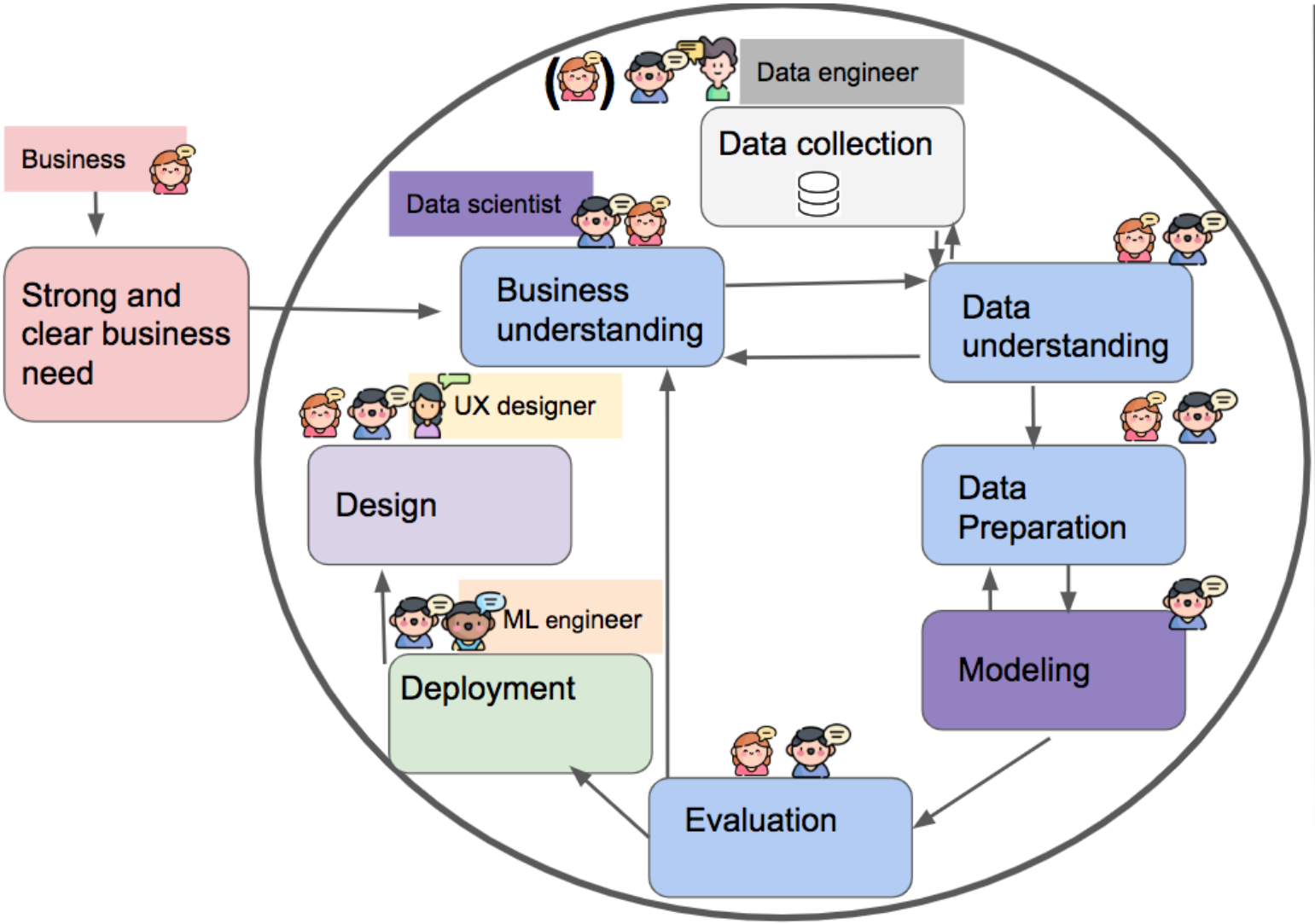
Linear Supervised Learning

M. Ndaoud

STEPS OF THE PROJECT - COMPLETE VIEW OF TASKS AND INTERACTIONS

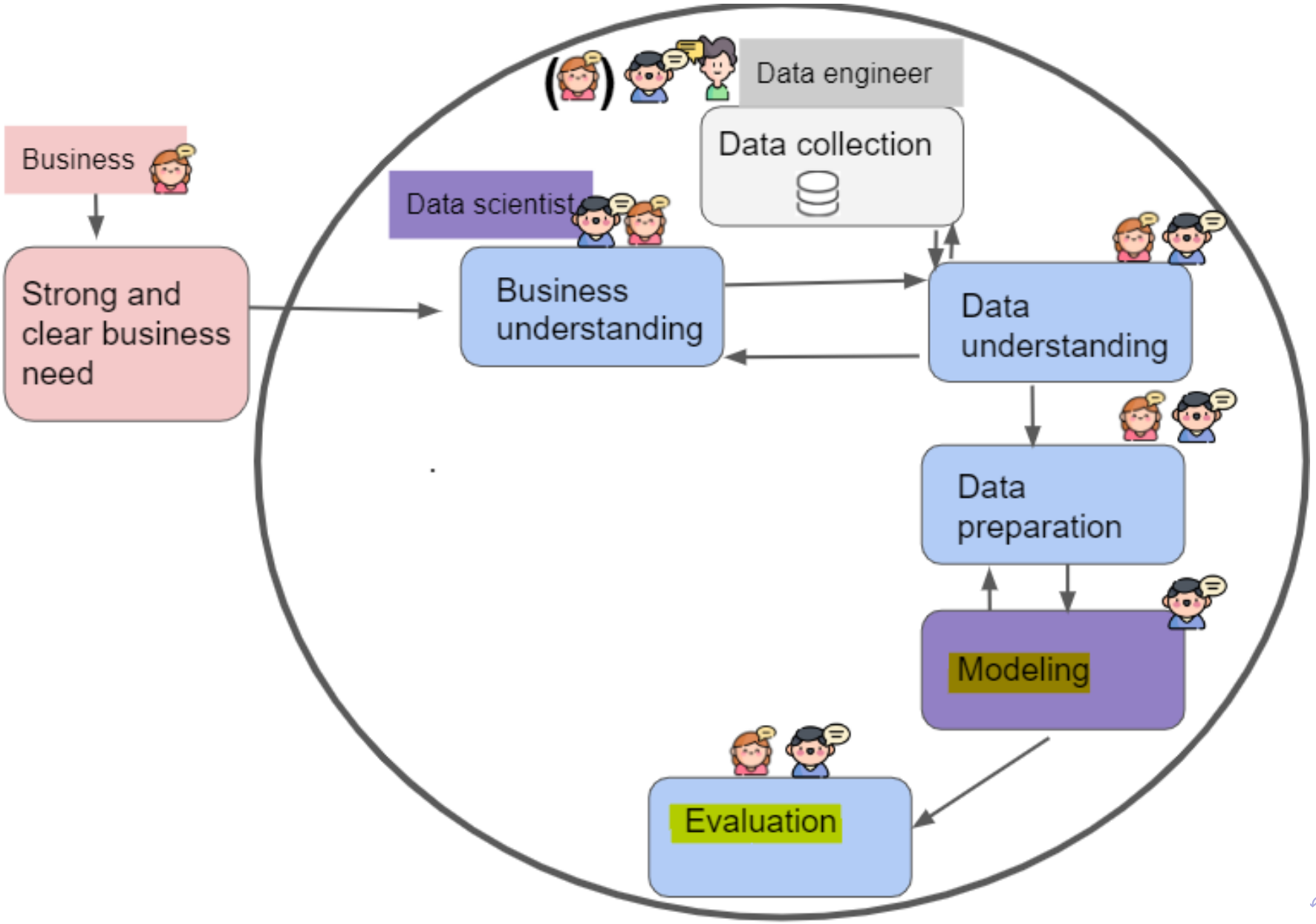


Data science projects are iterative in nature

Fail fast approach
Speed is key

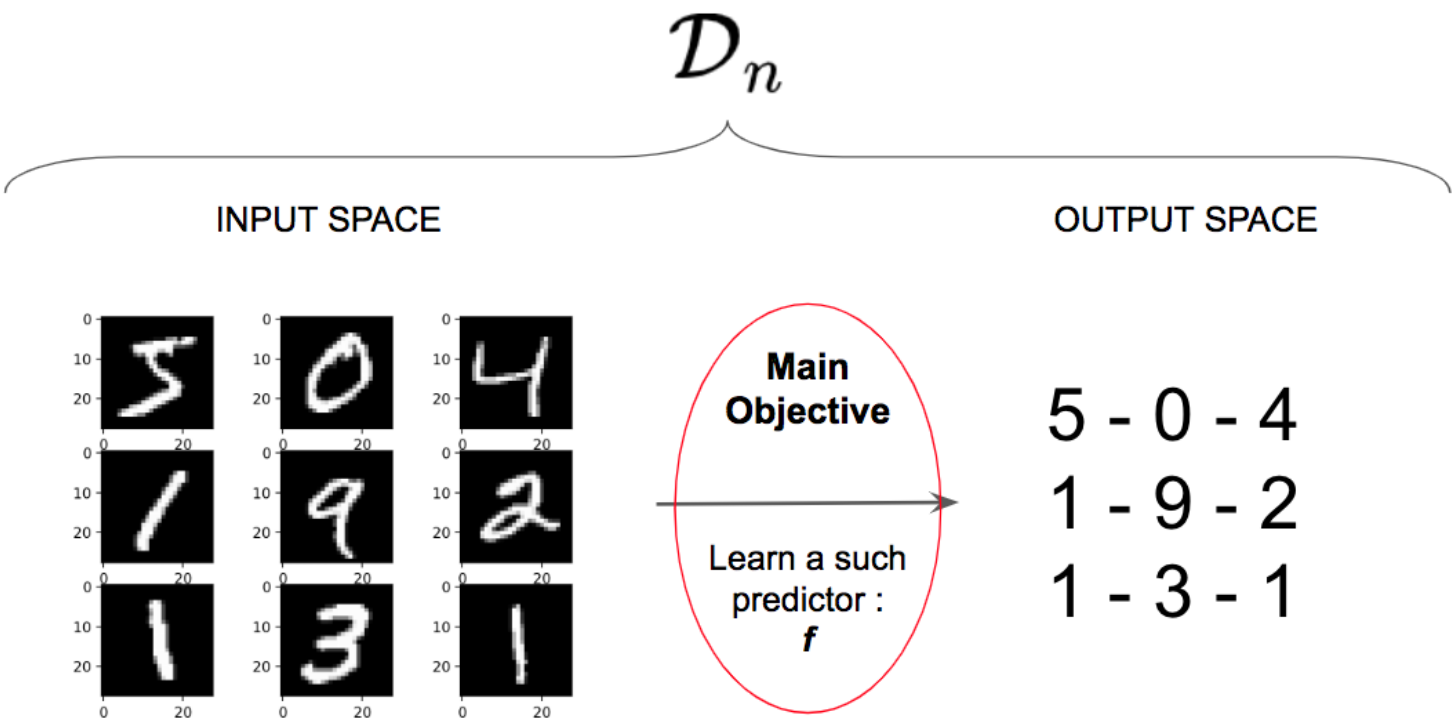
Easy data access boost efficiency

THIS LESSON - FOCUS ON MODEL SELECTION



INTRODUCTION

MNIST Dataset : Digit Recognizer [1]



- How to learn and build the "best" prediction function ?
- What does 'best' mean ?

1. https ://www.kaggle.com/code/ngbolin/mnist-dataset-digit-recognizer

INTRODUCTION

Boston house price prediction [2]

$$\mathcal{D}_n$$

INPUT SPACE

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT
0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	4.98
0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	9.14
0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	4.03
0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	2.94
0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	5.33

Main Objective

Learn a such predictor : ***f***

OUTPUT SPACE

24.0
21.6
34.7
33.4
36.2

prices (standardized)

