ÉCOLE SUPÉRIEURE EN SCIENCES ET TECHNOLOGIES DE L'INFORMATIQUE ET DU NUMÉRIQUE

FUNDAMENTALS OF DATA SCIENCE AND DATA MINING

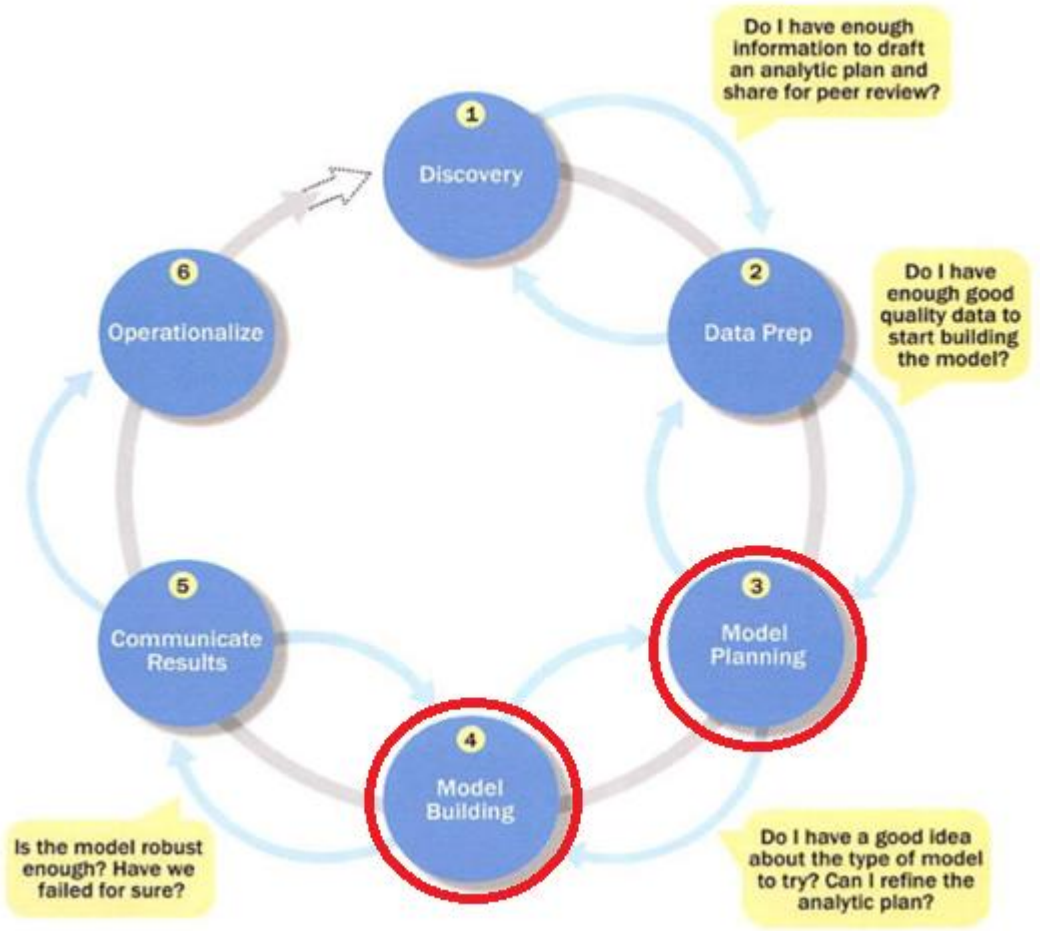
CHAPTER 3:
DATA MINING: GENERALITIES

Dr. Chemseddine Berbague

2023-2024

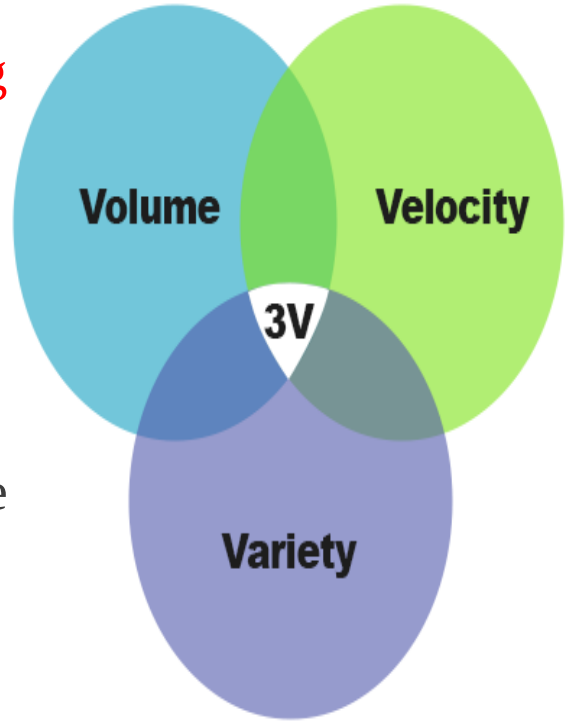
OUTLINE

1. Model planning
2. Model building



PROBLEMATIC

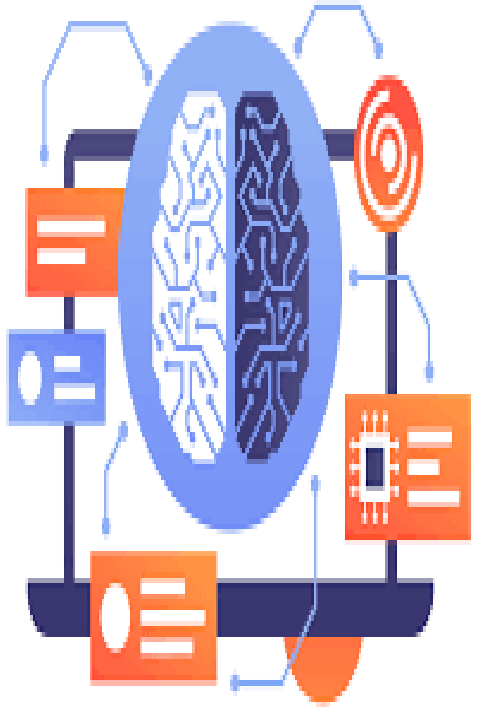
- **Data collection and storage** technologies has led to accumulate vast amounts of data.
- **Traditional data analysis** tools and techniques face hard challenging limitations to handle this data.
- This challenges depend on:
  - The nature of data (non-traditional structure).
  - The objective of the data analysis.
- **Data mining** was developed as **a new box of technologies** that provide sophisticated algorithms for processing large volumes of data.
  - Exploring and analyzing new types of data.
  - Analyzing old types of data in new ways.
  - Build predictive models.



WHAT IS MODEL TRAINING OR BUILDING?

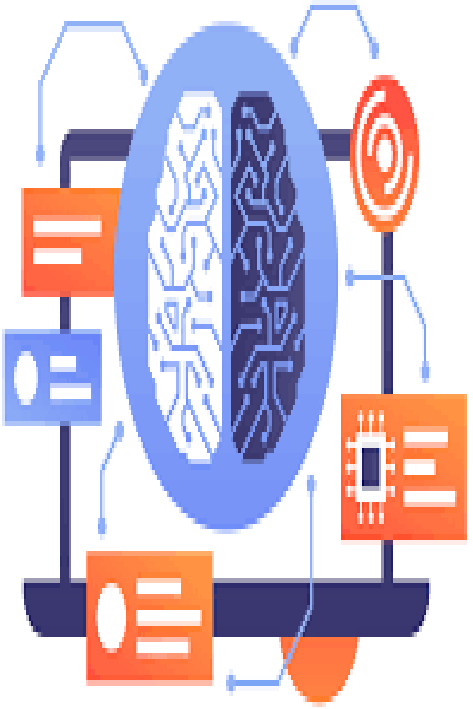
"Model training is the phase in the data science development lifecycle where practitioners try to fit the best combination of weights and bias to a machine learning algorithm to minimize a loss function over the prediction range."

