Classification Models

Data Boot Camp

Lesson 19.2

Class Objectives

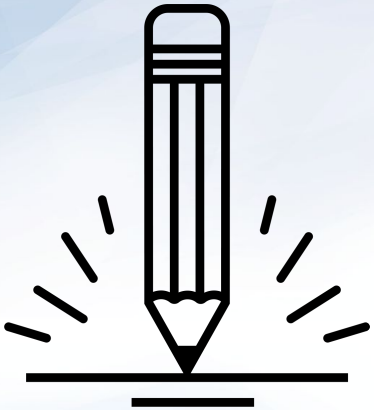
By the end of this lesson, you will be able to:

Calculate and apply fundamental classification algorithms: logistic regression, support vector machine (SVM), and k-nearest neighbors (KNN).

Quantify and validate classification models by using confusion matrix.

Implement one-hot encoding in Pandas and scaling and normalization with scikit-learn.


WELCOME



Activity: Logistic Regression and Confusion Matrix Warm-Up

In this activity, you will apply logistic regression to predict whether a specified breast tumor is benign or malignant by using computed features from digitized images. You'll then create the associated confusion matrix.

Suggested Time:
10 Minutes

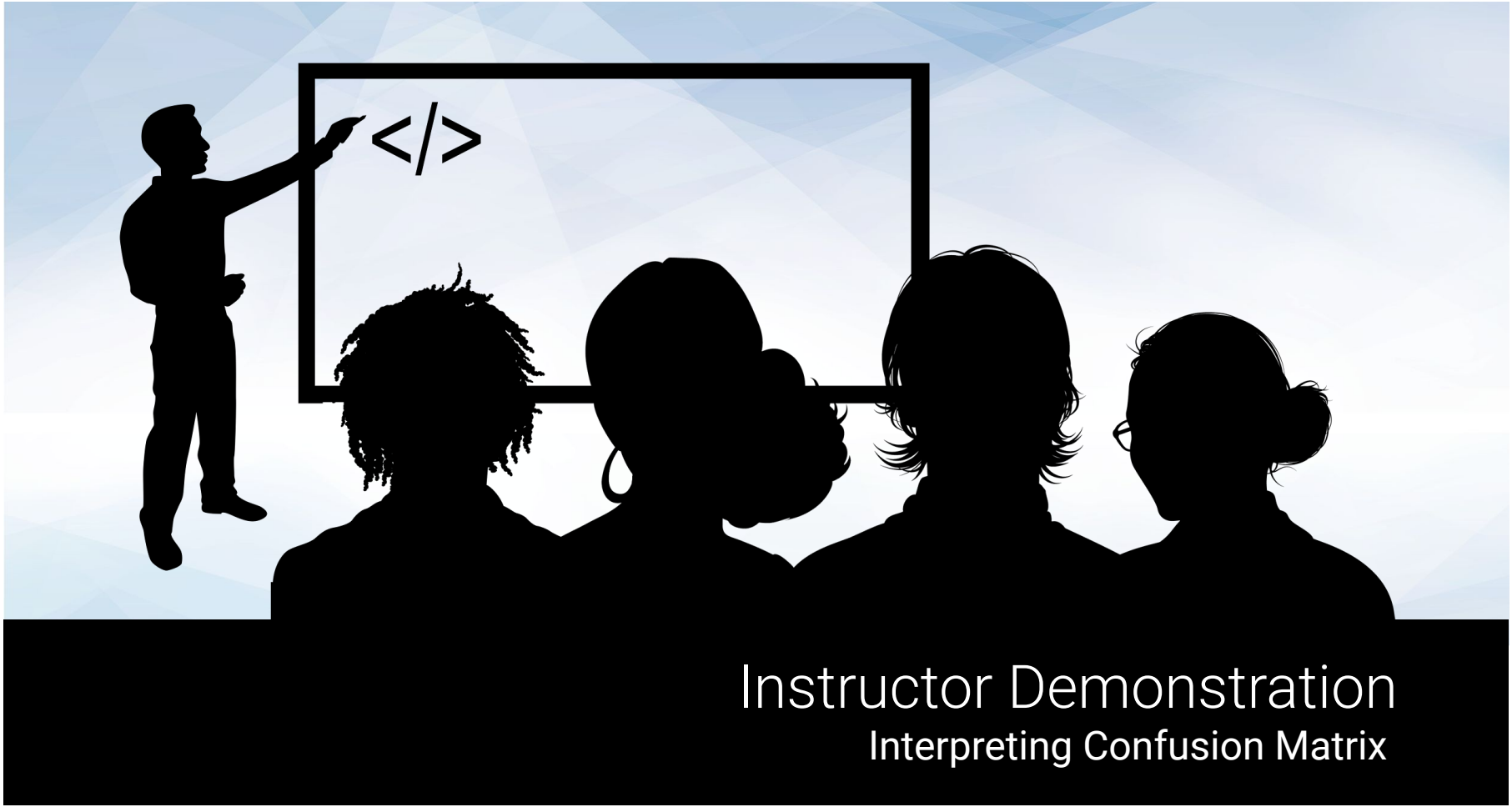
Instructions:

Activity: Logistic Regression and Confusion Matrix Warm-Up

1. Split your data into training and testing data.
2. Create a logistic regression model with Sklearn.
3. Fit the model to the training data.
4. Compute the accuracy score for the testing data and the training data separately.
5. Create a confusion matrix based on the testing dataset and the predicted values.

- **Hints:**
  - You might receive a warning that the solver fails to converge. For now, you can ignore that warning.

Let's Review



Instructor Demonstration

Interpreting Confusion Matrix

Instructor Do: Interpreting Confusion Matrix

- A Confusion Matrix compares the predicted values from a model against the actual values. The entries of the confusion matrix are the number of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

	Predicted True	Predicted False
Actually True	113 (True Positives)	12 (False Negatives)
Actually False	31 (False Positives)	36 (True Negatives

Instructor Do: Interpreting Confusion Matrix

➔ **Accuracy**

Is the percentage of correct predictions.

$$\text{Accuracy} = TP + TN / (TP + TN + FP + FN)$$

Instructor Do: Interpreting Confusion Matrix

➔ **Precision**

Is the percentage of positive predictions that are correct.

$$\text{Precision} = TP / (TP + FP)$$

Instructor Do: Interpreting Confusion Matrix

➔ **Sensitivity/Recall**

Is the percentage of actual positive results that are predicted correctly. Sometimes it is called the **True Positive Rate**.

$$\text{Sensitivity} = TP / (TP + FN)$$

Instructor Do: Interpreting Confusion Matrix

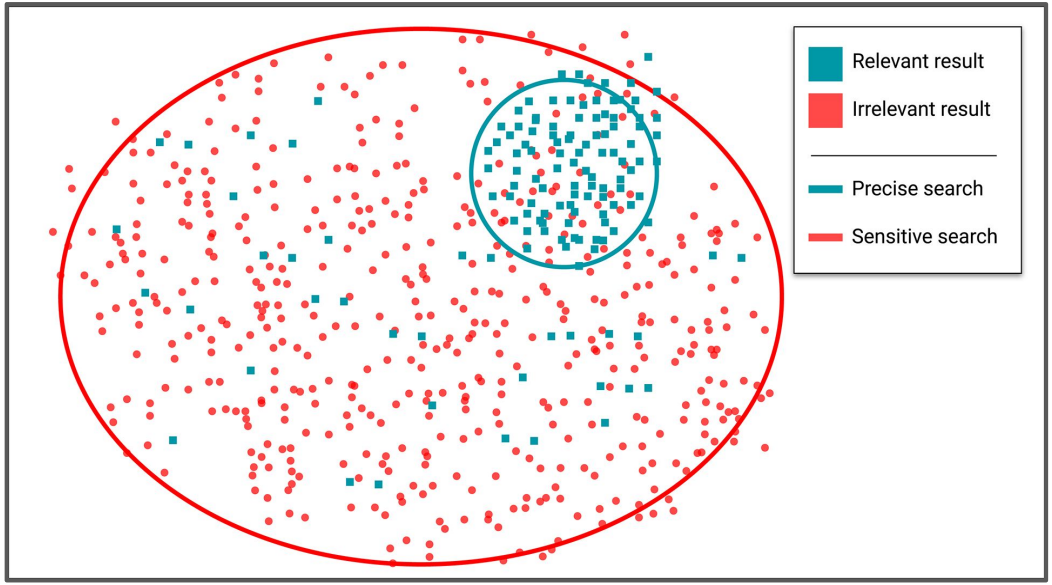
➔ **Precision/Sensitivity Tradeoff**

Precision is a measure of how reliable a positive classification is.

