NLP for SE/AI Techniques

Agenda

- Midterm exam solution
- Text extractions (cont.)
  - TF-IDF
  - Word Embedding
- Basic Feature selection
- Sampling methods in ML

TF-IDF

- TF-IDF is the product of two
  the inverse document frequ

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

$$TF-IDF(t,d) = TF(t,d) * IDF(t)$$

- t = term
- d = document
- Tf(t,d) = frequency of term t, in document d/Total number of terms in document d

- Idf(t) = log (total number of documents/number of documents with term t in it

TF-IDF (cont.)

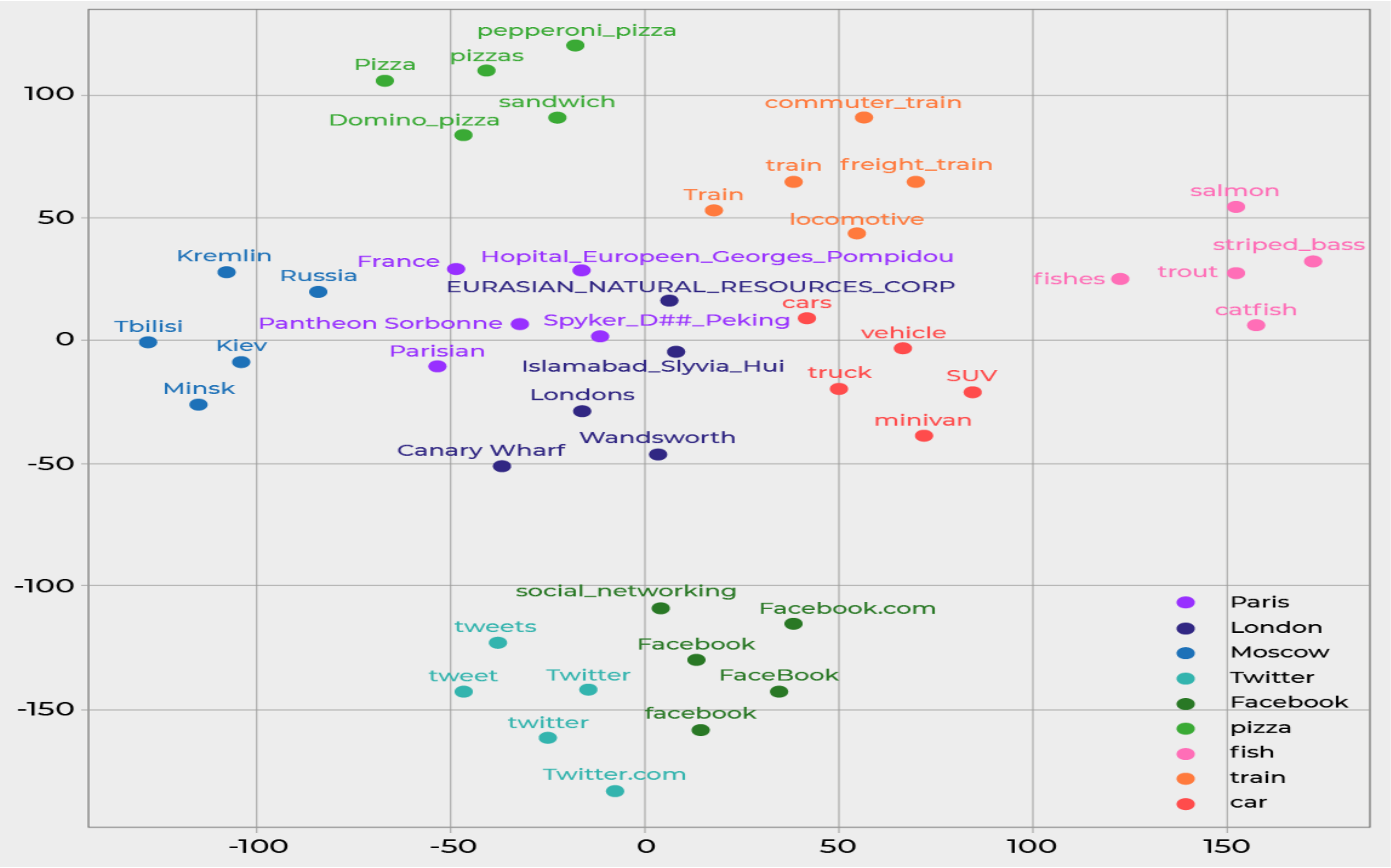
- If a word appears in all the documents, then its inverse document frequency is 1.
- Similarly, if the word appears in few documents, then its inverse document frequency is much higher than 1.
- Alternatively, we can take a log transform of Inverse Document Frequency. Why? Let’s see, Consider we have 10000 documents, and each of these documents has the word the. The IDF score becomes 1. Now, consider a word like market, and it appears in 100 documents, then its IDF score becomes 10000/100 = 100.

```python
import math
from collections import defaultdict

# Sample documents
docs = [
    "the sky is blue",
    "the sun is bright",
    "the sun in the sky is bright",
    "we can see the shining sun, the bright sun"
]

# Calculate term frequency (TF)
def compute_tf(text):
    tf_text = defaultdict(int)
    for word in text.split():
        tf_text[word] += 1
    for word in tf_text:
        tf_text[word] = tf_text[word] / float(len(text.split()))
    return tf_text

# Calculate inverse document frequency (IDF)
def compute_idf(word, corpus):
    return math.log(len(corpus) / sum([1.0 for i in corpus if word in i]))

# Calculating TF-IDF
def compute_tf_idf(corpus):
    documents_list = []
    idf_values = defaultdict(float)

    # Compute IDF for each word
    all_words = set(word for doc in corpus for word in doc.split())
    for word in all_words:
        idf_values[word] = compute_idf(word, corpus)

    # Compute TF-IDF for each document
    for document in corpus:
        tf_idf = {}
        tf_values = compute_tf(document)
        for word, value in tf_values.items():
            tf_idf[word] = value * idf_values[word]
        documents_list.append(tf_idf)
```

```python
.feature_extraction.text import TfidfVectorizer

uments

 is blue",
 is bright",
 in the sky is bright",
 see the shining sun, the bright sun"


fidfVectorizer object
 TfidfVectorizer()

 vectorizer on the documents
 = vectorizer.fit_transform(docs)

e names (words)
s = vectorizer.get_feature_names_out()

e TF-IDF matrix
.toarray()
```

Word Embedding

- Word emb
where wor
vocabulary
of real nun
GloVe, and
- Powerful i
of a word i
semantic
similarity,
words, etc



Word Embedding

	animal	fluffiness	dangerous	spooky
aardvark	0.97	0.03	0.15	0.04
black	0.07	0.01	0.20	0.95
cat	0.98	0.98	0.45	0.35
duvet	0.01	0.84	0.12	0.02
zombie	0.74	0.05	0.98	0.93

