Cross Validation Techniques

Tushar B. Kute,
http://tusharkute.com

Cross Validation

- Cross-validation is a statistical method used to estimate the performance (or accuracy) of predictive statistical models.
- It is used to protect against overfitting in a predictive model, particularly in a case where the amount of data may be limited.
- In cross-validation, you make a fixed number of folds (or partitions) of the data, run the analysis on each fold, and then average the overall error estimate.

Cross Validation

- When dealing with a predictive modeling task, you have to properly identify the problem so that you can pick the most suitable algorithm which can give you the best score. But how do we compare the models?
- Say, you have trained the model with the dataset available and now you want to know how well the model can perform.
- One approach can be that you are going to test the model on the dataset you have trained it on, but this may not be a good practice.

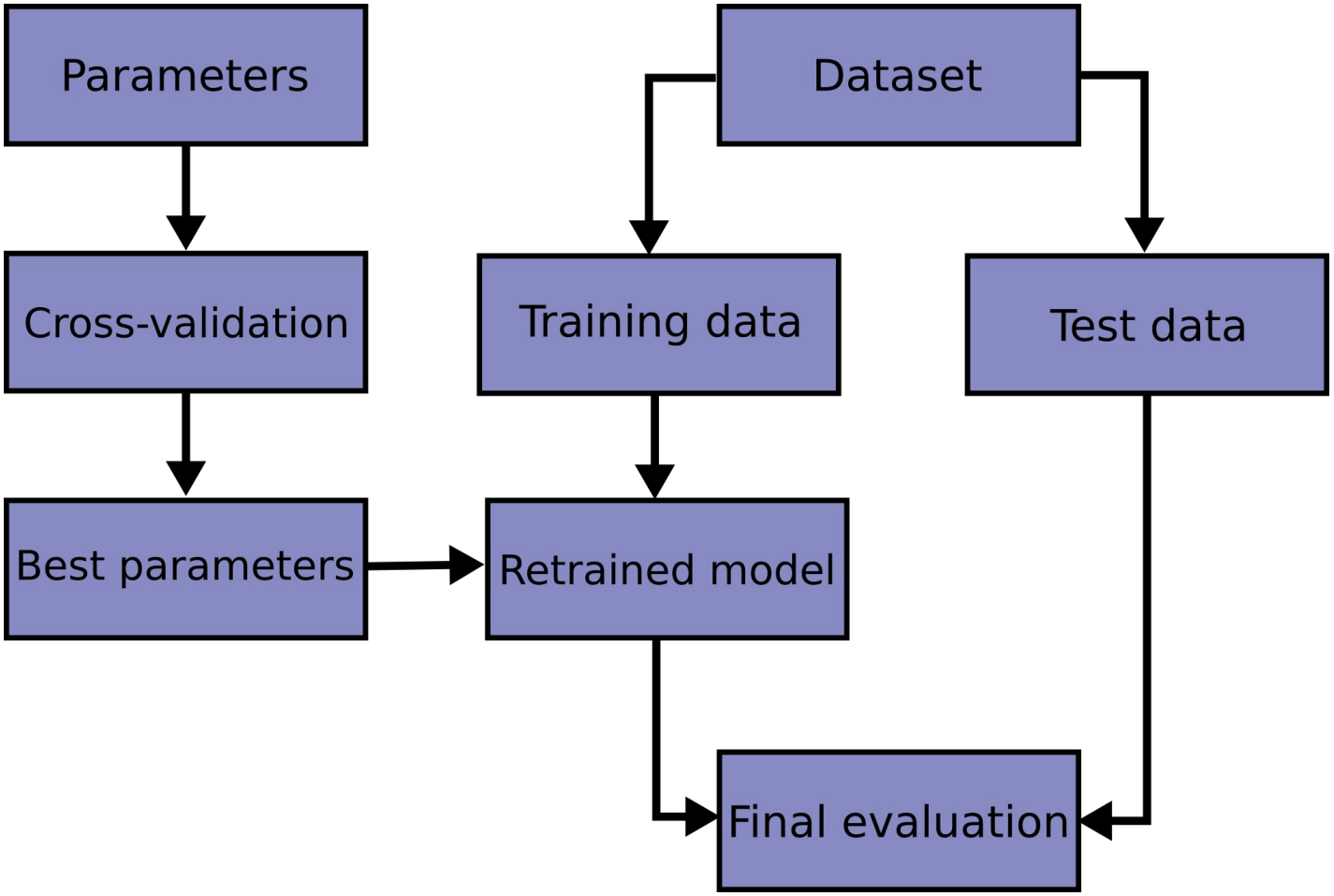
Cross Validation

- So what is wrong with testing the model on the training dataset?
- If we do so, we assume that the training data represents all the possible scenarios of real-world and this will surely never be the case.
- Our main objective is that the model should be able to work well on the real-world data, although the training dataset is also real-world data, it represents a small set of all the possible data points(examples) out there.

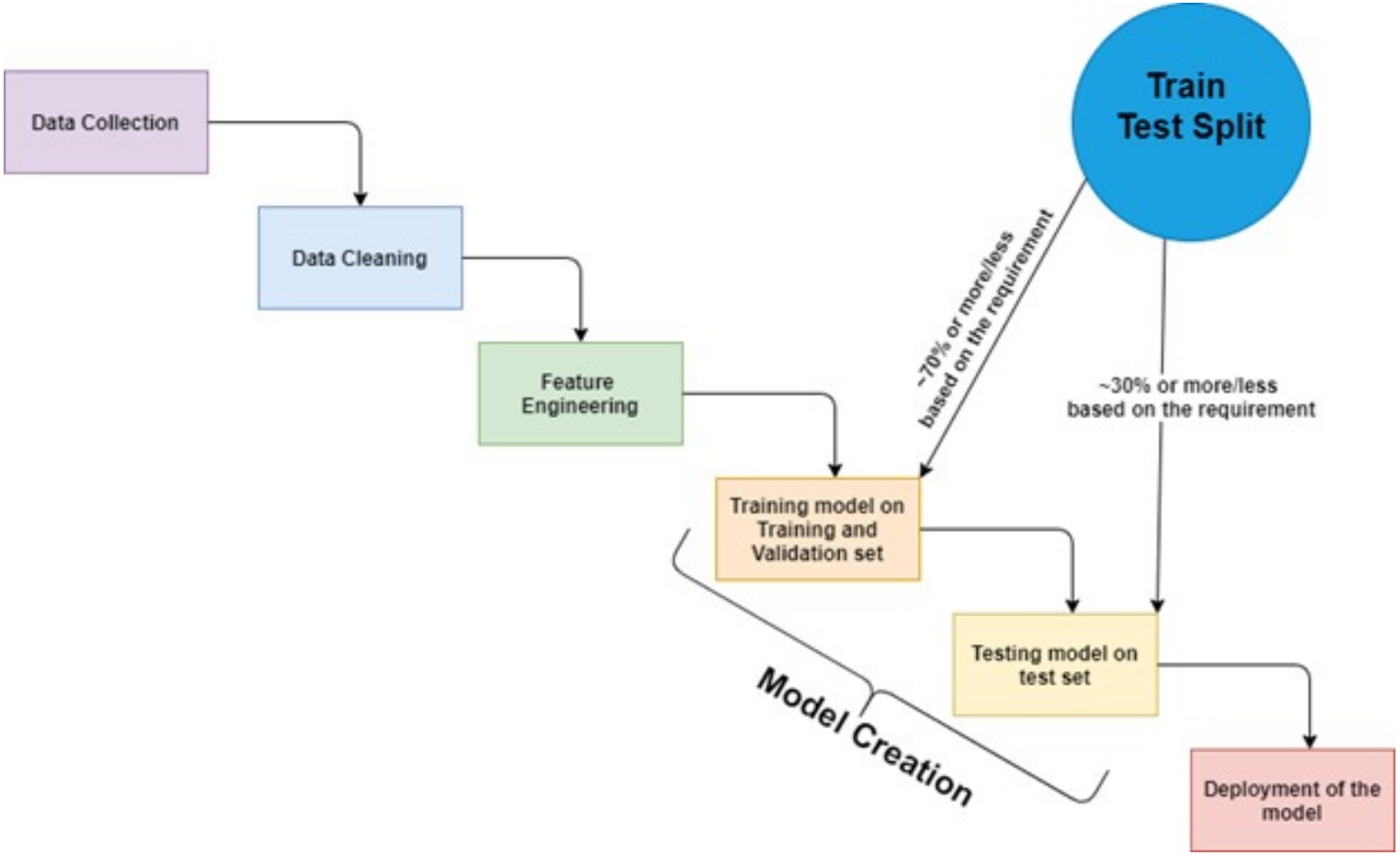
Cross Validation

- To know the real score of the model, it should be tested on the data that it has never seen before and this set of data is usually called testing set.
- But if we split our data into training data and testing data, aren’t we going to lose some important information that the test dataset may hold?

