

# Supervised Learning

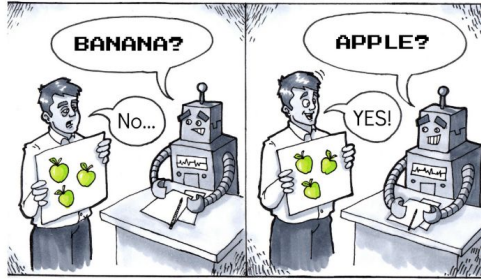
Let's use the first 20 minutes of the class to finish and review the last class exercises.

We will kick-off today's topics at 17:50

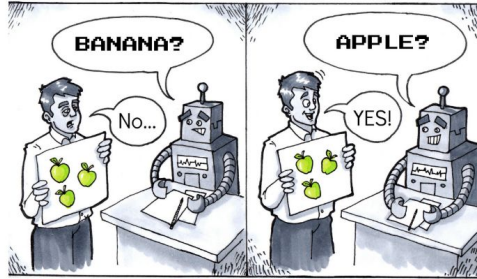


# Machines that Learn

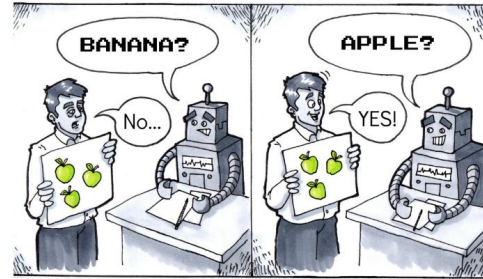
Today.



**Supervised Learning**



**Supervised Learning**



**Supervised Learning**

Cross-Validation

Hyperparameter Search

# Supervised Learning

Some of the last class observations

- When I change the train and test split the results change, should I do this? There's not that much data, is my model robust?
- What K of KNN is the best? Do I try it out manually?

# Supervised Learning

Robustness.

The ability of a model to **generalize** well to **unseen data** and perform **consistently** under **different conditions**.

# Supervised Learning

It is not **generalizing** well.

# Supervised Learning

It is not generalizing well.

Its either **underfitting** or **overfitting**

# Supervised Learning

It is not generalizing well.

Its either **underfitting** or **overfitting**

The model is too simple to capture the patterns in the data.

The model learns the training data too well, including noise, and fails to generalize to new data.

# Supervised Learning

It is not generalizing well.

Its either **underfitting** or **overfitting**

The model is too simple to capture the patterns in the data - **underfitting**

The model learns the training data too well, including noise, and fails to generalize to new data - **overfitting**



# Supervised Learning

It is not generalizing well.

Its either **underfitting** or **overfitting**

The model is too simple to capture the patterns in the data - **underfitting**

The model learns the training data too well, including noise, and fails to generalize to new data - **overfitting**

Note: if you increase test-size and the evaluation metrics improve, the model was *probably* overfitting.

# Supervised Learning

It is not generalizing well.

Its either **underfitting** or **overfitting**

The model is too simple to capture the patterns in the data - **underfitting**

The model learns the training data too well, including noise, and fails to generalize to new data - **overfitting**

Note: if you increase test-size and the evaluation metrics improve, the model was *probably* overfitting.

# Supervised Learning

When it is not generalizing well.

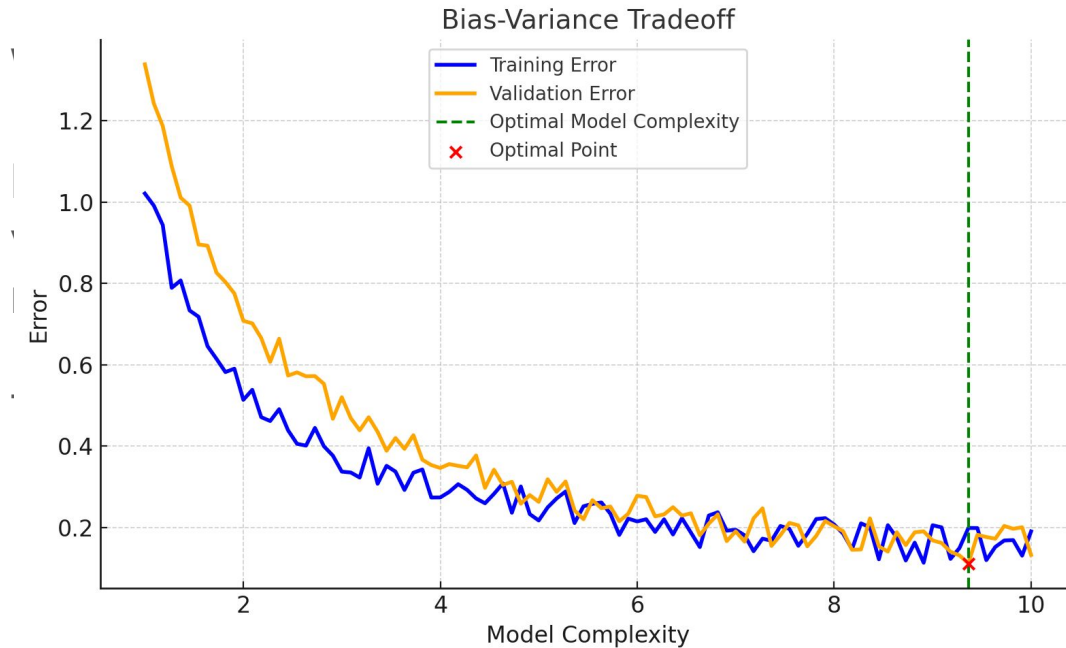
We also talk often about the Bias-Variance Trade-off.