--C3.AI



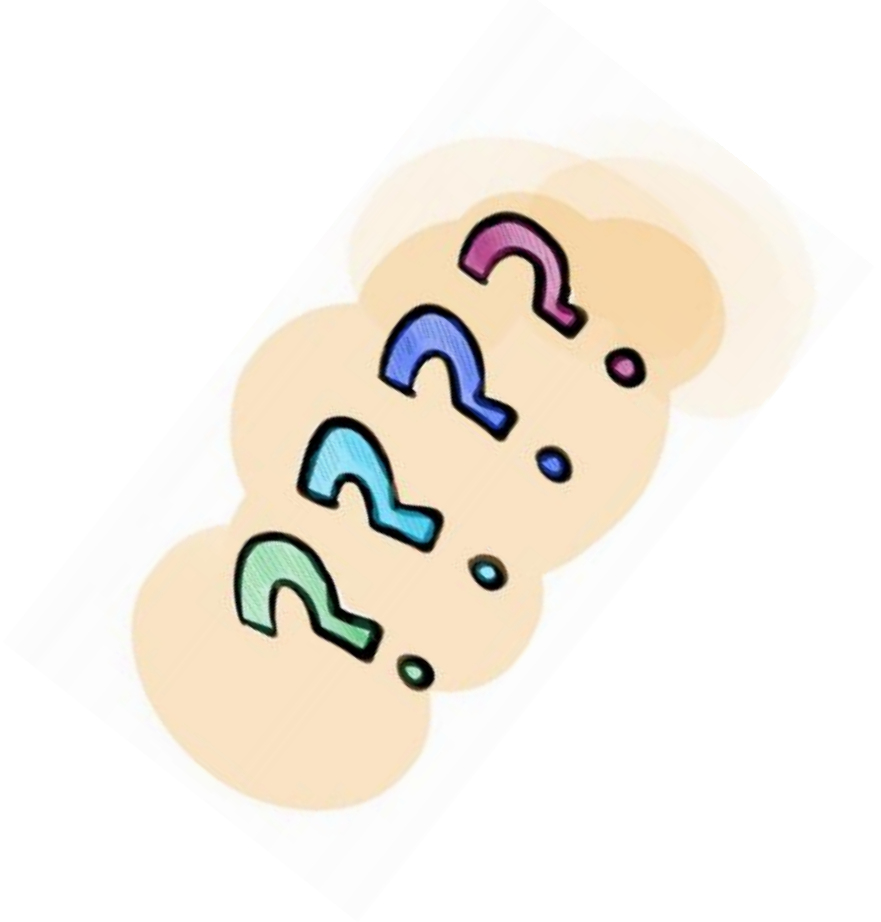
WHAT IS MODEL TRAINING OR BUILDING?

- In **supervised learning** tasks such as ***classification***:
  - The Model training estimates near optimal parameters of the relationship between data features and a target data.
  - While, in **unsupervised learning** such in ***clustering***:
    - The model training estimates near optimal parameters of the relationship between data features themselves.
  - Model training uses a **loss functions** such as the **MSE** to optimize the machine learning algorithms (i.e., to measure how good is the algorithm at estimation).
  - The choice of the loss function depends on (a) the objective of data modeling, (b) type of data and (c) the algorithm.

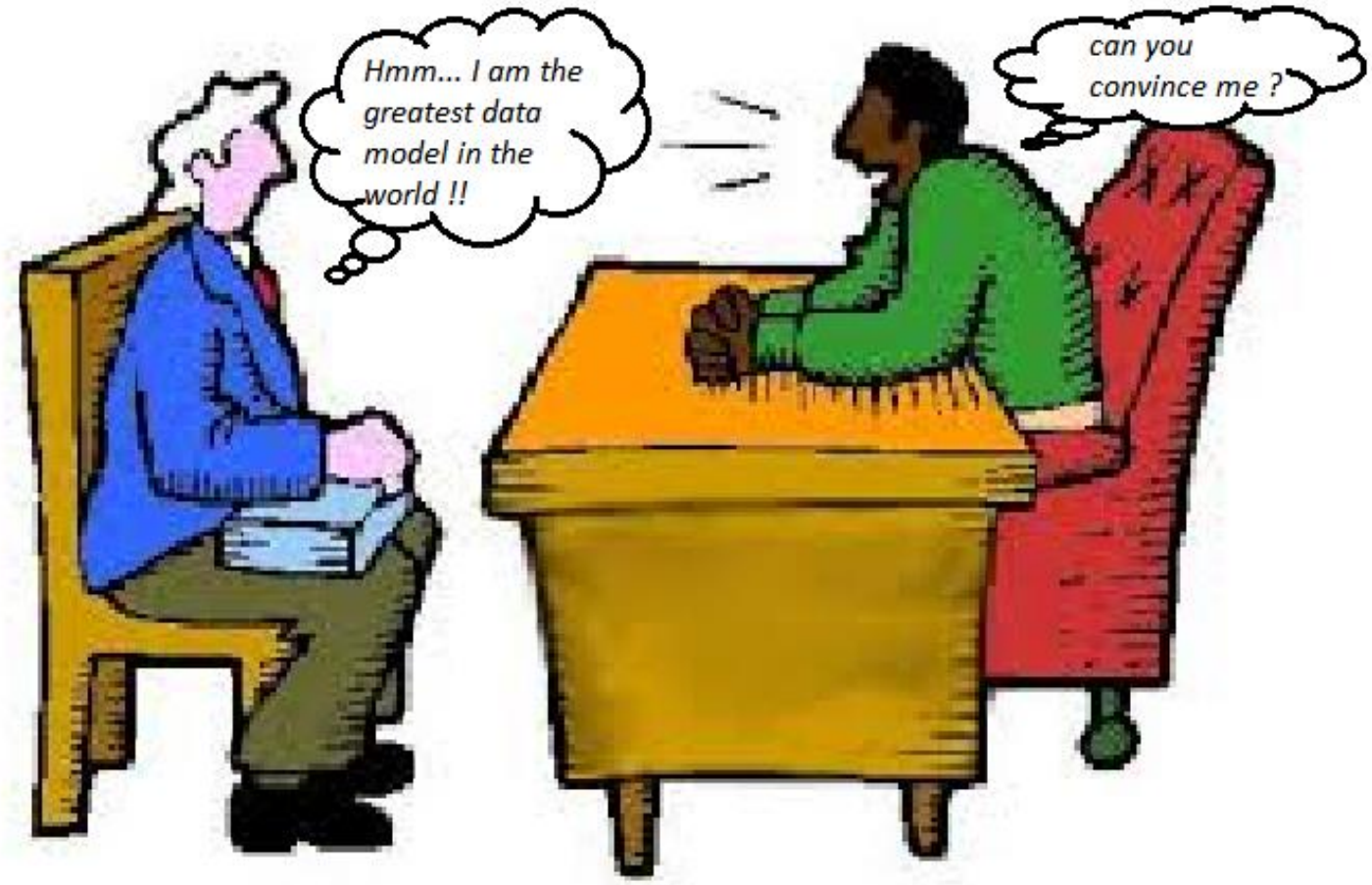


SOME INTERESTING QUESTIONS !