Cross Validation



Cross Validation – Where?



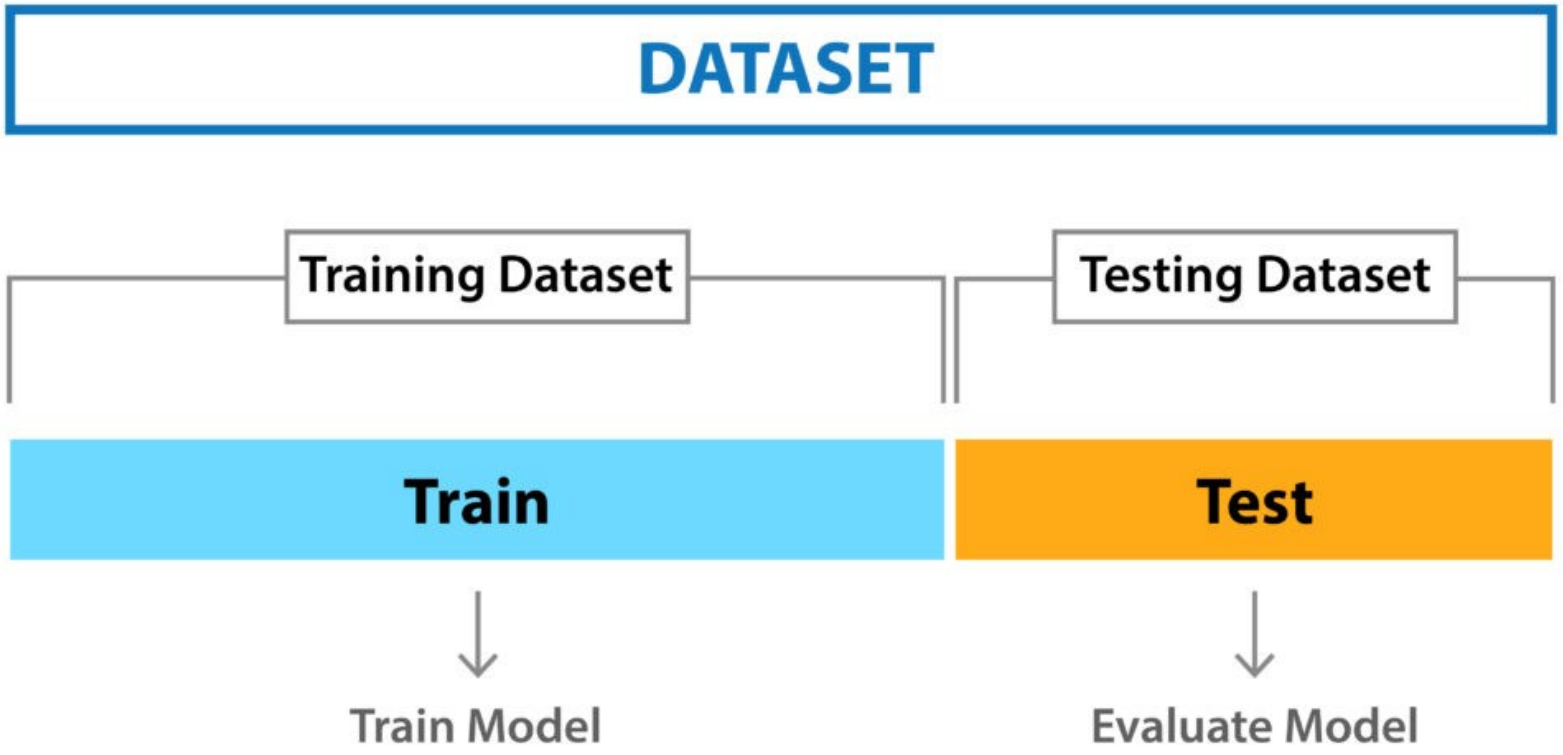
Cross Validation – Types

- There are different types of cross validation methods, and they could be classified into two broad categories –
  - Non-exhaustive
    - Holdout
    - K-Fold
    - Stratified K-fold
  - Exhaustive Methods.
    - Leave-P-Out cross validation
    - Leave-1-Out cross validation

Holdout Method

- This is a quite basic and simple approach in which we divide our entire dataset into two parts viz- training data and testing data.
- As the name, we train the model on training data and then evaluate on the testing set.
- Usually, the size of training data is set more than twice that of testing data, so the data is split in the ratio of 70:30, 75:25 or 80:20.

Holdout Method



Holdout Method

- In this approach, the data is first shuffled randomly before splitting.
- As the model is trained on a different combination of data points, the model can give different results every time we train it, and this can be a cause of instability.
- Also, we can never assure that the train set we picked is representative of the whole dataset.

Holdout Method

- Also when our dataset is not too large, there is a high possibility that the testing data may contain some important information that we lose as we do not train the model on the testing set.
- The hold-out method is good to use when you have a very large dataset, you're on a time crunch, or you are starting to build an initial model in your data science project.

Holdout Method

16	A
13	A
13	A
12	A
15	B
12	B
11	B
10	B
10	B
10	B
8	B
10	C
9	C
8	C
6	C
7	D
6	D
6	D
6	D
5	D

16	A
13	A
12	A
15	B
11	B
10	B
10	B
10	B
8	B
10	C
8	C
6	C
7	D
6	D
6	D
5	D

Train Set

13	A
12	B
9	C
6	D

Test Set

K-Fold Cross Validation

- K-fold cross validation is one way to improve the holdout method.
- This method guarantees that the score of our model does not depend on the way we picked the train and test set.
- The data set is divided into k number of subsets and the holdout method is repeated k number of times.

