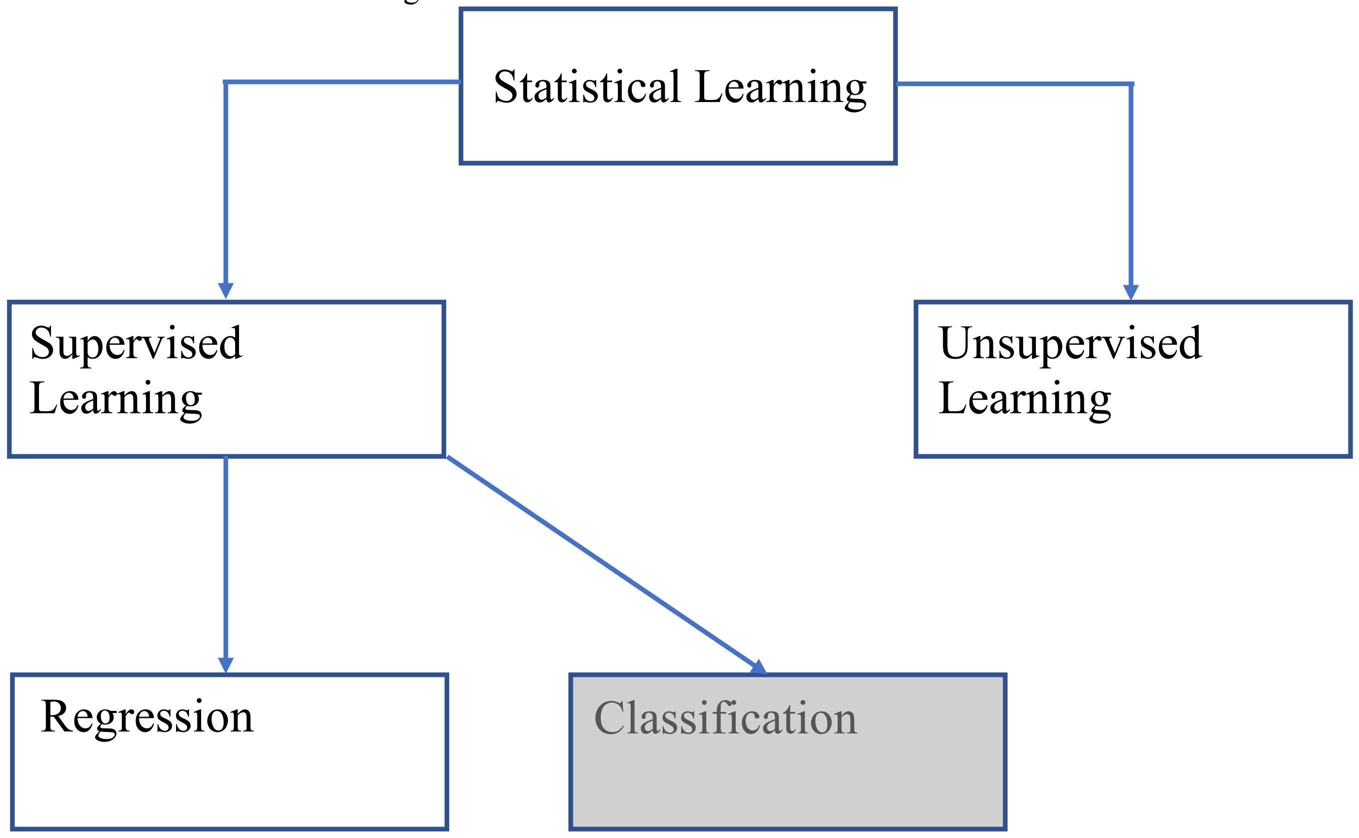
STAT 380 – kNN Classification (Lecture 11)

EXAMPLE 1: What do the following situations have in common?

- We are developing a spam filter that will predict whether an incoming email is spam or not.
- Using driver characteristics, we wish to predict whether the driver will file a claim.
- Using student GPAs, we want to predict whether they will be accepted into grad school.
- When deciding whether to extend a line of credit, a bank wishes to categorize the applicant as high-risk, average-risk, or low-risk.
- An online banking service must be able to determine whether or not a transaction being performed on the site is fraudulent, on the basis of the user's IP address, past transaction history, and so forth.
- On the basis of DNA sequence data for a number of patients with and without a given disease, a biologist would like to figure out which DNA mutations are deleterious (disease-causing) and which are not.

