Lecture Notes
on
Machine Learning Concept 1(CSE 3967)

Centre for Data Science
Faculty of Engineering and Technology (ITER)
Siksha 'O' Anusandhan (Deemed to be University)
August 2025

Contents

Chapter 1

Introduction to Machine Learning

1.1 What is Machine Learning?

Machine Learning (ML) is a branch of Artificial Intelligence (AI) that focuses on building systems that can learn from data and improve their performance over time without being explicitly programmed.

1.1.1 How do Machines Learn?

The basic machine learning process consists of three parts:

1. **Data Input**: Past data or information is utilized as a basis for future decision-making.
2. **Abstraction**: Input data is represented in a broader way through the underlying algorithm.
3. **Generalization**: The abstracted representation is generalized to form a framework for making decisions.

Abstraction

During the machine learning process, knowledge is fed in the form of input data. The raw data cannot be used directly. Abstraction helps derive a conceptual map (a model) that summarizes the knowledge representation of the raw data.

A model may take any of the following forms:

- Computational blocks like if/else rules
- Mathematical equations
- Data structures like trees or graphs
- Logical groupings of similar observations

Choice of Model

The model choice depends on several aspects:

- **Type of problem**: forecasting, prediction, trend analysis, segmentation, etc.
- **Nature of data**: completeness, data types, missing values, etc.
- **Domain**: e.g., fraud detection requiring rapid inference.

Once the model is chosen, the next task is to fit the model based on input data.

Generalization

Generalization means tuning the abstracted knowledge so that it can make decisions on unseen (test) data. This is difficult because:

1. The trained model may be too aligned with training data (overfitting).
2. Test data may contain characteristics not present in the training data.

Hence, decision-making often uses approximate or heuristic approaches, similar to human intuition.

1.2 Types of Machine Learning

Machine Learning can be classified into three main types:

1. **Supervised Learning** – Predictive learning.
2. **Unsupervised Learning** – Descriptive learning.
3. **Reinforcement Learning** – Goal-based self-learning.

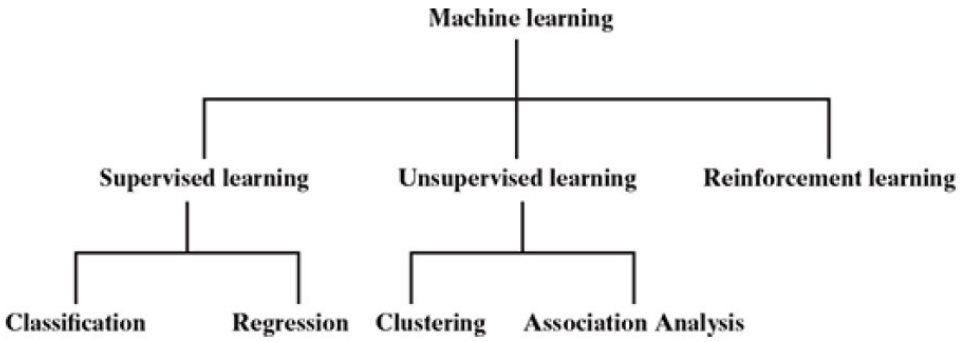


FIG: Types of machine learning

1.2.1 Supervised Learning

Labeled training data is provided. A predictive model is built and used to assign labels to test data.

Examples:

- Predicting game results
- Tumor classification (benign or malignant)
- Price prediction (stocks, real estate)
- Spam vs non-spam email classification

When predicting:

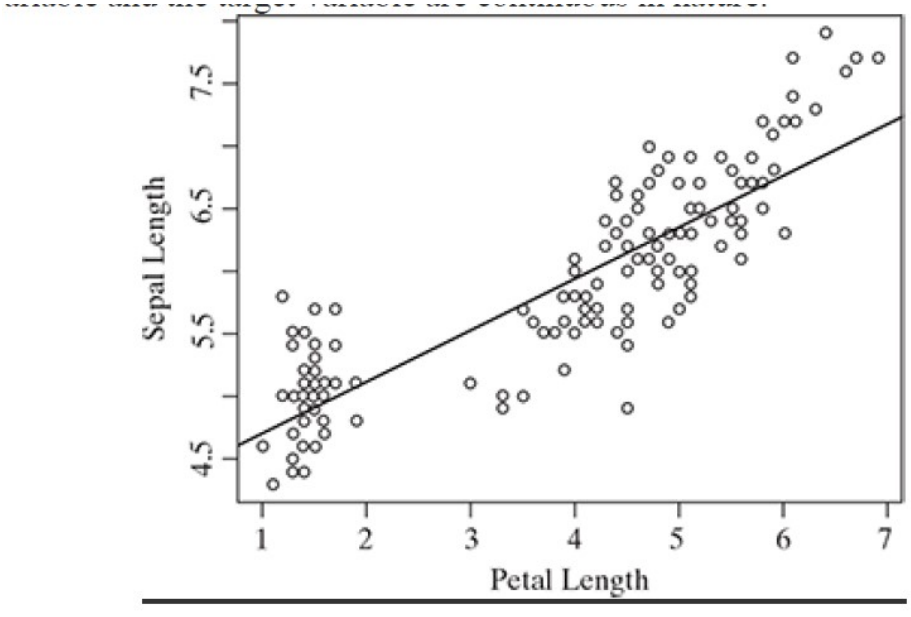
- A **categorical variable** → **Classification**
- A **continuous variable** → **Regression**

Classification

Classification predicts a categorical target feature called a *class*.
Applications:

- Image classification
- Disease prediction
- Win–loss prediction
- Natural calamity prediction
- Handwriting recognition

Regression



In regression, the target is numerical. A typical linear regression model is:

$$y = mx + c$$

where x is the predictor variable and y is the target.

Applications:

- Retail demand forecasting
- Sales prediction
- Real estate price prediction
- Weather forecasting
- Job market skill demand forecasting

1.2.2 Unsupervised Learning

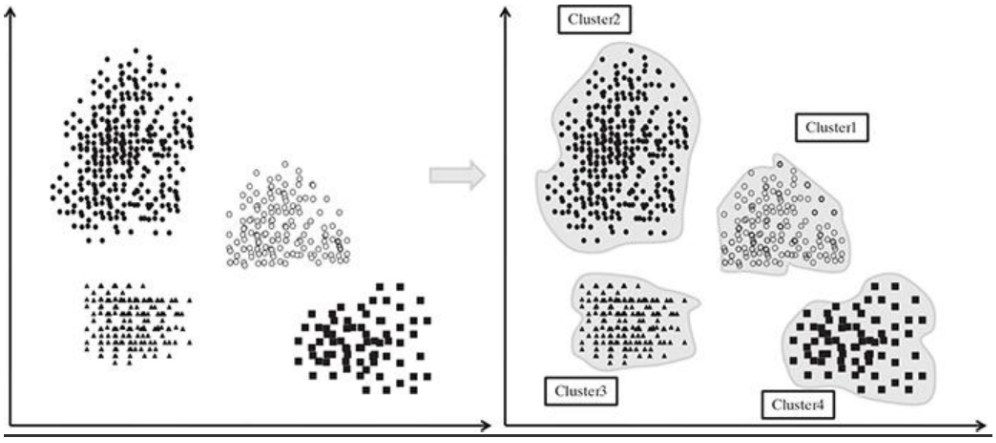
There is no labeled data. The goal is to analyze the dataset and discover natural groupings or patterns.

Key application:

- Customer segmentation

The main type is:

- **Clustering**: Grouping similar objects together.



1.2.3 Reinforcement Learning

An agent learns by interacting with an environment and receiving rewards.

Example: A child learning to walk:

- **Agent**: the child
- **Environment**: ground with obstacles
- **Action**: walking attempts
- **Reward**: successful steps

Modern example: **Self-driving cars** – must consider speed, traffic, road, weather, and make decisions like start/stop, accelerate/decelerate, left/right turns.