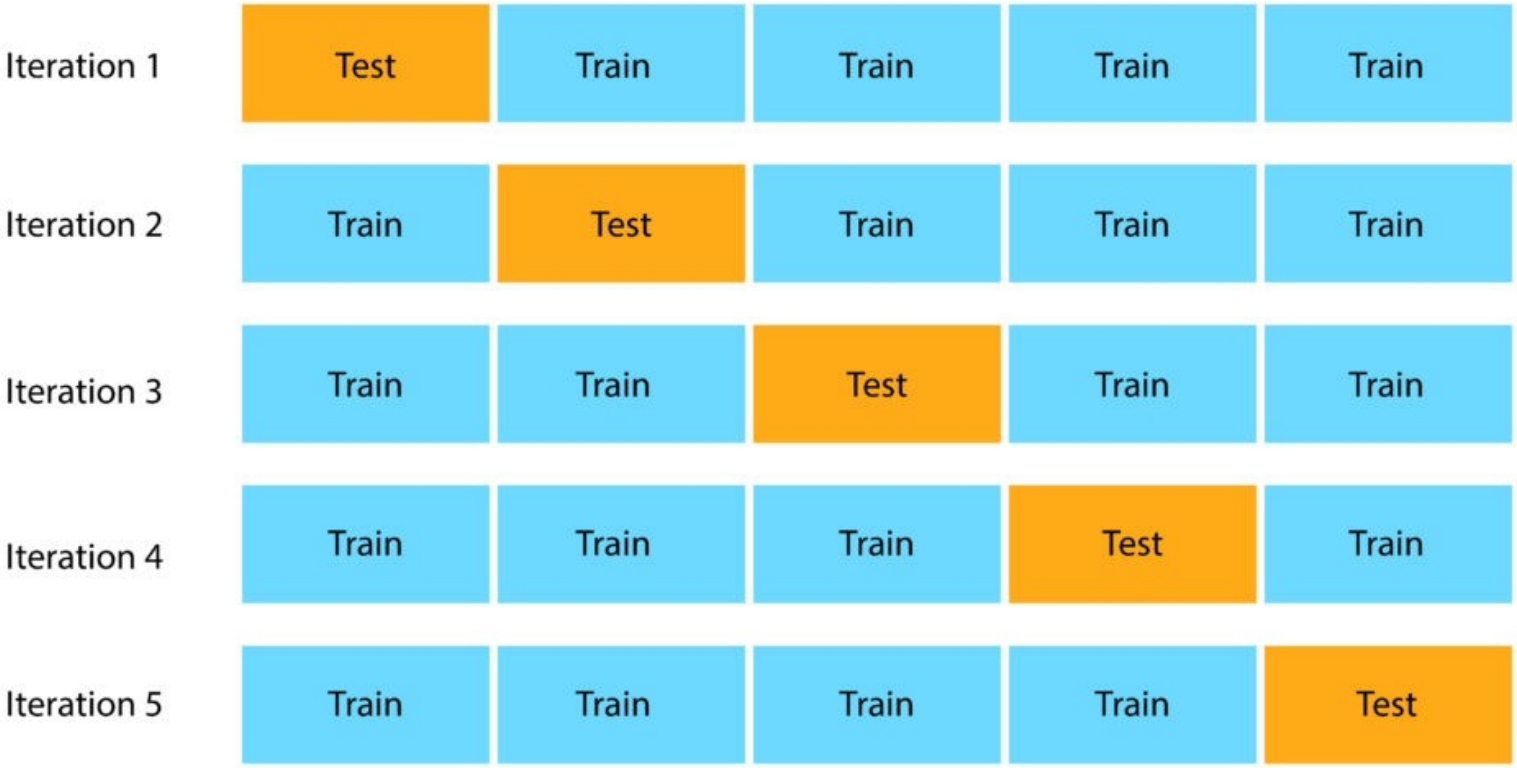
K-Fold Cross Validation

- Let us go through this in steps:
  - Randomly split your entire dataset into k number of folds (subsets)
  - For each fold in your dataset, build your model on k – 1 folds of the dataset. Then, test the model to check the effectiveness for kth fold
  - Repeat this until each of the k-folds has served as the test set
  - The average of your k recorded accuracy is called the cross-validation accuracy and will serve as your performance metric for the model.

K-Fold Cross Validation

- Because it ensures that every observation from the original dataset has the chance of appearing in training and test set, this method generally results in a less biased model compare to other methods.
- It is one of the best approaches if we have limited input data.
- The disadvantage of this method is that the training algorithm has to be rerun from scratch k times, which means it takes k times as much computation to make an evaluation.