CRIM: Per capita crime rate by town
CHAS: Charles River dummy variable
NOX: Nitric oxide concentration (parts per 10 million)
RM: Average number of rooms per dwelling
DIS: Weighted distances to five Boston employment centers
RAD: Index of accessibility to radial highways
PTRATIO: Pupil-teacher ratio by town
etc.

2. https ://www.kaggle.com/code/shreayan98c/boston-house-price-prediction

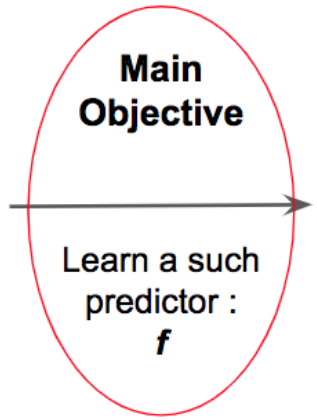
INTRODUCTION

Recession detection in African countries [3]

$$\mathcal{D}_n$$

INPUT SPACE

pop	emp	emp_to_pop_ratio	hc	agriculture	fish	total_change
12.2	6.2	0.5	1.8	220.6	1262.7	0.2
29.8	15.4	0.5	1.5	220.6	1262.7	0.2
55.8	25.3	0.5	1.7	220.6	1262.7	0.2
15.9	5.3	0.3	1.6	220.6	1262.7	0.2
7.8	3.5	0.4	1.8	220.6	1262.7	0.2



OUTPUT SPACE

Growth bucket

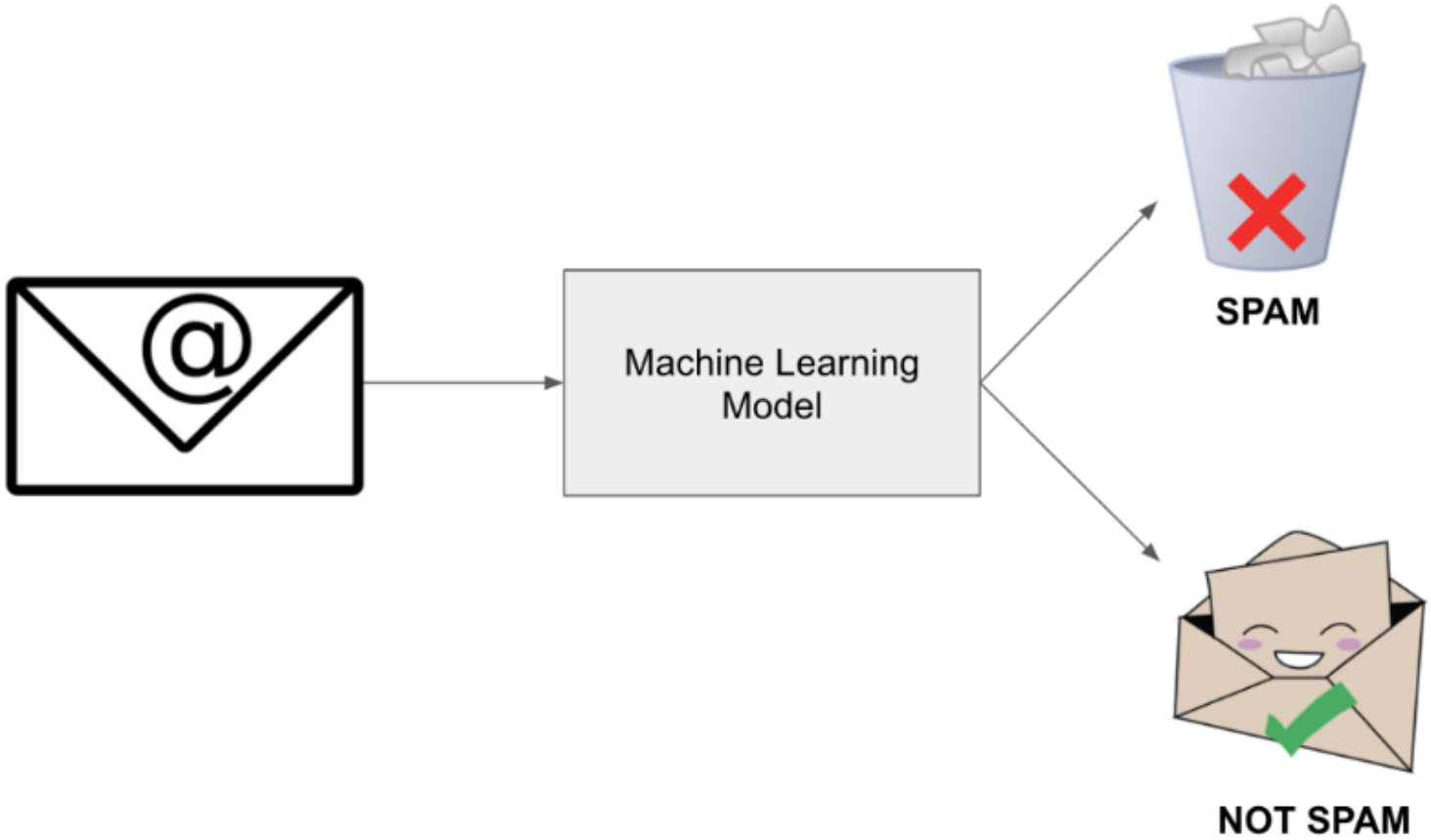
0
1
0
0
0

pop: Population (in millions)
emp: Number of persons engaged (in millions)
emp_to_pop_ratio: Ratio of Employed Persons to Total Population
hc: Human capital index
agriculture: Year-on-Year Percentage Change - Agriculture
etc.

3. https ://www.kaggle.com/code/maistrishid/recession-detection-in-african-countries