1.2.4 Comparison – supervised, unsupervised, and reinforcement learning

SUPERVISED	UNSUPERVISED	REINFORCEMENT
This type of learning is used when you know how to classify a given data, or in other words classes or labels are available.	This type of learning is used when there is no idea about the class or label of a particular data. The model has to find pattern in the data.	This type of learning is used when there is no idea about the class or label of a particular data. The model has to do the classification – it will get rewarded if the classification is correct, else get punished.
Labelled training data is needed. Model is built based on training data.	Any unknown and unlabelled data set is given to the model as input and records are grouped.	The model learns and updates itself through reward/ punishment.
The model performance can be evaluated based on how many misclassifications have been done based on a comparison between predicted and actual values.	Difficult to measure whether the model did something useful or interesting. Homogeneity of records grouped together is the only measure.	Model is evaluated by means of the reward function after it had some time to learn.
There are two types of supervised learning problems – classification and regression.	There are two types of unsupervised learning problems – clustering and association.	No such types.
Simplest one to understand.	More difficult to understand and implement than supervised learning.	Most complex to understand and apply.
Standard algorithms include • Naïve Bayes • *k*-nearest neighbour (kNN) • Decision tree • Linear regression • Logistic regression • Support Vector Machine SVM), etc.	Standard algorithms are • *k*-means • Principal Component Analysis (PCA) • Self-organizing map (SOM) • Apriori algorithm • DBSCAN etc.	Standard algorithms are • Q-learning • Sarsa
Practical applications include • Handwriting recognition • Stock market prediction • Disease prediction • Fraud detection, etc.	Practical applications include • Market basket analysis • Recommender systems • Customer segmentation, etc.	Practical applications include • Self-driving cars • Intelligent robots • AlphaGo Zero (the latest version of DeepMind's AI system playing Go)

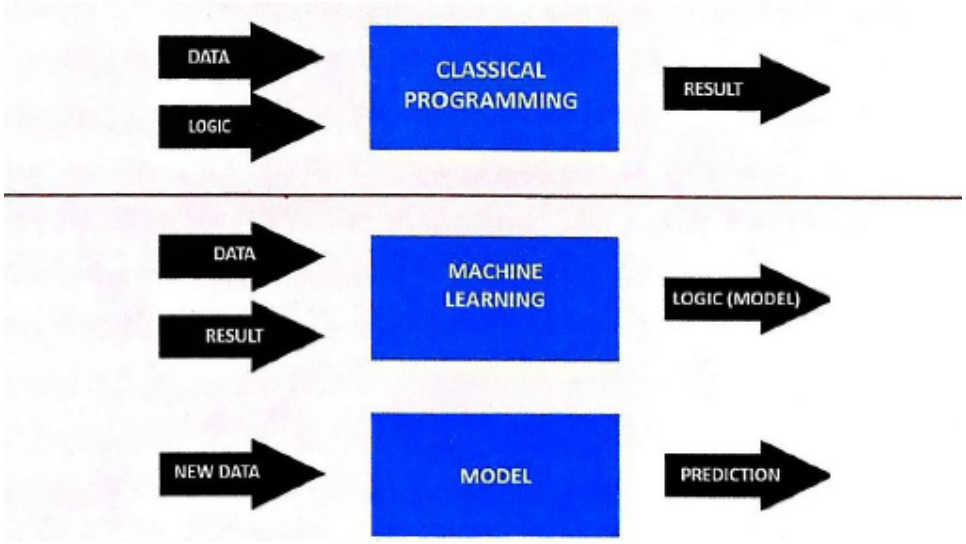
1.3 Applications of Machine Learning

- Banking and Finance
- Insurance
- Healthcare

1.4 Machine Learning Vs Classical Programming

Machine learning algorithms work very much like a child, solving problems very differently from traditional programming methods. They are able to do the following things:

- Work out the general principle by observing the data(learning patterns in the input)
- Handle the probabilistic nature of the outcome
- Strengthen the rules of inference by looking at more examples
- Learn from past errors and improve



1.5 State-of-the-Art Languages and Tools in ML

Although ML algorithms can be implemented in any language (Java, C/C++, .NET), certain languages/tools are widely used:

- Python
- R
- MATLAB
- SAS

Python

Python (created by Guido van Rossum in 1991) is widely used in ML. Powerful libraries include:

- NumPy – mathematical functions
- SciPy – algorithms and advanced mathematics
- Matplotlib – numerical plotting
- Scikit-learn – classification, regression, and clustering algorithms

Chapter 2

Preparing to Model

Objective of the Chapter

This chapter gives a detailed view of how to understand the incoming data and create basic understanding about the nature and quality of the data. This information, in turn, helps to select and then how to apply the model. So, the knowledge imparted in this chapter helps a beginner take the first step towards effective modelling and solving a machine learning problem.

2.1 Machine Learning Activities

The first step in machine learning activity starts with data. In case of supervised learning, it is the labelled training data set followed by test data which is not labelled. In case of unsupervised learning, there is no question of labelled data but the task is to find patterns in the input data. A thorough review and exploration of the data is needed to understand the type of the data, the quality of the data and relationship between the different data elements. Based on that, multiple pre-processing activities may need to be done on the input data before we can go ahead with core machine learning activities. Following are the typical preparation activities done once the input data comes into the machine learning system:

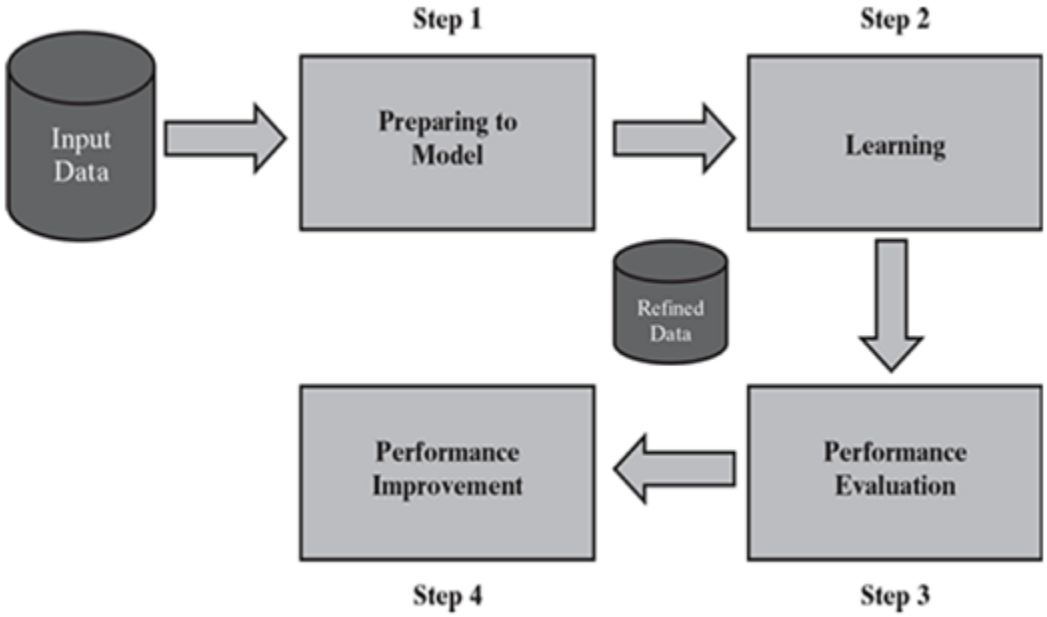
- Understand the type of data in the given input data set.
- Explore the data to understand the nature and quality.
- Explore the relationships amongst the data elements, e.g. inter-feature relationship.
- Find potential issues in data.
- Do the necessary remediation, e.g. impute missing data values, etc., if needed.
- Apply pre-processing steps, as necessary.

