Neuron Network

Activate functions

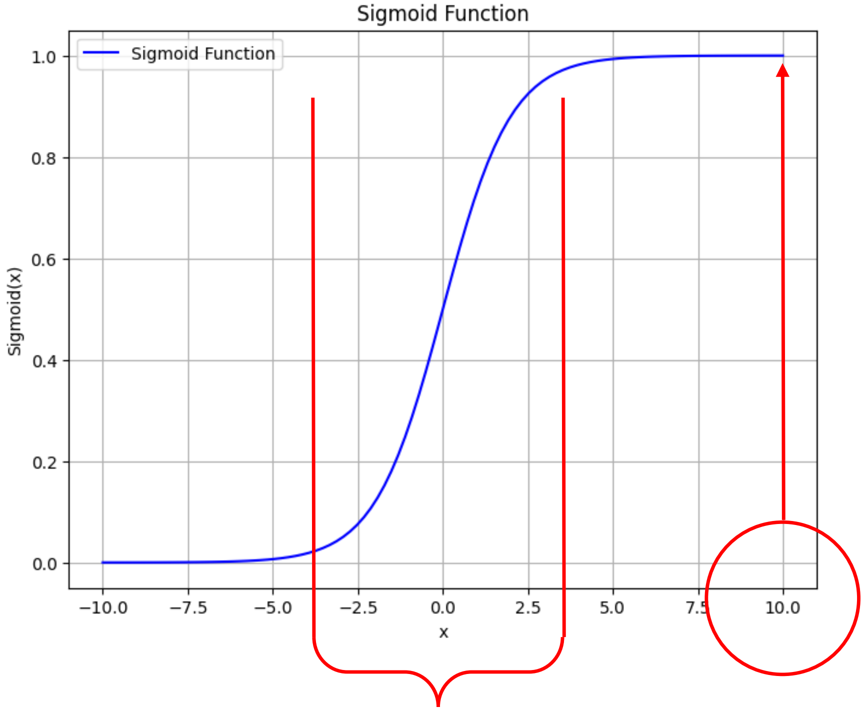
- Sigmoid Function (Logistic Function):

$\sigma(x) = \frac{1}{1+e^{-x}}$

```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

- It compresses the input to the range between 0 and 1, commonly used in binary classification tasks, although less prevalent in deep neural networks due to the vanishing gradient problem.

Sigmoid

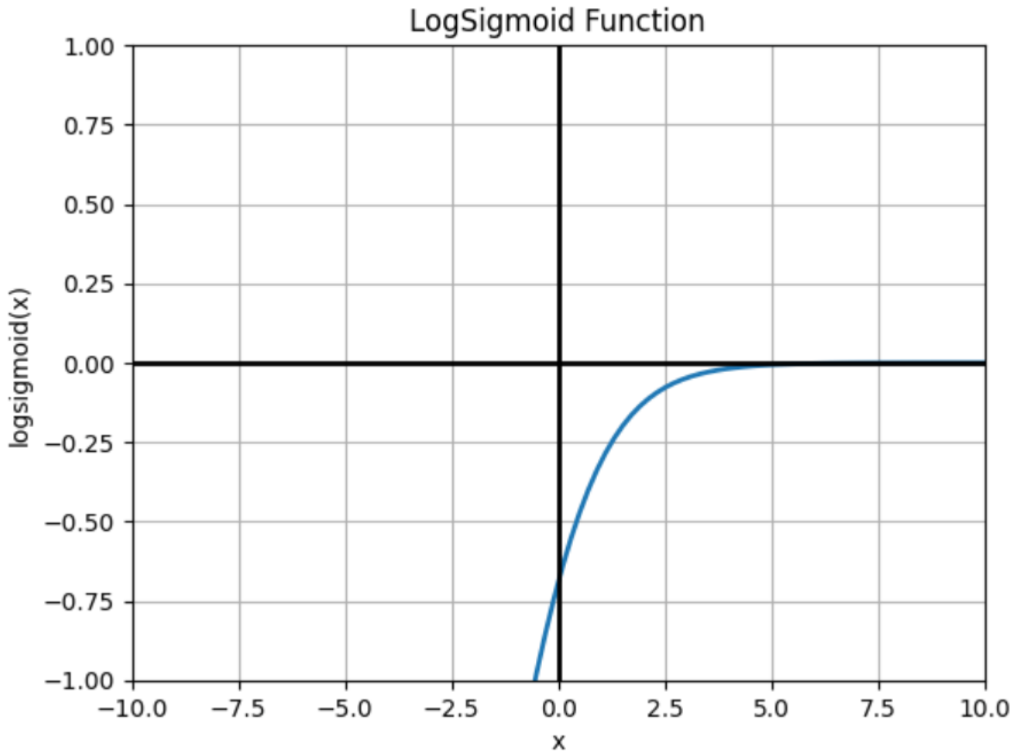


-3 ~ +3

1000
100
10

LogSigmoid

$$f(x) = \log(\frac{1}{1+e^{-x}})$$



LogSigmoid