Contents

SPAM DETECTION

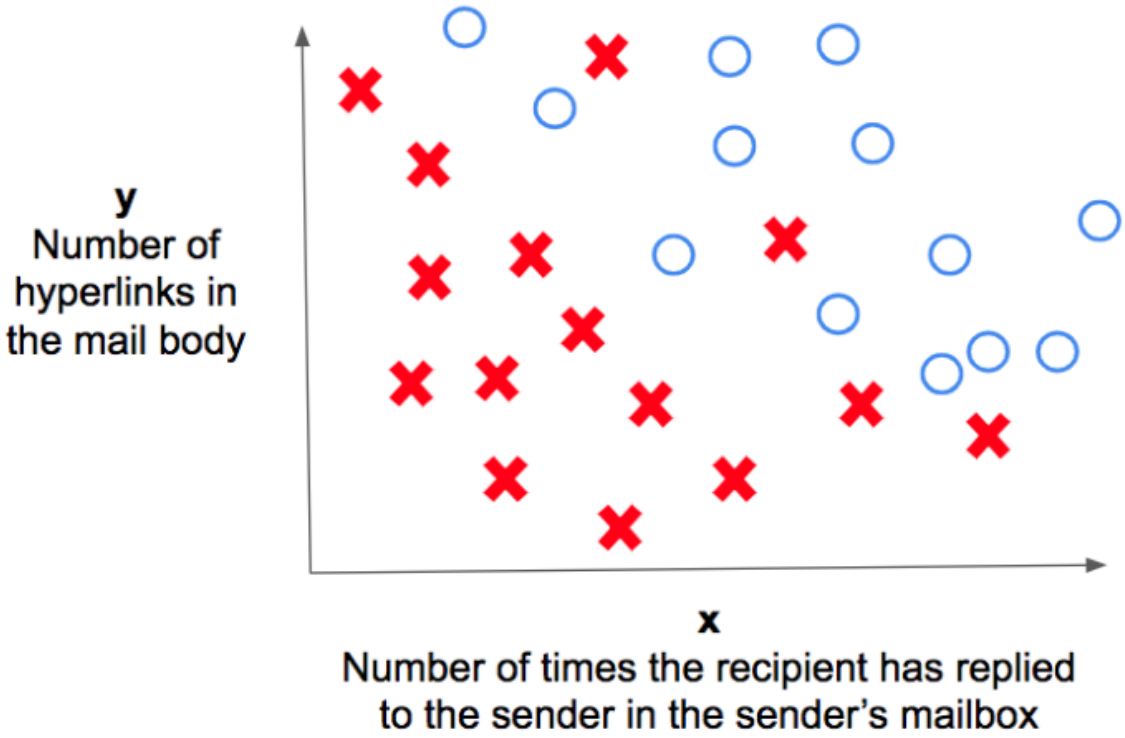


2D SPAM DETECTION MODEL

Example : You have to create a simplified (2D) spam detection ML model

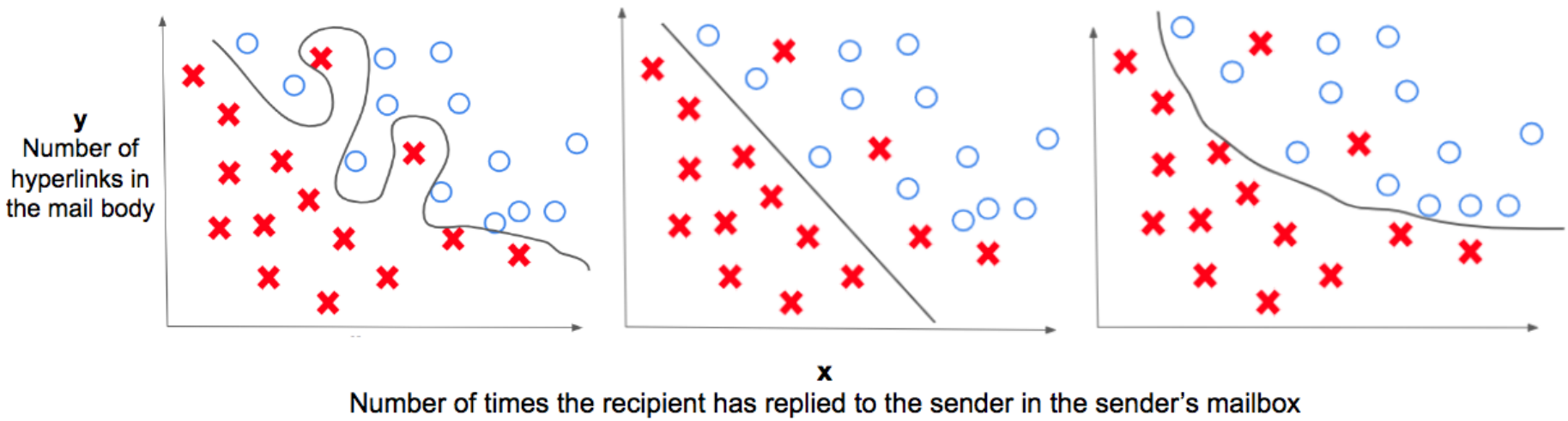
Spam :

Not spam :

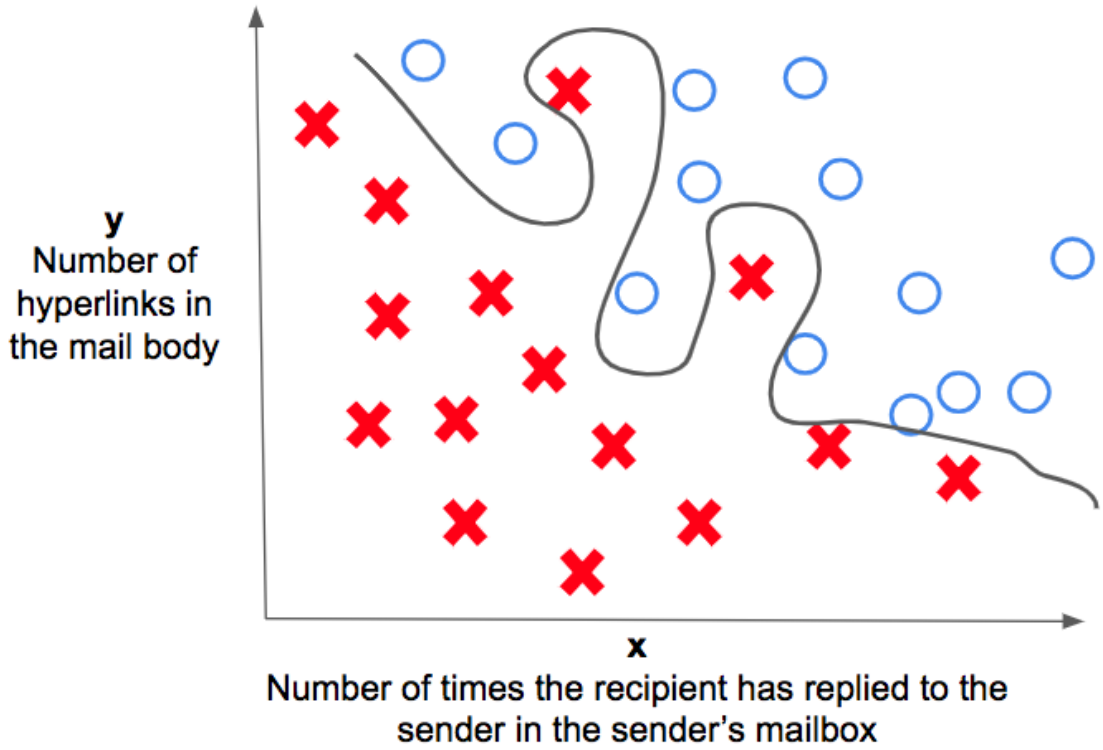


DECISION BOUNDARIES

What is the best model for this 2D features problem ?



OVERFITTING

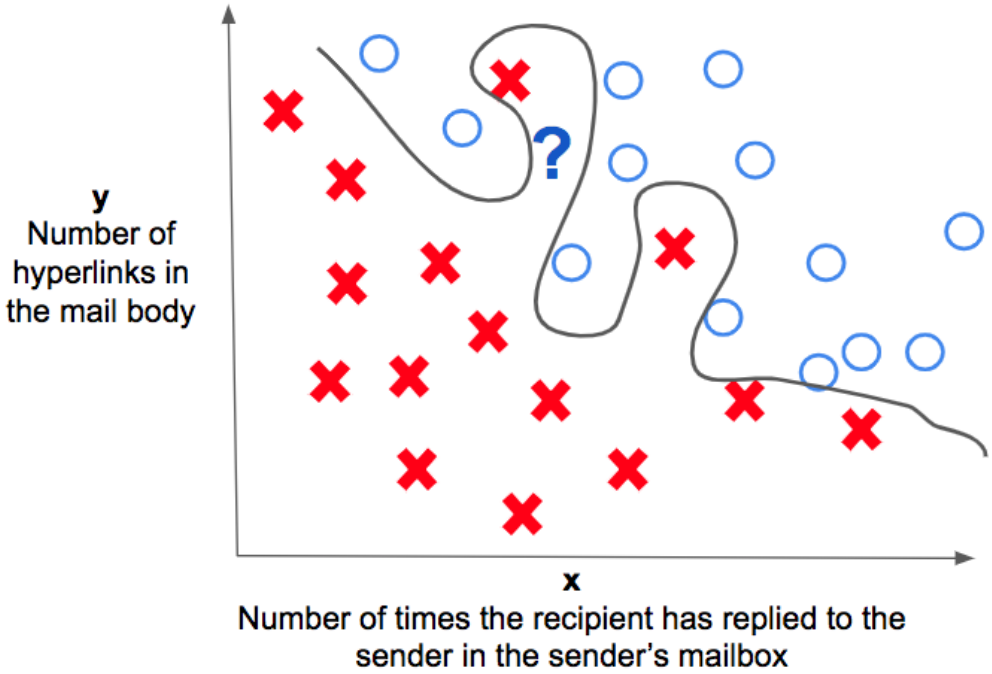


- It fits the data perfectly !
- Example : Complex classifier. Deep neural networks.

OVERFITTING : NEW QUERY POINT

What will happen with a new dataset of emails ?

- There is a high chance that the question mark won't be a spam. . .

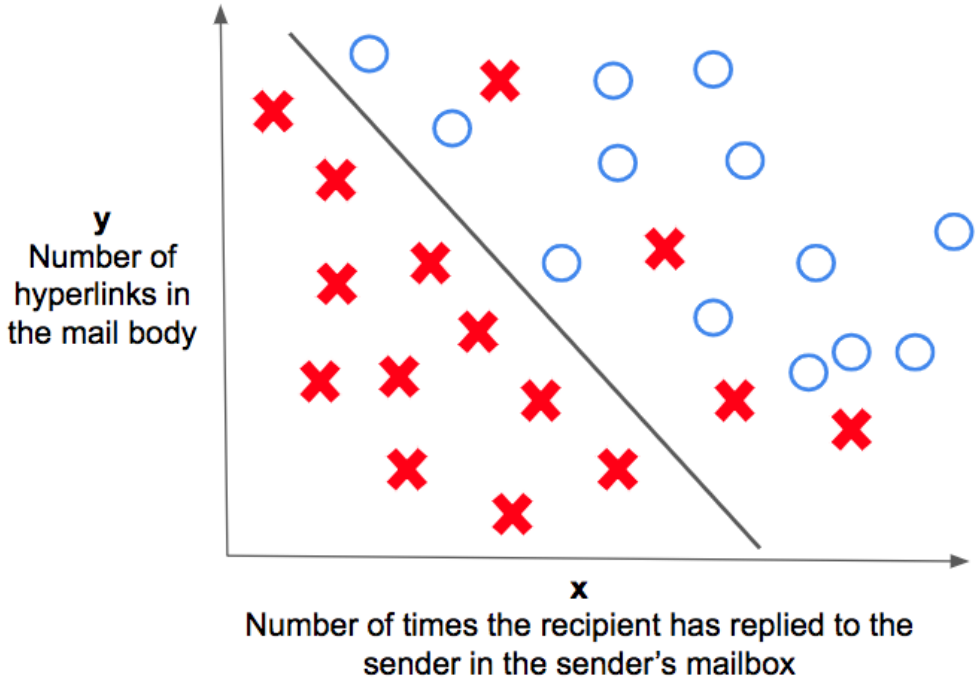


- **Variance :** The error due to sensitivity to small fluctuations in the dataset.
- **High variance** $\Longrightarrow$ **Overfitting the data** : We fitted too well on the train dataset !