NOTE: Consider the following:



Definition: Broadly speaking, supervised learning involves building a statistical model for predicting, or estimating, an output based on one or more inputs.

NOTE: Input variables go by many names including: x, predictor, regressor, independent variable, feature, input, explanatory variable, and attribute.

NOTE: The output variable also has many names including: y, response, dependent variable, target, output.

Definition: In unsupervised learning, there are inputs but no supervising output; nevertheless we can learn relationships and structure from such data.

NOTE: Supervised learning problems fall into two broad categories: regression and classification. Regression techniques are used when the output/response variable is quantitative. Classification techniques are used when the response is categorical.

IDEA: We now want to learn how to build models when the response variable (y) is categorical. This process known as classification.

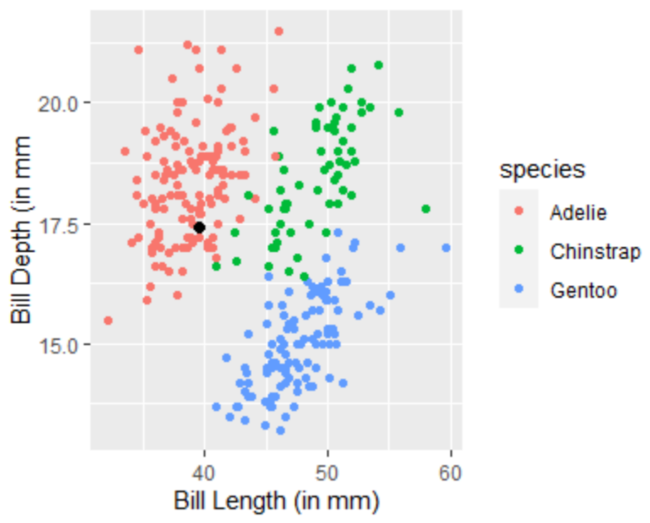
EXAMPLE 2: The problem uses the dataset penguins from the palmerpenguins package.

a. Remove any observations with missing data. Then, perform a 90/10 training/testing split on the penguins dataset. Use a seed of 321.

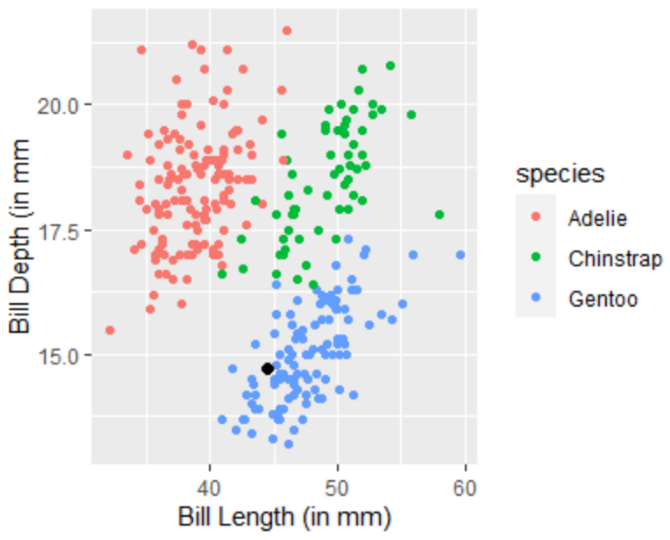
```r
#Create object named penguins in Environment; Requires palmerpenguins package
data("penguins")

#Omit NA's
penguins <- na.omit(penguins)

#Train/Test split (90/10, 321)
set.seed(321)
train_ind <- sample(1:nrow(penguins), floor(0.9*nrow(penguins)))
set.seed(NULL)

Train <- penguins[train_ind, ]
Test <- penguins[-train_ind, ]
```

b. Using the training data, create a scatterplot showing the relationship between bill length and bill depth. Use a different color for each species. Add a black circle of size 2 for the observation corresponding to the 1st observation in the testing set. Predict the species of this penguin.