```
import numpy as np
import matplotlib.pyplot as plt

def logsigmoid(x):
return np.log(1 / (1 + np.exp(-x)))

# 生成 x 值
x = np.linspace(-10, 10, 100)

# 計算 logsigmoid 函數的值
y = logsigmoid(x)

# 繪製 logsigmoid 函數
plt.plot(x, y, linewidth=2)
plt.title('LogSigmoid Function')
plt.xlabel('x')
plt.ylabel('logsigmoid(x)')
plt.grid(True)

# 設定 x 軸範圍
plt.xlim(-10, 10)

# 設定 y 軸範圍
plt.ylim(-1, 1)

# 加粗 x=0 和 y=0 線條
plt.axhline(0, color='black', linewidth=2)
plt.axvline(0, color='black', linewidth=2)

plt.show()
```

Tanh

- Tanh Function (Hyperbolic Tangent Function):

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

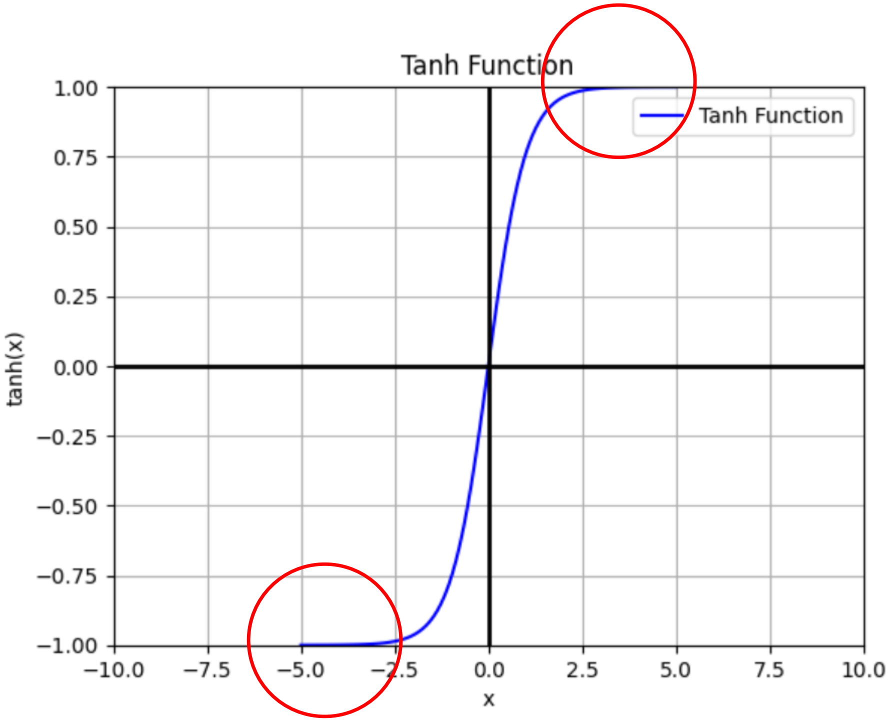
```python
def tanh(x):
    return np.tanh(x)
```

- It compresses the input to the range between -1 and 1, similar to the sigmoid but with a wider output range, also facing the vanishing gradient problem.

Tanh