Bias is the error from overly simplistic models that underfit the data, while variance is the error from overly complex models that overfit to training data noise.

The key to robustness is finding the balance between the two.

# Supervised Learning

When it is not generalizing well.



ff.

underfit the data, while  
not overfit to training data

between the two.

# Supervised Learning

Robustness.

The ability of a model to **generalize** well to **unseen data** and perform **consistently** under **different conditions**.

One of the best ways to assess this is with cross-**validation**.

# Supervised Learning

## Cross Validation

A technique to evaluate model performance by **splitting the data into multiple subsets (folds)** and **training/testing the model on different combinations of these subsets**

# Supervised Learning

## Cross Validation

A technique to evaluate model performance by **splitting the data into multiple subsets (folds)** and **training/testing the model on different combinations of these subsets**



using validation datasets\*

# Supervised Learning

## Cross Validation

Cross-validation provides a **reliable** and **systematic approach to evaluate models**, especially when data is limited, **reducing the risk** of overfitting or underfitting



# Supervised Learning

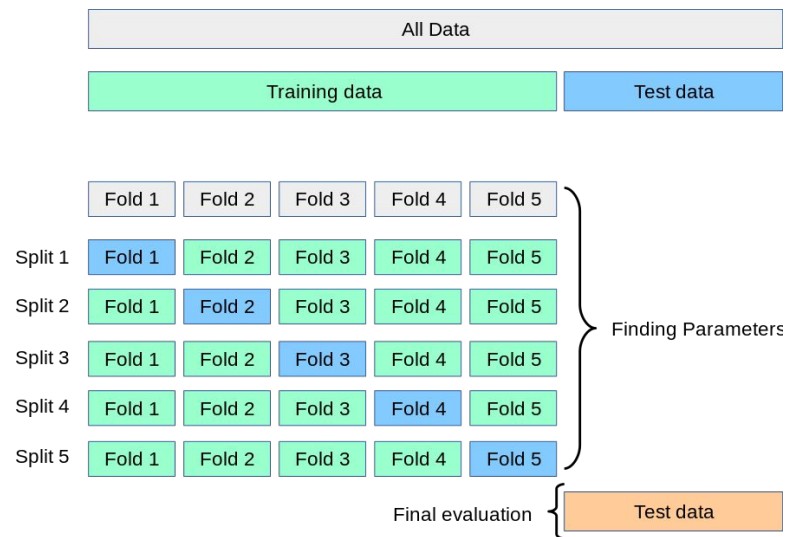
## Cross Validation Schemes

- K-Fold Cross Validation
- Leave-One-Out Cross Validation
- Stratified Cross Validation
- Time-Series Cross Validation
- Repeated Cross-Validation

# Supervised Learning

## Cross Validation Schemes

- K-Fold Cross Validation



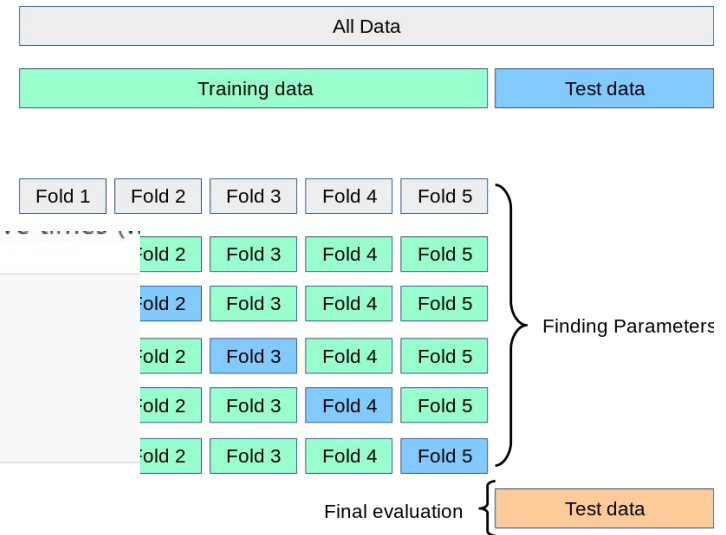
# Supervised Learning

## Cross Validation Schemes

- K-Fold Cross Validation  
(`sklearn.model_selection.cross_val_score`)

splitting the data, training a model and computing the score 5 consecutive times (5

```
>>> from sklearn.model_selection import cross_val_score
>>> clf = svm.SVC(kernel='linear', C=1, random_state=42)
>>> scores = cross_val_score(clf, X, y, cv=5)
>>> scores
array([0.96..., 1. , 0.96..., 0.96..., 1. ])
```