- How to choose the right data model ?
- What are the different training types ?
- How to train a data model ?
- What is a loss function ?
- How to validate a data model ?
- What are some simple examples of data modeling ?



1. Model selection



MODEL PLANNING OR MODEL SELECTION !

"When we have a variety of models of different complexity (e.g., linear or logistic regression models with different degree polynomials, or KNN classifiers with different values of K), how should we pick the right one?"

— Page 22, Machine Learning: A Probabilistic Perspective, 2012.

MODEL SELECTION: CRITERIA

- Model selection is to choose among many candidates, the most appropriate model.
- Model selection can be subject to few criterion:
  - Model features VS stakeholders requirements and constraints.
  - Model performances VS Time and resources.
  - Model performances VS Naive models.
  - Model performances VS Other competitors.
  - Model Complexity, Model Maintainability..Etc.
  - ➢ **Check model planning stage seen in chapter 1.**

- The two main classes of model selection techniques are: (A) **probabilistic measures** and (B) **resampling methods.**

MODEL SELECTION: (A) USING PROBABILISTIC MEASURES

- A model having fewer parameters is supposed to be better at generalization.
  - Akaike Information Criterion (AIC):
  - Bayesian Information Criterion (BIC).
  - Minimum Description Length (MDL).
- Such measures are used with simple linear methods, where the calculation of complexity is tractable.

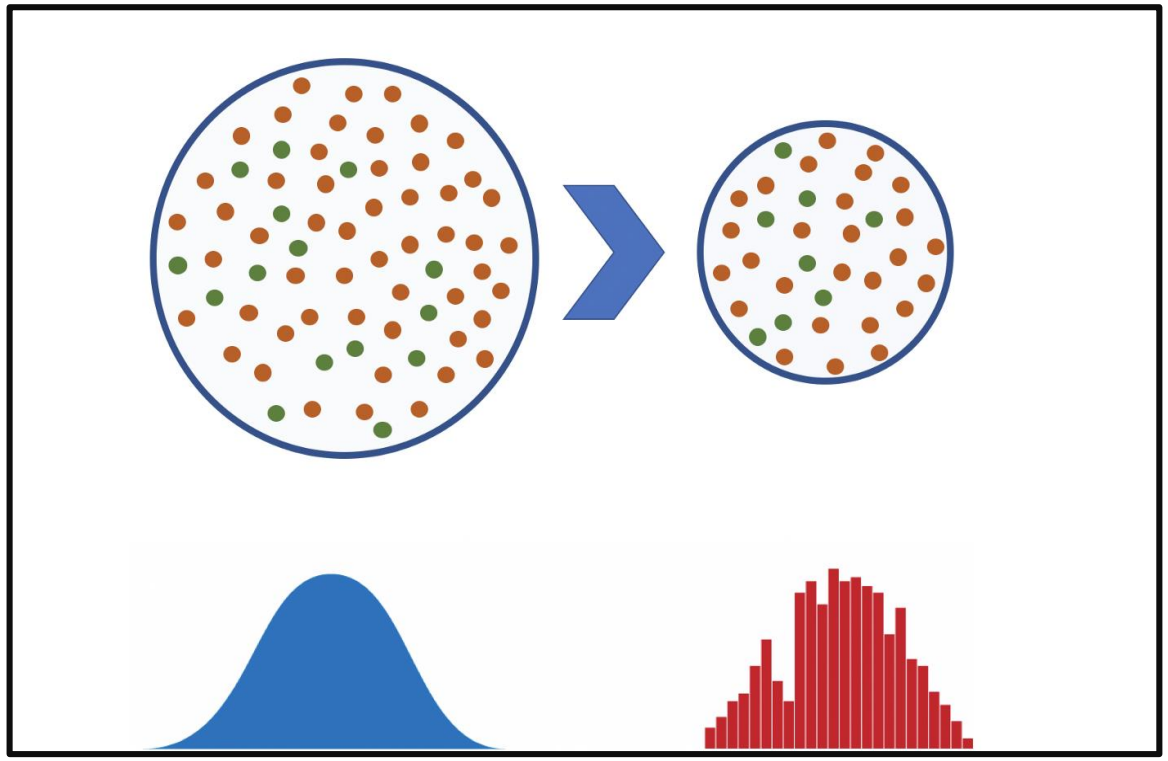
- Formulas:

$$AIC = (2K - 2log(L))/N$$
$$BIC = K * log(N) - 2log(L)$$
$$MDL = L(h) + L(D|h)$$

- where:
  - K: number of independent variables
  - L: maximum likelihood.
  - D: predictions made by the model
  - N: number of samples/data points.
  - L(h): number of bits to represent the model.

MODEL SELECTION: (B) USING RESAMPLING METHODS

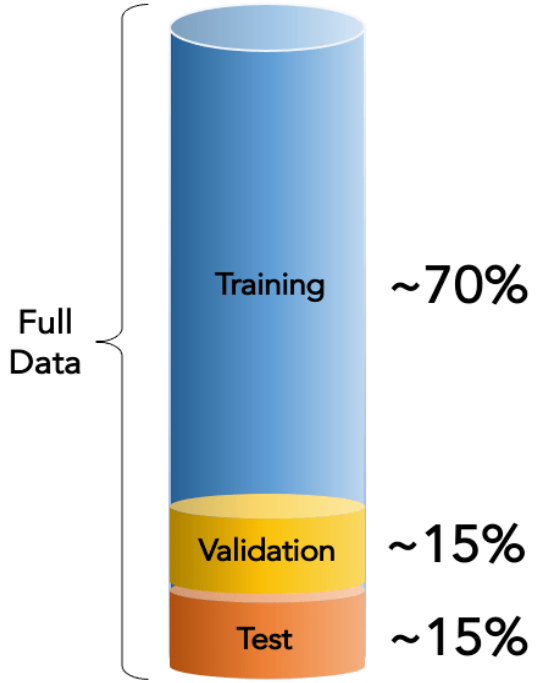
1. Non-probability sampling
    1. Cluster-based split VS Quota split
    2. Convenience sampling
2. Probability based sampling:
    1. Random split
    2. Cluster-based split
    3. Time-based split
    4. K-Fold Cross-Validation
        1. Stratified K-Fold
        2. Bootstrap