Word Embedding with sklearn

- pip install genism
- import gensim.downloader as api

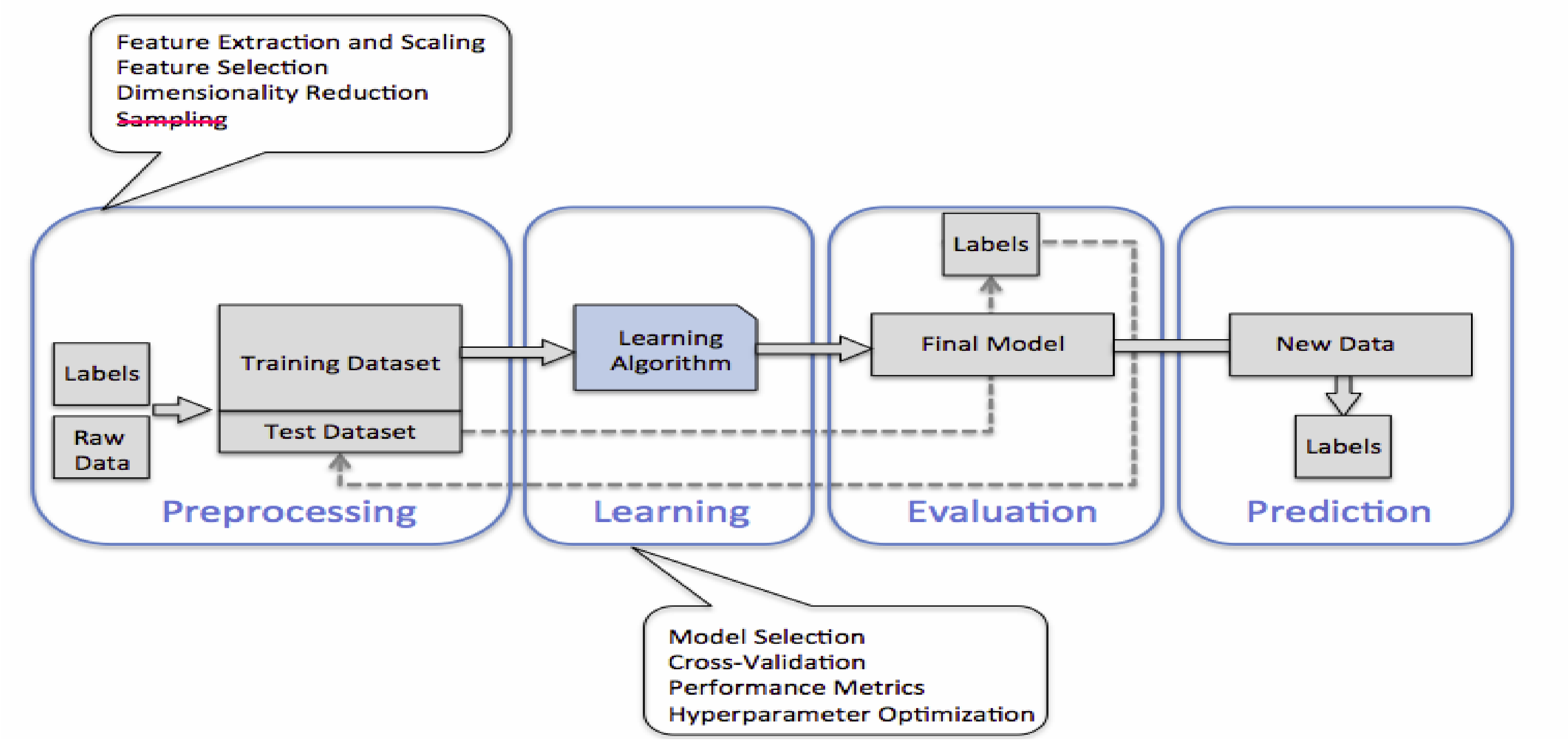
- # Load a pre-trained Word2Vec model (this could take some time and requires internet)
- model = api.load('word2vec-google-news-300')

- # Example: Get the embedding for a word
- word_embedding = model['computer']

- print(f"Embedding for 'computer':\n{word_embedding}")

- # You can also perform operations like finding similar words
- similar_words = model.most_similar('computer', topn=5)
- print("\nSimilar words to 'computer':")
- for word, similarity in similar_words:
-     print(f"{word}: {similarity}")

A roadmap for building machine learning systems



Feature selection using Variance threshold

- High variance = good indication
- Low variance = not so good

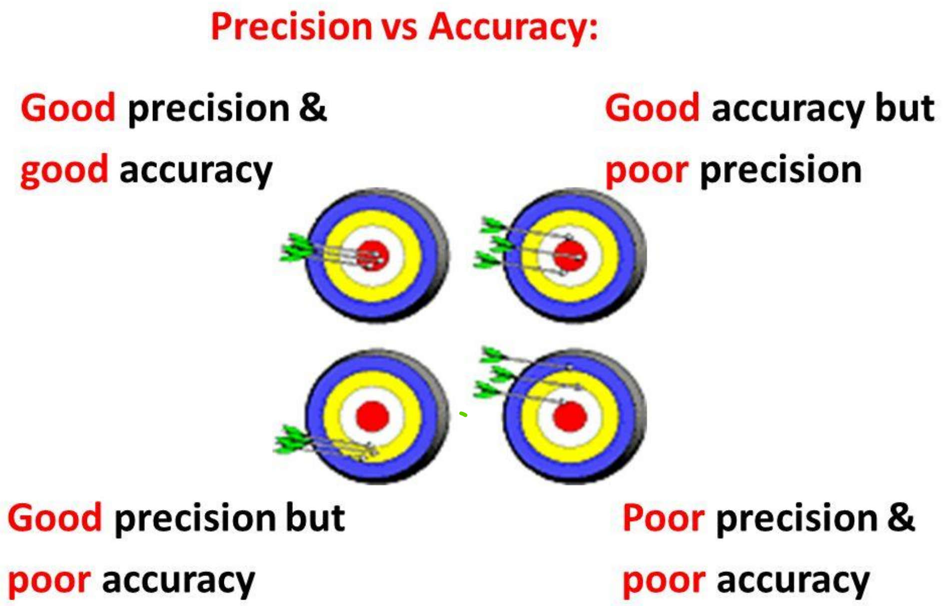
	a	b	c	d
0	1	4	0	1
1	2	5	0	1
2	4	6	0	1
3	3	8	0	1
4	1	11	0	1
5	4	11	0	1
6	4	1	0	1

