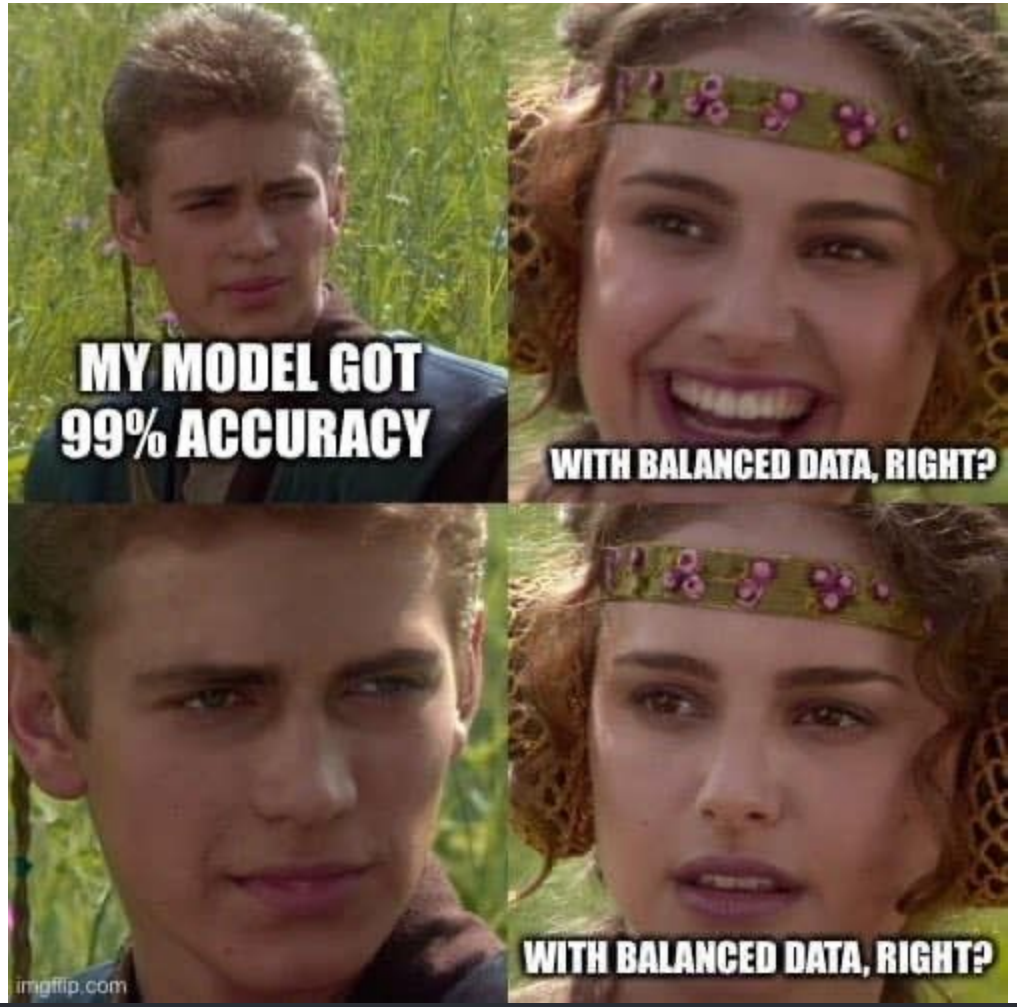

MY MODEL GOT
99% ACCURACY
WITH BALANCED DATA, RIGHT?
WITH BALANCED DATA, RIGHT?
imgflip.com

# Warm Up

Type in chat:

How can we be certain our models will perform well in the real world with new data?

# Announcements

1. Belt Exam for Stack 2
    a. March 11th - March 13th
    b. Must have attended 80% of classes
    c. Must have passed 90% of assignments
    d. All assignments & resubmits from weeks 1 & 2 are due by Friday at 9am PST (March 11th)

2. Week 3 Assignments
    a. Highly encourage you to finish these before the belt exam.

3. Project 2 - Final
    a. Still in the process of grading this assignment

# Tips and Tricks

## CopyPasta

**Copy/Pasting Code:**

1. Hurts your learning
    a. Slower coder
    b. Less code understanding
    c. Unprepared for technical interviews
2. Causes errors:
    a. Most common source of serious errors
    b. Code looks amateur

## Challenge

**I challenge you to type all code for the rest of this program.**
**(no copying and pasting)**

# Tips and Tricks

## Project 2, part 1: Choosing Data

Look for data that:

1. Has a target column to predict.
2. With a target that is likely predictable using available features.
3. Is not overly complex or overly simple.
    a. 6-15 columns that you mostly understand and can explain
    b. 1000-100,000 rows with not too much missing data (less than 10%)
4. Is something you are interested in.
    a. Be ready to answer the question: Why did this project interest you?