2.1 RESAMPLING METHODS : RANDOM SPLIT

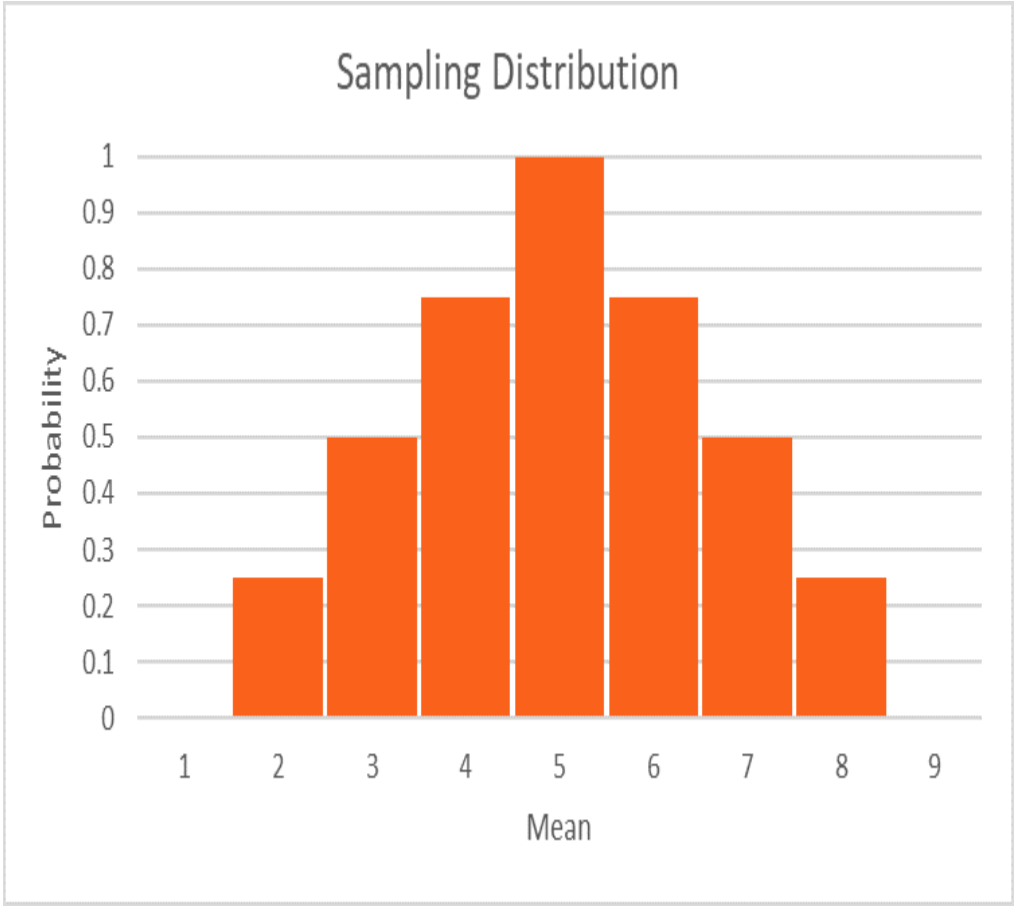
```
import random

fin = open("/path/to/input.txt", 'rb')
f75out = open("/path/to/75-percent-output.txt", 'wb')
f25out = open("/path/to/25-percent-output.txt", 'wb')
for line in fin:
    r = random.random()
    if r < 0.75:
        f75out.write(line)
    else:
        f25out.write(line)
fin.close()
f75out.close()
f
```



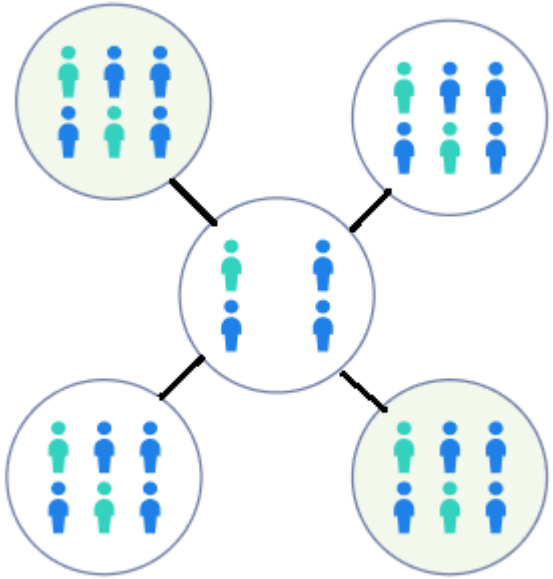
2.1 RESAMPLING METHODS : RANDOM SPLIT

- **Random sampling** consists of randomly selecting a subset (i.e., a sample of data examples.) from a bigger population with equal selection probability for each member of the population.
- **Random sampling** should generate a smaller sample compared to the original population, but conserves same statistical characteristics (i.e., data distribution.).
  - **How can we be sure ?**



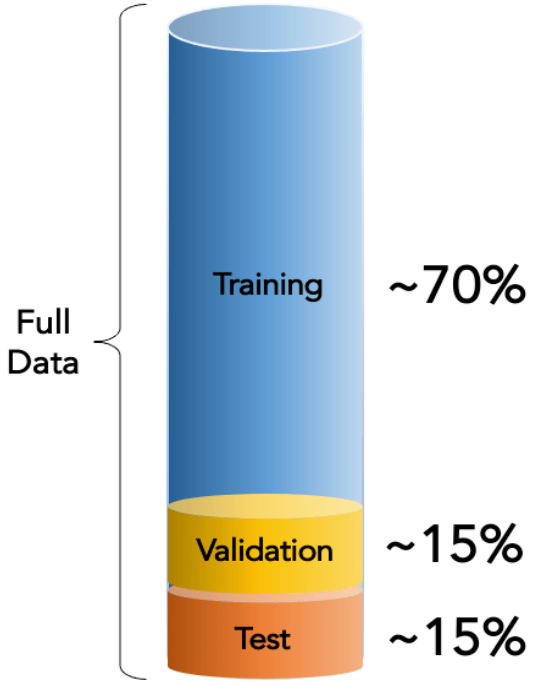
2.2 RESAMPLING METHODS : CLUSTER BASED SPLIT

- **Cluster-based split** applies a clustering before picking members of the final sample.
- It may performed in different ways:
  - Apply clustering, then randomly/ conveniently selecting a portion of members from each cluster.
  - Apply clustering, then selecting randomly a subset of clusters and merge it to generate the final sample.
- Cluster-based split ensure that the final data sample acquires some statistical diversity (i.e., similar frequency distribution).



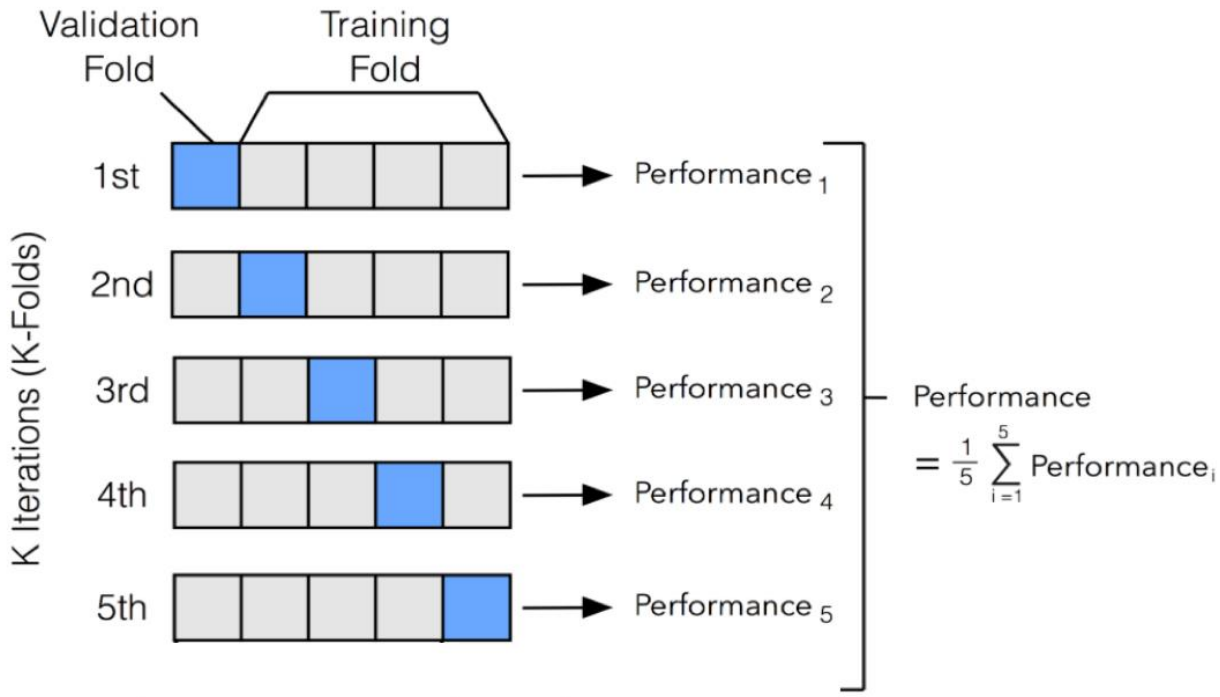
2.3 RESAMPLING METHODS :TIME-BASED SPLIT

