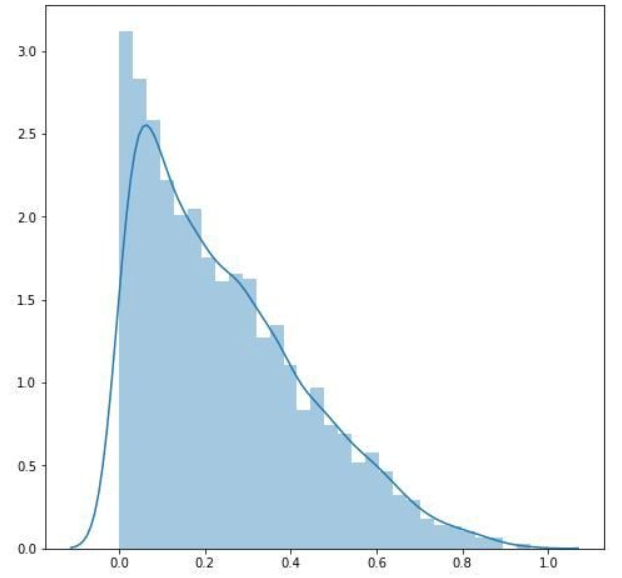
Yeo-Johnson and Box-Cox transformations are mathematical functions designed to transform non-normal dependent variables into a normal shape. These transformations are used to stabilize variance and make data more normally distributed, which can be beneficial for many statistical techniques and to improve accuracy of statical test or model.

The Box-Cox transformation is named after statisticians George Box and David Cox who developed the technique in the 1960s. It involves taking the natural logarithm of a variable and then raising it to some power (lambda) which is determined by maximum likelihood estimation. The lambda value will depend on how skewed the data is, meaning that a different lambda will be used for different data sets. This transformation can be used in regression, ANOVA, and many other applications where there is a need to transform non-normal data into normal form. Box-Cox transformation is a statistical technique that transforms your target variable so that your data closely resembles a [normal distribution](https://builtin.com/data-science/empirical-rule).

In many statistical techniques, we assume that the errors are normally distributed. This assumption allows us to construct confidence intervals and conduct hypothesis tests. By transforming your target variable, we can hopefully normalize our errors, if they are not already normal. Additionally, transforming our variables can improve the predictive power of our models because transformations can cut away white noise.

**What Is Box-Cox Transformation and Target Variable?**

Box-Cox transformation is a statistical technique that involves transforming your target variable so that your data follows a normal distribution. A target variable is the variable in your analytical model that you are trying to estimate. Box-Cox transformation helps to improve the predictive power of your analytical model because it cuts away white noise.

A blue graph with a point

Description automatically generated with medium confidence

Original data Transformed data

**Box-Cox Transformation**:

* Applicable only to positive data. The Box-Cox transformation often makes the distribution more symmetric and closer to normal.
* The transformation is parameterized by a λ value, which determines the form of the transformation.

**Yeo-Johnson Transformation**:

* An extension of the Box-Cox transformation that can handle both positive and negative data (more flexible).
* Like Box-Cox, it is also parameterized by a lambda (λ) value.

**Box-Cox Transformation equation:**

If w is our transformed variable and “y” is our target variable, then the Box-Cox transformation equation looks like this:

A close up of a math problem

Description automatically generated

λ: Transformation parameter (determined during the transformation process to maximize the likelihood function).  “*t”*is the time period and lambda is the parameter that we choose. You can also perform the Box-Cox transformation on non-time series data.

Notice what happens when lambda equals one. In that case, our data shifts down, but the shape of the data does not change. If the optimal value for lambda is one, then the data is already normally distributed, and the Box-Cox transformation is unnecessary.

**How Do You Choose Lambda?**

We choose the value of lambda that provides the best approximation for the normal distribution of our response variable.

[SciPy](https://scipy.org/) has a boxcox function that will choose the optimal value of lambda for us.

scipy.stats.boxcox(): Simply pass a 1-D array into the function and it will return the Box-Cox transformed array and the optimal value for lambda.