**Try Googling ‘datasets for machine learning’ and look through articles, sources, lists, repositories.**

# Learning Goals

At the end of this class you will be able to:

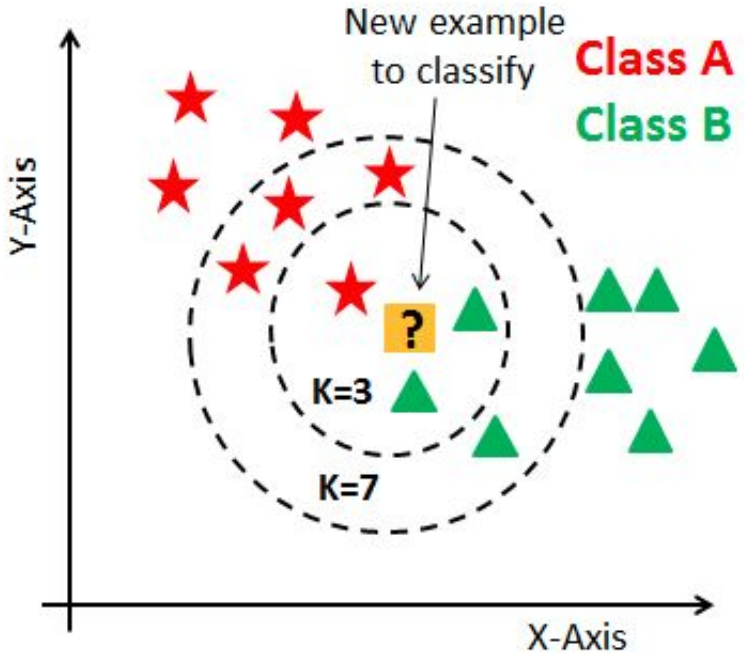
1. Know how a KNN model classifies a new data point
2. Read a confusion matrix to understand what kinds of mistakes your model is making
3. Determine which classification metric is most important for a given business problem
4. Evaluate a classification model in sklearn

# Classification Models

# K-Nearest Neighbors (KNN)

You decide the K number of neighbors

**Regularization: Adjust K**



# Classification Evaluation Metrics

- Accuracy
- Precision
- Recall (Sensitivity)
- Specificity
- F1 Score

# Classification Evaluation Metrics

**Binary Classification:**

- Predict 1 of 2 classes, positive (1) or negative (0)

**Multiclass Classification**

- Predict 1 of 3 or more classes.

