Feature Scaling

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
df = pd.read_csv("./data/Churn_Modelling.csv")
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           9946 non-null   object 
 6   Age              9700 non-null   float64
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(3), int64(8), object(3)
memory usage: 1.1+ MB
```

`Gender` has 54 missing values

`Age` has 300 missing values

In [ ]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
```

1. What is Normalization?

Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

2. What is Standardization?

Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.

In [ ]:
```
df.head()
```

Out[ ]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Ten
0	1	15634602	Hargrave	619	France	Female	42.0	
1	2	15647311	Hill	608	Spain	Female	41.0	
2	3	15619304	Onio	502	France	Female	42.0	
3	4	15701354	Boni	699	France	Female	39.0	
4	5	15737888	Mitchell	850	Spain	Female	43.0	

In [ ]:
```
df.describe().round(2)
```

Out[ ]:

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	Num
count	10000.00	10000.00	10000.00	9700.00	10000.00	10000.00	
mean	5000.50	15690940.57	650.53	38.92	5.01	76485.89	
std	2886.90	71936.19	96.65	10.49	2.89	62397.41	
min	1.00	15565701.00	350.00	18.00	0.00	0.00	
25%	2500.75	15628528.25	584.00	32.00	3.00	0.00	
50%	5000.50	15690738.00	652.00	37.00	5.00	97198.54	
75%	7500.25	15753233.75	718.00	44.00	7.00	127644.24	
max	10000.00	15815690.00	850.00	92.00	10.00	250898.09	

- `Task-1` NORMALIZATION

In [ ]:
```
df.head(5)
```

Out[ ]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Ten
0	1	15634602	Hargrave	619	France	Female	42.0	
1	2	15647311	Hill	608	Spain	Female	41.0	
2	3	15619304	Onio	502	France	Female	42.0	
3	4	15701354	Boni	699	France	Female	39.0	
4	5	15737888	Mitchell	850	Spain	Female	43.0	

In [ ]:

```
# New datafram only of Age and Tenure col
new_df = pd.DataFrame(df,columns = ['Age', 'Tenure'])
```

In [ ]:

```
new_df.head(5)
```

Out[ ]:

	Age	Tenure
0	42.0	2
1	41.0	1
2	42.0	8
3	39.0	1
4	43.0	2

In [ ]:

```
# replacing null values to mean
new_df['Age']=new_df['Age'].fillna(new_df['Age'].mean())
```

In [ ]:

```
new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Age     10000 non-null  float64
 1   Tenure  10000 non-null  int64  
dtypes: float64(1), int64(1)
memory usage: 156.4 KB
```

In [ ]:

```
scaler = MinMaxScaler() # Instantiating the MinMaxScaler() function
normalized_df = scaler.fit_transform(new_df)
print(normalized_df)
```

```
[[0.32432432 0.2       ]
 [0.31081081 0.1       ]
 [0.32432432 0.8       ]
 ...
 [0.24324324 0.7       ]
 [0.32432432 0.3       ]
 [0.13513514 0.4       ]]
```

- `Task-2` STANDARDIZATION

In [ ]:
```
scaler = StandardScaler()
standardized_df = scaler.fit_transform(new_df)
print(standardized_df)
```

```
[[ 0.29783904 -1.04175968]
 [ 0.20100192 -1.38753759]
 [ 0.29783904  1.03290776]
 ...
 [-0.28318369  0.68712986]
 [ 0.29783904 -0.69598177]
 [-1.05788067 -0.35020386]]
```