Berkeley Engineering | BerkeleyHaas

PROFESSIONAL CERTIFICATE IN MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE

Module 12
Classification and k-Nearest Neighbors

Office Hours with Viviana Márquez
November 30, 2023

AGENDA

- Required activities for Module 12
- Content review Module 12: Classification and k-Nearest Neighbors
- Code examples from the industry
- Questions

AGENDA

- Required activities for Module 12
- Content review Module 12: Classification and k-Nearest Neighbors
- Code examples from the industry
- Questions

Required Activities for Module 12

- 🤖 Codio Activity 12.1: Introduction to K-Nearest Neighbors
- 🤖 Codio Activity 12.2: Identifying the Best K
- 🤖 Codio Activity 12.3: Decision Boundaries
- 🤖 Codio Activity 12.4: Accuracy, Precision, and Recall
- 🤖 Codio Activity 12.5: Confusion Matrices and Metrics for Classification
- 🤖 Codio Activity 12.6: Evaluation Curves: Precision vs. Recall and ROC
- 🤖 Codio Activity 12.7: KNN for Regression and Imputation
- 🤖 Quiz 12.1: Classification and K-Nearest Neighbors
- Try-It 12.1: Choosing the Right Metric

Berkeley Engineering | BerkeleyHaas

AGENDA

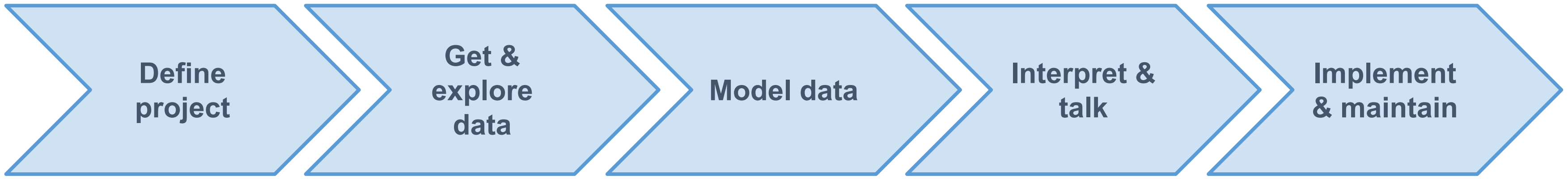
- ✅ Required activities for Module 12
- Content review Module 12: Classification and k-Nearest Neighbors
- Code examples from the industry
- Questions

Content review Module 12: Classification and k-Nearest Neighbors

- Classification models
- Performance metrics
- K-NN algorithm

Berkeley Engineering | BerkeleyHaas

The Machine Learning pipeline



Define project
- Specify business problem
- Acquire domain knowledge

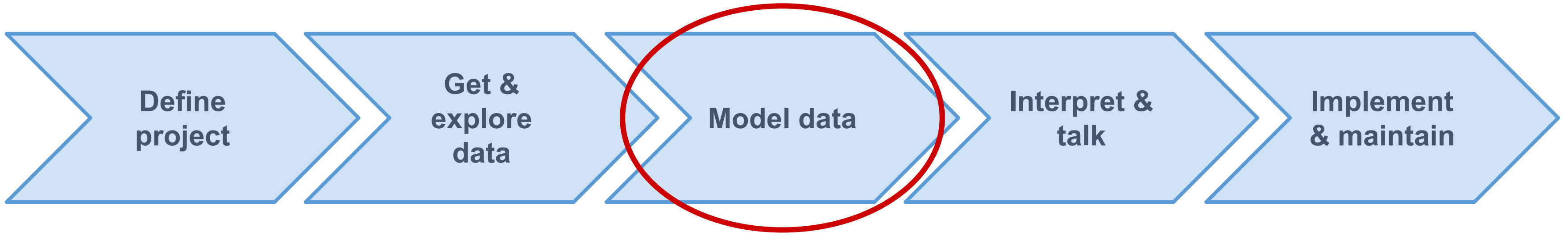
Get and explore data
- Find appropriate data
- Exploratory Data Analysis
- Clean and pre-process data
- Feature engineering

Model data
- Determine ML task
- Build candidate models
- Select model based on performance metrics

Interpret & talk
- Interpret model
- Communicate model insights

Implement & maintain
- Set up function to predict on new data
- Document process
- Monitor and maintain model

The Machine Learning pipeline



Define project
- Specify business problem
- Acquire domain knowledge

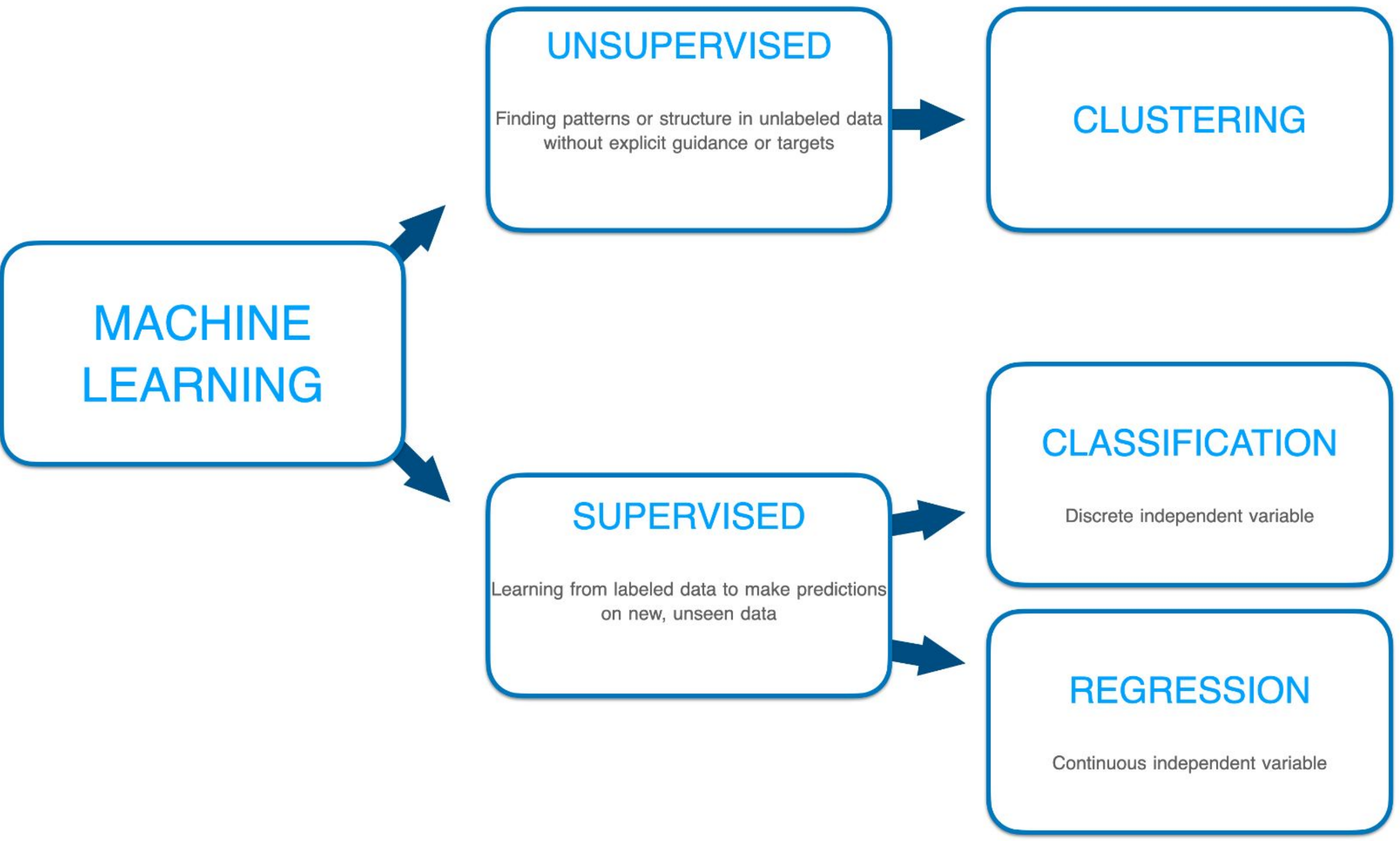
Get and explore data
- Find appropriate data
- Exploratory Data Analysis
- Clean and pre-process data
- Feature engineering

Model data
- Determine ML task
- Build candidate models
- Select model based on performance metrics

