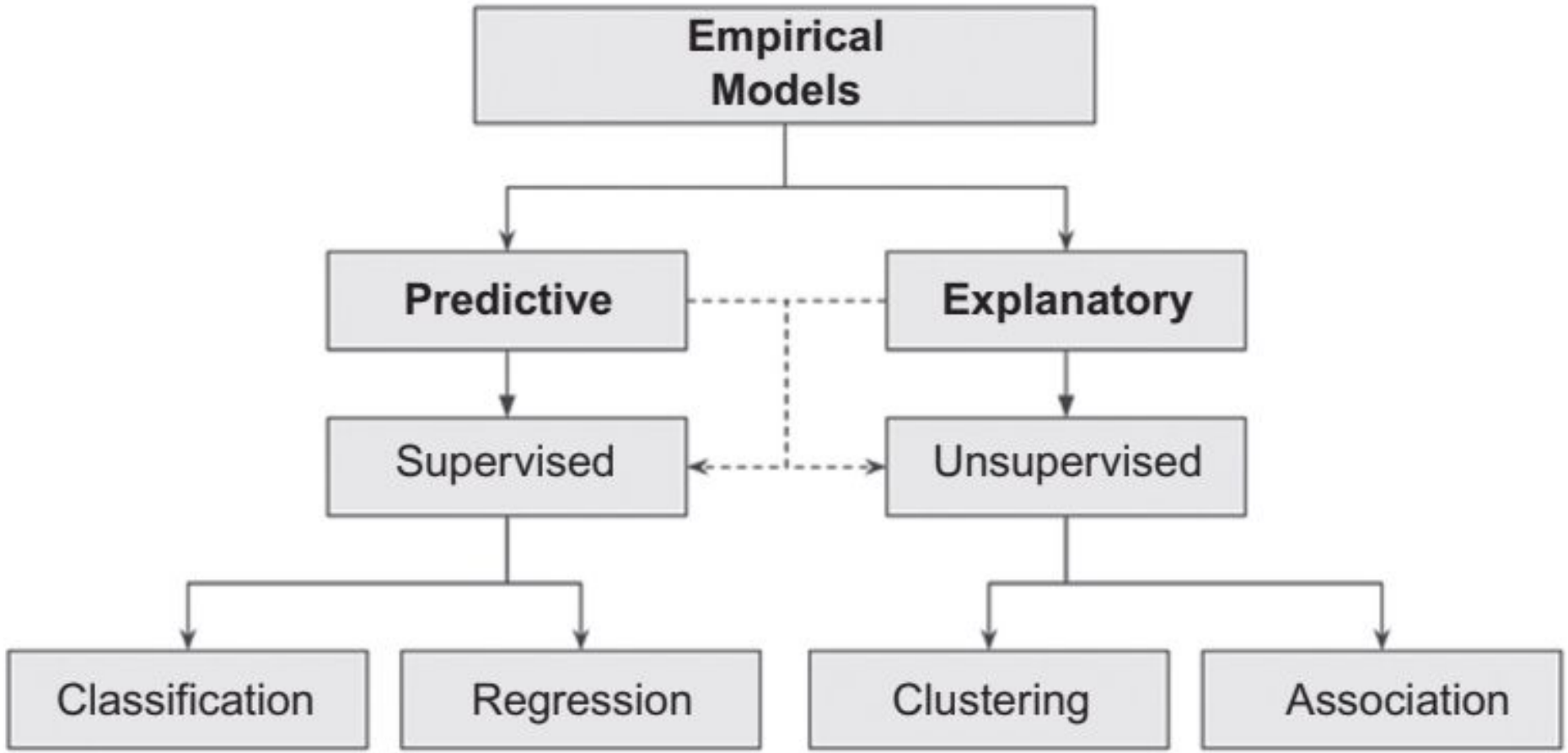
Machine Learning

ML Models

ML Models


Empirical Models
Predictive
Explanatory
Supervised
Unsupervised
Classification
Regression
Clustering
Association

Supervised ML

- There are two major types of supervised machine learning problems:
  - Classification
    - predict discrete labels or categories.
  - Regression
    - predict continuous numerical values.

