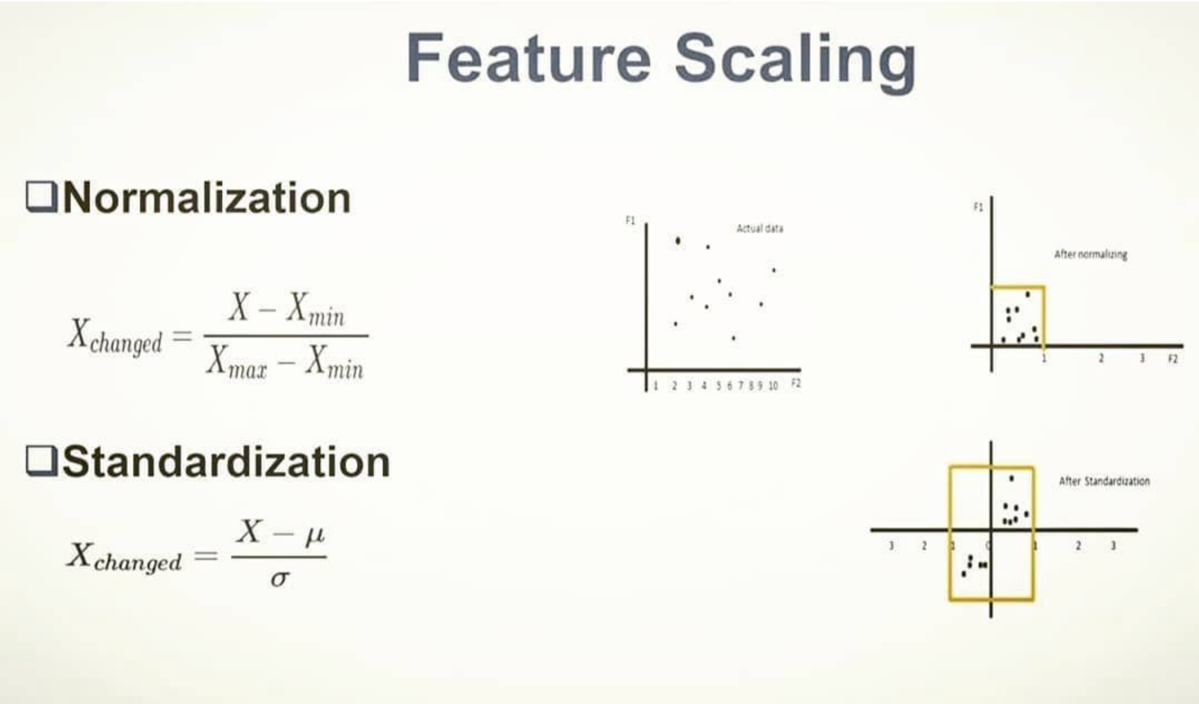
Scaling

Feature scaling in machine learning is one of the most critical steps during the pre-processing of data before creating a machine learning model. Scaling can make a difference between a weak machine learning model and a better one. Machine learning algorithms like linear regression, logistic regression, neural network, etc. that use gradient descent as an optimization technique require data to be scaled. Distance algorithms like KNN, K-means, and SVM are most affected by the range of features. This is because behind the scenes they are using distances between data points to determine their similarity. Tree-based algorithms, on the other hand, are fairly insensitive to the scale of the features.

The most common techniques of feature scaling are Normalization and Standardization. Normalization is used when we want to bound our values between two numbers, typically, between [0, 1] or [-1, 1]. While Standardization transforms the data to have zero mean and a variance of 1, they make our data unitless.



Imports

In [1]:
```python
import numpy as np
import pandas as pd
```

Read CSV File

In [2]:
```python
df = pd.read_csv('Salaries_Encoded.csv', index_col=0)
```

In [3]:
```python
df.head()
```

Out[3]:

	BasePay	OvertimePay	OtherPay	Benefits	Year_2012	Year_2013	Year_2014

Id	BasePay	OvertimePay	OtherPay	Benefits	Year_2012	Year_2013	Year_2014
Id							
2	155966.02	245131.88	137811.38	0.0	0	0	0
3	212739.13	106088.18	16452.60	0.0	0	0	0
4	77916.00	56120.71	198306.90	0.0	0	0	0
5	134401.60	9737.00	182234.59	0.0	0	0	0
6	118602.00	8601.00	189082.74	0.0	0	0	0