Once the data is prepared for modelling, then the learning tasks start off. As a part of it, do the following activities:

- The input data is first divided into parts – the training data and the test data (called holdout). This step is applicable for supervised learning only.
- Consider different models or learning algorithms for selection.
- Train the model based on the training data for supervised learning problem and apply to unknown data. Directly apply the chosen unsupervised model on the input data for unsupervised learning problem.

After the model is selected, trained (for supervised learning), and applied on input data, the performance of the model is evaluated. Based on options available, specific actions can be taken to improve the performance of the model, if possible.

Step #	**Step Name**	**Activities Involved**
Step 1	Preparing to Model	• Understand the type of data in the given input data set • Explore the data to understand data quality • Explore the relationships amongst the data elements, e.g. inter-feature relationship • Find potential issues in data • Remediate data, if needed • Apply following pre-processing steps, as necessary: – Dimensionality reduction – Feature subset selection
Step 2	Learning	• Data partitioning / holdout • Model selection • Cross-validation
Step 3	Performance evaluation	• Examine the model performance, e.g. confusion matrix in case of classification • Visualize performance trade-offs using ROC curves
Step 4	Performance improvement	• Tuning the model • Ensembling • Bagging • Boosting



2.2 Basic Types of Data in Machine Learning

Before starting with types of data, let's first understand what a data set is and what are the elements of a data set. A data set is a collection of related information or records. The information may be on some entity or some subject area. For example, we may have a data set on students in which each record consists of information about a specific student. Again, we can have a data set on student performance which has records providing performance i.e. marks on the individual subjects. Each row of a data set is called a record. Each data set also has multiple attributes, each of which gives information on a specific characteristic. For example, in the data set on students, there are four attributes namely Roll Number, Name, Gender, and Age, each of which understandably is a specific characteristic about the student entity. Attributes can also be termed as feature, variable, dimension or field.

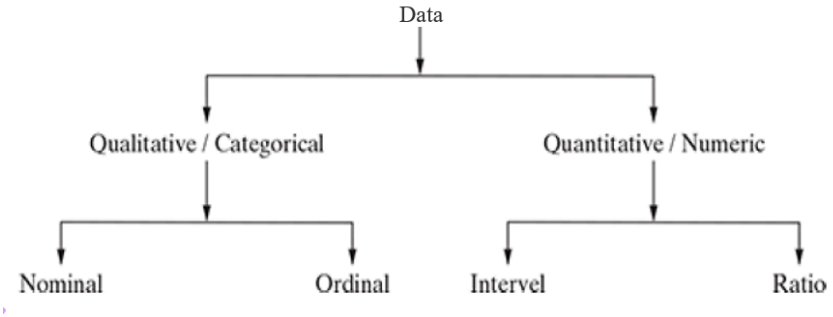
Student data set:

Roll Number	Name	Gender	Age
129/011	Mihir Karmarkar	M	14
129/012	Geeta Iyer	F	15
129/013	Chanda Bose	F	14
129/014	Sreenu Subramanian	M	14
129/015	Pallav Gupta	M	16
129/016	Gajanan Sharma	M	15

Student performance data set:

Roll Number	Maths	Science	Percentage
129/011	89	45	89.33%
129/012	89	47	90.67%
129/013	68	29	64.67%
129/014	83	38	80.67%
129/015	57	23	53.33%
129/016	78	35	75.33%

Now that a context of data sets is given, let's try to understand the different types of data that we generally come across in machine learning problems. Data can broadly be divided into following types:



2.2.1 Qualitative Data

Qualitative data provides information about the quality of an object or information which cannot be measured. For example, if we consider the quality of performance of students in terms of 'Good', 'Average', and 'Poor', it falls under the category of qualitative data. Also, name or roll number of students are information that cannot be measured using some scale of measurement. So they would fall under qualitative data. Qualitative data is also called **categorical data**.

Qualitative data can be further subdivided into two types:

1. Nominal data
2. Ordinal data

Nominal Data

Nominal data is one which has no numeric value, but a named value. It is used for assigning named values to attributes. Nominal values cannot be quantified.

Examples:

- Blood group: A, B, O, AB
- Nationality: Indian, American, British, etc.
- Gender: Male, Female, Other

Ordinal Data

Ordinal data in addition to possessing the properties of nominal data, can also be naturally ordered. This means ordinal data also assigns named values to attributes but unlike nominal data, they can be arranged in a sequence of increasing or decreasing value so that we can say whether a value is better than or greater than another value. Examples of ordinal data are

- Customer satisfaction: Very Happy, Happy, Unhappy, etc.

- Grades: A, B, C, etc.
- Hardness of Metal: Very Hard, Hard, Soft, etc.

As basic counting is possible for nominal and ordinal data, the mode can be identified for both. Since ordering is possible in case of ordinal data, median and quartiles can be identified in addition. Mean can still not be calculated.

2.2.2 Quantitative Data

Quantitative data relates to information about the quantity of an object – hence it can be measured. For example, if we consider the attribute 'marks', it can be measured using a scale of measurement. Quantitative data is also termed as numeric data. There are two types of quantitative data:

1. Interval data
2. Ratio data

Interval Data

Interval data is numeric data for which not only the order is known, but the exact difference between values is also known. An ideal example of interval data is Celsius temperature. The difference between each value remains the same in Celsius temperature. For example, the difference between 12°C and 18°C degrees is measurable and is 6°C as in the case of difference between 15.5°C and 21.5°C. Other examples include date, time, etc. For interval data, mathematical operations such as addition and subtraction are possible. For that reason, for interval data, the central tendency can be measured by mean, median, or mode. Standard deviation can also be calculated.

Ratio Data

Ratio data represents numeric data for which exact value can be measured. Absolute zero is available for ratio data. Also, the variables in can be added, subtracted, multiplied, or divided. The central tendency can be measured by mean, median, or mode and methods of dispersion such as standard deviation. Examples of ratio data include height, weight, age, salary, etc.

2.3 Exploring Structure of Data

By now, we understand that in machine learning, we come across two basic data types – numeric and categorical. With this context in mind, we can delve deeper into understanding a data set. We need to understand that in a data set, which of the attributes are numeric and which are categorical in nature. This is because, the approach of exploring numeric data is different than the approach of exploring categorical data.

2.3.1 Exploring Numerical Data

There are two most effective mathematical plots to explore numerical data – box plot and histogram.

Understanding Central Tendency

To understand the nature of numeric variables, we can apply the measures of central tendency of data, i.e. mean and median.

Mean, by definition, is the sum of all data values divided by the count of data elements. For example, the mean of a set of observations – 21, 89, 34, 67, and 96 is calculated as:

$$\text{Mean} = \frac{21 + 89 + 34 + 67 + 96}{5} = 61.4$$

If the above set of numbers represents marks of 5 students in a class, the mean marks, or the value falling in the middle of the range is 61.4.

Median, on the contrary, is the value of the element appearing in the middle of an ordered list of data elements. If we consider the above 5 data elements, the ordered list would be – 21, 34, 67, 89, and 96. Since there are 5 data elements, the 3rd element in the ordered list is considered as the median. Hence, the median value of this set of data is 67.

If we observe that for certain attributes the deviation between values of mean and median is quite high, we should investigate those attributes further and try to find out the root cause along with the need for remediation.

Understanding Data Spread

Now that we have explored the central tendency of the different numeric attributes, we have a clear idea of which attributes have a large deviation between mean and median. Let us look closely at those attributes.

To drill down further, we need to examine the entire range of values of the attributes, though not at the level of individual data elements as that may be too vast to review manually. Therefore, we take a granular view of the data spread in the form of:

1. Dispersion of data
2. Position of the different data values

Measuring Data Dispersion Consider the data values of two attributes:

1. Attribute 1 values: 44, 46, 48, 45, and 47
2. Attribute 2 values: 34, 46, 59, 39, and 52

Both sets of values have a mean and median of 46. However, the first set of values (Attribute 1) is more concentrated or clustered around the mean/median, whereas the second set of values (Attribute 2) is more dispersed.