Classification

- Classification is sometimes separated into
  - *binary classification*
    - special case of distinguishing between exactly two classes.
  - *multiclass classification*
    - classification between more than two classes.

Binary Classification

- You can think of binary classification as trying to answer a yes/no question.
- Classifying emails as either spam or not spam is an example of a binary classification problem.
- In this binary classification task, the yes/no question being asked would be “Is this email spam?”

Multiclass Classification

- The iris example, on the other hand, is an example of a multiclass classification problem.
- Another example is predicting what language a website is in from the text on the website. The classes here would be a pre-defined list of possible languages.

Regression

- For regression tasks, the goal is to predict a continuous number, or a *floating-point number* in programming terms (or *real number* in mathematical terms).
- Predicting a person's annual income from their education, their age, and where they live is an example of a regression task. When predicting income, the predicted value is an *amount*, and can be any number in a given range.
- Another example of a regression task is predicting the yield of a corn farm given attributes such as previous yields, weather, and number of employees working on the farm. The yield again can be an arbitrary number.

Classification vs Regression

- An easy way to distinguish between classification and regression tasks is to ask whether there is some kind of continuity in the output.
- If there is continuity between possible outcomes, then the problem is a regression problem.

Generalization

- In supervised learning,
  - we want to build a model on the training data and
  - then be able to make accurate predictions on new, unseen data that has the same characteristics as the training set that we used.
- If a model is able to make accurate predictions on unseen data, we say it is able to *generalize* from the training set to the test set.
- We want to build a model that is able to generalize as accurately as possible.

Overfitting

- Building a model that is too complex for the amount of information we have is called *overfitting*.
- Overfitting occurs when you fit a model too closely to the particularities of the training set and obtain a model that works well on the training set but is not able to generalize to new data.