# Supervised Learning

## Cross Validation Schemes

- Leave-One-Out Cross Validation

Leave-One-Out

	$x_1$	$x_2$	$x_3$	$y$	
1					Training Set
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					
28					
29					
30					Testing Set

# Supervised Learning

## Cross Validation Schemes

- Leave-One-Out Cross Validation  
(`sklearn.model_selection.LeaveOneOut`)  
I.e leave one out and repeat process N times,  
where N is the time number of observations

Leave-One-Out

	$x_1$	$x_2$	$x_3$	$y$	
1					Training Set
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					
28					
29					
30					Testing Set

# Supervised Learning

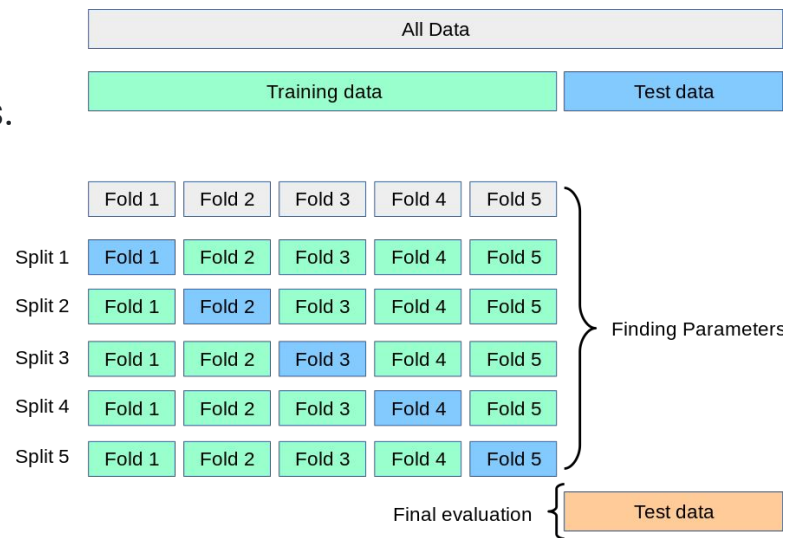
## Cross Validation Schemes

- Stratified Cross Validation

# Supervised Learning

## Cross Validation Schemes

- **Stratified Cross Validation**  
(`sklearn.model_selection.StratifiedKFold`)  
is a variation of K-Fold that returns stratified folds.  
The folds are made by preserving the percentage  
of samples for each class.



# Supervised Learning

## Cross Validation Schemes

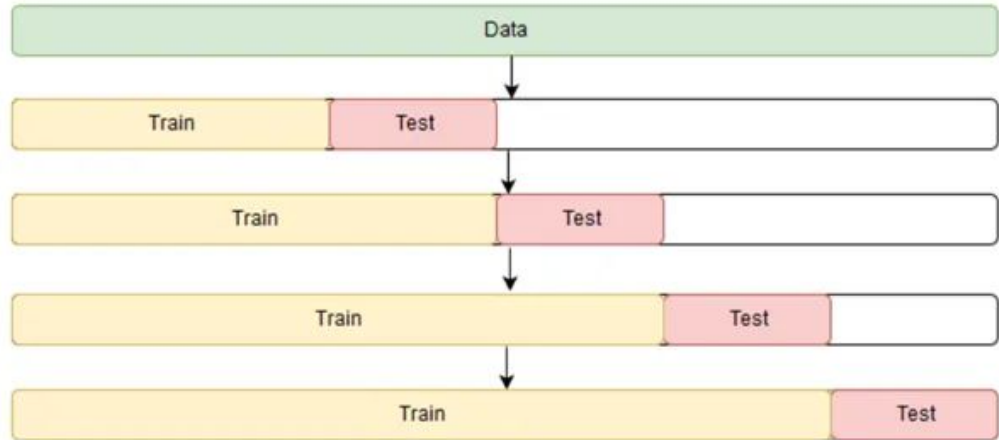
- Time-Series Cross Validation



# Supervised Learning

## Cross Validation Schemes

- Time-Series Cross Validation  
(`sklearn.model_selection.TimeSeriesSplit`)  
i.e folds are sequential