K-fold Cross Validation



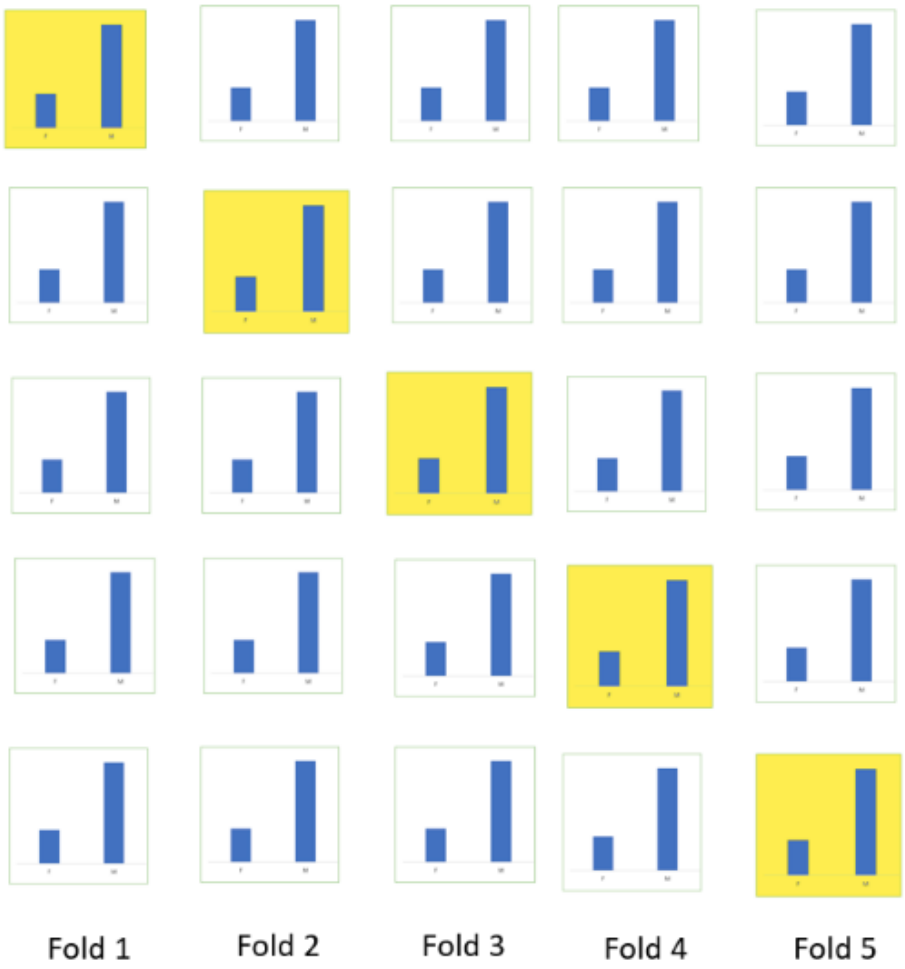
Stratified K-fold

- Using K Fold on a classification problem can be tricky.
- Since we are randomly shuffling the data and then dividing it into folds, chances are we may get highly imbalanced folds which may cause our training to be biased.
- For example, let us somehow get a fold that has majority belonging to one class(say positive) and only a few as negative class.
- This will certainly ruin our training and to avoid this we make stratified folds using stratification.

Stratified K-fold

- Stratification is the process of rearranging the data so as to ensure that each fold is a good representative of the whole.
- For example, in a binary classification problem where each class comprises of 50% of the data, it is best to arrange the data such that in every fold, each class comprises of about half the instances.

How it works?



