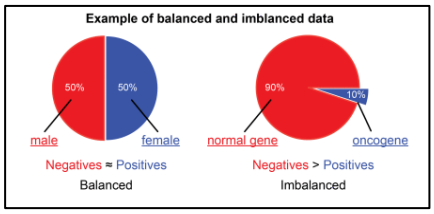
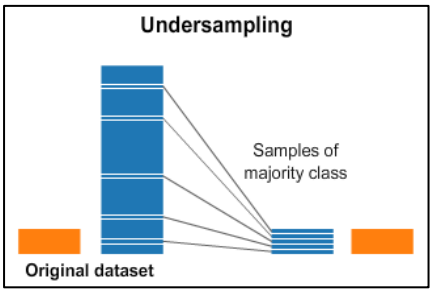
- **Imbalanced vs balanced dataset**

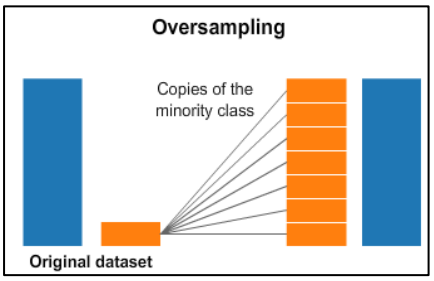


  - If $n1 \sim n2$ it is balanced dataset ($n1: n2 = 50: 50$)
  - If $n1 \ll n2$ it is imbalanced dataset ($n1: n2 = 90: 10$) here the results will be biased towards majority class

- **Methods to handle imbalanced dataset**

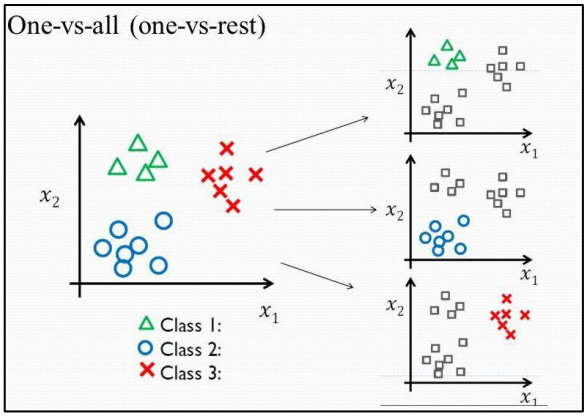


  - **Undersampling**
    - If dataset has n points with minority class having n_1 samples (10%) & majority class having n_2 samples (90%)
    - Here, we copy all the data from minority class into new dataset. And randomly select values from this majority class with count equivalent to n_1.
    - But we are losing a lot of data. In above case we are losing around 80% of data.



  - **Over sampling**
    - If dataset has n points with minority class having n_1 samples (10%) & majority class having n_2 samples (90%)
    - Here, we copy all the data from majority class into new dataset. And copy values from minority class multiple times so that count n_1.

- **Multi-class classification (one vs Rest)**

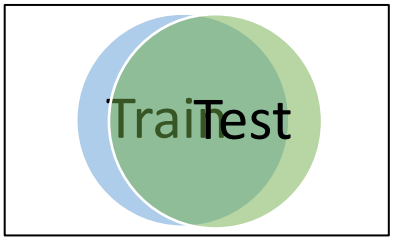


  - Multiclass problems involve classifying x_q into one of N classes
  - For c classes we will do One vs rest classification c times such that each time we will check How many points belong to 1st class and how many points belongs to rest of classes.
  - For the ML algorithms which supports Multi-class classification, there is no need of 1 vs rest e.g. KNN
  - But algorithms like Logistic Regression, SVM, etc are designed to solve binary classification problems. So, we need to use one vs rest

- **Train and test set differences**
  - When the training & test data follows different distribution, it gives low cv error & high-test error
  - To check whether train & test data has different distributions, we create another dataset $D_n{}'$ from original D_n dataset where we concatenate x & y
  - In $D_n{}'$ Dtrain is represented by a class label 1 and Dtest is represented by a class label 0.