- **Random split** can not be applied on time series, thus more dedicated techniques such as **time-based split** can be used. In this technique, a time threshold is fixed, on which data can be spitted. As an example, we have energy consumption of 10 seasons, we can keep 9 first seasons for training, and the 10^{th} season for the test.
- **Window sets** is another alternative which can be used when the dataset is small. In this case, based on time (e.g., dates.) a training set is selected, where the test set is a portion of future data. After that, the training data is shifted to future, and also the testing data.

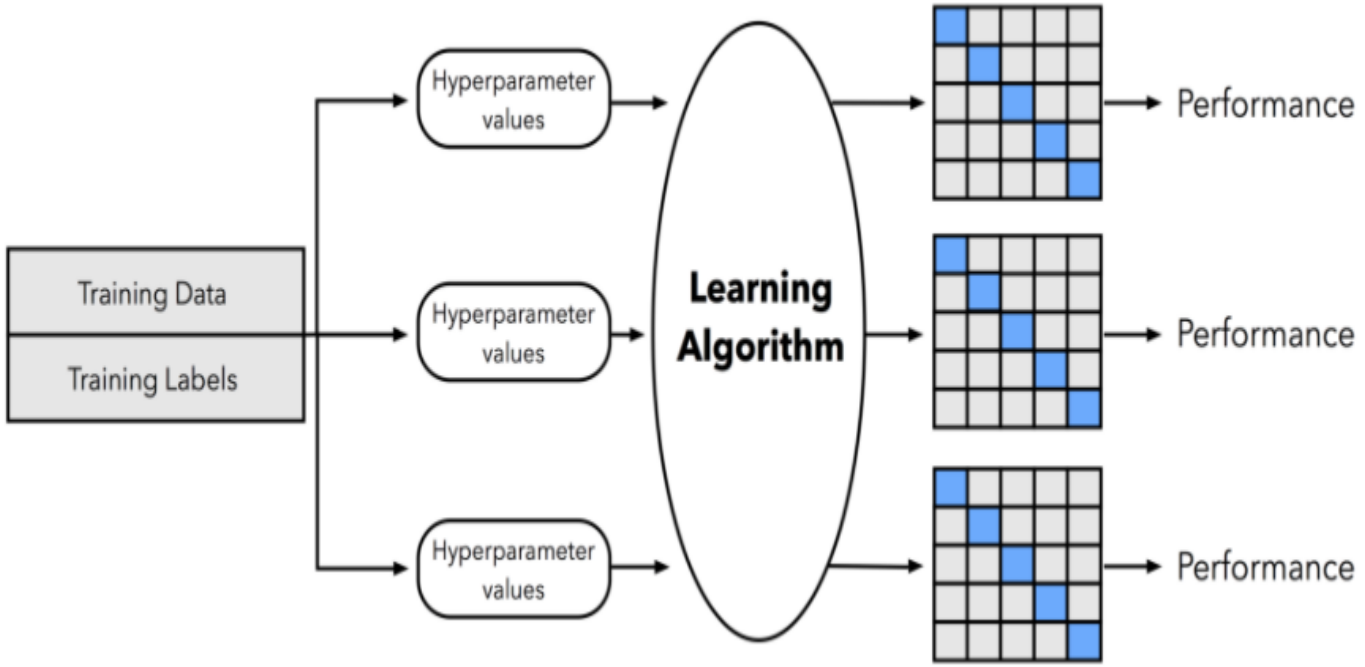


2.4 MODEL SELECTION: USING K-FOLD VALIDATION

- Using K-fold Cross validation to sample data.
  a. Then, parameters tuning.
  b. Then, selecting the best configurations