Feature selection using Variance threshold

```
from sklearn.feature_selection import VarianceThreshold

var_thr = VarianceThreshold(threshold = 0.25) #Removing both constant
and quasi-constant
var_thr.fit(train1)

var_thr.get_support()

array([False,  True,  True,  True,  True,  True,  True,  True,
False])
```

Performance Evaluation of Classifiers

- **Simplest measure : rate of correct predictions**
- **Confusion matrix**
- **Precision- How many selected items are relevant??**
- **Recall – How many relevant items are selected?**
- F-measure (consider both precision and recall)
- ROC Area



Model construction

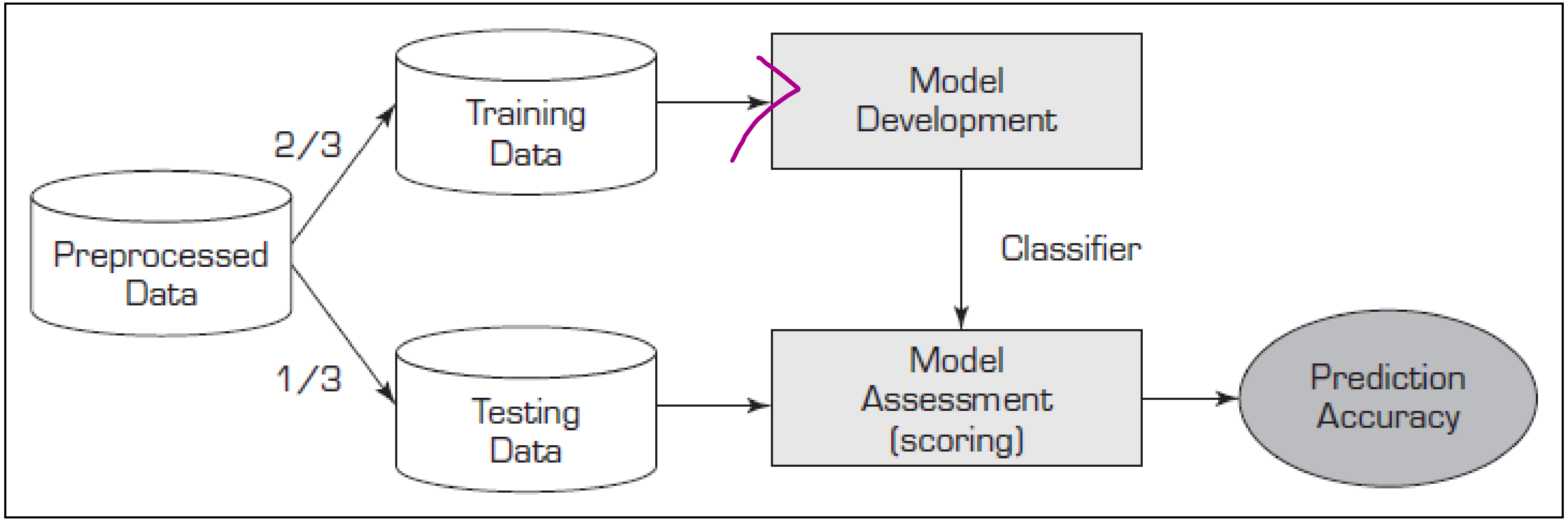
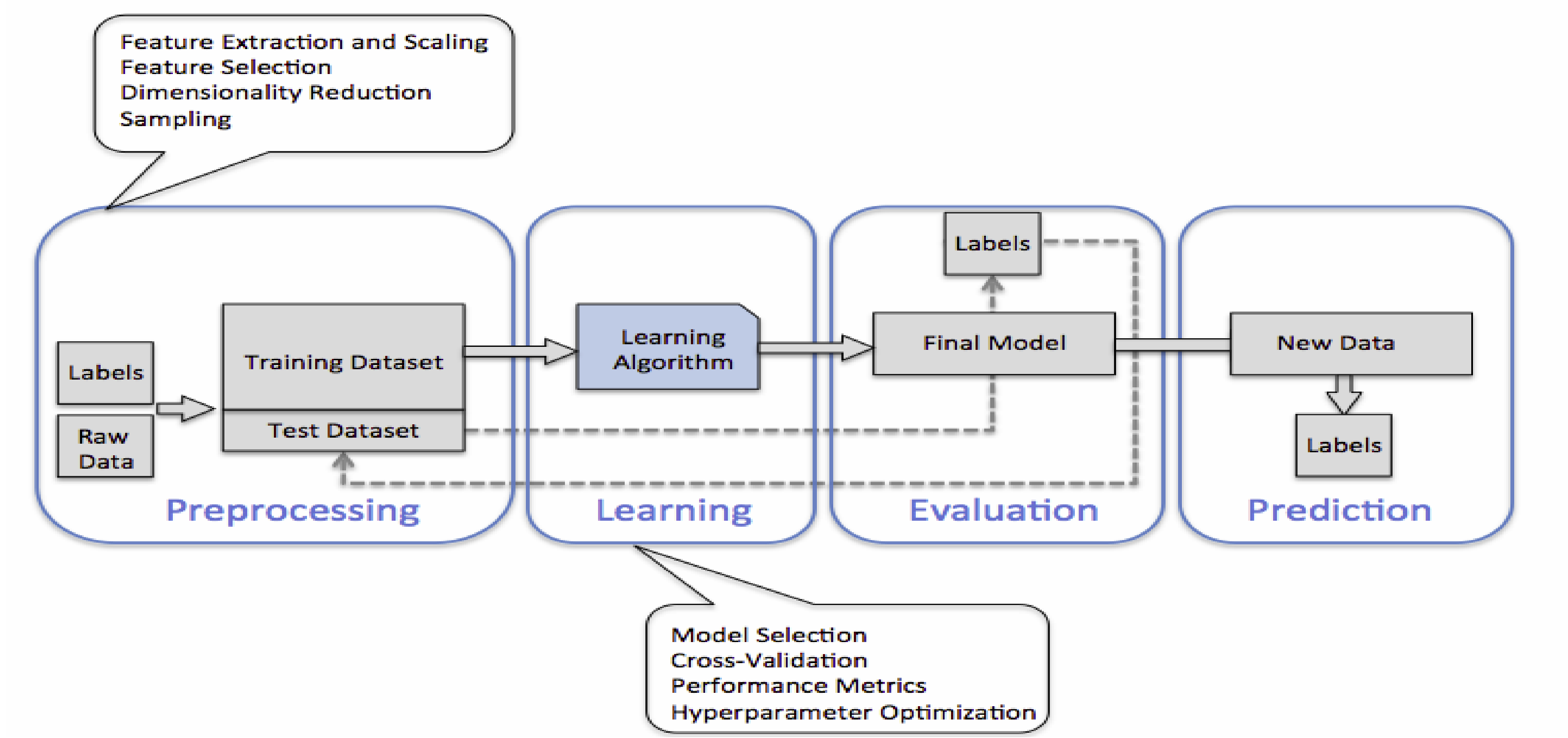


FIGURE 5.9 Simple Random Data Splitting.