# Supervised Learning

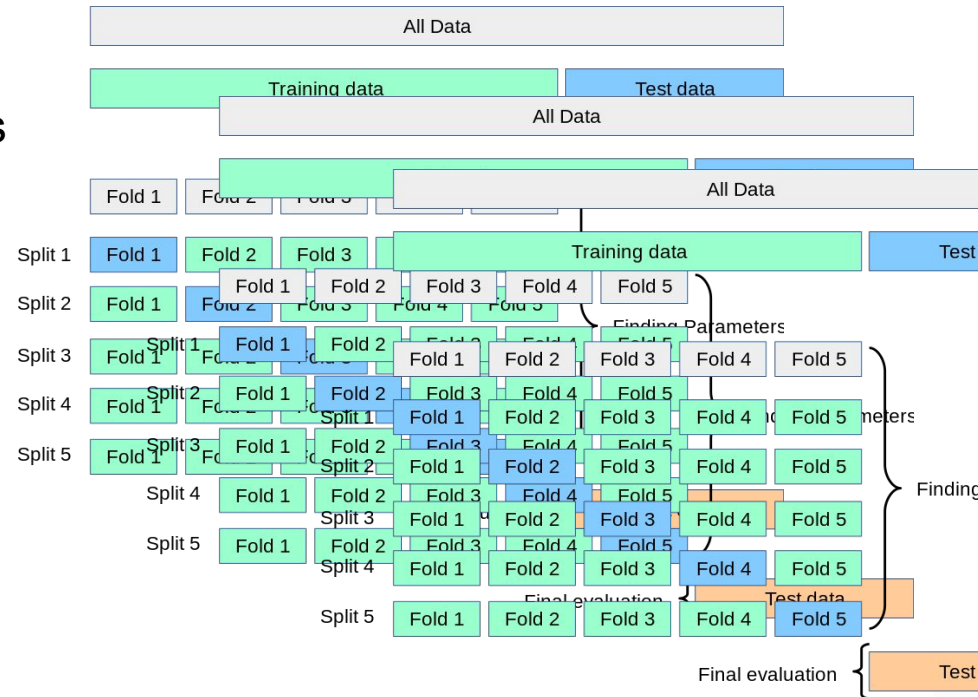
## Cross Validation Schemes

- Repeated Cross Validation

# Supervised Learning

## Cross Validation Schemes

- Repeated Cross Validation  
i.e repeat k-fold validation multiple times  
(with different splits)



# Supervised Learning

First an example



# Supervised Learning

Let's go back to our classification notebooks and do some cross validation.

You can duplicate your notebook or use the same. You must adapt your code.

The questions that you need to be able to answer are:  
Which of the algorithms is more robust ?



# Supervised Learning

Some of the last class observations

- When I change the train and test split the results change, should I do this? There's not that much data, is my model robust?
- **What K of KNN is the best? Do I try it out manually?**

# Supervised Learning

Some of the last class observations

- When I change the train and test split the results change, should I do this? There's not that much data, is my model robust?
- **What K of KNN is the best? Do I try it out manually?**  
Only if you want, but no. Not really.

# Supervised Learning

Let's take a step back.



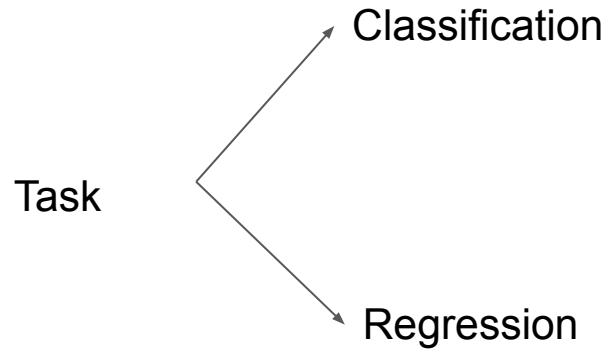
# Supervised Learning

What are we solving ?

Task

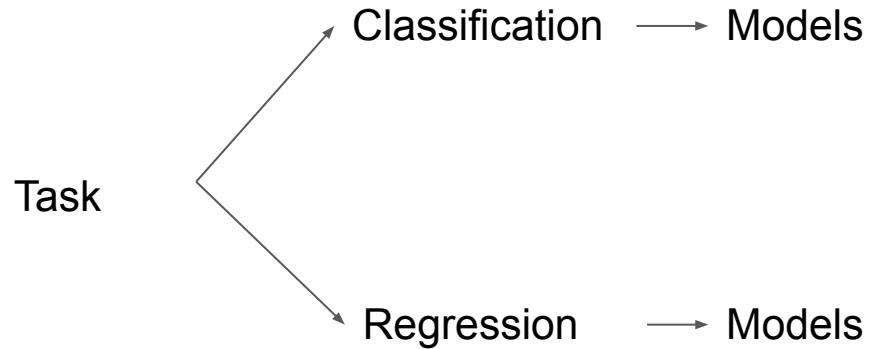
# Supervised Learning

What are we solving ?



