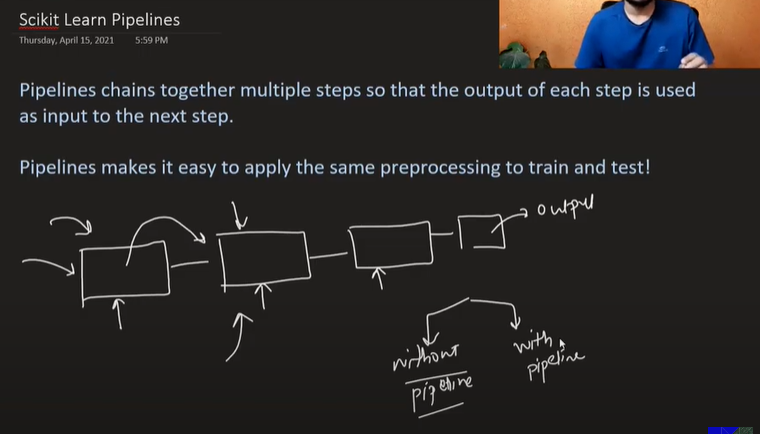
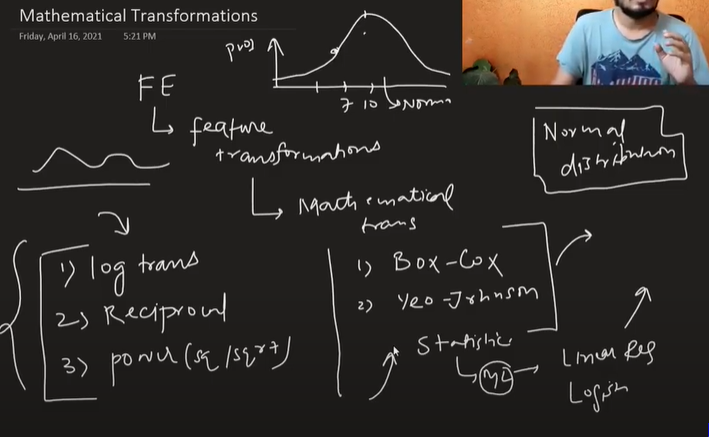
**Machine Learning Pipelines A-Z | Day 29 | 100 Days of Machine Learning**

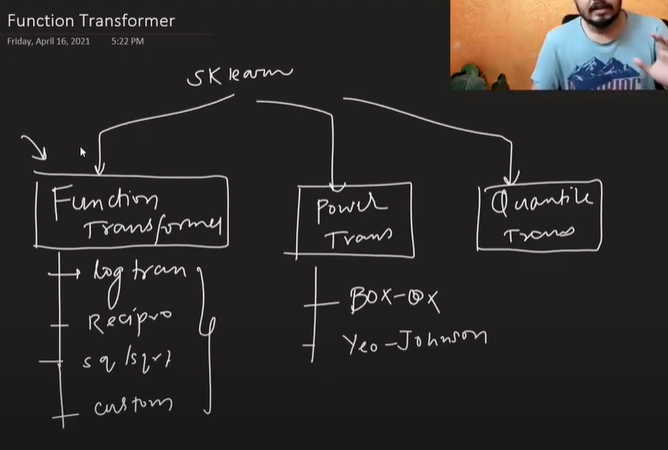


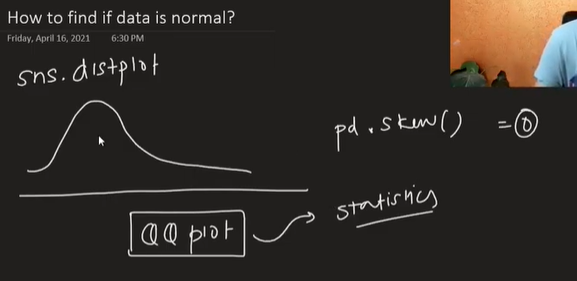
In Scikit Learn Pipelines we can holistically build a pipeline of set of steps that we regularly perform individually such as imputing null values, encoding and training the model