A roadmap for building machine learning systems



Population

- A population is the collection of items of interest
- Usually defined as ‘*N*’

Sample

- A valid alternative to a census
- Budget constraints
- Time constraints
- Urgent need of data
- Subset of the population
- Usually denoted with ‘*n*’

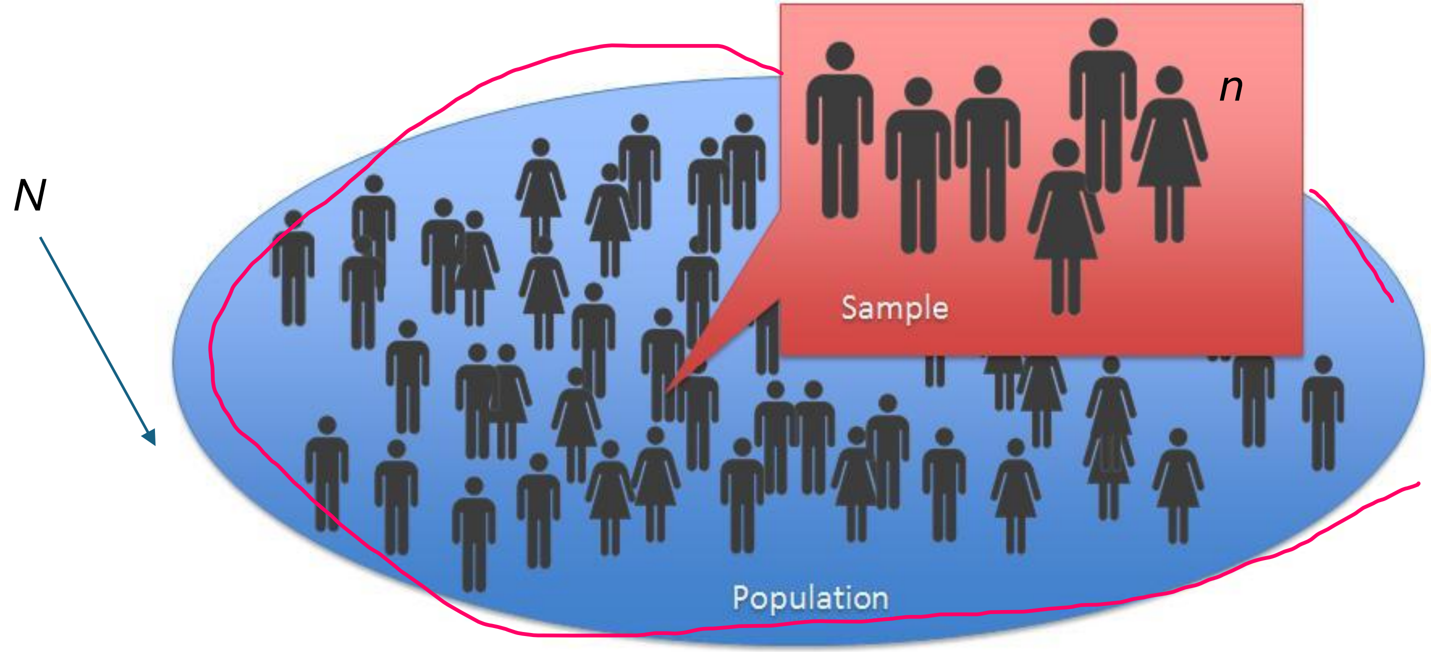


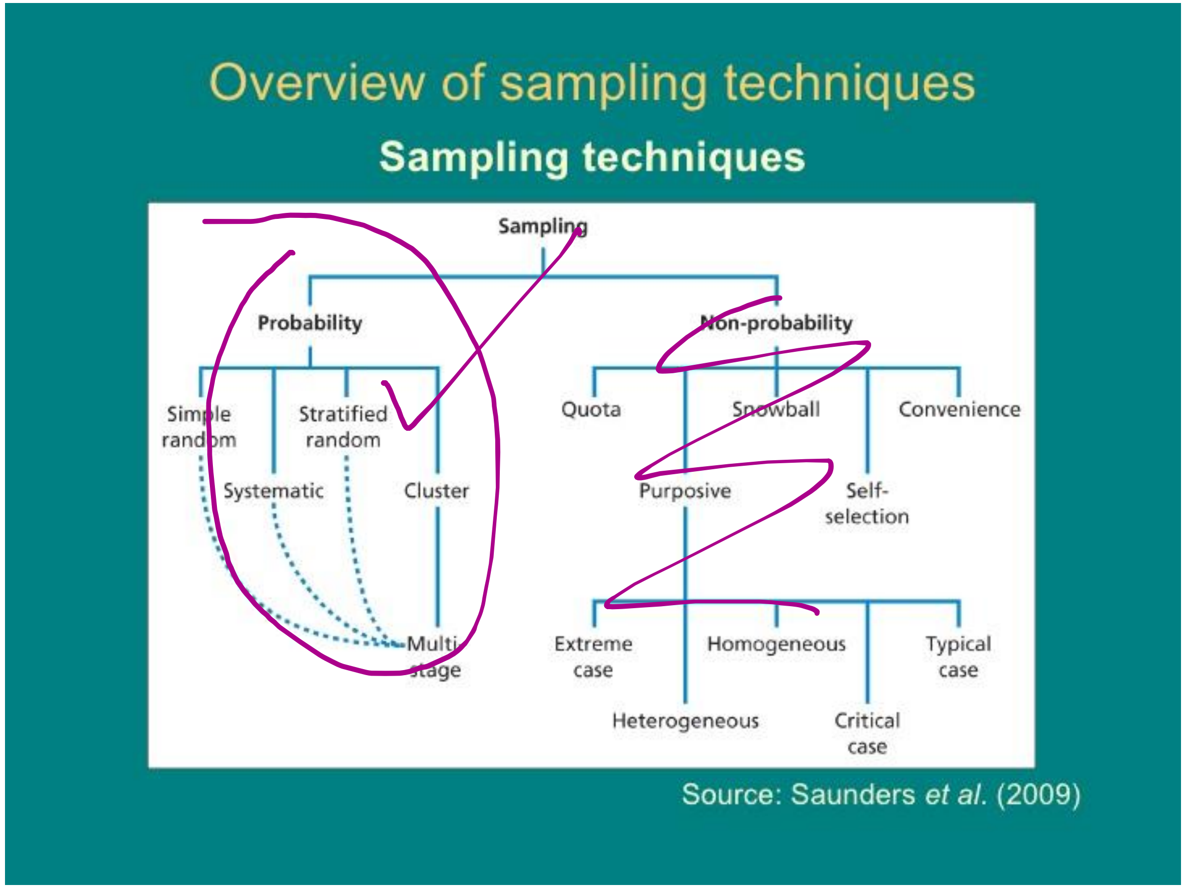
Image sources: http://www.niqcgroup.com/what-is-sampling-and-its-objective/ (Accessed, September 2018)

Why Sample?

