

CSS 330 Data wrangling and visualization

Lecture 11

Introduction to Machine Learning

Instructor: Nabigazinova Elnura

Machine Learning

Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial **intelligence** based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention.



scikit-learn

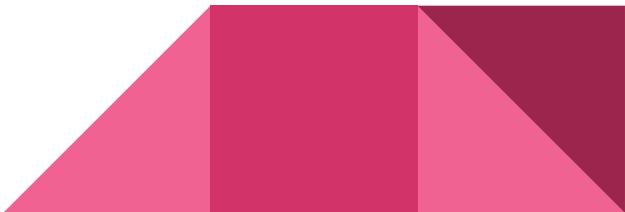
Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license



Machine Learning

One definition: "Machine learning is the semi-automated extraction of knowledge from data"

- **Knowledge from data:** Starts with a question that might be answerable using data
 - **Automated extraction:** A computer provides the insight
 - **Semi-automated:** Requires many smart decisions by a human
- 

Learning Paradigms:

- **Supervised learning:** Making predictions using data
- **Unsupervised learning:** Extracting structure from data
- **Reinforcement learning:** Neural Network



Supervised learning

Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output: $Y = f(X)$

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.



Supervised learning

It is called supervised learning because the process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers, the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.



Supervised learning

Step1 (Train): You provide the system with data that contains photos of apples and let it know that these are apples. This is called labeled data.

Step2 (Test): The model learns from the labeled data and the next time you ask it to identify an apple, it can do it easily.

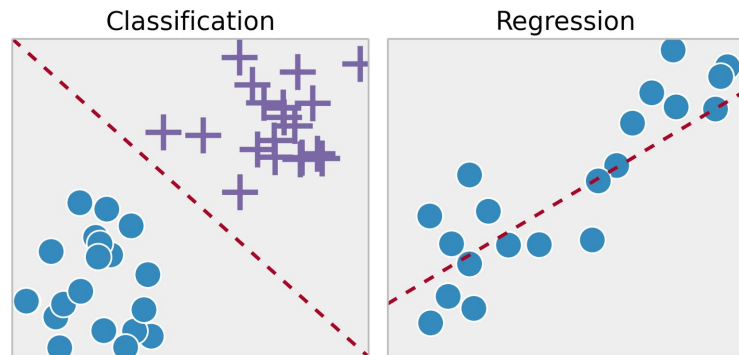


Supervised learning

Supervised learning problems can be further grouped into regression and classification problems:

Classification: A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.

Regression: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

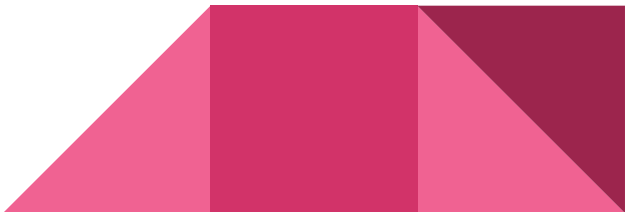


Unsupervised learning

Unsupervised learning is where you only have input data (X) and no corresponding output variables.

The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.

These are called unsupervised learning because unlike supervised learning above there is no correct answers and there is no teacher. Algorithms are left to their own devices to discover and present the interesting structure in the data

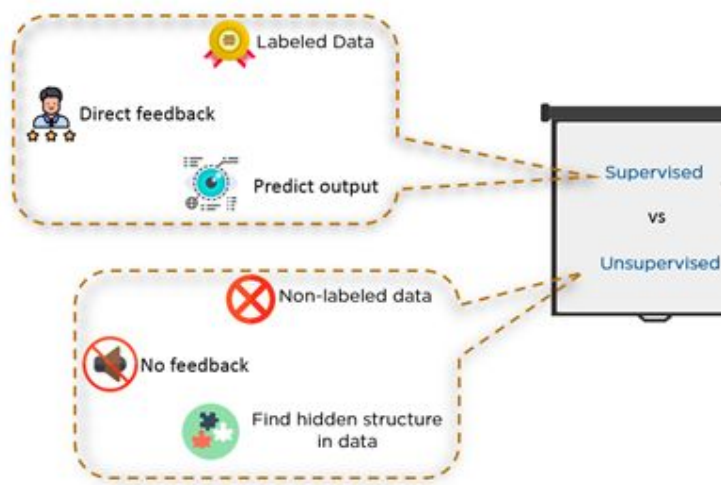


Supervised learning vs Unsupervised learning

1. Type of input data – In case of Supervised Learning, the **input data is labeled** and in case of Unsupervised Learning, the **input data is non labeled**.

2. Feedback – In case of Supervised Learning, the **system learns from the output and keeps it in mind** while in case of unsupervised learning, **there is no feedback involved**.

3. Function – Supervised Learning is generally used to **predict data** whereas, Unsupervised Learning is used to **find hidden structure in the data**.



Reinforcement learning

Video games are full of reinforcement cues. Complete a level and earn a badge. Defeat the bad guy in a certain number of moves and earn a bonus. Step into a trap — game over.

These cues help players learn how to improve their performance for the next game. Without this feedback, they would just take random actions around a game environment in the hopes of advancing to the next level.

The idea behind Reinforcement Learning is that an agent will learn from the environment by interacting with it and receiving rewards for performing actions.