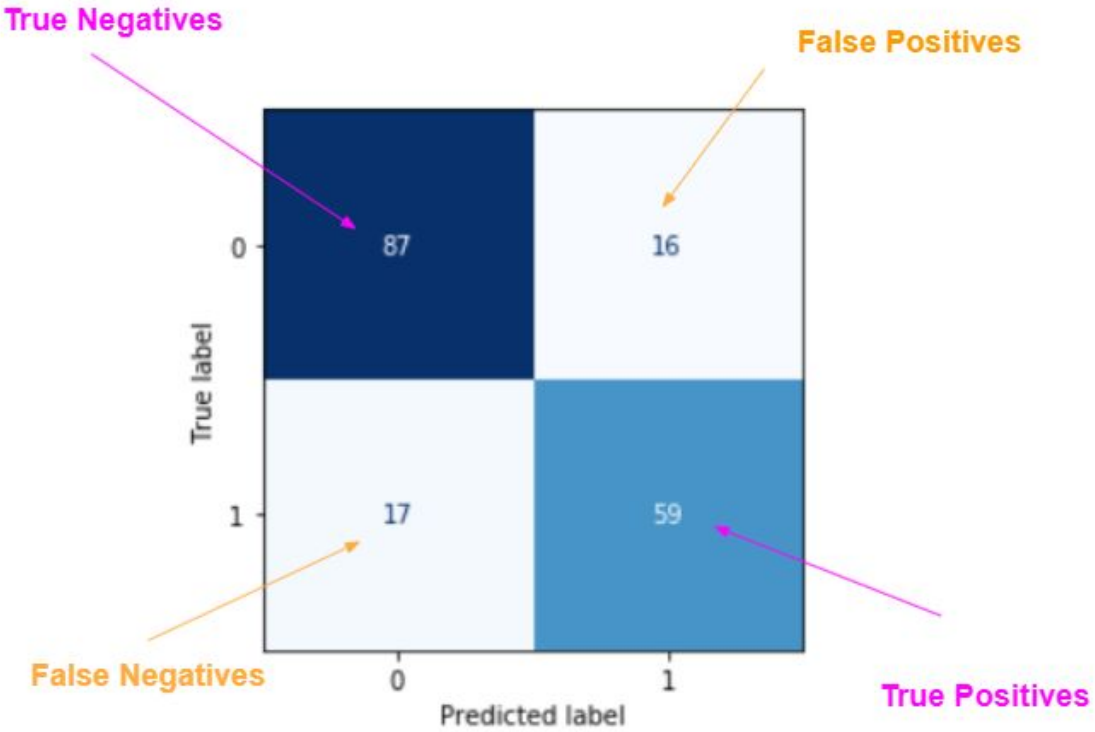
# Classification Evaluation Metrics

**We are going to start with Binary Classification Metrics**

# The Confusion Matrix:

A more complete story

Let's look at this example of a confusion matrix (this is also in the learn platform)
To help us understand the terminology and calculations in classification metrics.

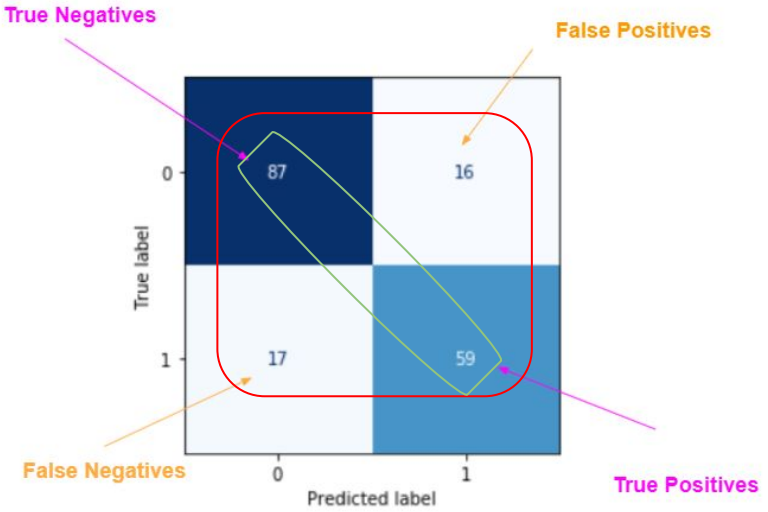


# Accuracy

The percent of correct choices out of the total number of predictions

**Accuracy**

$$\frac{\text{True Positives + True Negatives}}{\text{All Predictions}}$$



# Accuracy

**Pros**:

1. One number describes all classes
2. Easy to interpret
3. Works the same with binary or multiclass

# Accuracy

**Pros**:

1. One number describes all classes
2. Easy to interpret
3. Works the same with binary or multiclass

**Cons**:

1. Does not tell the whole story
2. Can be misleading when classes not balanced

# Accuracy Depends on Class Balance

```
1   y_train.value_counts()
```

```
1    40
0    36
2    35
Name: target, dtype: int64
```

```
1   y_train.value_counts(normalize=True)
```

```
1    0.360360
0    0.324324
2    0.315315
Name: target, dtype: float64
```

## Test Set

| Prediction | |
|---|---|
| 0 | |
| 0 | |
| 0 | |
| 0 | |
| 0 | |
| 0 | |
| 0 | |
| 0 | |
| 0 | |
| 0 | |
| 0 | |

A simple **baseline model** predicts 0 for every row in the test set

# Test Set

| Prediction | Actual |
|---|---|
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 1 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |

How did our baseline do?

The baseline got 9 out of 10 correct.

That is an accuracy of 90%!!!

While it is tempting to be excited about an overall accuracy of 90%, we didn't really achieve anything with our baseline!

➔ If we were trying to identify people who had a disease, we failed!

While it is tempting to be excited about an overall accuracy of 90%, we didn't really achieve anything with our baseline!

➔ If we were trying to identify people who had a disease, we failed!

➔ If we were trying to identify spam emails, we failed!

While it is tempting to be excited about an overall accuracy of 90%, we didn't really achieve anything with our baseline!