- Give you better results
- Less computation time
- Less cost in data collection
- Its impossible to study the whole population

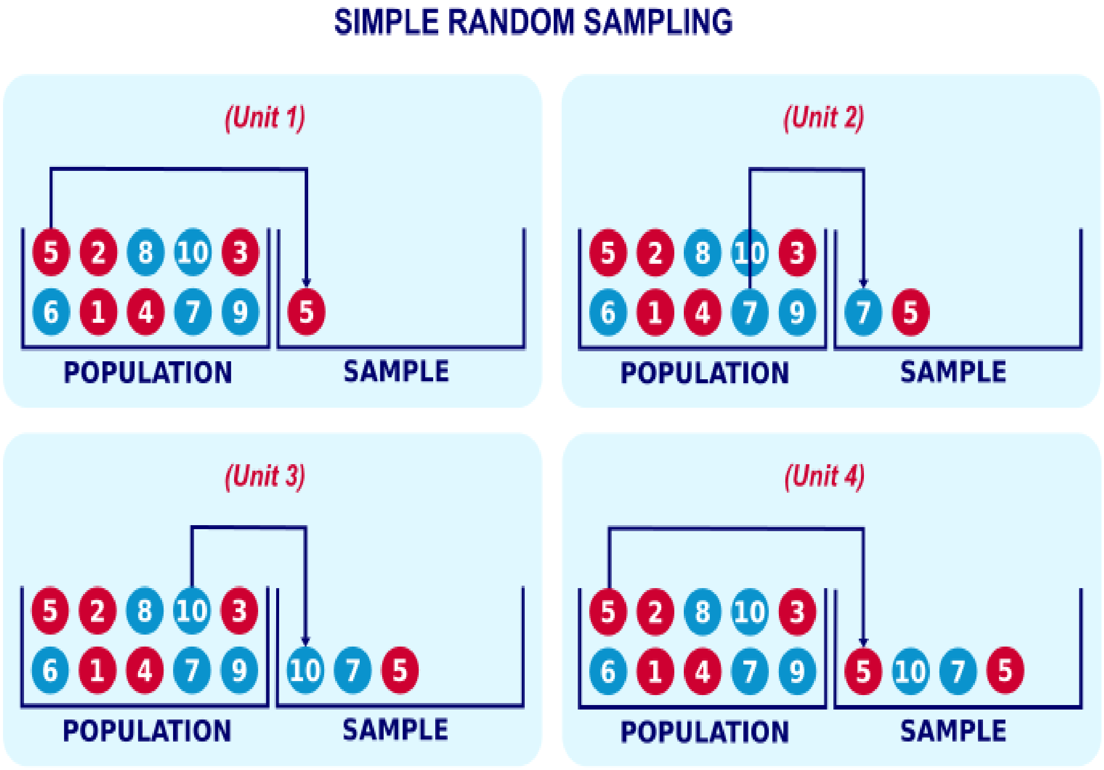
Types of Samples

- **Probability samples**
  - Simple random sample with replacement
  - Simple random sample w/o replacement
  - Stratified random sample
  - Cluster sample
- Equal chance to be select
- Truly represent the pop.
- **Non-probability samples**
  - Quota
  - Etc.
- Do not have equal change to of being selected
- Poor ~~generalizable~~



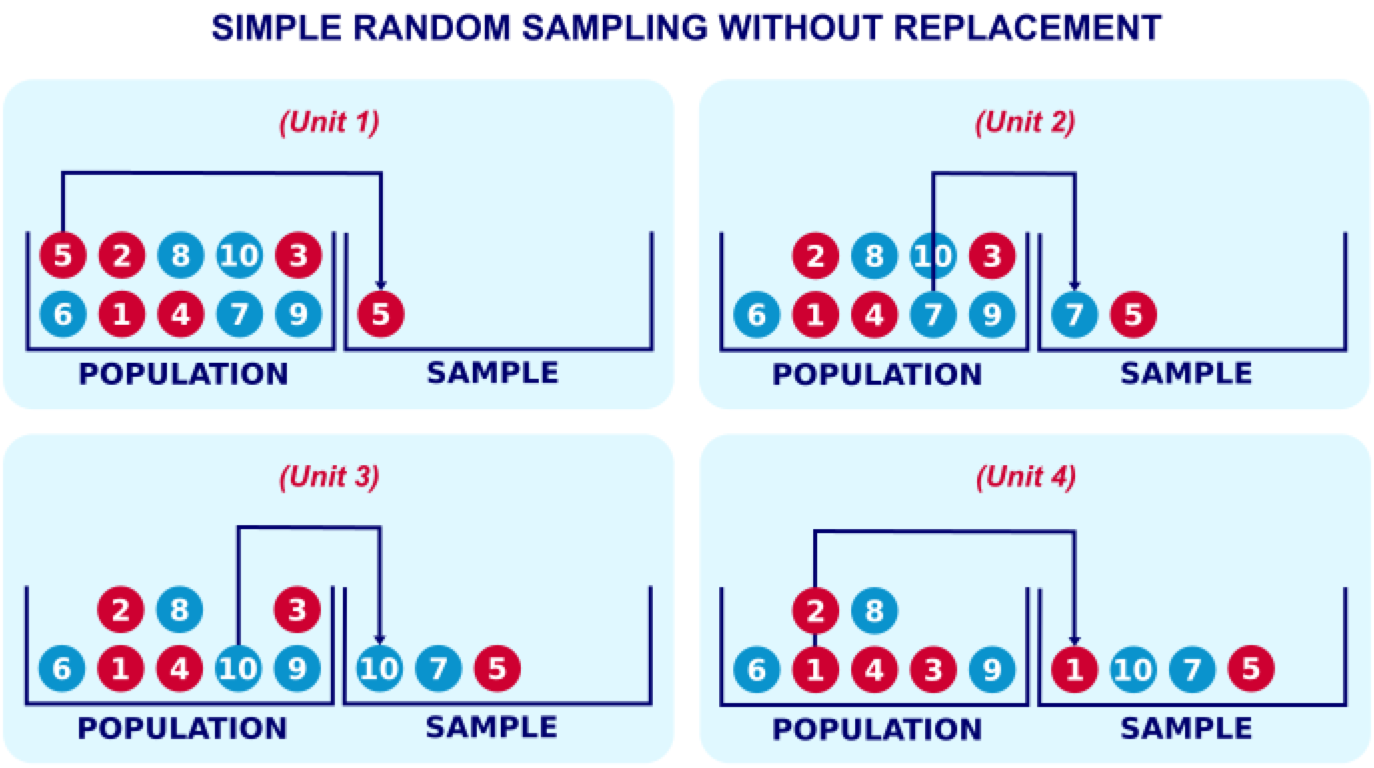
Simple Random Sampling (SRS)

