Project Report

FEATURE-ENGINEERING-AND-FEATURE-SELECTION

Submitted by

GOPAL SINGH

PG DIPLOMA IN DATA SCIENCE AND TECHNOLOGY,
CCSD,
JC BOSE UNIVERSITY OF SCIENCE AND TECHNOLOGY,
FARIDABAD.

Mentor

MR. RANJAN PRASAD

CCSD,
JC BOSE UNIVERSITY OF SCIENCE AND TECHNOLOGY,
FARIDABAD.

A major project report is submitted in CCSD of JC BOSE University Of Science And Technology in partial fulfilment of the requirements for Post Graduation Diploma in Data Science And Analytics.

Date:

Place:

Approved by

Professor Name

FEATURE-ENGINEERING-AND-FEATURE-SELECTION

By Gopal Singh

Accepted in partial fulfillment of the requirements for the Degree of Post Graduation Diploma in Data Science and Analytics

_____	_____
Professor name	External Examiner
Date:	Date:

Contents

1. Abstract

Building scalable machine learning as a service, or MLaaS, is critical to enterprise success. Key to translate machine learning project success into program success is to solve the evolving convoluted data engineering challenge, using local and global data. Enabling sharing of data features across a multitude of models within and across various line of business is pivotal to program success.

A feature is basically any input into an ML model. It is a set of variables that are incorporated into an ML model with the intention to improve model performance and accuracy. Features are derived values extracted from files and tables (a database) — and more importantly, computed from one or more tables. These are usually grouped together to minimize operational overhead and optimize storage. A feature, for example, can be any column with a calculated, flagged or one hot encoded value

A feature store is a central repository for storing documented, curated, and access-controlled features. It is a central place to store features that are properties of data, be it in the form of statistical derivations, piece of text, image pixel coordinates, aggregated value of purchase history, etc. This enables feature management to be uniform, reliable, reusable and governed. A feature store shouldn't be perceived as a type of new data store. In fact it should be recognized as a store of feature recipes with occasional time dependencies. For example, a feature like "number of login attempts in the last hour" is used in fraud models, customer service models and retention models. Each model will be computing it at a different historical point — the fraud model perhaps during a logon attempt, the customer service call at the point someone calls a call center and the retention model a certain date for the model to be built. But when the feature goes into production, it needs to run every single time a customer calls the call center

A feature store enables reusability of features across the enterprise, as existing features are visible to all potential users (e.g., business analysts, business intelligence developers, data scientists, etc.) across the business domain. The feature store supports feature enrichment, ranking, discovery, lineage (both data to feature and feature to model) and lifecycle management.

Both development and model serving teams need a diverse feature set, which can be met easily through the store. This will enable both teams to discover, store and manage features, while also decommissioning features that are no longer needed.

2. Acknowledgements

This work would not have been possible without the support of many people. I wish to express my sincere gratitude to all those who gave me the possibility to complete this thesis.

First of all, I would like to thank my supervisor, Mr Ranjan Prasad YMCA UST,Faridabad. for helping me realize this project and for his guidance during this work and giving me the opportunity to work in one of well-equipped lab of the institution. His valuable ideas and suggestions were immensely helpful which make this thesis work possible.

I would also like to thank Mr. Rajinder Chitoria (Data Scientist) and Co-Founder of Froyo ,Faridabad for granting me opportunity for Training at Froyo technology by Antrix Academy at Faridabad.

And Most importantly, I want to thank my family for providing unlimited support and helping me realize my dreams.

3. INTRODUCTION

What is feature engineering?

Feature engineering refers to a process of selecting and transforming variables/features in your dataset when creating a predictive model using machine learning.

What is Feature Selection?

Feature selection is the process where you automatically or manually select the features that contribute the most to your prediction variable or output.

Everyone can just pass data from models using sklearn or do automl stuffs using pycaret or any other python library but the idea is what will make the winners different from rest of us so answer is very simple: feature engineering and selection phases of machine learning pipeline will.

The idea that there are different ways to represent attributes in a model, and that some of these representations are better than others, leads to the idea of feature engineering - the process of creating representations of data that increase the effectiveness of a model.

Note that model effectiveness is influenced by many things. Obviously, if the attribute has no relationship to the outcome then its representation is irrelevant. However, it is very important to realize that there are a multitude of types of models and that each has its own sensitivities and needs. For example:

- Some models cannot tolerate attributes that measure the same underlying quantity (i.e., multicollinearity or correlation between attributes).
- Many models cannot use samples with any missing values.
- Some models are severely compromised when irrelevant attributes are in the data.

Feature engineering and variable selection can help mitigate many of these issues. The goal of this notebook is to help practitioners build better models by focusing on the attributes. “Better” depends on the context of the problem but most likely involves the following factors: accuracy, simplicity, and robustness. To achieve these characteristics, or to make good trade-offs between them, it is critical to understand the interplay between attributes used in a model and the type of model. Accuracy and/or simplicity can sometimes be improved by representing data in ways that are more palatable to the model or by reducing the number of variables used.

4. Visualization Numeric Features

One of the first steps of the exploratory data process when the ultimate purpose is to predict the output, is to create visualizations that help get knowledge of the output and then to uncover relationships between the attributes and the output. Knowledge about the output can be gained by creating a histogram or box plot.

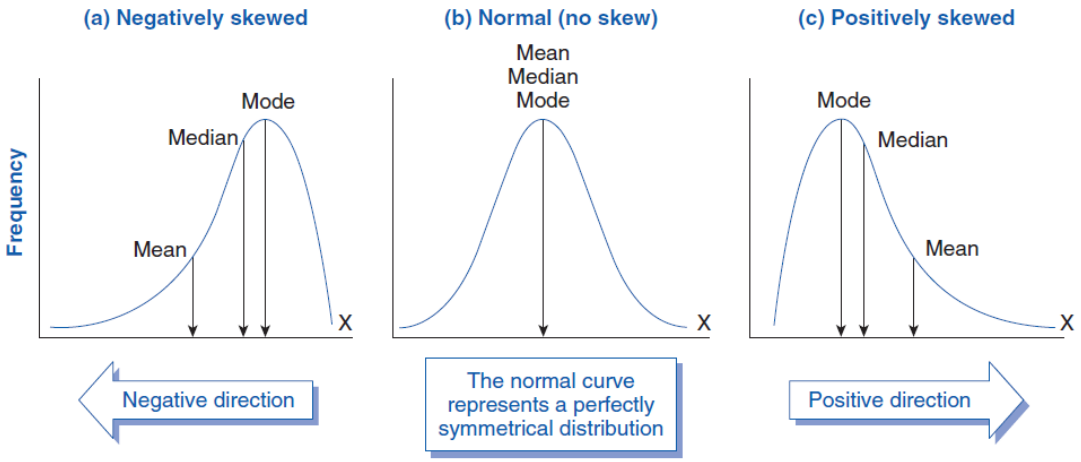
Important characteristics between attributes can be identified by examining

- Scatter plots of individual Attributes and the output labels,
- A pairwise correlation plot among the Attributes,
- A projection of high-dimensional Attributes into a lower dimensional space,
- Line plots for time-based Attributes,
- The first few levels of a regression or classification tree,
- A heatmap across the samples and Attributes, or
- Mosaic plots for examining associations among categorical variables.

These visualizations provide insights that should be used to inform the initial models. It is important to note that some of the most useful visualizations for exploring the data are not necessarily complex or difficult to create. In fact, a simple scatter plot can elicit insights that a model may not be able to uncover, and can lead to the creation of a new attributes or to a transformation of a attributes or the output that improves model performance. The challenge here lies in developing intuition for knowing how to visually explore data to extract information for improvement.

So, lets focus on Univariate visualizations. These plots are used to understand the distribution of a single variable. A few common univariate visualizations are box-and-whisker plots (i.e., box plot), violin plots, or histograms.

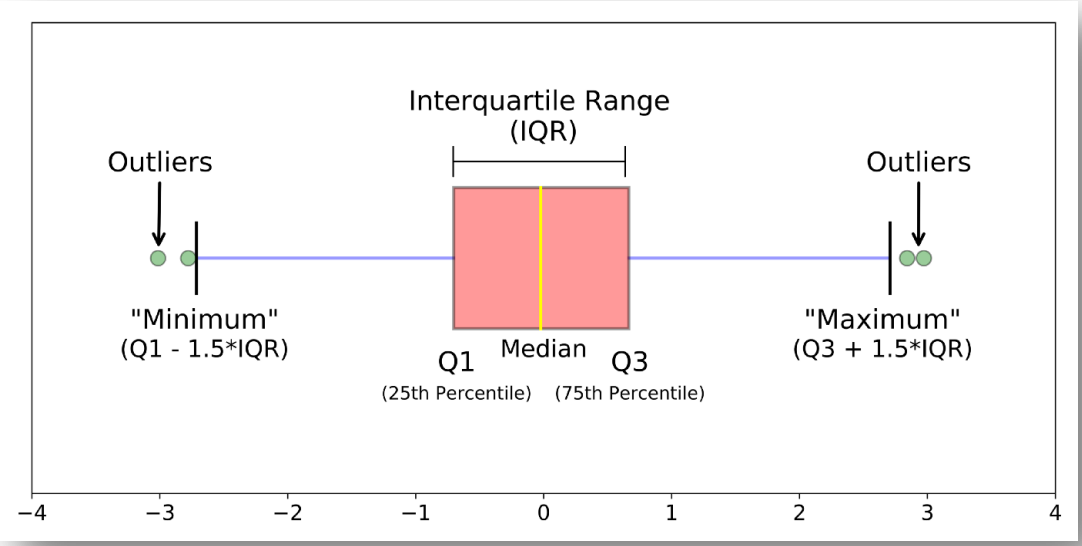
Skewness also plays key role in understanding the distribution of the attribute will elaborate on skewness further in the notebook.



a. Box Plot

Because the foremost goal of modeling is to understand variation in the attribute, the first step should be to understand the distribution of the attribute. For a continuous attribute such as the Free Sulfur Dioxide attribute, it is important to understand if the attribute has a symmetric distribution, if the distribution has a decreasing frequency of larger observations (i.e., the distribution is skewed), if the distribution appears to be made up of two or more individual distributions (i.e., the distribution has multiple peaks or modes), or if there appears to be unusually low or high observations (i.e outliers).

Box Plot help us visualize distribution of single attribute which further help us understanding dataset.



Moving towards technical definition of Box Plots it is a method for graphically demonstrating the locality, spread and skewness groups of numerical data through their quartiles. In addition to the box on a box plot, there can be lines (which are called whiskers) extending from the box indicating variability outside the upper and lower quartiles, thus, the plot is also termed as the box-and-whisker plot and the box-and-whisker diagram. Outliers that differ significantly from the rest of the dataset may be plotted as individual points beyond the whiskers on the box-plot.

Median The median (middle quartile) marks the mid-point of the data and is shown by the line that divides the box into two parts. Half the scores are greater than or equal to this value and half are less.

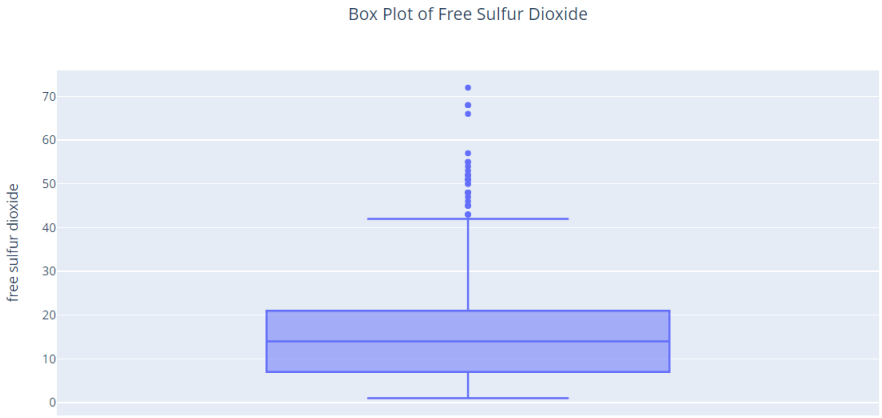
Inter-quartile range(IQR) The middle “box” represents the middle 50% of scores for the group. The range of scores from lower to upper quartile is referred to as the inter-quartile range. The middle 50% of scores fall within the inter-quartile range.

Upper quartile Seventy-five percent of the scores fall below the upper quartile.

Lower quartile Twenty-five percent of scores fall below the lower quartile.

Whiskers The upper and lower whiskers represent scores outside the middle 50%. Whiskers often (but not always) stretch over a wider range of scores than the middle quartile groups.

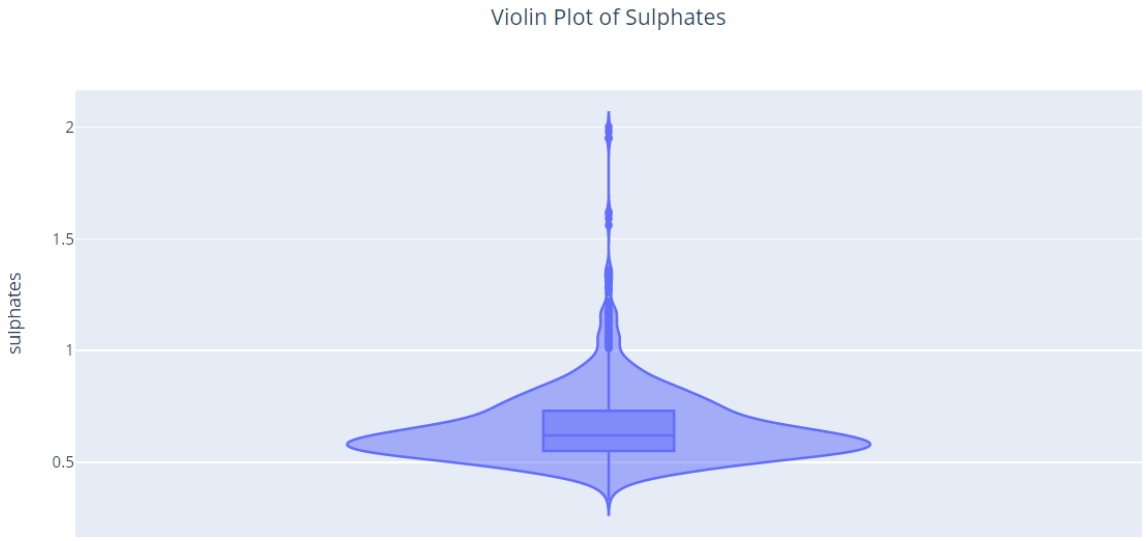
```
from plotly import express as px
import plotly.graph_objects as go

fig = go.Figure(px.box(df, y = 'free sulfur dioxide', title = 'Box Plot of Free Sulfur Dioxide'))
fig.update_layout(title_x=0.5)
fig.show()
```



b. Violin Plot

Violin plot is created by generating a density or distribution of the data and its mirror image. In the above figure we can see the violin plot, where we can now see the many distinct peaks in citric acid distribution. The lower quartile, median, and upper quartile can be added to a violin plot to also consider this information in the overall assessment of the distribution.

```
1 fig = go.Figure(px.violin(df, y = 'sulphates', title = 'Violin Plot of Sulphates', box = True))
2 fig.update_layout(title_x=0.5)
3 fig.show()
```

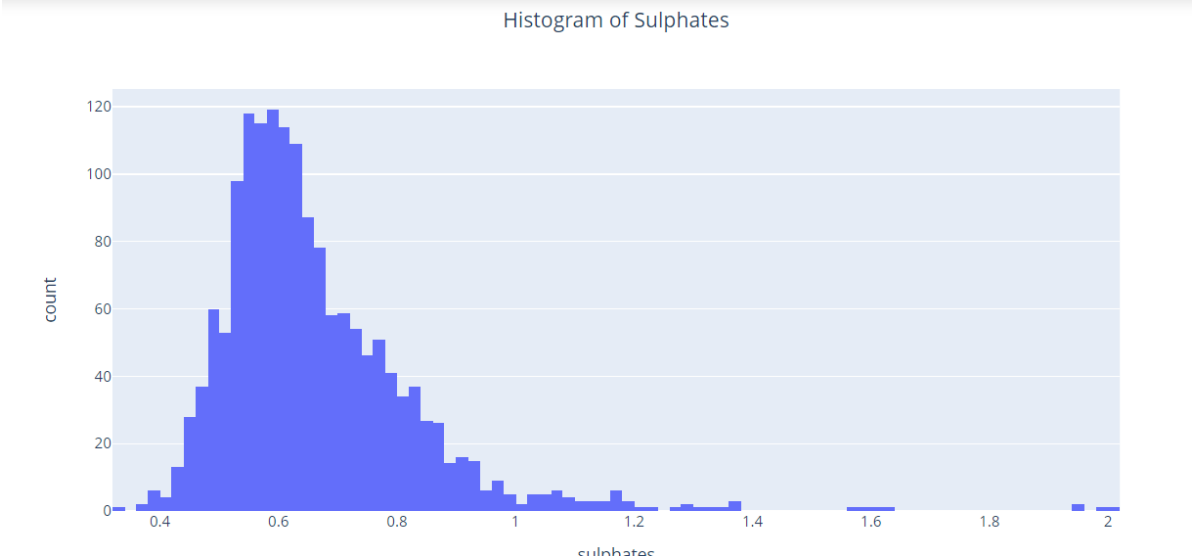


c. Histograms

A histogram is a graphical representation that organizes a group of data points into user-specified ranges. Similar in appearance to a bar graph, the histogram condenses a data series into an easily interpreted visual by taking many data points and grouping them into logical ranges or bins.

Histograms are commonly used in statistics to demonstrate how many of a certain type of variable occurs within a specific range. For example, a census focused on the demography of a country may use a histogram to show how many people are between the ages of 0 - 10, 11 - 20, 21 - 30, 31 - 40, 41 - 50, etc. This histogram would look similar to the example below.

```
fig = go.Figure(px.histogram(df, x = 'sulphates', title = 'Histogram of Sulphates'))
fig.update_layout(title_x=0.5)
fig.show()
```



5. Engeneering Numerical Attribute

Attributes which are continuous are subject to have problems with modeling process. For example, models that construct relationships between the attributes and the output that are based on the rank of the attribute values rather than the actual value, like trees, are immune to attribute distributions that are skewed or to individual samples that have unusual values (i.e., outliers). Other models such as K-nearest neighbors and support vector machines are much more sensitive to attributes with skewed distributions or outliers. Continuous attributes that are highly correlated with each other is another regularly occurring scenario that presents a problem for some models but not for others.

Continuous attributes with commonly occurring issues. The attributes may:

- Be on vastly different scales.
- Follow a skewed distribution where a small proportion of samples are orders of magnitude larger than the majority of the data (i.e., skewness).
- Contain a small number of extreme values.
- Be censored on the low and/or high end of the range.
- Have a complex relationship with the output and be truly predictive but cannot be adequately represented with a simple function or extracted by sophisticated models.
- Contain relevant and overly redundant information. That is, the information collected could be more effectively and efficiently represented with a smaller, consolidated number of new attributes while still preserving or enhancing the new attributes relationships with the output.

The techniques in this notebook have been organized into three general categories.

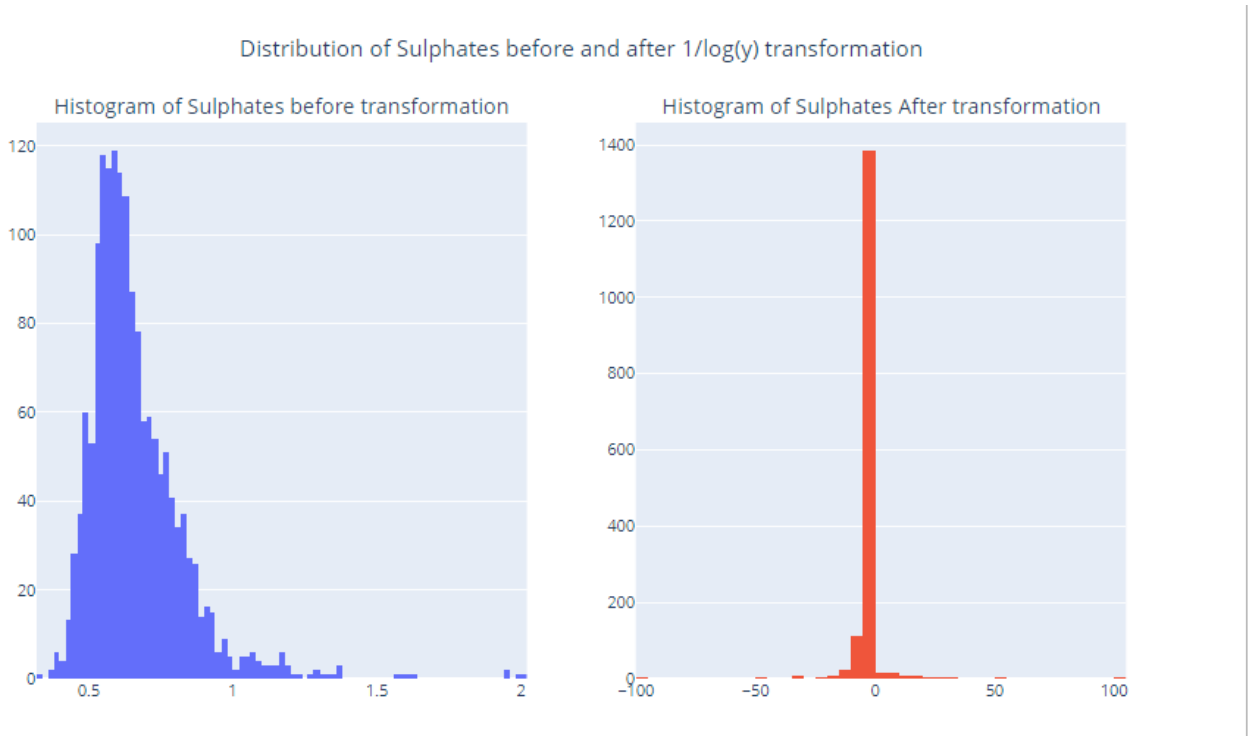
- 1:1 Transformations
- 1:Many Transformations
- Many:Many Transformations

a. One To One Transformation

i. Log Transformation

There are a variety of modifications that can be made to an individual attribute that might improve its utility in a model. The first type of transformations to a single attribute discussed here are those that change the scale of the data. A good example is the transformation described in plot below. In that case, the attribute residual sugar had very skewed distributions and it was shown that using the inverse of the log values improves distribution.

```
from plotly.subplots import make_subplots
import plotly.graph_objects as go

fig = make_subplots(rows=1, cols=2,
                    subplot_titles=("Histogram of Sulphates before transformation", "Histogram of Sulphates After transforma

fig.append_trace(go.Histogram(x = df['sulphates']), row = 1, col = 1)

fig.append_trace(go.Histogram(x = 1/(np.log(df['sulphates'])), nbinsx=40), row = 1, col = 2)

fig.update_layout(height=600, width= np.inf, title_text="Distribution of Sulphates before and after 1/log(y) transformation"
fig.show()
```



ii. Box-Cox And Yeo-Johnson Transformation

A Box-Cox transformation (Box and Cox, 1964) was used to estimate this transformation. The Box-Cox procedure, originally intended as a transformation of a model's outcome, uses maximum likelihood estimation to estimate a transformation parameter λλ in the equation

$$y_i^{(\lambda)} = \begin{cases} \dfrac{(y_i + \lambda_2)^{\lambda_1} - 1}{\lambda_1} & \text{if } \lambda_1 \neq 0, \\ \ln(y_i + \lambda_2) & \text{if } \lambda_1 = 0, \end{cases}$$

In this procedure, λλ is estimated from the data. Because the parameter of interest is in the exponent, this type of transformation is called a **power transformation**. Some values of λλ map to common transformations, such as λλ = 1 (no transformation), λλ = 0 (log), λλ =

0.5 (square root), and $\lambda\lambda = -1$ (inverse). As you can see, the Box-Cox transformation is quite flexible in its ability to address many different data distributions.

It is important to note that the Box-Cox procedure can only be applied to data that is strictly positive.

To address this problem, **Yeo and Johnson (2000)** devised an analogous procedure that can be used on any numeric data.

Also, note that both transformations are unsupervised since, in this application, the outcome is not used in the computations. While the transformation might improve the attribute distribution, it has no guarantee of improving the model. However, there are a variety of parametric models that utilize polynomial calculations on the attribute data, such as most linear models, neural networks, and support vector machines. In these situations, a skewed attribute distribution can have a harmful effect on these models since the tails of the distribution can dominate the underlying calculations.

We can implement Box-Cox and Yeo-Johnson transformation using sklearn's power transforamation.

```
:  1 from sklearn.preprocessing import PowerTransformer
   2
   3 # Yoe-Johnson Transformation
   4 pt = PowerTransformer(method = 'yeo-johnson')
   5 pt.fit(df['sulphates'].values.reshape(-1, 1))
   6 x_yj = pt.transform(df['sulphates'].values.reshape(-1, 1))
   7 print(f"Value of Lambda : {pt.lambdas_}")
   8
```

```
Value of Lambda : [-4.03103339]
```

```
:  1 # Box-Cox Transformation
   2 pt = PowerTransformer(method = 'box-cox')
   3 pt.fit(df['sulphates'].values.reshape(-1, 1))
   4 x_bc = pt.transform(df['sulphates'].values.reshape(-1, 1))
   5 print(f"Value of Lambda : {pt.lambdas_}")
```

```
Value of Lambda : [-1.06205896]
```

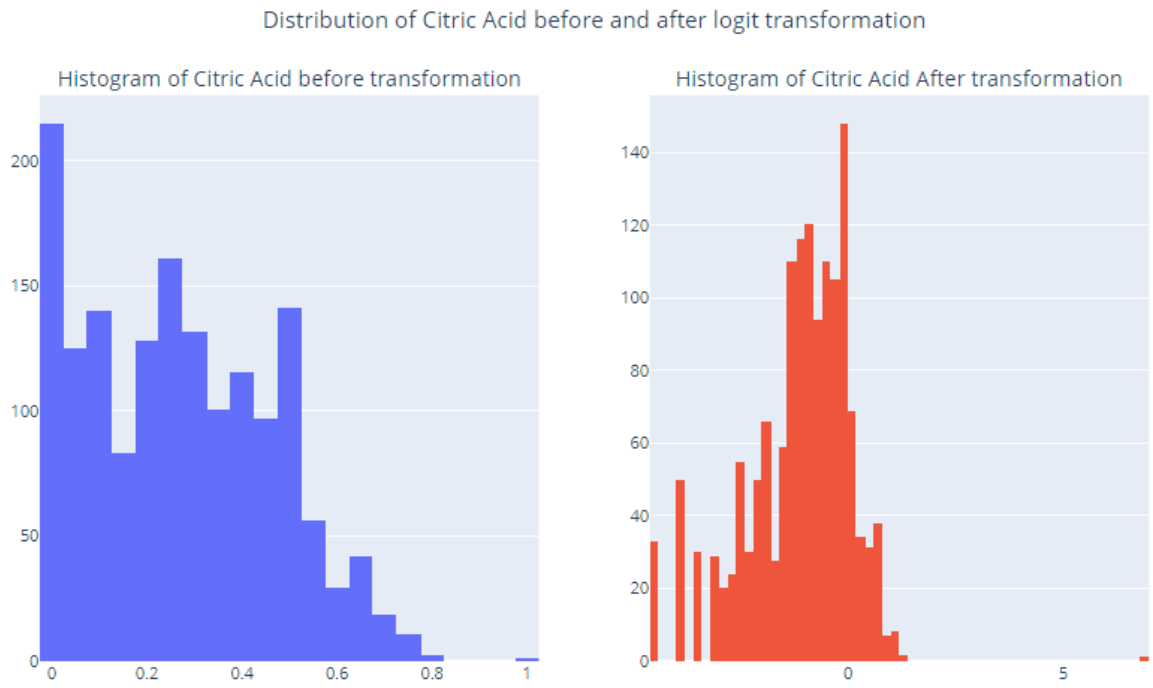
iii. Logistic Transformation

Another important transformation to an individual variable is for a variable that has values bounded between zero and one, such as proportions. The problem with modeling this type of outcome is that model predictions might may not be guaranteed to be within the same boundaries. For data between zero and one, the logit transformation could be used. If p is the variable, the logit transformations is

$$logit(p) = \ln\left(\frac{p}{1-p}\right)$$

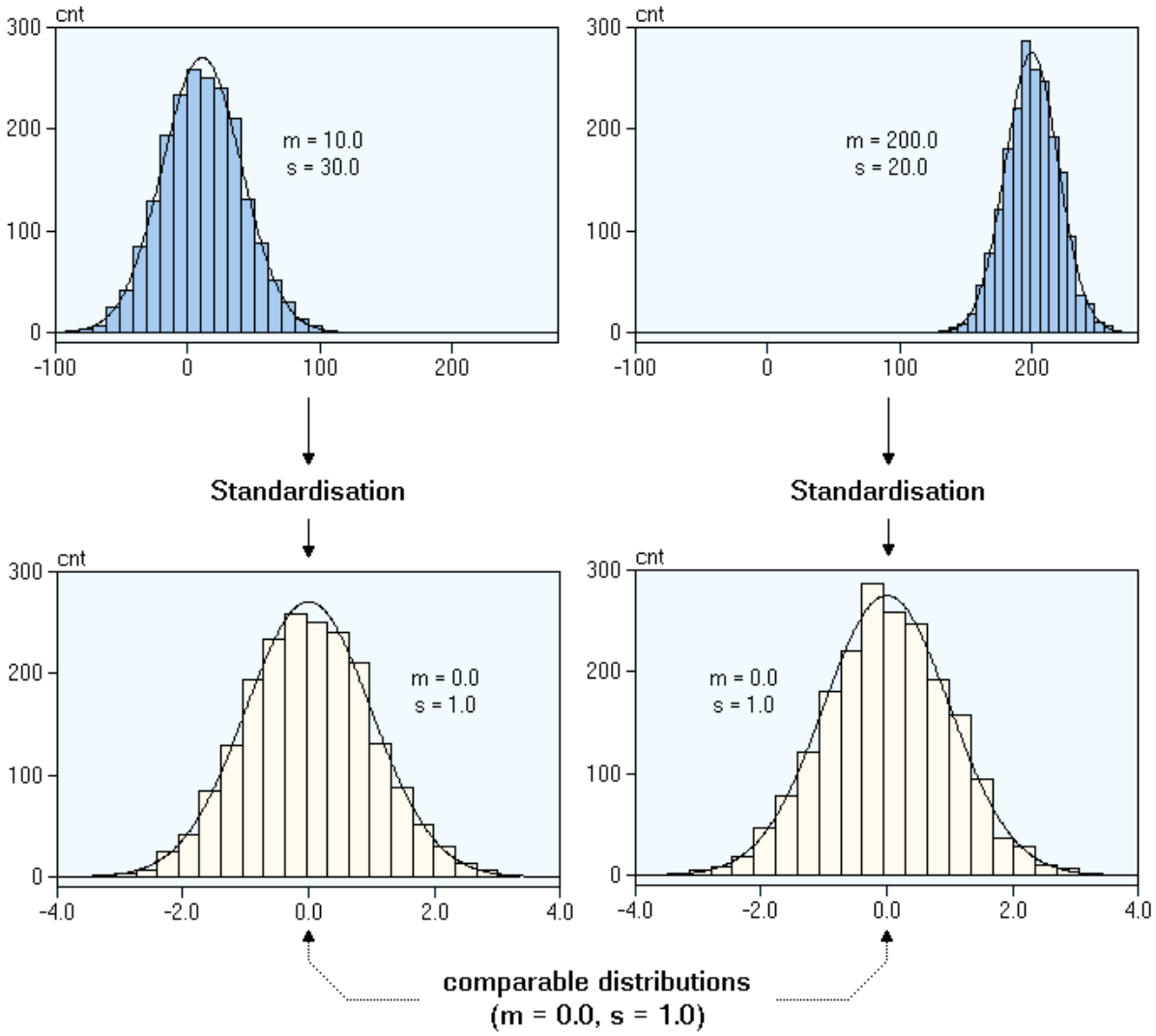
This transformation changes the scale from values between zero and one to values between negative and positive infinity. On the extremes, when the data are absolute zero or one, a small constant can be added or subtracted to avoid division by zero. Once model predictions are created, the inverse logit transformation can be used to place the values back on their original scale. An alternative to the logit transformation is the arcsine transformation. This is primarily used on the square root of the proportions (e.g., y = arcsine(sqrt(p)).

```
from plotly.subplots import make_subplots
import plotly.graph_objects as go

x_citric_acid = np.log(df['citric acid']/(1 - df['citric acid'] + 10e-4))

fig = make_subplots(rows=1, cols=2,
                    subplot_titles=("Histogram of Citric Acid before transformation", "Histogram of Citric Acid After transf

fig.append_trace(go.Histogram(x = df['citric acid']), row = 1, col = 1)

fig.append_trace(go.Histogram(x = x_citric_acid), row = 1, col = 2)

fig.update_layout(height=600, width= np.inf, title_text="Distribution of Citric Acid before and after logit transformation",
fig.show()
```

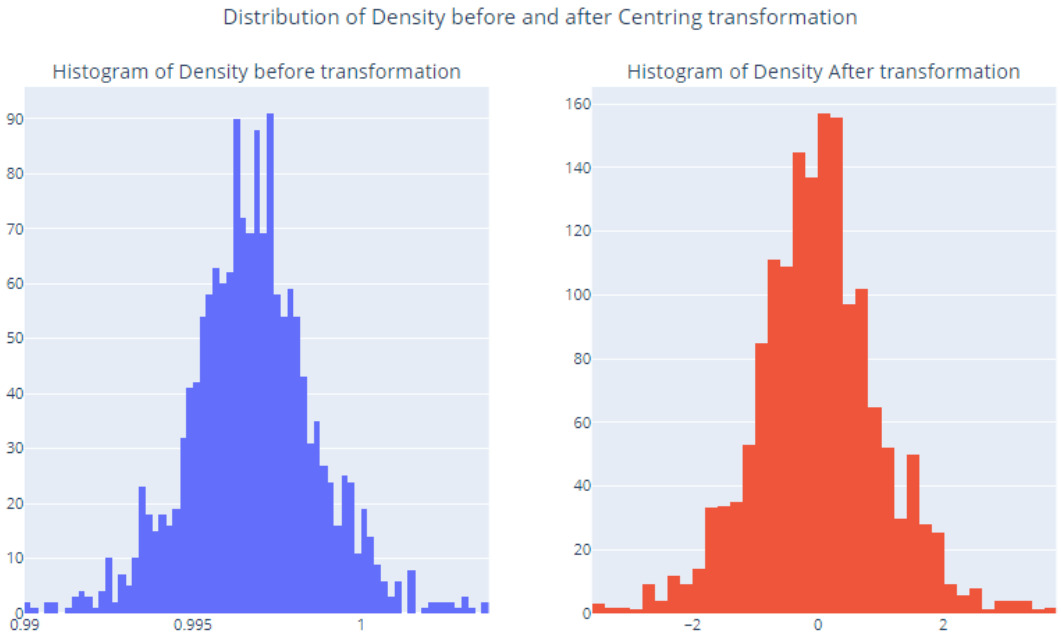


iv. Centring

Another common technique for modifying the scale of a predictor is to standardize its value in order to have specific properties. Centering a predictor is a common technique. The predictor's training set average is subtracted from the predictor's individual values. When this is applied separately to each variable, the collection of variables would have a common mean value (i.e., zero). Similarly, scaling is the process of dividing a variable by the corresponding training set's standard deviation. This ensures that that variables have a standard deviation of one.



Alternatively, range scaling uses the training set minimum and maximum values to translate the data to be within an arbitrary range (usually zero and one). Again, it is emphasized that the statistics required for the transformation (e.g., the mean) are estimated from the training set and are applied to all data sets (e.g., the test set or new samples). These transformations are mostly innocuous and are typically needed when the model requires the predictors to be in common units. For example, when the distance or dot products between predictors are used (such as K-nearest neighbors or support vector machines) a standardization procedure is essential.



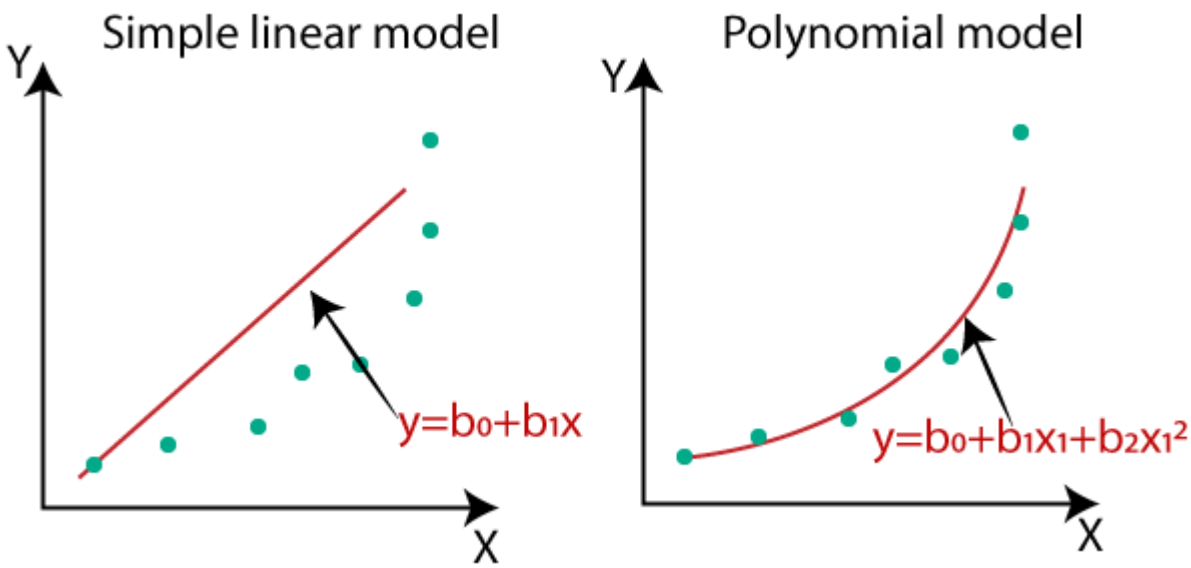
b. One to Many Tranformation

1:Many transformations can be made on a single numeric predictor to expand it to many predictors. These one-to-many transformations of the data can be used to improve model performance. Will discuss following 1:Many transformation methods:-

- Nonlinear Features via Basis Expansions and Splines
- Discretize the attributes

i. Non linear via basis expandion and splines

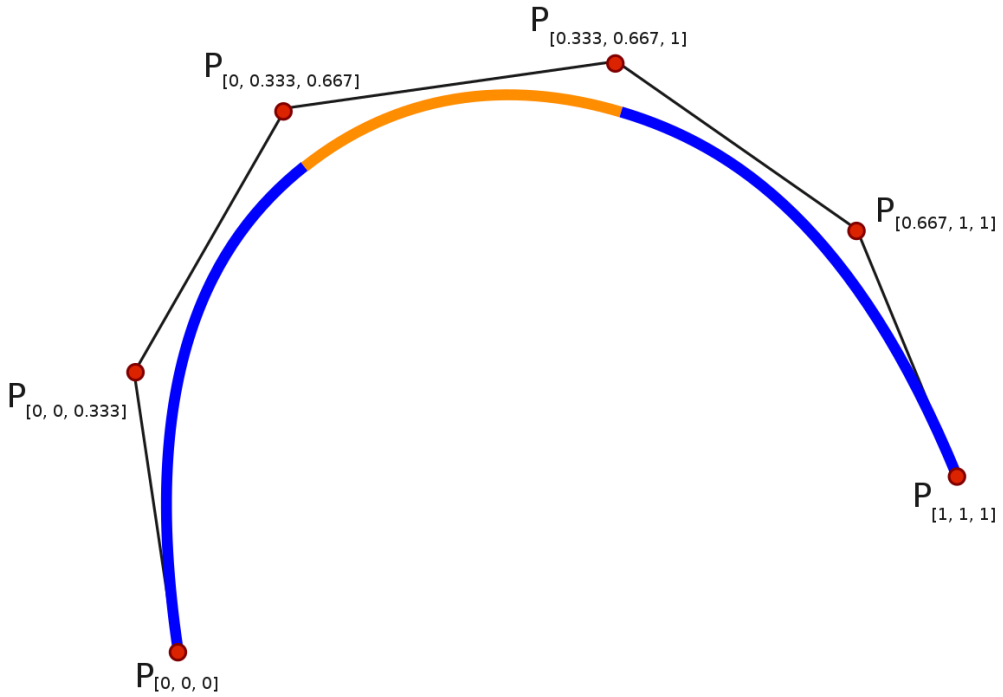
A basis expansion of a attribute x can be achieved by deriving a set of functions fi(x) that can be combined using a linear combination. For a continuous attribute x, a quadratic basis expansion is



To use this basis expansion in the model, the original column is augmented by one new features with squared version of the original. The b(b0, b1 and, b2) values could be estimated using basic linear regression. If the true trend were linear, the rest one regression parameters would presumably be near zero (at least relative to their standard error). Also, the linear regression used to determine the coefficients for the basis expansion could be estimated in the presence of other regressors.

This type of basis expansion, where the pattern is applied globally to the attribute, can often be insufficient. Because at some local point trends may be increase but according to global function(basis expansion) it is decreasing in nature. An alternative method to creating a global basis function that can be used in a regression model is a **polynomial spline**.

Here, the basis expansion creates different regions of the predictor space whose boundaries are called knots.



The polynomial spline uses polynomial functions, usually cubic, within each of these regions to represent the data. We would like the cubic functions to be connected at the knots, so specialized functions can be created to ensure an overall continuous function. Here, the number of knots controls the number of regions and also the potential complexity of the function. If the number of knots is low (perhaps three or less), the basis function can represent relatively simple trends. Functions with more knots are more adaptable but are also more likely to overfit the data.

In above image we created 5 different polynomial regions means 5 different polynomial regression equations and then join them all after smoothing in one.

The knots are typically chosen using percentiles of the data so that a relatively equal amount of data is contained within each region. For example, a spline with three regions typically places the knots at the 33.3% and 66.7% percentiles. Once the knots are determined, the basis functions can be created and used in a regression model. There are many types of polynomial splines and the approach described here is often called a natural cubic spline.

How many knots should be used? This is a tuning parameter that can be determined using grid search or through visual inspection of the smoothed results

ii. Descretize the attributes

Binning, also known as categorization or discretization, is the process of translating a quantitative variable into a set of two or more qualitative buckets (i.e., categories). For example, a variable might be translated into quantiles; the buckets would be for whether the numbers fell into the first 25% of the data, between the 25th and median, etc. In this case, there would be four distinct values of the binned version of the data.

There are a few apparent reasons for subjecting the data to such a transformation:

- Some feel that it simplifies the analysis and/or interpretation of the results. Suppose that a person's age was a predictor and this was binned by whether someone was above 40 years old or not. One might be able to make a statement that there is a 25% increase in the probability of the event between younger and older people. There is no discussion of per-unit increases in the response.
- Binning may avoid the problem of having to specify the relationship between the predictor and outcome. A set of bins can be perceived as being able to model more patterns without having to visualize or think about the underlying pattern.
- Using qualitative versions of the attributes may give the perception that it reduces the variation in the data. Unfortunately, this artificial reduction in variation can lead to a high false positive rate when the predictor is unrelated to the outcome.

Note: The strategy should not be part of the normal operating procedure. Categorizing predictors should be a method of last resort.

c. Many to Many Transformation

When creating new features from multiple predictors, there is a possibility of correcting a variety of issues such as outliers or collinearity. It can also help reduce the dimensionality of the predictor space in ways that might improve performance and reduce computational time for models. Will discuss following Many:Many transformations techniques in this notebook

Principal Component Analysis

i. Principal Component analysis

Question may arise why PCA in Feature Engineering? Specifically, the objective of PCA is to find linear combinations of the original attributes such that the combinations summarize the maximal amount of variation in the original predictor space. From a statistical perspective, variation is synonymous with information. So, by finding combinations of the original attributes that capture variation, we find the subspace of the data that contains the information relevant to the attributes.

An important side benefit of this technique is that the resulting PCA scores are uncorrelated. This property is very useful for modeling techniques (e.g., multiple linear regression, neural networks, support vector machines, and others) that need the attributes to be relatively uncorrelated.

PCA is a particularly useful tool when the available data are composed of one or more clusters of attributes that contain redundant information (e.g., attributes that are highly correlated with one another).

Principal component regression, which describes the process of first reducing dimension and then regressing the new component scores onto the response is a stepwise process.

Lets try to fetch information of linear correlated attributes from the data.

```
from sklearn.decomposition import PCA

pca = PCA(n_components = 8)

X = df.drop("density", axis = 1)
y = df['density']

x_pca = pca.fit_transform(X)

pc = linear_regression(x_pca, y.values.reshape(-1, 1))
```

```
Mean Squared Error = 8.428507189961776e-07
```

6. Feature Selection

In 3rd section of this notebook we have studied tools for engineering features (or attributes) to put them in a form that enables models to better find the predictive signal relative to the outcome. Some of these techniques, like 1-1 transformations or dimensions reduction methods, lead to a new predictor set that has as many or fewer attributes than the original data. Other transformations, like basis expansions, generate more features than the original data.

The hope is that some of the newly engineered attributes capture a predictive relationship with the outcome. But some may not be relevant to the outcome. Moreover, many of the original attributes also may not contain predictive information. **For a number of models, predictive performance is degraded as the number of uninformative attributes increases. Therefore, there is a genuine need to appropriately select attributes for modeling.**

One Question can ask a Question at one phase of ML process we created features using feature engineering techniques and at another phase we are try to remove few features. The answer is pretty simple both phases are important in Machine Learning. We have seen importance od feature engineering lets focus our discussion on feature scaling now.

- Some models, notably support vector machines and neural networks, are sensitive to irrelevant attributes.
- Other models like linear or logistic regression are vulnerable to correlated attributes. Removing correlated attributes will reduce multicollinearity and thus enable these types of models to be fit.
- Even when a predictive model is insensitive to extra attributes, it makes good scientific sense to include the minimum possible set that provides acceptable results. In some cases, removing attributes can reduce the cost of acquiring data or improve the throughput of the software used to make predictions.

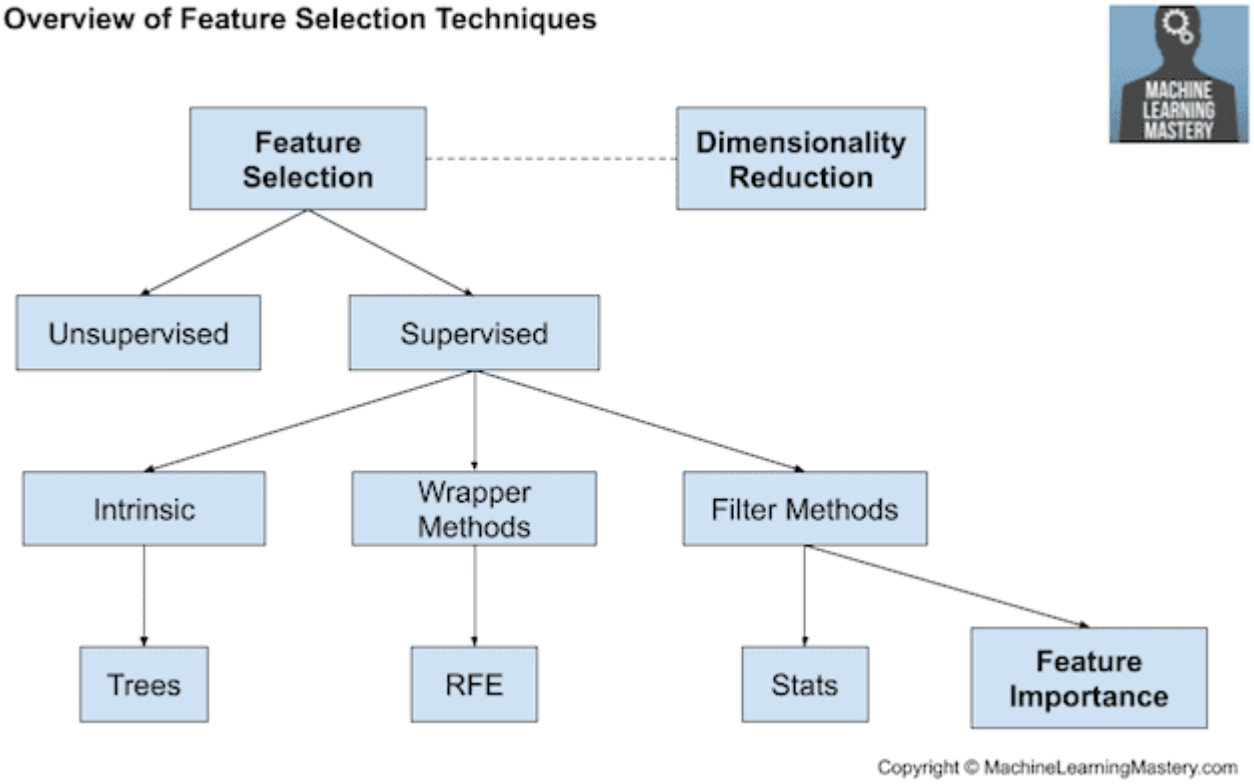
Rest of the notebook will focus on supervised feature selection where the choice of which attribute to retain is guided by their effect on the outcome.

a. Classes Of Feature Selection Methodology

Feature selection methodologies fall into three general classes:

- Intrinsic (or implicit) methods.
- Filter Methods.
- Wrapper methods.

Intrinsic methods have feature selection naturally incorporated with the modeling process, whereas filter and wrapper methods work to marry feature selection approaches with modeling techniques. The most seamless and important of the three classes for reducing features are intrinsic methods.



i. Intrinsic Method

There are some machine learning algorithms that perform feature selection automatically as part of learning the model. We might refer to these techniques as intrinsic feature selection methods.

The most seamless and important of the three classes for reducing features are intrinsic methods.

Few intrinsic methods are penalized regression models like Lasso and decision trees, including ensembles of decision trees like random forest. Although some algorithms, such as random forest, deliberately force splits on irrelevant predictors when constructing the tree.

The main advantage of implicit feature selection methods is that they are relatively fast since the selection process is embedded within the model fitting process; no external feature selection tool is required.

The main downside to intrinsic feature selection is that it is model-dependent. In addition, some intrinsic models (like single tree models) use a greedy approach to feature selection. Greedy approaches usually identify a narrow set of features that have sub-optimal predictive performance.

ii. Filter Method

There are some machine learning algorithms that perform feature selection automatically as part of learning the model. We might refer to these techniques as intrinsic feature selection methods.

The most seamless and important of the three classes for reducing features are intrinsic methods.

Few intrinsic methods are penalized regression models like Lasso and decision trees, including ensembles of decision trees like random forest. Although some algorithms, such as random forest, deliberately force splits on irrelevant predictors when constructing the tree.

The main advantage of implicit feature selection methods is that they are relatively fast since the selection process is embedded within the model fitting process; no external feature selection tool is required.

The main downside to intrinsic feature selection is that it is model-dependent. In addition, some intrinsic models (like single tree models) use a greedy approach to feature selection. Greedy approaches usually identify a narrow set of features that have sub-optimal predictive performance.

iii. Wrapper Method

Wrapper methods use iterative search procedures that repeatedly supply attribute subsets to the model and then use the resulting model performance estimate to guide the selection of the next subset to evaluate.

If successful, a wrapper method will iterate to a smaller set of attributes that has better predictive performance than the original attribute set.

Wrapper methods can take either a greedy or non-greedy approach to feature selection. A greedy search is one that chooses the search path based on the direction that seems best at the time in order to achieve the best immediate benefit.

While this can be an effective strategy, it may show immediate benefits in predictive performance that stall out at a locally best setting.

A nongreedy search method would re-evaluate previous feature combinations and would have the ability to move in a direction that is initially unfavorable if it appears to have a potential benefit after the current step. This allows the non-greedy approach to escape being trapped in a local optimum.

An example of a greedy wrapper method is backwards selection (otherwise known as **recursive feature elimination or RFE**). Here, the attributes are initially ranked by some measure of importance. An initial model is created using the complete attribute set. The next model is based on a smaller set of attributes where the least important have been removed. This process continues down a prescribed path (based on the ranking generated by the importance) until a very small number of attributes are in the model.

Note : This approach to feature selection will likely fail if there are important interactions between attributes where only one of the attributes is significant

Wrappers have the potential advantage of searching a wider variety of attribute subsets than simple filters or models with built-in feature selection. They have the most potential to find the globally best attribute subset (if it exists). The primary drawback is the computational time required for these methods to find the optimal or near optimal subset.

Another disadvantage of wrappers is that they have the most potential to overfit the attributes to the training data and require external validation