Week 15: Classification and Final Review

Data 8 Tutoring

# 1. Classification

## Key Concepts

**Definition**

Classification is used to make predictions based on existing data. Some questions that we can answer with classification are:

* Is person A going to vote for a certain politician?
* Is a certain purchase an instance of credit card fraud?
* Do I have a certain disease?

*Observations*: existing individuals in a population that you have data on.

*Attributes*: characteristics of the individuals that you will build the classifier for. In Data 8, attributes are binary (yes or no, 1 or 0).

*Population*: A larger group of individuals, who you don’t know the attributes for. A classifier is built in order to predict the attributes of those in the population.

**Training and Testing Data**

The reason why we’ve made a classifier is so that we can make predictions on new data from our **underlying population**. How do we know whether the classifier we’ve made actually makes accurate predictions?

We can split our original dataset into two sets at random: **training data** and **testing data**, in order to validate the classifier’s accuracy. A typical training and testing proportion split might be 80/20.

## Practice Problems

**1.1** Sue and Avery are deciding on which kind of nearest neighbors classifier they want to use. Avery says that using a larger number of neighbors will *always* result in more accurate predictions. Sue disagrees. Who is right, and why?

(Hint: Think about what happens when we have a data set with *n* points and we use an *n*-nearest neighbors classifier.

Sue is right. If we have n points and use a n-nearest neighbors classifier, we will always get the same classification. Consider a data set of 13 points, 10 of which are blue and 3 of which are yellow. Using a 7-nearest neighbor classifier would always result in a blue classification, no matter where the unknown point is. Therefore, using a larger number of neighbors does not always result in more accurate predictions.



**2.1** In order to make the model as accurate as possible, should we use all of our data to train the model?

No we shouldn’t, because we need to test the final model out on some data to see how good it is. If we train the model on all of the available data, then testing it on the same data will yield an error of 0 (think about why this makes sense). If we have no way of knowing how good the model is, then the model is useless.

**2.2** How should we split our data into training and testing sets? Why?

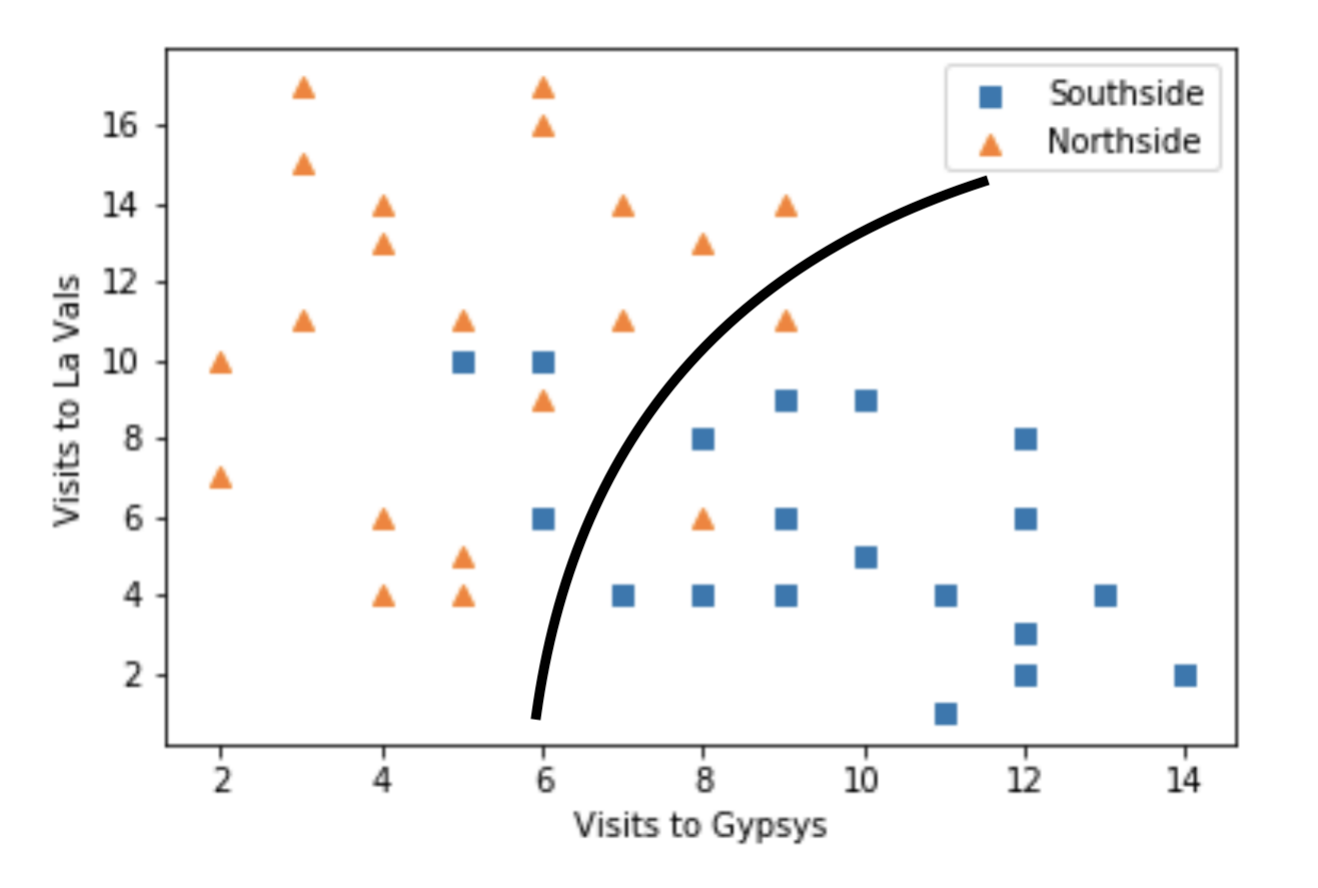
We should split the data randomly. If not, we risk training only on some pattern that appears in the training set, and our classifier won’t generalize well to the test set and the real world.

