Feature Engineering & Feature Selection

Handling Missing Values:

What is Missing Values:

Missing data is defined as the values or data that is not stored (or not present) for some variable/s in the given dataset.

Missing data, or missing values, **occur when you don't have data stored for certain variables or participants**. Data can go missing due to incomplete data entry, equipment malfunctions, lost files, and many other reasons. In any dataset, there are usually some missing data.

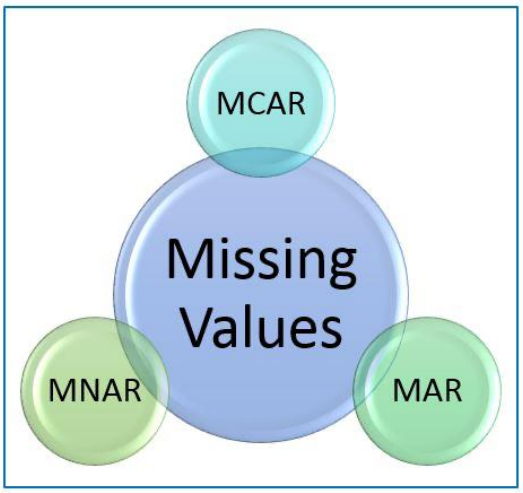
Why Missing Data present in dataset:

There can be multiple reasons why certain values are missing from the data. Reasons for the missing data from the dataset affect the approach of handling missing data. So, it's necessary to understand why the data could be missing.

Some of the reasons are listed below:

- Past data might get corrupted due to improper maintenance.
- Observations are not recorded for certain fields due to some reasons. There might be a failure in recording the values due to human error.
- The user has not provided the values intentionally.

Types of Missing values:



- **Missing Completely at Random (MCAR):**

  In MCAR, the probability of data being missing is the same for all the observations.
  In this case, there is no relationship between the missing data and any other values observed or unobserved (the data which is not recorded) within the given dataset.

  That is, missing values are completely independent of other data. There is no pattern.
  In the case of MCAR, the data could be missing due to human error, some system/equipment failure, loss of sample, or some unsatisfactory technicalities while recording the values.

  For Example, suppose in a library there are some overdue books. Some values of overdue books in the computer system are missing. The reason might be a human error like the librarian forgot to type in the values. So, the missing values of overdue books are not related to any other variable/data in the system. It should not be assumed as it's a rare case. The advantage of such data is that the statistical analysis remains unbiased.

- **Missing At Random (MAR):**

  Missing at random (MAR) means that the reason for missing values can be explained by variables on which you have complete information as **there is some relationship between the missing data and other values/data.**:

  In this case, **the data is not missing for all the observations. It is missing only within sub-samples of the data and there is some pattern in the missing values.**

  For example, if you check the survey data, you may find that all the people have answered their 'Gender' but 'Age' values are mostly missing for people who have answered their 'Gender' as 'female'. (The reason being most of the females don't want to reveal their age.)

  So, the probability of data being missing depends only on the observed data.

  In this case, the variables 'Gender' and 'Age' are related and the reason for missing values of the 'Age' variable can be explained by the 'Gender' variable but you cannot predict the missing value itself.

  Suppose a poll is taken for overdue books of a library. Gender and the number of overdue books is asked in the poll. Assume that most of the females answer the poll and men are less likely to answer. So why the data is missing can be explained by another factor that is gender.

  In this case, the statistical analysis might result in bias.
  Getting an unbiased estimate of the parameters can be done only by modelling the missing data.

- **Missing Not at Random (MNAR):**

  Missing values depend on the unobserved data.

  If there is some structure/pattern in missing data and other observed data cannot explain it, then it is Missing Not at Random (MNAR).

  If the missing data does not fall under the MCAR or MAR then it can be categorized as MNAR.
  It can happen due to the reluctance of people in providing the required information. **A specific group of people may not answer some questions in a survey**.

  For example, suppose the name and the number of overdue books is asked in the poll for a library. So, most of the people having no overdue books are likely to answer the poll. People having more overdue books are less likely to answer the poll.

  So, in this case, the missing value of the number of overdue books depends on the people who have more books overdue.

Another example, people having less income may refuse to share that information in a survey.
In the case of MNAR as well the statistical analysis might result in bias.

Why do we need to take care about Handling the Missing Values:

It is important to handle the missing values appropriately.

- Many machine learning algorithms fail if the dataset contains missing values. However, algorithms like K-nearest and Naive Bayes support data with missing values.
- You may end up building a biased machine learning model which will lead to incorrect results if the missing values are not handled properly.
- Missing data can lead to a lack of precision in the statistical analysis.

How to handle the missing values:

Analyze each column with missing values carefully to understand the reasons behind the missing values as it is crucial to find out the strategy for handling the missing values.

1. **Deleting the Missing Values:**

   Generally, this approach is not recommended. It is one of the quick and dirty techniques one can use to deal with missing values.

   If the missing value is of the type **Missing Not at Random (MNAR),** then it should not be deleted.
   If the missing value is of type **Missing at Random (MAR)** or **Missing Completely At Random (MCAR)** then it can be deleted.

   The disadvantage of this method is one might end up deleting some useful data from the dataset.
   There are 2 ways one can delete the missing values:

   - **Deleting the entire row:** If a row has many missing values, then you can choose to drop the entire row. If every row has some (column) value missing then you might end up deleting the whole data.

   - **Deleting the entire column:** If a certain column has many missing values, then you can choose to drop the entire column.

2. **Imputing the Missing Value:**

   There are different ways of replacing the missing values. You can use the python libraries Pandas and Sci-kit learn as follows:

   - **Replacing With Arbitrary Value:** If you can make an educated guess about the missing value then you can replace it with some arbitrary value using the following code.

   - **Replacing With Mean, Median, Mode:**

   - **Replacing with previous value – Forward fill**

   - **Replacing with next value – Backward fill**

   - **Interpolation:** Missing values can also be imputed using interpolation. Pandas interpolate method can be used to replace the missing values with different interpolation methods like ‘polynomial’, ‘linear’, ‘quadratic’. Default method is ‘linear’.