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 101716 entries, 2 to 148621
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   BasePay      101716 non-null  float64
 1   OvertimePay  101716 non-null  float64
 2   OtherPay     101716 non-null  float64
 3   Benefits     101716 non-null  float64
 4   Year_2012    101716 non-null  int64  
 5   Year_2013    101716 non-null  int64  
 6   Year_2014    101716 non-null  int64  
dtypes: float64(4), int64(3)
memory usage: 6.2 MB
```

In [5]:

```
df.describe()
```

Out[5]:

	BasePay	OvertimePay	OtherPay	Benefits	Year_2012	Year_2013	Yea
count	101716.000000	101716.000000	101716.000000	101716.000000	101716.000000	101716.000000	101716
mean	66039.414888	4367.986505	3269.793674	17784.721382	0.231960	0.239530	0
std	42944.744270	10652.098794	7692.445042	17294.099478	0.422085	0.426799	0
min	6.040000	-0.010000	-7058.590000	0.000000	0.000000	0.000000	0
25%	32171.617500	0.000000	0.000000	0.000000	0.000000	0.000000	0
50%	64436.695000	0.000000	624.000000	18236.795000	0.000000	0.000000	0
75%	94864.125000	3393.670000	3486.000000	32730.600000	0.000000	0.000000	0
max	319275.010000	245131.880000	342802.630000	96570.660000	1.000000	1.000000	1

Feature Selection

In [6]:

```
X = df.drop('BasePay', axis=1)
y = df['BasePay']
```

Train Test Split

In [7]:

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

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101
```

Standardization

Standardization is a 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.

$$x_{scaled} = \frac{x - mean}{sd}$$

Standardization can be helpful in cases where the data follows a Gaussian distribution. However, this does not have to be necessarily true. Also, unlike normalization, standardization does not have a bounding range. So, even if you have outliers in your data, they will not be affected by standardization.

In [8]:
```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

In [9]:
```
standard = StandardScaler()
```

In [11]:
```
X_train.head()
```

Out[11]:

	OvertimePay	OtherPay	Benefits	Year_2012	Year_2013	Year_2014
Id						
58135	0.0	0.00	28818.13	1	0	0
108345	0.0	40.04	0.00	0	1	0
29909	0.0	259.00	0.00	0	0	0
65485	0.0	0.00	2450.59	1	0	0
35252	0.0	0.00	0.00	0	0	0

In [14]:
```
X_train_scaled = standard.fit_transform(X_train)
```

In [15]:
```
X_test_scaled = standard.transform(X_test)
```

In [19]:
```
print(X_train_scaled[:5])
```

```
[[-0.41058129 -0.42008418  0.64179953  1.82487558 -0.56023702 -0.56715949]
 [-0.41058129 -0.41489784 -1.02543986 -0.54798256  1.78495881 -0.56715949]
 [-0.41058129 -0.38653614 -1.02543986 -0.54798256 -0.56023702 -0.56715949]
 [-0.41058129 -0.42008418 -0.88366383  1.82487558 -0.56023702 -0.56715949]
 [-0.41058129 -0.42008418 -1.02543986 -0.54798256 -0.56023702 -0.56715949]]
```

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. This scaler is sensitive to outliers.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Normalization is good to use when you know that the distribution of your data does not follow a Gaussian distribution. This can be useful in algorithms that do not assume any distribution of the data like K-Nearest Neighbors and Neural Networks.

In [20]:
```
# Min Max Scaler
```

In [21]:
```
min_max = MinMaxScaler()
```

In [22]:
```
X_train_scaled = min_max.fit_transform(X_train)
```

In [23]:
```
X_test_scaled = min_max.transform(X_test)
```

In [24]:
```
X_train.head()
```

Out[24]:

Id	OvertimePay	OtherPay	Benefits	Year_2012	Year_2013	Year_2014
58135	0.0	0.00	28818.13	1	0	0
108345	0.0	40.04	0.00	0	1	0
29909	0.0	259.00	0.00	0	0	0
65485	0.0	0.00	2450.59	1	0	0
35252	0.0	0.00	0.00	0	0	0

In [29]:
```
print(X_train_scaled[:5])
```

```
[[5.76210327e-08 2.01753998e-02 3.15982516e-01 1.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [5.76210327e-08 2.02898452e-02 0.00000000e+00 0.00000000e+00
  1.00000000e+00 0.00000000e+00]
 [5.76210327e-08 2.09156934e-02 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [5.76210327e-08 2.01753998e-02 2.68700153e-02 1.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [5.76210327e-08 2.01753998e-02 0.00000000e+00 0.00000000e+00
  0.00000000e+00 0.00000000e+00]]
```

Now, X_train_scaled, X_test_scaled, y_train and y_test can be used in Machine Learning models.