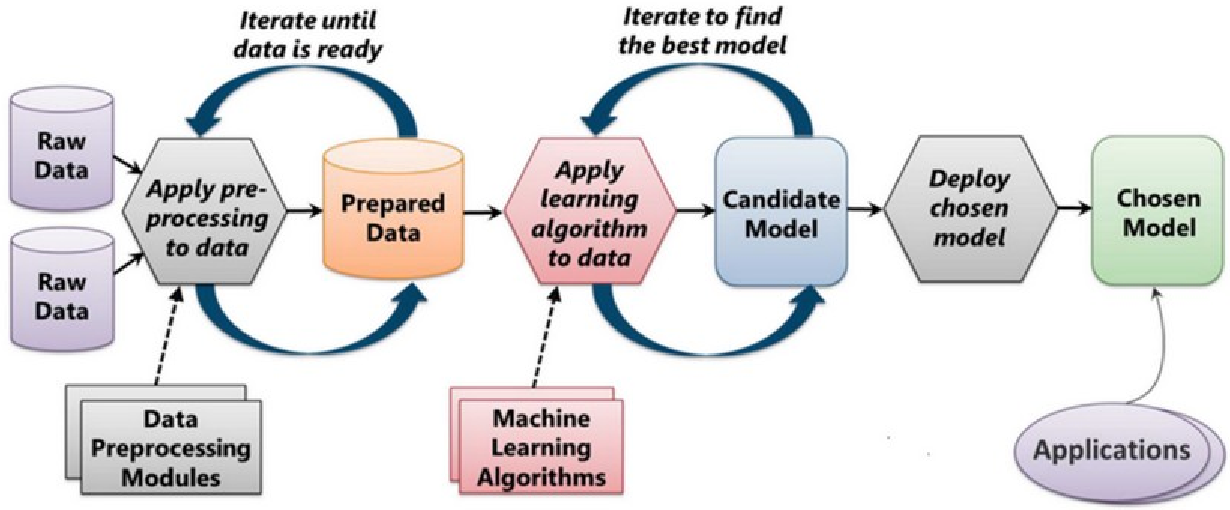
Data Preprocessing & Feature Selection

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviours or trends, and is likely to contain many errors. **Data preprocessing** is a proven method of resolving such issues.

In Real world data are generally incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data. Noisy: containing errors or outliers. Inconsistent: containing discrepancies in codes or names.

The Machine Learning Process



Iterate until data is ready
Iterate to find the best model
Raw Data
Raw Data
Apply pre-processing to data
Prepared Data
Apply learning algorithm to data
Candidate Model
Deploy chosen model
Chosen Model
Data Preprocessing Modules
Machine Learning Algorithms
Applications

From "Introduction to Microsoft Azure" by David Chappell

Few Data Preprocessing Techniques

1. Handling the missing value

2. Data Transformation
    a. Scaling (Min-Max Normalization)
    b. Mean - Normalization
    c. Standardization (Z score)
    d. Binarize Data (Make Binary)

3. Handling Categorical Data
    a. Lable Encoding
    b. One Hot Encoding

1. **Handling the missing values:**

   1. **Removing data (Row or Column):** This method commonly used to handle the null values. Here, we either delete a particular row if it has a null value for a particular feature or a particular column if it has missing values more than some threshold.

      This method is advised only when there are enough samples in the data set. One has to make sure that after we have deleted the data, there is no addition of bias. Removing the data will lead to loss of information which may not give the expected results while predicting the output.

   2. **Imputation:** This strategy can be applied on a feature which has numeric data. We can calculate the **mean, median or mode** of the feature and replace it with the missing values. This is an approximation which can add variance to the data set. But the loss of the data can be negated by this method which yields better results compared to removal of rows and columns.

      Replacing with any of the above three approximations is a statistical approach of handling the missing values. This method is also called as **leaking the data** while training. Another way is to approximate it with the deviation of neighbouring values. This works better if the data is linear.