OVERFITTING : ACTION

We were **too specific** : we need to **generalize** a bit more !

UNDERFITTING

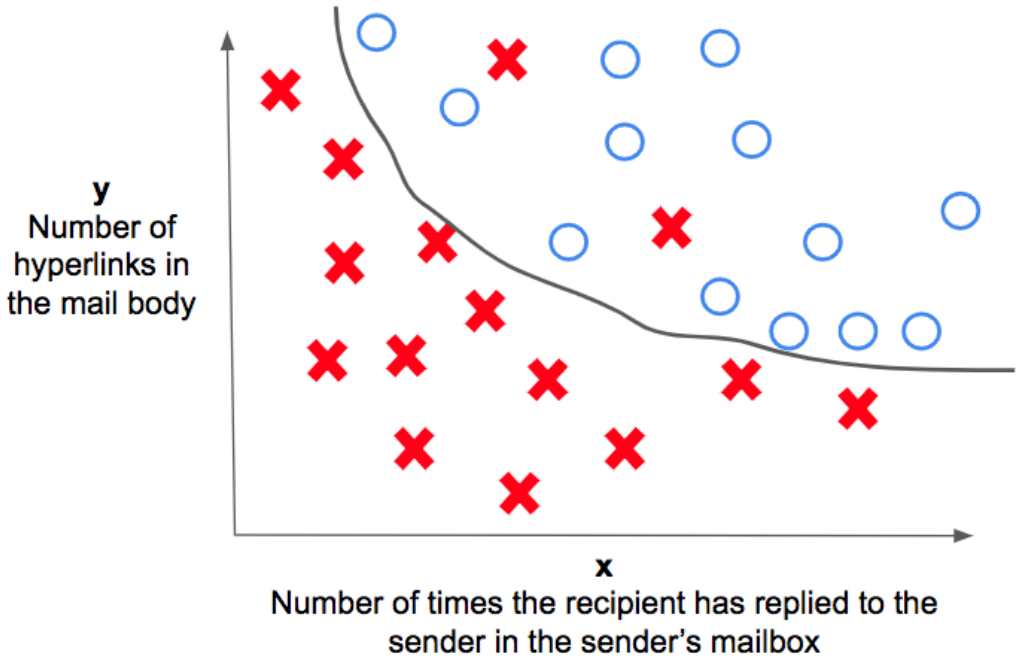


- **Bias :** the **inability** for a machine learning method to **capture the true relationship** between data.
- **High bias** $\Longrightarrow$ **Underfitting the data** : we have bad results on the dataset used to train our model.
- Example : Simple classifier as a logistic regression to fit these data.

UNDERFITTING : ACTION

We were too **general** : we need to build a model that learns better from the train dataset

GOOD FIT



- **"Somewhere in between"** : a model that learns well from the train dataset, but also, **generalizes well**.
- Example : medium level complexity classifier, such as trees.

BIAS AND VARIANCE DECOMPOSITION

Bias and **Variance** are error measures based on average performance across possible training sets.

$$\boxed{Error = Bias + Variance}$$

As shown in the previous course, the error of an estimator can be decomposed between its bias and its variance.

With a **machine learning model** :

- The estimator is the predicted function
- The global error is its empirical risk

BIAS AND VARIANCE DEFINITIONS

Variance :
The error due to **sensitivity to small fluctuations in the dataset**.

A high variance model :

- learns too much from the data it has been trained on.
- Considers the noise as something to learn from while it should not.

Bias :
The **error due to wrong/in-accurate assumptions** that the learning algorithm makes during training.

It is the difference between the average prediction of our model and the correct value which we are trying to predict.

A high bias model :

- Can cause an algorithm to miss the correct relationship between features and the target output.

LINK BETWEEN OVERFITTING / UNDERFITTING AND VARIANCE / BIAS

- **Overfitting :**

Occurs when the model **captures the noise and the outliers in the data** along with the underlying pattern.

- **Noise** : unexplained and random variation inherent to the data or introduced by variables of no interest.

These models are usually complex.

These models usually have high variance and low bias.

- **Underfitting :**

Occurs when the model is **unable to capture the underlying pattern of the data**.

These models are usually simple which are unable to capture the complex patterns in the data.

These models usually have a low variance and a high bias.

EVALUATE BIAS AND VARIANCE

Easy to visualize bias and variance with 2 features.

How to determine bias and variance in high dimensional problems ?

Different metrics of errors on different datasets **hold-out**

- Train set
- Test set

EVALUATE ERROR (BIAS AND VARIANCE) : HOLD-OUT

Hold-out : when you split up your dataset into a 'train' and 'test' set, after randomly shuffle it.

Total number of examples

Train set	Test set
Used to **train** the model (used to fit the parameters of the model)	Used to see how well that model **performs on unseen data.**
Common split : 80% train set	Common split: 20% test
Used to evaluate **bias**	Comparison of both: to evaluate **variance**

In practice :