Interpolation is a technique used to estimate values of a function or a set of data points within a given range. In other words, it is the process of estimating unknown values of a function or a set of data points by using known values at neighbouring points. Interpolation is commonly used in various fields such as mathematics, engineering, computer science, and data analysis to estimate values of functions or missing data points. There are different methods of interpolation such as linear, polynomial, spline, and kriging interpolation, and the choice of method depends on the type of data and the application.

- **Linear Interpolation:**

  Linear interpolation is a method to estimate values that are between two known data points. It is a process of finding a linear equation that passes through two data points, and then using this equation to estimate the value at a new data point that lies between the original two data points. In other words, it is a way to estimate missing or intermediate data points based on known data points. Linear interpolation assumes that the relationship between the data points is linear, which means that there is a constant rate of change between the two known points.

- **Polynomial Interpolation:**

  Polynomial interpolation is a type of interpolation where a polynomial function is fitted to a set of data points to approximate a curve or a function. The polynomial function is a mathematical expression that can be used to represent the relationship between the independent variable (x) and the dependent variable (y).

  The general form of a polynomial function is:

  f(x) = a0 + a1x + a2x^2 + ... + anx^n

  where a0, a1, a2, ..., an are coefficients that determine the shape and behaviour of the polynomial function, and n is the degree of the polynomial.

  In polynomial interpolation, the coefficients of the polynomial function are determined by fitting the function to a set of data points, such that the function passes through each point. The degree of the polynomial is determined by the number of data points available and the desired accuracy of the interpolation.

  Polynomial interpolation is commonly used in engineering, science, and mathematics to approximate complex functions or curves that cannot be easily represented by simple functions or equations. It is also used in computer graphics and image processing to generate smooth curves or surfaces that pass through a set of control points.

- **Quadratic Interpolation:**
  Quadratic interpolation is a type of polynomial interpolation in which a quadratic polynomial is used to approximate a set of data points. It involves fitting a parabolic curve between three adjacent data points to estimate the value of a function at an intermediate point. The quadratic polynomial is given by:

  f(x) = ax^2 + bx + c

  where a, b, and c are coefficients that are determined using the values of the three adjacent data points. The process of quadratic interpolation is useful in cases where the data points have a non-linear relationship, and a linear or polynomial model is not sufficient for accurately estimating the function's value at an intermediate point.

3. Imputing the Missing Values for Categorical Features:

- **Impute the Most Frequent Value**
- **Impute the Value "missing", which treats it as a Separate Category**

4. Imputation the Missing values using Sci-kit learn:

- **Univariate Approach:**
  In a Univariate approach, only a single feature is taken into consideration. You can use the class SimpleImputer and replace the missing values with mean, mode, median or some constant value.

- **Multivariate Approach:**
  In a multivariate approach, more than one feature is taken into consideration. There are two ways to impute missing values considering the multivariate approach. Using KNNImputer or IterativeImputer classes.
  Let's take an example of a titanic dataset.

  Suppose the feature 'age' is well correlated with the feature 'Fare' such that people with lower fares are also younger and people with higher fares are also older.

  In that case, it would make sense to impute low age for low fare values and high age for high fares values. So here we are taking multiple features into account by following a multivariate approach.

- **Nearest Neighbours Imputations (KNNImputer):**

  KNNImputer is a machine learning algorithm used for imputing missing values in a dataset. It is a type of imputation technique where missing values in a dataset are replaced by the K-Nearest Neighbours of each sample with missing values.

  In KNNImputer, the missing values in a feature vector are imputed by taking the average of K-Nearest Neighbours of that feature vector. The distance between the feature vectors is calculated using Euclidean distance or other distance metrics. K is a user-defined parameter that determines the number of nearest neighbours to be considered for imputing the missing values.

  KNNImputer is a simple yet effective technique for handling missing values in a dataset, and it can be applied to both numerical and categorical data. However, it may not be suitable for datasets with a large number of missing values, as it can be computationally expensive. Additionally, the choice of K can have a significant impact on the imputation results, and selecting the optimal value of K can be a challenging task.

  Here are the steps that KNNImputer follows to fill in missing values:

  1. Calculate the distance metric between the data points. Euclidean distance is the most commonly used distance metric in KNNImputer.
  2. Select the value of K, which represents the number of nearest neighbours to use in the imputation process.
  3. Find the K closest neighbours to the missing data point based on the distance metric.

4. For continuous variables, calculate the mean value of the neighbours and use it to fill in the missing value. For categorical variables, calculate the most common value of the neighbours and use it to fill in the missing value.
5. Repeat the process for all missing values in the dataset.

KNNImputer is particularly useful when the missing data is related to the values of the other features in the dataset. By using the values of the nearest neighbours to impute the missing value, KNNImputer can provide better estimates than other imputation techniques that use simple mean or median values.

Handling Categorical Data:

Various way to handle the Categorical Data:

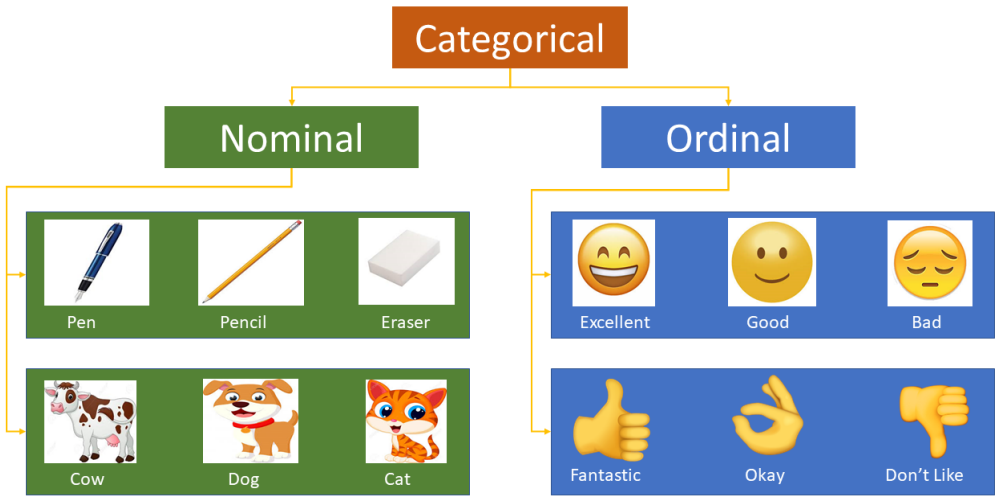
After handle missing values in the dataset, the next step was to handle categorical data.