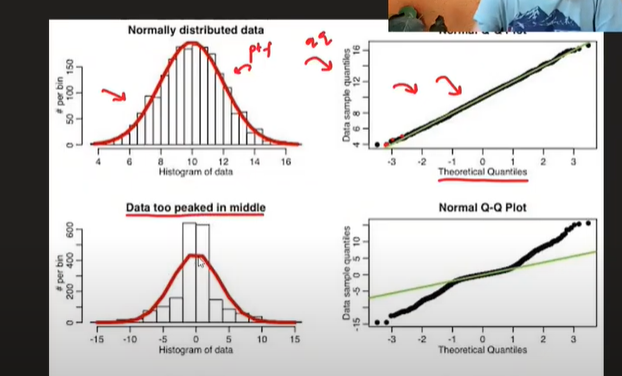
# Function Transformer | Log Transform | Reciprocal Transform | Square Root Transform

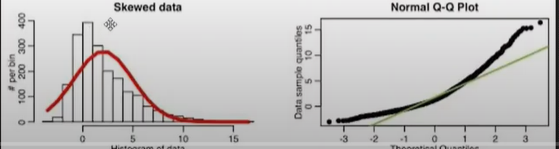


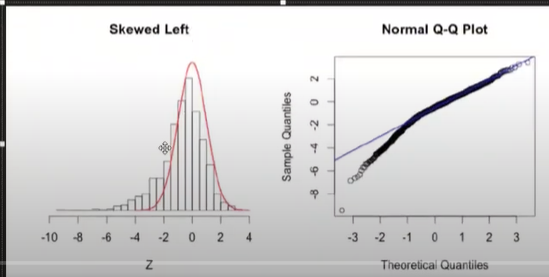
These Feature transformations are used to transform data to normal distribution. Statistical models such as Linear Regression and Logistic Regression demands the normal distribution and hence these feature transformation converts any kind of distribution into normal distribution