**Limitations:** If interpretation is your goal, then the Box-Cox transformation may be a poor choice. If lambda is a non-zero number, then the transformed target variable may be more difficult to interpret than if we simply applied a log transform.

A second issue is that the Box-Cox transformation usually gives the median of the forecast distribution when we revert the transformed data to its original scale. Occasionally, we want the mean, not the median, and there are other ways to do that.

**Effect of transformation on Range**

* **Box-Cox Transformation**:
  + The transformation changes the distribution and scale of the data depending on the value of λ.
  + For some values of λ, the transformed data can be compressed or expanded, leading to a new range.
  + For example, with λ=0 (log transformation), the range of the data might be reduced because the logarithm reduces the magnitude of large values more than small ones.

**Comparison of Transformation to Normalization and Scaling**

* **Normalization (Min-Max Scaling)**:
  + Normalization typically scales the data to a fixed range, such as [0, 1] or [-1, 1], without changing the shape of the data distribution.
  + It directly adjusts the range of the data based on its minimum and maximum values.
* **Standardization (Z-score Scaling)**:
  + Standardization adjusts the data to have a mean of 0 and a standard deviation of 1. This can change the range, but the primary focus is on adjusting the distribution around the mean.
* **Box-Cox and Yeo-Johnson**:
  + These transformations focus on stabilizing variance and making data more normally distributed, not on fitting the data into a specific range.
  + The range change is a side effect of the transformation process and depends on the specific λ.

**Practical Impact**

* **Range Expansion/Compression**: Depending on λ, the transformed data might have a wider or narrower range than the original data. For example:
  + If λ<1, the transformation might compress the data, reducing the range.
  + If λ>1, the transformation might expand the data, increasing the range.
* **No Fixed Range**: Unlike normalization, there’s no fixed range (like [0, 1]) that the data will fit into after the transformation.

**Normalization vs. Transformation**: Unlike normalization, which specifically adjusts the range, transformations like Box-Cox and Yeo-Johnson aim to make data more normally distributed, with range change as a byproduct.

If you need the data in a specific range, you might still need to normalize or scale the data after applying the transformation, depending on the requirements of your analysis.

**Data normalization**

To normalize data in Python using scikit-learn, also known as sklearn is used. **When you normalize data, you change the scale of the data. Data is commonly rescaled to fall between 0 and 1, because machine learning algorithms tend to perform better, or converge faster, when the different features are on a smaller scale.** Before training machine learning models on data, it’s common practice to normalize the data first to potentially get better, faster results. Normalization also makes the training process less sensitive to the scale of the features, resulting in better coefficients after training. This process of making features more suitable for training by rescaling is called feature scaling.

Using the scikit-learn preprocessing.normalize() Function to Normalize Data

You can use the scikit-learn preprocessing.normalize() function to normalize an array-like dataset.

The normalize() function scales vectors individually to a unit norm so that the vector has a length of one. The default norm for normalize() is L2, also known as the Euclidean norm. The L2 norm formula is the square root of the sum of the squares of each value. Although using the normalize() function results in values between 0 and 1, it’s not the same as simply scaling the values to fall between 0 and 1. The output shows that all the values are in the range 0 to 1. If you square each value in the output and then add them together, the result is 1, or very close to 1.

You can normalize a one dimensional NumPy array using the normalize() function.

**Normalizing Columns from a DataFrame Using the normalize() Function:**

In a pandas DataFrame, features are columns and rows are samples. You can convert a DataFrame column into a NumPy array and then normalize the data in the array.

The output shows that the normalize() function converts all the values between 0 and 1 so that the square root of the sum of the squares of the values equals one. In other words, the values were scaled to a unit length using the L2 norm.

**Normalizing Datasets by Row or by Column Using the normalize() Function**

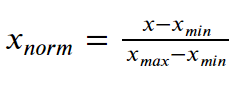
When you normalize a dataset without converting features, or columns, into arrays for processing, the data is normalized by row. The default axis for the normalize() function is 1, which means that each sample, or row, is normalized.

**Using the scikit-learn preprocessing.MinMaxScaler() Function to Normalize Data:**

You can use the scikit-learn preprocessing.MinMaxScaler() function to normalize each feature by scaling the data to a range.