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df, y, test_size=0.2)
```

EVALUATE BIAS AND VARIANCE : HOLD-OUT

How to determine bias and variance in **high dimensional problems** ?

EVALUATE BIAS AND VARIANCE : HOLD-OUT

How to determine bias and variance In high dimensional problems ?

- 1st : What is the performance of a human ?
  - Example :
    Email classification : human level performance gets nearly 0% error.

Train set error	2%
Test set error	13%
Conclusion	high variance

EVALUATE BIAS AND VARIANCE : HOLD-OUT

How to determine bias and variance In high dimensional problems ?

- 1st : What is the performance of a human ?
  - Example :

    Email classification : human level performance gets nearly 0% error.

Train set error	2%	17%
Test set error	13%	18%
Conclusion	high variance	high bias

EVALUATE BIAS AND VARIANCE : HOLD-OUT

How to determine bias and variance In high dimensional problems ?

- 1st : What is the performance of a human ?
  - Example :

    Email classification : human level performance gets nearly 0% error.

Train set error	2%	17%	17%
Test set error	13%	18%	30%
Conclusion	high variance	high bias	high bias and high variance

EVALUATE BIAS AND VARIANCE : HOLD-OUT

How to determine bias and variance In high dimensional problems ?

- 1st : What is the performance of a human ?
  - Example :

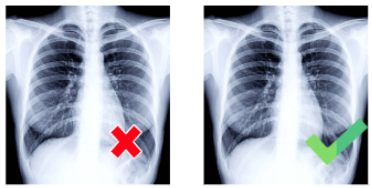
    Email classification : human level performance gets nearly 0% error.

Train set error	2%	17%	17%	2%
Test set error	13%	18%	30%	4%
Conclusion	high variance	high bias	high bias and high variance	low bias and low variance

EVALUATE BIAS AND VARIANCE : HOLD-OUT

How to determine bias and variance In high dimensional problems ?

- 1st : What is the performance of a human ?
  - Example :
    Cardiovascular abnormalities (chest x-rays image).
    Quick visual assessment by a radiologist gets nearly 15% error.



Train set error	17%
Test set error	18%
Conclusion	low bias and low variance

EVALUATE BIAS AND VARIANCE : HOLD-OUT

How to determine bias and variance in high dimensional problems ?

- **Train set error :**

  How well you are fitting on your training examples ?

  Do we have a bias problem ?

- **Train set error and Test set error :**

  How well you are fitting on new examples ?

  How bad the variance is ? = How much does your error go from the training set to the test set ?

What if one subset of our data has only spam emails ?

Example : train set full of spam emails.

EVALUATE BIAS AND VARIANCE : HOLD-OUT

What if one subset of our data has only spam emails ?

- It will result in **overfitting**, even though we're trying to avoid it !

$\longrightarrow$ The hold-out process may not be enough to avoid problems.

$\longrightarrow$ This is where **cross validation** comes in.
The accuracy of our results will be a better indicator for the overall emails !

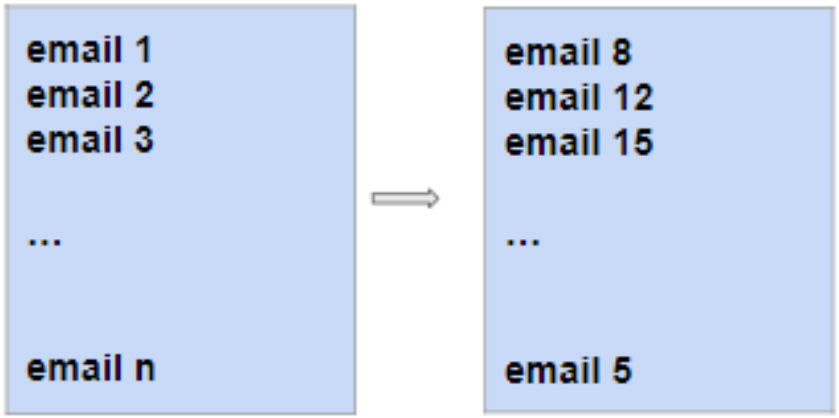
Benefits of cross validation

- Less sensitive to the **data split**.
- The variance of errors is reduced.

CROSS VALIDATION WITH A K-FOLD EVALUATION

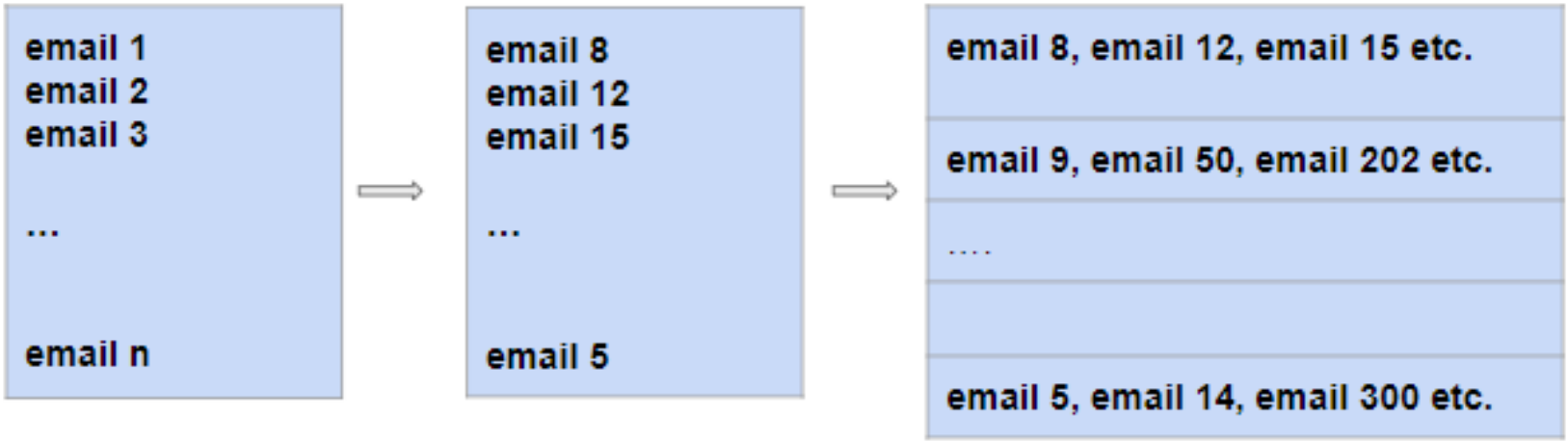
Very similar to train/test split, but it's applied to more subsets.

1. Shuffle the dataset randomly.



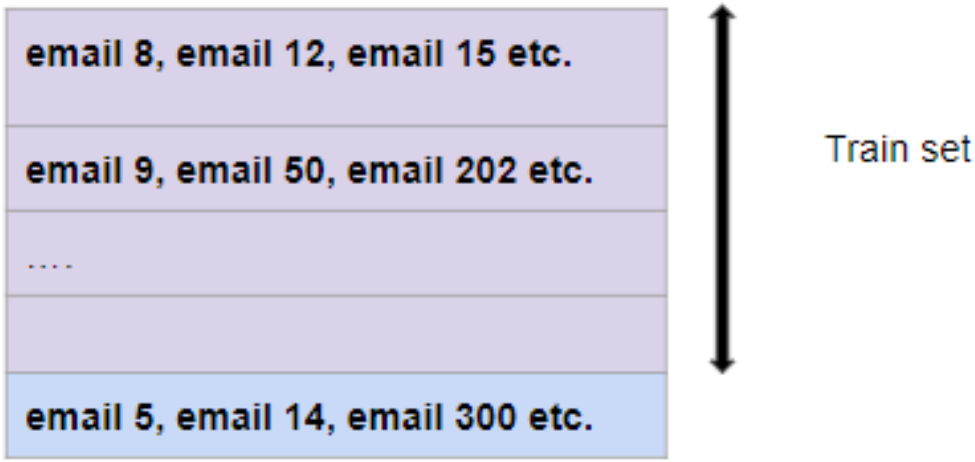
CROSS VALIDATION WITH A K-FOLD EVALUATION

1. Shuffle the dataset randomly.
2. Split data into k subsets (folds).
   - example : 5 folds



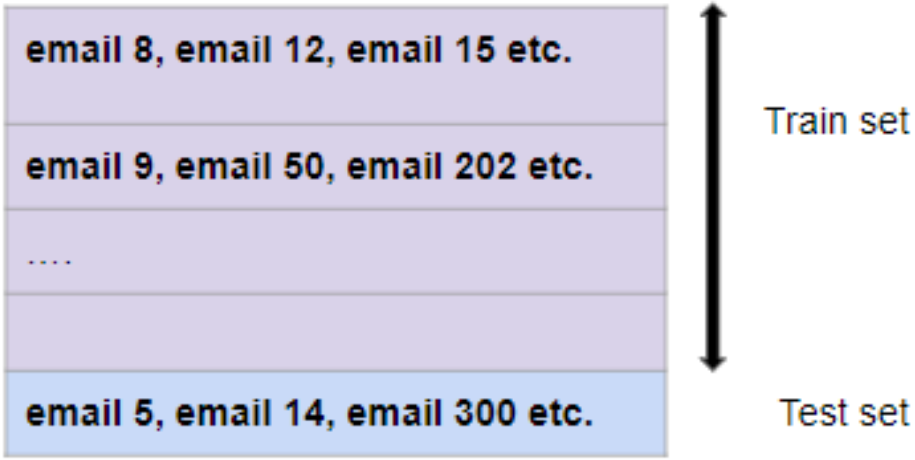
CROSS VALIDATION WITH A K-FOLD EVALUATION

1. Shuffle the dataset randomly.
2. Split data into k subsets (folds).
   - example : 5 folds
3. Train on k-1 one of those subset.



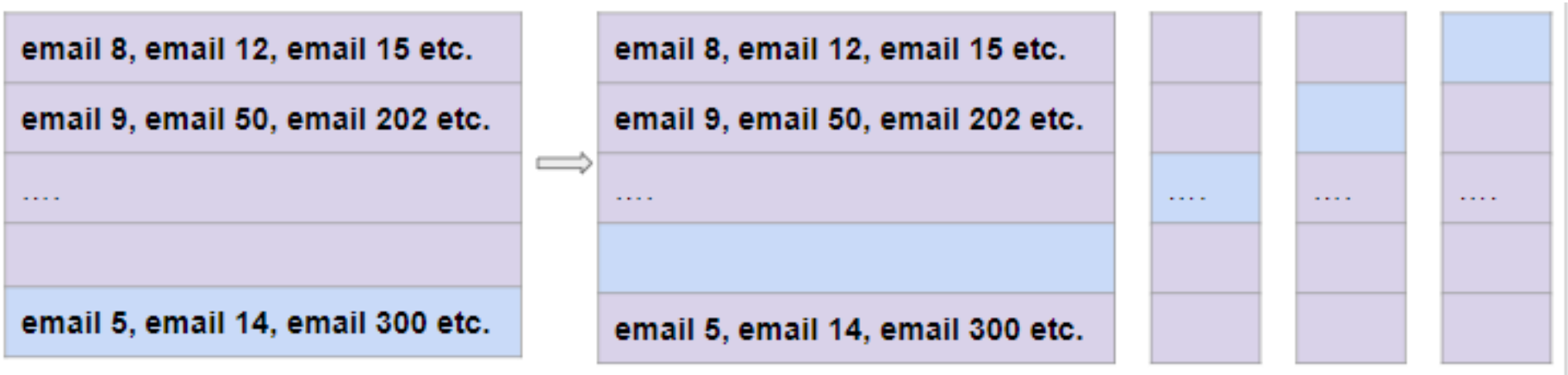
CROSS VALIDATION WITH A K-FOLD EVALUATION

1. Shuffle the dataset randomly.
2. Split data into k subsets (folds).
   - example : 5 folds
3. Train on k-1 one of those subset.
4. Hold the last subset for test.



CROSS VALIDATION WITH A K-FOLD EVALUATION

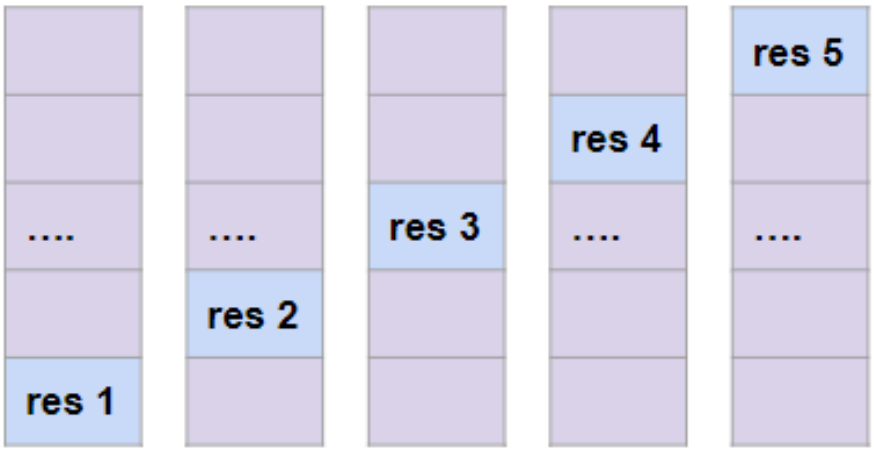
1. Shuffle the dataset randomly.
2. Split data into k subsets (folds).
   - example : 5 folds
3. Train on k-1 one of those subset.
4. Hold the last subset for test.
5. Reiterate k times the procedure by choosing another fold for the test set. $\longrightarrow$ KEY !



CROSS VALIDATION WITH A K-FOLD EVALUATION

Very similar to train/test split, but it's applied to more subsets.

1. Shuffle the dataset randomly.
2. Split data into k subsets (folds).
   - example : 5 folds
3. Train on k-1 one of those subset.
4. Hold the last subset for test.
5. Reiterate k times the procedure by choosing another fold for the test set.
6. Average the test metrics from those k experiments (do the same with the train metrics).



$$\frac{1}{k}\sum_{i=1}^{k} metric_i$$

The assessment of the model will be **more accurate** compared to the hold-out !

- It matters less how the data gets divided
  $\longrightarrow$ every data point : 1 time in test set, k-1 times in the train set.
- The variance of the resulting estimate is reduced as k is increased.

CROSS VALIDATION WITH A K-FOLD EVALUATION

In practice :