2 examples of Supervised learning

2 examples Unsupervised learning

2 example Reinforcement learning




How does machine learning "work"?

1. First, train a **machine learning model** using **labeled data**
 - "Labeled data" has been labeled with the outcome
 - "Machine learning model" learns the relationship between the attributes of the data and its outcome
2. Then, make **predictions** on **new data** for which the label is unknown

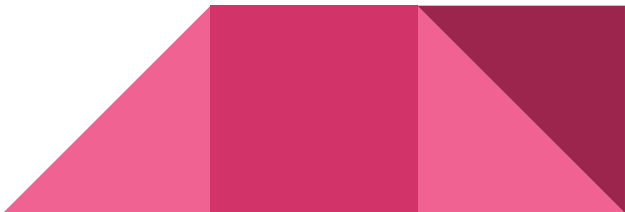
The primary goal of supervised learning is to build a model that "generalizes": It accurately predicts the **future** rather than the **past**!



Machine learning terminology

- Each row is an **observation** (also known as: sample, example, instance, record)
 - Each column is a **feature** (also known as: predictor, attribute, independent variable, input, regressor, covariate)
 - Each value we are predicting is the **response** (also known as: target, outcome, label, dependent variable)
 - **Classification** is supervised learning in which the response is categorical
 - **Regression** is supervised learning in which the response is ordered and continuous
- 

Requirements for working with data in scikit-learn

1. Features and response are **separate objects**
 2. Features and response should be **numeric**
 3. Features and response should be **NumPy arrays**
 4. Features and response should have **specific shapes**
- 


Confusion matrix

A confusion matrix is a table that is often used to **describe the performance of a classification model** (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.



Confusion matrix

Terminology

- **true positives (TP)**: These are cases in which we predicted yes (they have the disease), and they do have the disease.
 - **true negatives (TN)**: We predicted no, and they don't have the disease.
 - **false positives (FP)**: We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
 - **false negatives (FN)**: We predicted no, but they actually do have the disease. (Also known as a "Type II error.")
- 

Cancer classification problem

- Let's say we are solving a classification problem where we are predicting whether a person is having cancer or not.
- Let's give a label to our target variable:
1: When a person is having cancer
0: When a person is NOT having cancer.

		Actual	
		Positives(1)	Negatives(0)
Predicted	Positives(1)	TP	FP
	Negatives(0)	FN	TN

Cancer classification problem

- **True Positives (TP):** The case where a person is actually having cancer(1) and the model classifying his case as cancer(1) comes under True positive.
- **True Negatives (TN):** The case where a person NOT having cancer and the model classifying his case as Not cancer comes under True Negatives.
- **False Positives (FP):** A person NOT having cancer and the model classifying his case as cancer comes under False Positives.
- **False Negatives (FN):** A person having cancer and the model classifying his case as No-cancer comes under False Negatives.

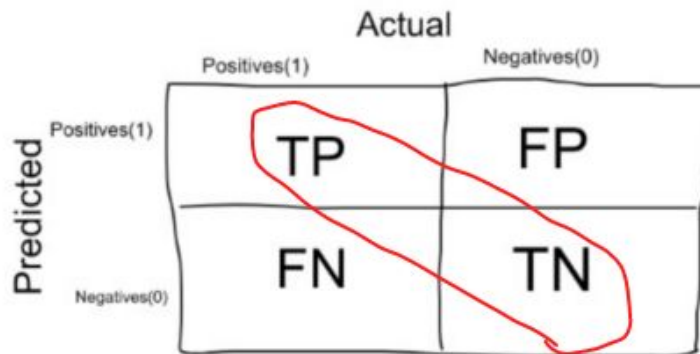
		Actual	
		Positives(1)	Negatives(0)
Predicted	Positives(1)	TP	FP
	Negatives(0)	FN	TN

Accuracy

- Accuracy in classification problems is the number of correct predictions made by the model over all kinds predictions made.
- In the Numerator, are our correct predictions (True positives and True Negatives)(Marked as red in the fig above) and in the denominator, are the kind of all predictions made by the algorithm(Right as well as wrong ones).
- Accuracy is a good measure when the target variable classes in the data are nearly balanced.

Ex:60% classes in our fruits images data are apple and 40% are oranges.

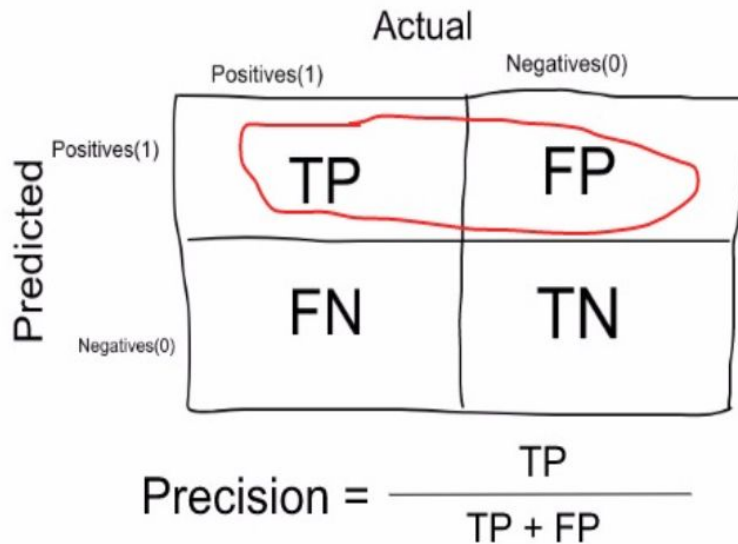
		Actual	
		Positives(1)	Negatives(0)
Predicted	Positives(1)	TP	FP
	Negatives(0)	FN	TN