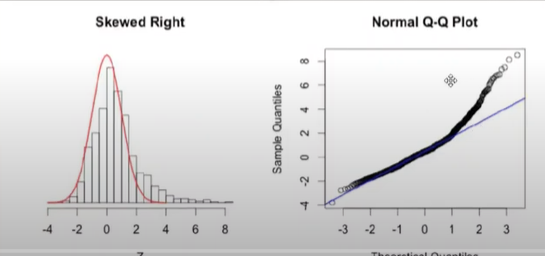


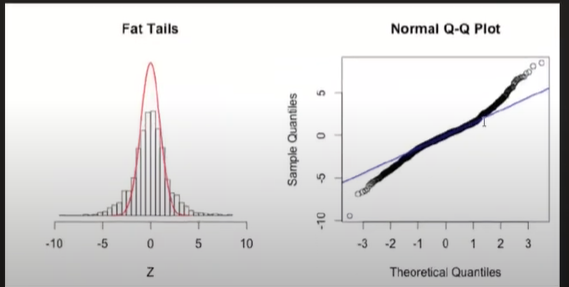


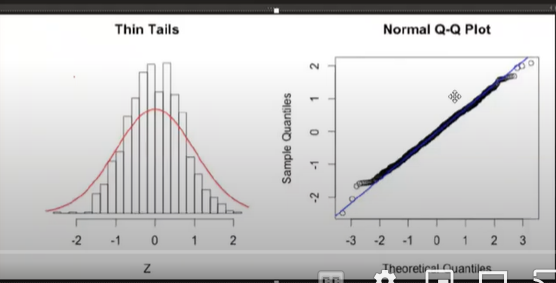








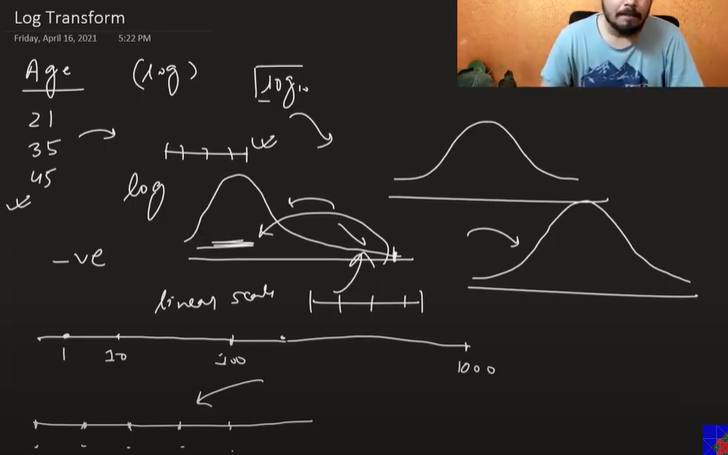


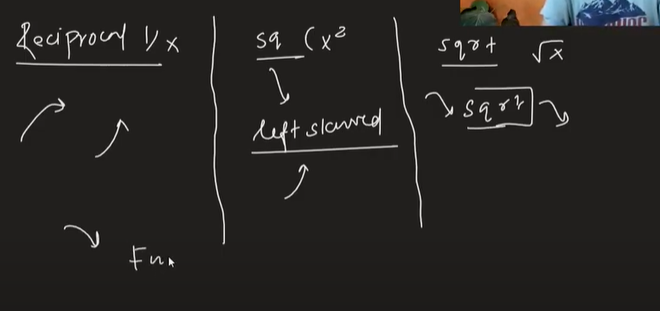


**Log Transformer**

It converts the bigger scales into smaller linear scales in equidistance

It cannot be applied on the negative values but it is useful in the right skewed data





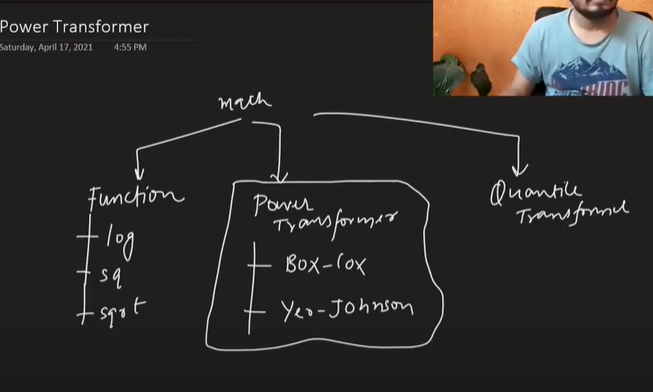
Square transform is used in case of left skewed data

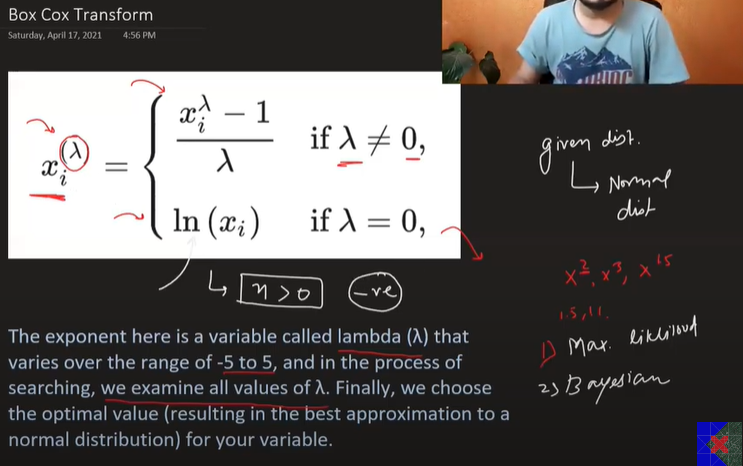
Difference b/w np.log and np.log1p is that 1p adds 1 and then applies log, while the dataset has some zeros we use 1p else we use np.log

Applying log transformation to normally distributed data leads to less accuracy while it doesn’t change results that much if we apply it to tree based models

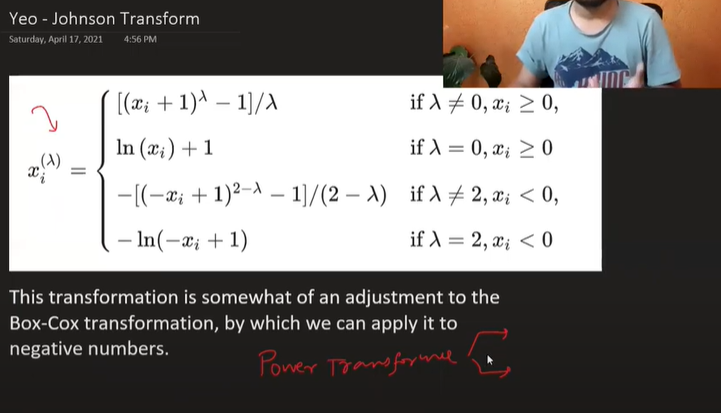
It gives better results in Linear/ Logistic Regression models