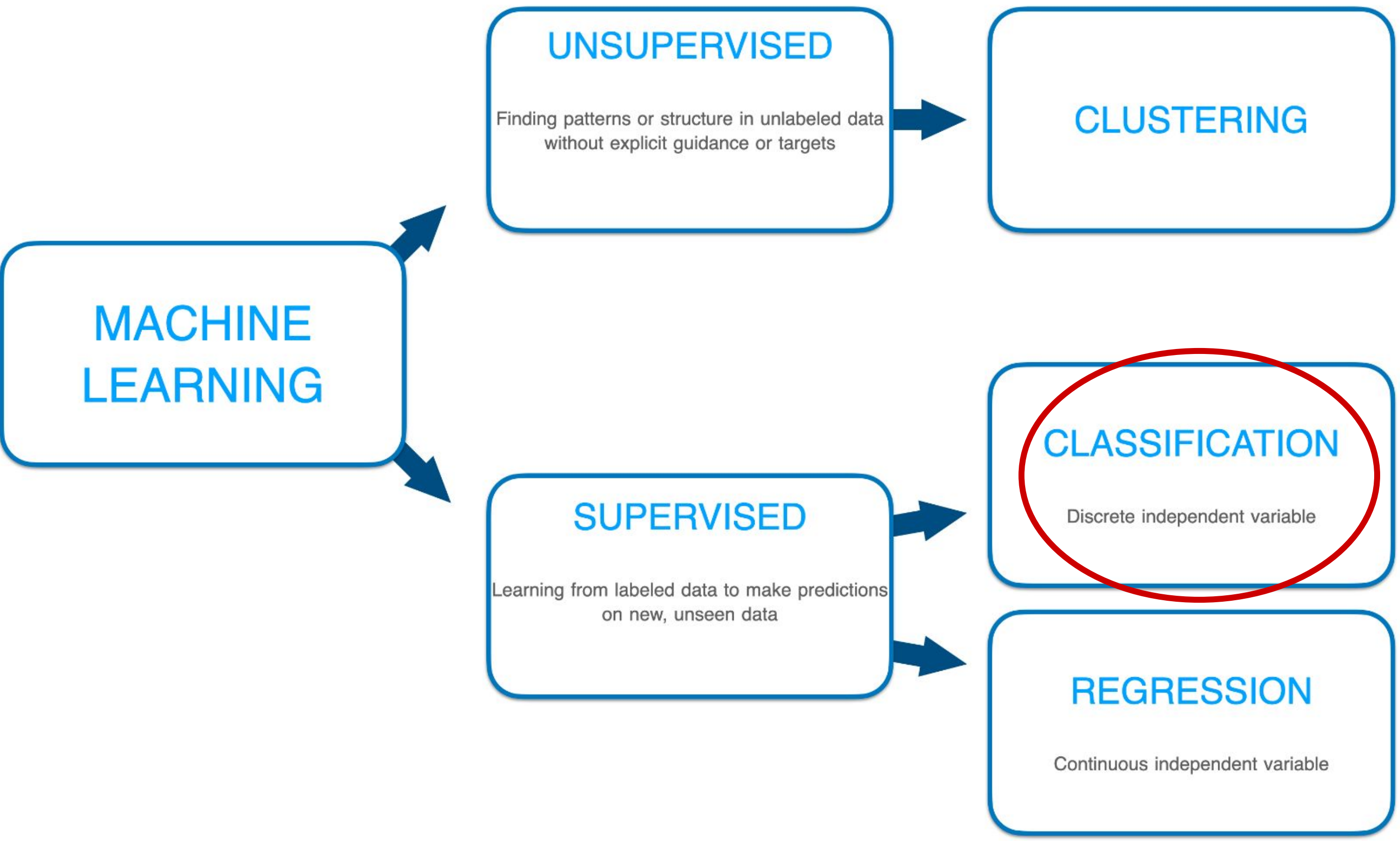
Interpret & talk
- Interpret model
- Communicate model insights

Implement & maintain
- Set up function to predict on new data
- Document process
- Monitor and maintain model

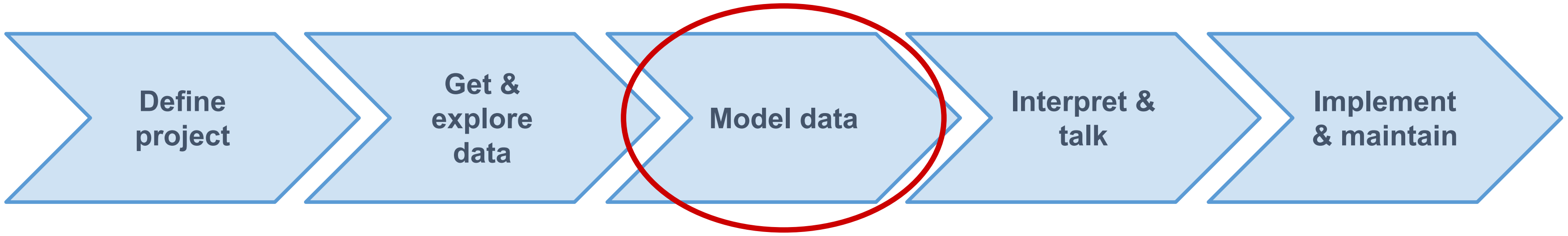
Do we have labels? Is my target variable discrete?


MACHINE LEARNING
UNSUPERVISED
Finding patterns or structure in unlabeled data without explicit guidance or targets
CLUSTERING
SUPERVISED
Learning from labeled data to make predictions on new, unseen data
CLASSIFICATION
Discrete independent variable
REGRESSION
Continuous independent variable

Do we have labels? Is my target variable discrete?


MACHINE LEARNING
UNSUPERVISED
Finding patterns or structure in unlabeled data without explicit guidance or targets
CLUSTERING
SUPERVISED
Learning from labeled data to make predictions on new, unseen data
CLASSIFICATION
Discrete independent variable
REGRESSION
Continuous independent variable

The Machine Learning pipeline



Define project
- Specify business problem
- Acquire domain knowledge

Get and explore data
- Find appropriate data
- Exploratory Data Analysis
- Clean and pre-process data
- Feature engineering

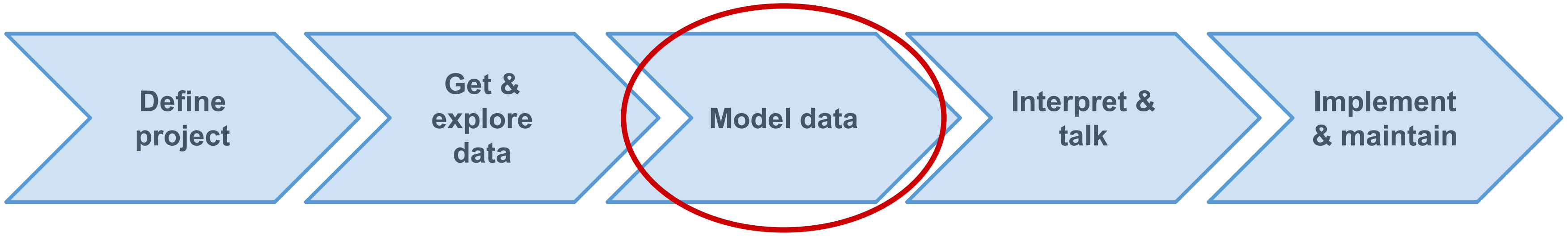
Model data
- Determine ML task
- Build candidate models
- Select model based on performance metrics

Interpret & talk
- Interpret model
- Communicate model insights

Implement & maintain
- Set up function to predict on new data
- Document process
- Monitor and maintain model

Berkeley Engineering | BerkeleyHaas

The Machine Learning pipeline



Define project
- Specify business problem
- Acquire domain knowledge

Get and explore data
- Find appropriate data
- Exploratory Data Analysis
- Clean and pre-process data
- Feature engineering

Model data
- Determine ML task
- Build candidate models
- Select model based on performance metrics

Interpret & talk
- Interpret model
- Communicate model insights

Implement & maintain
- Set up function to predict on new data
- Document process
- Monitor and maintain model

Performance metrics

- Measurements used to evaluate how well a machine learning model is performing.

- The choice of performance metrics depends on the nature of the problem, whether it's a classification, regression, clustering task, and so on.

Performance metrics

- Measurements used to evaluate how well a machine learning model is performing.
- The choice of performance metrics depends on the nature of the problem, whether it's a classification, regression, clustering task, and so on.