Sensitivity is a measure of how many observations with a positive condition will be correctly diagnosed.

Can you think of situations where precision would be more important? What about when sensitivity is more important?

Sometimes precision is better, sometimes sensitivity is better.



Instructor Do: Interpreting Confusion Matrix

➔ **F1 Score**

Also called the harmonic mean, balances **Precision** and **Sensitivity**.

$$F1 = 2(Precision * Sensitivity) / (Precision + Sensitivity)$$

Instructor Do: Interpreting Confusion Matrix

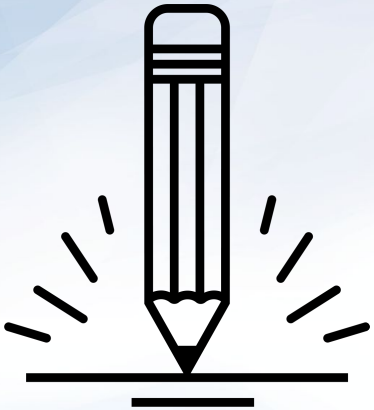
➔ **Recap**

Accuracy = TP + TN / (TP + TN + FP + FN)

Sensitivity = TP / (TP + FN)

Precision = TP / (TP + FP)

F1 = 2(Precision * Sensitivity) / (Precision + Sensitivity)



Activity: Interpret a Confusion Matrix

In this activity, you will create a logistic regression model to predict diabetes for the Pima Diabetes dataset. You'll then interpret the confusion matrix that the model produces.

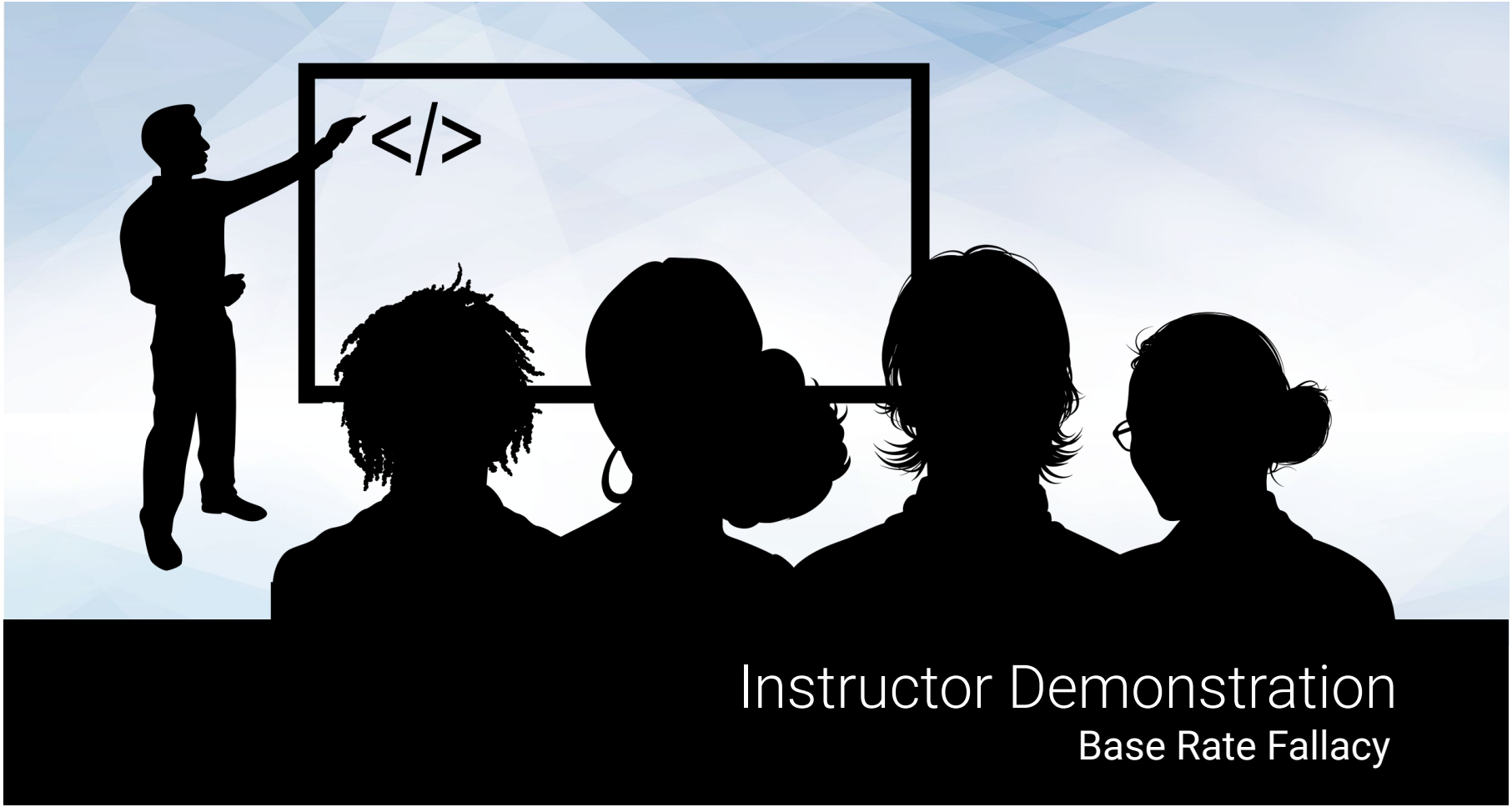
Suggested Time:
15 Minutes

Instructions:

Activity: Interpret a Confusion Matrix

1. Load the Pima Diabetes dataset, and then split the data into training and testing sets.
2. Fit a logistic regression model to the training set.
3. Create a confusion matrix of the predicted vs. the actual outcomes for the testing set.
4. Manually calculate the precision, sensitivity, and F1 score of the model by using the values from the confusion matrix.
5. Create the classification report, and compare it to your manually calculated scores.
6. Answer the following questions: Is your model more precise or more sensitive? Which is more important for this model?

Let's Review



Instructor Demonstration

Base Rate Fallacy

Instructor Do: Base Rate Fallacy

➔ Imbalanced data and the Base Rate Fallacy.

Even a model with relatively high accuracy can run into problems if used on an imbalanced population.

Imagine that we use our breast cancer detection model (with 95% accuracy) in the real world.

The usefulness of our model will depend on how common malignant tissue samples are.

Let’s consider our model on 1,000 samples tested.