# Power Transformer | Box - Cox Transform | Yeo - Johnson Transform





Box cox transformation doesn’t work on zero and negative values

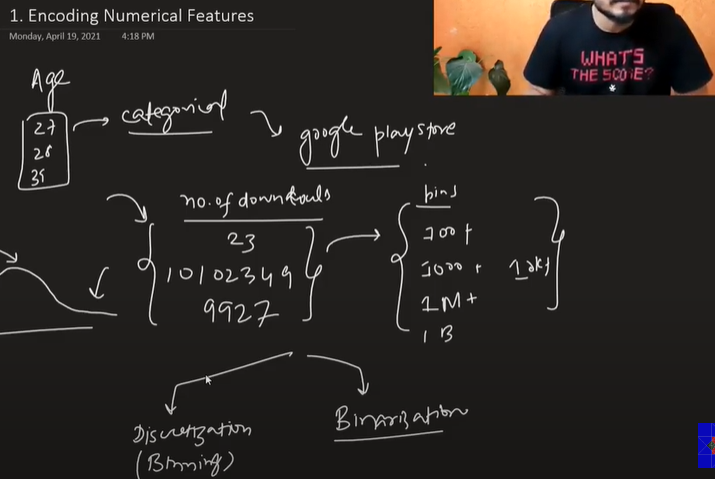


Lambda value is calculated using maximum likelihood or Bayesian stats

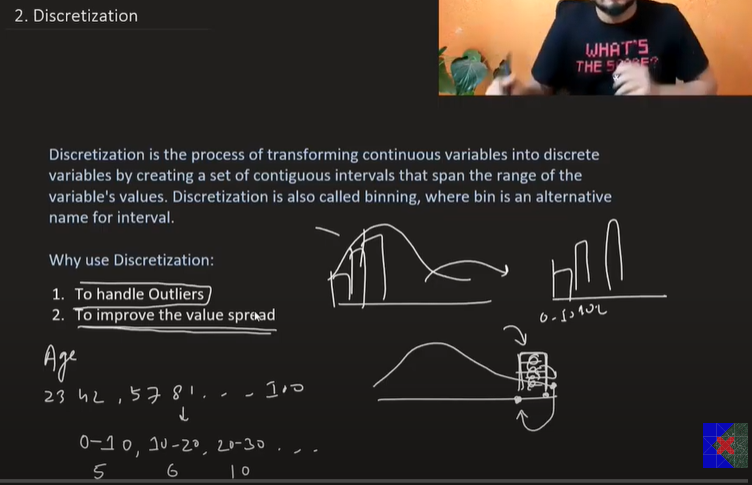
We don’t need to apply standardization separately while applying PowerTransformer()

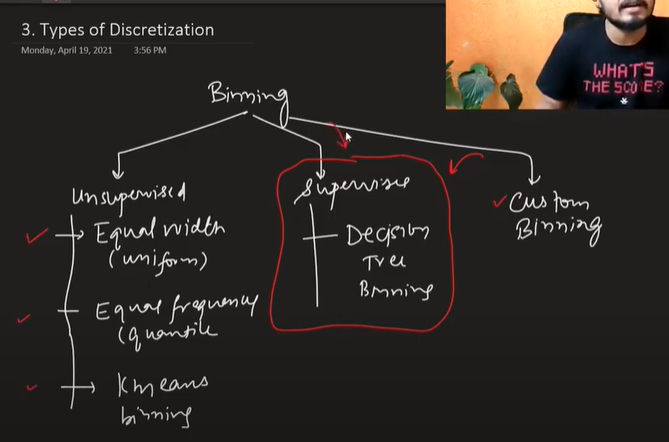
When we are using box-cox transformation, if the values are zero it won’t work. So we add 0.00001 to avoid these scenarios

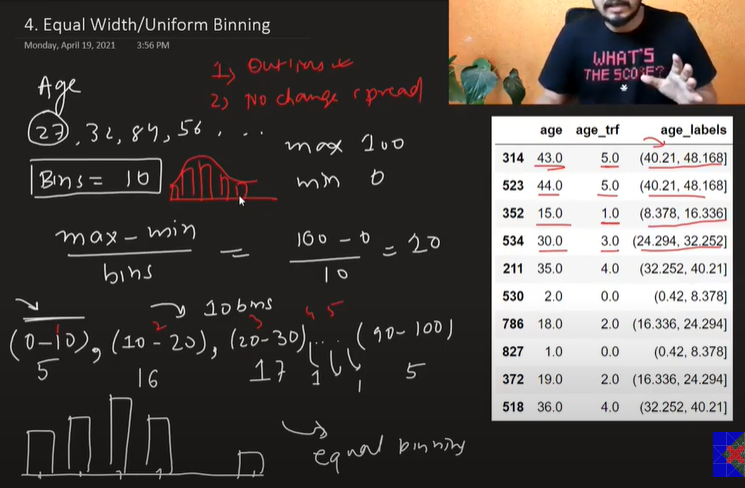
# Binning and Binarization | Discretization | Quantile Binning | KMeans Binning



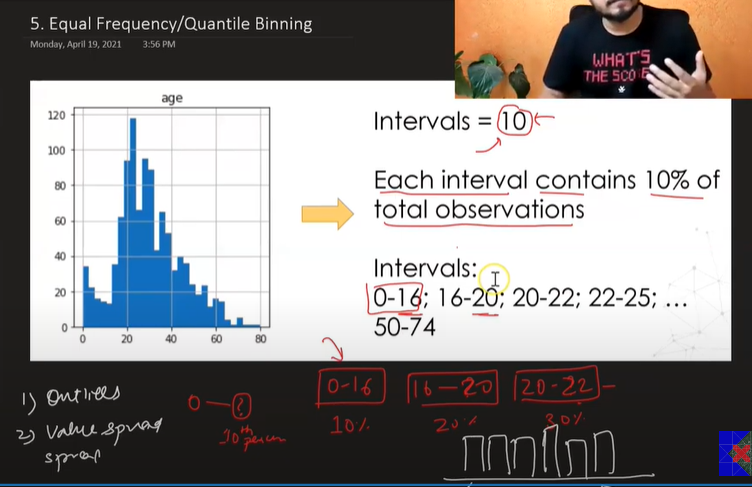
Now in some scenarios encoding the numerical values can do the trick





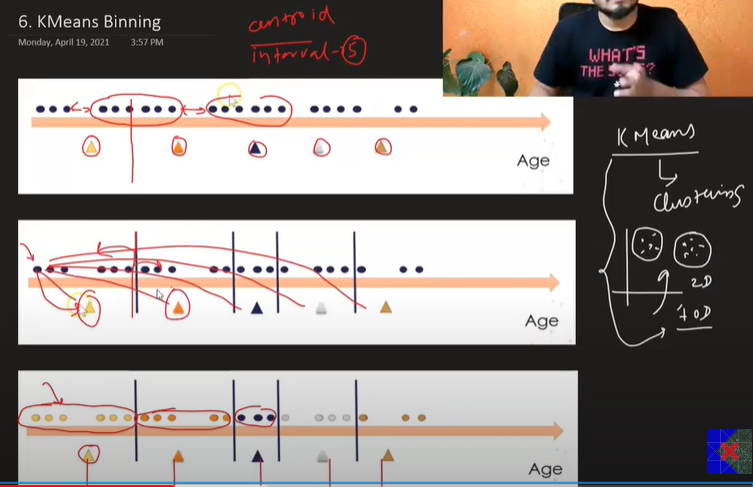


Equal width/ Uniform binning helps to handle outliers and there is no change in the spread



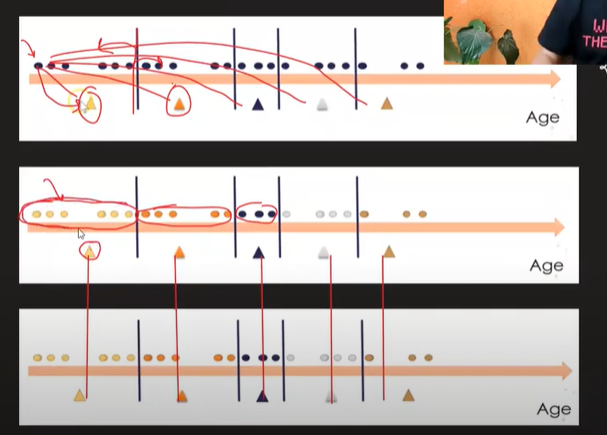
In this type of binning, the bin size may not be equal but each bins stores 10 percentile data. So 10 intervals will score 100 percentile data