$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

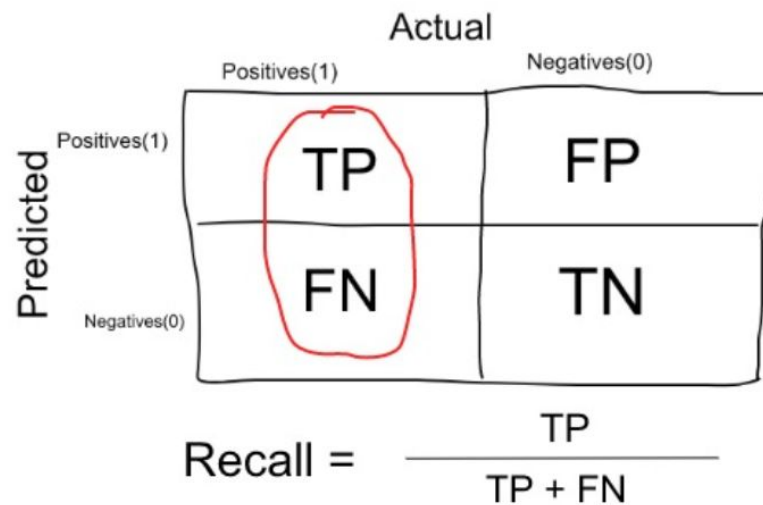
Precision

- Precision is a measure that tells us what proportion of patients that we diagnosed as having cancer, actually had cancer. The predicted positives (People predicted as cancerous are TP and FP) and the people actually having a cancer are TP.
- Ex: In our cancer example with 100 people, only 5 people have cancer. Let's say our model is very bad and predicts every case as Cancer. Since we are predicting everyone as having cancer, our denominator(True positives and False Positives) is 100 and the numerator, person having cancer and the model predicting his case as cancer is 5. So in this example, we can say that Precision of such model is 5%.



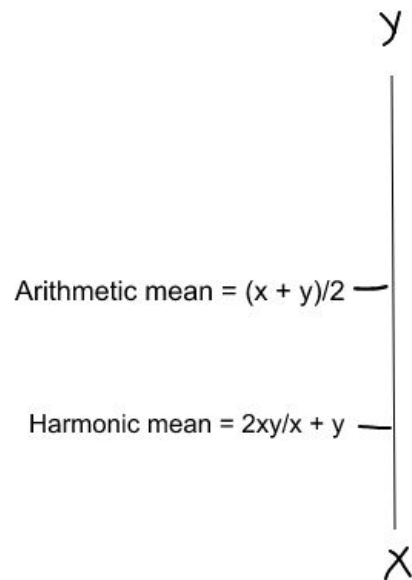
Recall

- Recall is a measure that tells us what proportion of patients that actually had cancer was diagnosed by the algorithm as having cancer. The actual positives (People having cancer are TP and FN) and the people diagnosed by the model having a cancer are TP. (Note: FN is included because the Person actually had a cancer even though the model predicted
- Ex: In our cancer example with 100 people, 5 people actually have cancer. Let's say that the model predicts every case as cancer.
So our denominator(True positives and False Negatives) is 5 and the numerator, person having cancer and the model predicting his case as cancer is also 5(Since we predicted 5 cancer cases correctly). So in this example, we can say that the Recall of such model is 100%. And Precision of such a model(As we saw above) is 5%



F1 score

- The Harmonic mean is given by the formula shown in the figure on the left.
- Harmonic mean is kind of an average when x and y are equal. But when x and y are different, then it's closer to the smaller number as compared to the larger number.
- For our previous example, F1 Score = Harmonic Mean(Precision, Recall)
- F1 Score = $2 * \text{Precision} * \text{Recall} / (\text{Precision} + \text{Recall}) = 2 * 3 * 100 / 103 = 5\%$