➔ If we were trying to identify people who had a disease, we failed!

➔ If we were trying to identify spam emails, we failed!

➔ If we were trying to identify fraudulent credit card purchases, we failed!

★ There is more to evaluating a classification model than just accuracy!

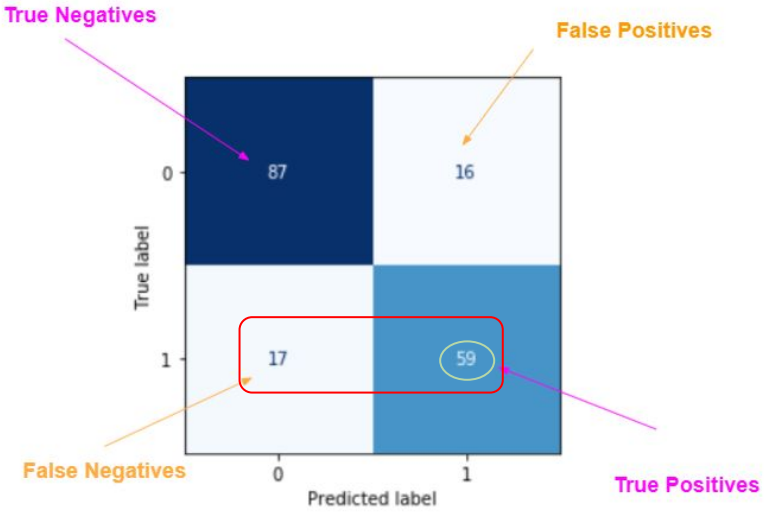
★ And there is absolutely more to making predictions than just our baseline!

# Recall: (Also called “Sensitivity”)

Is our model able to identify the positive cases?

**Recall**

$$\frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$



# Recall: (Also called “Sensitivity”)

Is our model able to identify the positive cases?

**Pros:**

1. Minimizes False Negatives: good at finding positive class
2. Tells a piece of the story

# Recall: (Also called “Sensitivity”)

Is our model able to identify the positive cases?

**Pros:**

1. Minimizes False Negatives: good at finding positive class
2. Tells a piece of the story

**Cons:**

1. Does not penalize false positives
2. Perfect recall can be achieved by over-predicting positives

## Test Set

| Prediction | Actual |
|---|---|
| 1 | 0 |
| 1 | 0 |
| 1 | 0 |
| 1 | 0 |
| 1 | 0 |
| 1 | 1 |
| 1 | 0 |
| 1 | 0 |
| 1 | 0 |
| 1 | 0 |

How did this model do?

True Positives = 1
False Negatives = 0
Recall = 1 / 1+0

The model found all of the positive samples.

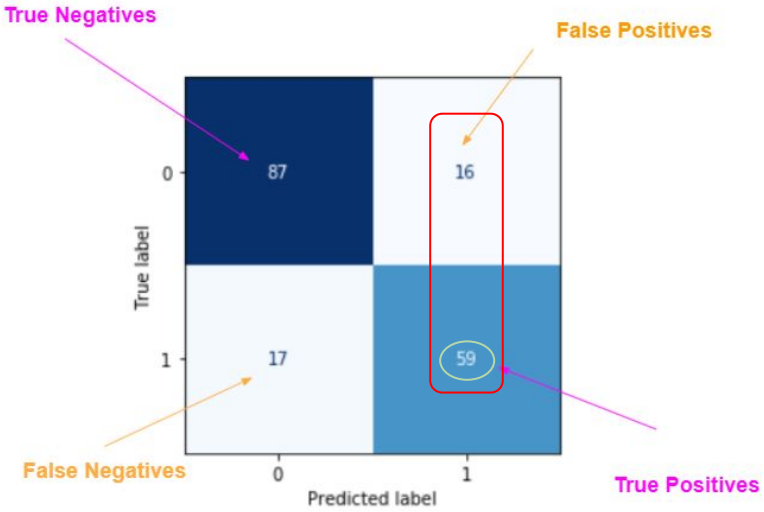
That is a recall of 100%!!!

# Precision (AKA Positive Predictive Value):

Are predicted positives really positive?

**Precision**

$$\frac{\text{True Positives}}{\text{True Positives + False Positives}}$$



# Precision:

Are predicted positives really positive?

**Pros:**

1. Minimizes false positives
2. Prevents over predicting positive class
3. Tells a piece of the story

# Precision:

Are predicted positives really positive?