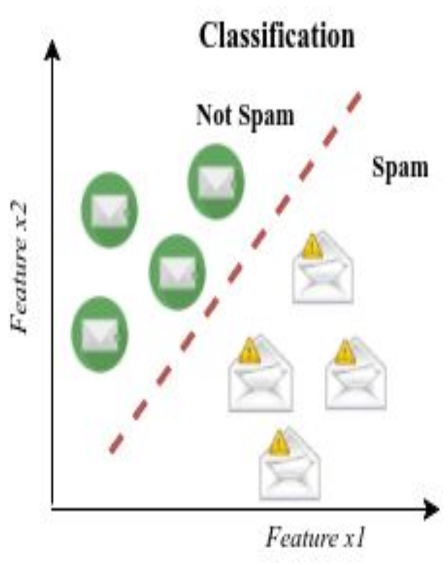
Metrics for regression models	Metrics for classification models
In a regression problem, you're trying to predict a continuous outcome variable (like the price of a house). Performance metrics in regression evaluate the difference between the true and predicted values, often based on errors.	In a classification problem, you're trying to predict which category or class an observation belongs to (like spam vs. non-spam emails). Performance metrics in classification evaluate how well the model can correctly classify the observations.

Performance metrics

- Measurements used to evaluate how well a machine learning model is performing.
- The choice of performance metrics depends on the nature of the problem, whether it's a classification, regression, clustering task, and so on.

Metrics for regression models	Metrics for classification models
• MAE (Mean Absolute Error) • Average Error • MAPE (Mean Absolute Percentage Error) • **RMSE (Root Mean Squared Error)** • SST (Total Sum of Squared Errors) • R-Squared • **Adjusted R-Squared** • and many more… (External link)	• Misclassification rate • **Accuracy** • Sensitivity • Specificity • **Recall** • **Precision** • **F1** • **ROC-AUC** • and many more… (Wiki link)

Performance metrics for classification models



Performance metrics in classification evaluate how well the model can correctly classify the observations.

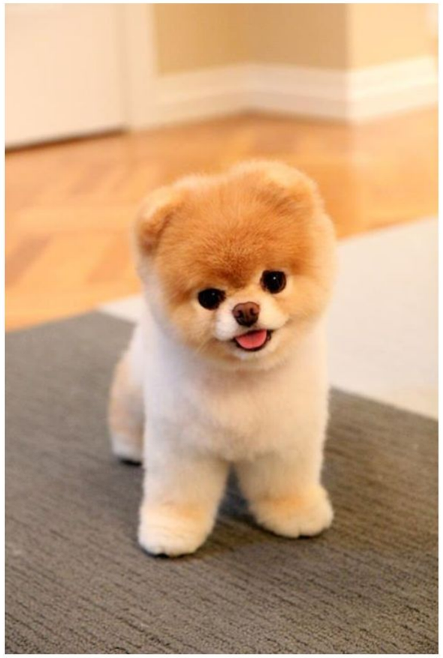
Performance metrics for classification models

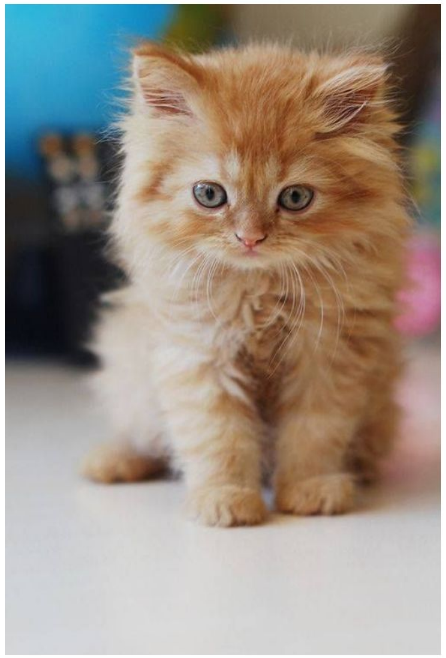
There are three types of classification models:

- Binary
- Multiclass
- Multilabel

Binary classification

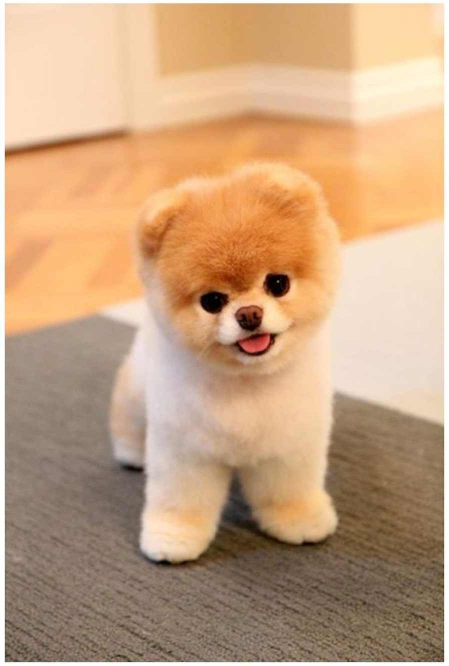
Classify two classes

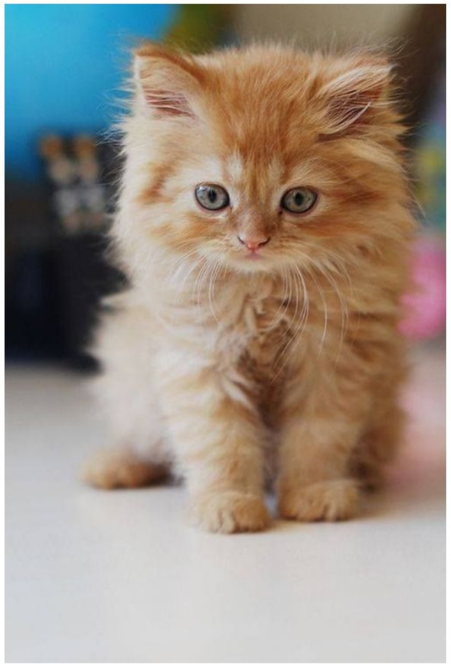




Multiclass classification

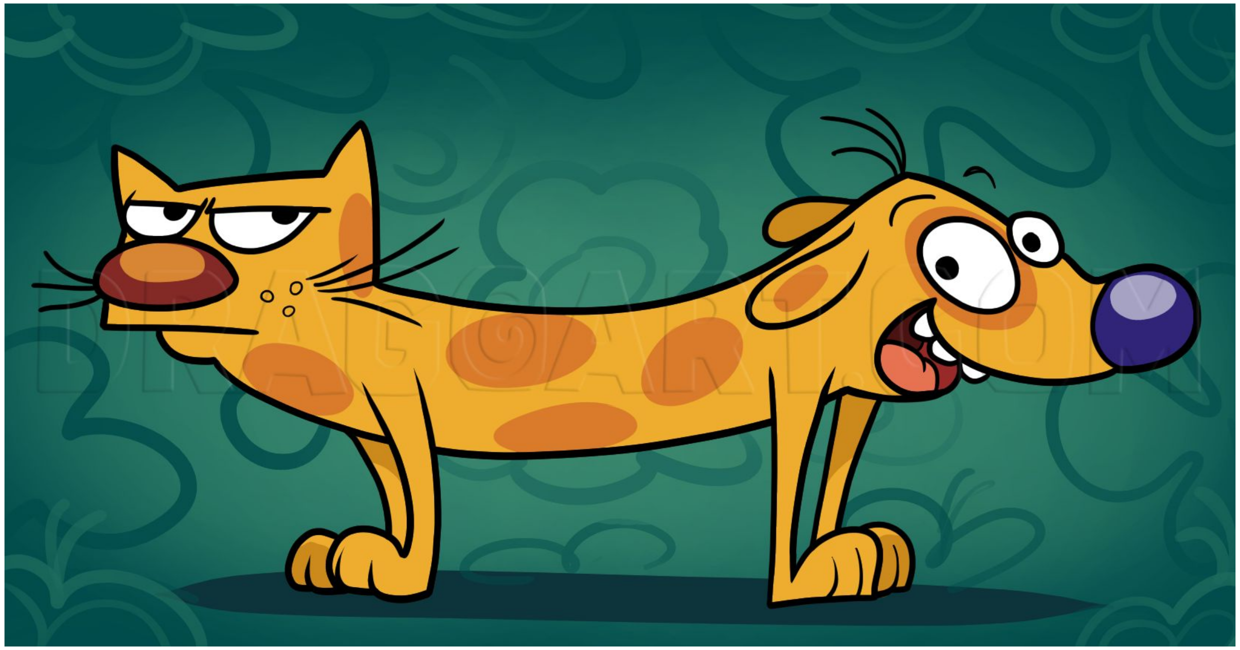
Classify more than two classes





Classification models

Multilabel: When a single observation has more than one label

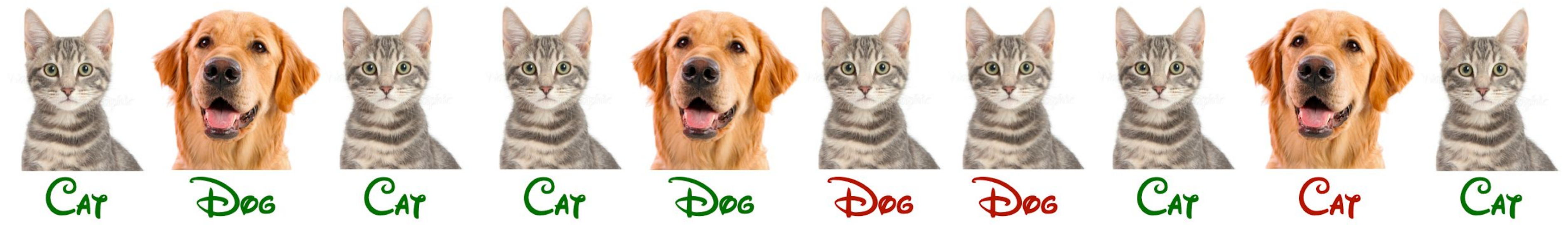


When using a classification model for prediction, you can get four results:

Correct classification class 1: True Positive (TP)	Correct classification class 2: True Negative (TP)	Incorrect classification class 2: False Positive (FP)	Incorrect classification class 1: False Negative (FN)

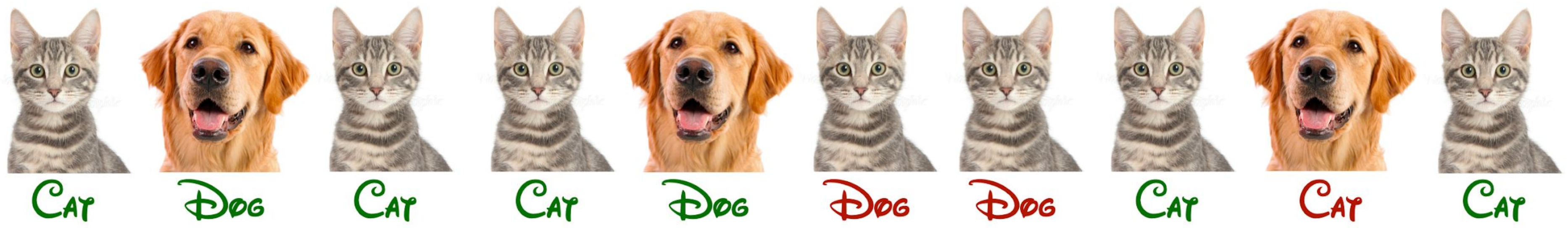
Accuracy

The number of correct predictions divided by the total number of predictions



Accuracy

The number of correct predictions divided by the total number of predictions

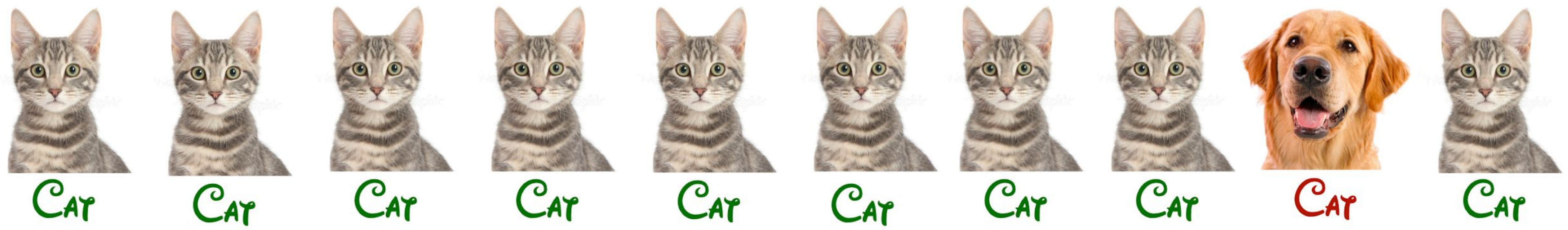


Accuracy 7/10 = 0.7 or 70% accuracy

Issues with accuracy

Issues with accuracy

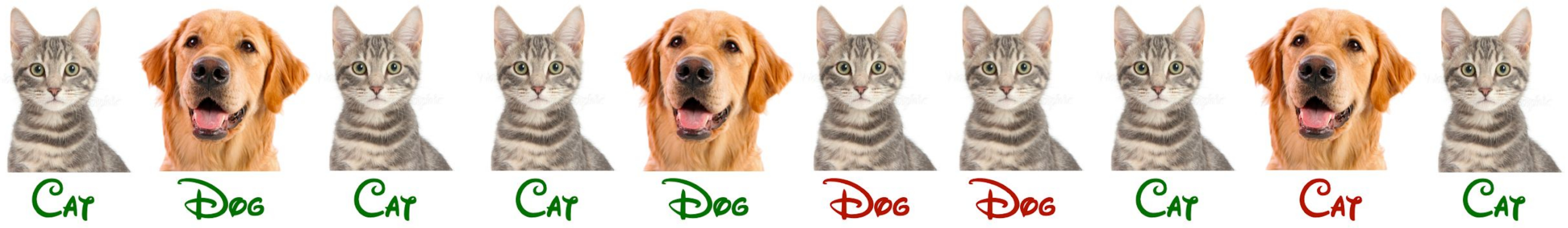
It needs a balanced data set



Accuracy 9/10 = 0.9 or 90% accuracy

Recall

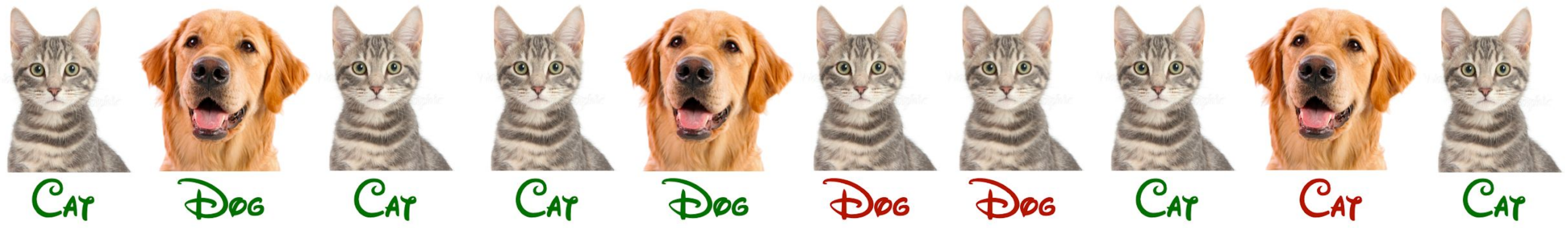
The ability of the model to retrieve **all** relevant cases within a data set



- Let dog be true positive (TP) and cat be true negative (TN)

Recall

The ability of the model to retrieve **all** relevant cases within a data set

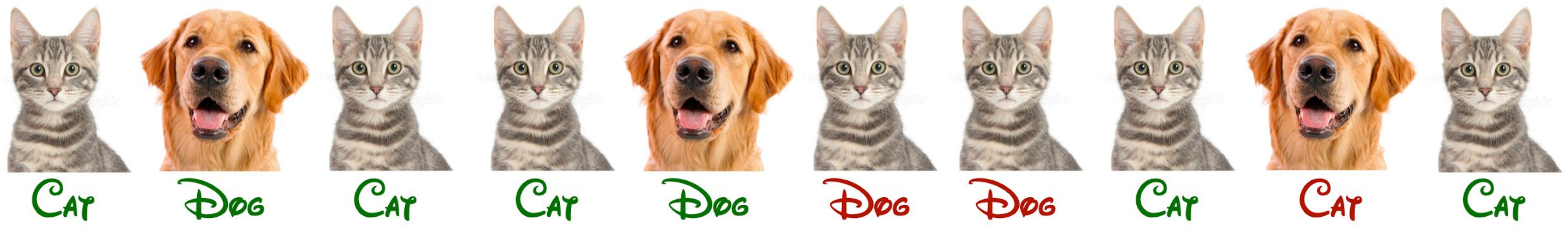


- Let dog be true positive (TP) and cat be true negative (TN)

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{everything that is actually positive}} = \frac{2}{3} = 67\%\ \text{recall}$$