```
from sklearn.model_selection import KFold
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

k = 5
kf = KFold(n_splits=k, random_state=None, shuffle=True)

model = LogisticRegression(solver= 'liblinear')

acc_score = []

for train_index , test_index in kf.split(X):
  X_train , X_test = X.iloc[train_index,:],X.iloc[test_index,:]
  y_train , y_test = y.iloc[train_index] , y.iloc[test_index]
  model.fit(X_train,y_train)
  pred_values = model.predict(X_test)
  acc = accuracy_score(pred_values , y_test)
  acc_score.append(acc)

avg_acc_score = sum(acc_score)/k
```

A **for** loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

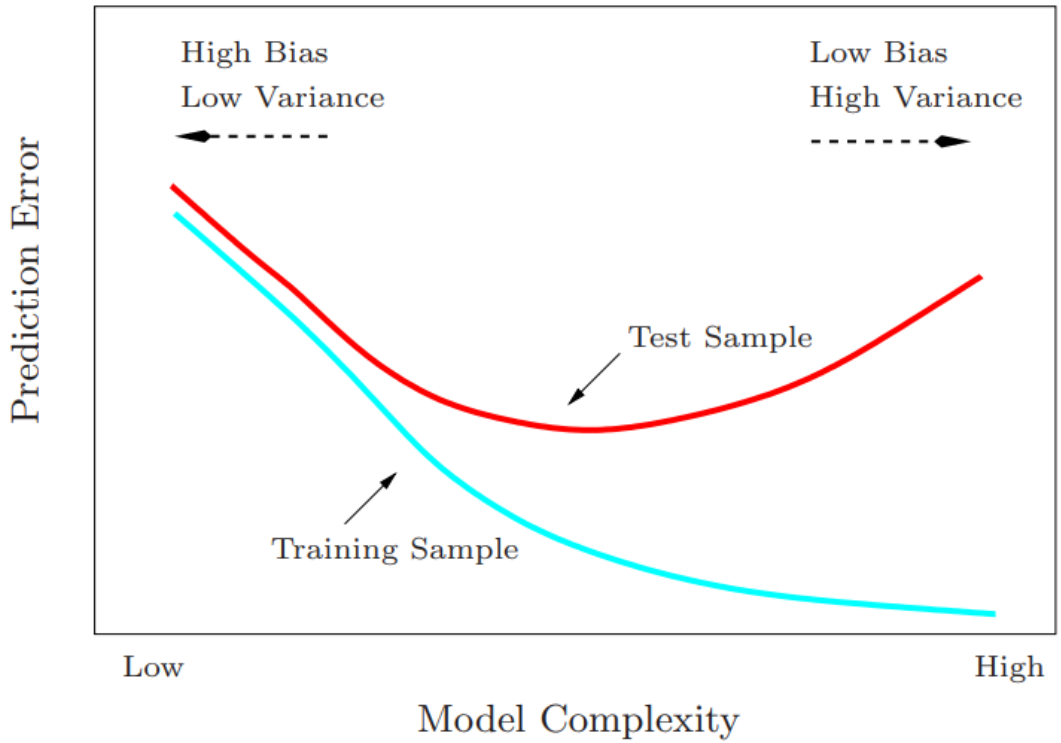
In practice :

```
fruits = ["apple",
"banana", "cherry"]

for x in fruits:

  print(x)
```

FIRST TECHNIQUES TO REDUCE OVERFITTING (WILL BE COMPLETED IN THE NEXT LESSONS)

What to do in case of overfitting ?

- Increase training data.
- Reduce model complexity.



6

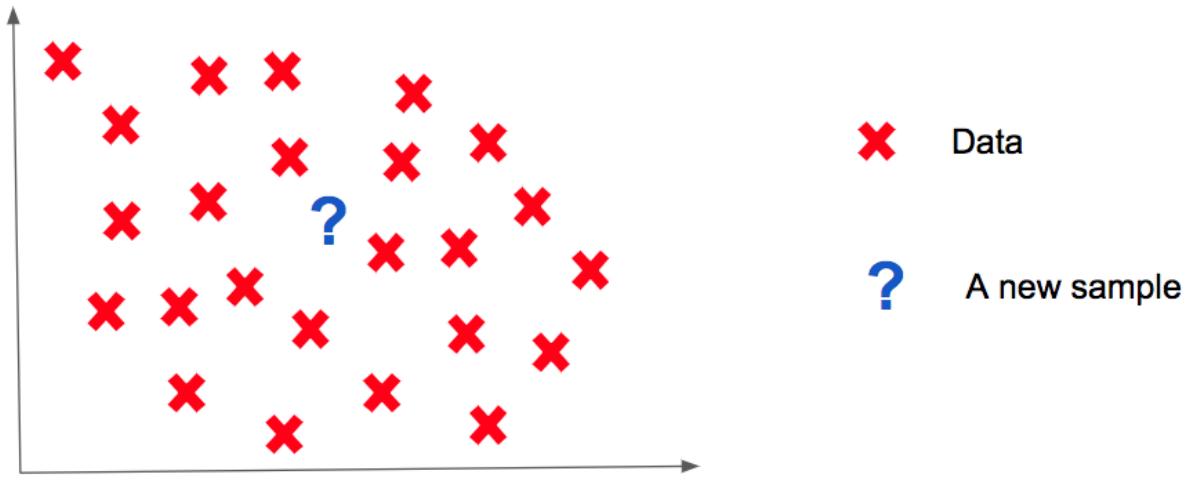
6. Hastie, T., Tibshirani, R., Friedman, J. H., Friedman, J. H. (2009). The elements of statistical learning : data mining, inference, and prediction (Vol. 2, pp. 1-758). New York : springer.

Contents

K-NEAREST NEIGHBORS INTUITION

- Part 1 $\Longrightarrow$ need to choose a prediction **ensemble** ($G \in \mathcal{F}(X, Y)$)

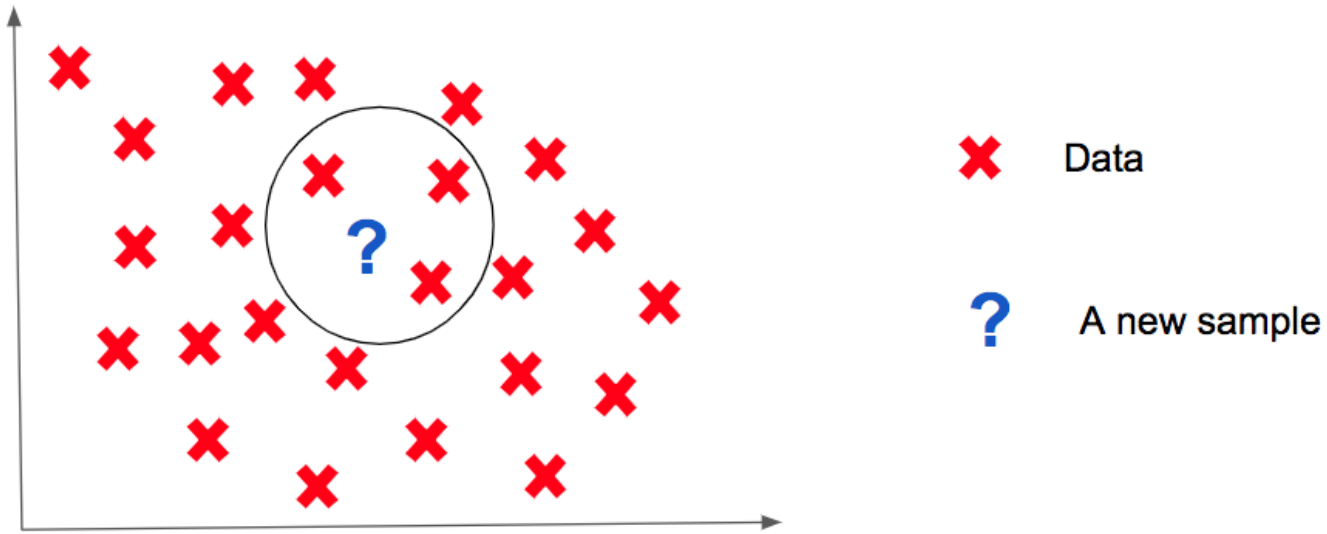
The **most simple** and **popular** is the family of k nearest neighbors (or KNN) predictor.



If we want to generalize a property to our new point *(classification or regression)*.

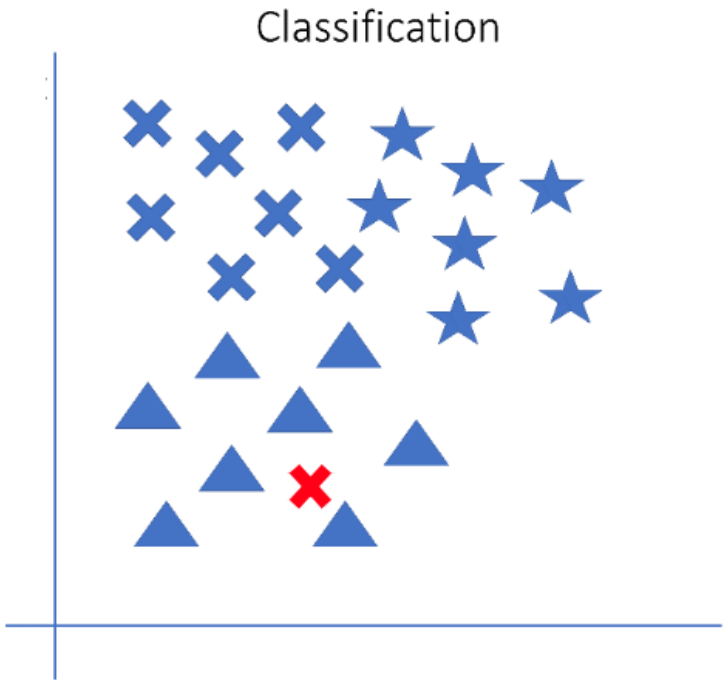
What can we do ?

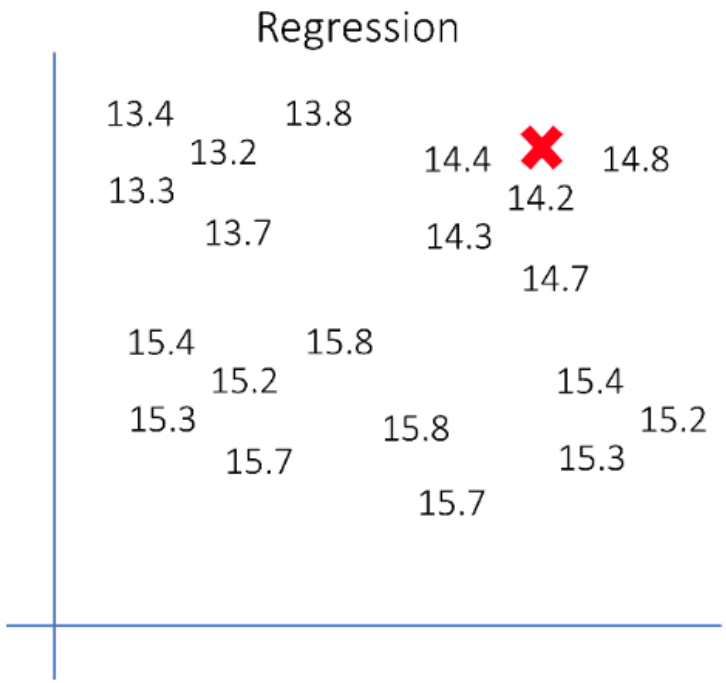
K-NEAREST NEIGHBORS INTUITION



Look and learn from it closest neighbors !

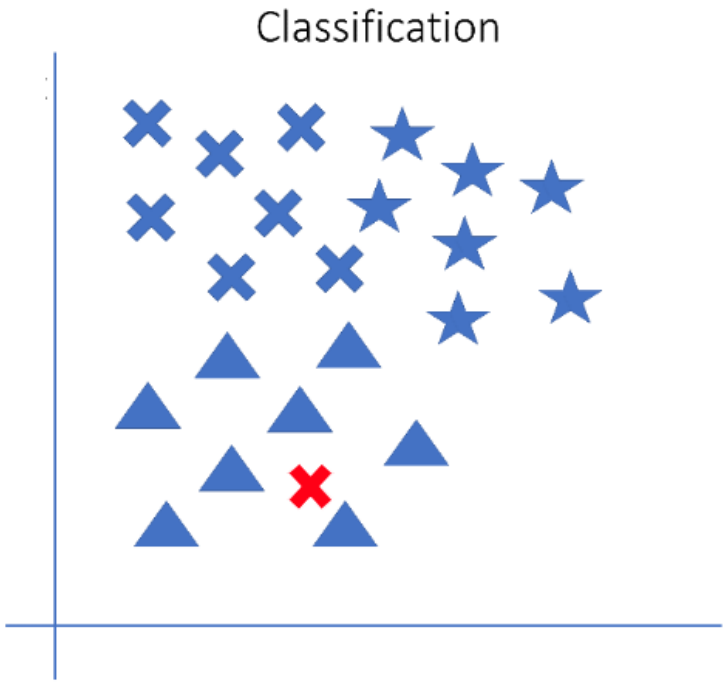
K-NEAREST NEIGHBORS INTUITION

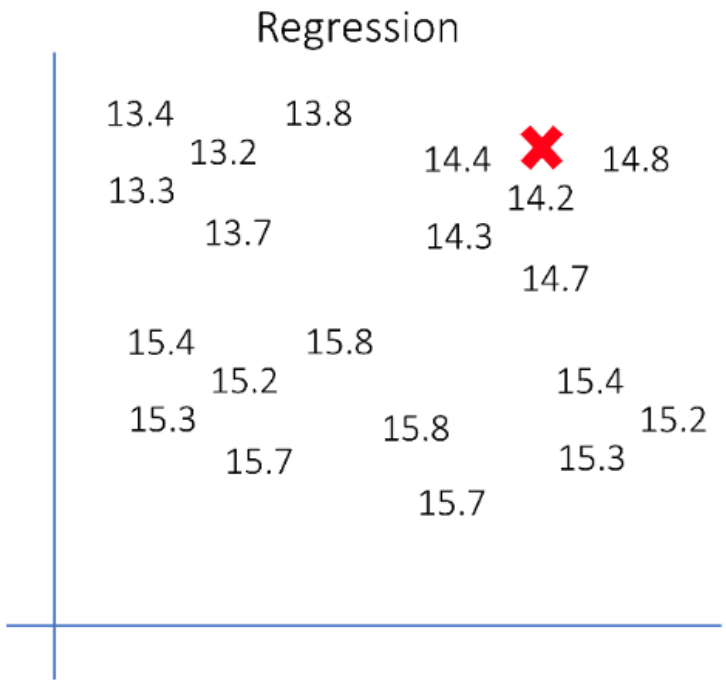




How to define the neighborhood of a point ?

K-NEAREST NEIGHBORS INTUITION





How to define the neighborhood of a point ?

- Number of neighbors
- Distance to quantify the proximity of two points

K-NEAREST NEIGHBORS FOR REGRESSION

Algorithm 1 K-NN for Regression

Hyperparameters :
- d : distance
- k : number of neighbors

Training step :
- Keep in memory the sample $\mathcal{D}_n = \{(X_1, Y_1), ..., (X_n, Y_n)\}$

Predicting step : with a new query point x_0
- Find the k training points, $x(r)$ with $r = 1...k$, closest in distance to x_0.

$$d(x_0, x_{(0)}) \leq \cdots \leq d(x_0, x_{(k)})$$

- Predict the target of x_0 $(f(x_0))$ using average among the found neighbors targets

$$f(x_0) = \frac{1}{k} \sum_{i=r}^{k} y_{(r)}$$

K-NEAREST NEIGHBORS FOR CLASSIFICATION

Algorithm 2 K-NN for Classification

Hyperparameters :
- d : distance
- k : number of neighbors

Training step :
- Keep in memory the sample $\mathcal{D}_n = \{(X_1, Y_1), ..., (X_n, Y_n)\}$

Predicting step : with a new query point x_0
- Find the k training points, $x(r)$ with $r = 1...k$, closest in distance to x_0.

$$d(x_0, x_{(0)}) \leq \cdots \leq d(x_0, x_{(k)})$$

- Classify x_0 using majority vote among the k found neighbors (ties broken at random)

$$f(x_0) = \mathbb{1}_{(\frac{1}{k}\sum_{i=r}^{k} y_{(r)} \geq 1/2)}$$

K-NEAREST NEIGHBORS - REMARKS

Remarks

- Despite simplicity, KNN is **successful** in large number of problems (e.g. mnist).
- KNN are **memory-based**, and require no model to be fit.
- Choosing an appropriated **distance** (key) is not always straightforward :
  - Qualitative and ordinal features, and how to combine them for mixed data
  - Most of the time $\rightarrow$ **Euclidean distance** with previous **standardization** of the data

Standardization of the data :

"Different attributes are measured on different scales, so if the Euclidean distance formula were used directly, the effect of some attributes might be completely **dominated** *by others that had larger scales of measurement. Consequently, it is usual to normalize all attribute values"* [7]

Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1.

$$x_{stand} = \frac{x - mean(x)}{std(x)}$$

7. Witten, I. H., Frank, E. (2016). Data mining : practical machine learning tools. Morgan Kaufmann Series in Data Management Systems

APPLICATION ON A DATASET

KNN Regression with the Boston House Dataset [8]

INPUTS

1. CRIM - per capita crime rate by town
2. ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS - proportion of non-retail business acres per town.
4. CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
5. NOX - nitric oxides concentration (parts per 10 million)
6. RM - average number of rooms per dwelling
7. AGE - proportion of owner-occupied units built prior to 1940
8. DIS - weighted distances to five Boston employment centres
9. RAD - index of accessibility to radial highways
10. TAX - full-value property-tax rate per $10,000
11. PTRATIO - pupil-teacher ratio by town

[redacted]

13. LSTAT - % lower status of the population

OUTPUT

House Prices

(expressed in $1000s)

8. The dataset for this project originates from the UCI Machine Learning Repository. The Boston housing data was collected in 1978 and each of the 506 entries represent aggregated data about 14 features for homes from various suburbs in Boston, Massachusetts.

APPLICATION ON A DATASET - EVALUATION

KNN Regression with the Boston House Dataset

Dataset : $\mathcal{D}_n = \{(X_1, Y_1), ..., (X_n, Y_n)\}$
Predictor : K-nearest neighbors

- Distance : Euclidian
- k : 5

Evaluation method : cross validation (3 folds)

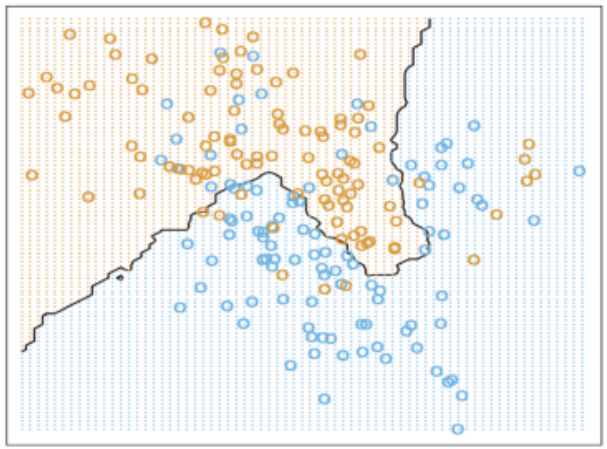
In practice :

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import cross_validate
k = 5
knn = KNeighborsRegressor(n_neighbors=k)
results = cross_validate(knn, X=boston, y=target, cv=3,
scoring="neg_mean_squared_error")
neg_mse_folds = results["test_score"]
mse_folds = - neg_mse_folds
print("mse folds: ", mse_folds)
print("mean mse: ", round(np.mean(mse_folds), 2))
print("std mse: ", round(np.std(mse_folds), 2))
```