To measure the extent of dispersion of data, the variance is calculated using the following formula:

$$\text{Variance}\ (x) = \frac{\sum_{i=1}^{n} x_i^2}{n} - \left(\frac{\sum_{i=1}^{n} x_i}{n}\right)^2$$

where x is the variable or attribute whose variance is to be measured and n is the number of observations or values of variable x.

Standard deviation of a data is measured as follows:
Standard deviation (σ)= $\sqrt{Variance(x)}$.
Larger value of variance or standard deviation indicates more dispersion in the data and vice versa.

Measuring Data Value Position When the data values of an attribute are arranged in an increasing order, we have seen earlier that median gives the central data value, which divides the entire data set into two halves. Similarly, if the first half of the data is divided into two halves so that each half consists of one-quarter of the data set, then that median of the first half is known as first quartile or Q1. In the same way, if the second half of the data is divided into two halves, then that median of the second half is known as third quartile or Q3. The overall median is also known as second quartile or Q2. So, any data set has five values - minimum, first quartile (Q1), median (Q2), third quartile (Q3), and maximum.

2.3.2 Plotting and Exploring Numerical Data

Box Plots

Now that we have a fairly clear understanding of the data set attributes in terms of spread and central tendency, let's try to make an attempt to visualize the whole thing as a box-plot. A box plot is an extremely effective mechanism to get a one-shot view and understand the nature of the data. The box plot (also called box and whisker plot) gives a standard visualization of the five-number summary statistics of a data, namely minimum, first quartile (Q1), median (Q2), third quartile (Q3), and maximum.

Histogram

Histogram is another plot which helps in effective visualization of numeric attributes. It helps in understanding the distribution of a numeric data into series of intervals, also termed as 'bins'. The important difference between histogram and box plot is:

- The focus of histogram is to plot ranges of data values (acting as 'bins'), the number of data elements in each range will depend on the data distribution. Based on that, the size of each bar corresponding to the different ranges will vary.
- The focus of box plot is to divide the data elements in a data set into four equal portions, such that each portion contains an equal number of data elements.

Histograms might be of different shapes depending on the nature of the data, e.g. skewness. These different patterns give us a quick understanding of the data and thus act as a great data exploration tool.

2.3.3 Exploring Categorical Data

We have seen there are multiple ways to explore numeric data. However, there are not many options for exploring categorical data. As we have read in the earlier section on types of data, statistical measure "mode" is applicable on categorical attributes. As we know, like mean and median, mode is also a statistical measure for central tendency of a data. Mode of a data is the data value which appears most often. In context of categorical attribute, it is the category which has highest number of data values. Since mean and median cannot be applied for categorical variables, mode is the sole measure of central tendency. An attribute may have one or more modes. Frequency distribution of an attribute having single mode is called 'unimodal', two modes are called 'bimodal' and multiple modes are called 'multimodal'.

2.3.4 Exploring Relationship Between Variables

Till now we have been exploring single attributes in isolation. One more important angle of data exploration is to explore relationship between attributes. There are multiple plots to enable us explore the relationship between variables. The basic and most commonly used plot is scatter plot.

Scatter Plot

A scatter plot helps in visualizing bivariate relationships, i.e. relationship between two variables. It is a two-dimensional plot in which points or dots are drawn on coordinates provided by values of the attributes. For example, in a data set there are two attributes – attr_1 and attr_2. We want to understand the relationship between two attributes, i.e. with a change in value of one attribute, say attr_1, how does the value of the other attribute, say attr_2, changes. We can draw a scatter plot, with attr_1 mapped to x-axis and attr_2 mapped in y-axis. So, every point in the plot will have value of attr_1 in the x-coordinate and value of attr_2 in the y-coordinate. As in a two-dimensional plot, attr_1 is said to be the independent variable and attr_2 as the dependent variable.

Two-Way Cross-Tabulations

Two-way cross-tabulations (also called cross-tab or contingency table) are used to understand the relationship of two categorical attributes in a concise way. It has a matrix

format that presents a summarized view of the bivariate frequency distribution. A cross-tab, very much like a scatter plot, helps to understand how much the data values of one attribute changes with the change in data values of another attribute.

2.4 Data Quality and Remediation

2.4.1 Data Quality

Success of machine learning depends largely on the quality of data. A data which has the right quality helps to achieve better prediction accuracy, in case of supervised learning. However, it is not realistic to expect that the data will be flawless. We have already come across at least two types of problems:

- Certain data elements without a value or data with a missing value.
- Data elements having value surprisingly different from the other elements, which we term as outliers.

There are multiple factors which lead to these data quality issues. Following are some of them:

Incorrect sample set selection: The data may not reflect normal or regular quality due to incorrect selection of sample set. For example, if we are selecting a sample set of sales transactions from a festive period and trying to use that data to predict sales in future. In this case, the prediction will be far apart from the actual scenario, just because the sample set has been selected in a wrong time. Similarly, if we are trying to predict poll results using a training data which doesn't comprise of a right mix of voters from different segments such as age, sex, ethnic diversities, etc., the prediction is bound to be a failure. It may also happen due to incorrect sample size. For example, a sample of small size may not be able to capture all aspects or information needed for right learning of the model.

Errors in data collection: resulting in outliers and missing values

- In many cases, a person or group of persons are responsible for the collection of data to be used in a learning activity. In this manual process, there is the possibility of wrongly recording data either in terms of value (say 20.67 is wrongly recorded as 206.7 or 2.067) or in terms of a unit of measurement (say cm. is wrongly recorded as m. or mm.). This may result in data elements which have abnormally high or low value from other elements. Such records are termed as outliers.
- It may also happen that the data is not recorded at all. In case of a survey conducted to collect data, it is all the more possible as survey responders may choose not to respond to a certain question. So the data value for that data element in that responder's record is missing.

2.4.2 Data remediation

The issues in data quality, as mentioned above, need to be remediated, if the right amount of efficiency has to be achieved in the learning activity. Out of the two major areas mentioned above, the first one can be remedied by proper sampling technique. This is a completely different area – covered as a specialized subject area in statistics. We will not cover that in this book. However, human errors are bound to happen, no matter whatever checks and balances we put in. Hence, proper remedial steps need to be taken for the second area mentioned above. We will discuss how to handle outliers and missing values.

Handling outlier

Outliers are data elements with an abnormally high value which may impact prediction accuracy, especially in regression models. Once the outliers are identified and the decision has been taken to amend those values, you may consider one of the following approaches. However, if the outliers are natural, i.e. the value of the data element is surprisingly high or low because of a valid reason, then we should not amend it.

- **Remove outliers:**If the number of records which are outliers is not many, a simple approach may be to remove them
- **Imputation:**One other way is to impute the value with mean or median or mode. The value of the most similar data element may also be used for imputation.
- **Capping:** For values that lie outside the 1.5|×| IQR limits, we can cap them by replacing those observations below the lower limit with the value of 5th percentile and those that lie above the upper limit, with the value of 95th percentile.

If there is a significant number of outliers, they should be treated separately in the statistical model. In that case, the groups should be treated as two different groups, the model should be built for both groups and then the output can be combined.

Handling Missing Values

In a data set, one or more data elements may have missing values in multiple records. As discussed above, it can be caused by omission on part of the surveyor or a person who is collecting sample data or by the responder, primarily due to his/her unwillingness to respond or lack of understanding needed to provide a response. It may happen that a specific question (based on which the value of a data element originates) is not applicable to a person or object with respect to which data is collected. There are multiple strategies to handle missing value of data elements. Some of those strategies have been discussed below.

Eliminate records having a missing value of data elements:

In case the proportion of data elements having missing values is within a tolerable limit, a simple but effective approach is to remove the records having such data elements. This

is possible if the quantum of data left after removing the data elements having missing values is sizeable. However, this will not be possible if the proportion of records having data elements with missing value is really high as that will reduce the power of model because of reduction in the training data size.