Most of the Machine learning algorithms cannot handle categorical variables unless we convert them to numerical values. Many algorithm's performances vary based on how Categorical variables are encoded.

Categorical variables can be divided into two categories:

1. **Nominal Data:** The nominal data called labelled/named data. Allowed to change the order of categories, change in order doesn't affect its value. For example, Gender (Male/Female/Other), Age Groups (Young/Adult/Old), etc.

2. **Ordinal Data:** Represent discretely and ordered units. Same as nominal data but have ordered/rank. Not allowed to change the order of categories. For example, Ranks: 1st/2nd/3rd, Education: (High School/Undergrads/Postgrads/Doctorate), etc.



There are many ways we can encode these categorical variables as numbers and use them in an algorithm.

Types of Encoding Techniques:

1. **For Nominal Categorical Variables:**
    a. One Hot Encoding,
    b. One Hot Encoding with Many Categories,
    c. Mean Encoding,

2. **For Odinal Categorical Variables:**
    a. Label Encoding,
    b. Target Guided Odinal Encoding,

One Hot Encoding:

In this method, we map each category to a vector that contains 1 and 0, denoting the presence or absence of the feature. The number of vectors depends on the number of categories for features. This method produces many columns that slow down the learning significantly if the number of the category is very high for the feature. Pandas has **get_dummies** function, which is quite easy to use.

Dummy Variable Trap: Suppose you've 5 features (columns), instead of creating 5 dummies you'll create 4 dummies (features/columns), it's called dummy variable trap mechanism.

Advantages:

- Easy to use and fast way to handle categorical column values.

Disadvantages:

- get_dummies method is not useful when data have many categorical columns.
- If the category column has many categories leads to add many features into the dataset. (Suppose you have 500 columns the using onehot encoding techniques you are created 499 columns, that means you increase dimension of your dataset and it leads to **Curse of Dimensionality**.

One Hot Encoding with Many Categories:

In this encoding techniques you take top most frequents categories of a features. **(KDD)**
Suppose, you have 'fea1' feature, and 50 categories present in this feature (e.g.: {'ab':10, 'cc':43, 'df':22, 'df':9, 'sk':90, ...). And you see that there are 10 categories occurs most frequently (e.g.: 'cc':43, 'gf':34, ...).

So, then you take this top most categories (here it 10) and apply one hot encoding.

Mean Encoding:

we replace category with the mean value with respect to target column.

Advantages:

- Capture information within labels or categories, rendering more predictive features.
- Create a monotonous relationship between the independent variable and the target variable.

Disadvantages:

- May leads to overfit the model, to overcome this problem cross-validation is use most of the time.

Label Encoding:

In this encoding, each category is assigned a value from 1 through N (where N is the number of categories for the feature. One major issue with this approach is there is no relation or order between these classes, but the algorithm might consider them as some order or some relationship.

Target Guided Odinal Encoding:

Here, the category of the column has been replaced with its depending join probability ranking with respect to Target column.

Suppose, you have two columns (fea1 and output), based on each and every category, I'll try to find out the mean of this particular categories. Once you have these mean, now you have to assign the rank on this categories and replace them.

Advantages:

- It doesn't affect the volume of the data i.e., not add any extra features.
- Helps the machine learning model to learn faster.

Disadvantages:

- Typically, mean or joint probability encoding leads for over-fitting.
- Hence, to avoid overfitting cross-validation or some other approach is required most of the time.

Count Frequency Encoding:

Replace each category with its frequency/number of time that category occurred in that column.

Advantage:

- East to implement.
- Not increasing any extra features.

Disadvantage:

- Not able to handle the same number of categories i.e., provide the same values to both categories.

Probability Ratio Encoding:

Here category of the column is replaced with a probability ratio with respect to Target variable.

Example with respect to titanic dataset:

- p1 = take the probability of survived based on the cabin features.
- p2 = take the probability of not survived based on the cabin features.
- ratio = p1(survived) / p2(not survived)
- replace with it categorical features.

Advantages:

- Not increase any extra feature.
- Captures information within the labels or category hence creates more predictive features.
- Creates a monotonic relationship between the variables and the target. So it's suitable for linear models.

Disadvantages:

- Not defined when the denominator is 0.
- Same as the above two methods lead to overfitting to avoid and validate usually cross-validation has been performed.

Binary Encoding:

In this technique, each category in a categorical variable is represented by a binary string. For example, 'Red' can be represented by the binary string '001', 'Green' can be represented by '010', and 'Blue' can be represented by '100'. This technique is useful when the number of categories is high.

Hashing Encoding:

This technique converts a categorical variable into a fixed-length vector. Each category is hashed into a random number between 0 and the length of the vector. This technique is useful when the number of categories is very high.

Leave-One-Out Encoding:

This technique is similar to Target encoding, but it is less prone to overfitting. In this technique, each category in a categorical variable is replaced with the mean of the target variable for that category, leaving out the current sample.

Features Transformation & Scaling:

Introduction:

In my machine learning journey, more often than not, I have found that feature pre-processing is a more effective technique in improving my evaluation metric than any other step, like choosing a model algorithm, hyperparameter tuning, etc.

Feature pre-processing is one of the most crucial steps in building a Machine learning model. Too few features and your model won't have much to learn from. Too many features and we might be feeding unnecessary information to the model. Not only this, but the values in each of the features need to be considered as well.

We know that there are some set rules of dealing with categorical data, as in, encoding them in different ways. However, a large chunk of the process involves dealing with continuous variables. There are various methods of dealing with continuous variables. Some of them include converting them to a normal distribution or converting them to categorical variables, etc.

These techniques are:

- **Feature Transformation and**
- **Feature Scaling.**

Why need Features Transformation & Scaling:

Oftentimes, we have datasets in which different columns have different units – like one column can be in kilograms, while another column can be in centimetres. Furthermore, we can have columns like income which can range from 20,000 to 100,000, and even more; while an age column which can range from 0 to 100(at the most). Thus, Income is about 1,000 times larger than age.