	Predicted Malignant	Predicted Benign
Malignant	36.4% (True Positive)	2.1% (False Negative)
Benign	2.8% (False Positive)	58.7% (True Negative)

Base Rate: 50%

Instructor Do: Base Rate Fallacy

➔ 500 out of every 1,000 tissue samples are actually malignant.

If a given tissue sample has a 50% chance of being malignant (i.e. 500 out of every 1,000 tissue samples), then we would expect our model to still be fairly useful. We would want to reduce the False Negatives, but someone who receives a result of “malignant” could be fairly confident that they received the correct result.

	Predicted Malignant	Predicted Benign
Malignant	473 (True Positive)	27 (False Negative)
Benign	23 (False Positive)	477 (True Negative)

Base Rate: 5%

Instructor Do: Base Rate Fallacy

➔ 50 out of every 1,000 tissue samples are actually malignant.

But in the case where malignant samples are much rarer, we have a much different picture.

Our True Positive rate is 94.5%, so 47 out of the 50 malignant samples will be positively identified. However, the False Positive rate of 4.5% means 43 of the benign samples will also be identified as malignant.

Nearly half of the positive results are false positives!

	Predicted Malignant	Predicted Benign
Malignant	47 (True Positive)	3 (False Negative)
Benign	43 (False Positive)	907 (True Negative)

This is an example of the Base Rate Fallacy

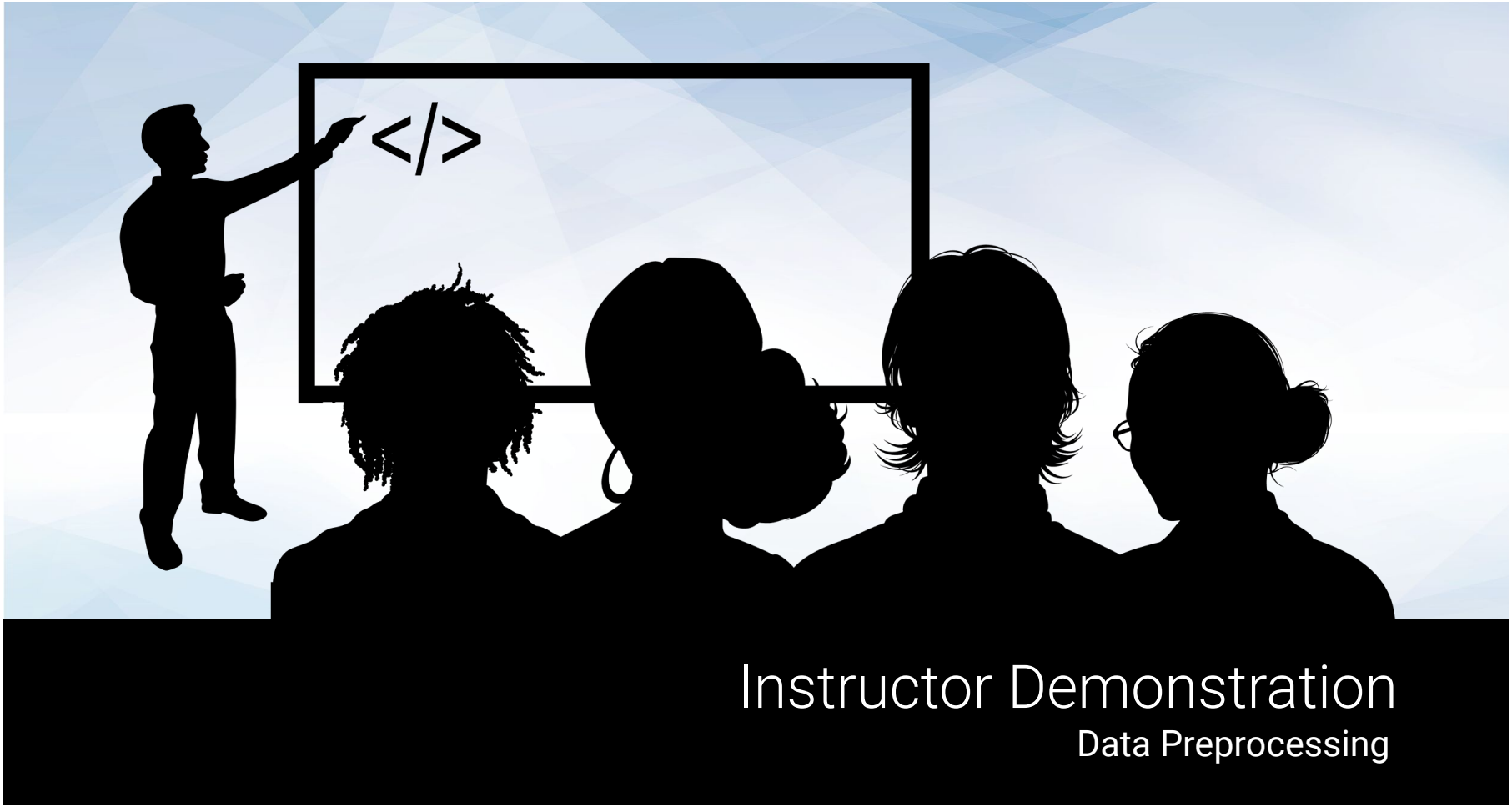
Instructor Do: Base Rate Fallacy

- Specifically, this instance is the **false positive paradox**.
- A seemingly accurate test may have surprising results when trying to detect a relatively rare occurrence.
- This is one application of a result in probability theory called **Bayes Theorem**.
- Bayes Theorem is so important, there's an entire paradigm of statistics based on it: **Bayesian Statistics.**

Countdown timer

15:00

(with alarm)



Instructor Demonstration

Data Preprocessing

Instructor Do: Data Preprocessing

➔ Real life data almost always needs to be processed before it can be used in a machine learning algorithm

- Two major preprocessing steps are **converting categorical data** and **scaling**
- Categorical data is non-numeric data, like the day of the week or a person's education level, and needs to be converted to numeric data.
- Some machine learning algorithms are sensitive to large data values, so features need to be scaled to standardized ranges.

Instructor Do: Data Preprocessing

- **Label Encoding**
  - Turns categorical variables into a series of integers, for example, “Sunday” becomes 0, “Monday” becomes 1, “Tuesday” becomes 2, and so on.
  - It can cause problems though, because the difference between Saturday and Sunday in our previous example is -6, but the difference for other consecutive days is +1.
- **One-Hot Encoding**
  - Instead creates new “dummy” features for each category with 0 and 1 as boolean values. So the *Weekday* feature becomes 6 new features: *isSunday*, *isMonday*, *isTuesday*, *isWednesday*, *isThursday*, and *isFriday*.
  - Why is there no *isSaturday*? Because *isSaturday* can be reconstructed from the other six. This means *isSaturday* is **collinear** with the other dummy features. Including *isSaturday* is an example of “the dummy trap” which can cause errors in the machine learning model.
  - For each observation, only one of the new features will have a value of 1, hence the name “one-hot encoding”.