Precision

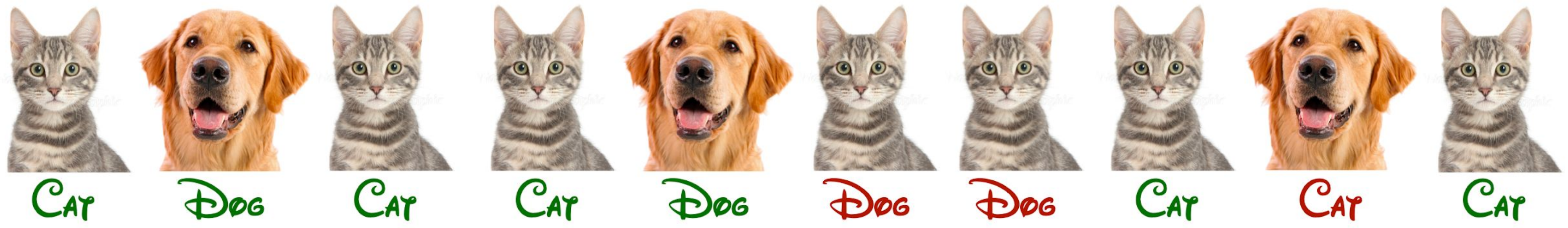
The ability of the model to find **only** the relevant cases within a data set



- Let dog be true positive (TP) and cat be true negative (TN)

Precision

The ability of the model to find **only** the relevant cases within a data set

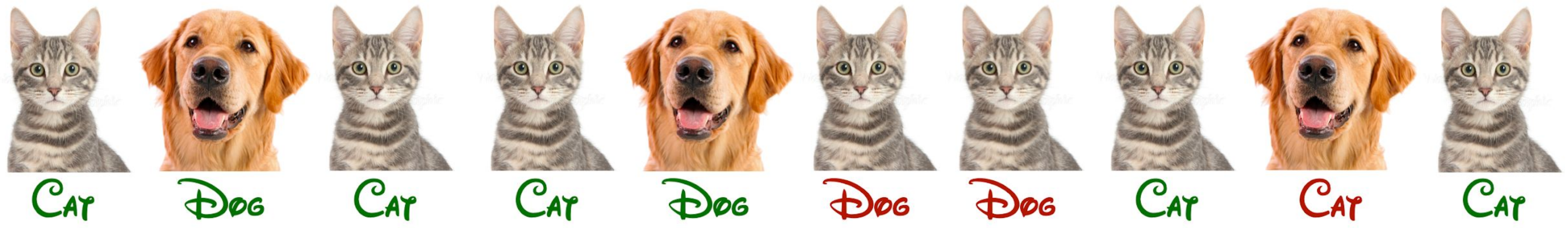


- Let dog be true positive (TP) and cat be true negative (TN)

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{TP}{\text{everything that got classified as positive}} = \frac{2}{4} = 50\% \text{ precision}$$

F1

It is used to find an optimal balance between precision and recall



- Let dog be true positive (TP) and cat be true negative (TN)

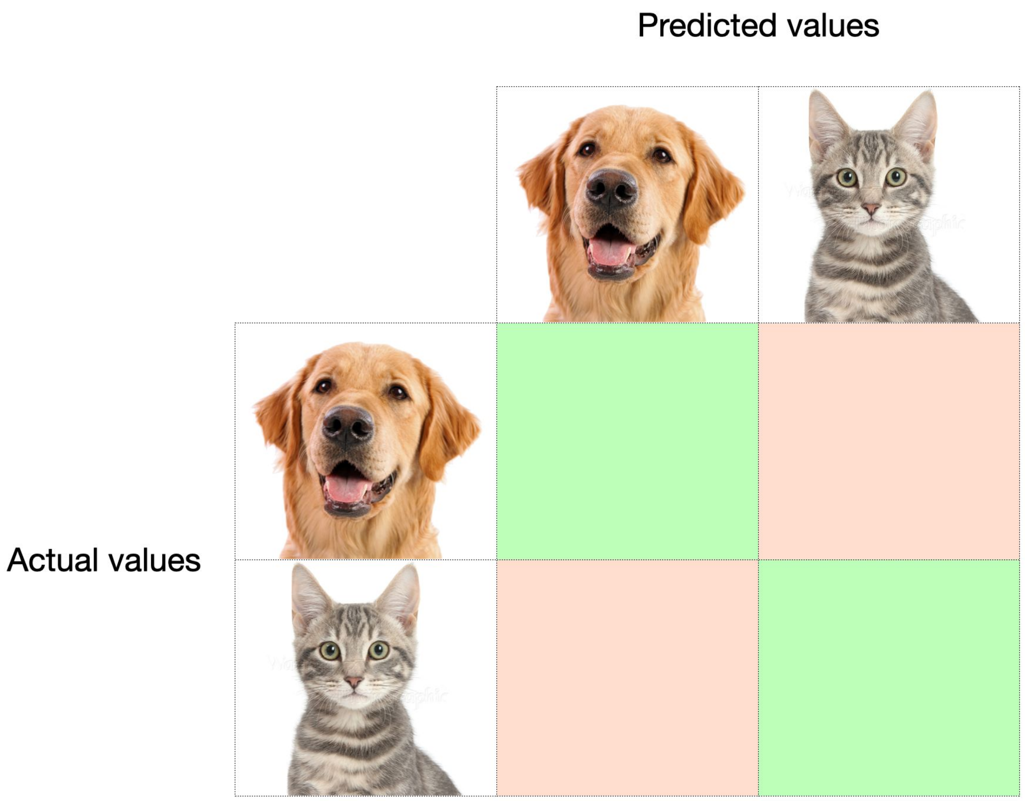
$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = 2 \cdot \frac{(2/3) \cdot (2/4)}{(2/3) + (2/4)} = \frac{4}{7} = 57\%$$

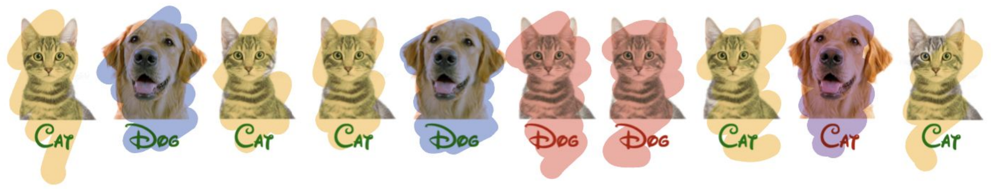
Confusion matrix

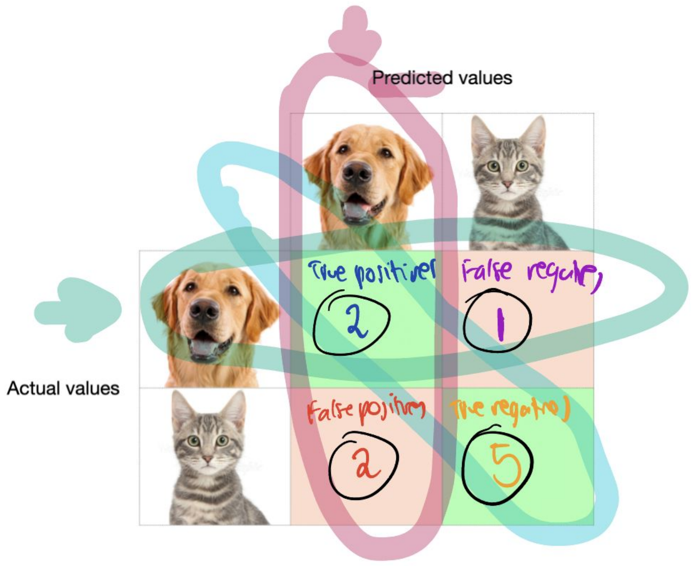
- Tool that helps visualize the performance of a classification model
- The name stems from the fact that it makes it easy to see whether the system is confusing two classes

		Predicted values	
		Positive	Negative
Actual values	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Exercise