**Pros:**

1. Minimizes false positives
2. Prevents over predicting positive class
3. Tells a piece of the story

**Cons:**

1. Does not encourage finding positives
2. Perfect precision can be achieve by predicting all negatives

## Test Set

| Prediction | Actual |
|---|---|
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 1 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |

How did our baseline do?

True Positives = 0
False Positives = 0
Precision = 0 / 0+0
(technically undefined)

All predicted positives are positive

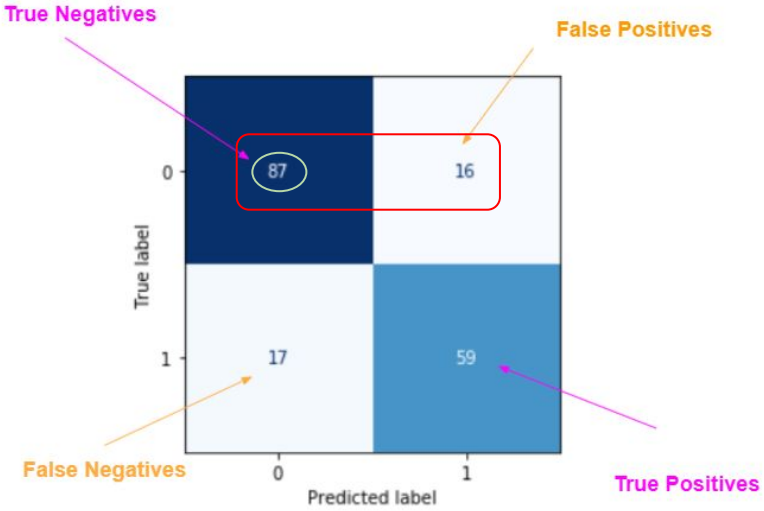
That is a precision of 100%!!!

# Specificity: (Recall on Negative Class)

Is our model able to identify the negative cases?

**Specificity**

$$\frac{\text{True Negatives}}{\text{True Negatives + False Positives}}$$



# Specificity:

Is our model identifying negatives as negatives?

Our original baseline had perfect specificity because it predicted ALL negatives.

But once again this is not useful!

We don’t want to miss those positive cases! (disease, spam, fraud)

**Pros and Cons are Similar to Precision**

## Test Set

| Prediction | Actual |
|---|---|
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 1 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |

How did our baseline do?

The baseline successfully found all negative samples

True negatives = 9
False positives = 0

9 / 9 + 0 = 1!
That is an specificity of 100%!!!

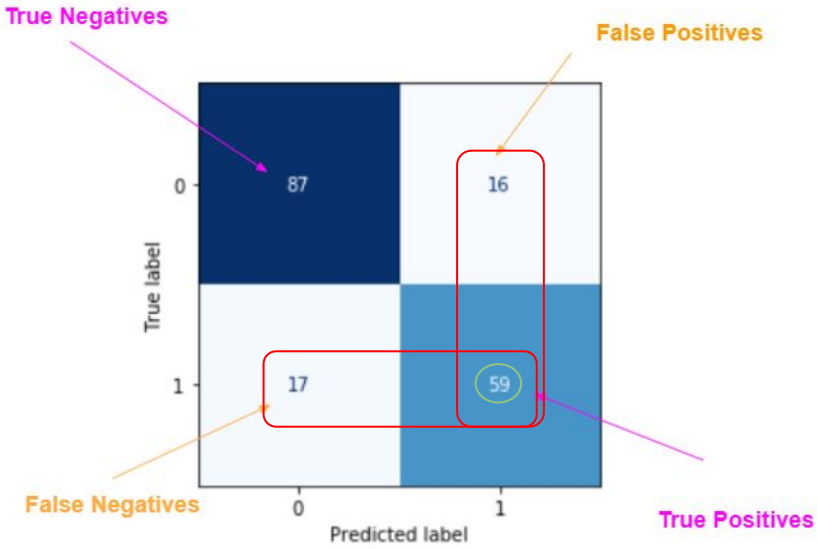
# F1 Score:

## Is our model finding a balance of positives and negatives?

TNR (True Negative Rate)

$$= 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Balances Precision and Recall



# F1 Score:

Are precision and recall both good?

**Pros:**

1. Gives a more complete picture
2. Minimizes false positives AND false negatives

# F1 Score:

Are precision and recall both good?