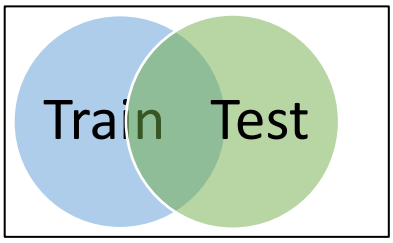
$$D_n = \{(x_i\ y_i)_1^n \nearrow\ x_i \epsilon \mathbb{R}^d\ , y_i \epsilon \{0,1\}\}$$
$$D_n{}' = \{(x_i{}'\ y_i{}')_1^n \nearrow\ x_i \epsilon\ Concatenate(x_i\ y_i)\ , y_i \epsilon \{0,1\}\}$$

  - We then ask the classifier to separate the points in two classes.
  - Case1- Train & Test almost overlapping



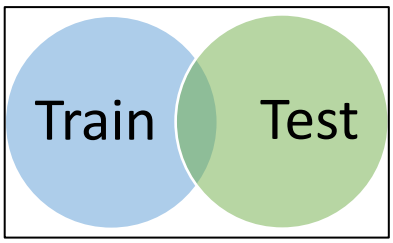
    - Classifier accuracy is low
    - Distributions are very similar

- Case2- Train & Test less overlapping



  - Classifier accuracy is medium
  - Distributions are less similar

- Case2- Train & Test less overlapping

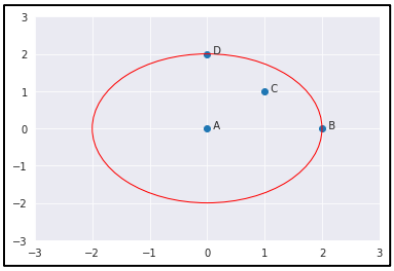


  - Classifier accuracy is high
  - Distributions are very different

- If distributions of Train & Test get changed over time then there will not be perfect overlap of +ve and -ve points of train and test data. there might be some overlap. In case of classifier we want them to perfectly overlap.

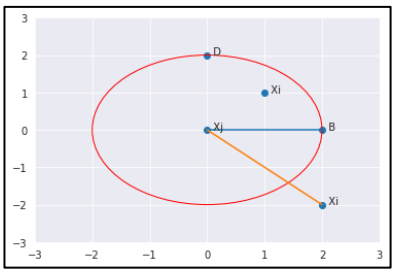
- **Impact of outliers**
  - Outliers are points that are distant from remaining observations. They can potentially bias any analysis performed on the dataset.
  - It is very important to detect and adequately deal with outliers.
  - ML algorithms are sensitive to the range and distribution of attribute values.
  - Data outliers can spoil and mislead the training process resulting in longer training times, less accurate models and ultimately poorer results.

- **K distance & Neighbourhood**



  - K-distance is the distance between the point, and it's K^{th} nearest neighbour.
  - K-neighbours $N_k(A)$ includes a set of points that lie in or on the circle of radius K-distance.
  - K-neighbours can be more than or equal to the value of K. How's this possible?
  - if K =2, $N_k(A) = C, B, D, ||N_2(A)|| = 3$
  - $\left|\left|N_k(point)\right|\right| \geq K$

- **Reachability-Distance**

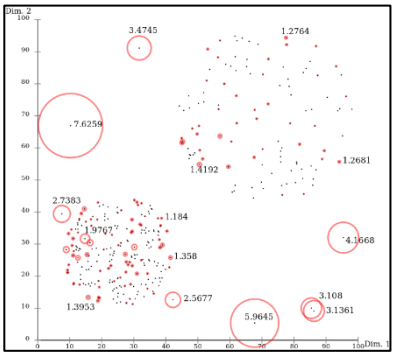


  - It is defined as the maximum of K-distance of x_j and the distance between x_i and x_j
  - The distance measure is problem-specific (Euclidean, Manhattan, etc.)
  - $\boxed{reachability\ dist(x_i, x_j) = \max\left(k_{dist}(x_j), dist(x_i, x_j)\right)}$
  - if x_i lies within the K-neighbors of x_j, the $RD = k_{dist}(x_j)$
  - else $RD = dist(x_i, x_j)$