```python
import numpy as np
import matplotlib.pyplot as plt
# Define the range for x values
x_values = np.linspace(-5, 5, 100)
# Compute y values using tanh function
y_values = np.tanh(x_values)
# Plot the tanh function
plt.plot(x_values, y_values, label='Tanh Function', color='b')
# Add labels and title
plt.xlabel('x')
plt.ylabel('tanh(x)')
plt.title('Tanh Function')
# Add grid
plt.grid(True)
# Add legend
plt.legend()
# 設定 x 軸範圍
plt.xlim(-10, 10)
# 設定 y 軸範圍
plt.ylim(-1, 1)
# 加粗 x=0 和 y=0 線條
plt.axhline(0, color='black', linewidth=2)
plt.axvline(0, color='black', linewidth=2)

# Show plot
plt.show()
```

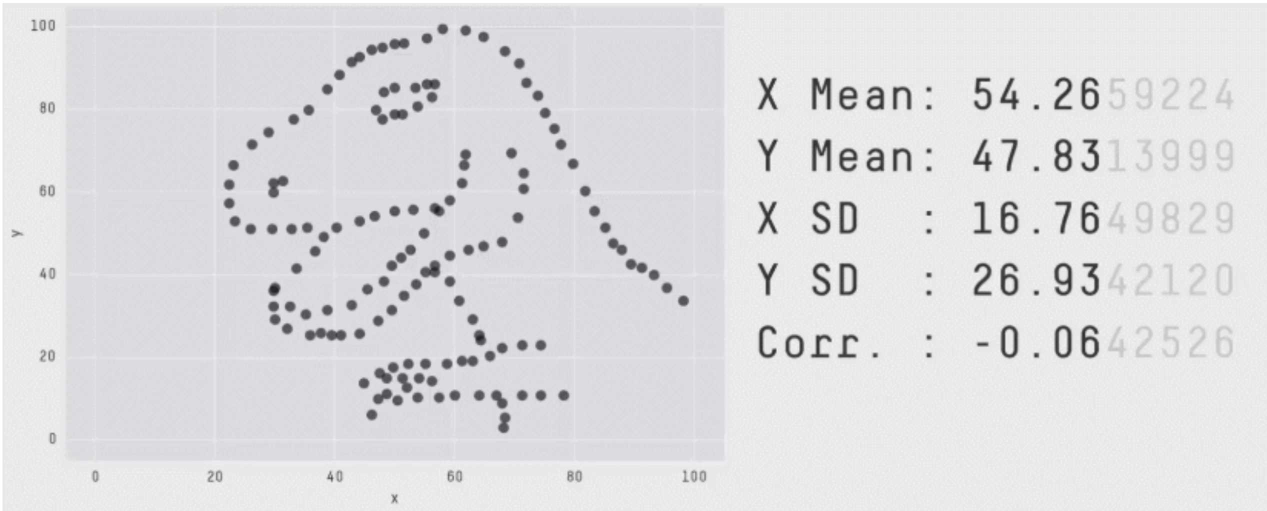


The input values to the tanh function should also **not be too large & small**; otherwise, the model may encounter training difficulties because the values will **approach 1 & -1.**

Exploration Data Analysis (EDA)

探索式資料分析

Quickly and easily understand the characteristics of data from various perspectives using <u>descriptive statistics</u>, <u>statistical plotting</u>, <u>visualization</u>, and <u>other techniques</u>.



圖片來源：
https://baubimedi.medium.com/速記ai課程-統計與資料分析-四-3cf14683b98f

- Data volume:
- Target features (目標特徵):
- Noisy data/Outliers (雜訊數據/ 異常值):
  - Noisy data/outliers refer to values that are **observed in error**, such as a person's age being recorded as **300 years old**, which is likely an erroneous observation. Outliers, on the other hand, are values that may be correct but deviate significantly from the average.
  - For a normally distributed dataset, outliers can be values that are 3 to 6 standard deviations away from the mean. When these values exceed 5% of the dataset, we need to address them.
- Missing values:
- Qualitative features (定性特徵): Qualitative features are **non-numeric data** represented **in text, graphics, audio,** or other non-numeric formats. We need to check if the dataset contains qualitative features. If qualitative features are present, we'll need to use data encoding techniques to process them.

Outliers processing

- The term "outliers" refers to data points in a sample that **significantly deviate from the rest of the data points**; outliers are also known as "anomalies."

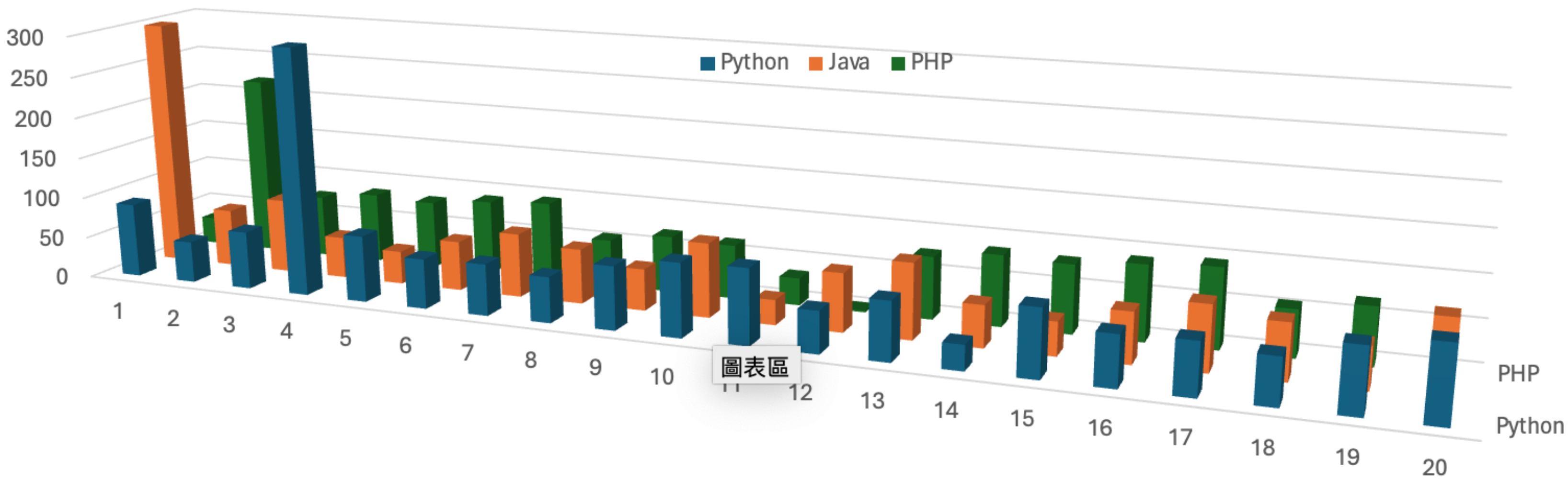
- Having too many outliers can introduce bias to deep learning models.

- If necessary, we need to identify and analyze these outliers for processing.

- However, **not all outliers require processing**, as some outliers may represent meaningful values in practical applications.