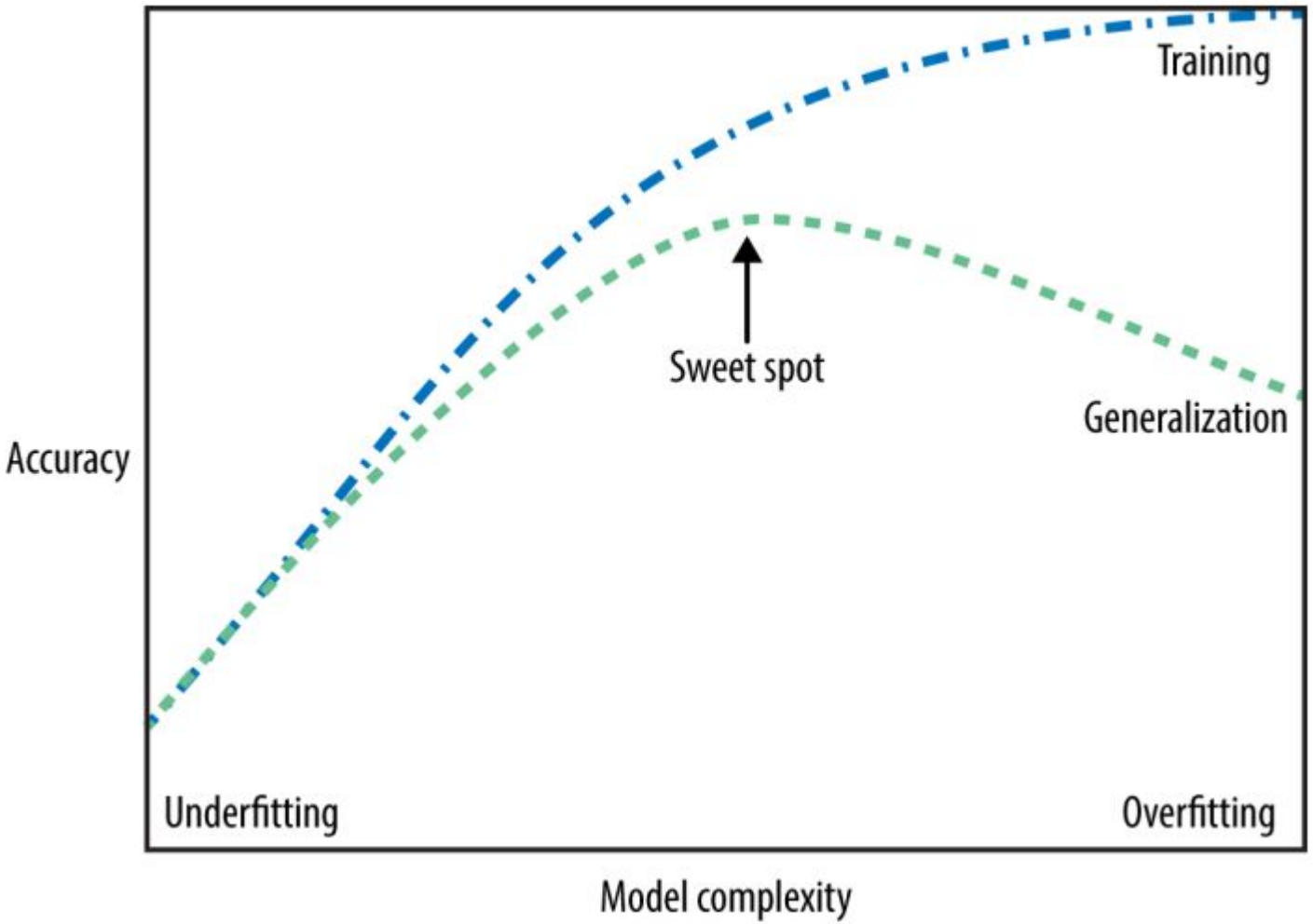
Underfitting

- Choosing too simple a model is called *underfitting.*
- This kind of model might not be able to capture all the aspects of and variability in the data, and your model will do badly even on the training set.

Accuracy vs Model Complexity

- The more complex we allow our model to be, the better we will be able to predict on the training data.
- However, if our model becomes too complex, we start focusing too much on each individual data point in our training set, and the model will not generalize well to new data.
- There is a sweet spot in between that will yield the best generalization performance.
- This is the model we want to find.

Accuracy vs Model Complexity



Relation of Model Complexity to Dataset Size

- Model complexity is intimately tied to the variation of inputs contained in your training dataset:
  - the larger variety of data points your data- set contains, the more complex a model you can use without overfitting.
- Usually, collecting more data points will yield more variety, so larger datasets allow building more complex models.
- However, simply duplicating the same data points or collecting very similar data will not help.

KNN Model

KNN Model

- The *k*-NN algorithm is arguably the simplest machine learning algorithm.
- Building the model consists only of storing the training dataset.
- To make a prediction for a new data point, the algorithm finds the closest data points in the training dataset—its “nearest neighbors”.

KNN Model

- Step-by-Step explanation of how KNN works is discussed below:
- **Step 1: Calculating measure of proximity**
  - To measure the similarity between target and training data points measure of proximity is used. measure of proximity is calculated between each of the data points in the dataset and target point.
- **Step 2: Selecting the optimal value of K**
  - K represents the number of nearest neighbors that needs to be considered while making prediction.

KNN Model

- **Step 3: Finding Nearest Neighbors**
  - The k data points with the smallest distances to the target point are the nearest neighbors.
- **Step 4: Voting for Classification or Taking Average for Regression**
  - In the *classification* problem, the class labels of K-nearest neighbors are determined by performing majority voting. The class with the most occurrences among the neighbors becomes the predicted class for the target data point.
  - In the *regression* problem, the class label is calculated by taking average of the target values of K nearest neighbors. The calculated average value becomes the predicted output for the target data point.

Measure of Proximity

- The effectiveness of the k-NN algorithm hinges on the determination of how similar or dissimilar a test record is when compared with the memorized training record.
- A measure of proximity between two records is a measure of the proximity of its attributes.
- To quantify similarity between two records, there is a range of techniques available such as
  - Distance
  - Weights
  - Correlation similarity
  - Simple matching coefficient
  - Jaccard similarity
  - Cosine similarity

Measure of Proximity

- k-NN models can handle categorical inputs but the distance measure will be either 1 or 0.
- To mitigate the problem caused by different measures and units, all the inputs of k-NN are normalized, where the data values are rescaled to fit a particular range.
- Normalizing all the attributes provides a fair comparison between them.
- Normalization can be performed using a few different methods.

Distance Metrics

- The Euclidean distance is the most common distance measure for numeric attributes.
- Manhattan, and Chebyshev distance measures are sometimes used to calculate the distance between two numeric data points.

Distance Metrics

- The *Euclidean distance* is the root of the sum of the squared differences of all attributes in the dataset.

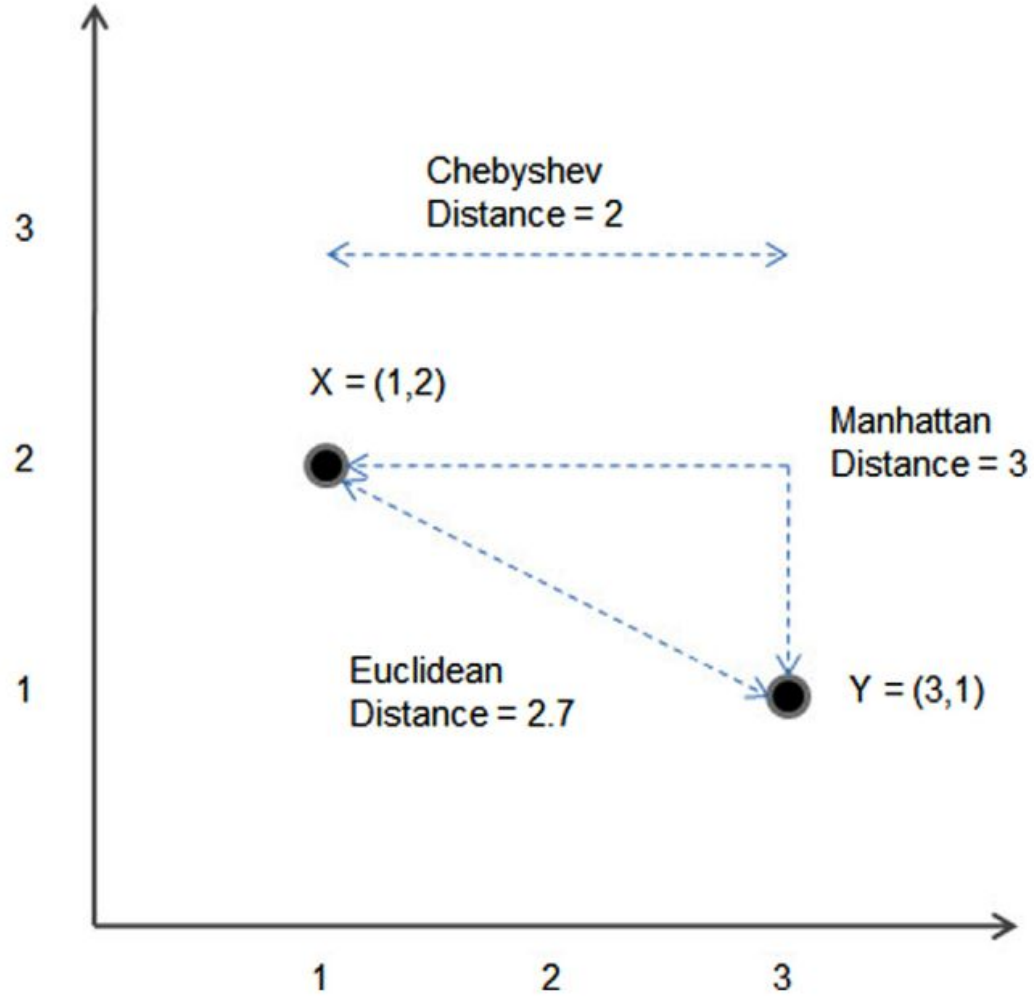
Distance Metrics

- The *Manhattan distance* is the sum of the difference between individual attributes.