Imputing missing values:

Imputation is a method to assign a value to the data elements having missing values. Mean/mode/median is most frequently assigned value. For quantitative attributes, all missing values are imputed with the mean, median, or mode of the remaining values under the same attribute. For qualitative attributes, all missing values are imputed by the mode of all remaining values of the same attribute. However, another strategy may be identify the similar types of observations whose values are known and use the mean/median/mode of those known values.

Estimate missing values:

If there are data points similar to the ones with missing attribute values, then the attribute values from those similar data points can be planted in place of the missing value. For finding similar data points or observations, distance function can be used.

2.5 Data Pre-processing

2.5.1 Dimensionality Reduction

High-dimensional data sets need a high amount of computational space and time. At the same time, not all features are useful – they degrade the performance of machine learning algorithms. Most of the machine learning algorithms perform better if the dimensionality of data set, i.e. the number of features in the data set, is reduced. Dimensionality reduction helps in reducing irrelevance and redundancy in features. Also, it is easier to understand a model if the number of features involved in the learning activity is less.
Dimensionality reduction refers to the techniques of reducing the dimensionality of a data set by creating new attributes by combining the original attributes. The most common approach for dimensionality reduction is known as Principal Component Analysis (PCA). PCA is a statistical technique to convert a set of correlated variables into a set of transformed, uncorrelated variables called principal components.
Another commonly used technique which is used for dimensionality reduction is Singular Value Decomposition (SVD).

2.5.2 Feature Subset Selection

Feature subset selection or simply called feature selection, both for supervised as well as unsupervised learning, try to find out the optimal subset of the entire feature set which significantly reduces computational cost without any major impact on the learning accuracy. It may seem that a feature subset may lead to loss of useful information as certain features are going to be excluded from the final set of features used for learning. However, for elimination only features which are not relevant or redundant are selected.

A feature is considered as irrelevant if it plays an insignificant role (or contributes almost no information) in classifying or grouping together a set of data instances. All irrelevant features are eliminated while selecting the final feature subset. A feature is potentially redundant when the information contributed by the feature is more or less same as one or more other features. Among a group of potentially redundant features, a small number of features can be selected as a part of the final feature subset without causing any negative impact to learn model accuracy.

Questions

1. Create a Pandas Series from a Python list [10, 20, 30, 40, 50] with custom indexes ['a','b','c','d','e'].
   (a) Print the first 3 elements.
   (b) Access the element at index 'c'.

2. Create a Series from a NumPy array of random integers between 1–100 (size = 10).
   (a) Find the maximum, minimum, and mean values.
   (b) Apply a function to square each element.

3. Given a Series with values [5, np.nan, 8, np.nan, 12]:
   (a) Check for missing values
   . (b) Fill missing values with forward fill (ffill).
   (c) Drop missing values.

4. Convert a Python dictionary data = {'Math': 85, 'Science': 90, 'English': 88} into a Series. Retrieve the value for 'Science'.

5. Create a Data Frame using a dictionary: data = { 'Name': ['Amit', 'Riya', 'John', 'Sara'], 'Age': [25, 30, 22, 28], 'Salary': [50000, 60000, 55000, 65000] }
   (a) Select all rows where Age $>$ 25.
   (b) Select rows where Salary is between 55,000 and 65,000.

6. Create a DataFrame: data = { 'Department': ['HR','IT','HR','IT','Finance'], 'Employee': ['A','B','C','D','E'], 'Salary': [40000, 50000, 42000, 55000, 60000] }
   (a) Find average salary per department.
   (b) Count employees in each department.
   (c) Sort employees by Salary in ascending order.
   (d) Sort by Department then by Salary (descending).

7. Create a DataFrame with duplicate rows.
   (a) Identify the duplicates and Display only the duplicate rows.
   (b) Drop duplicates while:
   • Keeping the first occurrence
   • Keeping the last occurrence.
   • Removing all duplicates.
   (c) Drop duplicates based on specific columns .
   (d) Count the number of duplicate rows using duplicated().sum().
   (e) Extract only the unique values from a single column (e.g., Name).

8. Create a DataFrame containing NaN values.
   (a) Detect missing values
   (b) Count missing values per column.
   (c) Drop rows:
   • With any NaN values.
   • Where all values are NaN.
   • Where NaN appears in specific columns (like Age or Salary).
   (d) Fill missing values:
   • With a fixed value (e.g., 0 or "Unknown").
   • With the mean of the column.
   • Using forward fill and backward fill.
   • Using linear interpolation.
   (e) Compare the results of forward fill vs interpolation on the same dataset.

Chapter 3

Modelling and Evaluation

3.1 Introduction

In the learning process, abstraction is a significant step as it represents raw input data in a summarized and structured format, such that a meaningful insight is obtained from the data. This structured representation of raw input data to the meaningful pattern is called a **model**. The model might have different forms: mathematical equation, graph or tree structure, computational block, etc.

The decision regarding which model is to be selected for a specific data set is taken by the learning task, based on the problem to be solved and the type of data. For example hen the problem is related to prediction and the target field is numeric and continuous, the regression model is assigned. The process of assigning a model, and fitting a specific model to a data set is called **model training**.

The heuristic (or shortcut) used in learning systematically favours certain outcomes over others, results in erroneous result. If the outcome is systematically incorrect, the learning is said to have a **bias**. A machine learning algorithm creates its cognitive capability by building a mathematical formulation or function, known as **target function**, based on the features in the input data set. In machine learning someone has to provide some non-learnable parameters, also called **hyper-parameters**.

3.2 Selecting a Model

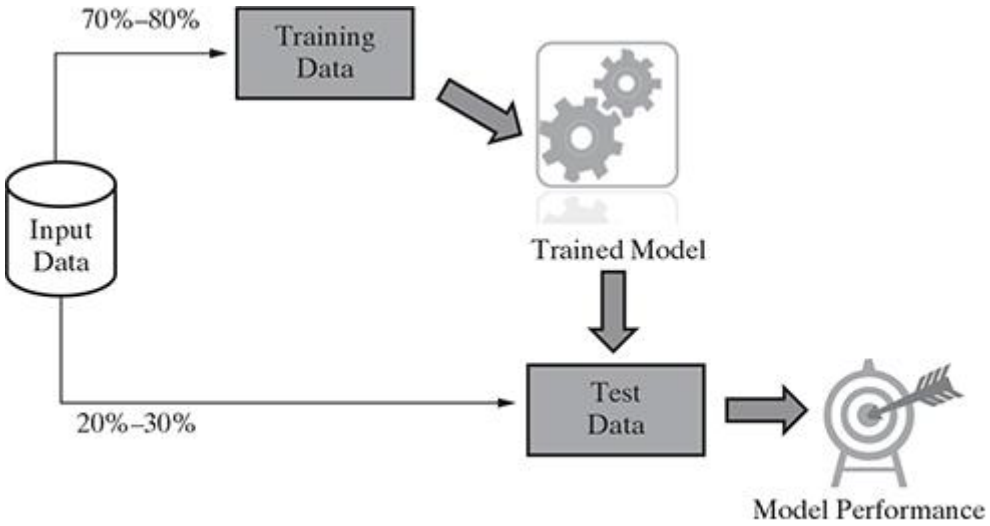
In machine learning paradigm, the potential causes of disturbance, e.g. average income of the local population, weapon sales, the inflow of immigrants, etc. are **input variables** (also called predictors, attributes, features, independent variables, or simply variables). The number of criminal incidents is an **output variable** (also called response, target or dependent variable). Input variables can be denoted by X, while individual input variables are represented as $X_1, X_2, X_3, \ldots, X_n$ and output variable by symbol Y. The relationship between X and Y is represented in the general form: $Y = f(X) + e$, where 'f ' is the target function and 'e' is a random error term.