But how can we be sure that the model treats both these variables equally? When we feed these features to the model as is, there is every chance that the income will influence the result more due to its larger value. But this doesn't necessarily mean it is more important as a predictor. So, to give importance to both Age, and Income, we need feature scaling.

In most examples of machine learning models, you would have observed either the Standard Scaler or MinMax Scaler. However, the powerful sklearn library offers many other feature transformations scaling techniques as well, which we can leverage depending on the data we are dealing with.

Data Transformation Techniques:

- **MinMax Scaler (Normalization):**
  The MinMax scaler is one of the simplest scalers to understand. It just scales all the data between 0 and 1. The formula for calculating the scaled value is-

  x_scaled = (x – x_min) / (x_max – x_min)

  Thus, a point to note is that it does so for every feature separately. Though (0, 1) is the default range, we can define our range of max and min values as well.

- **Standarization or Standard Scaler:**
  Standarization is a scaler technique where the values are centred around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.
  mean = 0 standard deviation = 1

  x_scaled = x – (mean / std_dev)

- **MaxAbs Scaler:**
  In simplest terms, the MaxAbs scaler takes the absolute maximum value of each column and divides each value in the column by the maximum value.

  Thus, it first takes the absolute value of each value in the column and then takes the maximum value out of those. This operation scales the data between the range **[-1, 1]**.

- **Robust Scaler:**
  If you have noticed in the scalers we used so far, each of them was using values like the mean, maximum and minimum values of the columns. All these values are sensitive to outliers. If there are too many outliers in the data, they will influence the mean and the max value or the min value. Thus, even if we scale this data using the above methods, we cannot guarantee a balanced data with a normal distribution.

  Robust Scaler is a data pre-processing technique used in machine learning to scale numerical data. It is particularly useful when the data has outliers, which can skew the scaling of the data using other methods such as Min-Max Scaler or Standard Scaler.

  The Robust Scaler, as the name suggests is not sensitive to outliers. This scaler-

- removes the median from the data,
- scales the data by the InterQuartile Range (IQR).

Inter-Quartile Range is nothing but the difference between the first and third quartile of the variable. The interquartile range can be defined as-

IQR = Q3 – Q1

Thus, the formula would be:

x_scaled = (x – x_median) / IQR

❖ **Quantile Transformer Scaler:**

One of the most interesting feature transformation techniques that I have used, the **Quantile Transformer Scaler converts the variable distribution to a normal distribution.** and scales it accordingly. Since it makes the variable normally distributed, it also deals with the outliers. Here are a few important points regarding the Quantile Transformer Scaler:

- It computes the cumulative distribution function of the variable.
- It uses this cdf to map the values to a normal distribution.
- Maps the obtained values to the desired output distribution using the associated quantile function.

Scaling Techniques:

❖ **Log Transformations:**

The Log Transform is one of the most popular Transformation techniques out there. It is primarily used to convert a skewed distribution to a normal distribution/less-skewed distribution. In this transform, we take the log of the values in a column and use these values as the column instead.

Why does it work? It is because the log function is equipped to deal with large numbers. Here is an example-

log (10) = 1
log (100) = 2, and
log (10000) = 4.

❖ **Recipocal Transformations:**

reciVal = (1 / features_value)

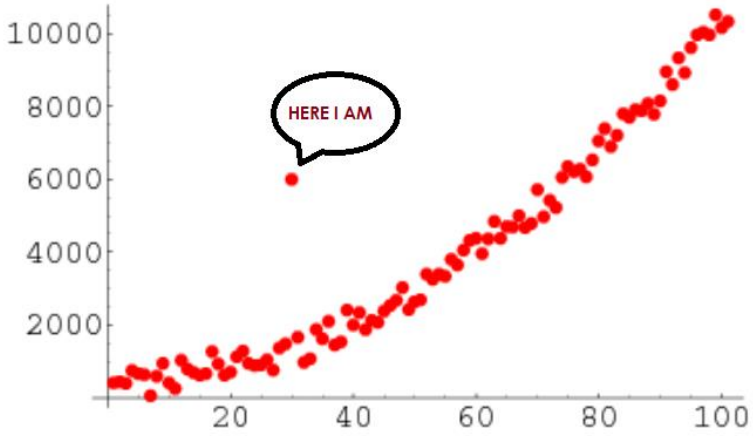
❖ **Square Root Transformation**
❖ **Box-Cox Transformation**
❖ **Others Custom Transformation**

Handling Outliers:

What are Outliers:

In terms of statistics, Outliers can be defined as, **"An Outlier is that observation which is significantly different from all other observations."**

From this definition, we can conclude that an outlier is something that is an **odd-one-out** or the one that is different from the crowd. Some statisticians formally define outliers as **'Observations having a different underlying behaviour than the rest of the observations.'**



An analogy of Outliers in Real-life Examples,

Example-1: In a class, we have 100 students and one student who always scores marks on the higher side concerning other students and its score is not much dependent on the Difficulty level of the exam. So, here we consider that guy as an outlier.

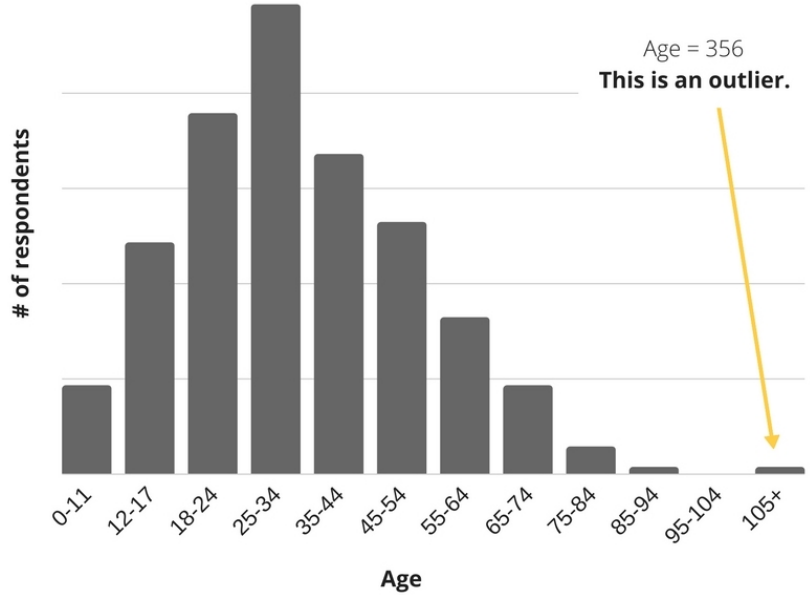
When Outliers dangerous:

Outliers are not always dangerous for our problem statement. In fact, outliers sometimes can be **helpful indicators**.

