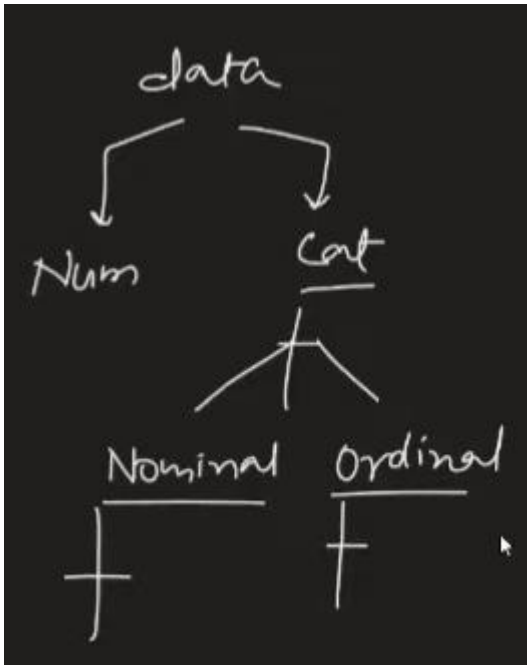
Machine Learning

Video 26:

Encoding Categorical Data | Ordinal Encoding | Label Encoding:

We know Feature engineering includes feature transformation, while feature scaling and categorical encoding are subparts of it.

We also know that data can be divided as:



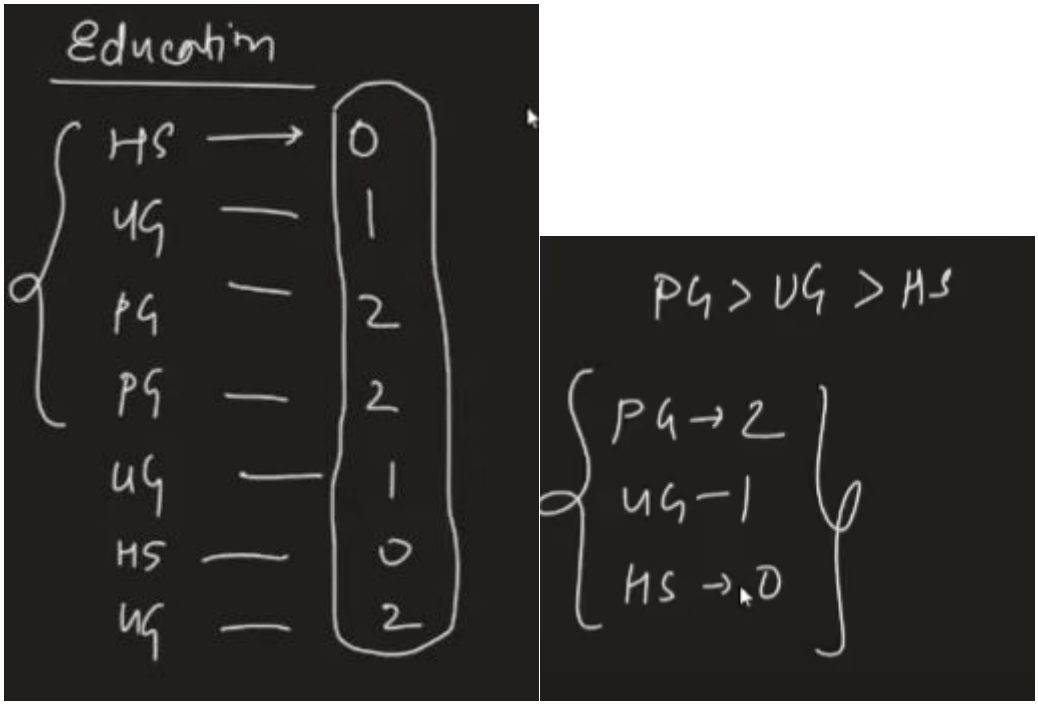
How to encode categorical data?
We will convert the strings into the number. The main two ways to do so are:
1. Ordinal encoding (on ordinal data)

2. One hot encoding (on Nominal data)

So, what is label encoding?

Label Encoding:
Label encoding is a technique in machine learning to convert categorical data into numerical values. Each unique category is assigned a distinct integer value. This method is useful for algorithms that require numerical input, but can introduce ordinal relationships where none exist, potentially leading to biases.

How it works:
Label encoding works by mapping each category in a feature to a unique integer. For example, "red" could be 0, "blue" 1, and "green" 2. This transformed data can then be fed into machine learning models. However, it assumes an inherent order among categories, which might not always be accurate.



Example:

Code link:

https://colab.research.google.com/drive/1KvFOG1GG4rbUkXH9uWeWBonMR9Neg5rh?usp=sharing

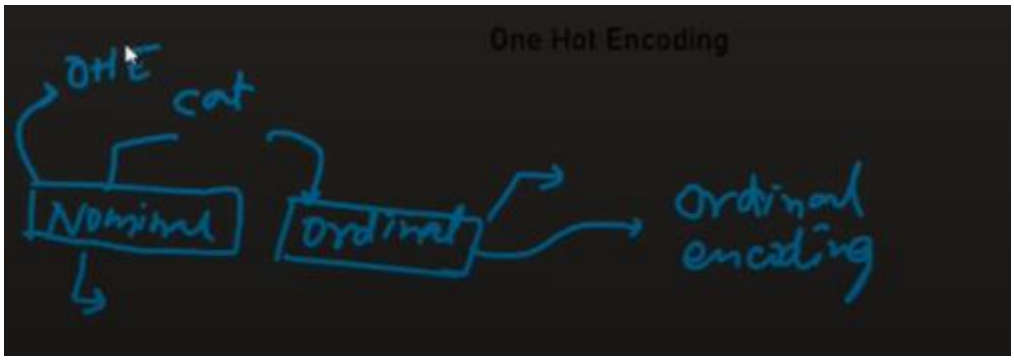
Data link:

https://github.com/campusx-official/100-days-of-machine-learning/tree/main/day26-ordinal-encoding

Video 27:

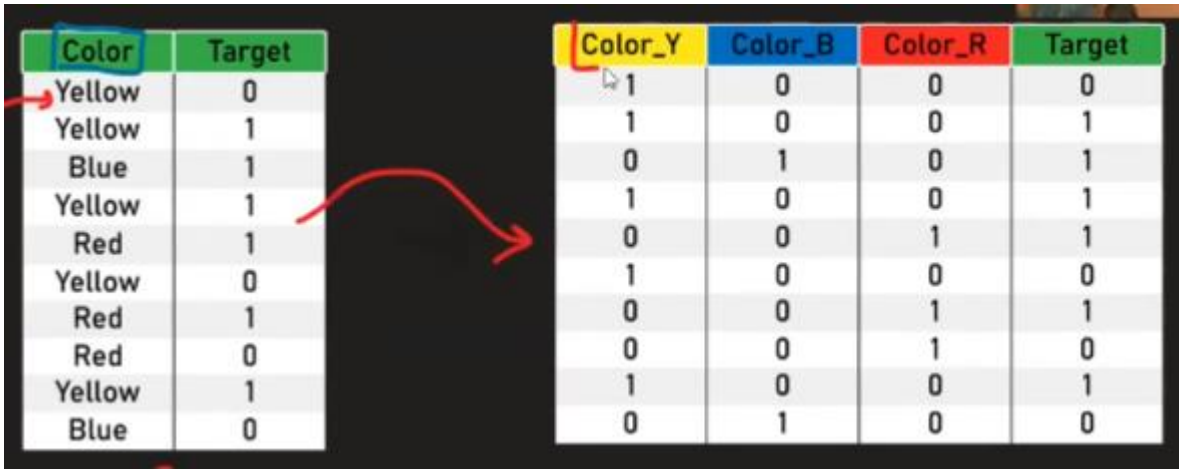
One Hot Encoding | Handling Categorical Data:

We know,



One-Hot Encoding:
One-hot encoding is a method to convert categorical variables into a format suitable for machine learning models. Each category is represented as a binary vector, with a "1" indicating the presence of that category and "0" indicating the absence.

How it works:
One-hot encoding creates a new binary column for each category in the original feature. For example, a "Color" feature with values ["red", "blue", "green"] will be transformed into three separate columns: "red", "blue", "green", with 1s and 0s indicating the presence of each color in the original data. This avoids implying any ordinal relationship.



Color	Target
Yellow	0
Yellow	1
Blue	1
Yellow	1
Red	1
Yellow	0
Red	1
Red	0
Yellow	1
Blue	0

Color_Y	Color_B	Color_R	Target
1	0	0	0
1	0	0	1
0	1	0	1
1	0	0	1
0	0	1	1
1	0	0	0
0	0	1	1
0	0	1	0
1	0	0	1
0	1	0	0

Dummy Variable Trap:
The dummy variable trap occurs when there is perfect multicollinearity between the dummy variables created during one-hot encoding. It happens when one dummy variable can be predicted by others, leading to redundancy and negatively impacting model performance by causing issues in regression models.

Multicollinearity occurs when two or more independent variables in a regression model are highly correlated with each other. This makes it difficult to determine the individual effect of each variable on the dependent variable, leading to unreliable coefficient estimates and reduced model interpretability.

How it works:
In one-hot encoding, if all dummy variables are used (e.g., for three categories, "red", "blue", and "green"), the model may overfit or produce unstable coefficients due to collinearity. To avoid the trap, one dummy variable is typically dropped to prevent redundancy and maintain the model's validity.

Thus, if we have n columns in input then we will use n-1 columns in one hot encoding.

One-Hot Encoding using Most Frequent:
One-hot encoding using the most frequent category is a variation where instead of creating binary columns for every unique category, the most frequent category in the feature is used as a reference. The rest of the categories are encoded relative to this most frequent category, often simplifying the model.

How it works:
In this method, all categories are initially one-hot encoded, but if a category appears less frequently, it is merged into the most frequent category. This can help reduce dimensionality and avoid the "dummy variable trap." It's particularly useful when there are too many rare categories that may not significantly impact the model's predictions.

Example:

Code link:

https://colab.research.google.com/drive/1KvFOG1GG4rbUkXH9uWeWBonMR9Neg5rh?usp=sharing

Data link:

https://github.com/campusx-official/100-days-of-machine-learning/tree/main/day27-one-hot-encoding

Video 28:

Column Transformer in Machine Learning | How to use Column Transformer in Sklearn:

A **ColumnTransformer** in machine learning applies different preprocessing steps to specific subsets of columns in a dataset. It allows transformation of numerical and categorical features separately, such as scaling numerical data and encoding categorical data, making it highly efficient for handling diverse feature types in machine learning models.

Code link:

https://colab.research.google.com/drive/1TOJ3xnDB6L1VqxxmAztOGg3EGZ8Rlw98?usp=sharing\

Data link:

https://github.com/campusx-official/100-days-of-machine-learning/tree/main/day28-column-transformer

Video 29:

Machine Learning Pipelines A-Z:

Pipelines chains together multiple steps so that the output of each step is used as input to the next step. Pipelines makes it easy to apply the same preprocessing to train and test!

Code link:

https://colab.research.google.com/drive/17USf87yBoVqbBc7U3zGURPb-mfXdaAcM?usp=sharing

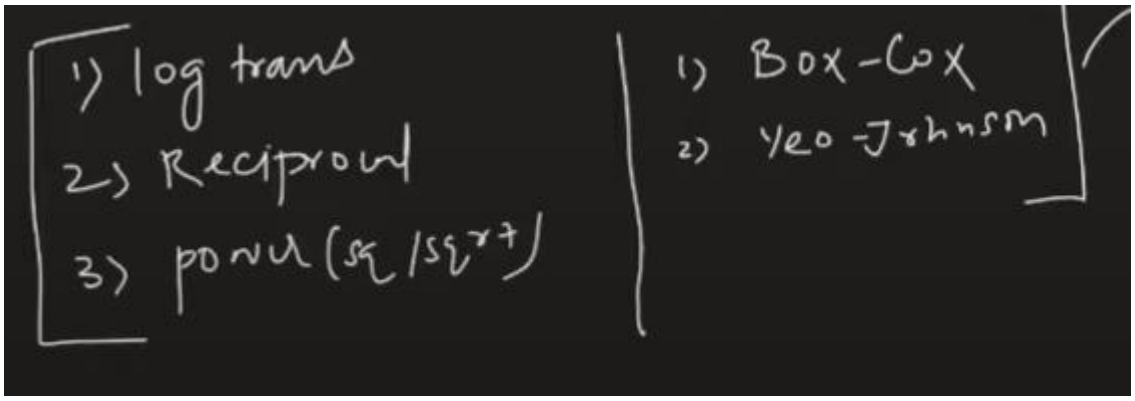
Data link:

https://github.com/campusx-official/100-days-of-machine-learning/tree/main/day29-sklearn-pipelines

Video 30:

Function Transformer | Log Transform | Reciprocal Transform | Square Root Transform:

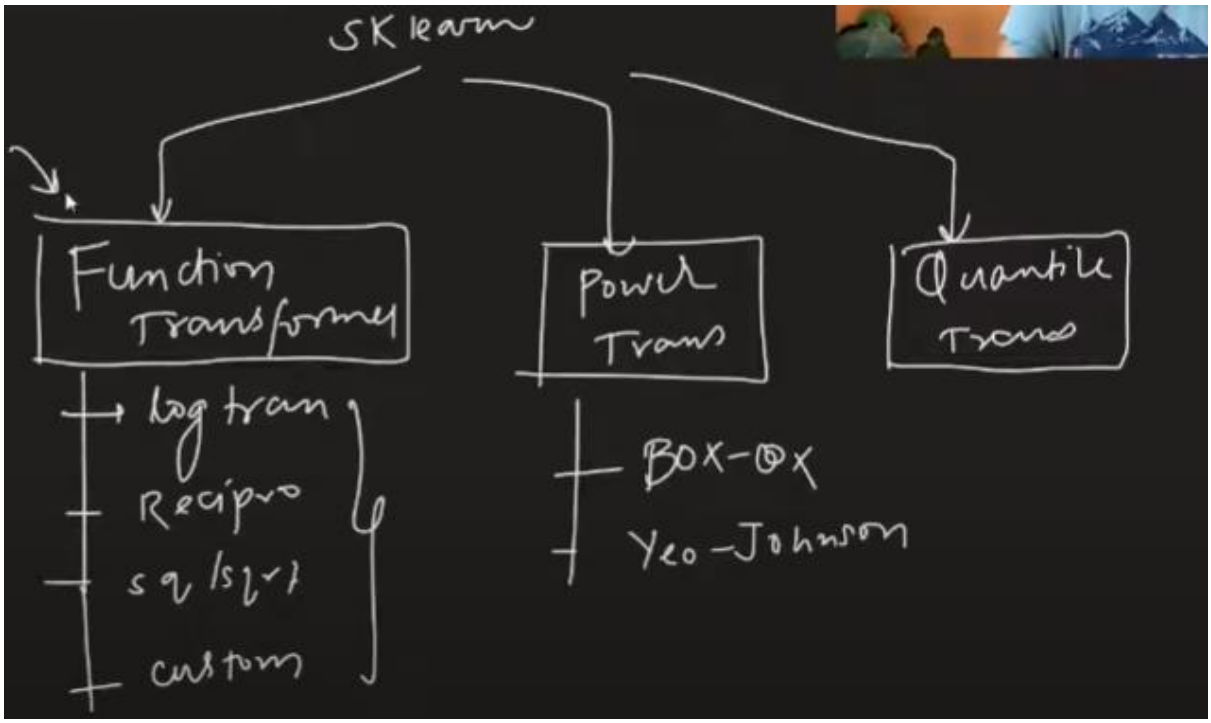
We will learn the following types of transforms:



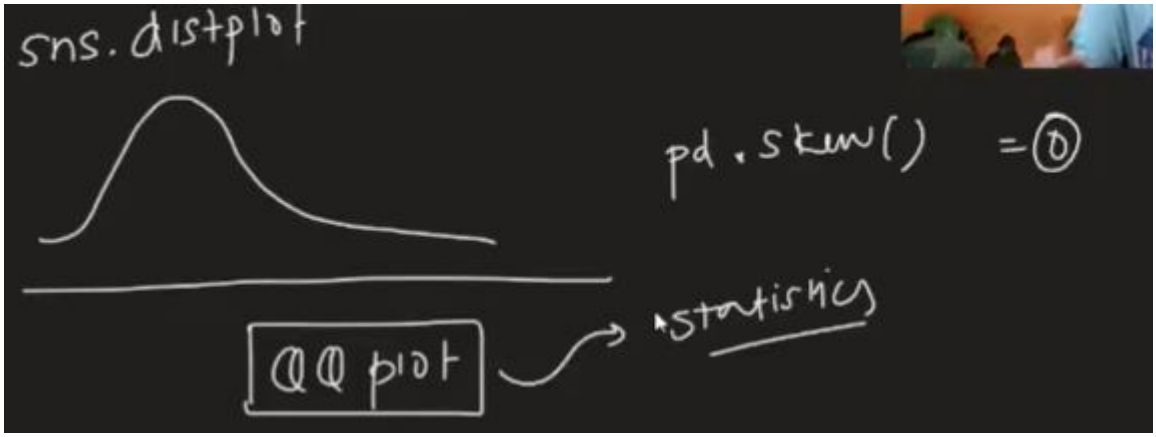
What it do?

It converts the data into normal distribution. Because, normal distribution makes the flow easy.

Here, we will learn the following:



How to find if data is normal?



What is log transform?

We may observe the normal distribution on the data such as right skewed data.

What is reciprocal transform?

Reciprocating the data.

Square transform? For left skewed data

Example:

Code link:

https://colab.research.google.com/drive/1NWRwQ4LAdTgsSsoY0LR3ffIDAYlhV_hs?usp=sharing

Data link:

https://github.com/campusx-official/100-days-of-machine-learning/tree/main/day30-function-transformer