- Mostly use sampling method
- Each ball has the same
- Chance of 0.1 of being sampled
- Ensue that every ball will have equal change of being included in the sample
- Duplicated samples appear
- Sample from an infinite population.
- AKA. (SRSWR)



Simple Random Sampling WithOut Replacement (SRSWOR)

- Each ball has only one chance to be sampled
- Sample from a finite population



Simple Random Sampling

- Easy to implement
- SRS has problem with high generalizability of findings.
- High cost of collecting data
- Stratified Sampling (STS)
  - Most efficient and precise
  - Very useful when dealing with imbalanced data set

Stratified Sampling

1. Population is split into groups called strata
2. Each strata has equal proportion of population
3. Sample is selected from each strata using SRS

Stratum	A	B	C
Population size	100	200	300
Sample fraction	50%	50%	50%
Final sample size	50	100	150

Cluster Sampling

- In cluster sampling a cluster represents as a sampling unit
- In stratified sampling only specific elements of strata are accepted as sampling unit

Cluster sampling

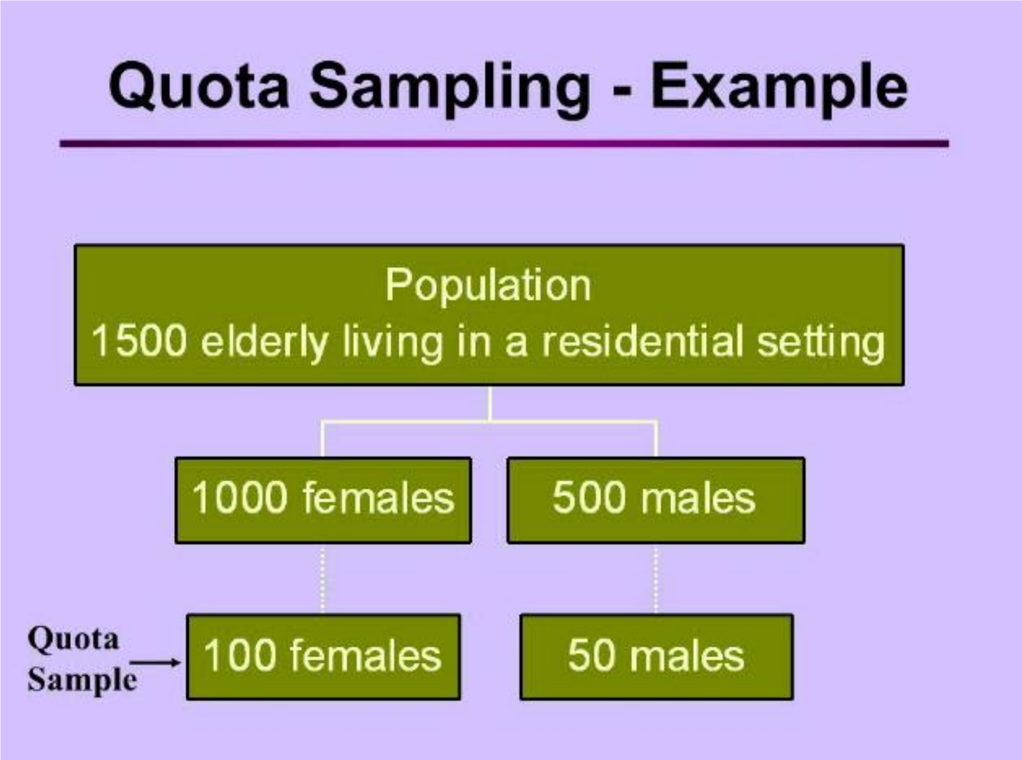
- Advantages
  - Very practical
  - Best **time and cost efficient** for large geographical areas
- Disadvantages
  - Require group information to be known (expertise in the domain)
  - Higher sampling error than other approaches

Example of Cluster sampling

- Assume that you would like to evaluate consumer spending behavior on various modes of transportation in Chiang Mai
- Chiang Mai has 25 districts (amphoe)
  - Select a cluster grouping as a sampling frame
    - 25 amphoe are not the sampling frame for the study
  - Mark each cluster with unique number
  - Use Probability sample (5 from 25 amphoe)