A **cost function** or **loss function** (also called **error function**) helps to measure the extent to which the model is going wrong in estimating the relationship between X and

Y. In that sense, cost function can tell how bad the model is performing. For example, R-squared is a cost function of regression model.

Machine learning is like an **optimization problem**, where we tune the parameters of a model to find the most suitable solution to a problem. The quality or optimality of a solution is measured using an **objective function**, which takes in data and model (along with parameters) as input and returns a value. Target is to find values of model parameter to maximize or minimize the return value.

3.3 Training a Model (For Supervised Learning)

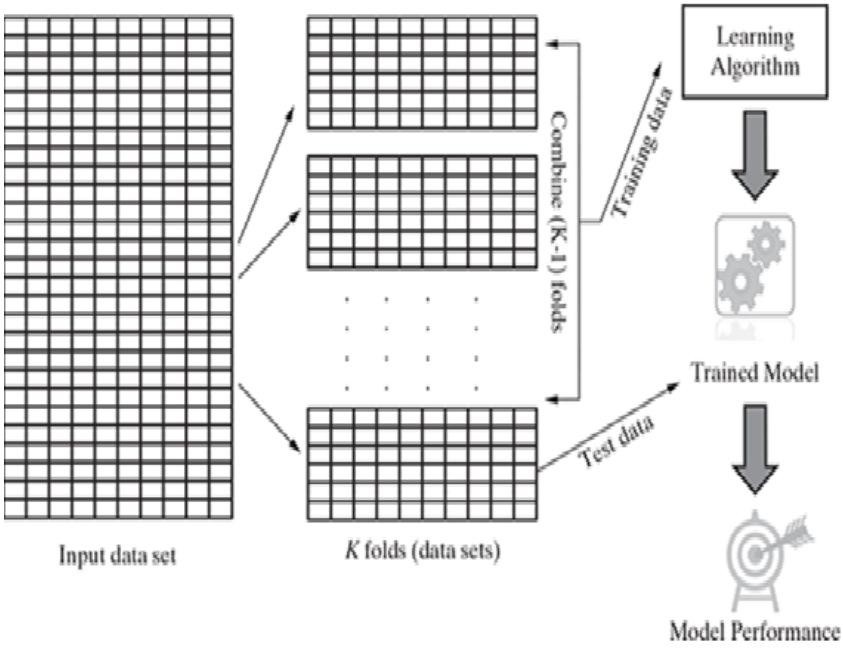


3.3.1 Holdout method

In supervised learning, the holdout method is used to evaluate model performance by splitting the labeled dataset into training data (70–80%) and test data (20–30%). The training set is used to build the model, while the test set is used to validate it by comparing predicted labels with actual labels. A key issue is that random splitting may create imbalanced class distributions, especially when some classes have fewer samples. To address this, stratified random sampling is used, which ensures each class is proportionally represented in both training and test sets. Sometimes, data is split into three sets: training, validation, and test. The validation set helps refine the model during iterations, while the test set is used only once to report the model's final performance.

3.3.2 K-fold Cross-validation method

The dataset is split into k equal parts (folds). The model is trained on k–1 folds and tested on the remaining 1 fold. This process is repeated k times, each time with a different fold as the test set. The performance is measured by taking the average accuracy (or other metric) across all k trials. Advantages: Uses all data for both training and testing. Reduces bias due to random partitioning. Common choice: k = 5 or k = 10.

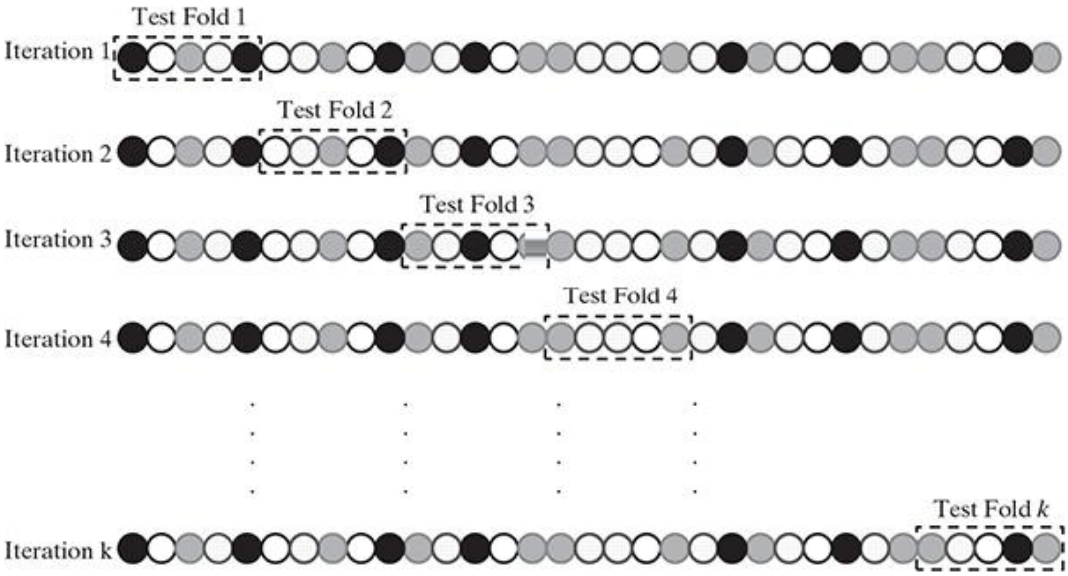


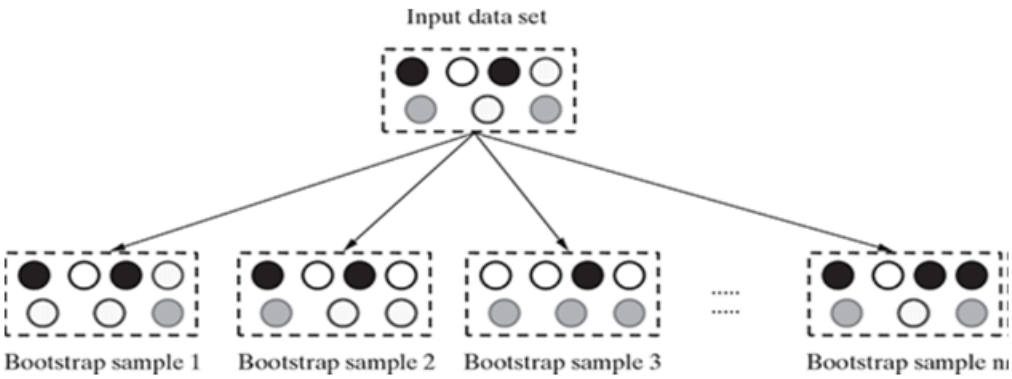
3.3.3 Leave-one-out cross-validation (LOOCV)

LOOCV is a special case of k-fold cross validation where k = N (the number of data points). The model is trained on N–1 samples and tested on the 1 remaining sample. This process repeats N times, each time leaving out a different sample for testing. Final performance is the average of all N test results. Advantages: Makes maximum use of the data for training. Gives an almost unbiased estimate of model performance. Disadvantages: Very computationally expensive for large datasets, since the model must be trained N times.

3.3.4 Bootstrap sampling

Bootstrap is a resampling technique used to estimate model performance or statistics when data is limited. From a dataset of size N, we create multiple bootstrap samples by randomly picking N items with replacement. Because of replacement, some items may appear multiple times, while others may be left out. The model is trained on each bootstrap sample, and performance is evaluated on the out-of-bag (OOB) data (the samples not selected). Final performance is obtained by averaging results across many bootstrap iterations. Advantages: Works well with small datasets. Provides an estimate of model accuracy, bias, and variance. Disadvantages: Computationally intensive if repeated many times.