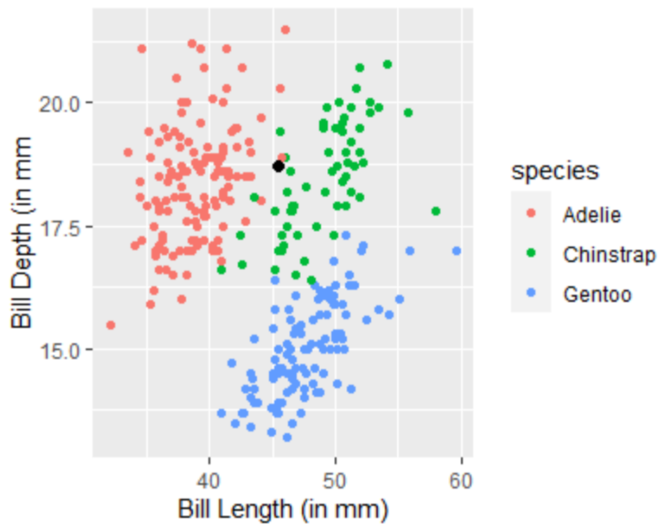
```r
ggplot(data = Train, mapping = aes(x = bill_length_mm,
                                   y = bill_depth_mm,
                                   color = species)) +
  geom_point() +
  geom_point(data = Test[1 , ], color = "black", size = 2) +
  labs(x = "Bill Length (in mm)",
       y = "Bill Depth (in mm)")
```



c. Include a black circle of size 2 for the observation corresponding to the 22nd observation in the testing set. Predict the species of this penguin.



d. Include a black circle of size 2 for the observation corresponding to the 27th observation in the testing set. Predict the species of this penguin.



e. Predict the species of each penguin using bill length and bill depth using kNN with 3 neighbors. Prep the data (omit missing, create indicators if necessary, scale the inputs, perform 90/10 training/testing split using a seed of 321). There are many ways to do this in R, but we'll use the knn function in the FNN library.

```r
#Create an object called penguins in my environment
data("penguins")

#Omit NA's
penguins <- na.omit(penguins)

#Scale Data
xvars <- c("bill_length_mm", "bill_depth_mm")
penguins[ , xvars] <- scale(penguins[ , xvars], center = TRUE, scale = TRUE)

# Train/Test split
set.seed(321)
train_ind <- sample(1:nrow(penguins), floor(0.9*nrow(penguins)))
set.seed(NULL)

Train <- penguins[train_ind, ]
Test <- penguins[-train_ind, ]

#Build kNN classification model
knn_res <- knn(train = Train[ , xvars, drop = FALSE],
               test = Test[ , xvars, drop = FALSE],
               cl = Train$species,
               k = 3)

#Access Predictions
knn_res #In general, do NOT include this in your code;
```

NOTE: There are some differences in the code between kNN regression and kNN classification

Aspect	kNN Regression	kNN Classification
FNN function		
Specifying Response in Train		
Accessing Predictions		
Assessing Quality of Predictions (Not shown in code above)		

f. Add the predictions to the test set and name the new variable pred_species.

```
#Add predictions to the Test set
Test <- Test %>%
          mutate(pred_species = knn_res)
```

g. We made the predictions for the 1st, 22nd, and 27th observations in the Test set. Were they correct?

```
Test[c(1, 22, 27), ] %>%
  select(species, pred_species)
```

```
## # A tibble: 3 × 2
##   species   pred_species
##   <fct>     <fct>
## 1 Adelie    Adelie
## 2 Gentoo    Gentoo
## 3 Chinstrap Chinstrap
```

EXAMPLE 3: If we want to assess the quality of our predictions, explain why it does not make sense to use:

$$\text{Mean Squared Error} = MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}(x_i)\right)^2$$

Confusion Matrix

Definition: A confusion matrix is a table that is often used to summarize the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

NOTE: In this class, we will present the confusion matrix as a table in which the rows represent the predicted values of the categorical variable, the columns represent the observed values of the categorical variable, and the number at the intersection of each row and column represents the number of observations in the combination of predicted and observed values. (Sometimes, the roles of the rows and columns will be switched.)

EXAMPLE 4: In Example 2 Part e., we were asked to predict the species of each penguin in the test set using bill length and bill depth. We used kNN with 3 nearest neighbors.

a. Create a confusion matrix to summarize the predictions on the test set.

```
#Confusion Matrix - Assumes you've added predictions to Test
table(Test$pred_species, Test$species)
```

```
##
##             Adelie Chinstrap Gentoo
##   Adelie        15         1      0
##   Chinstrap      0         7      1
##   Gentoo         0         0     10
```

```
#If predictions are not in Test, use: table(knn_res, Test$species)
```

b. How many observations were correctly predicted?

c. Find the overall accuracy of the classifier on the testing set.

Definition: The accuracy of a classifier is the proportion of predictions that were correct.

NOTE: To find the accuracy through code, use: