Lecture #2 Data-preprocessing

1.0 Introduction

2.0 Data pre-processing

- 2.1 Data visualization and Exploration

2.2 Data cleaning
- Detect and removal of blanks
- Detect and removal of duplicats
- Detect and removal of non-numeric
- Detet and removal of Outliers

2.3 Feature selection

2.4 Data Smoothing
- 2.4.1 Moving average
- 2.4.2 Exponential smoothing

3.0 Data Splitting methods

4.0 Summary

Lecture 2 Feature Engineering / Data-Preprocessing

Introduction

Data is used for ML Modeling.
Row data contains useful information and Undesired / Non-informative / Noise as well

Befor using Rowdata for ML Modeling, it Must be preprocessed to:

- → Make it clean.
- → Select the useful features and
- – Minimize the noise level.

Therefore this Chapter presents 3-parts.

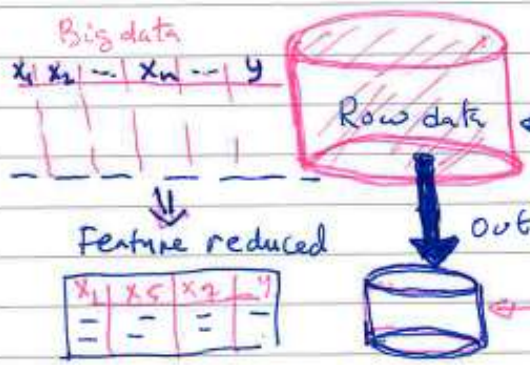
Feature Engineering / Data preprocessing

(1) Data Cleaning
- – Remove unrecorded / blanks
- – Remove Non-numerics
- – Remove duplicates
- – Remove outliers

→ Save file as "Cleaned.XLSX"

(2) Feature Selection
- → Select appropriate / useful features. → Features reduced
- → Save file as "Cleaned Seleded."

(3) Apply Smoothing filter
- → Save file as "Cleaned Seleded Fitered"



Apply Data preprocessing
a) Data Cleaning
b) Feature Selection
c) Smoothing

Cleaned Selected Filtered
→ This file will be used for ML Modeling

Lecture Data Preprocessing (Feature Engineering)

Before we start data processing, let us see the difference between Data and Information.

Data	Information
~ a collection of facts. eg. observations, figure,	~ A result of processing analysing and interpreting of data
~ data on it's own is meaningless	~ when data is analysed and interpreted it becomes a meaningfull information
~ Data does not depend on Information	~ Information depends on data
~ Data is not sufficient for making decision	~ But, you can make decision based on Information

Information is defined as knowledge gained through study, research, or instruction

Essentially, Information is the result of analysing and interpreting piece of data

Example 1 A set of data includes temperature readings in a location over several years

→ Without any additional context, those temperatures have no meaning

- However, when you analyse and organize that information, you could determine
  - → Seasonal temperature pattern
  - or → even broader climate trends
  - ⇒ Min / Max / average values...

→ To Summarize:

→ Only when the data is organized and compiled in a usefull way, it can provide INFORMATION that is benificial

Hopefully, the above example illustrates about Data and Information.

Example 2 Table 1 shows data X, Y

- to analyse the nature of the data
- Visualization allows about the data trend.

X	Y
0	0
1	3
2	4 ← Blank (Unrecorded)
3	4
3	4 ← duplicated
4	xx7 ← Object
5	2 56 ← Not numeric object
6	2975 ← outlier
7	8
9	10
11	12
Blank Unrecorded →	15
18	20
18	20 ← duplicated

⇐ As we can see data contains:

- → Unrecorded / blanks
- → duplicates
- → Non-numeric
- - Outliers
- - and noises

- These makes the quality of data "poor"

"Poor" data quality, results in "poor" information.

Data must therefore be cleaned before analaging and interprating

→ After cleaning the above dataset, the clean data will be

X	Y
0	0
1	3
3	4
7	8
9	10
11	12
18	20

→ However, ~~if~~ we have big-dataset, the data cleaning process will not be performed manually as shown above.
→ It will be performed by data analysis and processing tools, which are ~~perfroed~~ performed by the Computers.

Lesson 1 – Data must be cleaned befor analysing and interprating

Therfore the first part of the lesson is Data pre-processing / Feature Engineering.

This is the first step of Machine learning workflow as shown below.

2.0 Data - Pre processing

Before ML Modelling data must be pre processed (1) to make it clean
(2) to select the right features

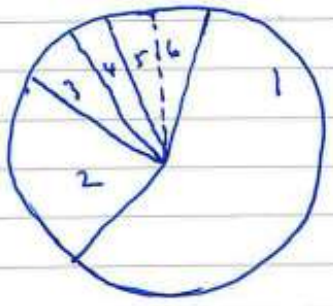
Feature Engineering:
is the process of selecting, manipulating, and transforming raw data into features that can be used in supervised learning

Feature is any measurable input that can be used in a predictive model.

Feature Engineering, in a simple term, is the act of converting observations into desired features using statistical or ML approach.

→ The goal is to improve performance

Data scientists spend most of their time on data, which is important to make models accurate. Example, Time spent



1 - Cleaning and organization
2 - Collecting data set
3 - Mining data
4 - Refining algorithm.
5 - Others
6 - Training Sets

The following present the typical data Preprocessing process. Here we will look at

(1) Data cleaning
(2) Feature Selection
(3) Data Smoothing

Step by Step guide: of (1) Data cleaning

Step 1 Import data important libraries Panda and matplotlib

Step 2. Import data

df = pd.read_excel ('Filename.xlsx')

Step 3 Information about the data .info()

- For modeling data must be numeric, not objet
- Data must have the same format = float

in Print (df.info())

Out:

	Header	Non-Null	dtype
0	X_1	1500 Non-null	float
1	X_2	1490 Non null	int64
2	X_3	1300 Non null	objet
3	Y	1495 non null	objett

Except for X_1, the rest int64 and object should be Converted to numeric

Step 4 Convert Objeat and int64 to numeric

(a) Convert int64 to float

Syntax:

df['X_2'] = df['X_2'].astype (float)

(b) Each objects Convert to numeric

- objects are due to
  - Blanks
  - Strings
  - String with numerics

Syntax

df['Column name'] = Pd.to_numeric(df['Column name'], errors='coerse')

→ Coerse will convert the instances (blank / strings) to Nan. (Not a Numeric)
→ Later the Nan will be removed

Example

df['x_3'] = pd.to_numeric(df['x_3'], errors='coerse')

df['y'] = pd.to_numeric(df['y'], errors='coerse')

→ Check if int64 and objects are converted in df.info()

Out

	Header	Non-Null	dtype
0	x_1	1500 Non-Null	float
1	x_2	1490 Non-Null	float
2	x_3	1500 Non-Null	float
3	x_4	1495 Non-Null	float

— As we can see both int64 and object are converted to float

→ If you check the dataset →
- all the data int → are float 3.0
3 → 3.0
→
→ All object/blank → Nan

→ Step 5: Display statistical values/analysis
df.describe()

→ Step 6 Blank (Nan) data Removal
(6a) df.dropna(axis=0, inplace=True)
Remove row — do and return

(6b) Check if blanks are removed
df.info()

Step 7 Duplicates data must be REMOVED

(6a) First, check if duplicates exist
— use duplicated()

* calculate duplicates

```
dups = df.duplicated()
```

* Report if exist. True = detected! False = Not

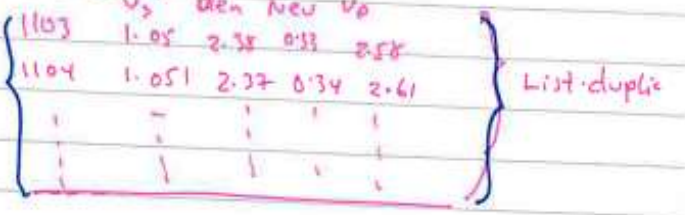
```
Print(dups.any())
```

* List all duplicate row

```
print(df[dups])
```

Out

- True



(6b) How to DELETE row that Contain Duplicate data ?

— use drop_duplicates()

- Check size of data before duplicate Removal

```
Print(df.shape)
```

- Delete duplicat rows

```
df.drop_duplicates(inplace = True)
```

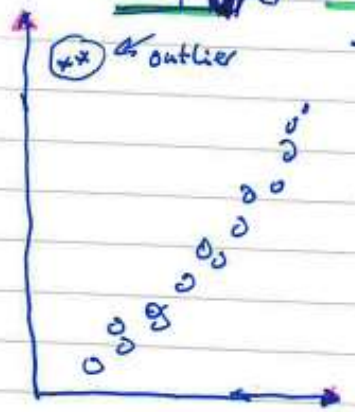
- check size after removal

```
Print(df.shape)
```

Output

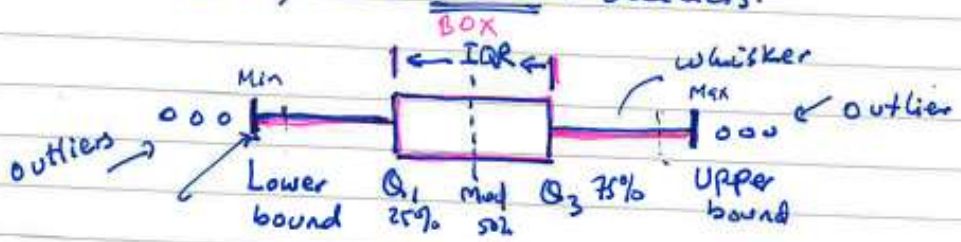
(1111, 4) → Before remove
(1103, 4) → After remove

Step 8 Outliers removal



- Outliers are data that differ significantly from the rest of the dataset.
- → These dataset may be plotted as individual points beyond the ~~whisk~~ whiskers on the box-plot
- → Box-plot display variation in samples of the population

1st - we need to identify outliers
2nds, we remove outliers.



$IQR = Q_3 - Q_1$.

To remove outlier → below lowerbound

Lower bound = $Q_1 - 1.5\,IQR$

~~Upper~~

To remove above upper bound

Upper bound = $Q_3 + 1.5\,IQR$.

(1) Detection

```
# IQR
        Q1 = np.percentile(df, 25)
        Q3 = np.percentile(df, 75)
or      Q1, Q2 = np.percentile(df, [25, 75])
IQR = Q3 - Q1
LB = Q1 - 1.5 * IQR
Ub = Q3 + 1.5 IQR
        Print('oldshape:' df.shape)
```

Using target Column.

upper bound

upper = np.where(df['Vp'] >= (Q3+1.5*IQR))

lower bound

lower = np.where(df['Vp'] <= (Q1-1.5*IQR))

(2) Removing outlier → use drop( )

df.drop(upper[0], inplace = True)
df.drop(lower[0], inplace = True)

print('New shape:' df.shape)

[dataframe.drop(row_index, inplace=True)

* The row from the dataset given in the row index will be removed
* Inplace = True → used to instruct Python to make required change in the original dataset.

Ex. df.drop(list[0], inplace=True)

- row index can be ONLY one value
- OR List of values

OR

Outlier detection

Outliers = [x for x in df['Vp'] if x < lower or x > upper]

print('Identified outlier: %d' %len(outliers))

Outlier Removal

df = df[(df['Vp'] > lower) & (df['Vp'] < upper)]

Print('Non-outlier: %d' %len(df))

(2.3 Feature Selection

After the dataset has been cleaned, the final data-preprocessing step is to select the right features that are useful to a model in order to predict the target variable

→ Feature Selection is primarily focused on removing Non-informative or Redundant predictors (input features) from the dataset. So that the reduced number of input variables will be used for modeling.

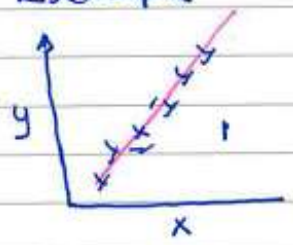
There are many methods. Among others, the most common technique is a correlation Coefficient method, which will be used in this course.

Pearson's Correlation Coefficient, r ⇒
→ is the most common way of measuring a linear Correlation b/w x, y

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \cdot \sum (y_i - \bar{y})^2}}$$

- It's value is between -1 and 1 that measures the STRENGTH and DIRECTION between two variables

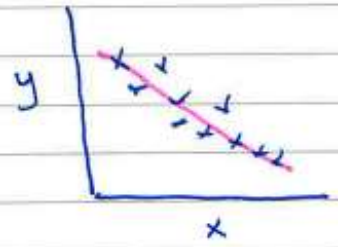
Example



Positive Correlation

→ as x increase, y also increase

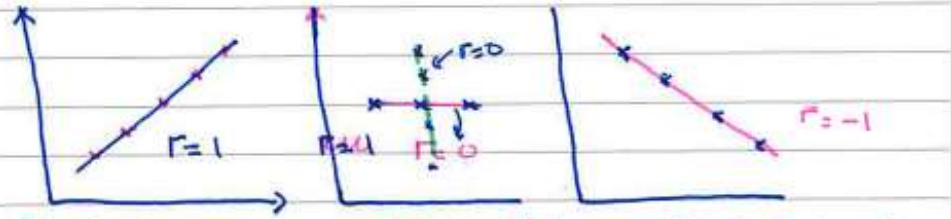
r is between 0 and 1



Negative correlation

→ as x increase, y decrease

r is between 0 and -1

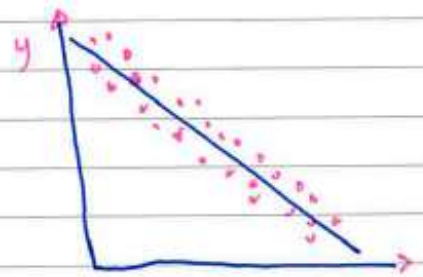


Perfect positive Correlation

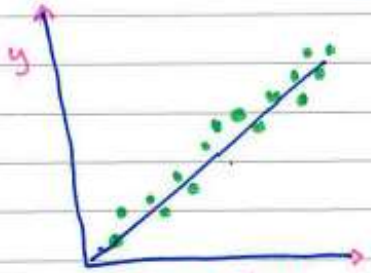
No correlation r=0

Perfect negative correlation

Example when r is greater than 0.5 or less than -0.5, the points are closer to the line of best fit

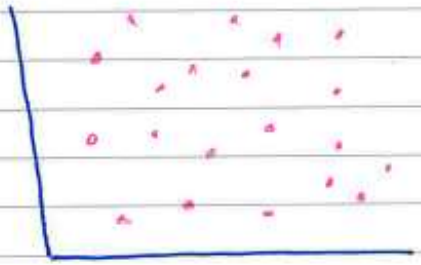


$r < -0.5$



$r > 0.5$

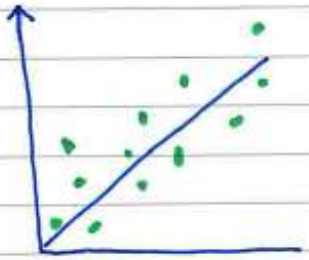
Example when $r = 0$, the line of best fit is not helpful in describing the relationship between the variables



$r = 0 \Rightarrow$ No correlation

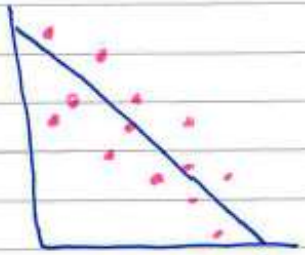
Example · Weak correlation

- When r is between 0 and 0.3 or between 0 and -0.3, the points are far from the best fit line



$0.3 > r > 0$

Weak positive correlation



$0 > r > -0.3$

Weak Negative correlation

Summary

Pearson Correlation Coeff r	Strength	direction
greater than 0.5	Strong	Positive
between 0.3 and 0.5	Moderate	>>
Between 0 and 0.3	Weak	Positive
0	None	None
between 0 and -0.3	Weak	Negative
between -0.3 and -0.5	Moderate	Neg.
less than -0.5	Strong	Neg

Example Feature Selection with Sklearn

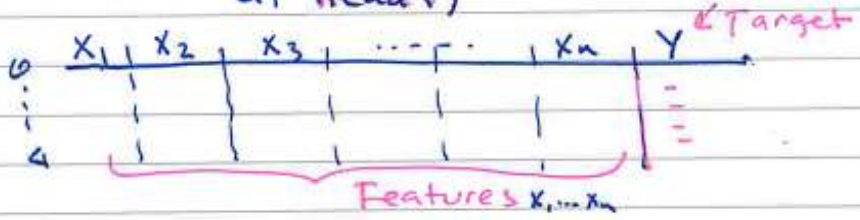
Step 1 Import important libraries.

```
Import Pandas as pd
import Matplotlib.pyplot as plt
Import Seaborn as sns
```

Step 2. Import data file and display (.head())

```
df = pd.read_excel('Cleaned.xlsx')
```

```
df.head()
```



Step 3 Calculate correlation between the Target and Feature variables

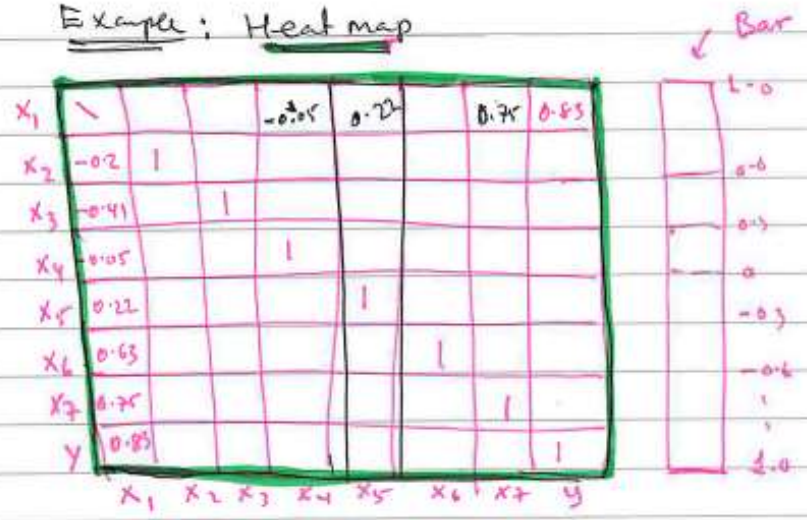
```
# Using Pearson Correlation, generate HEAT MAP
plt.figure(figsize=(12,10))
cor = df.corr()
sns.heatmap(cor, annot=True,
            cmap=plt.cm.Reds)
plt.show()
```

`.corr()`

`annot = True` → If true, write the "r" values in each cell

`cmap` → The mapping from data value to color space

Example: Heat map



Step 4: Apply criteria to select the appropriate features that would be correlated with target variable (y)

```
# Correlation with output/Target variable
Cor_target = abs(cor['Y'])

relevant_feature = cor_target[cor_target>0.5]

relevant_features
```

Output: $X_1 = 0.83$

$X_6 = 0.63$

$X_7 = 0.75$

$y = 1$

Step 5 Save Selected / cleand features

```
# Select colum
SelectedColumns = ['X1', 'X6', 'X7', 'Y']

# Selecting final Column
Data_ForML = df[SelectedColumns]
Data_ForML.head()
```

	X_1	X_6	X_7	y
0				
1				
:				
4				

Step 6. Saving the Selected Column for / to be used ML. modeling

```
DataTobeSaved = Data_ForML

# Export the DataFrame to an Excel file

DataTobeSaved.to_excel('FinalCleanedSelected.xlsx')
```

(24 Data Smoothing / Filtering

Sometimes data may contain spikes, which is due to / could be due to Noises.

- The Commonly used approach to minimize the spikes is by applying a fiter that Smoothout the signal

→ The two types of Smoothing functions are
(a) Moving average
(b) Exponential Smoothing.

Moving averge is the mean of a certain window data set.

$$y_{mva} = \frac{1}{M}\sum_{i=1}^{M} y_i \quad , \text{ m is window size}$$

1	y_1
2	y_2
3	y_3
4	y_4
5	y_5
⋮	⋮

$m = 3$

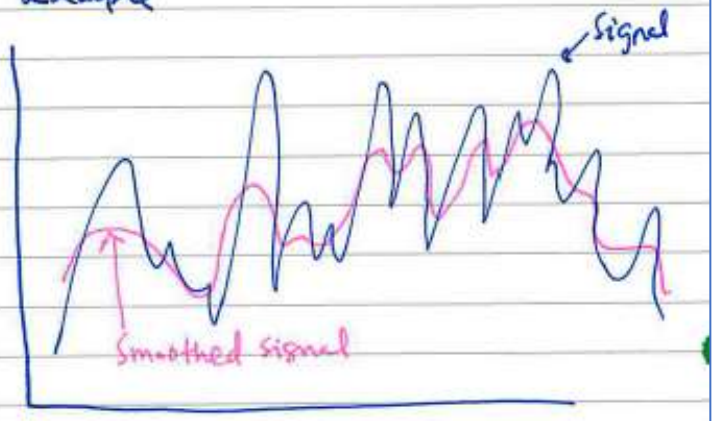
$y^1_{mva} = \frac{1}{3}(y_1 + y_2 + y_3)$

$y^2_{mva} = \frac{1}{3}(y_2 + y_3 + y_4)$

$y^3_{mva} = \frac{1}{3}(y_3 + y_4 + y_5)$

⋮

Example



Example: Here we will apply MOVING averge Smoothing fiter on the final selected features

→ The file name was "FinalCleanedSelated.xlsx')

Step 1 Import important libraries

```
Import pandas as pd
Import matplotlib.pyplot plt
```

Step 2: Import data file

```
df = pd.read_excel('Final Cleaned Selected.xlsx')
```

Step 3 Apply MOVING AVERAGE Fiter

```
DataFiter = df.rolling(window=5).mean()
```

Step 4. Save the fitered data as New Filename

```
DataFilter.to_excel('Final Cleand Selected Fiterd.xls')
```

The filterd file is saved as

Final Cleaned Selected Filtered.xlsx

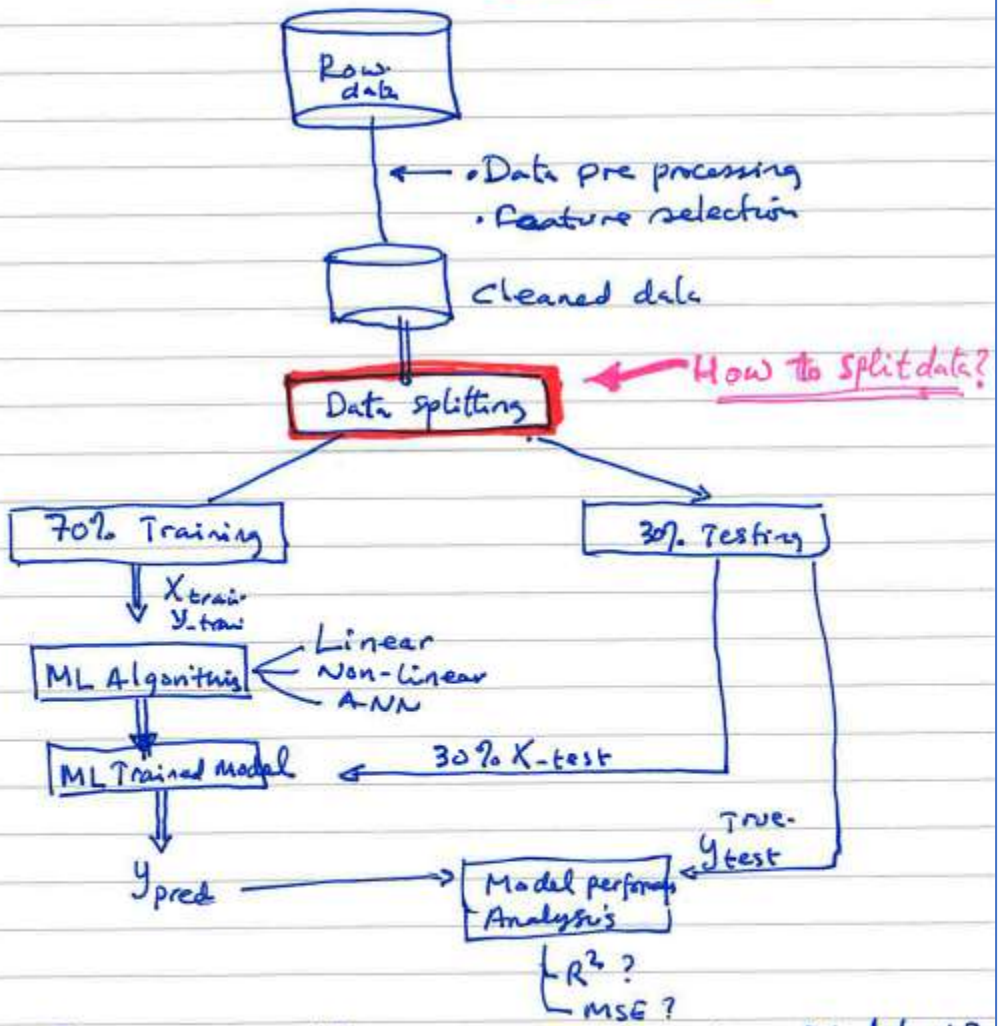
Note that For the ML. modeling, you will use either:

(a) Final Cleaned Selected.xlsx → Unfitered
(b) Final Cleaned Seleded Fitered.xlsx → filtered

3.0 Data Splitting

For machine learning modelling, the cleaned and selected features will be splitted into training and testing

- Training dataset (70%) will be used for modeling
- The remaining dataset (30%) will be used for Testing the model



In the next section, we will learn How to split dataset?

Assume that the final cleaned / and selected feature data set has been saved as "cleaned.xlsx"

→ import the file as

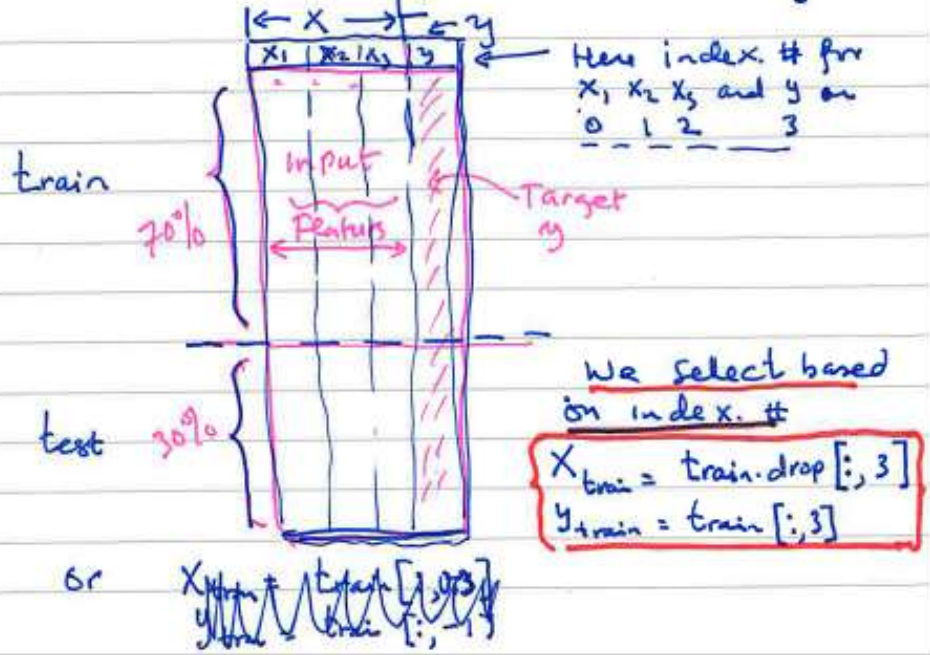
```
df = pd.read_excel('cleaned.xlsx')
```

