

Purpose: Feature engineering involves manipulating, selecting, or creating features (input variables) from raw data to improve model performance.

- Transformation: It includes transforming data through extraction, scaling, and other techniques to help models capture patterns and relationships.
- Selection: Choosing relevant features improves model simplicity, interpretability, and generalization by reducing noise and redundancy.
- Creation: New features can be engineered to capture important relationships or patterns not apparent in the original data.
- Missing Data: Handling missing values through techniques like imputation maintains data integrity.
- Categorical Variables: Encoding categorical variables as numerical values enables their use in machine learning algorithms.

- Interaction Features: Combining existing features to capture interactions between variables can enhance model performance.
- Domain Expertise: Feature engineering often requires domain knowledge to make informed decisions about creating meaningful features.
- Impact: Effective feature engineering can significantly enhance model accuracy and robustness.
- Experimentation: Iterative experimentation with different feature engineering techniques is key to finding optimal features for a given problem and dataset.



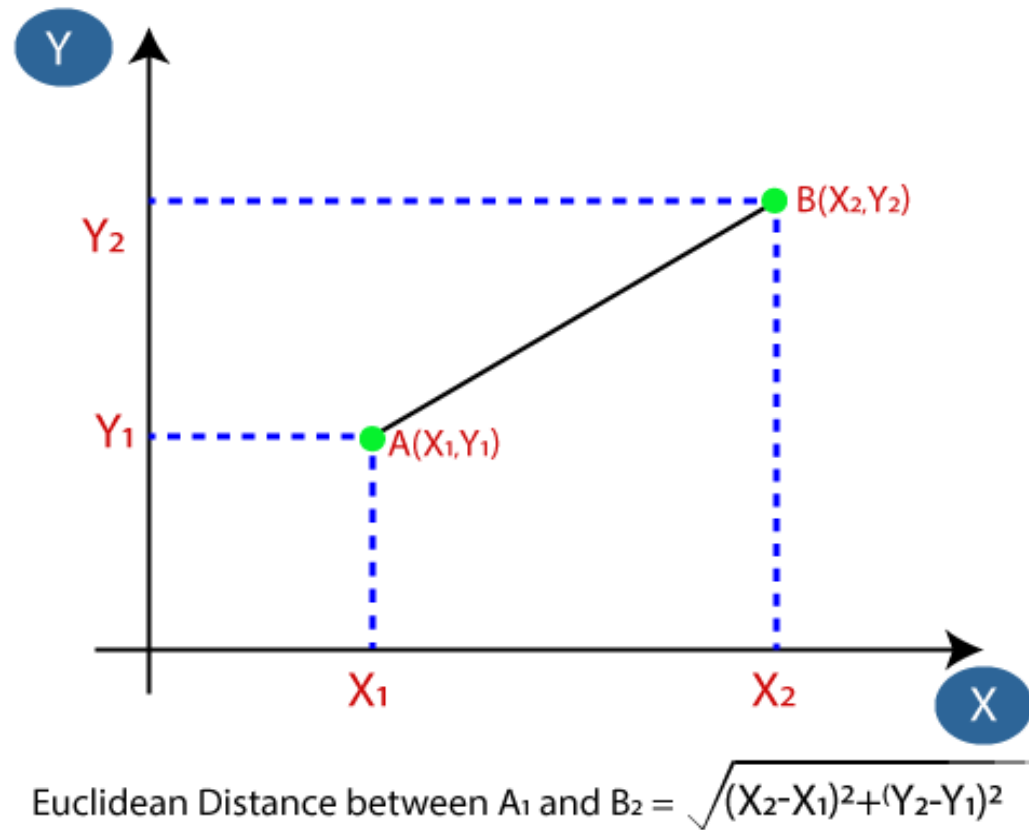
Before

	Marketing Spend	Administration	Transport
8	120542.52	148718.95	311613.29
3	144372.41	118671.85	383199.62
6	134615.46	147198.87	127716.82
41	27892.92	84710.77	164470.71
46	1315.46	115816.21	297114.46
47	0.00	135426.92	0.00
15	165349.20	122616.84	261776.23
9	123334.88	108679.17	304981.62
16	78013.11	121597.55	264346.06
24	77044.01	99281.34	140574.81
34	46426.07	157693.92	210797.67
31	61136.38	152701.92	88218.23
0	114523.61	136897.80	471784.10

After

```
[ 0.51045637, 0.65435014, 0.39465254,
[ 0.7717808 , -0.07058751, 0.85129231,
[ 0.66478369, 0.61767561, -0.77839882,
[-0.50556192, -0.88995663, -0.54395059,
[-0.79701687, -0.13948471, 0.30216642,
[-0.81144253, 0.33365719, -1.59308759,
[ 1.00181744, 0.02459211, 0.0767485 ,
[ 0.54107808, -0.311678 , 0.35234999,
[ 0.04406841, 0. , 0.09314111,
[ 0.03344102, -0.53841672, -0.69637939,
[-0.30232284, 0.87088664, -0.24843702,
[-0.14100596, 0.7504461 , -1.03035512,
[ 0.44445152, 0.36914469, 1.41636104,
[-0.56823598, 0.80121401, -1.41234414,
[-0.02069287, 0.15120209, 0.65982542,
[-0.20288224, -0.44731064, -0.22396041,
[ 0.63475197, -0.52554826, 0.72155723,
[-0.09199671, 0.75841129, -0.90966598,
[ 0.29255093, -0.71914339, 0. ,
[-0.49710869, 0.1316995 , -0.3101261 ,
[ 0.97164381, 0.71849438, 1.23848267,
[ 0.04819566, 0.77629811, 0.3188971 .
```

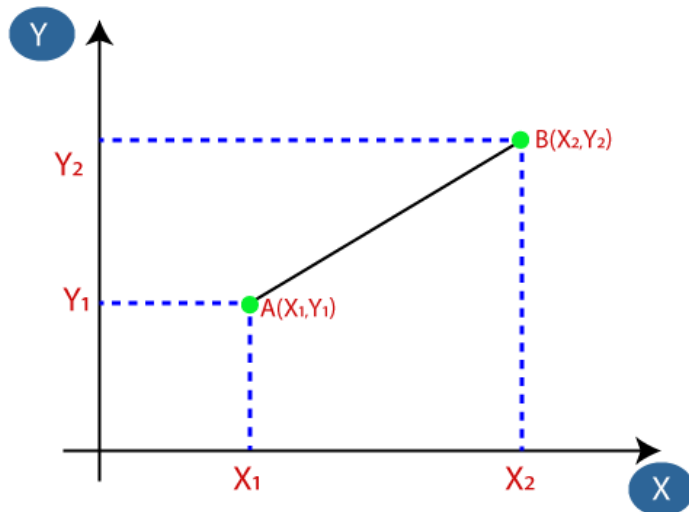
The **Euclidean Distance** between two points is calculated using a simple formula.



The **Manhattan Distance** between two points is calculated using a simple formula.

$$\text{Manhattan Distance} = |x_1 - x_2| + |y_1 - y_2|$$

$$\text{Manhattan Distance} = d(x, y) = \sum_{i=1}^n |x_i - y_i|$$



$$\text{Euclidean Distance between } A_1 \text{ and } B_2 = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

	Student	CGPA	Salary '000
0	1	3.0	60
1	2	3.0	40
2	3	4.0	40
3	4	4.5	50
4	5	4.2	52

- Distance AB before scaling $\Rightarrow \sqrt{(40 - 60)^2 + (3 - 3)^2} = 20$
- Distance BC before scaling $\Rightarrow \sqrt{(40 - 40)^2 + (4 - 3)^2} = 1$

	Student	CGPA	Salary '000
0	1	-1.184341	1.520013
1	2	-1.184341	-1.100699
2	3	0.416120	-1.100699
3	4	1.216350	0.209657
4	5	0.736212	0.471728

- Distance AB after scaling $\Rightarrow \sqrt{(1.1 + 1.5)^2 + (1.18 - 1.18)^2} = 2.6$
- Distance BC after scaling $\Rightarrow \sqrt{(1.1 - 1.1)^2 + (0.41 + 1.18)^2} = 1.59$

Few advantages of feature scaling the data are as follows:

1. It makes your training faster.
2. It prevents you from getting stuck in local optima.
3. It gives you a better error surface shape.

However, there are few algorithms such as Tree based algorithms and probability-based algo. that are not affected by the scaling of input data.

Examples of Algorithms where Feature Scaling matters:

1. **K-Means** uses the Euclidean distance measure here feature scaling matters.
2. **K-Nearest - Neighbours** also require feature scaling.
3. **Principal Component Analysis (PCA)**: Tries to get the feature with maximum variance, here too feature scaling is required.
4. **Gradient Descent**: Calculation speed increase as Theta calculation becomes faster after feature scaling.

Note: Naive Bayes, Decision Tree, Random Forest & All tree-based models are not affected by feature scaling.

Techniques to perform Feature Transformation:

- Normalization
- Standardization
- Log Transformation
- Robust Scaler
- Max Absolute Scaler

	Student	CGPA	Salary '000
0	1	3.0	60
1	2	3.0	40
2	3	4.0	40
3	4	4.5	50
4	5	4.2	52

Normalization:

$$X_{\text{new}} = \frac{X_i - \min(X)}{\max(x) - \min(X)}$$

Python Implementation:

```
from sklearn.preprocessing import MinMaxScaler  
scaler = MinMaxScaler()
```

	Student	CGPA	Salary '000
0	1	3.0	60
1	2	3.0	40
2	3	4.0	40
3	4	4.5	50
4	5	4.2	52

Standardization:
$$X_{\text{new}} = \frac{X_i - X_{\text{mean}}}{\text{Standard Deviation}}$$

Standard Deviation:
$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$$

Python Implementation:

```
from sklearn.preprocessing import StandardScaler  
scaler = StandardScaler()
```

σ = population standard deviation

N = the size of the population

x_i = each value from the population

μ = the population mean

The log transform can be applied as follows:

1. Check if the feature has any zero or negative values. If so, consider using a modified version of the log transform (e.g., adding a constant value or using the logarithm of the absolute values).
2. Add a small constant value (e.g., 5) to the feature before applying the logarithm. This is done to avoid taking the logarithm of zero or close-to-zero values, which would result in undefined or infinite values.
3. Apply the natural logarithm function (base e) to each value of the feature.

<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.FunctionTransformer.html>

In simplest terms, the **Max Absolute Scaler** takes the absolute **maximum** value of each column and **divides** each value in the column by the maximum value.

Formula:
$$x_{scaled} = \frac{x}{\max(x)}$$

Python Implementation:

```
from sklearn.preprocessing import MaxAbsScaler  
scaler = MaxAbsScaler()
```

Robust Scaler are robust to outliers. It is used to scale the feature to median and quantiles. Scaling using median and quantiles consists of subtracting the median to all the observations, and then dividing by the interquartile difference. The interquartile difference is the difference between the 75th and 25th quantile:

- IQR = 75th quantile - 25th quantile
- RobustScaler = $(X_i - X_{\text{Median}}) / \text{IQR}$

$$X_{\text{scale}} = \frac{x_i - x_{\text{med}}}{x_{75} - x_{25}}$$

Python Implementation:

```
from sklearn.preprocessing import RobustScaler  
RoSc=RobustScaler()
```

Video: <https://youtu.be/U9N-ELpCpc8>

Let's do it with PYTHON