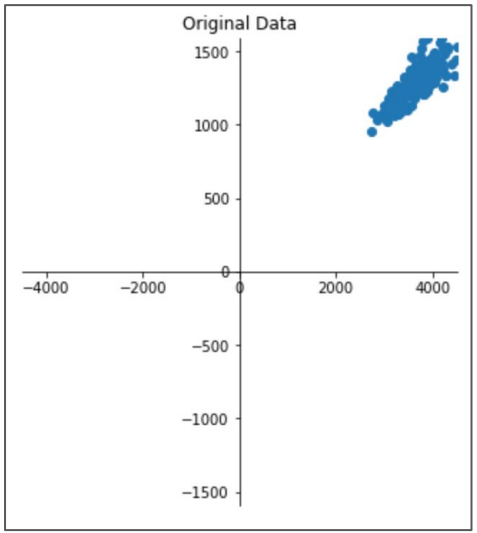
Instructor Do: Data Preprocessing

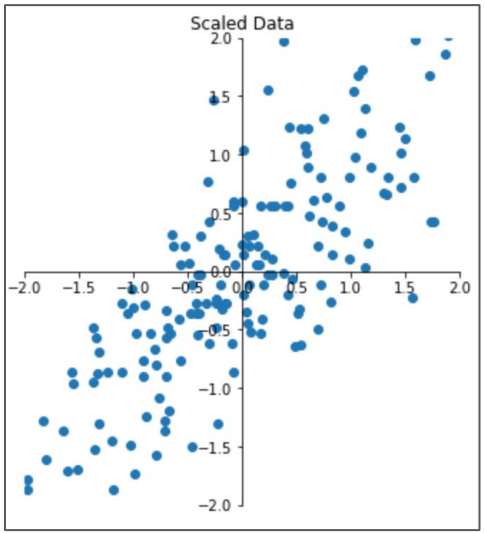
- The `get_dummies()` function.

  - In Pandas, the `get_dummies()` function performs one-hot encoding, and can be applied to an entire DataFrame at once and returns a DataFrame of dummy-coded data.
  - Setting the `drop_first` argument to `True` will automatically avoid the dummy trap.

Instructor Do: Data Preprocessing

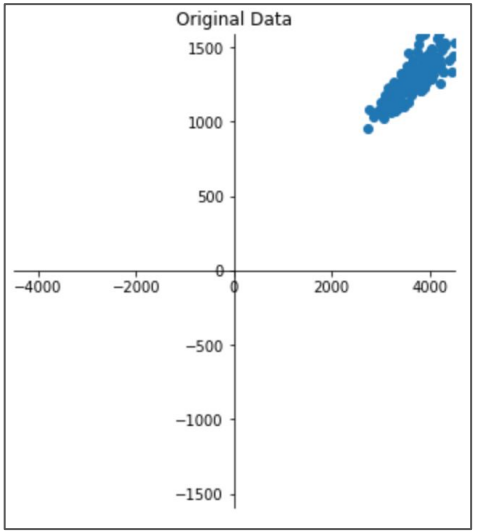
We want all features to be shifted to similar numeric scales so that the magnitude of one feature doesn't bias the model during training. Scikit-learn includes `StandardScaler`, which scales data to have a mean of 0 and variance of 1. `StandardScaler` is recommended when you do not have complete knowledge of your data.

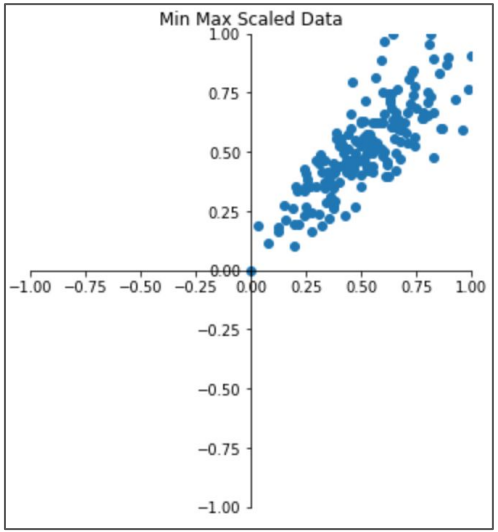




Instructor Do: Data Preprocessing

MinMaxScaler is another scaler available in Sci-Kit Learn, and scales feature data to a minimum of 0 and a maximum of 1.

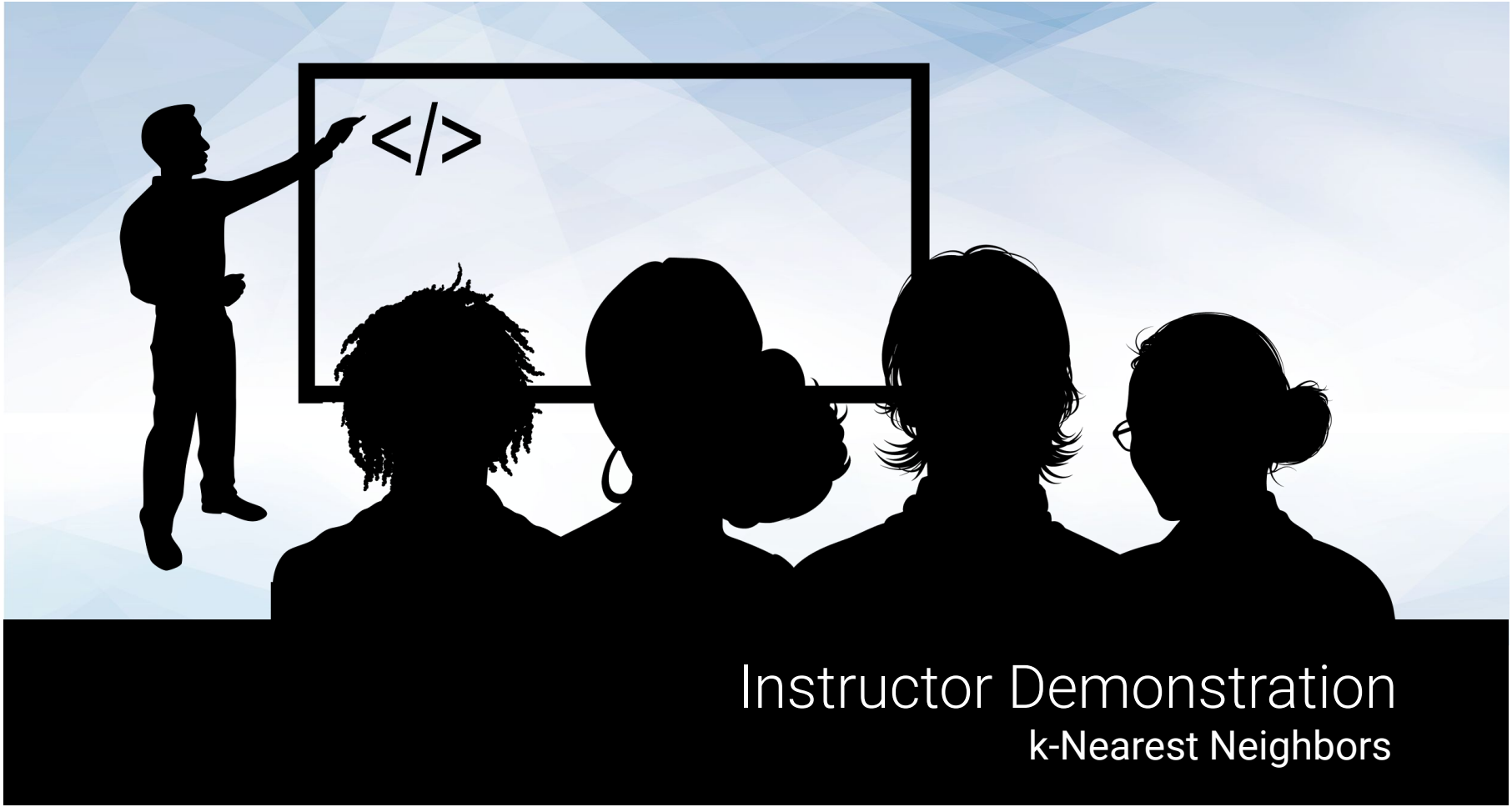




Instructor Do: Data Preprocessing

- Scikit-learn's preprocessing paradigm.