The MinMaxScaler() function scales each feature individually so that the values have a given minimum and maximum value, with a default of 0 and 1.

The formula to scale feature values to between 0 and 1 is:



In this formula for feature scaling Subtract the minimum value from each entry and then divide the result by the range, where range is the difference between the maximum value and the minimum value.

The values are scaled to have the default minimum value of 0 and maximum value of 1. You can also specify different minimum and maximum values for scaling using [feature\_range=(0, 2)], which means the output will be scaled with a minimum value of 0 and maximum value of 1.

See code: **https://www.digitalocean.com/community/tutorials/normalize-data-in-python**

**Stratified sampling and cluster sampling** are both probability sampling techniques used in research and data collection. However, they serve different purposes and are used under different conditions.

**1. Stratified Sampling**

**Purpose:** Stratified sampling aims to ensure that specific subgroups (strata) within a population are adequately represented in the sample. This technique is particularly useful when the population has distinct subgroups that may differ significantly in terms of the characteristic being studied.

**How It Works:**

1. **Identify Strata:** The population is divided into non-overlapping subgroups (strata) based on a specific characteristic, such as age, gender, income level, etc.
2. **Sample Selection:** A random sample is taken from each stratum. The sample size from each stratum can be proportional to the stratum’s size in the population or equal across strata.
3. **Combine Samples:** The samples from all strata are combined to form the overall sample.

**Example:** If you’re conducting a survey on job satisfaction in a company with departments like Sales, Engineering, and HR, you might divide the employees into these strata and then randomly sample within each department. This ensures that each department is represented in the final sample.

**Advantages:**

* **Representation:** Ensures that all subgroups are represented, reducing the risk of bias.
* **Precision:** Increases the precision of the estimates by reducing variability within strata.

**Disadvantages:**

* **Complexity:** Requires detailed information about the population to define strata.
* **Cost:** Can be more costly and time-consuming due to the need to sample separately within each stratum.

**2. Cluster Sampling**

**Purpose:** Cluster sampling is used when it’s impractical or costly to perform simple random sampling across an entire population. It’s particularly useful when the population is large and geographically dispersed.

**How It Works:**

1. **Identify Clusters:** The population is divided into clusters, which are often geographically based or naturally occurring groups like schools, neighborhoods, etc.
2. **Sample Selection:** Instead of sampling individuals, entire clusters are randomly selected. All individuals within the selected clusters may be surveyed (one-stage cluster sampling), or a random sample of individuals within each selected cluster may be taken (two-stage cluster sampling).
3. **Data Collection:** Data is collected from the selected clusters.

**Example:** If you’re studying educational outcomes in a large country, you might divide the country into regions (clusters) and then randomly select a few regions to study. Within these regions, you might survey all schools (one-stage) or randomly select a few schools to survey (two-stage).

**Advantages:**

* **Cost-Effective:** Reduces costs and logistical challenges by focusing on fewer locations or groups.
* **Feasibility:** Makes data collection more feasible when dealing with large populations.

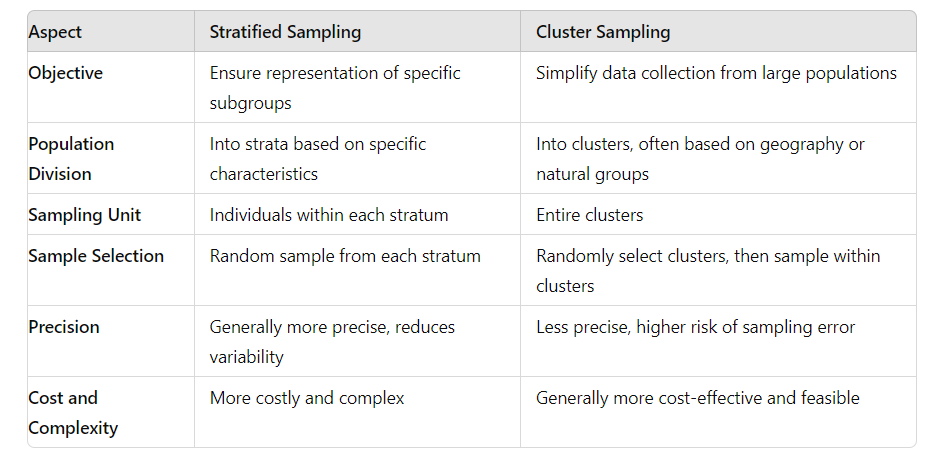
**Disadvantages:**

* **Higher Variability:** Increases the risk of sampling error because individuals within clusters may be more similar to each other than to those in other clusters.
* **Less Precision:** Estimates may be less precise compared to simple random sampling, especially if clusters are not representative of the population.

**When to Use Which?**

* **Stratified Sampling:** Use when you need to ensure representation from specific subgroups in the population, especially if those subgroups are important to your analysis.
* **Cluster Sampling:** Use when the population is large and spread out, making it impractical to sample from the entire population directly. It’s also useful when the cost and logistics of data collection need to be minimized.

**Key difference:**



In summary, stratified sampling focuses on ensuring representation across subgroups, while cluster sampling is about efficiency and practicality when dealing with large, dispersed populations.

**Binning**

Binning or converting numerical variables to categorical variables is a common data preprocessing technique used in various data analysis, machine learning, and statistical contexts. The reasons for doing this depend on the specific goals of the analysis or model. Here are some key reasons why binning or converting numerical variables to categorical variables might be necessary or beneficial:

**1. Handling Non-Linear Relationships**

In some cases, the relationship between a numerical feature and the target variable may not be linear. By binning the numerical variable, you can capture the non-linear relationship more effectively. For example, income might have a non-linear relationship with a likelihood of default on a loan, so binning income into categories (e.g., low, medium, high) might provide better predictive power in a model.

**2. Reducing Overfitting**

Binning can help reduce overfitting in machine learning models, especially when dealing with small datasets. By converting a continuous variable into a few discrete bins, you simplify the model, which can help it generalize better to new data.

**3. Improving Model Interpretability**

Categorical variables are often easier to interpret, For example, it's easier to understand the impact of different age groups (e.g., "18-25", "26-35") on a target variable (type of employment) than to interpret the effect of the exact age.

**4. Handling Outliers**

Binning can mitigate the influence of outliers in numerical data. For instance, extreme values can distort the results of a model, but if these values are binned into broader categories, their impact can be reduced.

**5. Simplifying Analysis**

In some analyses, especially in exploratory data analysis (EDA), binning can simplify the process of understanding and visualizing the data. For example, histograms or bar charts of binned data can be easier to interpret than complex scatter plots.

**6. Creating New Features**

Binning can be used to create new features that capture important patterns in the data. For example, if you have a continuous variable representing temperature, you might bin it into categories like "cold," "moderate," and "hot," which might align better with certain behaviors or outcomes.

**7. Dealing with Data Sparsity**

In datasets with a lot of sparse data (e.g., very few instances of certain values), binning can help reduce sparsity by grouping similar values together. This is particularly useful in cases like survival analysis or rare event prediction.

**8. Meeting Model Requirements**

Some machine learning models, like decision trees, can naturally handle both numerical and categorical variables, but others, like logistic regression, may require or perform better with categorical variables, especially if the relationship between the feature and the target variable is not linear.

**9. Improving Stability of Statistical Models**

In statistical models like regression, binning can improve the stability of the model by reducing the sensitivity to small changes in the data. For example, if you're modeling the effect of income on purchasing behavior, small fluctuations in income might not be meaningful, but broader income categories might provide more stable results.

**10. Alignment with Domain Knowledge**

In some cases, binning aligns better with domain knowledge or industry standards. For example, in healthcare, age might be categorized into standard ranges like "infant," "child," "adult," and "senior," which correspond to meaningful life stages.

**Example in Practice**

If you have a dataset with customer ages as a continuous variable, you might bin ages into categories like "18-25", "26-35", "36-45", etc., because:

* Different age groups may have different purchasing behaviors.
* It might be easier to interpret the effect of age on the target variable.
* You might want to reduce the impact of small variations in age on the model.