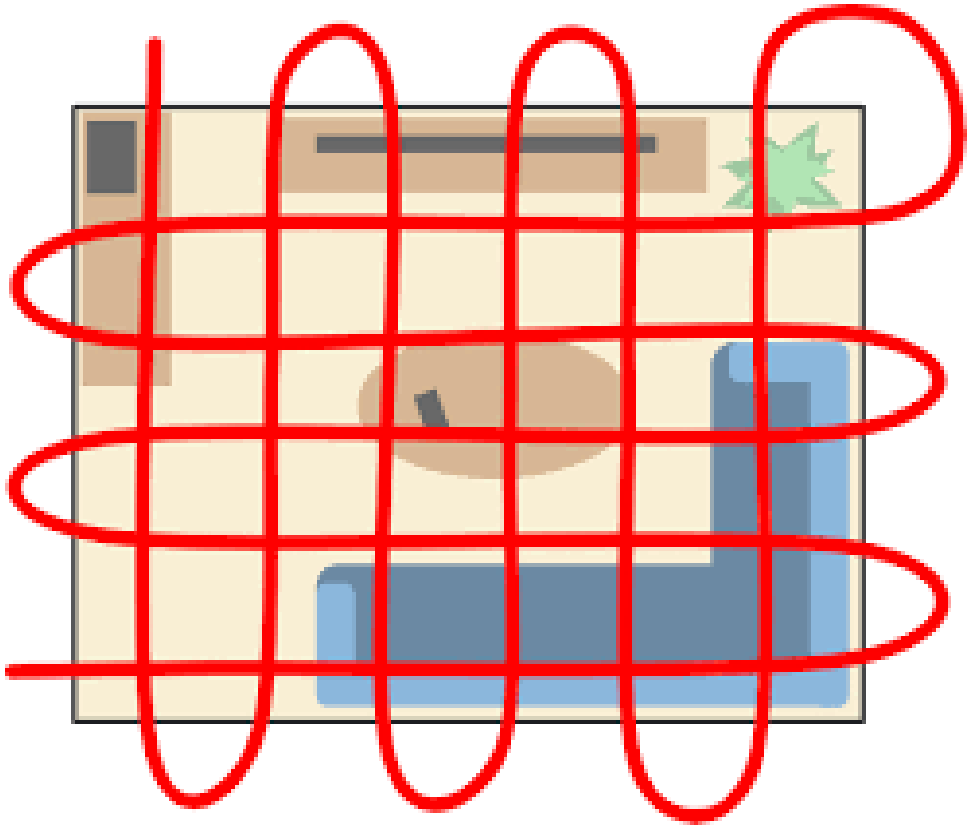
2.4 MODEL SELECTION: USING K-FOLD CROSS VALIDATION



(a) Tuning the parameters

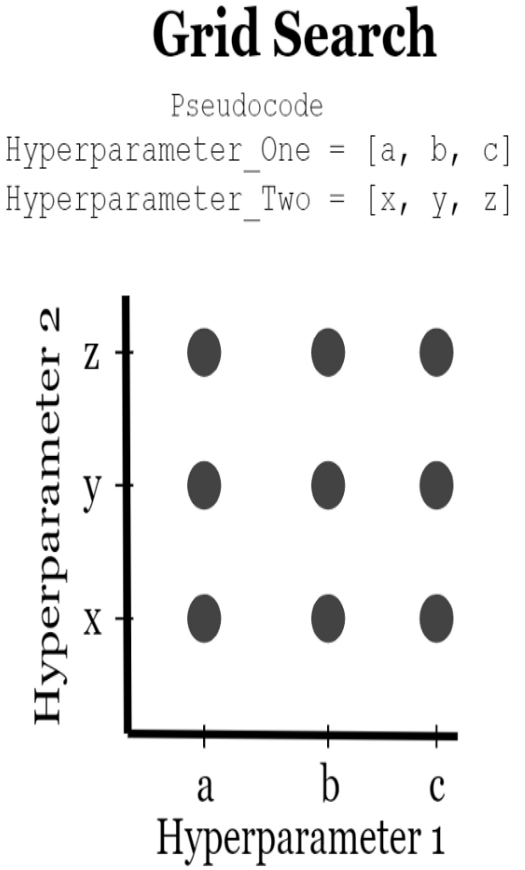
2.4 MODEL SELECTION: USING K-FOLD CROSS VALIDATION

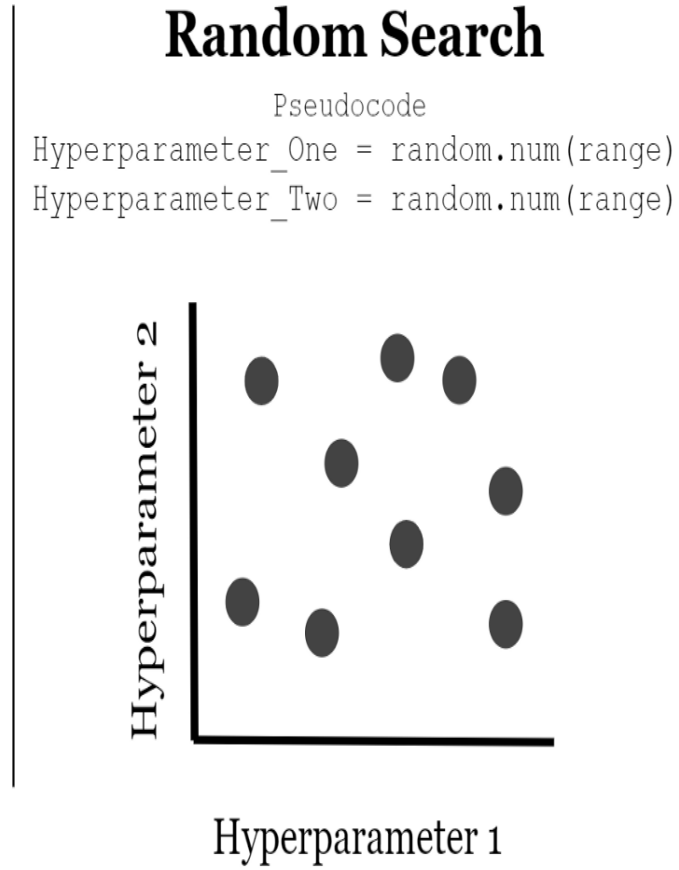
- (b) Tuning the parameters using ***Grid Search***:
  - This method uses an exhaustive algorithm to check all parameters combinations and select the best one.
  - A = (0.1,0.5,1); B= (0.1,0.5,1)
  - Tests = {(0.1,0.1), (0.1,0.5), (0.1,1), (0.5,0.1), (0.5,0.5), (0.5,0.1), (1,0.1), (1,0.5), (1,1),}.
  - Quality criterion should be specified.
  - This is a high complexity algorithm.



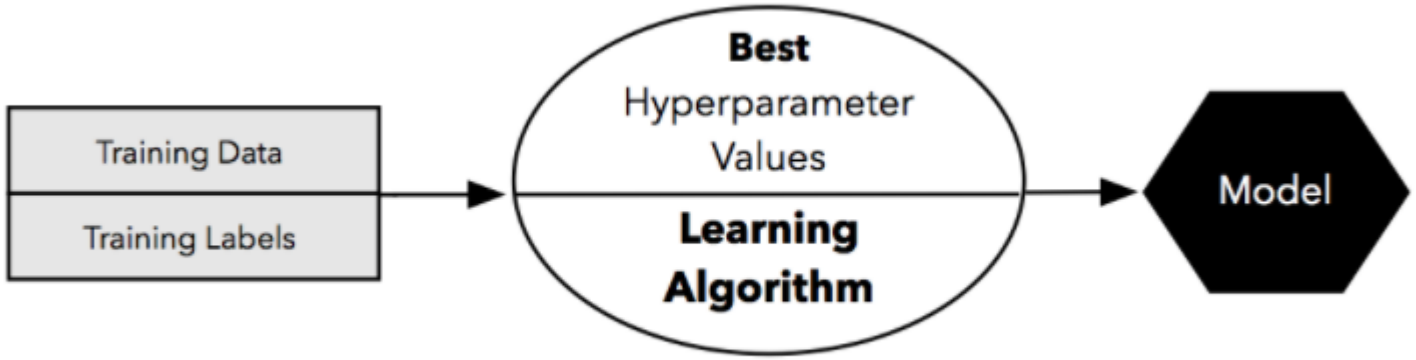
2.4 MODEL SELECTION: USING K-FOLD CROSS VALIDATION

- (b) Tuning the parameters using ***Random Search***
  - **Random search** is a better choice compared to gri search, when computation resources are limited.
  - In random search, we specify for a given hype parameter: (a) a distribution function, (b) samp values, and (c) select the best configuration.





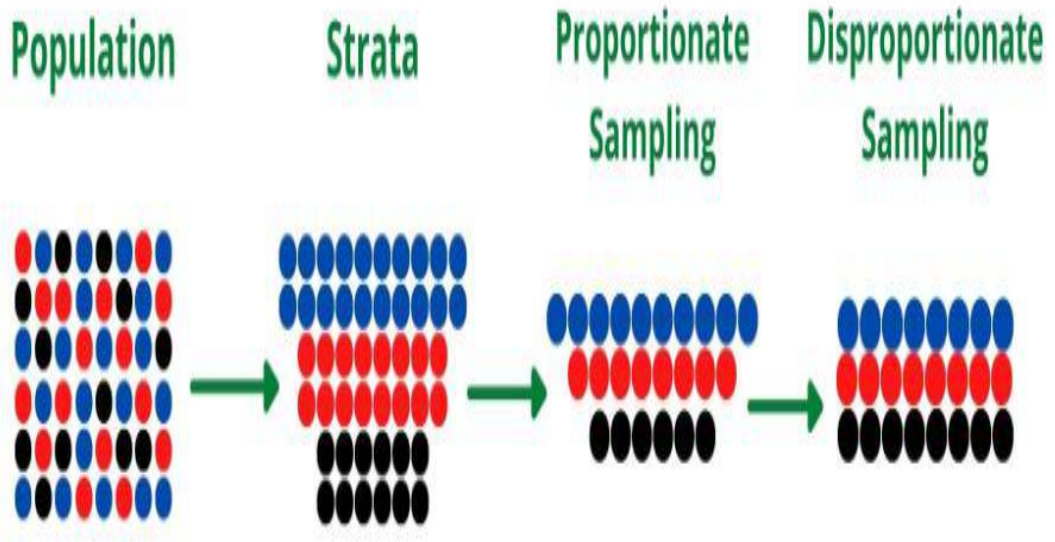
2.4 MODEL SELECTION: USING K-FOLD VALIDATION