Quota Sampling

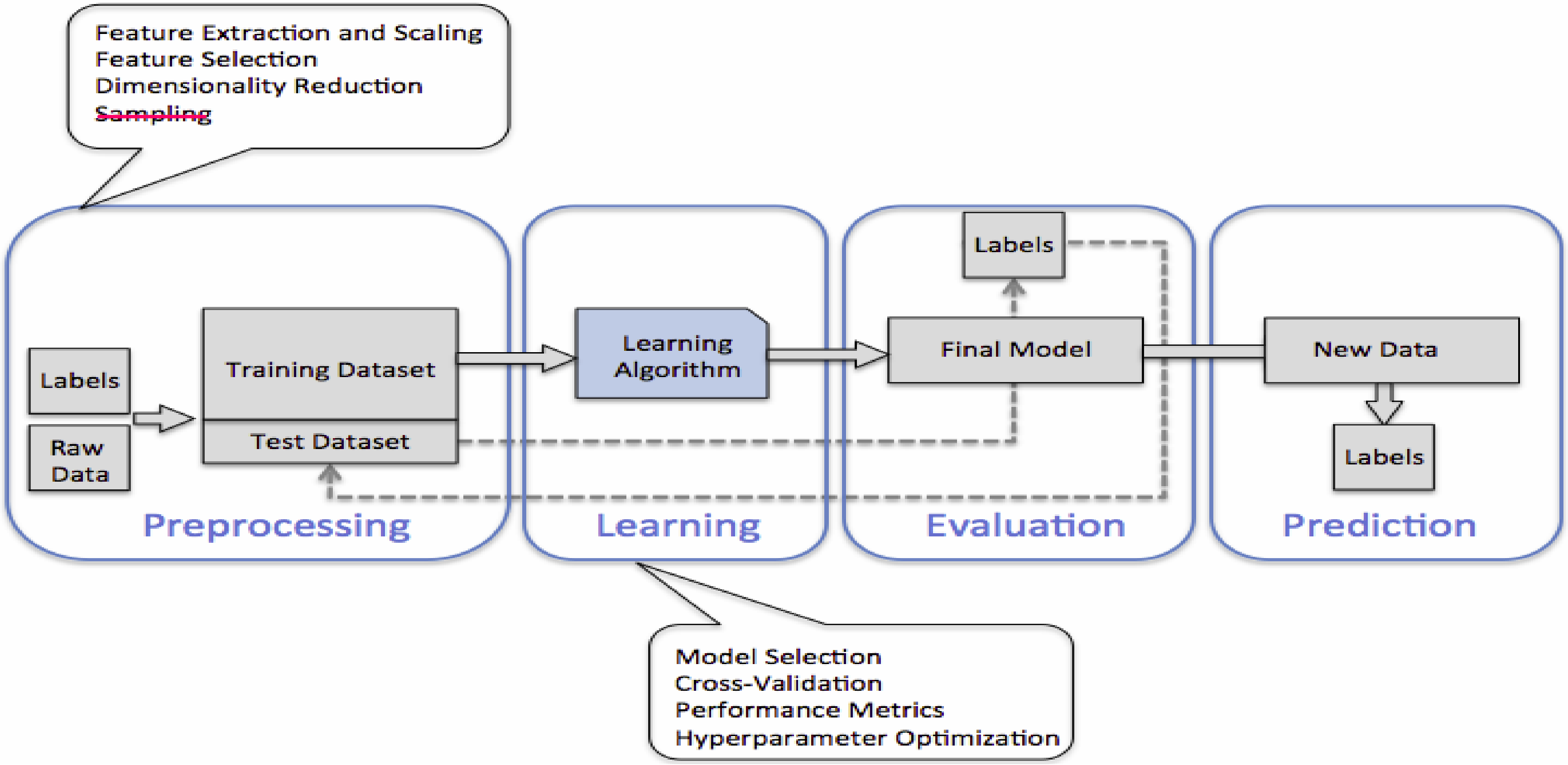
- The population is split as same as in stratified sampling
- Instead of randomly selection, quota sampling use non-probability sampling.
  - Interview tempted to interview those who look most helpful
- Lead to bias results?



Imbalanced data

- 2 types of learning approach of machine learning
  - Supervised Learning
    - There is a solution given for the machine learning algorithm
  - Unsupervised Learning
    - There is no solution given to the machine learning algorithm
- Both of them learn from the sampling data of different classes
- Imbalance problem happens when the size between different classes is radically different.

A roadmap for building machine learning systems

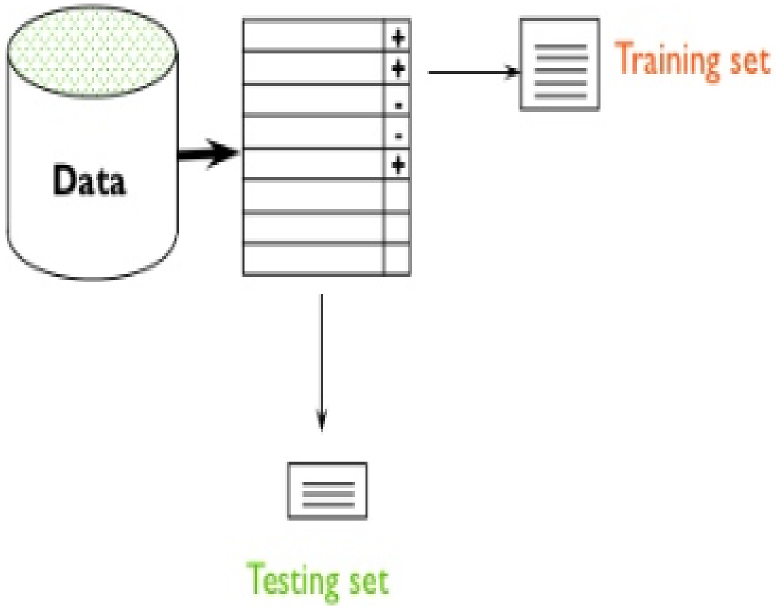


Training and Testing Dilemma

- What we usually expect
  - A large training data set
  - A large testing data set
- More often, we don’t have enough quality and quantity of data when doing analysis.

Hold-out method

- Good approach for a large data set, if we have more than 1,000 samples, including several hundred instances from each class.
- Split data into training data and testing data
- 80% for train, 20% for test or
- Build classifier using the train data
- And test with the test data



Sklearn holdout

- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Repeated Hold-out Method

- Use n iterations
  - More reliable by repeating the process with different subsamples
- In each iteration, certain proportion of dataset is randomly selected for training (possibly with stratification)
- The accuracy rate on different iteration then with be average

Stratified Hold-out method

- Similar to the simple hold-out
- However, we check that each class is represented in approximately equal proportion in the test dataset as it was in the overall dataset.

Possible issue with all the hold-out methods

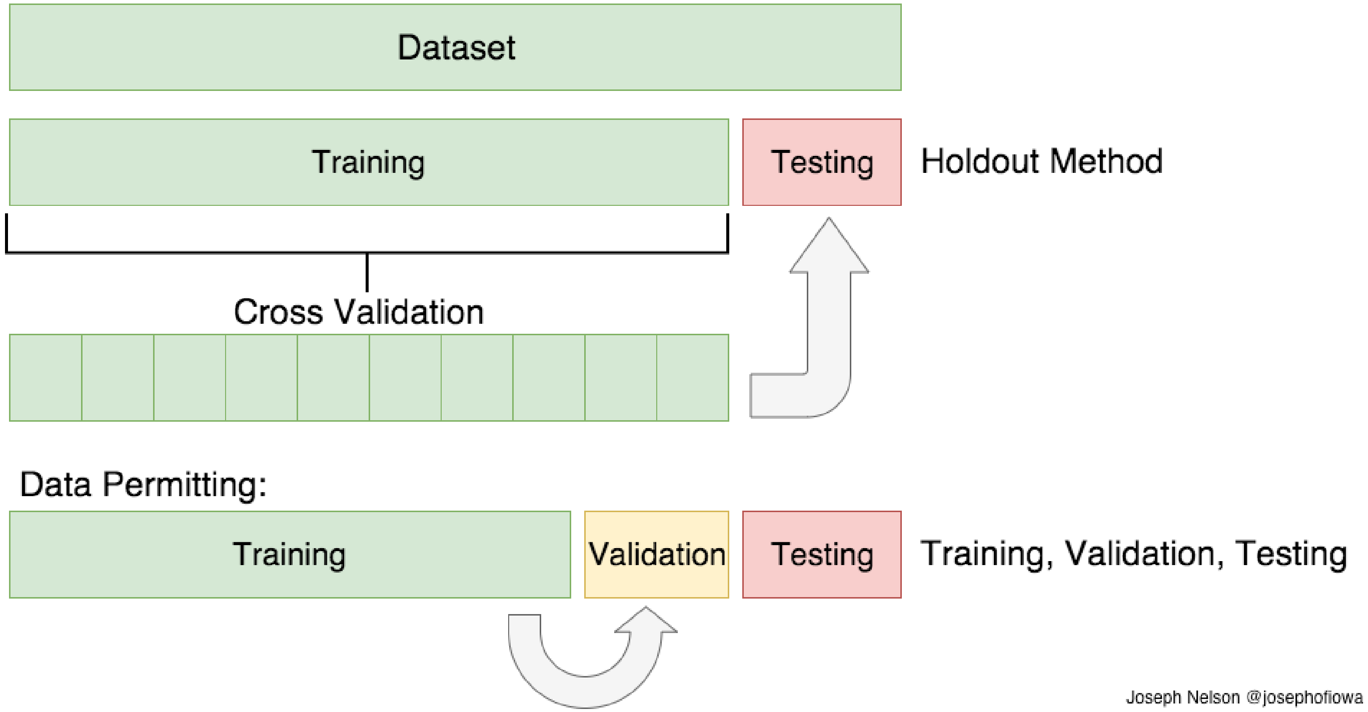
- Hold-out methods is still not optimal.
- Due to the proportion to be held out for testing is randomly selected
- So, the testing set may overlap.