- **Local reachability-density(A)**
  - LRD is an inverse of average reachability distance of point x_i from all its nearby points
  - More the average reachability distance (i.e., neighbours are far from the point), less density of points are present around a particular point.
  - Low values of LRD imply that the closest cluster is far from the point.

$$\boxed{LRD_K(x_i) = \left[\frac{\left|\left|N_k(x_i)\right|\right|}{\sum_{x_j \epsilon N_k(x_i)}\{reachability\ dist(x_i, x_j)\}}\right]}$$

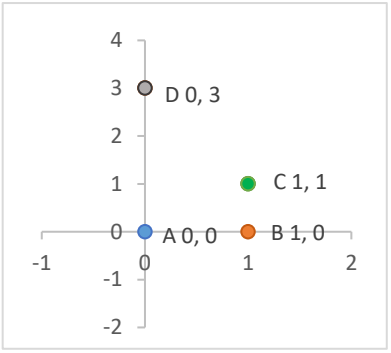
- **Local outlier factor**
  - LOF is the ratio of the average LRD of the K neighbours of x_i to the LRD of x_i.



$$LOF_K(x_i) = \left[\frac{\sum_{x_j \epsilon N_k(x_i)}\{LRD_K(x_j)\}}{||N_k(x_i)|| * LRD_K(x_i)}\right]$$

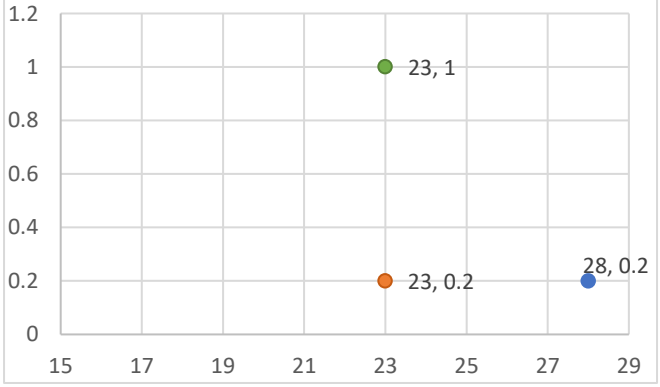
  - If the LRD of a point is much smaller than the LRD of its neighbours ($LOF \gg 1$), the point is far from dense areas and, hence, an outlier.
  - If the LRD of a point is similar to the LRD of its neighbours ($LOF \approx 1$), the point is close to dense areas and, hence, an inlier.
  - LOF values identify an outlier based on the local neighbourhood. Since there is no threshold value of LOF, the selection of a point as an outlier is user-dependent.

  - **Example**- 4 points: $A(0,0), B(1,0), C(1,1)$ *and* $D(0,3)$ and K=2. We will use LOF to detect one outlier among these 4 points.