```python
import numpy as np
import pandas as pd

scores={
'Python':[90,50,70,300,80,60,62, 55, 76, 88, 90, 50, 70, 30, 80, 60, 62, 55, 76, 88],
'Java':[300 , 70 , 90 , 50 , 40 , 60 , 77 , 66 , 50 , 89 , 30 , 70 , 90 , 50, 40, 60, 77, 66, 50, 89],
'PHP':[33, 220, 75, 85, 82, 90, 95, 56, 68, 65, 33, 2, 75, 85, 82, 90, 95, 56, 68, 65]
}
df=pd.DataFrame (scores)
print (df.shape)
```



Example: Greater than or less than 3 times the standard deviation are considered outliers.

```
outliers = {}
# 取出每行數據
for i in range(df.shape[1]):
    # 最小閥值
    min_t = df[df.columns[i]].mean()-(3*df[df.columns[i]].std())
    # 最大閥值
    max_t = df[df.columns[i]].mean()+(3*df[df.columns[i]].std())
    count = 0
    # 評估每行的數值是否存在異常值
    for j in range(df.shape[0]):
        data=df.iloc[j,i]
        if data < min_t or data > max_t:
            count += 1
    # 計算異常值百分比
    percentage = count / df.shape[0]
    # 存入字典變數outliers
    outliers[df.columns[i]] = "%.3f" % percentage

print(outliers)