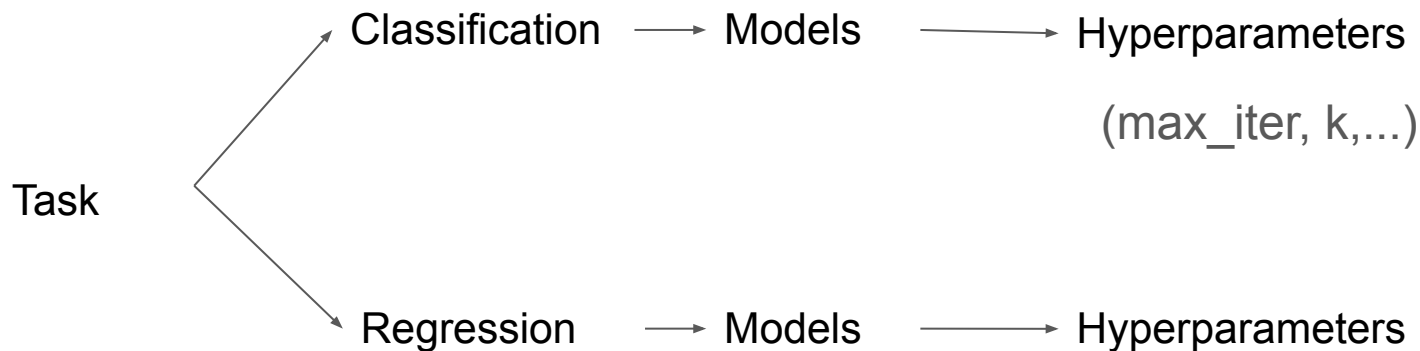
# Supervised Learning

How are we solving it?



# Supervised Learning

How are we solving it?



As data scientists we explicitly decide this, it's not learned\*

# Supervised Learning

Parameters

vs

Hyperparameters

# Supervised Learning

Parameters

vs

Hyperparameters

Are learned by the model,  
i.e they change during  
training

# Supervised Learning

## Parameters

Are learned by the model,  
i.e they change during  
training

vs

## Hyperparameters

Are decided before and  
fixed during training,  
i.e they are not learnt by  
the model and highly  
impact the model  
performance.

# Supervised Learning

## Hyperparameter Search

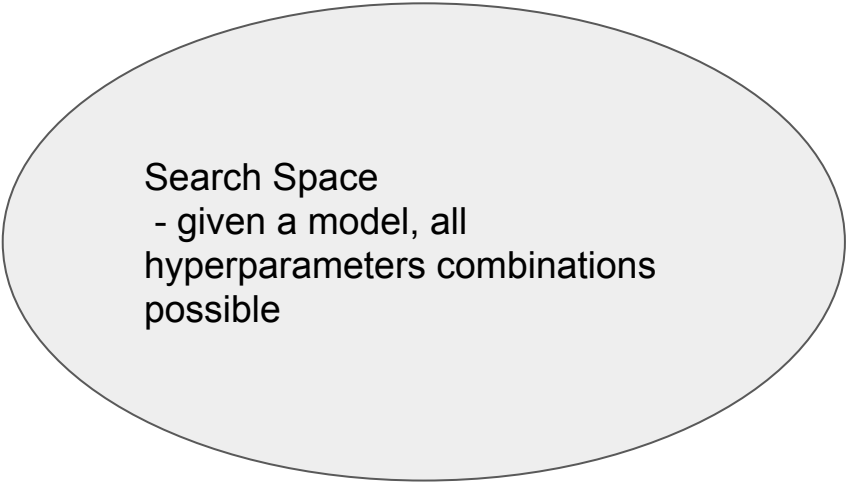
Find the best hyperparameters for the target task, model and existing data.



# Supervised Learning

## Hyperparameter Search

Find the best hyperparameters for the target task, model and existing data.



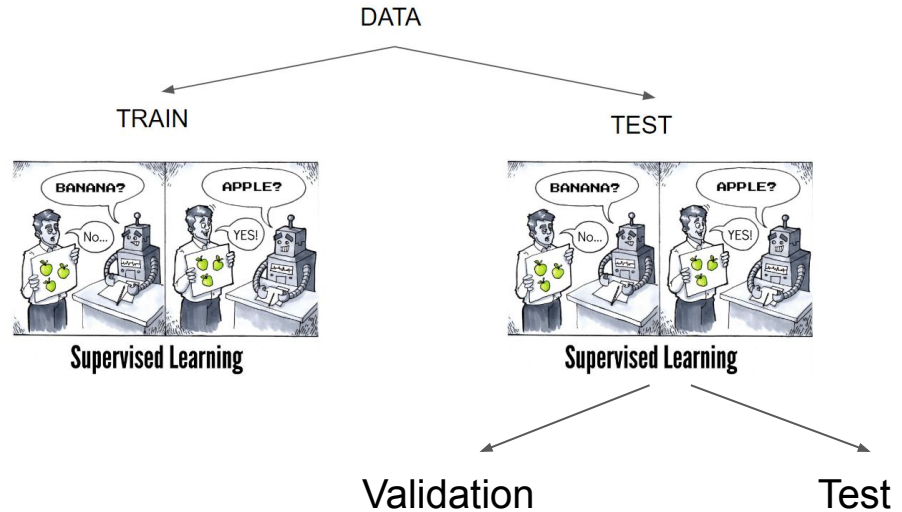
### Search Space

- given a model, all  
hyperparameters combinations  
possible

# Supervised Learning

## Hyperparameter search

Find the best hyperparameters for the target task, model and existing data.