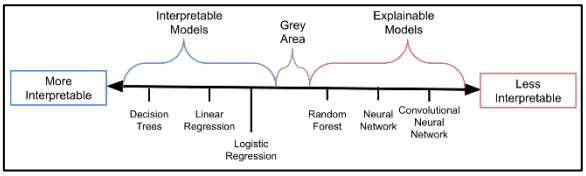
  - **Solution**- First, calculate the K-distance, distance between each pair of points, and K-neighbourhood of all the points with K=2. We will be using Manhattan distance as a measure of distance.
    - $K_{neighbourhood}(A) = \{B, C\}, ||N2(A)||_k = 2$
    - $K_{neighbourhood}(B) = \{A, C\}, ||N2(B)||_k = 2$
    - $K_{neighbourhood}(C) = \{B, A\}, ||N2(C)||_k = 2$
    - $K_{neighbourhood}(D) = \{A, C\}, ||N2(D)||_k = 2$
    - $LRD_2(A) = \left[\frac{||N_k(A)||}{RD(A,B)+RD(A,C)}\right] = \frac{2}{1+2} = \frac{2}{3} = 0.667$
    - $LRD_2(B) = \left[\frac{||N_k(B)||}{RD(B,A)+RD(B,C)}\right] = \frac{2}{2+2} = \frac{2}{4} = 0.500$
    - $LRD_2(C) = \left[\frac{||N_k(C)||}{RD(C,B)+RD(C,A)}\right] = \frac{2}{1+2} = \frac{2}{3} = 0.667$
    - $LRD_2(D) = \left[\frac{||N_k(D)||}{RD(D,A)+RD(D,C)}\right] = \frac{2}{3+3} = \frac{2}{6} = 0.334$
    - $LOF_2(A) = \left[\frac{LRD_2(B)+LRD_2(C)}{||N_k(A)||*LRD_2(A)}\right] = \frac{0.5+0.667}{2*0.667} = 0.874$
    - $LOF_2(B) = \left[\frac{LRD_2(A)+LRD_2(C)}{||N_k(B)||*LRD_2(B)}\right] = \frac{0667+0.667}{2*0.5} = 1.334$
    - $LOF_2(C) = \left[\frac{LRD_2(B)+LRD_2(A)}{||N_k(C)||*LRD_2(C)}\right] = \frac{0.5+0.667}{2*0.667} = 0.874$
    - $LOF_2(D) = \left[\frac{LRD_2(A)+LRD_2(C)}{||N_k(D)||*LRD_2(D)}\right] = \frac{0.667+0.667}{2*0.334} = 2$
    - Highest LOF among the four points is $LOF_2(D)$. Therefore, D is an outlier.

- **Impact of point scale on Euclidean distance**
  - Suppose we have dataset with having two features. Here f1 has features ranging from 0 to 100 & f2 has features from 0 to 1.
  - $Eucl.D(x_1, x_3) = \sqrt{(23-23)^2 - (1-0.2)^2} = 0.8$
    $Eucl.D(x_2, x_3) = \sqrt{(28-23)^2 - (0.2-0.2)^2} = 5$
  - From magnitude it may seem like distance between x_1 and x_3 is more but actually it's less when compared to x_2 and x_3
  - Euclidean distance is impacted by scale. column standardization should be applied before computing distances

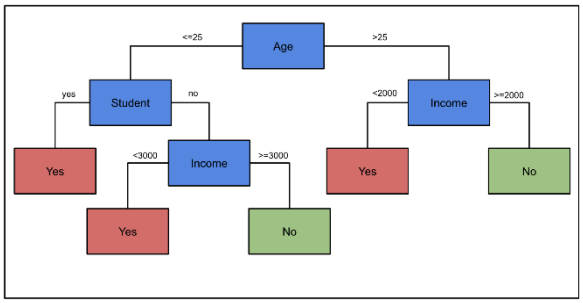


	f1	f2
x_1	23	0.2
x_2	28	0.2
x_3	23	1.0

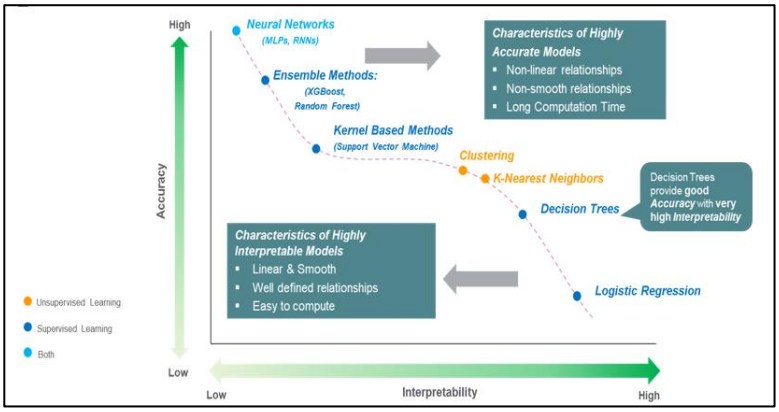
- **Explainable & Interpretable models - Reference**