Out:
{'Python': '0.050', 'Java': '0.050', 'PHP': '0.050'}
```

```
# Select all columns in the first row
df.iloc[0]

# Select all rows in the first column
df.iloc[:, 0]

# Select the element at the first row and first column
df.iloc[0, 0]

# Select elements from multiple rows and columns
df.iloc[[0, 1], [0, 1]]

# Select a range of rows and columns using slices
df.iloc[0:2, 0:2]
```

Set outliers as NaN values.

```
# 取出每行數據
for i in range(df.shape[1]):
    # 最小閥值
    min_t = df[df.columns[i]].mean()-(3*df[df.columns[i]].std())
    # 最大閥值
    max_t = df[df.columns[i]].mean()+(3*df[df.columns[i]].std())

    # 評估每列的數值是否存在異常值
    for j in range(df.shape[0]):
        data=df.iloc[j,i]
        if data < min_t or data > max_t:
            # 設為NaN
            df.iloc[j,i]=np.NaN
print(df)
```

```
     Python   Java    PHP
0      90.0    NaN   33.0
1      50.0   70.0    NaN
2      70.0   90.0   75.0
3       NaN   50.0   85.0
...
```

Handling missing values

STEP/**01** 匯入套件。

```
import numpy as np
import pandas as pd
```

STEP/**02** 建立有缺失值的數據集。

```
scores={
    'Python': [90, 50, 70, np.NaN, 80, 60, np.NaN, 55, 76, 88],
    'Java': [np.NaN, 70, 90, 50, 40, np.NaN, 77, 66, np.NaN, 89],
    'PHP': [33, np.NaN, 75, 85, 82, 90, 95, 56, 68, np.NaN]
}
df=pd.DataFrame(scores)
print(df.shape)

Out:
(10, 3)
```

STEP/**03** 計算數據集每行的缺失值數量。

```
print(df.isnull().sum())


Out:
Python      2
Java        3
PHP         2
dtype: int64
```

STEP/07 使用 Pandas 的 fillna() 方法，由後面的數據來填補缺失值。

```
df4 = df.fillna(method = 'bfill' )
print(df4)
```

```
DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)
```

```
# Fill NaN values with a specific value
df.fillna(0)

# Fill NaN values with the mean of each column
df.fillna(df.mean())

# Forward fill NaN values along the rows
df.fillna(method='ffill', axis=0)

# Backward fill NaN values along the columns
df.fillna(method='bfill', axis=1)
```

STEP/08 我們也可以用平均值來填補缺失值。首先計算數據集每行的平均值：

```
python_avg=df.Python.mean().round().astype(int)
java_avg=df.Java.mean().round().astype(int)
php_avg=df.PHP.mean().round().astype(int)
print(python_avg, java_avg, php_avg)

Out:
71 69 73
```

STEP/09 以平均值取代。

```
df.PHP.fillna(value=php_avg, inplace=True)
df.Java.fillna(value=php_avg, inplace=True)
df.Python.fillna(value=php_avg, inplace=True)
print(df)
```

Normalization

Scale data to a specified range or standardize it to a specific distribution.

- **Min-Max Normalization**

$$x_{norm} = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

```
df_orig=df
df=(df-df.min())/(df.max()-df.min())
print(df)
```

- **Z-score Normalization:** a distribution with a mean of 0 and a standard deviation of 1

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

```
# 取出原本的DataFrame
df=df_orig
df=(df-df.mean())/(df.std())
print(df)
```

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler

# Min-Max Normalization
min_max_scaler = MinMaxScaler()
normalized_data = min_max_scaler.fit_transform(data)

# Z-score Normalization
standard_scaler = StandardScaler()
normalized_data = standard_scaler.fit_transform(data)
```