**Pros:**

1. Gives a more complete picture
2. Minimizes false positives AND false negatives

**Cons:**

1. Hard to interpret
2. Hard to explain, especially to non-data scientists.
3. Not good for presentations

# Classification Metrics Summary

Accuracy

- Use when you have balanced data and you want to find out how many predictions were correct. Good for non-technical audiences.

Recall/Sensitivity

- Use recall when you want to evaluate the rate of false positives.
- e.g. Good to use with cases such as a cancer diagnosis

Precision

- Use precision when the cost of acting on a false positive is really high.
- e.g. Allocating resources/interventions for prisoners who are at risk for being a reoffender.

Specificity

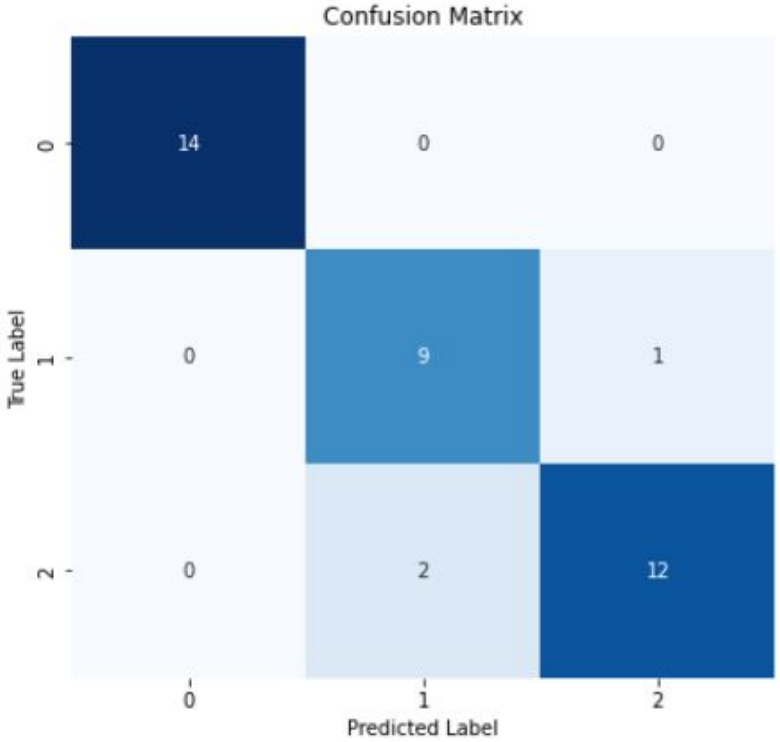
- This is recall on the negative class. Use specificity when you want to evaluate the rate of false negatives.

F1 Score

- Most informative about overall model quality, but is the most difficult to express to a non-technical audience.

# Multiclass Classification Metrics:

- Positive class must be defined to calculate Recall, Precision, Sensitivity, and F1 Score
- False predictions are any prediction other than true predictions
- Can require interpreting metrics on multiple classes to get a complete picture of model behavior.



# Classification Metrics in Python:

```
accuracy = accuracy_score(y_true, y_predictions)
recall = recall_score(y_true, y_predictions, average='macro')
precision = precision_score(y_true, y_predictions, average='macro')
f1 = f1_score(y_true, y_predictions, average='macro')
```

Remember that defining an average or a positive class is necessary for multiclass problems

# Classification Report

Use a print() statement to make classification_report look nice!

All Metrics on All Classes

```
from sklearn.metrics import classification_report

print('Classification Report for Testing Set')

test_report = classification_report(y_test, test_preds)
print(test_report)
```

```
Classification Report for Testing Set
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       0.91      1.00      0.95        10
           2       1.00      0.93      0.96        14

    accuracy                           0.97        38
   macro avg       0.97      0.98      0.97        38
weighted avg       0.98      0.97      0.97        38
```

Classes

Number of observations In each class.

Accuracy

Averages for Each Metric

# Confusion Matrix

```
1 from sklearn.metrics import confusion_matrix
```

```
print('Confusion Matrix for Testing Set')
test_conf_mat = confusion_matrix(y_test, test_preds)
print(test_conf_mat)
```

```
plt.figure(figsize=(8,6))
heatmap(test_conf_mat, xticklabels=iris.target_names,
        yticklabels=iris.target_names, cmap='Greens',
        annot=True)
plt.ylabel('True Values')
plt.xlabel('Predictied Values')
plt.title('Confusion Matrix')
plt.show()
```

Put it into code!

# CodeAlong Notebook!

Data

Challenge Notebook

Practice