Practicing Supervised Learning

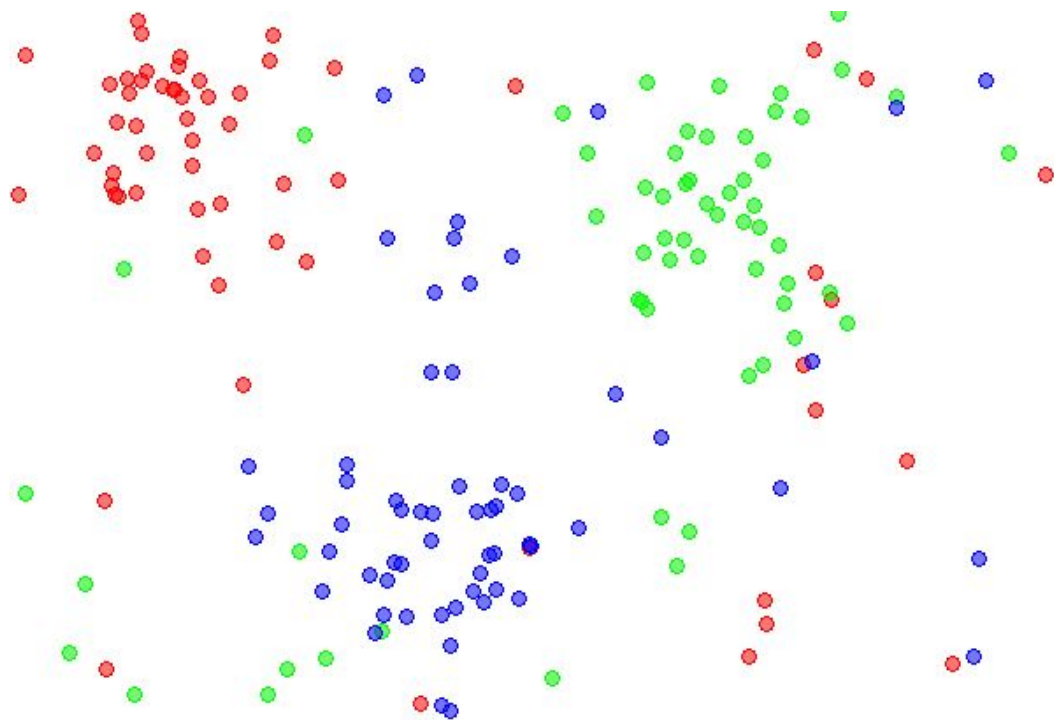


K-nearest neighbors (KNN) classification

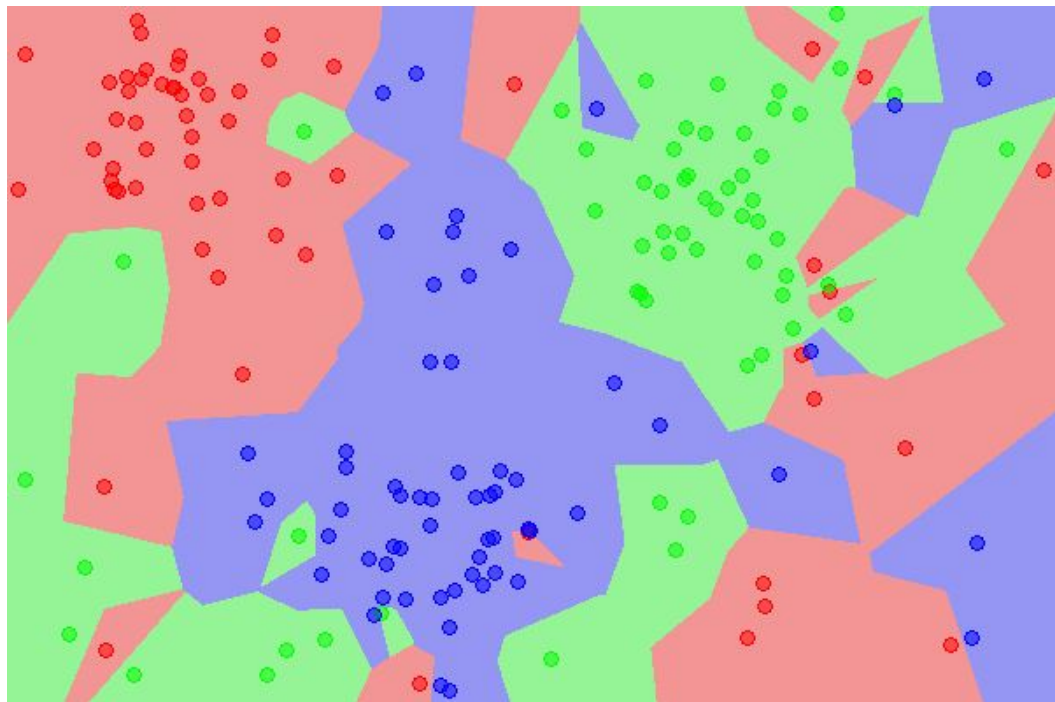
1. Pick a value for K.
2. Search for the K observations in the training data that are "nearest" to the measurements of the unknown iris.
3. Use the most popular response value from the K nearest neighbors as the predicted response value for the unknown iris. (iris is the open source dataset)