  - Preprocessors in scikit-learn follow the Fit -> Transform paradigm, similar to the Model -> Fit -> Predict paradigm for machine learning.
  - The preprocessor (for example, `StandardScaler`) is fit to training data and then can be used to transform training data, testing data, or data to be predicted by a trained model.
  - **Important:** Make sure you fit preprocessors to training data! If you fit your preprocessors before splitting your data, you are biasing the model with information from the testing set. Remember, the test dataset is supposed to represent new data for the model to predict.

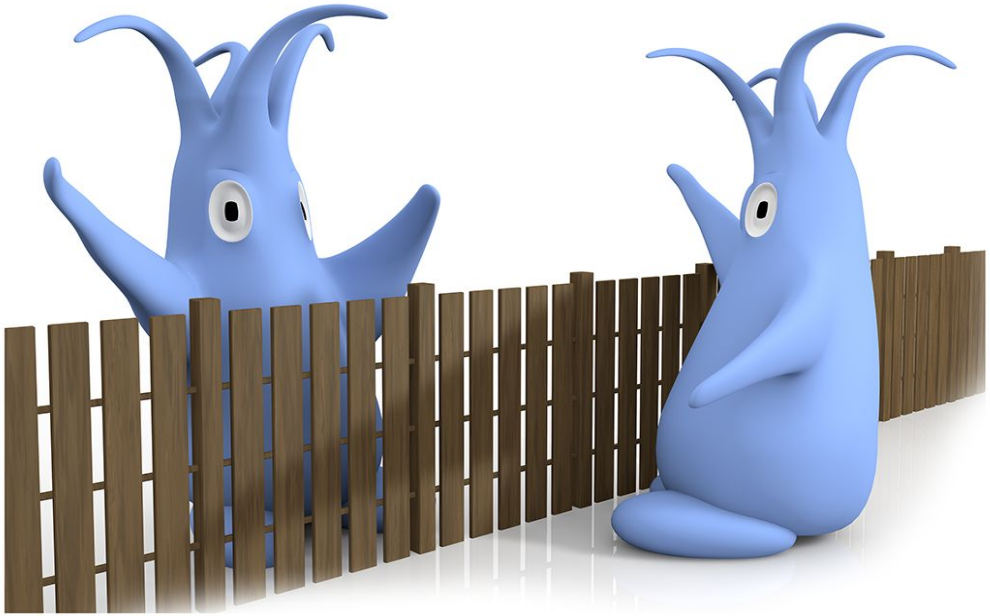


Instructor Demonstration

k-Nearest Neighbors

Instructor Do: KNN

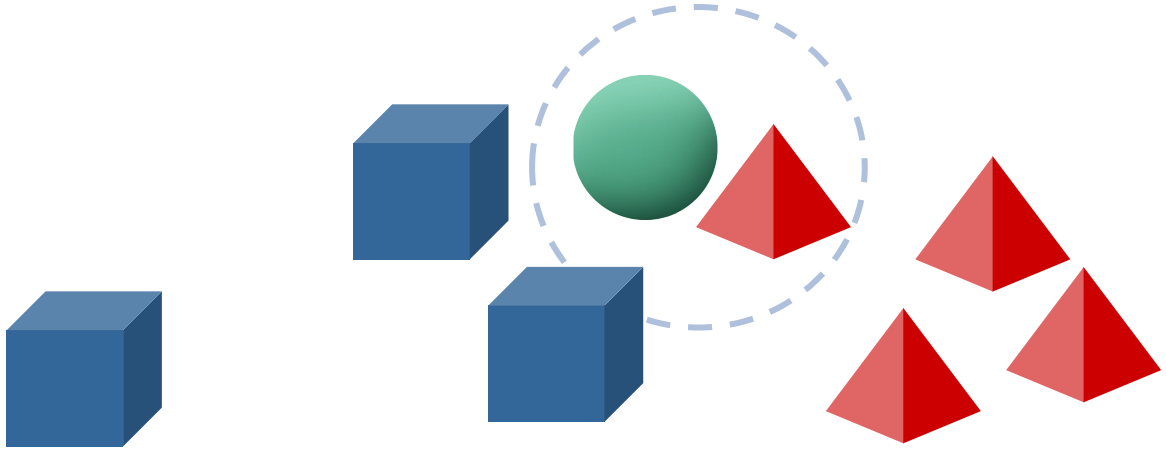
- k-Nearest Neighbors (KNN) is a simple and robust algorithm for classification (and sometimes regression).
- It has many benefits such as outlier insensitivity, ability to classify non-linear data, and high accuracy.
- It does require a lot of memory.



K = 1

Instructor Do: KNN

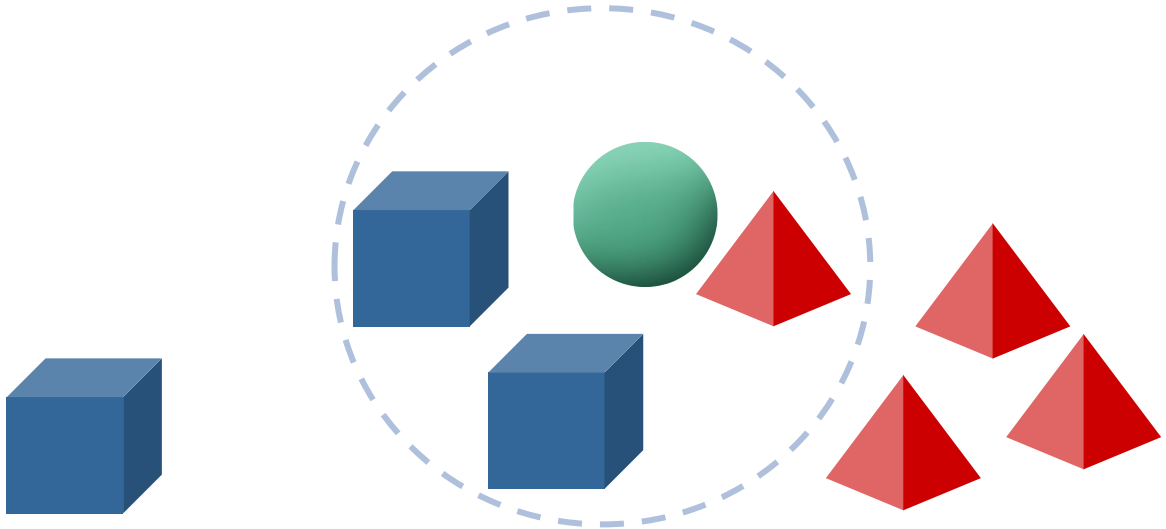
- When **k = 1** , this is simply the nearest neighbor. You find the point nearest to your new data point (the green circle) and that is the class that it will belong to. In this case, the closest neighbor is a red triangle, so the data point will belong to red.