3.3.5 Cross-validation vs. Bootstrapping

CROSS-VALIDATION	BOOTSTRAPPING
1. It is a special variant of holdout method, called repeated holdout. Hence uses stratified random sampling approach (without replacement). Data set is divided into 'k' random partitions, with each partition containing approximately $\frac{n}{k}$ number of unique data elements, where 'n' is the total number of data elements and 'k' is the total number of folds.	1. It uses the technique of Simple Random Sampling with Replacement (SRSWR). So the same data instance may be picked up multiple times in a sample.
2. The number of possible training/test data samples that can be drawn using this technique is finite.	2. In this technique, since elements can be repeated in the sample, possible number of training/test data samples is unlimited.

3.3.6 Lazy vs. Eager learner

Lazy Learner	Eager Learner
1. Build a generalized model during training (abstraction + generalization).	1. Do not build a model; directly use training data for prediction. Also called rote learning, instance-based learning, or non-parametric learning.
2. Training is time-consuming, but prediction is fast since the model is ready.	2. Training is fast, but prediction is slow since it requires comparing with stored data.
3. Examples: Decision Tree, SVM, Neural Networks.	3. Example: k-Nearest Neighbor (k-NN).

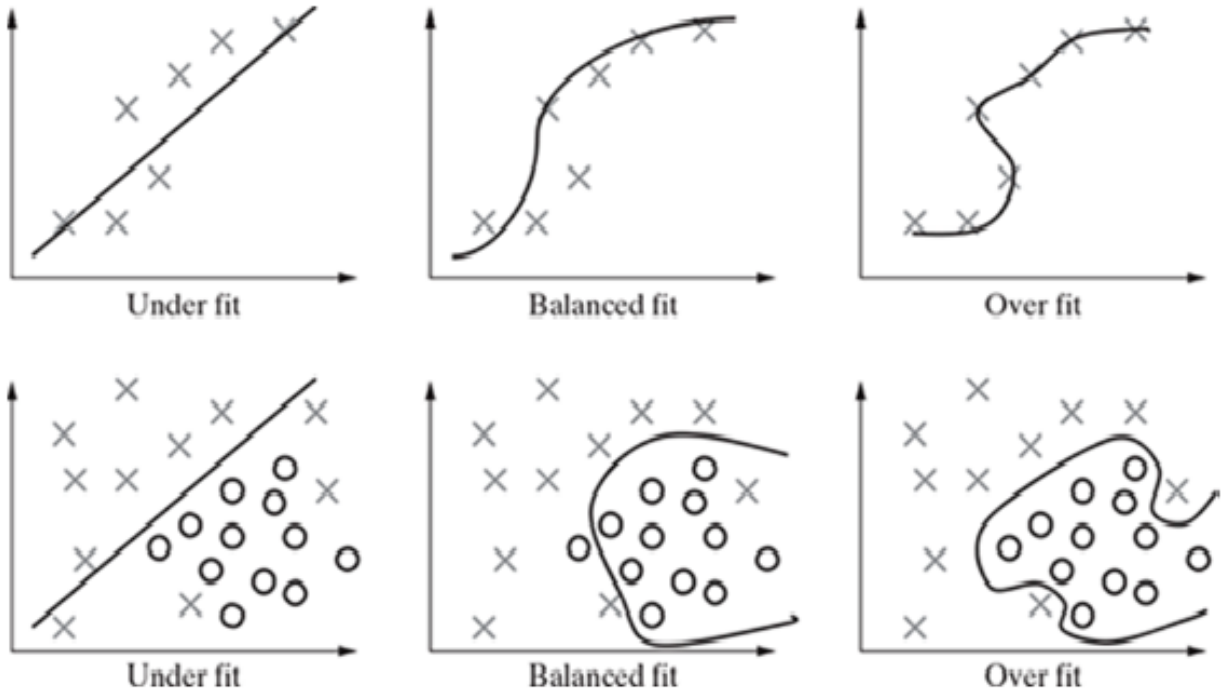
3.3.7 Parametric vs. Non-parametric Model

Parametric Model	Non-parametric Model
1. Have a finite number of parameters.	1. Have a potentially infinite number of parameters; model complexity grows with data.
2. Model size is fixed regardless of dataset size.	2. Do not assume a fixed functional form.
3. Learn a specific functional form (e.g., linear, polynomial).	More flexible, can capture complex relationships.
4. Examples: Linear Regression, Logistic Regression, SVM (with fixed kernel parameters).	4. Example: k-Nearest Neighbor, Decision Trees, Random Forests.

3.4 Model Representation and Interpretability

3.4.1 Underfitting vs. Overfitting

Underfitting	Overfitting
1. Model is too simple to capture the underlying data patterns.	1. Model is too complex and fits training data too closely, including noise and outliers.
2. Performs poorly on both training and test data.	2. Performs well on training data but poorly on test data (poor generalization).
3. Causes: using overly simple models (e.g., linear for non-linear data), insufficient training data, irrelevant features.	3. Causes: excessive model complexity, lack of validation.
4. Avoidance: • Use more training data. • Apply effective feature selection or better model choice.	4. Avoidance: • Use re-sampling techniques like k-fold cross validation. • Hold out a validation set. • Remove weak or irrelevant features/nodes.



3.4.2 Bias–Variance Trade-off

Bias Error:

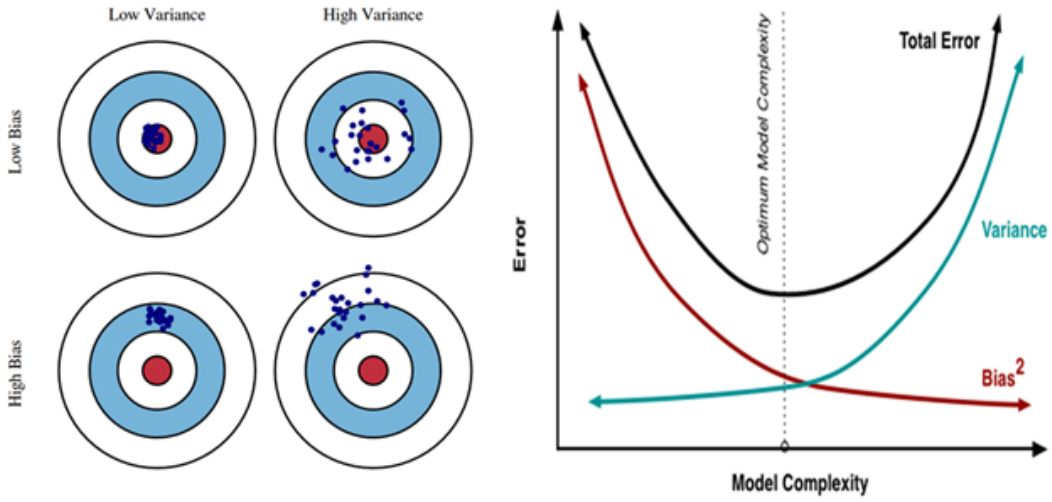
- Caused by simplifying assumptions in the model.
- Leads to underfitting → poor performance on both training and test data.
- Common in parametric models (e.g., Linear Regression, Logistic Regression).
- High bias → Low variance.

Variance Error

- Caused by sensitivity to small changes in training data.
- Leads to overfitting → good performance on training data, poor generalization on test data.
- Common in non-parametric models (e.g., k-NN, Decision Trees).
- High variance → Low bias.

Trade-off

- A simple model → high bias, low variance.
- A complex model → low bias, high variance.
- Goal: Find the right balance for best generalization.



3.5 Evaluating Performance of a Model

3.5.1 Evaluation for Classification Models

In a binary classification problem (class of interest = Win), predictions fall into four categories:

1. **True Positive (TP):**
   - Model predicts Win, and the team actually won.
   - Correctly identified positive case.
2. **False Positive (FP):**
   - Model predicts Win, but the team actually lost.
   - Incorrectly predicted as positive (also called Type I error).
3. **False Negative (FN):**
   - Model predicts Loss, but the team actually won.
   - Missed the positive case (also called Type II error).
4. **True Negative (TN):**
   - Model predicts Loss, and the team actually lost.
   - Correctly identified negative case.