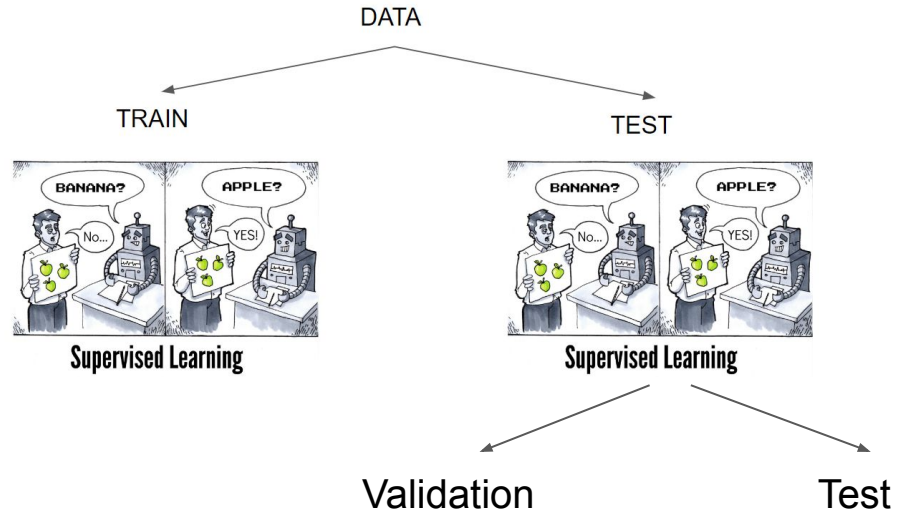


Reminder

# Supervised Learning

Hyperparameter search consists of:

- an estimator (regressor or classifier such as `sklearn.svm.SVC()`) - a model;
- a parameter space;
- a method for searching or sampling candidates;
- a cross-validation scheme
- a score function.

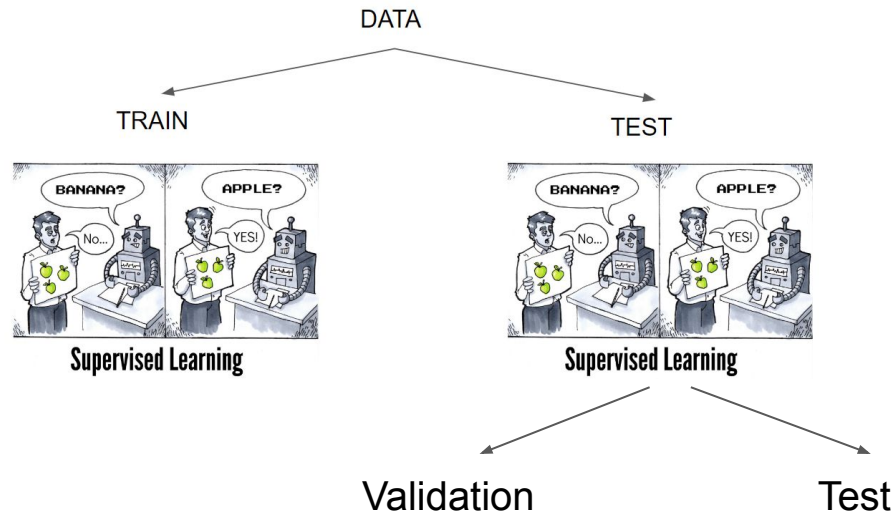


Reminder

# Supervised Learning

Hyperparameter search consists of:

- an estimator (regressor or classifier such as `sklearn.svm.SVC()`) - a model;
- a parameter space;
- **a method for searching or sampling candidates;**
- a cross-validation scheme
- a score function.



Reminder

# Supervised Learning

## Hyperparameter search

- Grid Search
- Random Search
- Bayesian Search



# Supervised Learning

## Hyperparameter search

- Grid Search  
(**systematically** evaluates a predefined set of hyperparameters combinations to find the best set of hyperparameter for a given model)  
(`sklearn.model_selection.GridSearchCV`)

# Supervised Learning

## Hyperparameter search

- Grid Search  
(`sklearn.model_selection.GridSearchCV`)

# Supervised Learning

## Hyperparameter search

- Grid Search  
(`sklearn.model_selection.GridSearchCV`)
- Random Search  
(randomly *samples* hyperparameter values from predefined distributions to efficiently explore the search space and find optimal hyperparameter configurations for machine learning models)  
(`sklearn.model_selection.RandomizedSearchCV`)



# Supervised Learning

## Hyperparameter search

- Grid Search  
(`sklearn.model_selection.GridSearchCV`)
- Random Search  
(`sklearn.model_selection.RandomizedSearchCV`)
- Bayesian Search