K = 3

Instructor Do: KNN

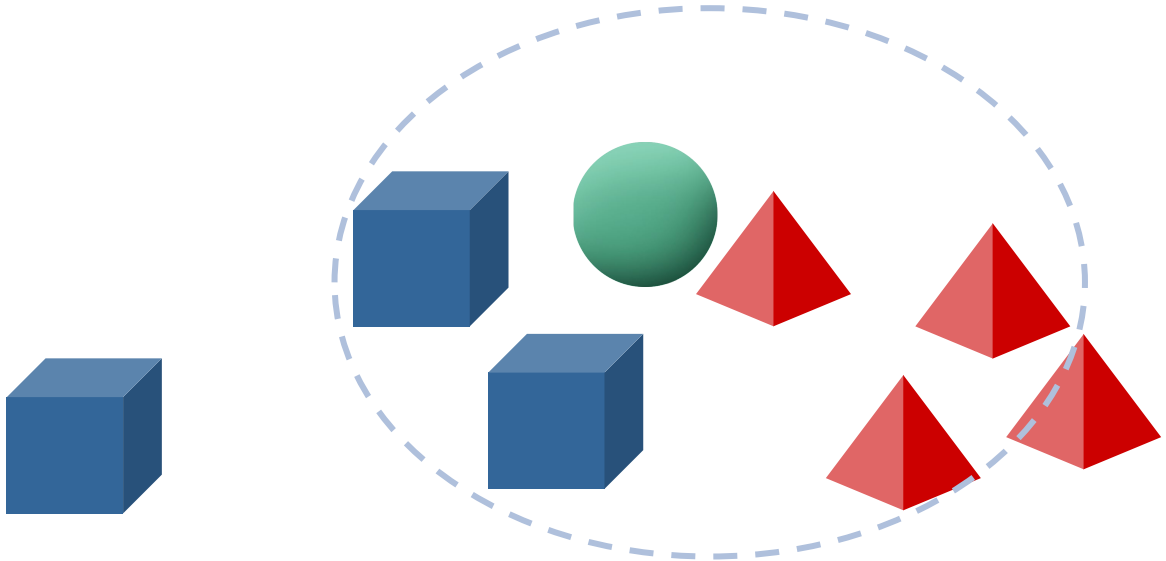
- When **k = 3** , we find the three closest neighbors. In this case, there are two blue squares and one red triangle, so the new data point will be grouped with the blue squares.



K = 5

Instructor Do: KNN

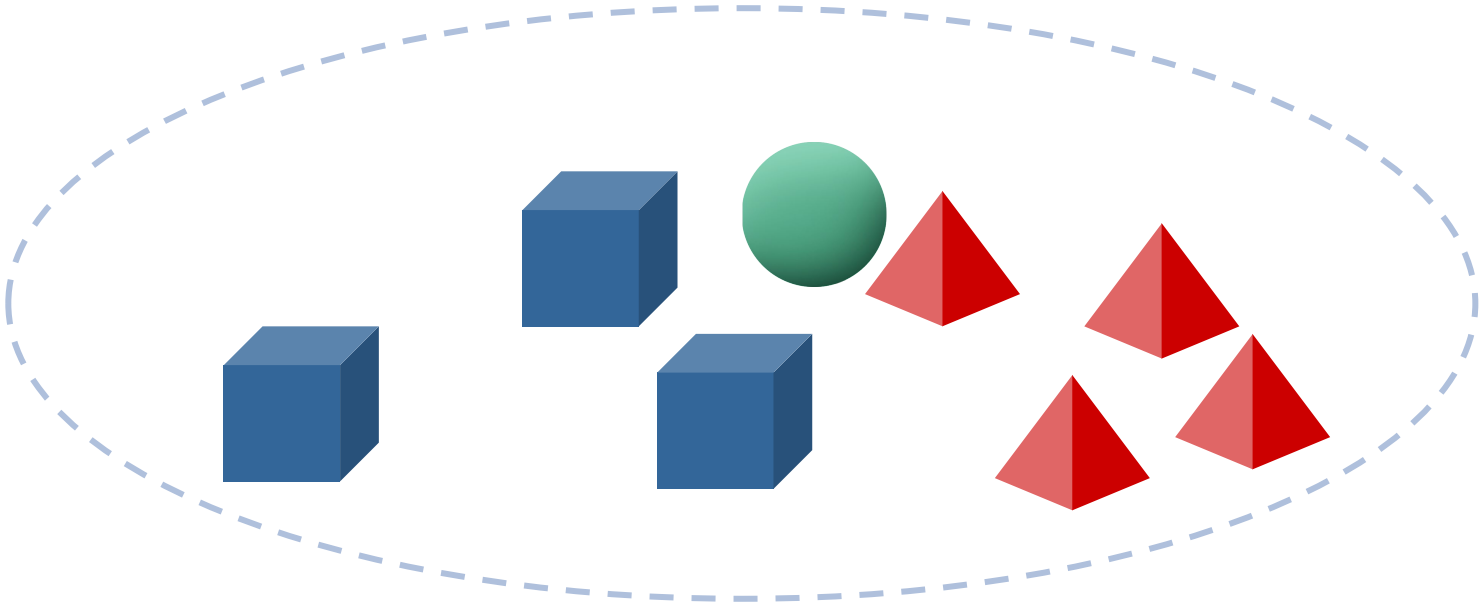
- When **k = 5**, there are three red triangles and two blue squares, so the new data point will belong to the red triangles.



K = 7

Instructor Do: KNN

- Finally, when **k = 7**, the majority are red triangles, so the new data point belongs to red.

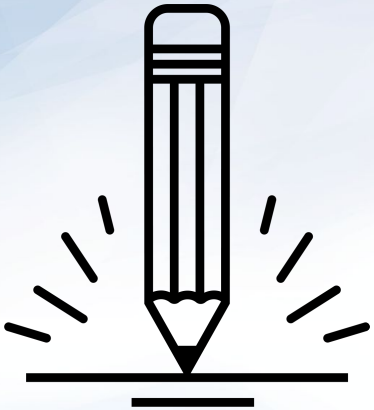


Instructor Do: KNN

- Because k can vary your results, the easiest technique for choosing a k value is to loop through a range of k and calculate the score. Choose the lowest value of k where the score starts to stabilize. **Note**: We only use odd numbers so there are no ties between classes.

```
for k in range(1, 20, 2):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(data, labels)
    score = knn.score(data, labels)
    print(f"K: {k}, Score: {score}")
```

Questions?



Activity: KNN

In this activity, you will determine the best k value in KNN to predict diabetes for the Pima Diabetes dataset.