They represent **errors in the measurement, bad data collection** (not careful while data collection), or simply show those variables that are not considered while collecting the data. Many data analysts are directly tempted to delete outliers. However, this is sometimes the wrong choice for our predictive analysis. One cannot recognize outliers while collecting the data for the problem statement; you won't know what data points are outliers until you begin analyzing the data. Since some of the statistical tests are sensitive to the outliers and therefore, the ability to detect them and treat them accordingly is an important part of data analytics.

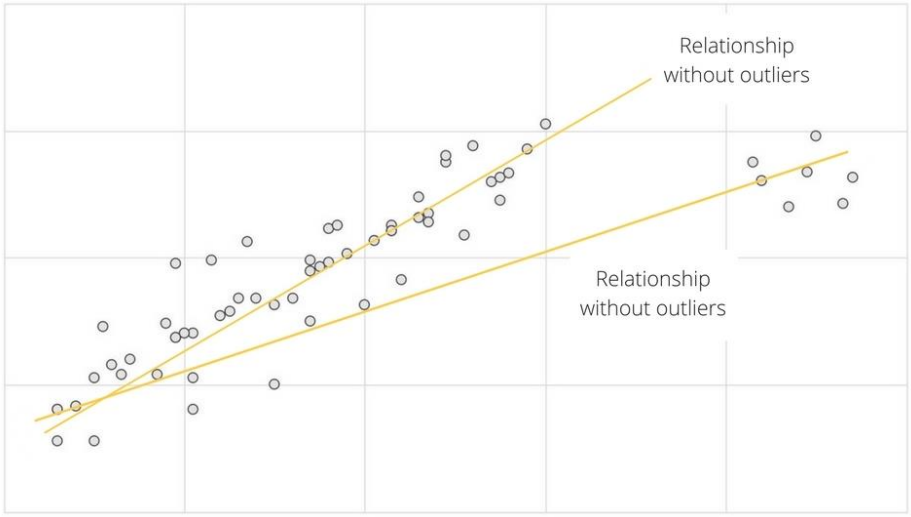
Let's consider the following three different scenarios,

- **Scenario-1:** Let's we have a data of Age for population and the age of a people in that data is 356, and we know that the age value 356 is not possible, so here this data point considered as an outlier and we not know what value we have to replace to this value. So, we have to remove the data point completely from our dataset.



- **Scenario-2:** Let's have a use case of credit card fraud detection, outlier analysis becomes important because here, the exception rather than the rule may be of interest to the analyst.

- **Scenario-3:** Let's have a regression problem, whereas hours of study are the independent variable and marks are a dependent variable. We have some outliers present, so they attract the line of regression to our side. To resolve this, we can create an IQ column then the outlier behaviour may be justified from the IQ column.



Which statistics are affected by the Outliers and not effected by the outliers:

- **Mean:** It is the only measure of central tendency that is always affected by an outlier since it is calculated as the sum of the observed values and then divide by the total number of observations. Since in the expression of mean, the total sum is included, and due to outliers, there are some abnormal values i.e., Outliers will affect this sum.

  Example: Let's outlier is having a bigger positive value than the other values that will make the sum large enough so that the mean will also be slightly larger while if the outlier has a very small value, then

the mean will also become a bit smaller. Hence the presence of outliers in our dataset can largely affect the mean.

- **Standard deviation (SD):** It is calculated with the help of every observation in the data set. It is a sensitive measure because it will be influenced by outliers since standard deviation is calculated by taking the difference of sample case from the mean, outliers will affect Standard deviation.

- **Range:** Most affected by the outliers since it is the difference b/w the max and min value present in the dataset.

When to drop or Keep the Outliers:

I believe that the dropping outlier is always a harsh step and should be taken only in extreme conditions when we're very sure that the **outlier is due to a measurement error**, which we generally do not know while doing analysis.

Sometimes outliers indicate a mistake in data collection. Other times, though, they can influence a data set, so it's important to keep them to better understand the dataset in the big picture.

Below are some examples that give you a clear idea about when you should and shouldn't drop outliers.

- **Drop an outlier if:**

  - **You know that it's completely wrong:**
    For example, if you have a really good sense of how range our data should fall in, like people's ages, which we discussed above in scenario-1, you can safely drop values outside of that range.

  - **You have a lot of data in hand:**
    When you have a lot of data in your hands, then your sample won't be hurt by dropping a questionable outlier.

  - **You have an option to going back:**
    You can go back and recollect and verify the questionable observations.

- **Don't drop an outlier if:**

  - **Your results are critical:**
    When your results are critical, then even minor changes will matter a lot.
    For example, You can feel better about dropping outliers of the dataset in which there are people's favourite TV shows, but not about the temperatures at which airplane seals fail.

  - **There are a lot of outliers:**
    For example, let's 25% of your data be outliers, then it means that something is interesting going on with your data that you need to look further into. You can relate this with scenario-2 which we discussed in the above section.

Detect the Outliers:

- **a. Using Visualization Tools:**

  - Box Plot
  - Scatter Plot

- **b. Discover Outliers with Mathematical Function:**
    - Z-Score
    - IQR Score

Handling/Remove/Correcting the Outliers:

Z-Score:

Well, while calculating the Z-score we re-scale and centre the data and look for data points which are too far from zero. These data points which are way too far from zero will be treated as the outliers. In most of the cases a threshold of 3 or -3 is used i.e., if the **Z-score value is greater than or less than 3 or -3 respectively, that data point will be identified as outliers**.