(b) Selecting the best configuration model

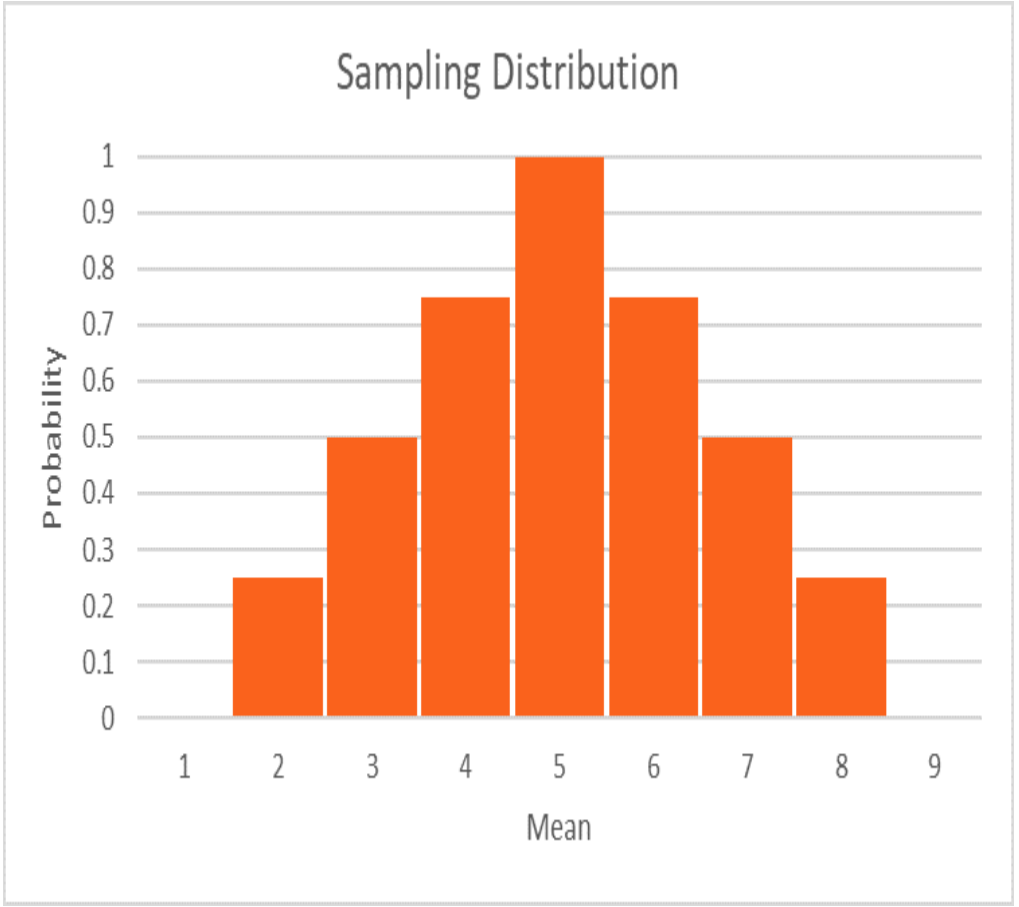
2.4.1 MODEL SELECTION: USING STRATIFIED K-FOLD

- In the stratified K-FOLD, we consider one additional criterion in selection the data, which is the target variable. The target variable is taken in consideration to avoid getting unbalanced data, which improves the accuracy of the model, and reduce bias.
- As an example, if we have a target variable with 2 classes, then stratified k-fold ensures that each test fold gets an equal ratio of the two classes when compared to the training set.

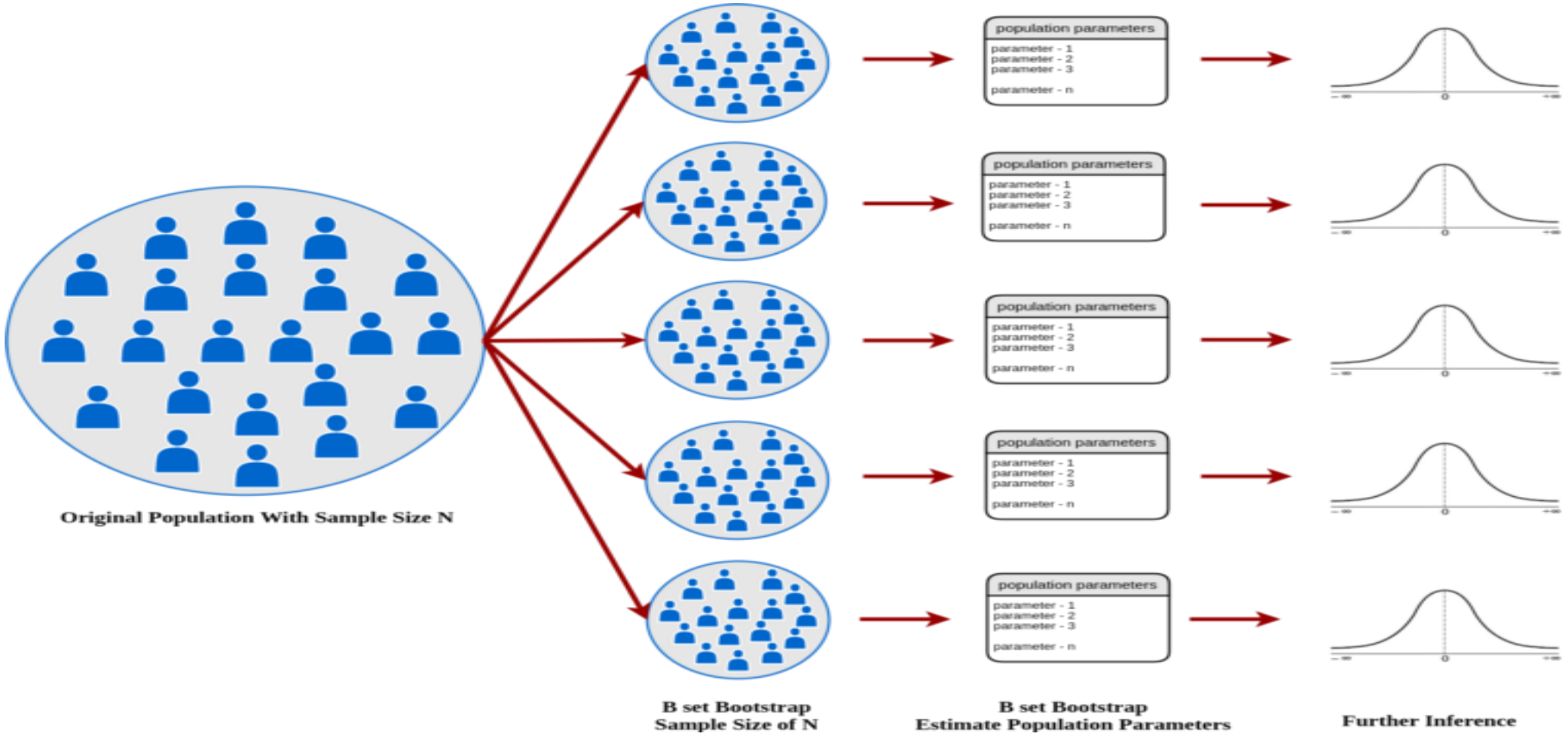


2.4.2 MODEL SELECTION: USING BOOTSTRAPPING

- In stats:
  - In large sample sizes, data sampling will be approximately normal, and the standard deviation will be almost the same as the standard error.
  - But:
    - When, the data sample size is small, we can't be sure that the sampling distribution is normal.
    - Thus, estimating the real standard error is harder.
  - Standard error is the standard deviation of the sample, where the distribution consists some statistical parameter (mean, frequency ..Etc.)