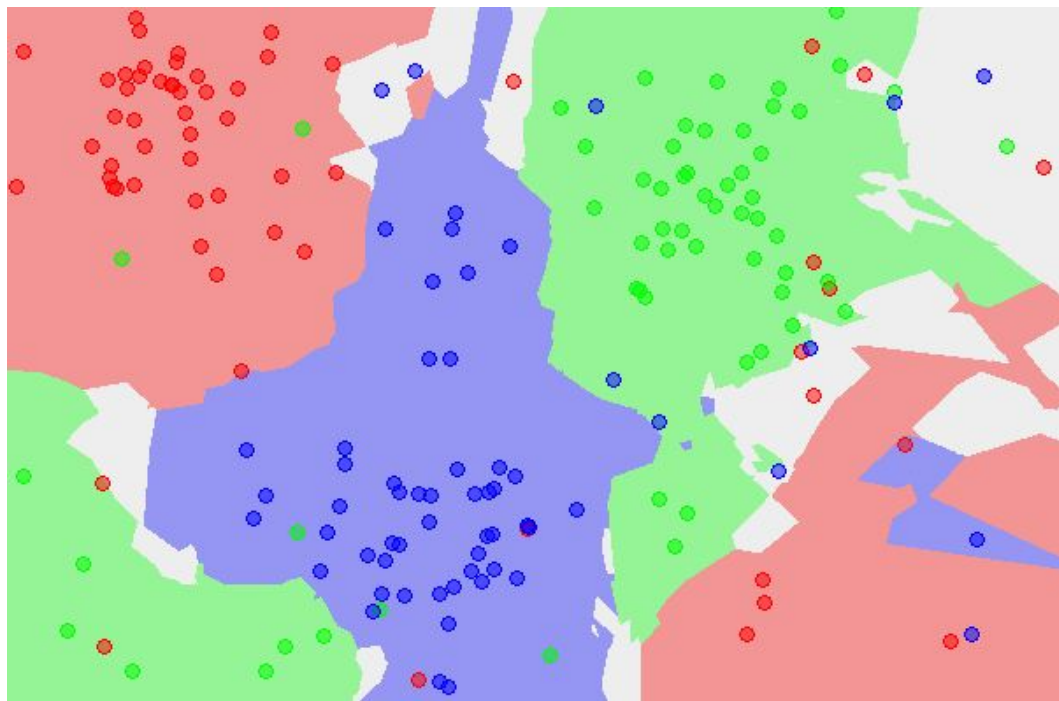
Example training data



KNN classification map (K=1)



KNN classification map (K=5)



scikit-learn 4-step modeling pattern

Step 1: Import the class you plan to use

```
>>>from sklearn.neighbors import KNeighborsClassifier
```

Step 2: "Instantiate" the "estimator"

- "Estimator" is scikit-learn's term for model
- "Instantiate" means "make an instance of"

```
>>>knn = KNeighborsClassifier(n_neighbors=1)
```



scikit-learn 4-step modeling pattern

Step 3: Fit the model with data (aka "model training")

Model is learning the relationship between X and y

Occurs in-place

```
>>> knn.fit(X, y)
```

Step 4: Predict the response for a new observation

- New observations are called "out-of-sample" data
- Uses the information it learned during the model training process

```
>>> knn.predict([[3, 5, 4, 2]])  
array([2])
```

- Returns a NumPy array
 - Can predict for multiple observations at once
- 

Useful links


Check following links for info:

- Main resource - <https://github.com/justmarkham/scikit-learn-videos>
- Learning Paradigms - <http://work.caltech.edu/library/014.html>
- Iris dataset - [UCI Machine Learning Repository](#)
- Dataset for the lab - <https://www.kaggle.com/paree24/development-index>



Cross-validation

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k -fold **cross-validation**.



Review of model evaluation procedures

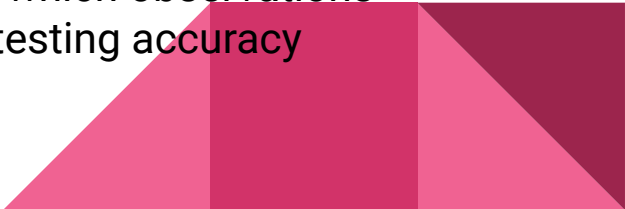
Motivation: Need a way to choose between machine learning models

- Goal is to estimate likely performance of a model on **out-of-sample data**

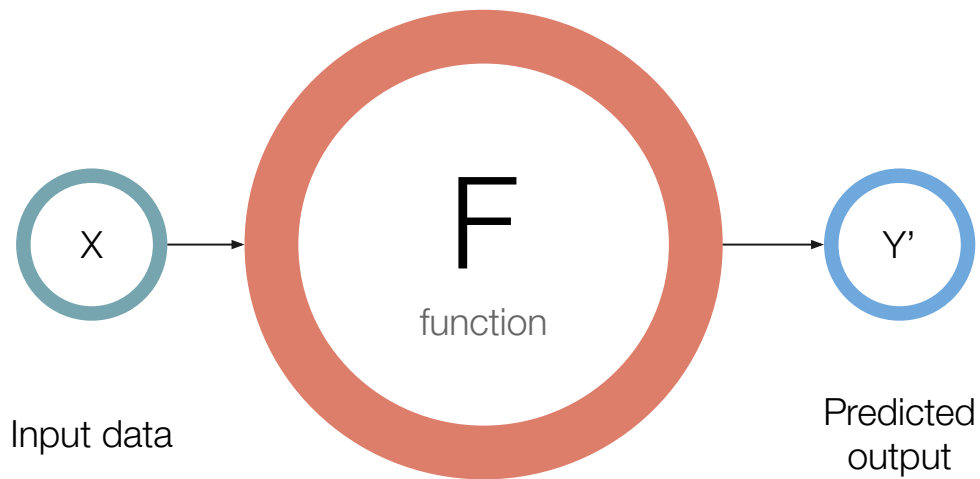
Initial idea: Train and test on the same data

- But, maximizing **training accuracy** rewards overly complex models which **overfit** the training data