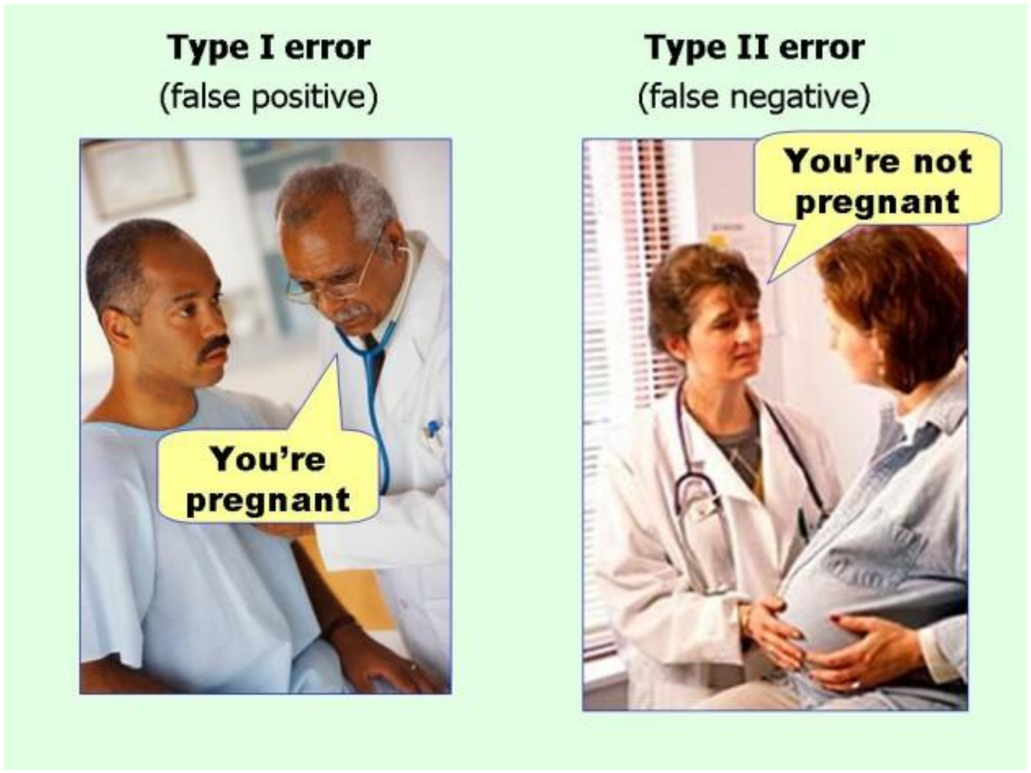


Let's calculate some metrics:

Recall: $\frac{TP}{\text{actually positive}} = \frac{2}{3} = 67\%$.

Precision: $\frac{TP}{\text{classified positive}} = \frac{2}{4} = 50\%$.

Accuracy: $\frac{\text{everything correctly}}{\text{everything}} = \frac{7}{10} = 70\%$.



- The main purpose of the confusion matrix is to obtain measures to compare the predicted values with the true values

- What constitutes a "good" measure depends on the situation

More classification performance metrics

- Terminology and derivations from a confusion matrix: here
- ROC (Receiver Operator Characteristic) graphs and AUC (the area under the curve): here

Summary: performance metrics for classification models

- Cheat Sheet

Metric	Formula	Description	Advantages	Disadvantages	Interpretation
Accuracy	(TP + TN) / (TP + FP + FN + TN)	The proportion of true results among the total number of cases examined.	Easy to interpret.	Can be misleading in imbalanced datasets.	If a model has 90% accuracy, this means that 90 out of 100 predictions are correct.
Precision	TP / (TP + FP)	The proportion of positive identifications that were actually correct.	Useful when the cost of false positives is high.	Not useful when the class distribution is imbalanced.	If the model's precision is 0.75, this means that 75% of the people the model identified as positive cases are actual positive cases.
Recall (Sensitivity)	TP / (TP + FN)	The proportion of actual positives that were correctly identified.	Useful when the cost of false negatives is high.	Not particularly useful when the class distribution is heavily skewed towards negatives.	If the model's recall is 0.8, this means that it was able to find 80% of all positive cases.
Specificity	TN / (TN + FP)	The proportion of actual negatives that were correctly identified.	Useful when the cost of false positives is high.	Not particularly useful when the class distribution is heavily skewed towards positives.	If the model's specificity is 0.7, this means that it correctly identified 70% of all negative cases.
F1 Score	2 * (Precision * Recall) / (Precision +	The harmonic mean of precision and	Useful for balancing precision and recall and for dealing with	Not interpretable as a statistical measure of	An F1 score of 0.7 indicates that the model is fairly good at identifying positive cases without

K-NN 🙅‍♀️ K-means

K-Nearest Neighbors (K-NN)

- Let's imagine you're creating a model to predict rent prices in San Francisco
- 🤔 How would people do it manually?
- Find a few comparable apts and then predict average price
- That's KNN! 💥

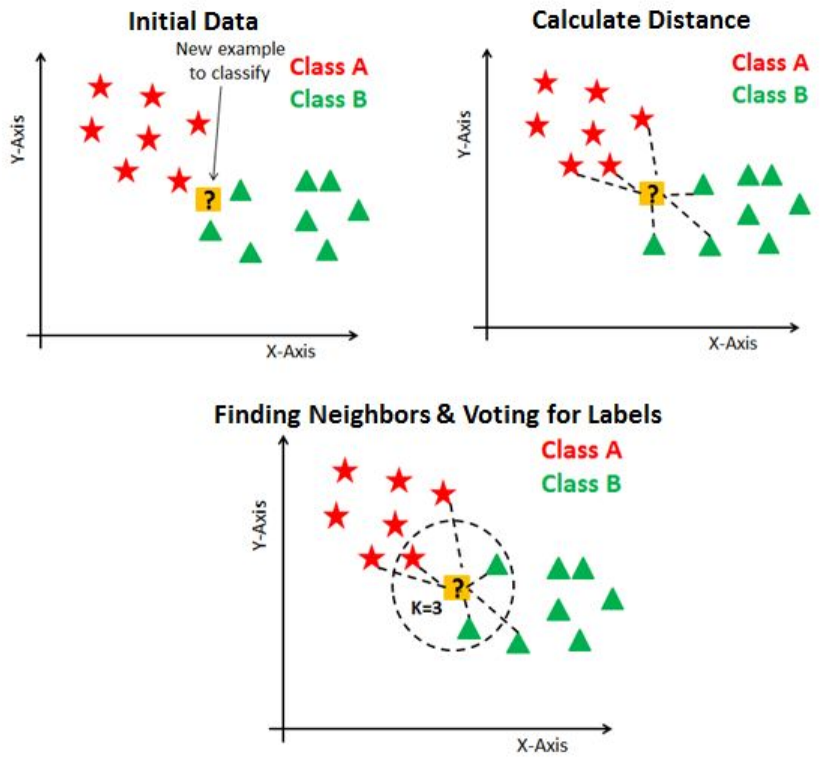
K-Nearest Neighbors (K-NN)

- **Non-parametric SUPERVISED** machine learning model
  - It DOES NOT assume a specific mathematical form and instead attempts to learn the pattern directly from the data
  - Model structure is determined from the data

- Versatile algorithm used for both classification and regression tasks

- Simple yet very useful :)

K-NN steps



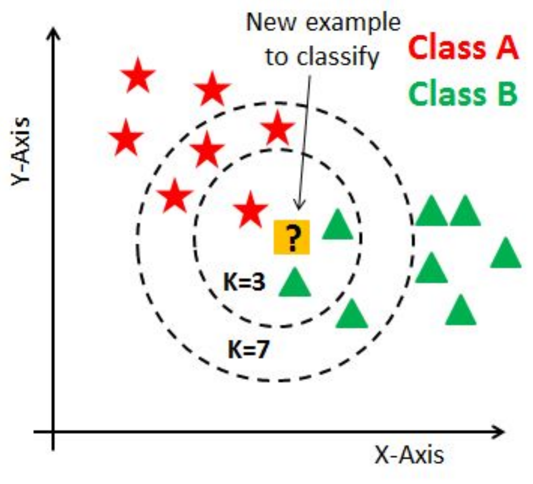
- **Define K**
  Choose the number of neighbors (K)

- **Calculate distance**
  For each instance that you want to predict, calculate the distance between that instance and all instances in the training set.

- **Find Nearest Neighbors**
  Identify the K instances that are nearest to the instance you want to predict

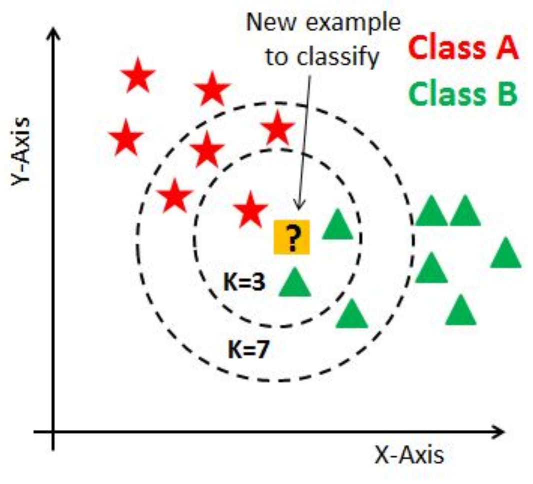
- **Make prediction**
  - **Classification**
    Most common class among its K-NN
  - **Regression**
    Typically the mean of its K-NN

👮‍♀️ Checkpoint # 1

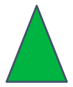


- What would be the label for the new instance if K = 3?