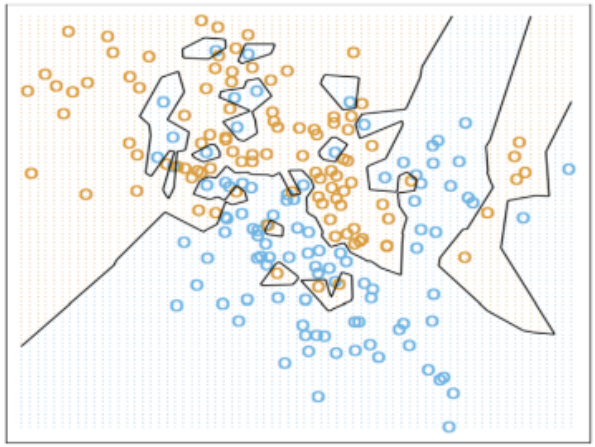
mse folds : [73.12062962 107.20463572 56.71895926]
mean mse : 79.01
std mse : 21.03

K-NEAREST NEIGHBORS - GOOD FIT / OVERFITTING

- What's the impact if we **increase or decrease** the number of neighbors ? (complexity of the model)
- Which of the following scenarios represent is : GOOD FIT / OVERFITTING



15-nearest neighbors



1-nearest neighbors [9]

How to find the best k ?

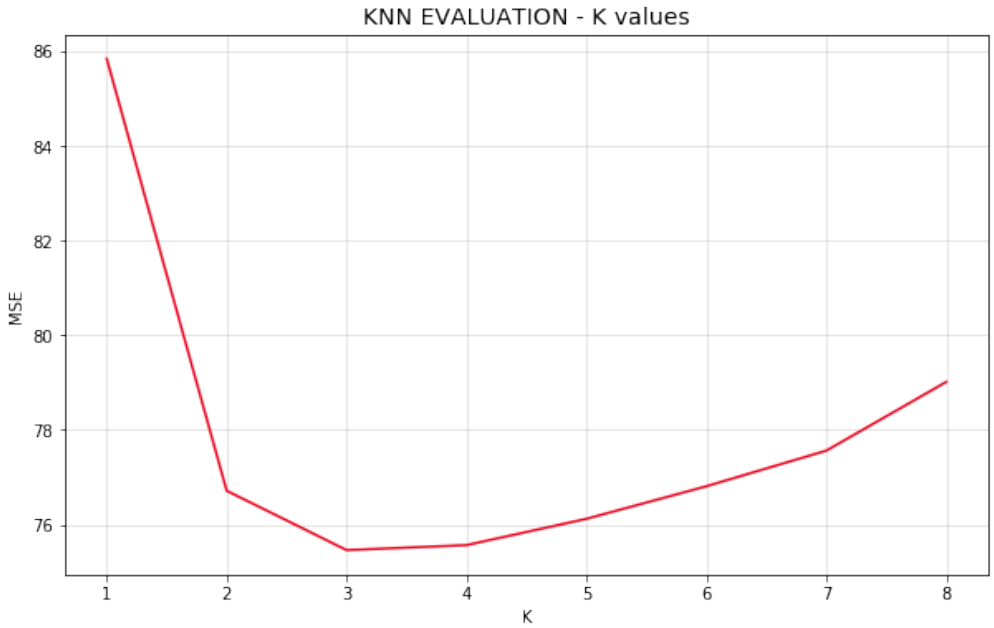
9. Hastie, T., Tibshirani, R., Friedman, J. H., Friedman, J. H. (2009). The elements of statistical learning : data mining, inference, and prediction (Vol. 2, pp. 1-758). New York : springer.

K-NEAREST NEIGHBORS - GRID SEARCH

In most cases, model learning depends on **hyperparameters** (e.g. k for the KNN algorithm).

- The default values chosen by libraries work in most cases.
- In others, it can be useful to optimize them.
- The simplest method is to **try different values** and then choose the one that minimizes the error : **GridSearch mechanism**.

Example : on the Boston house price prediction dataset

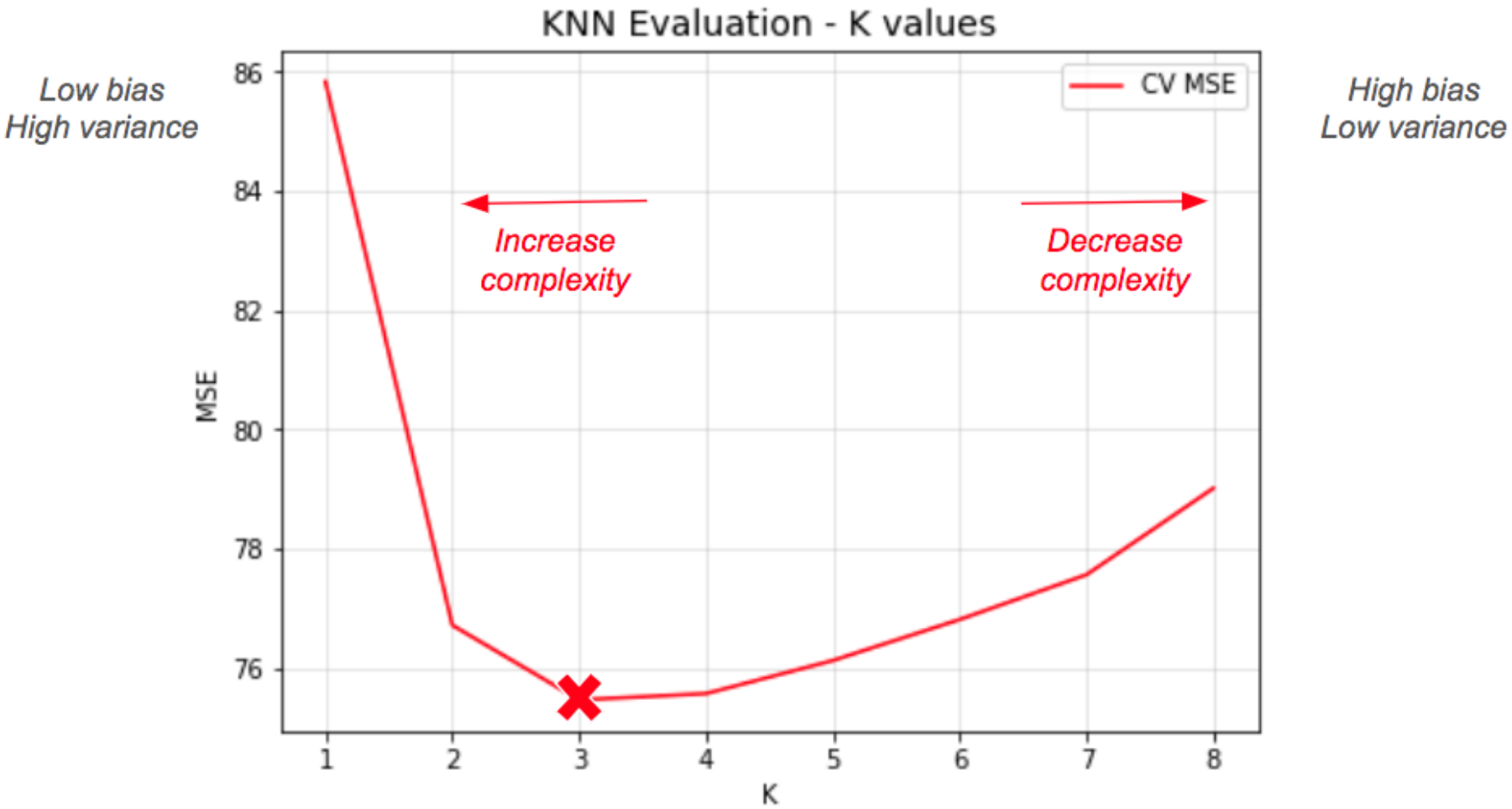


K-NEAREST NEIGHBORS - GRID SEARCH

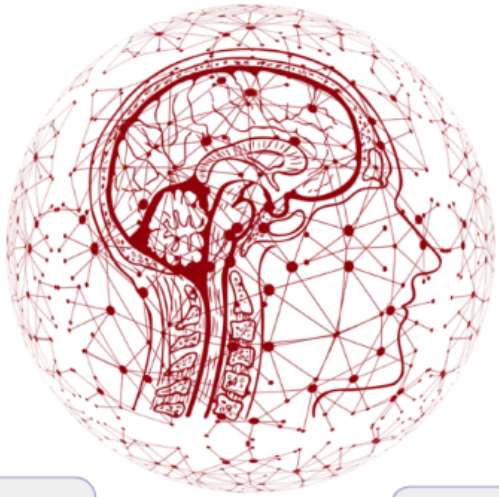
In practice :

```
k_grid = np.arange(1, 9, 1)
output_mean = []
for k in k_grid:
     knn = KNeighborsRegressor(n_neighbors=k)
     results_k = cross_validate(knn, X=boston, y=target, cv=3,
     scoring="neg_mean_squared_error")
     mse_k = - np.mean(results_k["test_score"])
     output_mean.append(mse_k)
```

K-NEAREST NEIGHBORS - INTUITION BEHIND BIAS VARIANCE TRADE-OFF



CONCLUSION

Overfitting
Bias
Variance
K-NN
K-folds
Train set
Test set
Loss function
Underfitting
KNeighborsClassifier

Cross-Validation
hyperparameters
Standardization
train_test_split()
For loop
Classifier
Gridsearch
KNeighborsRegressor
Predictive function
Hold-out
ERM
MSE