→ Here, we will see different methods of splitting training (X_{train}, y_{train}) and testing (X_{test}, y_{test}) datasets.

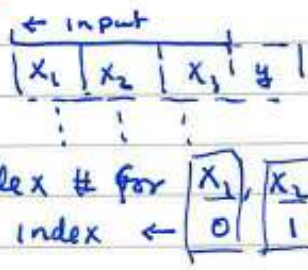
→ By convention we use capital letter X → for input features, and small letter y → for target output

Method #1

```
train = df[:(int(len(df)*0.7))]
test = df[(int((len(df))*0.7)):]
```



Method #2



X_{train} = train ['x_1', 'x_2', 'x_3'] .asfloat()

y_{train} = train ['y'] .asfloat()

X_{test} = test ['x_1', 'x_2', 'x_3'] .asfloat()

y_{test} = test ['y'] .asfloat()

Method #3

X_{train} = train [:, 0:3] → excluding index #3, 0, 1, 2 → for x_1 x_2 x_3 respectively

y_{train} = train [:, 3] ← use index #3, which is y

X_{test} = test [:, 0:3]

y_{test} = test [:, 3]

Method #4

X_{train} = train .iloc [:, 0:3] .value

y_{train} = train iloc [:, 3] .value

X_{test} = test .iloc [:, 0:3] .value

y_{test} = test .iloc [:, 3] .value

Method #5 → train_test_split method

This method is based on built-in sk-learn library / function

» For this the function should be first imported, which is train_test_split

- Import library

```
from sklearn.model_selection import
        train_test_split
```

- Import data

```
df = pd.read_excel('cleaned.xlsx')
```

Option 1 step 1

- Separate X and y features / target

```
X = df [['x1', 'x2', 'x3']] ← double
y = df['y']                    P
```

Option 2 step 2

- Split data for training and split

```
X_train, X_test, y_train, y_test = train_test_split(..
                X, y, test_size=0.3,
                random_state=1)
```

After this Modeling follows, Example

What is the purpose of random_states?

"Random_State" is a parameter in ~~that~~ train_test_split that CONTROLES the random number generator used to shuffle the data before splitting it.

> With random_state = None (default), we get different train and test across different excutions

> With random_state = 0, we get the same train and test sets across different excutions

→ The most popular are 0, or 42

> With random_state = 42, we get the same train and test sets across the different excution, but in this time the train and test sets are different from the previous case with random_state=0

The train and test sets directly affect the Model's performance score

Because we get different train and test sets with different integer values for random_state in the train_test_split() function, the value of the random state hyperparameter indirectly affect the Model performance score.

4.0 Summary

Data preprocessing is the first stage of Machine learning modeling.

The main focus on data preprocessing that we learned are

- Data cleaning
- Feature selection
- Data Smoothing / optional

Once the data is selected, the next stage is Modeling. This is performed by spliting the selected dataset into Training (for modeling) and testing (for prediction). For data spliting, chapter 3 of this note presents different methods.