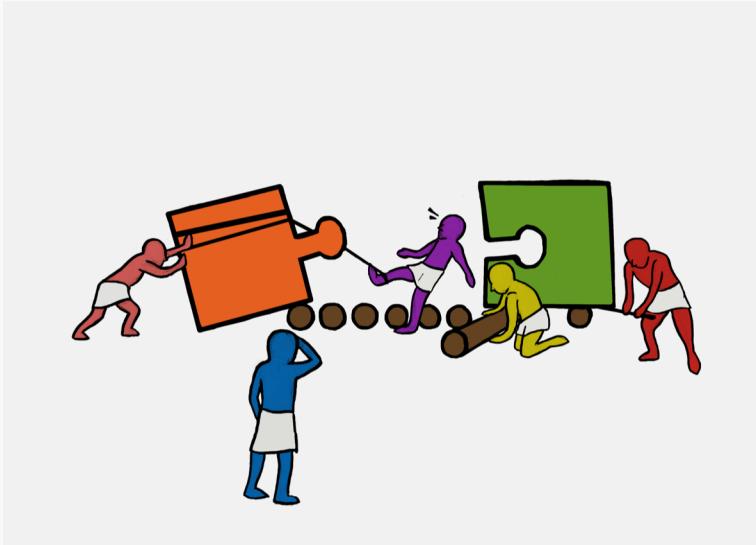
Extending the  $k$ -NN classification model to 3+ classes is much simpler!

Remember,  $k$ -NN is done in two steps:

- 1. Find you  $k$  neighbors:** this is still done in the exact same way
  - be careful of the scaling of your predictors!
- 2. Predict based on your neighborhood:**
  - Predicting probabilities: just use the observed proportions
  - Predicting classes: plurality wins!

# $k$ -NN for 3+ Classes: NFL Data





Exercise Time!

Ex. 1 (graded): Basic multiclass Logistic Regression and  $k$ -NN in  
sklearn (15+ min)

Ex. 2 (not graded): A little more thinking and understanding  
(30+ min)