IQR Score

Box plot use the IQR method to display data and outliers (shape of the data) but in order to be get a list of identified outlier, we will need to use the mathematical formula and retrieve the outlier data. (Use 5 number of summary)

The interquartile range (IQR), also called the midspread or middle 50%, or technically H-spread, is a measure of statistical dispersion, being equal to the difference between 75th and 25th percentiles, or between upper and lower quartiles,

IQR = Q3 – Q1.

In other words, the IQR is the first quartile subtracted from the third quartile; these quartiles can be clearly seen on a box plot on the data.

It is a measure of the dispersion similar to standard deviation or variance, but is much more robust against outliers.

IQR is somewhat similar to Z-score in terms of finding the distribution of data and then keeping some threshold to identify the outlier.

Handling Imbalanced Data:

Introduction:

Classification problems are quite common in the machine learning world. As we know in the classification problem, we try to predict the class label by studying the input data or predictor where the target or output variable is a categorical variable in nature.

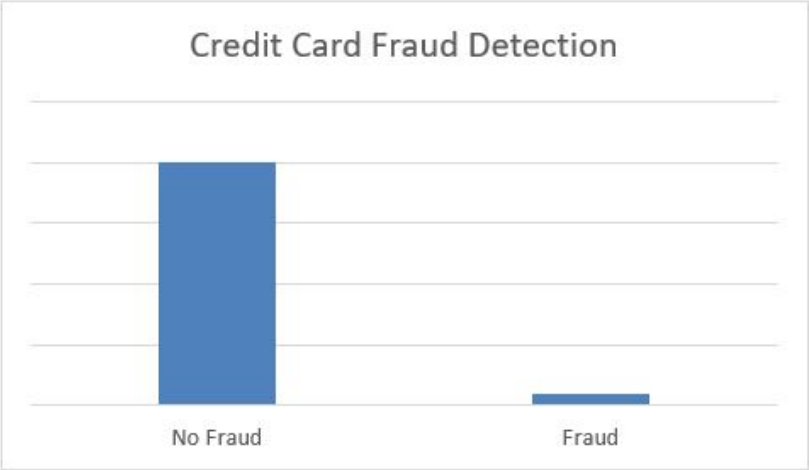
If you have already dealt with classification problems, you must have faced instances where one of the target class labels' numbers of observation is significantly lower than other class labels. This type of dataset is called an imbalanced class dataset which is very common in practical classification scenarios. Any usual approach to solving this kind of machine learning problem often yields inappropriate results.

In this article, I'll discuss the imbalanced dataset, the problem regarding its prediction, and how to deal with such data more efficiently than the traditional approach.

What is Imbalanced Dataset:

Imbalanced data refers to those types of datasets where the target class has an uneven distribution of observations, i.e., **one class label has a very high number of observations and the other has a very low number of observations.** We can better understand it with an example.

Let's assume that XYZ is a bank that issues a credit card to its customers. Now the bank is concerned that some fraudulent transactions are going on and when the bank checks their data, they found that for each 2000 transaction there are only 30 Nos of fraud recorded. So, the number of frauds per 100 transactions is less than 2%, or we can say more than 98% transaction is "No Fraud" in nature. **Here, the class "No Fraud" is called the majority class, and the much smaller in size "Fraud" class is called the minority class.**



Problems with imbalanced data for Classification:

If we explain it in a very simple manner, **the main problem with imbalanced dataset prediction is how accurately are we actually predicting both majority and minority class?**

Generally, imbalanced datasets is create a biased Model (ML or Dl).

Let's explain it with an example of disease diagnosis. Let's assume we are going to predict disease from an existing dataset where for every 100 records only 5 patients are diagnosed with the disease. So, the majority class is 95% with no disease and the minority class is only 5% with the disease. Now, assume our model predicts that all 100 out of 100 patients have no disease.

Sometimes when the records of a certain class are much more than the other class, our classifier may get biased towards the prediction. In this case, the confusion matrix for the classification problem shows how well our model classifies the target classes and we arrive at the accuracy of the model from the confusion matrix. It is calculated based on the total no of correct predictions by the model divided by the total no of predictions. In the above case it is (0+95) / (0+95+0+5) = 0.95 or 95%. It means that the model fails to identify the minority class yet the accuracy score of the model will be 95%.

Thus, our traditional approach of classification and model accuracy calculation is not useful in the case of the imbalanced dataset.

Approaches to deal with the imbalanced dataset problem:

- ❖ **Use or choose the right Evaluation Metrics:**

  Applying inappropriate evaluation metrics for model generated using imbalanced data can be dangerous. Imagine our training data is the one illustrated in graph above. If accuracy is used to measure the goodness of a model, a model which classifies all testing samples into "0" will have an excellent accuracy (99.8%), but obviously, this model won't provide any valuable information for us.

  In this case, other alternative evaluation metrics can be applied such as:

  - Precision/Specificity: how many selected instances are relevant.
  - Recall/Sensitivity: how many relevant instances are selected.
  - F1 score: harmonic mean of precision and recall.
  - MCC: correlation coefficient between the observed and predicted binary classifications.
  - AUC: relation between true-positive rate and false positive rate.

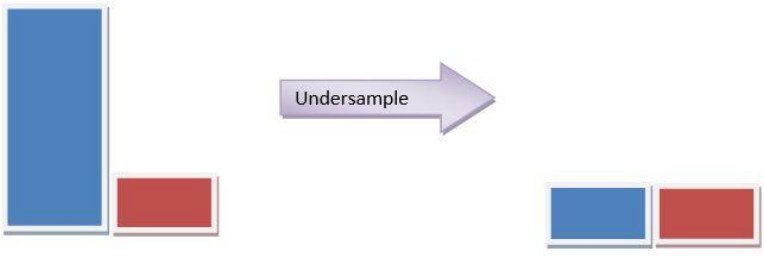
$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

- ❖ **Resampling (OverSampling & UnderSampling):**

  This technique is used to upsample or downsample the minority or majority class.

  UnderSampling or DownSampling:
  We can randomly delete rows from the **majority class** to match them with the minority class which is called undersampling.