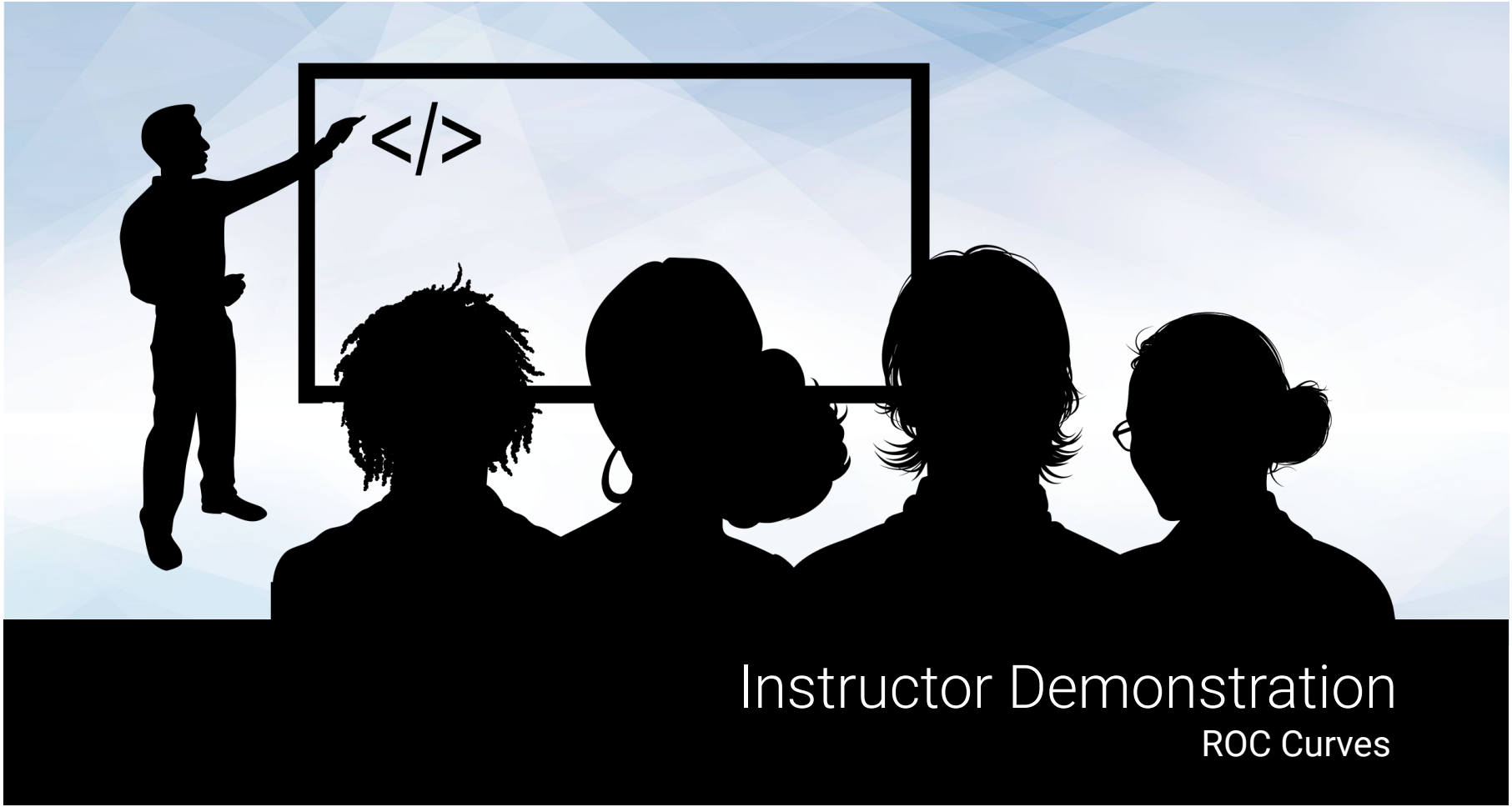
Suggested Time:
15 Minutes

Instructions:

Activity: KNN

1. Calculate the training and testing scores for k ranging from 1 to 20. Use only odd numbers for the k values.
2. Plot the k values for both the training and the testing data to determine where the best combination of scores occurs. This point has the optimal k value for your model.
3. Retrain your model by using the k value that you found to have the best scores. Print the score for this value.

Let's Review



Instructor Demonstration

ROC Curves

Instructor Do: ROC Curves

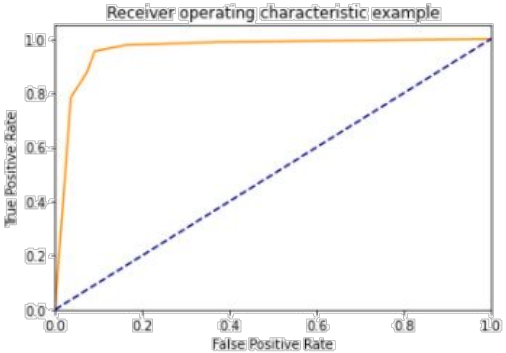
- Is a useful tool when predicting the probability of a binary outcome.
- It plots the false alarm rate versus the hit rate. False positive rate (x-axis) vs. the true positive rate (y-axis) for different candidates threshold values between 0.0 and 1.0.
- **Note: the true positive rate is also known as sensitivity and the false negative rate as specificity.**

Sensitivity = TP / (TP + FN)

vs.

Specificity = TN / (TN + FP)

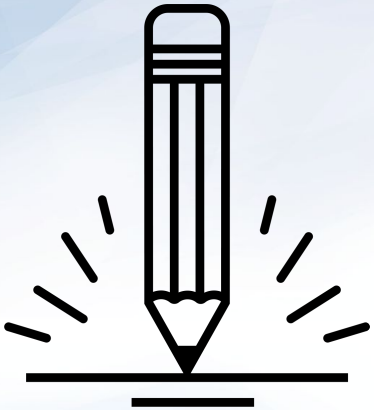
=



Why is ROC curves useful?

Instructor Do: ROC Curves

- Either in general or for different thresholds curves of different models can be directly compared.
- The area under the curve, referred as **AUC** can be used as a summary of the model skill.
- The shape of curve conveys useful information, such as the expected false positive rate, and the false negative rate.



Activity: ROC Curves

In this activity, you will create a KNN model to predict heart disease from the UCI Heart Disease dataset. You will then create an ROC curve for the model.