ADV. Evaluation Techniques (when we don't have enough or quality dataset)

- Cross Validation
- Stratified Cross Validation
- Leave-One-Out Cross validation
- Bootstrapping

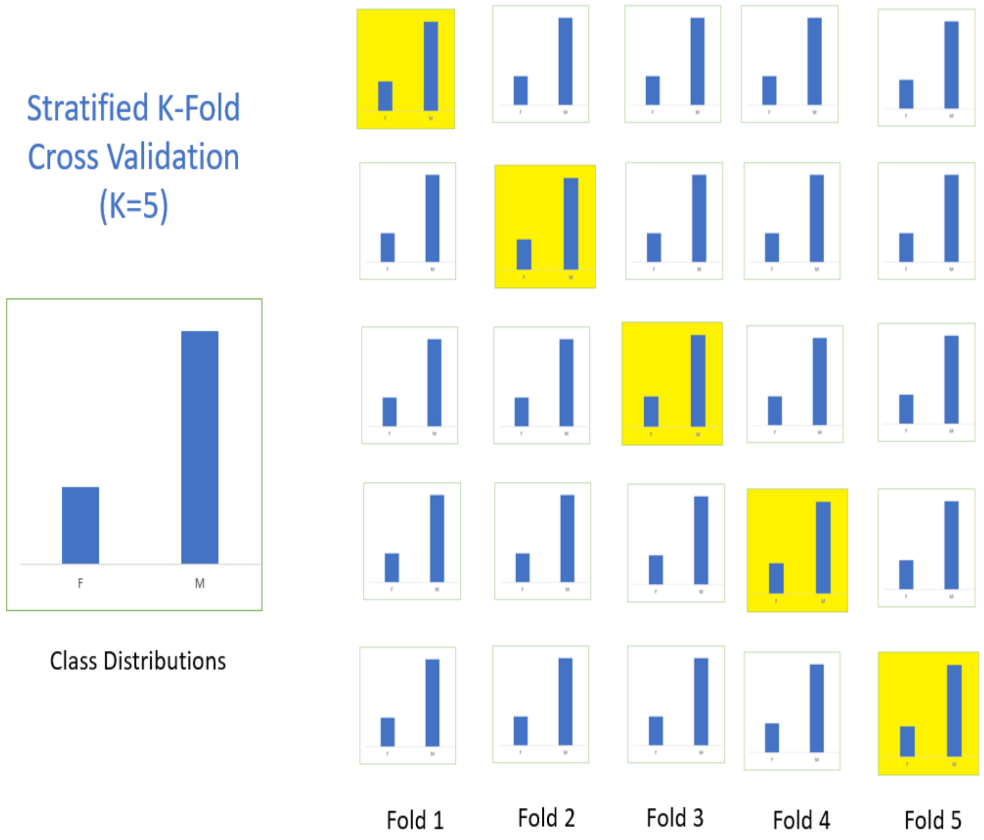
K-folds Cross-validation Method

- AKA. Rotation estimation
- Use to estimate a performance of the mode (i.e. mean of accuracy rate)



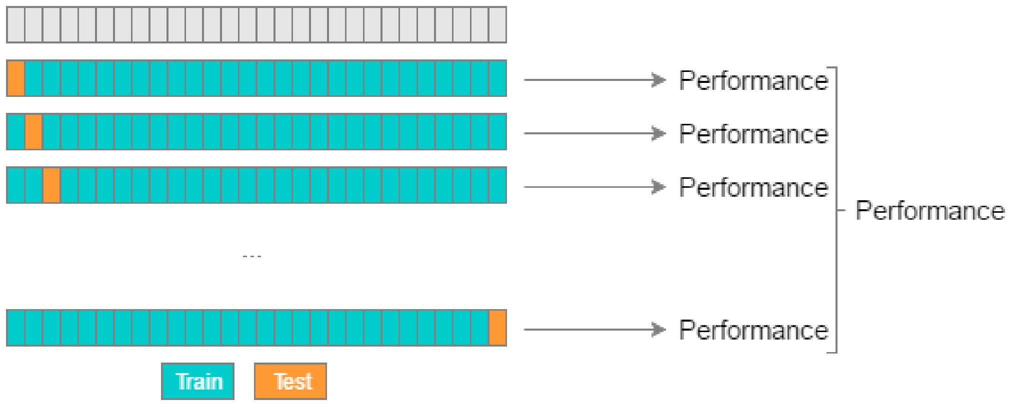
Stratified Cross-validation Method

- Same as Cross-validation but here we ensure that each fold is representative of all strata of the class.



Leave-one-out Cross-validation Method

- Cross-validation for small sample size.
- The number of folds is the same as the number of training instances.
- Advantages:
  - Makes the best use of the data
  - Involve no random sampling
- Disadvantages:
  - Took long time to run, computationally expensive



Programming 1 (Due next week)

- 1. Use the spam dataset that we have been working on
- 2. Apply TF-IDF technique we have just learned
- 3. Apply feature selection with variance threshold (use threshold level = 0.1 (try))
- 4. Report how many feature you have removed.
- 5. Apply stratified hold-out with 70:30 ratio, with no shuffle, random state = 1234
- 6. Report the shape of matrix for train and test set.
- 7. Report the top 10 and buttom 10 rows.
- Submit your work in MS team (no zip) just submit the iynpb file.