  - UnderSampling is used when you have huge records.
  - If you have less number of records then don't apply undersampling, apply oversampling.



  OverSampling or UpSampling:
  When we are using an imbalanced dataset, we can oversample the minority class using replacement. This technique is called oversampling.

  It leads to overfitting, because you repeat the minority class data.

  After sampling the data, we can get a balanced dataset for both majority and minority classes. So, when both classes have a similar number of records present in the dataset, we can assume that the classifier will give equal importance to both classes.

All about OverSampling or UpSampling:

Random Over Sampling:

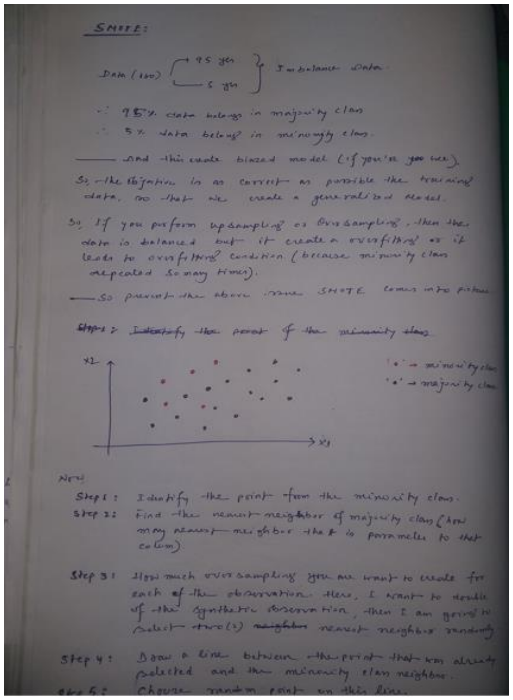
Random oversampling is the simplest oversampling technique to balance the imbalanced nature of the dataset. It balances the data by replicating the minority class samples. This does not cause any loss of information, but **the dataset is prone to overfitting as the same information is copied**.

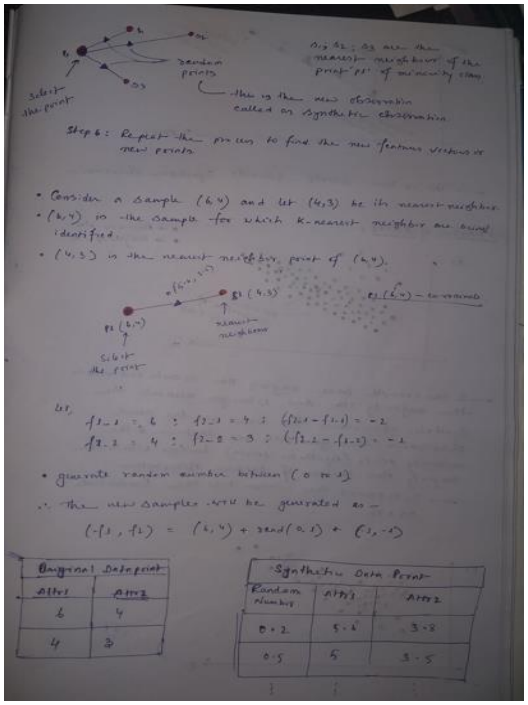
SMOTE (Synthetic Minority Oversampling Technique):

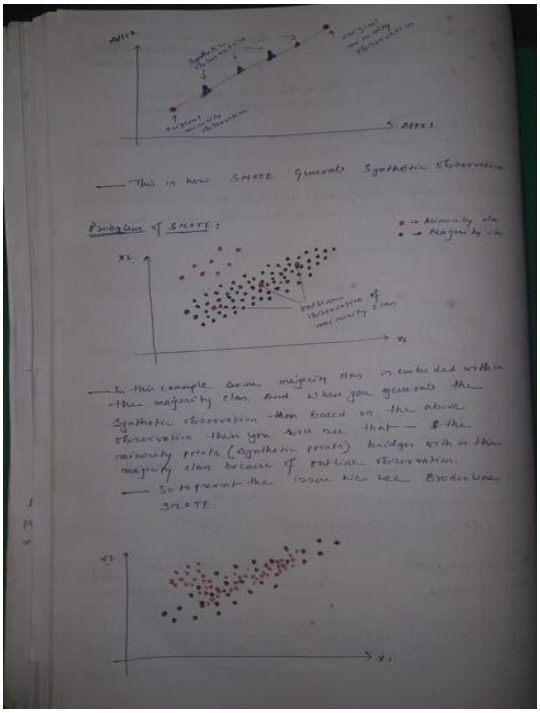
In the case of random oversampling, it was prone to overfitting as the minority class samples are replicated, here SMOTE comes into the picture. SMOTE stands for Synthetic Minority Oversampling Technique. It creates new synthetic samples to balance the dataset.

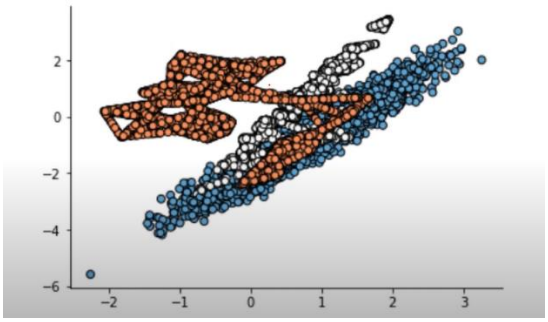
SMOTE works by utilizing a **k-nearest neighbour algorithm** to create synthetic data. Steps samples are created using Smote:

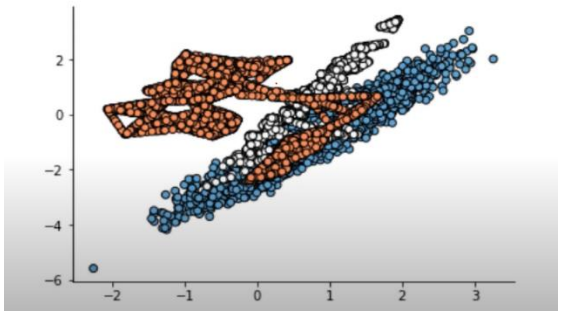
- Identify the feature vector and its nearest neighbours
- Compute the distance between the two sample points
- Multiply the distance with a random number between 0 and 1.
- Identify a new point on the line segment at the computed distance.
- Repeat the process for identified feature vectors.