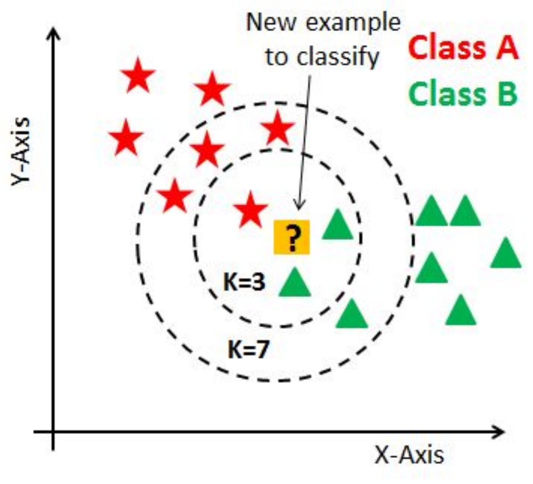
👮‍♀️ Checkpoint # 1



- What would be the label for the new instance if K = 3?

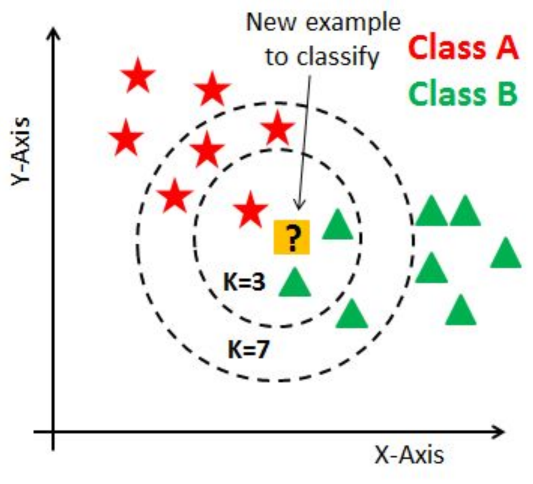


👮‍♀️ Checkpoint # 2



- What would be the label for the new instance if K = 7?

👮‍♀️ Checkpoint # 2



- What would be the label for the new instance if K = 7?

★

Checkpoint # 3

- What's the lowest and the highest possible value of *k*?

👮‍♀️ Checkpoint # 3

- What's the lowest and the highest possible value of *k*?
    - Lowest: *k*=1
    - Highest: *k*=*n* (*n* being the number of observations)

👮‍♀️ Checkpoint # 4

- What happens when $k=n$?
  - The majority class is **always** chosen
  - Underfitting (high bias, low variance)

When should you try K-NN?

- You have:
    - Labels
    - Many instances
    - Few features

- Useful for situations where the data does not follow a known distribution

- Need fast model updates

- Storing and querying the whole training set is easy

When should you try K-NN?

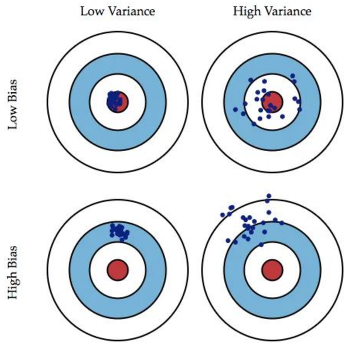
- **Advantages**
  - Simple to understand and explain
  - It can be used for classification and regression
  - Successful where the decision boundary is very irregular

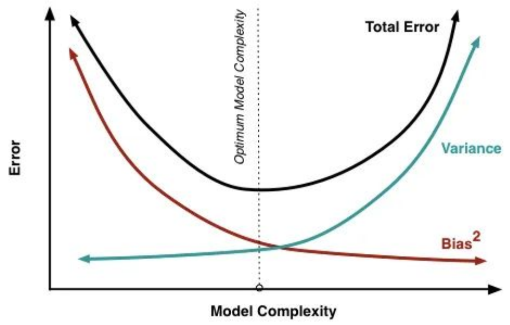
- **Disadvantages**
  - It must store all of the training data
  - It’s prediction phase can be slow when *n* is large
  - Sensitive to irrelevant features and scale of the data
  - Generally not as good as other supervised learning methods

How to pick the best *k*

- There are no predefined methods to find the most favorable value of *k*

- Some strategies:
  - Cross-validation
  - Square root rule: Choose *k* as the square root of the total number of points in your dataset
  - Domain knowledge
  - Derive a plot between the error rate and different *k*s





Resources
- [Library to plot decision boundaries]
- [KNN overview]

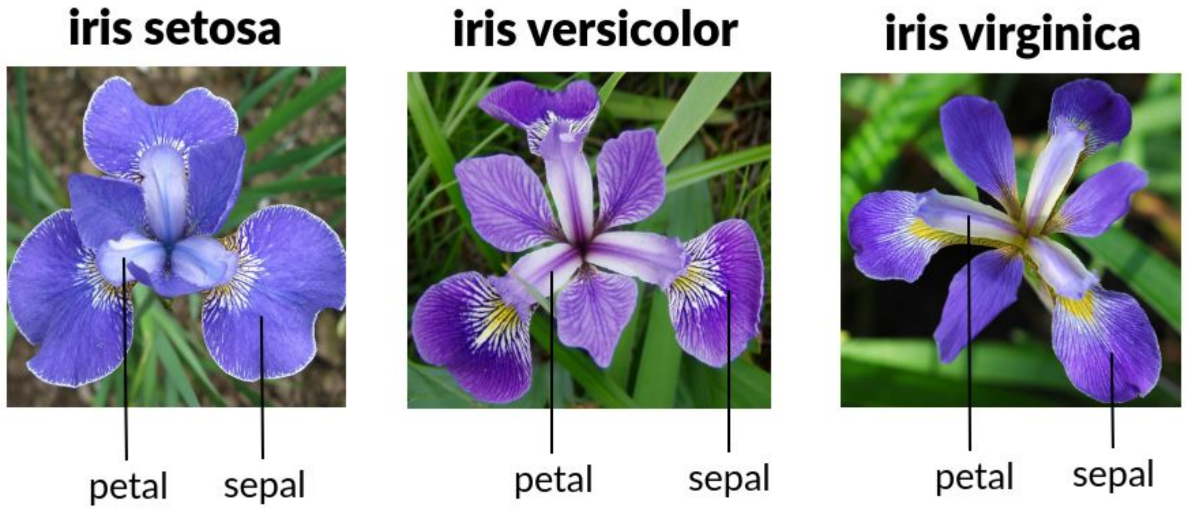
Berkeley Engineering | BerkeleyHaas

AGENDA

- ✅ Required activities for Module 12
- ✅ Content review Module 12: Classification and k-Nearest Neighbors
- Code examples from the industry
- Questions

Code

- https://colab.research.google.com/drive/1vKflTHvexomQv11qErSwE8PyGw_gqGKC?usp=sharing