Distance Metrics

- The *Chebyshev distance* is the maximum difference between all attributes in the dataset.

Distance Metrics



Distance Metrics

- All three aforementioned distance measures can be further generalized with one formula, the *Minkowski distance* measure.
- The distance between two points $X(x_1, x_2, \dots, x_n)$ and Y$(y_1, y, \dots, y_n)$ in n-dimensional space is given by:

$$d = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

- When $p = 1$, the distance measure is the Manhattan distance,
- when $p = 2$ the distance measure is the Euclidean distance, and
- when $p = \propto$ the distance measure is the Chebyshev distance.

Lazy Learner

- When the KNN algorithm gets the training data, it does not learn and make a model, it just stores the data.
- Instead of finding any discriminative function with the help of the training data, it follows instance-based learning and also uses the training data when it actually needs to do some prediction on the unseen datasets.
- As a result, KNN does not immediately learn a model rather delays the learning thereby being referred to as Lazy Learner.

The value of K

- The value of k is very crucial in the KNN algorithm to define the number of neighbors in the algorithm.
- There is no one proper method of finding the K value in the KNN algorithm. No method is the rule of thumb but you should try the following suggestions:
  - **Square Root Method:**
    - K should be the square root of the number of samples in the training dataset.
    - K should be chosen as the odd so that there are no ties. If the square root is even, then add or subtract 1 to it.
  - **Cross-Validation Method:**
    - Should use cross-validation to find out the optimal value of K in KNN.
    - Start with the minimum value of k **i.e, K=1**, and run cross-validation, measure the accuracy, and keep repeating till the results become consistent.
    - As the value of K increases, the error usually goes down after each one-step increase in K, then stabilizes, and then raises again. Finally, pick the optimum K at the beginning of the stable zone. This technique is also known as the **Elbow Method.**
  - **Domain Knowledge:**
    - Sometimes domain knowledge for a particular use case helps to find the optimum value of K (K should be an odd number).

Sepal Length	Sepal Width	Species
5.3	3.7	Setosa
5.1	3.8	Setosa
7.2	3.0	Virginica
5.4	3.4	Setosa
5.1	3.3	Setosa
5.4	3.9	Setosa
7.4	2.8	Virginica
6.1	2.8	Verscicolor
7.3	2.9	Virginica
6.0	2.7	Verscicolor
5.8	2.8	Virginica
6.3	2.3	Verscicolor
5.1	2.5	Verscicolor
6.3	2.5	Verscicolor
5.5	2.4	Verscicolor

Sepal Length	Sepal Width	Species
5.2	3.1	?

Find Distance

- Distance (Sepal Length, Sepal Width) = $\sqrt{(x-a)^2+(y-b)^2}$
- Distance (Sepal Length, Sepal Width) = $\sqrt{(5.2-5.3)^2+(3.1-3.7)^2}$
- Distance (Sepal Length, Sepal Width) = 0.608

Sepal Length	Sepal Width	Species	Distance
5.3	3.7	Setosa	0.608

Find Distance

Sepal Length	Sepal Width	Species	Distance
5.3	3.7	Setosa	0.608
5.1	3.8	Setosa	0.707
7.2	3.0	Virginica	2.002
5.4	3.4	Setosa	0.36
5.1	3.3	Setosa	0.22
5.4	3.9	Setosa	0.82
7.4	2.8	Virginica	2.22
6.1	2.8	Verscicolor	0.94
7.3	2.9	Virginica	2.1
6.0	2.7	Verscicolor	0.89
5.8	2.8	Virginica	0.67
6.3	2.3	Verscicolor	1.36
5.1	2.5	Verscicolor	0.60
6.3	2.5	Verscicolor	1.25
5.5	2.4	Verscicolor	0.75