# Supervised Learning

## Hyperparameter search

- Grid Search  
(`sklearn.model_selection.GridSearchCV`)
- Random Search  
(`sklearn.model_selection.RandomizedSearchCV`)
- [Bayesian Search](#)  
(is a probabilistic optimization technique that uses surrogate models to efficiently explore and optimize complex hyperparameter spaces in machine learning by iteratively selecting hyperparameters to evaluate based on a balance of **exploration** and **exploitation**)  
(`skopt.BayesSearchCV`)

# Supervised Learning

## Hyperparameter search

- Grid Search - for smaller datasets and known limited search space
- Random Search - a good balance between simplicity and efficiency
- Bayesian Search - slightly more complicated to setup but smarter than random search

# Supervised Learning

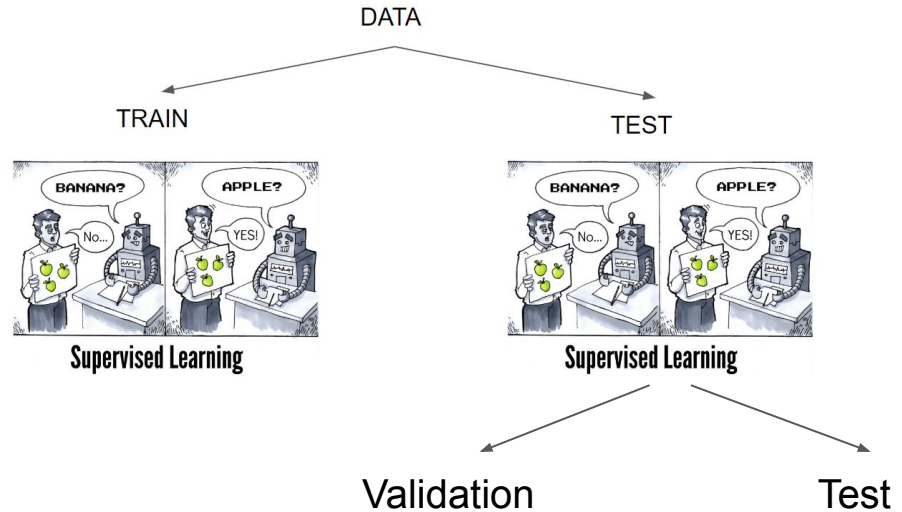
## Which One Is Used the Most in Practice?

- **Random Search** is the most widely used in practice because:
  - It strikes a good balance between simplicity and efficiency.
  - It scales well for large hyperparameter spaces without requiring domain-specific expertise to define a probabilistic model.
  - Often, randomly sampling hyperparameters is surprisingly effective since only a few hyperparameters tend to dominate performance.
- **Bayesian Search** is gaining popularity, especially in resource-constrained scenarios or for complex pipelines, but it requires more setup and understanding of surrogate models.
- **Grid Search** is less commonly used in practical scenarios, except for small, well-defined spaces, due to its inefficiency in scaling.

# Supervised Learning

Hyperparameter search consists of:

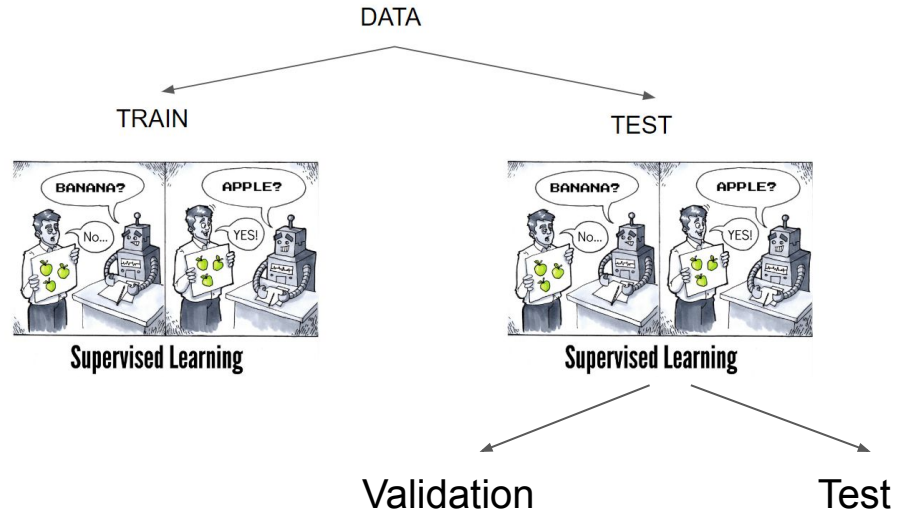
- an estimator (regressor or classifier such as `sklearn.svm.SVC()`);
- a parameter space;
- a method for searching or sampling candidates;
- **a cross-validation scheme**
- a score function.