  - An interpretable model or White-box models can be understood by a human without any other techniques. We can understand how these models make predictions by looking only at the model summary/ parameters.
  - We could also say that an interpretable model provides its own explanation. E.g. linear regression, decision tree
  - An explainable model does not provide its own explanation. these models are too complicated to be understood by humans and they require additional techniques to understand how they make predictions.
  - Interpretation is the process wherein we try to understand the predictions of a ML model.
  - E.g.- we have a decision tree that predicts if someone would default (Yes) or not default (No) on a car loan.
    - a 29-year old with $3000 monthly income makes an application. She was then given a loan by an automated underwriting system based on this model and it predicts that she will not default the loan.
    - Reasons for prediction- as the person is over 25 and her income $>= 2000$, the model predicts that the student will not default on a loan.

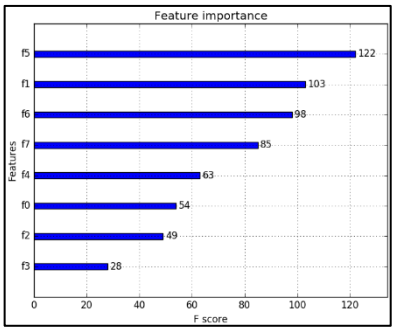


    - E.g., Linear regression- $Y = 100 * \text{age} + 10 * \text{income} + 200$
    - For every year a person ages their predicted maximum loan increases by $100. Similarly, the loan size increases as income increases. For the person (26 years old, $3000 income) the maximum loan size is predicted to be $32800.
  - Above models are simple, the decision tree with few nodes and the linear regression model only has 3 parameters.
  - Hence, it's interpretable but as models become more complicated, we can no longer understand them in this way.
  - ML model is a function where model features are the input and the predictions are the output. An explainable or **black-box** model is a function that is too complicated for a human to understand. We need an additional method/technique to be able to peer into the black-box and understand how the model works.
  - Techniques such as feature importance, LIME, SHAP, DeepLIFT (for neural networks) can be used for understanding **Black Box Model**
  - These techniques can only provide approximations of how the model actually makes predictions. To validate any conclusions, multiple techniques can be used in combination or they could be validated using data visualisations.
  - Domain knowledge can also be an important tool. Any results to the contrary of previous experience/knowledge should be analysed in more detail.



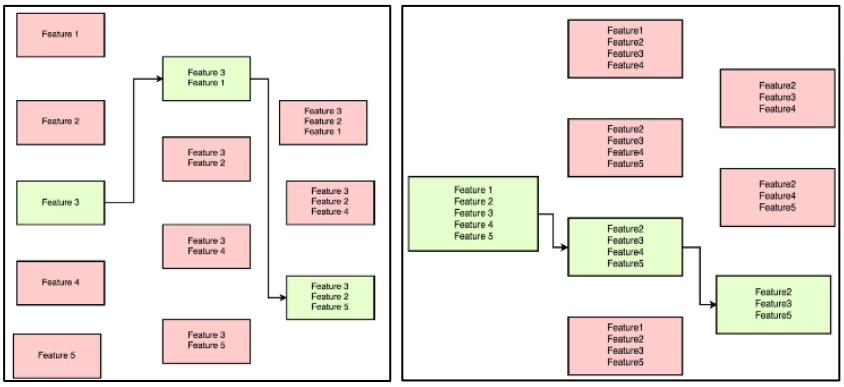
  - So, for higher interpretability, there can be the trade-off of lower accuracy.
  - This is because, in some cases, simpler models can make less accurate predictions but this really depends on the problem you are trying to solve.
  - For instance, you would get poor results using an interpretable model, such as logistic regression, to do image recognition.