**2.3** What would happen if we used 10% of the data for training, and 90% for testing? How much data do you think should be in training set?

Generally we want to do a 80/20 split. Things to keep in mind: The more data we use in our training set, the more we risk overfitting (there isn’t enough new data to test our model and it performs artificially well) If our training dataset is too small, we may not have enough data to train a high-performing model.

A student is trying to build a classifier that classifies Berkeley students as residents of Northside or Southside. The student has a random sample of Berkeley students all of whom live on Northside or Southside. For each student she records whether the student lives on Northside or Southside, the number of times the student went to La Val's (on Northside) in the last 6 months, and the number of times the student went to Gypsy's (on Southside) in the last 6 months.

**3.1** Draw a decision boundary for a 5 nearest neighbor classifier on the scatter plot below.



Hypothesis Testing/Inference Review

# 1 Testing Chance Models

Roulette is a casino game in which a ball falls into one pocket in a spinning wheel, and players bet on the color of the pocket in which the ball falls. If the player correctly picks the color, they win. 18 of the 38 pockets in a roulette wheel are red. You play 30 games of roulette, bet on red every time, and win 20 of those games. You become suspicious about the fairness of this roulette game.

**1.1** State a null and alternative hypothesis to see whether the roulette game is biased towards red.

Null Hypothesis:

The roulette game is fair, with a 18/38 chance of the ball falling into a red pocket. The observed number of wins on red is due to chance.

Alternative Hypothesis:

The ball is more likely to fall into red pockets.

**1.2** With your alternative hypothesis in mind, choose a test statistic and calculate its observed value. Your test statistic should be large for data favoring the alternative hypothesis.

Test Statistic:

The proportion of wins on red in 30 games.

Observed Value:

20/30 = 2/3

**1.3** Complete the function prop\_wins\_in\_30\_games(), which takes in no arguments and simulates playing the game 30 times and returns the proportion of wins when you guess red every time.

def prop\_wins\_in\_30\_games():

proportions = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

thirty\_games = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

prop\_wins = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

return \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

def prop\_wins\_in\_30\_games():

proportions = make\_array(18/38, 20/38)

thirty\_games = sample\_proportions(30, proportions)

prop\_wins = thirty\_games.item(0)

return prop\_wins

**1.4** Complete the code below to simulate an empirical distribution of the test statistic using 10000 iterations, storing the statistics in an array called simulated\_statistics.

simulated\_statistics = make\_array()

for i in \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_:

prop\_win = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

simulated\_statistics = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

simulated\_statistics = make\_array()

for i in np.arange(10000):

prop\_win = prop\_wins\_in\_30\_games()

simulated\_statistics = np.append(simulated\_statistics, prop\_win)

**1.5** Write a line of code to calculate the p-value.

p\_value = np.count\_nonzero(simulated\_statistics >= (2/3))/10000

**1.6** Suppose you find a p-value of 0.0103. What do you conclude about the null hypothesis, at a p-value cutoff of 5%?

Reject the null hypothesis, as the empirical p-value is below the cutoff.

# 2 A/B Testing

We are examining the weights of a population of cats and dogs. You are given a random sample from this population, stored in the table pets, which has two columns. The first column ‘Animal’ contains a string, either ‘Cat’ or ‘Dog’. The second column ‘Weight’ contains the weights of each of the animals in pounds as floats. You notice that the average weight of dogs in your sample is 2 pounds heavier than the average weight of cats in your sample.

**2.1** State a null and alternative hypothesis to see if dogs weigh more than cats on average in the population.

Null Hypothesis:

The distributions of weights of dogs and weights of cats come from the same underlying population. Any difference is due to chance.