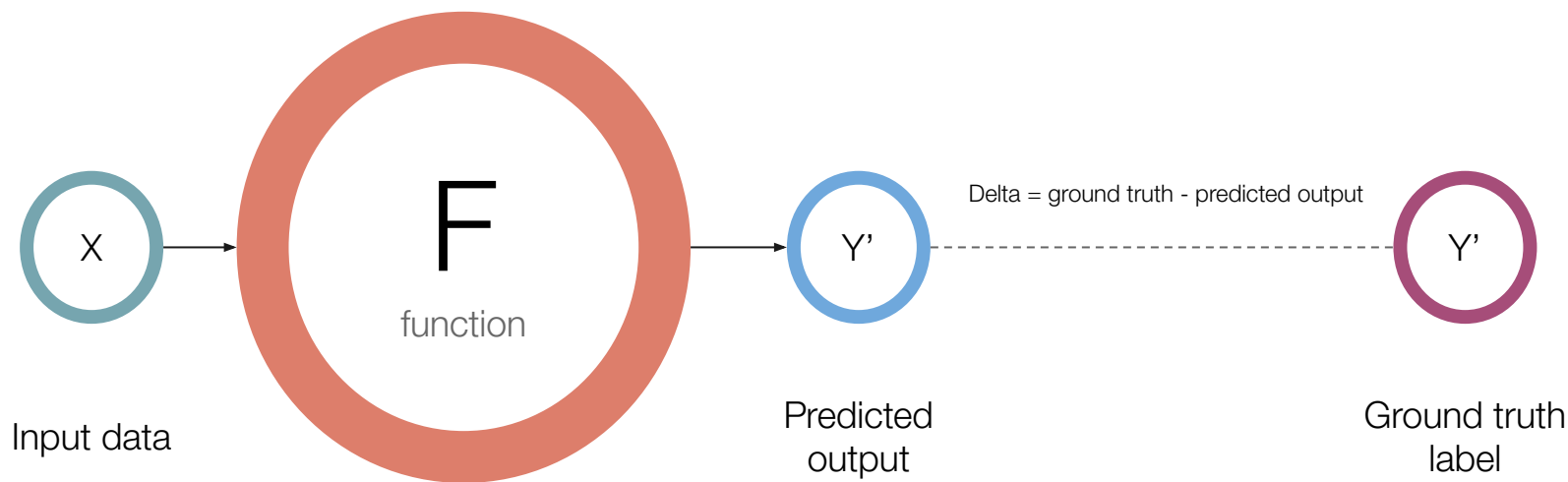
Alternative idea: Train/test split

- Split the dataset into two pieces, so that the model can be trained and tested on **different data**
 - **Testing accuracy** is a better estimate than training accuracy of out-of-sample performance
 - But, it provides a **high variance** estimate since changing which observations happen to be in the testing set can significantly change testing accuracy
- 