Splitting the dataset

- Split a dataset into <u>training, validation and testing</u> subsets

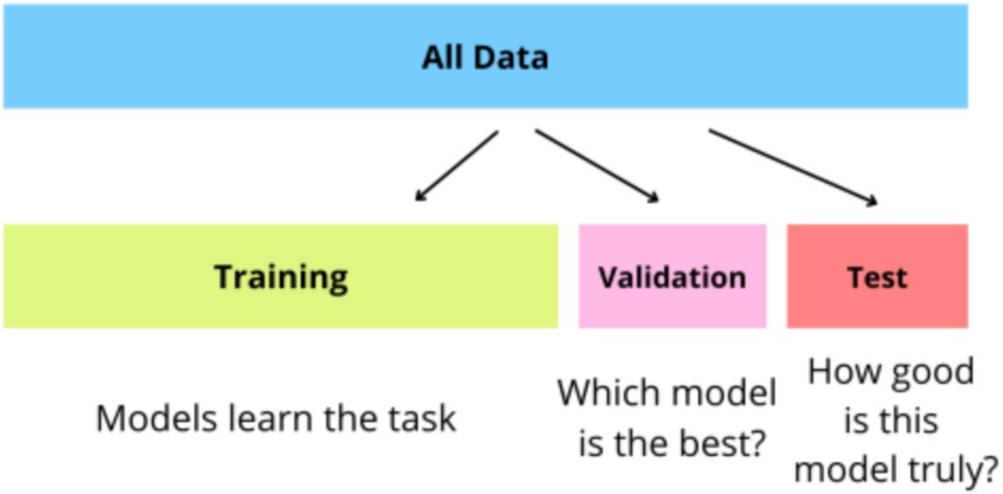


Figure from:
https://medium.com/@rahulchavan4894/understanding-train-test-and-validation-dataset-split-in-simple-quick-terms-5a8630fe58c8

Training : Validation : Testing ➔ 60:20:20 (Method 1)

STEP/01 取出原本的 DataFrame，打亂數據集的順序。

```
df=df_orig
x_shuffle=df.sample(frac=1, random_state=0)
print(x_shuffle)
```

```
# Randomly sample 5 rows from the DataFrame
df.sample(n=5)

# Randomly sample 10% of the rows from the DataFrame
df.sample(frac=0.1)

# Randomly sample 3 rows from the DataFrame with replacement
df.sample(n=3, replace=True)
```

STEP/02 定義訓練集、驗證集的索引結束值。

```
train_end = int(len(x_shuffle) * 0.6)   # 訓練集的索引結束值
dev_end = int(len(x_shuffle) * 0.8)     # 驗證集的索引結束值
print(train_end, dev_end)


Out:
6  8
```

STEP/**03 拆分數據集。**

```
x_train = x_shuffle.iloc[:train_end, :]          # 訓練集
x_dev = x_shuffle.iloc[train_end:dev_end, :]     # 驗證集
x_test = x_shuffle.iloc[dev_end:, :]             # 測試集
print(x_train.shape, x_dev.shape, x_test.shape)

Out:
(6, 3) (2, 3) (2, 3)
```

Training : Validation : Testing → 60:20:20 (Method 2)

```python
from sklearn.model_selection import train_test_split

# Assume 'X' is your feature matrix and 'y' is your target variable

# First, split the dataset into temporary and test sets (80% temporary, 20% test)
X_temp, X_test, y_temp, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Then, split the temporary set into training and validation sets (75% training, 25% validation)
# Since we want a 60:20:20 ratio, we'll use a ratio of 75:25 for the temporary set
X_train, X_val, y_train, y_val = train_test_split(X_temp, y_temp, test_size=0.25, random_state=42)

# Now, you have three subsets: X_train, X_val, and X_test for features,
# and y_train, y_val, and y_test for labels.
```

Exercise: Predicting the release year of a song

- The YearPredictionMSD dataset allows us to predict the release year of songs based on audio features.
- The songs mostly consist of Western commercial tracks ranging from 1922 to 2011, with a focus on songs from around the year 2000.

Dataset Information

Additional Information

You should respect the following train / test split:

train: first 463,715 examples

test: last 51,630 examples

It avoids the 'producer effect' by making sure no song from a given artist ends up in both the train and test set.

https://archive.ics.uci.edu/dataset/203/yearpredictionmsd

Code from: PyTorch深度學習入門與應用：必備實作知識與工具一本就學會
ISBN：9786263332591

Exercise: Predicting the release year of a song

Increase accuracy of the prediction

Submission requirements:

1. source code (predict.py)
2. PDF documents
   Explaining your strategy.
   Show the outputs (before and after)
3. Upload to e-learning before 4/8 14:10