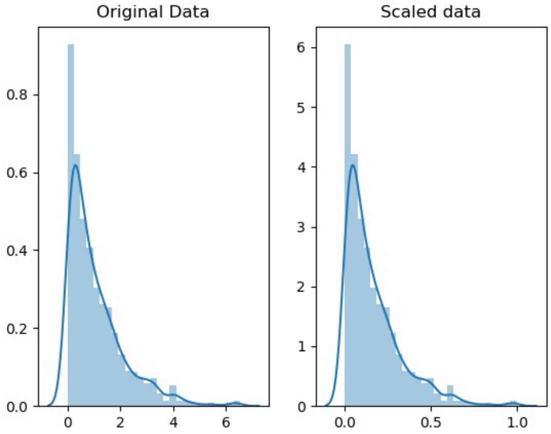
2. **Data Transformation**

   1. Scaling (Min-Max normalization): When your data is comprised of attributes with varying scales, many machine learning algorithms can benefit from rescaling the attributes to all have the same scale.

   Often this is referred to as normalization and attributes are often rescaled into the range between 0 and 1. This is useful for optimization algorithms used in the core of machine learning algorithms like gradient descent. It is also useful for algorithms that weight inputs like regression and neural networks and algorithms that use distance measures like K-Nearest Neighbours.

   The Formula:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

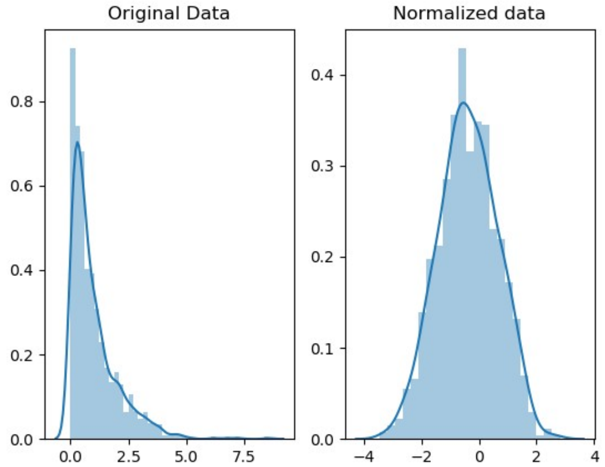


2. Mean-Normalization: The point of normalization is to change your observations so that they can be described as a normal distribution.

Normal distribution (Gaussian distribution), also known as the bell curve, is a specific statistical distribution where a roughly equal observations fall above and below the mean, the mean and the median are the same, and there are more observations closer to the mean.

The Formula is:

$$x' = \frac{x - x_{mean}}{x_{max} - x_{min}}$$



In scaling, the range of data is changed while in normalization the shape of the distribution of data is changed.

Data needs to be normalized if a machine learning or statistics technique that assumes normal distribution of data, is going to be used. e.g. linear regression, linear discriminant analysis (LDA) and Gaussian Naive Bayes.

3. Standardization (Z score): Standardization is a useful technique to transform attributes with a Gaussian distribution and differing means and standard deviations to a standard Gaussian distribution with a mean of 0 and a standard deviation of 1.

It is most suitable for techniques that assume a Gaussian distribution in the input variables and work better with rescaled data, such as linear regression, logistic regression and linear discriminate analysis.

The formula is:

$$x' = \frac{x - x_{mean}}{\sigma}$$

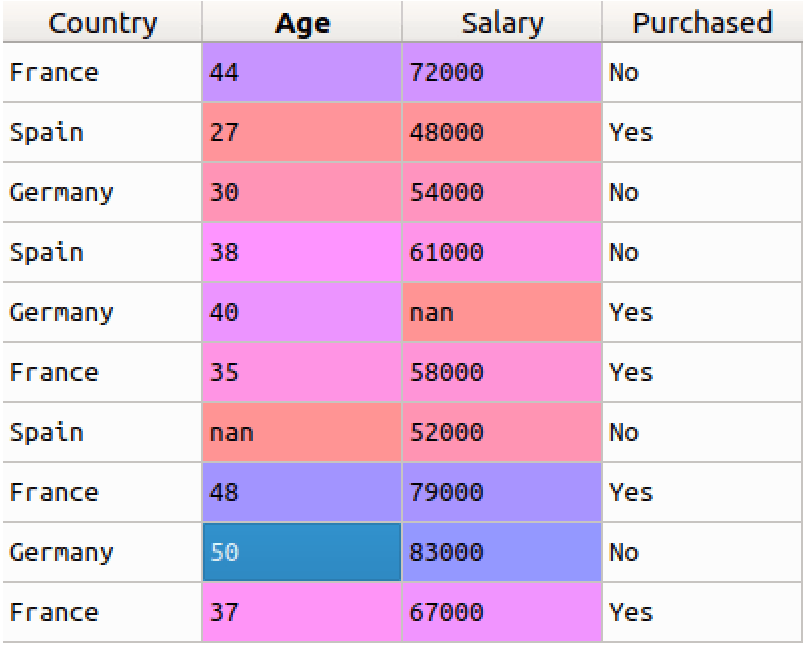
where x is the original feature vector, x_{mean} is the mean of that feature vector, and σ is its standard deviation.

4. Binarize Data (Make Binary): You can transform your data using a binary threshold. All values above the threshold are marked 1 and all equal to or below are marked as 0.
This is called binarizing your data or threshold your data. It can be useful when you have probabilities that you want to make crisp values. It is also useful when feature engineering and you want to add new features that indicate something meaningful.

3. **<u>Handling Categorical Data</u>**

There are mainly two types of encoders – **Label Encoder** and **One Hot Encoder**. They are used to convert categorical data, or text data, into numbers, which our predictive models can better understand.

1. Label Encoding: Let's consider the following data:

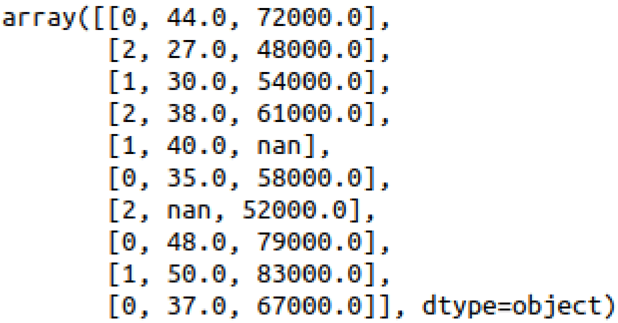


Country	Age	Salary	Purchased
France	44	72000	No
Spain	27	48000	Yes
Germany	30	54000	No
Spain	38	61000	No
Germany	40	nan	Yes
France	35	58000	Yes
Spain	nan	52000	No
France	48	79000	Yes
Germany	50	83000	No
France	37	67000	Yes

Data from SuperDataScience

In this example, the first column is the country column, which is all text. As you might know by now, we can't have text in our data if we're going to run any kind of model on it. So before we can run a model, we need to make this data ready for the model. And to convert this kind of categorical text data into model-understandable numerical data, we use the Label Encoder class.

Once data is encoded it will be changed as below



```
array([[0, 44.0, 72000.0],
       [2, 27.0, 48000.0],
       [1, 30.0, 54000.0],
       [2, 38.0, 61000.0],
       [1, 40.0, nan],
       [0, 35.0, 58000.0],
       [2, nan, 52000.0],
       [0, 48.0, 79000.0],
       [1, 50.0, 83000.0],
       [0, 37.0, 67000.0]], dtype=object)
```

That's all label encoding is about. But depending on the data, label encoding introduces a new problem. For example, we have encoded a set of country names into numerical data. This is actually categorical data and there is no relation, of any kind, between the rows.

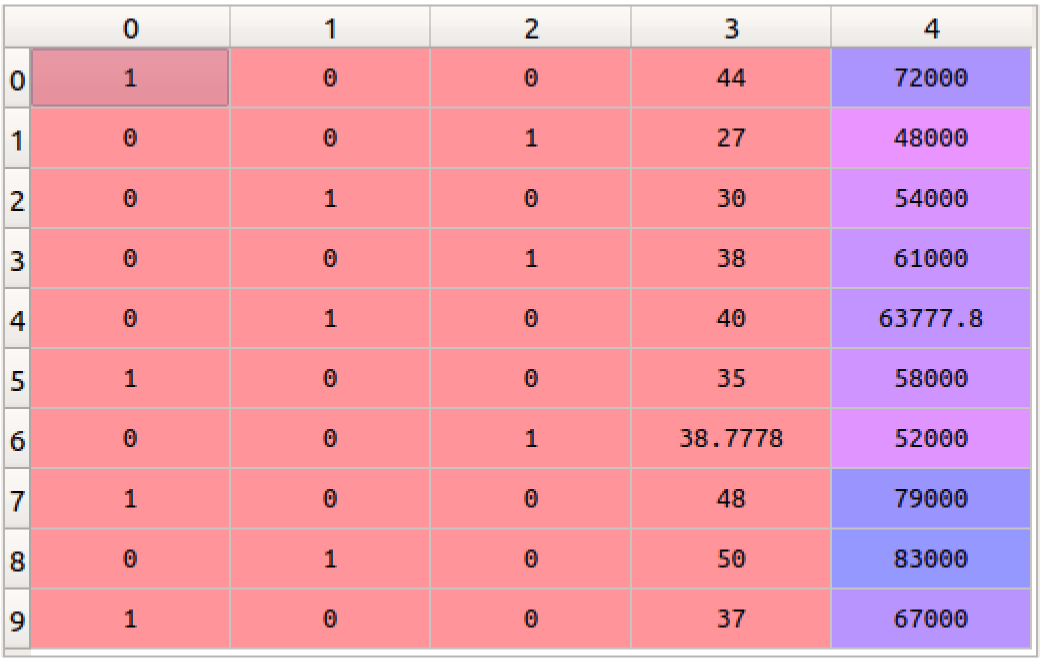
The problem here is, since there are different numbers in the same column, the model will misunderstand the data to be in some kind of order, 0 < 1 < 2. But this isn't the case at all. To overcome this problem, we use One Hot Encoder.

2. One Hot Encoding

Now, as we already discussed, depending on the data we have, we might run into situations where, after label encoding, we might confuse our model into thinking that a column has data with some kind of order or hierarchy, when we clearly don't have it. To avoid this, we 'OneHotEncode' that column.

What one hot encoding does is, it takes a column which has categorical data, which has been label encoded, and then splits the column into multiple columns. The numbers are replaced by 1s and 0s, depending on which column has what value. In our example, we'll get three new columns, one for each country—France, Germany, and Spain.

For rows which have the first column value as France, the 'France' column will have a '1' and the other two columns will have '0's. Similarly, for rows which have the first column value as Germany, the 'Germany' column will have a '1' and the other two columns will have '0's.



	0	1	2	3	4
0	1	0	0	44	72000
1	0	0	1	27	48000
2	0	1	0	30	54000
3	0	0	1	38	61000
4	0	1	0	40	63777.8
5	1	0	0	35	58000
6	0	0	1	38.7778	52000
7	1	0	0	48	79000
8	0	1	0	50	83000
9	1	0	0	37	67000

As you can see, we have three new columns with 1s and 0s, depending on the country that the rows represent.

Feature Selection

All of the features we find in the dataset might not be useful in building a machine learning model to make the necessary prediction. Using some of the features might even make the predictions worse. So, feature selection plays a huge role in building a machine learning model.

What is correlation?

Correlation is a statistical term which in common usage refers to how close two variables are to having a linear relationship with each other.

Ex. X1 = 2X2 --> High Correlation (Linearly Dependent)

Features with high correlation are more linearly dependent and hence have almost the same effect on the dependent variable (Y).

So, when two features have high correlation, we can drop one of the two features.

Exercise:

1) Perform all data preprocessing tasks and feature selection on "Exercise-CarData.csv"

Hint: Check "na_values" attribute of pd.read()