Find Rank

Sepal Length	Sepal Width	Species	Distance	Rank
5.3	3.7	Setosa	0.608	3
5.1	3.8	Setosa	0.707	6
7.2	3.0	Virginica	2.002	13
5.4	3.4	Setosa	0.36	2
5.1	3.3	Setosa	0.22	1
5.4	3.9	Setosa	0.82	8
7.4	2.8	Virginica	2.22	15
6.1	2.8	Verscicolor	0.94	10
7.3	2.9	Virginica	2.1	14
6.0	2.7	Verscicolor	0.89	9
5.8	2.8	Virginica	0.67	5
6.3	2.3	Verscicolor	1.36	12
5.1	2.5	Verscicolor	0.60	4
6.3	2.5	Verscicolor	1.25	11
5.5	2.4	Verscicolor	0.75	7

Find Rank

Sepal Length	Sepal Width	Species	Distance	Rank
5.3	3.7	Setosa	0.608	3
5.1	3.8	Setosa	0.707	6
7.2	3.0	Virginica	2.002	13
5.4	3.4	Setosa	0.36	2
5.1	3.3	Setosa	0.22	1
5.4	3.9	Setosa	0.82	8
7.4	2.8	Virginica	2.22	15
6.1	2.8	Verscicolor	0.94	10
7.3	2.9	Virginica	2.1	14
6.0	2.7	Verscicolor	0.89	9
5.8	2.8	Virginica	0.67	5
6.3	2.3	Verscicolor	1.36	12
5.1	2.5	Verscicolor	0.60	4
6.3	2.5	Verscicolor	1.25	11
5.5	2.4	Verscicolor	0.75	7

Find
Nearest Neighbors
k=1: Setosa
k=3: Setosa
k=5: Setosa

Sepal Length	Sepal Width	Species	Distance	Rank
5.3	3.7	Setosa	0.608	3
5.1	3.8	Setosa	0.707	6
7.2	3.0	Virginica	2.002	13
5.4	3.4	Setosa	0.36	2
5.1	3.3	Setosa	0.22	1
5.4	3.9	Setosa	0.82	8
7.4	2.8	Virginica	2.22	15
6.1	2.8	Verscicolor	0.94	10
7.3	2.9	Virginica	2.1	14
6.0	2.7	Verscicolor	0.89	9
5.8	2.8	Virginica	0.67	5
6.3	2.3	Verscicolor	1.36	12
5.1	2.5	Verscicolor	0.60	4
6.3	2.5	Verscicolor	1.25	11
5.5	2.4	Verscicolor	0.75	7

Advantages

- **No Training Period**
  - It stores the training dataset and learns from it only when we use the algorithm for making the real-time predictions on the test dataset.
- **Faster**
  - The KNN algorithm is much faster than other algorithms which require training.
- **Easy to implement and understand**
  - To implement the KNN algorithm, we need only two parameters i.e. the value of K and the measure of proximity.
  - Since both the parameters are easily interpretable therefore they are easy to understand.

Disadvantages

- **Does not work well with large datasets**
  - In large datasets, the cost of calculating the distance between the new point and each existing point is huge which decreases the performance of the algorithm.
- **Does not work well with high dimensions**
  - With the increasing number of dimensions, it becomes difficult to calculate the distance for each dimension.
- **Need feature scaling**
  - We need to do feature scaling (standardization and normalization) on the dataset before feeding it to the KNN algorithm otherwise it may generate wrong predictions.
- **Sensitive to Noise and Outliers**
  - KNN is highly sensitive to the noise present in the dataset and requires manual imputation of the missing values along with outliers removal.