SMOTE works only for continuous data

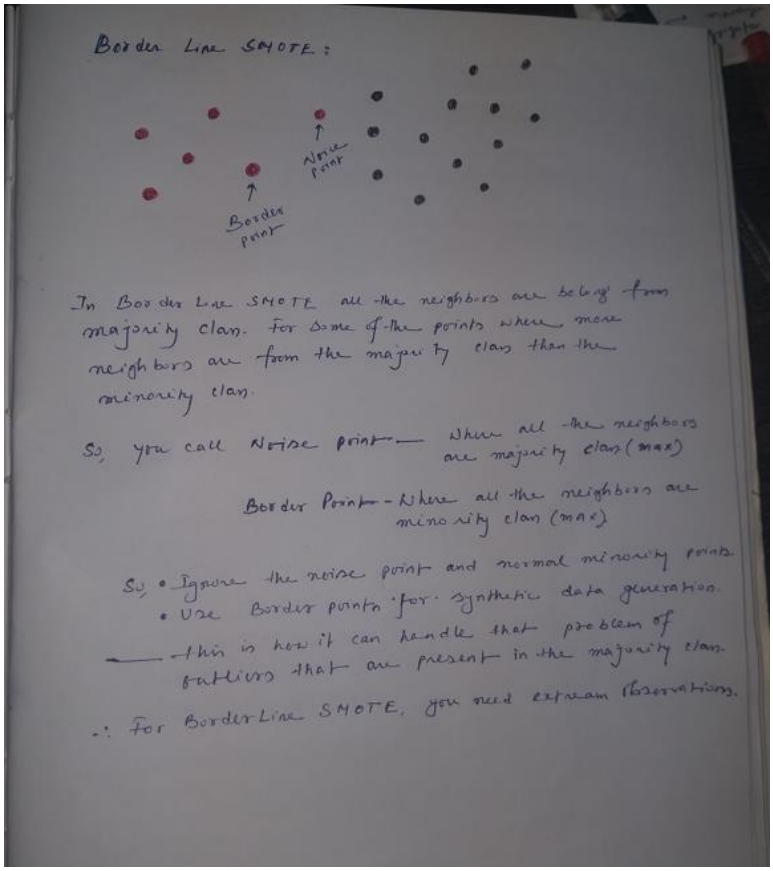
Borderline-SMOTE:

Borderline-SMOTE is a variation of the SMOTE. Just like the name implies, it has something to do with the border.

So, unlike with the SMOTE, where the synthetic data are created randomly between the two data, **Borderline-SMOTE only makes synthetic data along the decision boundary between the two classes.**

Also, there are two kinds of Borderline-SMOTE; there are Borderline-SMOTE1 and Borderline-SMOTE2. The differences are simple; Borderline-SMOTE1 also oversampled the majority class where the majority data are causing misclassification in the decision boundary, while Borderline-SMOTE2 only oversampled the minority classes.

The performance (when apply Broderline SMOTE) doesn't differ much from the model trained with the SMOTE oversampled data. This means that we should focus on the features instead of oversampling the data.



SMOTE-NC:

I have mention that SMOTE only works for continuous features. So, what to do if you have mixed (categorical and continuous) features? In this case, we have another variation of SMOTE called SMOTE-NC (Nominal and Continuous).

You might think, then, just transform the categorical data into numerical; therefore, we had a numerical feature for SMOTE to use. The problem is when we did that; we would have data that did not make any sense. For example, in the churn data above, we had 'IsActiveMember' categorical feature with the data either 0 or 1. If we oversampled this data with SMOTE, we could end up with oversampled data such as 0.67 or 0.5, which does not make sense at all.

This is why we need to use SMOTE-NC when we have cases of mixed data. The premise is simple, we denote which features are categorical, and SMOTE would resample the categorical data instead of creating synthetic data.

Borderline-SMOTE SVM or SVM-SMOTE:

Another variation of Borderline-SMOTE is Borderline-SMOTE SVM, or we could just call it SVM-SMOTE.

The main differences between SVM-SMOTE and the other SMOTE are that instead of using K-nearest neighbours to identify the misclassification in the Borderline-SMOTE, the technique would incorporate the SVM algorithm.

In the SVM-SMOTE, the borderline area is approximated by the support vectors after training SVMs classifier on the original training set. Synthetic data will be randomly created along the lines joining each minority class support vector with a number of its nearest neighbours.

What special about Borderline-SMOTE SVM compared to the Borderline-SMOTE is that more data are synthesized away from the region of class overlap. It focuses more on where the data is separated.

Adaptive Synthetic Sampling (ADASYN):

Borderline Smote gives more importance or creates synthetic points using only the extreme observations that are the border points and ignores the rest of minority class points. This problem is solved by the ADASYN algorithm, as it creates synthetic data according to the data density.

The synthetic data generation is inversely proportional to the density of the minority class. A comparatively larger number of synthetic data is created in regions of a low density of minority class than higher density regions.

In other terms, in the less dense area of the minority class, the synthetic data are created more.

> ADASYN is another variation from SMOTE. ADASYN takes a more different approach compared to the Borderline-SMOTE. While Borderline-SMOTE tries to synthesize the data near the data decision boundary, ADASYN creates synthetic data according to the data density.
>
> The synthetic data generation would be inversely proportional to the density of the minority class. It means more synthetic data are created in regions of the feature space where the density of minority examples is low, and fewer or none where the density is high.
>
> In simpler terms, in an area where the minority class is less dense, the synthetic data are created more. Otherwise, the synthetic data is not made so much.

KMeans SMOTE:

K-Means SMOTE is an oversampling method for class-imbalanced data. It aids classification by generating minority class samples in safe and crucial areas of the input space. **This method avoids the generation of noise and effectively** overcomes imbalances between and within classes.

K-Means SMOTE working steps:

- Cluster the entire data using the k-means clustering algorithm.
- Select clusters that have a high number of minority class samples
- Assign more synthetic samples to clusters where minority class samples are sparsely distributed.