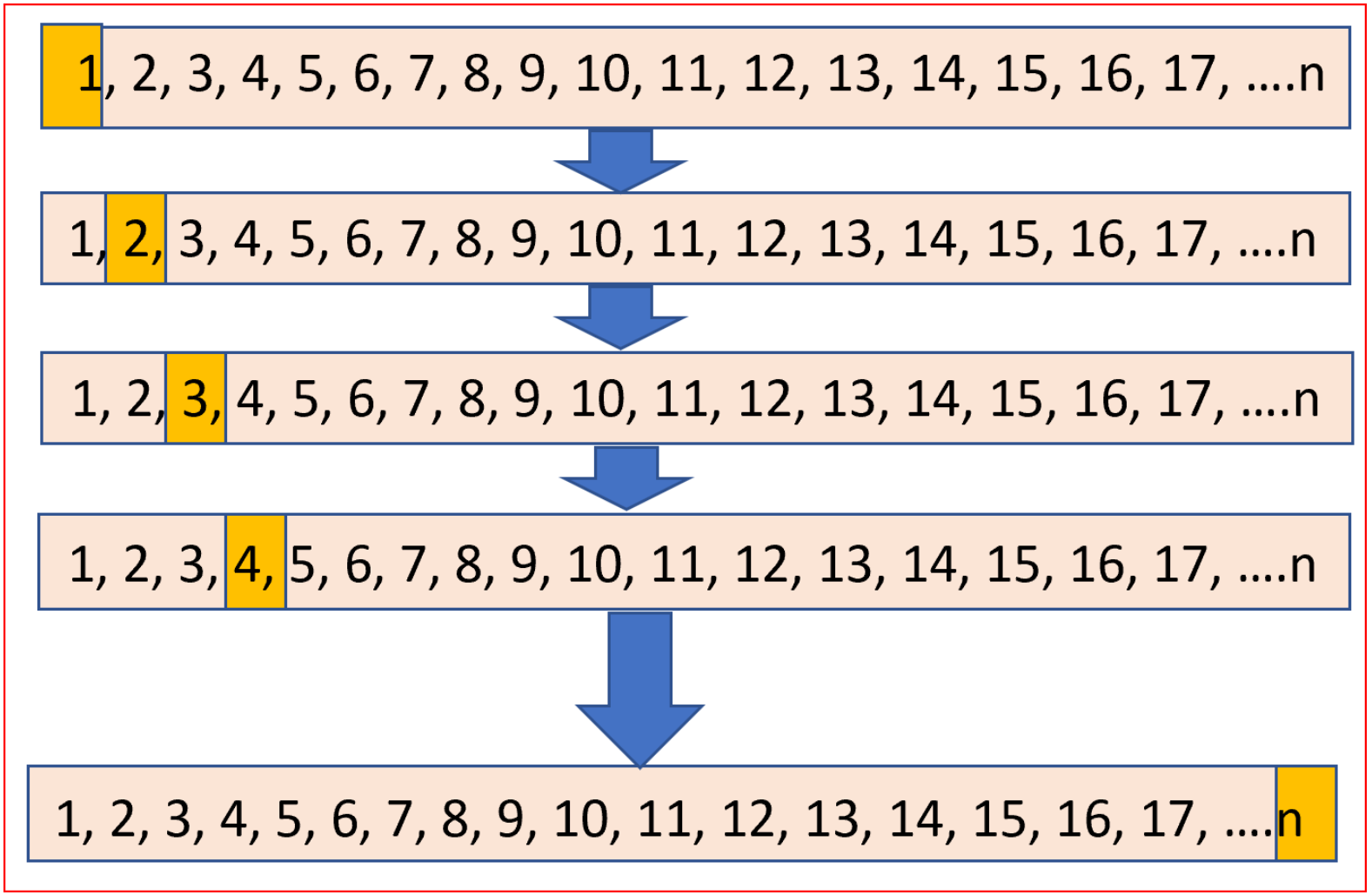
Exhaustive Methods

- Exhaustive cross validation methods and test on all possible ways to divide the original sample into a training and a validation set.

Leave One Out Cross Validation

- In this type of cross validation, the number of folds (subsets) equals to the number of observations we have in the dataset.
- We then average ALL of these folds and build our model with the average.
- We then test the model against the last fold. Because we would get a big number of training sets (equals to the number of samples), this method is very computationally expensive and should be used on small datasets.
- If the dataset is big, it would most likely be better to use a different method, like kfold.

How it works?


1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ...n
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ...n
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ...n
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ...n
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ...n

Leave P-Out Cross Validation

- When using this exhaustive method, we take p number of points out from the total number of data points in the dataset(say n).
- While training the model we train it on these (n – p) data points and test the model on p data points.
- We repeat this process for all the possible combinations of p from the original dataset.
- Then to get the final accuracy, we average the accuracies from all these iterations.

Leave P-Out Cross Validation

- This is an exhaustive method as we train the model on every possible combination of data points.
- Remember if we choose a higher value for p, then the number of combinations will be more and we can say the method gets a lot more exhaustive.

Rolling Cross Validation

- For time-series data the above-mentioned methods are not the best ways to evaluate the models. Here are two reasons as to why this is not an ideal way to go:
  - Shuffling the data messes up the time section of the data as it will disrupt the order of events
  - Using cross-validation, there is a chance that we train the model on future data and test on past data which will break the golden rule in time series i.e. “peaking in the future is not allowed”.

Rolling Cross Validation

- Keeping these points in mind we perform cross validation in this manner
  - We create the fold (or subsets) in a forward-chaining fashion.
  - Suppose we have a time series for stock prices for a period of n years and we divide the data yearly into n number of folds. The folds would be created like:

    iteration 1: training [1], test [2]

    iteration 2: training [1 2], test [3]

    iteration 3: training [1 2 3], test [4]

    iteration 4: training [1 2 3 4], test [5]

    iteration 5: training [1 2 3 4 5], test [6]

    .

    .

    .

    iteration n: training [1 2 3 ….. n-1], test [n]

Useful resources

- www.mygreatlearning.com
- https://neptune.ai
- www.geeksforthegeeks.org
- www.scikit-learn.org
- www.towardsdatascience.com
- www.medium.com
- www.analyticsvidhya.com
- www.kaggle.com
- www.stephacking.com
- www.github.com

Useful resources

- www.mygreatlearning.com
- https://neptune.ai
- www.geeksforthegeeks.org
- www.scikit-learn.org
- www.towardsdatascience.com
- www.medium.com
- www.analyticsvidhya.com
- www.kaggle.com
- www.stephacking.com
- www.github.com

Thank you

This presentation is created using LibreOffice Impress 5.1.6.2, can be used freely as per GNU General Public License

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com