- **Feature Importance**



  - It refers to techniques that assign a score to input features based on how useful they are at predicting a target. E.g., correlation scores, coefficients calculated by linear models, decision trees, and permutation importance scores.
  - FI scores provides features which are relevant for target variable. This can be assessed further by domain expert and basis for gathering more or different data.
  - It also provides insight into that specific model and which features are the most important and least important to the model when making a prediction
  - To perform dimensionality reduction, features with low scores can be deleted & features with high scores can be retained

- **Forward Feature selection & Backward Elimination**



  - **Forward Selection** is an iterative method in which we start with having no feature in the model.
  - In each iteration, we add the feature which best improves our model till an addition of a new variable does not improve the performance of the model.
    1. If dataset has d features, train a ML model with only one feature & note down the accuracy. The feature which gives high accuracy that is selected say fs_1
    2. Now at second iteration, retrain the model with pair of fs1 & other features. Select pair which gives high accuracy say fs_2
    3. Repeat these steps up to fs_d and stop when there is no significant improvement in accuracy
  - **Backward Elimination:** we start with all the features and remove the least significant feature at each iteration which improves the performance of the model. We repeat this until no improvement is observed on removal of features.

- **Handling categorical and numerical features**
  - Categorical data are variables that contain label values rather than numeric values. E.g. (Male, Female), (America, India, England)
  - Such label data can't be directly fed to ML algorithm. They need to be encoded as vector.
  - **Integer or Label Encoding**
    - As a first step, each unique category value is assigned an integer value. It is easily reversible.
    - But with numbers, we are inducing an artificial order in the categories. machine learning algorithms may learn this relationship which may result in poor performance or unexpected results

Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

    - E.g., ratings: we can convert that into numbers (V. good, good, Avg, bad, V. bad) = (5, 4, 3, 2, 1). In this case ordinal relation is required to learn
    - E.g., in figure we are indirectly saying chicken is greater than apple & broccoli is greater than chicken. That is absurd.
  - **One-Hot Encoding**

One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

    - In this case, each categorical variable will be represented as vector of size equal to number of unique variables. At place of feature in vector, the value will be 1 for other features it will be 0
    - But due to this dimensionality of the dataset increases and the dataset will be a sparse matrix as in each row there be only 1 non-zero element.

- **Mean-Replacement**
  - The categorical column will be replaced with mean of target variable for respective category

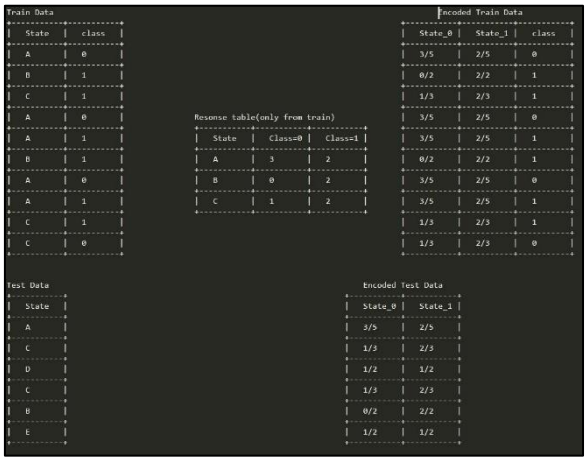
...	State	Score
	California	0.4
	New York	0.1
	Texas	0.9
	New York	0.2
	California	0.5
	Texas	0.8

Avg. California: 0.45
Avg. New York: 0.15
Avg. Texas: 0.85

...	State	Score
	0.45	0.4
	0.15	0.1
	0.85	0.9
	0.15	0.2
	0.45	0.5
	0.85	0.8

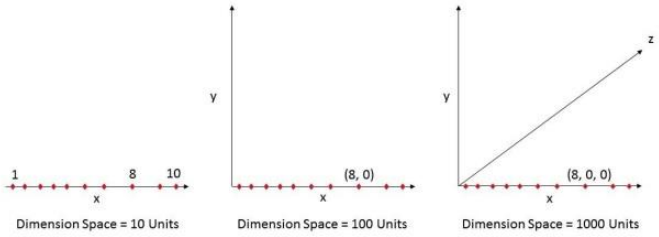
.