2.4.2 MODEL SELECTION: USING BOOTSTRAPPING



2.4.2 MODEL SELECTION: USING BOOTSTRAPPING

- Two important parameters in the bootstrapping:
  - The size of the population *N*.
  - The number of repetitions (i.e., or the number of samples) *R*.
- Bootstrapping consists of 4 steps:
  - ***Acquire an initial population data*** **:** a population P of **N** member.
  - ***Create R Bootstrap Sample of Size N*:** create *R* random sampling by randomly selecting -with replacement- *N* member from *P*.
  - ***Estimate Population Parameters for each Bootstrap Sample:*** evaluate the resulting of population parameters (mean, standard deviation, precision, …etc.) on the LEVEL of each sample separately.
  - ***Further Inference:*** evaluate the (averaged/combined) results of all samples such as using mean, standard error, confidence interval,… etc.

2.4.2 MODEL SELECTION: USING BOOTSTRAPPING

- **Summary:**

  + suitable for small data sets as it adopts a sampling with replacement.

  + performs well with noisy data (i.e., containing so many outliers.).

  - doesn't perform well when data distribution is not fair enough (i.e., incomplete data observations).

  - May require high calculation resources.

  - Can not be applied on time series data, as the bootstrap method is based on the assumption of data independence.

```
# scikit-learn bootstrap
from sklearn.utils import resample
# data sample
data = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6]
# prepare bootstrap sample
boot = resample(data, replace=True, n_samples=4, random_state=1)
print('Bootstrap Sample: %s' % boot)
# out of bag observations
oob = [x for x in data if x not in boot]
print('OOB Sample: %s' % oob)
```

2.4.2 MODEL SELECTION: EXAMPLE USING BOOTSTRAPPING

```
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.utils import resample
from sklearn.metrics import accuracy_score

# convert outcome to a categorical type
categories=['ALIVE', 'EXPIRED']
cohort['actualhospitalmortality'] = pd.Categorical(cohort['actualhospitalmortality'], categories=categories)

# add the encoded value to a new column
cohort['actualhospitalmortality_enc'] = cohort['actualhospitalmortality'].cat.codes
cohort[['actualhospitalmortality_enc','actualhospitalmortality']].head()

# define features and outcome
features = ['apachescore']
outcome = ['actualhospitalmortality_enc']

# partition data into training and test sets
X = cohort[features]
y = cohort[outcome]
x_train, x_test, y_train, y_test = train_test_split(X, y, train_size = 0.7, random_state = 42)
```

2.4.2 MODEL SELECTION: EXAMPLE USING BOOTSTRAPPING

```
X = cohort[features]
y = cohort[outcome]
x_train, x_test, y_train, y_test = train_test_split(X, y, train_size = 0.7, random_state = 42)

# restructure data for input into model
# note: remove the reshape if fitting to >1 input variable
x_train = x_train.values.reshape(-1, 1)
y_train = y_train.values.ravel()
x_test = x_test.values.reshape(-1, 1)
y_test = y_test.values.ravel()

# train model
reg = LogisticRegression(random_state=0)
reg.fit(x_train, y_train)

# bootstrap predictions
accuracy = []
n_iterations = 1000
for i in range(n_iterations):
    X_bs, y_bs = resample(x_train, y_train, replace=True)
    # make predictions
    y_hat = reg.predict(X_bs)
    # evaluate model
    score = accuracy_score(y_bs, y_hat)
    accuracy.append(score)
```

2.4.2 MODEL SELECTION: EXAMPLE USING BOOTSTRAPPING

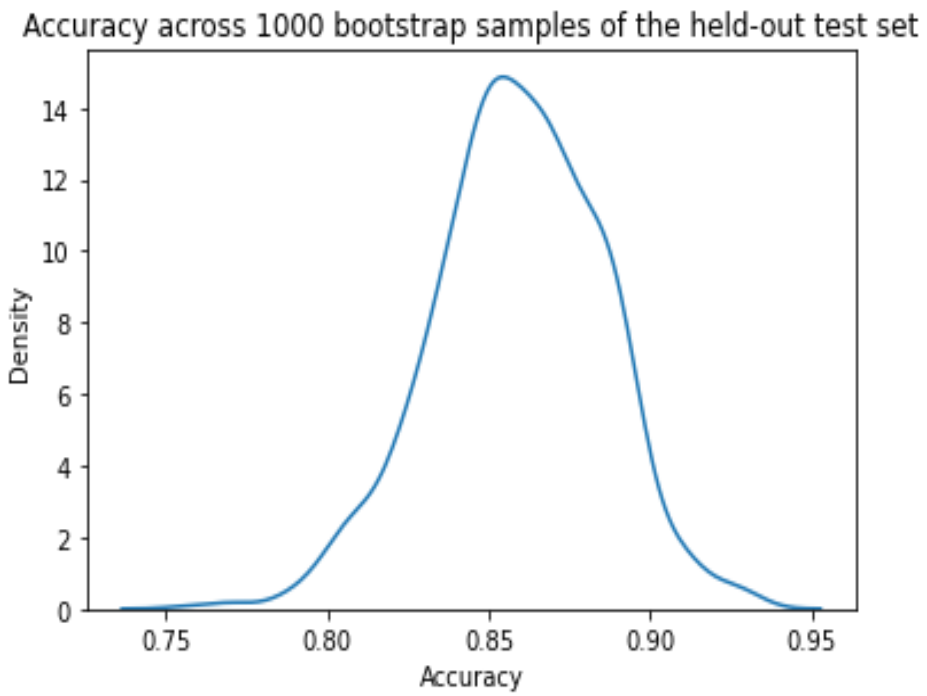
```
import seaborn as sns
# plot distribution of accuracy
sns.kdeplot(accuracy)
plt.title("Accuracy across 1000 bootstrap samples of the held-out test set")
plt.xlabel("Accuracy")
plt.show()
```

- **Using percentile method:**

```
# get median
median = np.percentile(accuracy, 50)

# get 95% interval
alpha = 100-95
lower_ci = np.percentile(accuracy, alpha/2)
upper_ci = np.percentile(accuracy, 100-alpha/2)

print(f"Model accuracy is reported on the test set. 1000 bootstrapped samples "
      f"were used to calculate 95% confidence intervals.\n"
      f"Median accuracy is {median:.2f} with a 95% a confidence "
      f"interval of [{lower_ci:.2f},{upper_ci:.2f}].")
```



2.4.2 MODEL SELECTION: EXAMPLE USING BOOTSTRAPPING

- Bootstrapping is a resampling technique, sometimes confused with cross-validation.
- Bootstrapping allows us to generate a distribution of estimates, rather than a single point estimate.
- Bootstrapping allows us to estimate uncertainty, allowing computation of confidence intervals.

➢ For full online article check this.