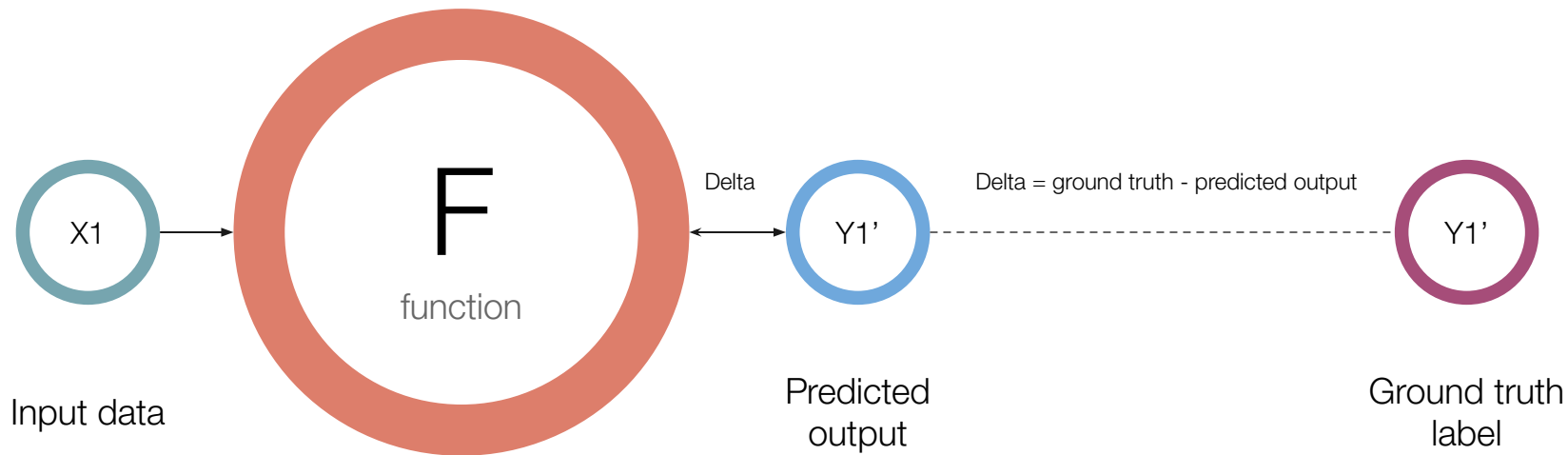
Training



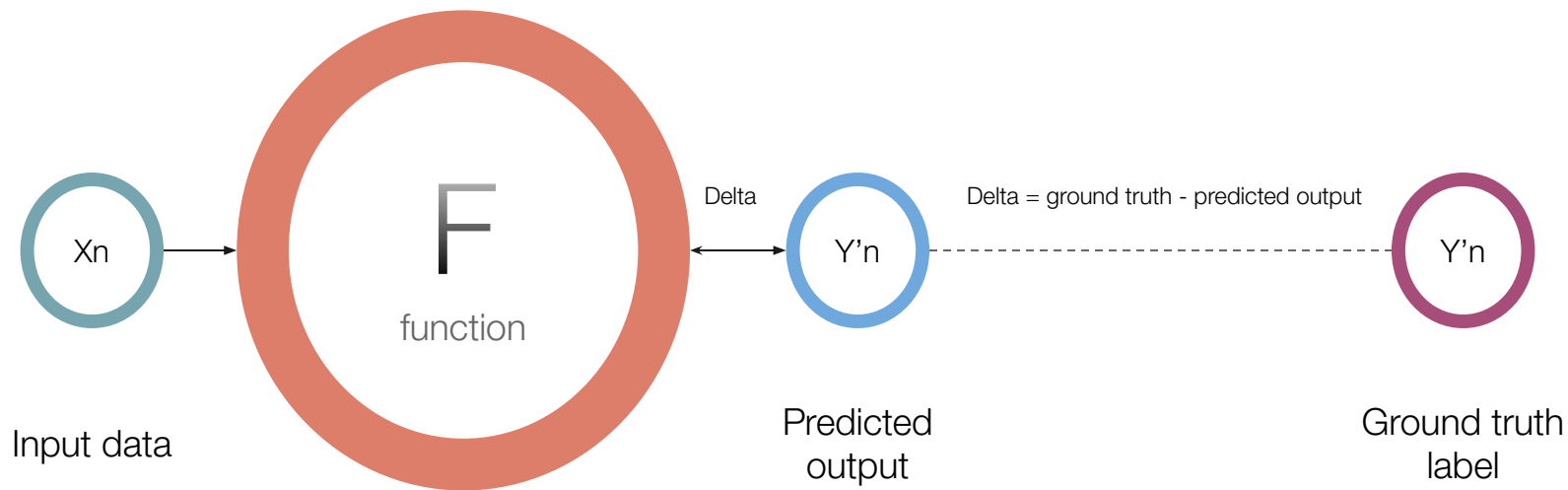
Training



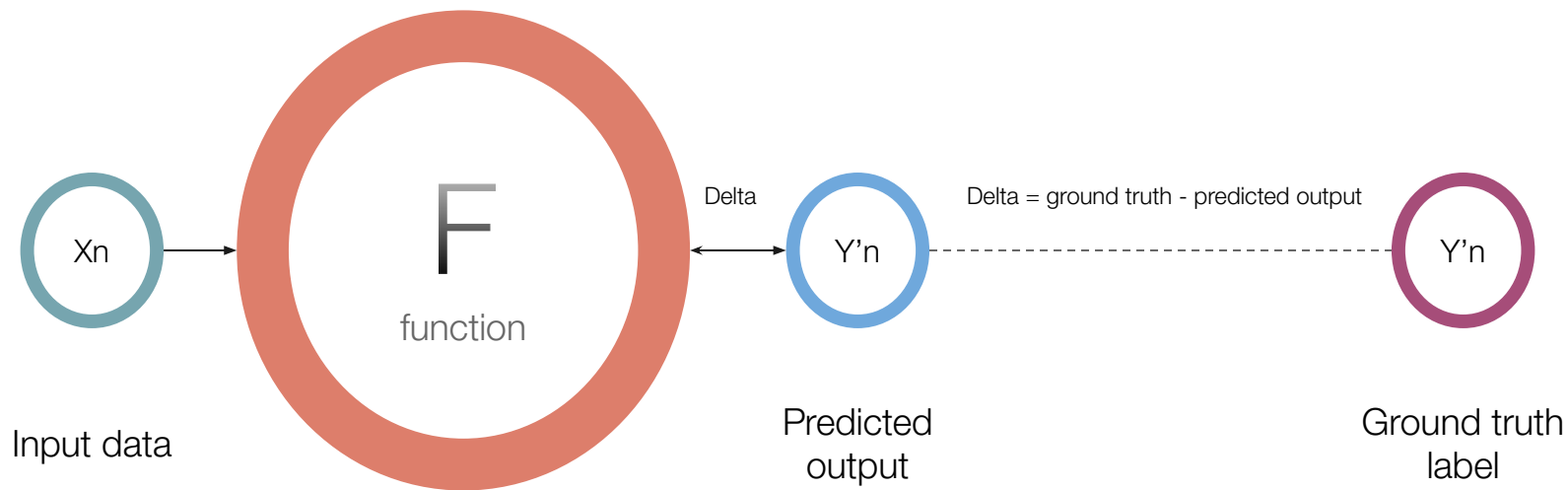
Training (iteration #1)



Training (iteration #n)