- **Using domain knowledge**
  - Example for country we can replace the value with distance from some reference country; or coordinate location on map; say we have some fact that person near equator are tall and person away from equator is short which is stated by a domain expert.
- **Response Encoding**
  - we represent the probability of the data point belonging to a particular class given a category.
  - So, for a K-class classification problem, we get K size vector to represent the categorical variable
    E.g. in train data, State A belongs to 3 data points out of 5 when class label is 0 & for class label 1, it belongs to 2 out of 5 points. So, state A will be replaced with 3/5 & 2/5

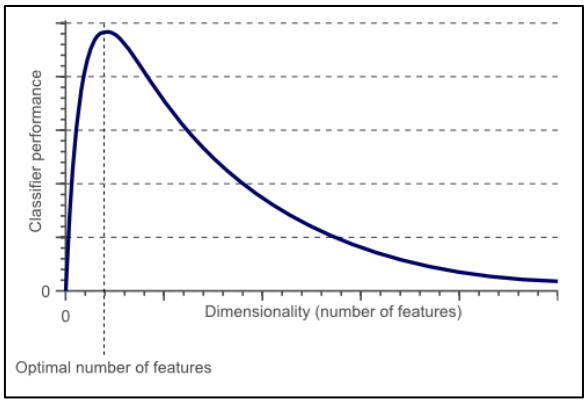


- **Handling missing values by imputation**
  - If dataset have missing values, it causes problems for many ML algorithms. e.g. Nan (Not a Number), 'NULL', -1
  - It is good practice to replace missing values for each column in input data before modelling prediction task. This is called missing data imputation.
  - **Simple Imputation**
    - By taking mean, median or mode of non-missing values
    - In pandas, fillna can be used to replace NA's with a specified value.
  - **Output Based Imputation**
    - By taking mean, median or mode of non-missing values where class is same as missing value.
      E.g. for a feature, 10 missing values are present and out of them 7 belongs to class1
      7 values will be replaced with mean, median or mode of other values in feature where class label is 1.
  - **Imputation along with additional features**
    - In this method, we will impute the values and in addition to them we will also add an additional feature saying that this was a missing feature in original dataset
    - This is being done as in some cases, missing values may represent an information that at this place data collection was not possible
  - **K-nearest neighbour (KNN) imputation**
    - We will divide the dataset such that all the missing values will be part of test data & non-missing values will be part of train data.
    - we will consider missing values as y and will predict it by using KNN

- **Curse of dimensionality**
  - Curse of Dimensionality refers to a set of problems that arise when working with high-dimensional data.
  - As the number of features 'd' grows, the number of datapoints we require to generalize accurately grows exponentially. As the dimensions increase the data becomes sparse and as the data becomes sparse it becomes hard to generalize the model.
  - Fig. shows 10 data points in one dimension i.e. there is only one feature in the data set. It can be easily represented on a line with only 10 values, $x = 1, 2, 3 \ldots 10$.
  - But if we add one more feature, same data will be represented in 2 dimensions causing increase in dimension space to 10*10 =100. And again, if we add 3rd feature, dimension space will increase to 10*10*10 = 1000. As dimensions grows, dimensions space increases exponentially.
    10^1 = 10
    10^2 = 100
    10^3 = 1000 and so on...It is clear that as dimensionality increases the number of data points required to generalize model also increases exponentially



  - **Hughes Phenomenon**
    - It shows that as the number of features increases, the classifier's performance increases till optimal number of features. Adding more features based on the same size as the training set will then degrade the classifier's performance.



  - **Curse of Dimensionality in Distance Function**
  - An increase in the number of dimensions d gives vector of features that represents each observation in the Euclidean space. E.g. for 2D data, it will represent points in 2D space

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i{}^2 - y_i{}^2)}$$

  - Each new dimension adds a non-negative term to the sum, so the distance increases as d increases
  - As the number of features grows for a given number of observations, the feature space becomes sparse
  - KNN is very susceptible to overfitting due to the curse of dimensionality. we can think of even the closest neighbours being too far away in a high-dimensional space to give a good estimate.