This is also robust to the outliers and the makes the spread uniform



This type of binning is used in scenarios where the data is in clusters

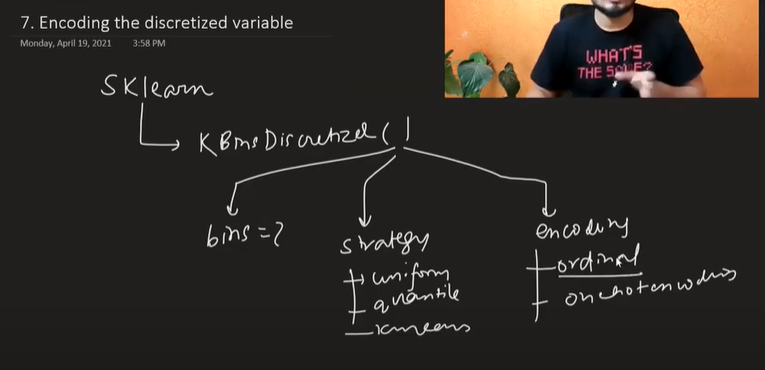
We define centroid and calculated distance of all the points from each centroid. The point which is closer to the respective centroid is considered and that point is mapped in same cluster

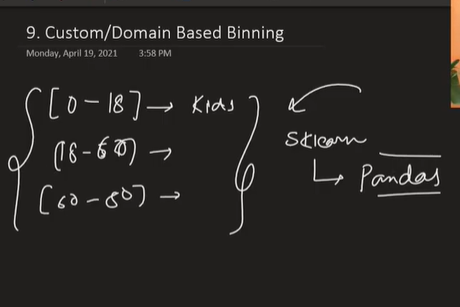


Now when all the clusters are formed, we take the mean of all the points in the clusters. After calculating the mean, we shift the centroid to the position of mean of respective cluster

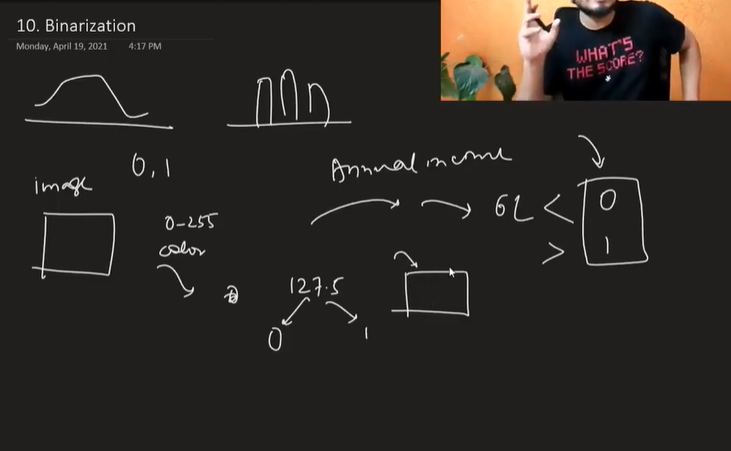
Now, we perform the same steps of clustering and shifting the centroids until when the difference b/w the original position and the new position is the same

The final position of centroid is the bins and intervals



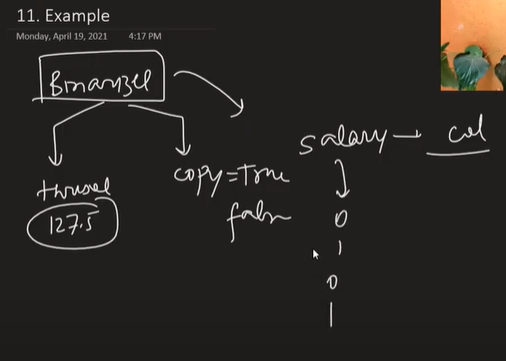


Custom/Domain Binning is performed manually based on domain expertise. We can use pandas pd.cut for this



In Binarization, we split the continuous values into either 0 or 1

It is mostly used in image processing



# Handling Mixed Variables | Feature Engineering

Handling a column that contains mixed datatype values can be a challenge. If there are such scenarios, we split the column into two separate columns