Alternative Hypothesis:

The weights of dogs, on average, are greater than the weights of cats in the population

**2.2** With your alternative hypothesis in mind, choose a test statistic and calculate its observed value. Your test statistic should be large for data favoring the alternative hypothesis.

Test Statistic:

The difference between the mean dog weight and mean cat weight

Observed Value:

2

**2.3** Complete the function one\_shuffled\_table\_stat() which takes in no arguments and returns one value of the test statistic.

def one\_shuffed\_table\_stat():

shuf\_table = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

shuf\_weights = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

shuf\_tbl\_with\_weights = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

grouped\_with\_mean = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dog\_mean = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

cat\_mean = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

return \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

def one\_shuffed\_table\_stat():

shuffled\_table = pets.sample(with\_replacement = False)

shuffled\_weights = shuffled\_table.column(‘Weights’)

shuffled\_tbl\_with\_weights = pets.with\_column(‘

Shuffled Weights ’, shuffled\_weights)

grouped\_with\_mean = shuffled\_tbl\_with\_weights.group(‘Animal’, np.mean)

dog\_mean = grouped\_with\_mean.where(‘Animal’, ‘Dog’).column(2).item(0)

cat\_mean = grouped\_with\_mean.where(‘Animal’, ‘Cat’).column(2).item(0)

return dog\_mean - cat\_mean

**2.4** Complete the below code to simulate an empirical distribution of 5000 test statistics under the assumptions of the null hypothesis.

diffs = make\_array()

for i in np.arange(5000):

diff = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

diffs = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

diffs = make\_array()

for i in np.arange(5000):

diff = one\_shuffed\_table\_stat()

diffs = np.append(diffs, diff)

**2.5** Write a line of code to calculate your p-value.

p\_value = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

p\_value = np.count\_nonzero(diffs >= 2)/5000

**2.6** Suppose you find a p-value of 0.13. What do you conclude, at a p-value cutoff of 5%?

0.13 > 0.05, so we fail to reject your null hypothesis.

# 3 Regression Inference

You take a sample of Data 8 students and ask them about their daily consumption of coffee and their midterm exam scores. Assume you are given a table students with the columns cups and score. The column cups contains the daily consumption of coffee and score contains the midterm exam score for each student from the sample. You perform linear regression and find a slope of 0.13 points per cup of coffee.

**3.1** State a null and alternative hypothesis to see whether this slope was due to randomness in your sample.

Null Hypothesis:

The slope of the true line is 0.

Alternative Hypothesis:

The slope of the true line is not 0.

**3.2** Write a function slope that takes in a table, tbl,, and returns the slope of the least-squares line using the first column to predict values of the second column.

def slope(tbl):

x = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

y = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

x\_su = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

y\_su = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

r = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

slope = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

return slope

def slope(tbl):

x = tbl.column(0)

y = tbl.column(1)

x\_su =(x-np.mean(x))/np.std(x)

y\_su =(y-np.mean(y))/np.std(y)

r = np.mean(x\_su\*y\_su)

slope = r \* (np.std(y) / np.std(x))

return slope

**3.3** Complete the code to generate 5000 bootstrap resample slopes and then calculate a 95% confidence interval for the slope. Assume the function slope(tbl) has been implemented correctly.

slopes = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
for i in \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_:  
 resample\_slope = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 slopes = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

left\_end = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
right\_end = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
interval = make\_array(left\_end, right\_end)

slopes = make\_array()  
for i in np.arange(5000):  
 resample\_slope = slope(students.sample())  
 slopes = np.append(slopes, resample\_slope)

left\_end = percentile(2.5, slopes)  
right\_end = percentile(97.5, slopes)  
interval = make\_array(left\_end, right\_end)

**3.4** Suppose you find the confidence interval [0.02, 0.24]. What do you conclude about your hypotheses at a p-value cutoff of 5%? What about at a p-value cutoff of 10%?

We observe that our 95% confidence interval does not contain 0. Therefor e, at a p-value cutoff of 5%, we know we can reject the null hypothesis. At 10%, we can still reject the null hypothesis because we know that a 90% confidence interval would be *narrower* than a 95% confidence interval and would therefore still exclude 0. At a 1% cutoff, we would not be able to say anything about rejecting or failing to reject the null hypothesis - this is because the confidence interval is wider, meaning we wouldn’t be able to tell if 0 is in the 99% confidence interval.

3.5 Your friend who hasn’t taken Data 8 looks at your result and asks you what the confidence interval means. Which one of the following is a correct response?

1. For 95% of students, there is a relationship between coffee consumption and midterm score.
2. There is a 95% probability that the true slope is between 0.02 and 0.24.
3. There is a 95% probability that our sampling process (and the code above) produces an interval that contains the true slope.