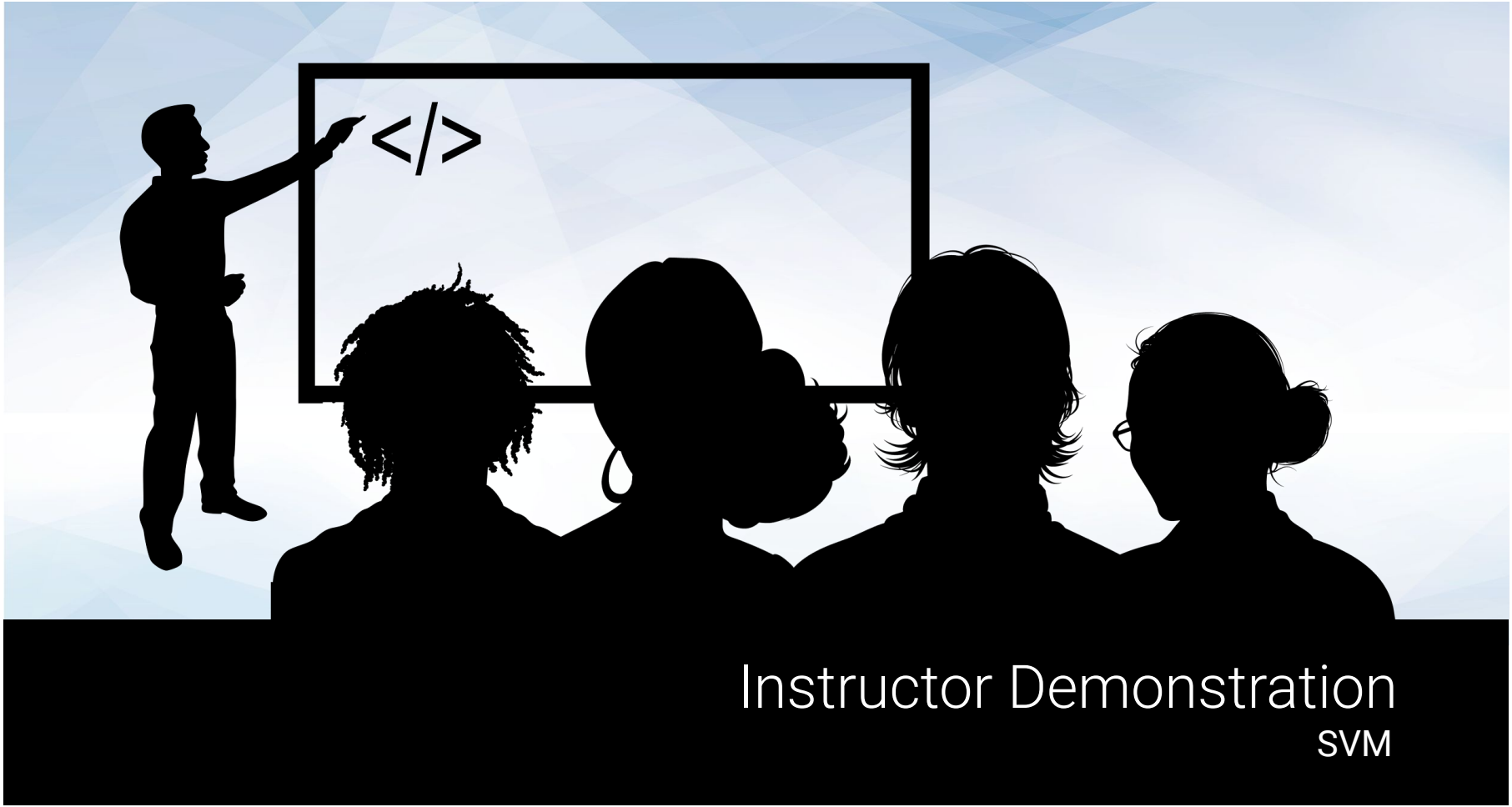
Suggested Time:
15 Minutes

Instructions:

Activity: ROC Curves

1. Load the UCI Heart Disease dataset, and then split the data into training and testing sets.
2. Find an optimal value of k for a KNN model.
3. Print the confusion matrix and the classification report for your model.
4. Create a plot of the ROC curve for your model.

Let's Review

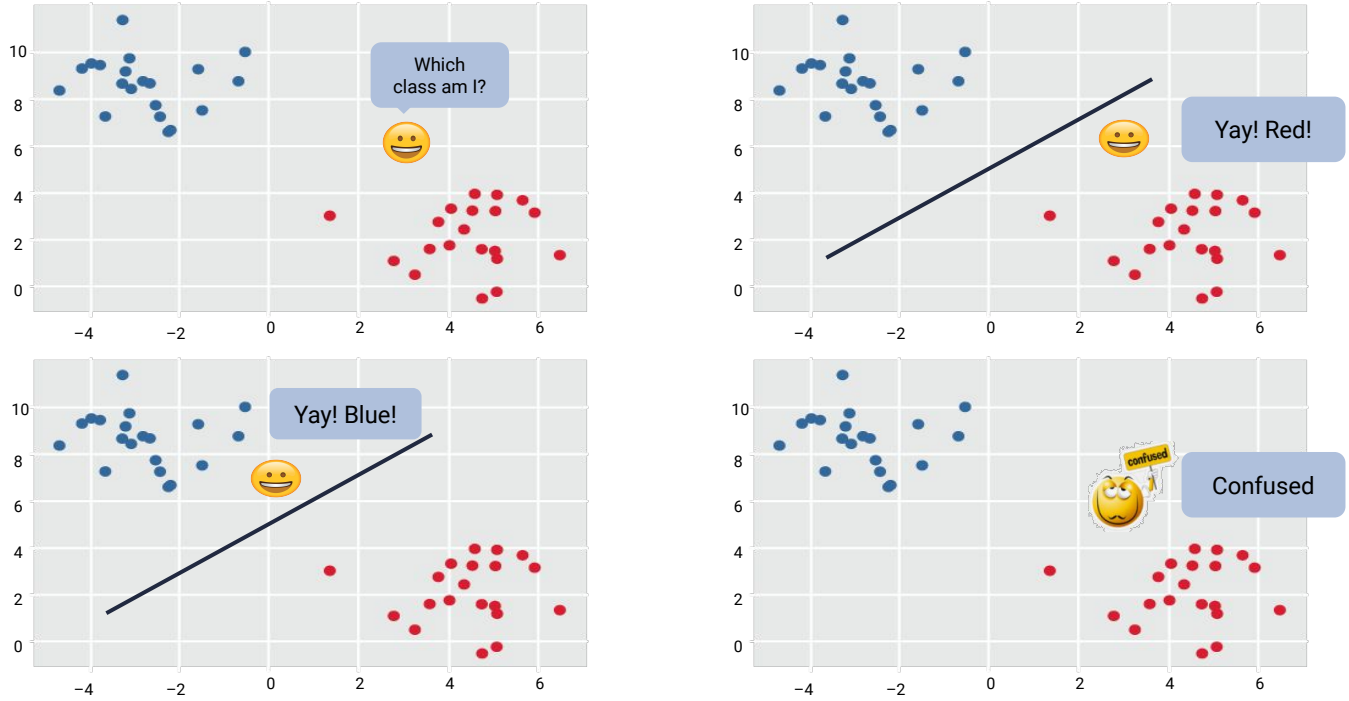


Instructor Demonstration

SVM

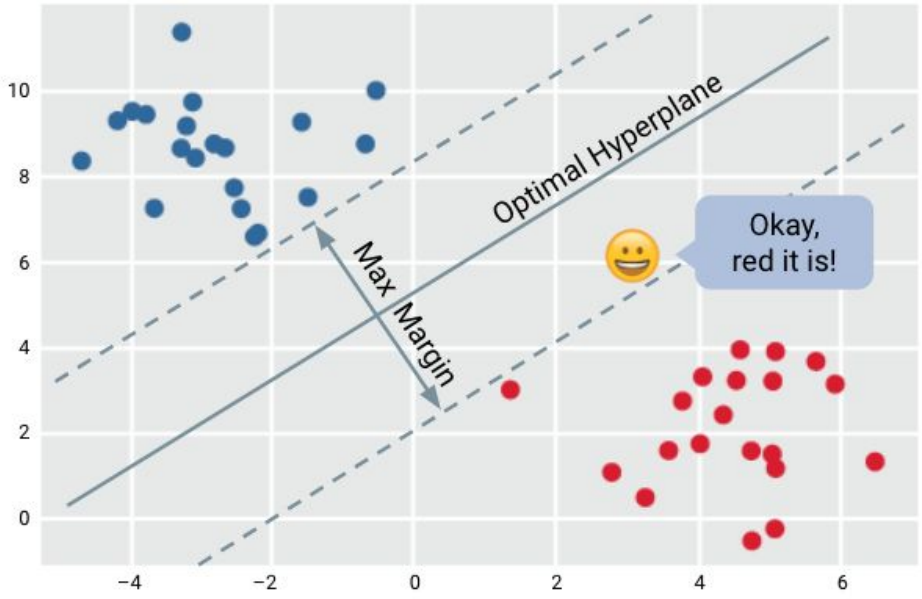
Instructor Do: SVM

- Linear classifiers attempt to draw a line that separates the data, but which line best separates the groups?

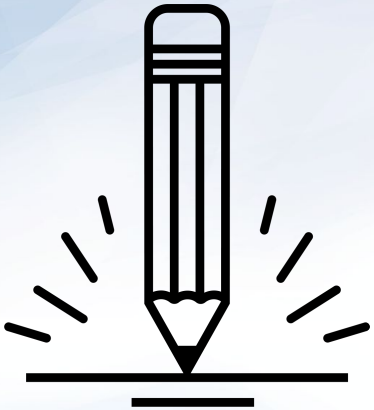


Instructor Do: SVM

- The Support Vector Machines (SVM) algorithm finds the optimal hyperplane that separates the data points with the largest margin possible.



Questions?



Activity: SVM

In this activity, you will apply a support vector machine classifier to predict diabetes for the Pima Diabetes dataset.

Suggested Time:
15 Minutes

Instructions:

Activity: SVM

1. Import a support vector machine classifier, and fit the model to the data.
2. Compute the classification report for this model by using the testing data.

Let's Review