# Supervised Learning

## Dataset splits

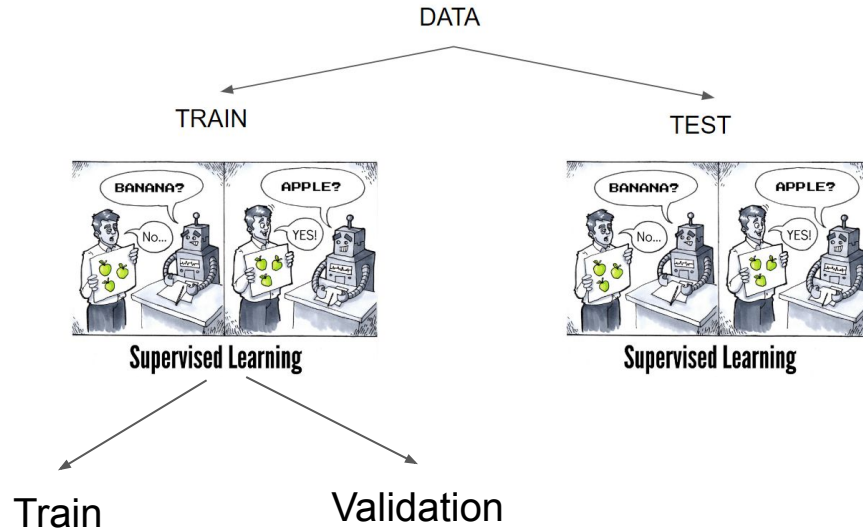
- **Train** is used to train the selected model with the selected hyperparameters
- **Validation** is used to validate the trained models
- **Test** is used to test the validated model with the higher selected metric



# Supervised Learning

## Dataset splits

- **Train** is used to train the selected model with the selected hyperparameters
- **Validation** is used to validate the trained models
- **Test** is used to test the validated model with the higher selected metric



# Supervised Learning

Hyperparameter search  
consists of:

- an estimator (regressor or classifier such as `sklearn.svm.SVC()`);
- a parameter space;
- a method for searching or sampling candidates;
- a cross-validation scheme
- **a score function.**

F1 Score  
Precision  
Recall  
Accuracy



# Supervised Learning

Hyperparameter search  
consists of:

- an estimator (regressor or classifier such as `sklearn.svm.SVC()`);
- a parameter space;
- a method for searching or sampling candidates;
- a cross-validation scheme
- a score function.

***In practice sklearn (and other libs)  
does most of it for us***

# Supervised Learning

Hyperparameter search  
consists of:

- an estimator (regressor or classifier such as `sklearn.svm.SVC()`);
- a parameter space;
- a method for searching or sampling candidates;
- a cross-validation scheme
- a score function.



# Supervised Learning

Let's go back to our classification notebooks and do some hyperparameter search.

You can duplicate your notebook or use the same. You must adapt your code.

The questions that you need to be able to answer are:

Which hyperparameters are the more suitable in all of the algorithms we've tried so far?



# Supervised Learning

[Submit.](#)

Oops it's actually a regression problem.



# Supervised Learning

## Sidenote

If we have hyperparameter search to look for the best combination of hyperparameters, could we have model search ?  
To automatically look for the best model ?  
i.e “automate” a data scientist/machine learning engineer job ?

# Supervised Learning

**Demo ?**

# Supervised Learning

**Let's explore.**

# Supervised Learning

**Packaging a model.**



# Supervised Learning

## Packaging a model.

Use formats like `joblib`, `pickle`, or ONNX for portability.

```
#after training and model is ready
import joblib
joblib.dump(model, 'model.pkl')

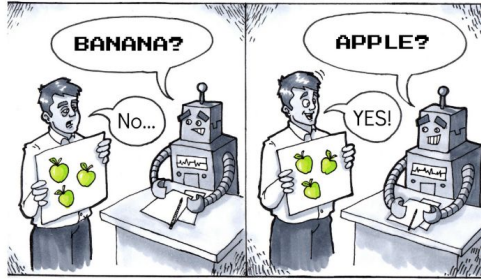
#on server
from fastapi import FastAPI
import joblib

app = FastAPI()
model = joblib.load('model.pkl')

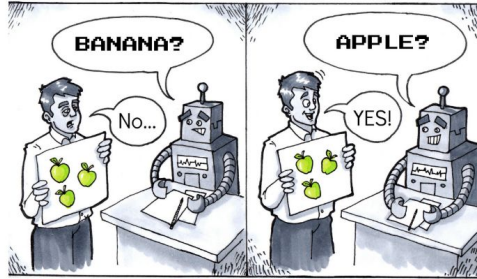
@app.post("/predict")
def predict(features: dict):
    return {"prediction":
        model.predict([features])}
```

# Machines that Learn

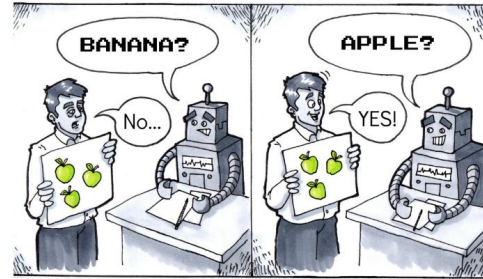
Next up



**Supervised Learning**



**Supervised Learning**



**Supervised Learning**

Multi-Class  
Data Augmentation