AGENDA

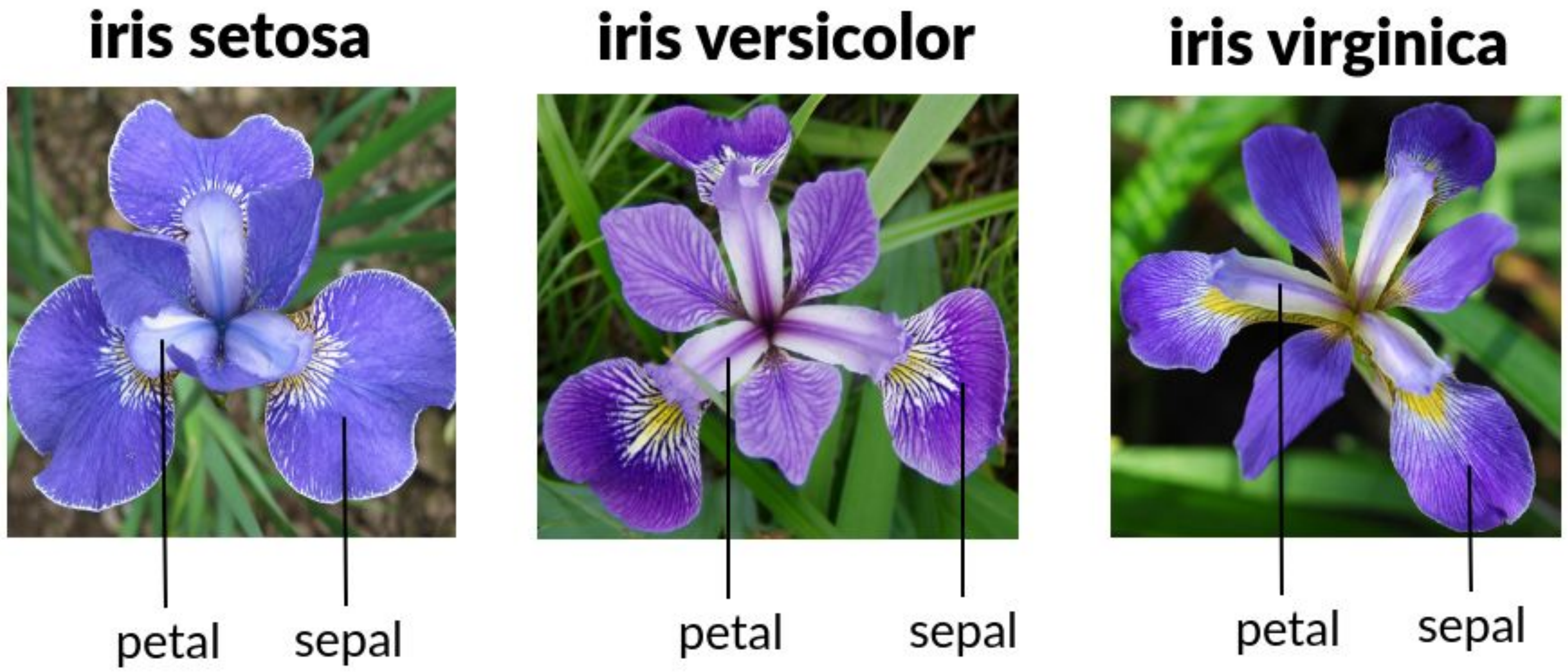
- ✅ Required activities for Module 12
- ✅ Content review Module 12: Classification and k-Nearest Neighbors
- ✅ Code examples from the industry
- Questions

QUESTIONS?

AGENDA

- ✅ Required activities for Module 12
- ✅ Content review Module 12: Classification and k-Nearest Neighbors
- ✅ Code examples from the industry
- ✅ Questions

APPENDIX


iris setosa
petal
sepal
iris versicolor
petal
sepal
iris virginica
petal
sepal

Train vs val dataset

When we train a model, we typically have parameters and hyperparameters.

- **Parameters** are what the model learns from the training data, like the weights in a neural network or the coefficients in a linear regression.
- **Hyperparameters** are higher-level structural settings for our learning algorithm, and they're not learned from the data. These could include the learning rate in a neural network or the depth of a decision tree.

Train vs val dataset

When we train a model, we typically have parameters and hyperparameters.

- **Parameters** are what the model learns from the training data, like the weights in a neural network or the coefficients in a linear regression.
- **Hyperparameters** are higher-level structural settings for our learning algorithm, and they're not learned from the data. These could include the learning rate in a neural network or the depth of a decision tree.

Now, we could try out different hyperparameters and see how they affect the model's performance on the test set. **But here's the issue:** If we tune our hyperparameters based on how well they perform on the test data, we could end up **overfitting** to our test set. That means our model would do really well on our test data, but it wouldn't generalize well to new data.

Train vs val dataset

This is where the validation set comes in.

We can use it to tune our hyperparameters. We train our model on the training data with various hyperparameters, then evaluate their performance on the validation set. The hyperparameters that make the model perform best on the validation set are the ones we choose.

Train vs val dataset

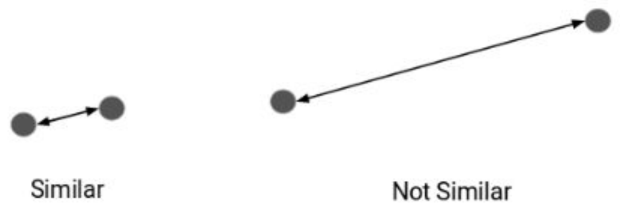
This is where the validation set comes in.

We can use it to tune our hyperparameters. We train our model on the training data with various hyperparameters, then evaluate their performance on the validation set. The hyperparameters that make the model perform best on the validation set are the ones we choose.

So, the validation set allows us to tune our model's hyperparameters, choose between different model structures, and make other decisions, all without touching our test set.

Distance metrics

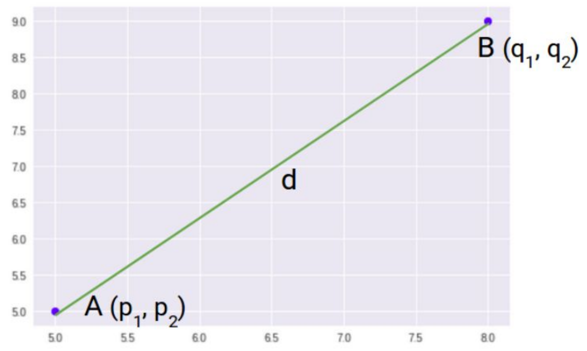
- Step 2 in K-NN: Calculate distance
- A distance metric is a function that defines the distance between elements
- It quantifies the similarity between two data points
- Choosing an effective distance metric improves the performance of our model



Common distance metrics

- **Euclidean distance**
  - Most common metric
  - Straight line distance between two points in the Euclidean space

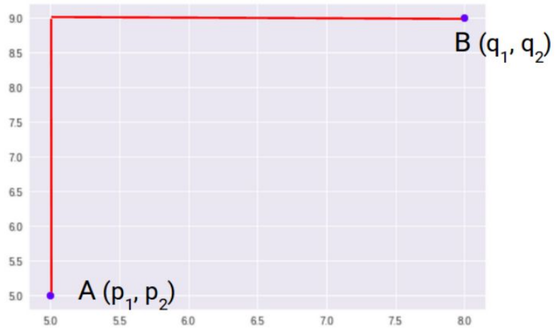
$$D_e = \left(\sum_{i=1}^{n} (p_i - q_i)^2 \right)^{1/2}$$



Common distance metrics

- **Manhattan distance**
  - Also known as City Block distance
  - Works better with high dimensional data

$$D_m = \sum_{i=1}^{n} |p_i - q_i|$$



Common distance metrics

- **Minkowski distance**
  - Generalization of both Euclidean and Manhattan distances
  - p=2 Euclidean
  - p=1 Manhattan
  - You can experiment with different values of p

$$D = \left(\sum_{i=1}^{n} |p_i - q_i|^p \right)^{1/p}$$

Minkowski

$$D_e = \left(\sum_{i=1}^{n} (p_i - q_i)^2 \right)^{1/2}$$

Euclidean

$$D_m = \sum_{i=1}^{n} |p_i - q_i|$$

Manhattan

Feature scaling (data normalization)

- It’s a method used to standardize the range of independent variables
- Why?
  - When features are at different scales, algorithms based on distances might not perform well
  - A feature with a high range of values can dominate the outcome
- Scaling methods:
  - Min-Max scaling
  - **Standard scaling (z-score normaliz**
  - Robust scaling

$$z = \frac{x_i - \mu}{\sigma}$$

⚠️ When do you do feature scaling?

- It's crucial to fit the scaler on your training data and not the entire dataset. This is to avoid data leakage
  - If you were to fit the scaler on the entire dataset, it would calculate scaling parameters (like the mean and standard deviation for standardization) that reflect the distribution of the entire dataset, test set included

- Once the scaler is fitted on the training set, it can then be used to transform the training set, as well as the validation and test sets. This means the test data is scaled according to the statistics of the training data, which mirrors the real-life scenario where we apply the model to unseen data

⚠️ When do you do feature scaling?

To recap:

- Fit the scaler to the training data.
- Transform the training data.
- Use the fitted scaler to transform the validation/test data when needed.

This approach ensures that the model evaluation is fair and unbiased, and it simulates the real-world application of a model as closely as possible.

Curse of dimensionality

- A phenomenon where the feature space becomes increasingly sparse as the number of dimensions increase. This leads to data being isolated or distant from each other, making it difficult to identify patterns or make predictions

- In K-NN, distances between points are key to the algorithm's effectiveness. In high-dimensional space, points tend to be far apart, leading to less meaningful nearest neighbors. The distance between nearest and farthest neighbors tends to become less distinguishable, which reduces the effectiveness of K-NN