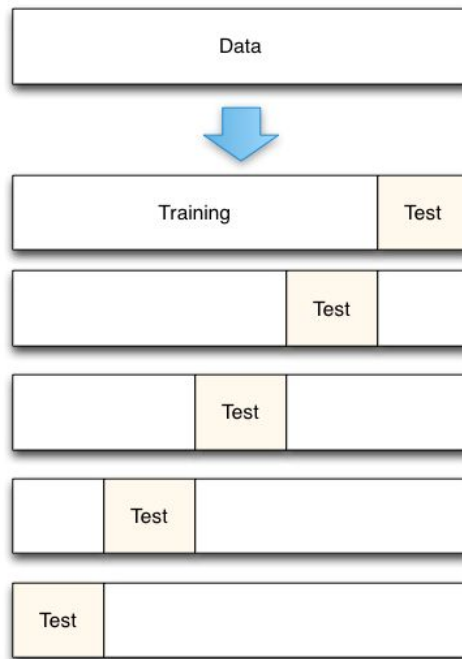


Training (iteration #n)



Steps for K-fold cross-validation

- Split the dataset into K equal partitions (or "folds").
- Use fold 1 as the testing set and the union of the other folds as the training set.
- Calculate testing accuracy.
- Repeat steps 2 and 3 K times, using a different fold as the testing set each time.
- Use the average testing accuracy as the estimate of out-of-sample accuracy.



Cross-validation recommendations

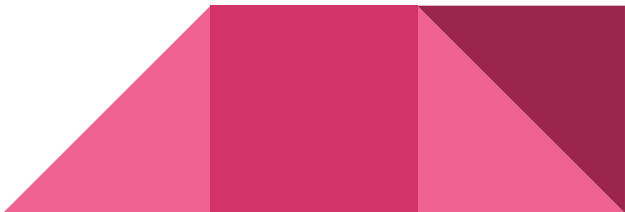
- K can be any number, but $K=10$ is generally recommended
- For classification problems, stratified sampling is recommended for creating the folds
 - a. Each response class should be represented with equal proportions in each of the K folds
 - b. scikit-learn's `cross_val_score` function does this by default



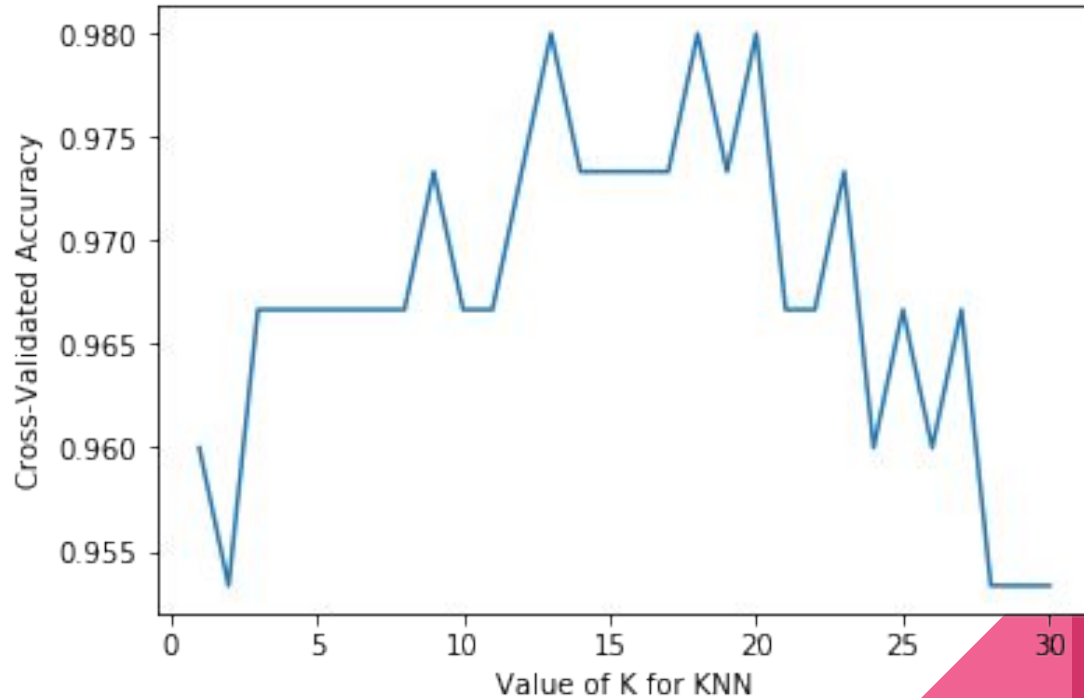
Cross-validation example: parameter tuning

Goal: Select the best tuning parameters (aka "hyperparameters") for KNN on the iris dataset

```
# search for an optimal value of K for KNN
k_range = list(range(1, 31))
k_scores = []
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X, y, cv=10, scoring='accuracy')
    k_scores.append(scores.mean())
print(k_scores)
```



Cross-validation example: parameter tuning



Useful links

Check following links for info:

- Cross validation -
https://scikit-learn.org/stable/modules/cross_validation.html
- Dataset for the lab -
<https://www.kaggle.com/miroslavsabo/young-people-survey>



Assignment 8

Dataset - <https://www.kaggle.com/miroslavsabo/young-people-survey>

Solve a classification problem to predict a one of the Phobias value in the dataset.

Use cross-validation process and selection of best model and best hyperparameters.

- Use any model in this task
- Select only 5 columns (features)
- Convert data in dataset to numeric data as necessary
- Explain the process of best model selection
- Submit your *.ipynb and *.pdf with explanations of your decisions.

For any descriptive information about dataset refer to the link at the top.