Confusion Matrix (Win as Positive class):

	Actual Win	**Actual Loss**
Predicted Win	True Positive (TP)	False Positive (FP)
Predicted Loss	False Negative (FN)	True Negative (TN)

Matrices for Evaluation:

1. Model accuracy: Total number of correct classifications

$$\text{Model accuracy} = \frac{TP + TN}{TP + FP + FN + TN}.$$

2. Error rate: The percentage of misclassifications

$$\text{Error rate} = \frac{FP + FN}{TP + FP + FN + TN}.$$

3. Kappa value: Indicates the adjusted model accuracy

$$\text{Kappa value}(k) = \frac{P(a) - P(p_r)}{1 - P(p_r)}.$$

$P(a)$= Proportion of observed agreement between actual and predicted in overall data set

$$= \frac{TP + TN}{TP + FP + FN + TN}.$$

$P(p_r)$= Proportion of expected agreement between actual and predicted data both in case of class of interest as well as other class

$$= \frac{TP + FP}{TP + FP + FN + TN} \times \frac{TP + FN}{TP + FP + FN + TN}$$
$$+ \frac{FP + TN}{TP + FP + FN + TN} \times \frac{FP + TN}{TP + FP + FN + TN}.$$

4. Sensitivity/TPR (True Positive Rate): measures the proportion of positive cases which were correctly classified.

$$\text{True Positive Rate} = \frac{TP}{TP + FN}.$$

5. FPR (False Positive Rate): measures the proportion of negative cases which were not correctly classified.

$$\text{False Positive Rate} = \frac{FP}{FP + TN}.$$

6. Specificity: measures the proportion of negative cases which have been correctly classified.

$$\text{Specificity} = \frac{TN}{FP + TN}.$$

7. Precision: gives the proportion of positive predictions which are truly positive.

$$\text{Precision} = \frac{TP}{TP + FP}.$$

8. Recall: gives the proportion of correct prediction of positives to the total number of positives.

$$\text{Recall} = \frac{TP}{TP + FN}.$$

9. F-measure: takes the harmonic mean of precision and recall

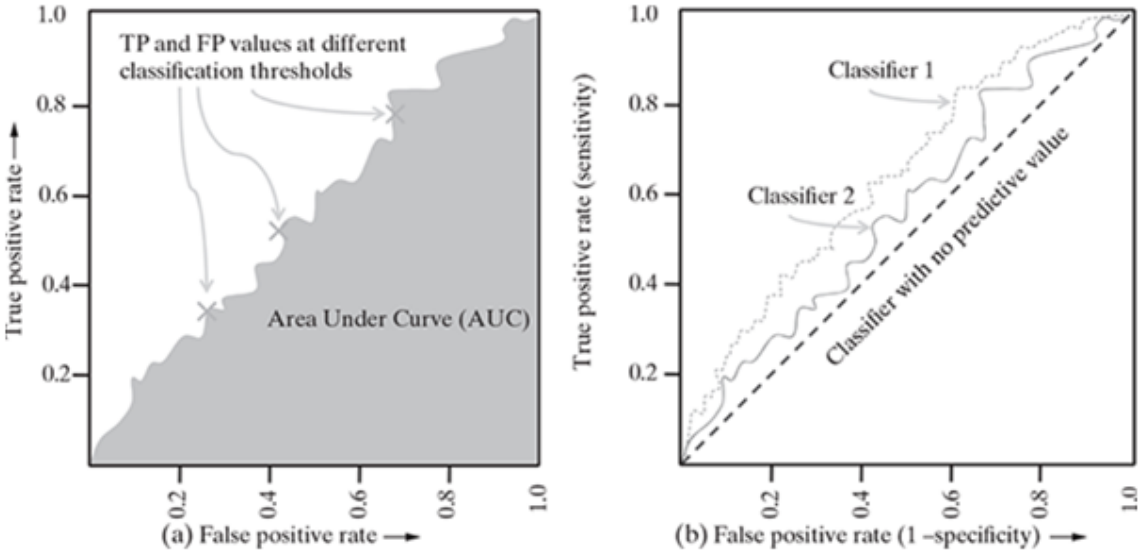
$$\text{F-measure} = \frac{2 \times Precision \times Recall}{Precision + Recall}.$$

10. Receiver operating characteristic (ROC) curves: In the ROC curve, the FPR is plotted (in the horizontal axis) against TPR (in the vertical axis) at different classification thresholds.

    The area under curve (AUC) value is the area of the two-dimensional space under the curve extending from (0, 0) to (1, 1), where each point on the curve gives a set of true and false positive rates at a specific classification threshold. This curve gives

an indication of the predictive quality of a model. AUC value ranges from 0 to 1, with an AUC of less than 0.5 indicating that the classifier has no predictive ability and for predictive values from 0.5 to 1.0, we have 0.5 – 0.6 : almost no predictive ability; 0.6 – 0.7 : weak predictive ability; 0.7 – 0.8 : fair predictive ability; 0.8 – 0.9 : good predictive ability; 0.9 – 1.0 : excellent predictive ability.

In Fig 3.7(b), the AUC of classifier 1 is more than the AUC of classifier 2. So, we can draw the inference that classifier 1 is better than classifier 2.



Example: : Consider the case of win/loss prediction of a cricket match.

	ACTUAL WIN	ACTUAL LOSS
Predicted Win	85	4
Predicted Loss	2	9

Here, TP=85, FP=4, FN=2, TN=9. Total Prediction = TP+FP+FN+TN = 100.

$$\text{Model accuracy} = \frac{TP+TN}{TP+FP+FN+TN} = \frac{85+9}{85+4+2+9} = \frac{94}{100} = 94\%.$$

$$\text{Error rate} = \frac{FP+FN}{TP+FP+FN+TN} = \frac{4+2}{85+4+2+9} = \frac{6}{100} = 6\% = 1-\text{Model accuracy}.$$

$$\text{Kappa value}(k) = \frac{P(a)-P(p_r)}{1-P(p_r)},$$

where $P(a)$= Proportion of observed agreement between actual and predicted in overall data set

$$= \frac{TP+TN}{TP+FP+FN+TN} = \frac{85+9}{85+4+2+9} = \frac{94}{100} = 0.94.$$

and $P(p_r)$= Proportion of expected agreement between actual and predicted data both in case of class of interest as well as other class

$$= \frac{TP+FP}{TP+FP+FN+TN} \times \frac{TP+FN}{TP+FP+FN+TN}$$
$$+ \frac{FP+TN}{TP+FP+FN+TN} \times \frac{FP+TN}{TP+FP+FN+TN}$$
$$= \frac{85+4}{85+4+2+9} \times \frac{85+2}{85+4+2+9} + \frac{2+9}{85+4+2+9} \times \frac{4+9}{85+4+2+9}$$
$$= \frac{89}{100} \times \frac{87}{100} + \frac{11}{100} \times \frac{13}{100} = 0.7886.$$

Hence,

$$\text{Kappa value}(k) = \frac{0.94 - 0.7886}{1 - 0.7886} = 0.7162.$$

$$\text{True Positive Rate} = \frac{TP}{TP+FN} = \frac{85}{85+2} = \frac{85}{87} = 97.7\%.$$

$$\text{False Positive Rate} = \frac{FP}{FP+TN} = \frac{4}{4+9} = \frac{4}{13} = 30.8\%.$$

$$\text{Specificity} = \frac{TN}{FP+TN} = \frac{9}{9+4} = \frac{9}{13} = 69.2\% = 1 - FPR.$$

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{85}{85+4} = \frac{85}{89} = 95.5\%.$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{85}{85+2} = \frac{85}{87} = 97.7\%.$$

$$\text{F-measure} = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{2 \times 0.955 \times 0.977}{0.955 + 0.977} = 96.6\%.$$

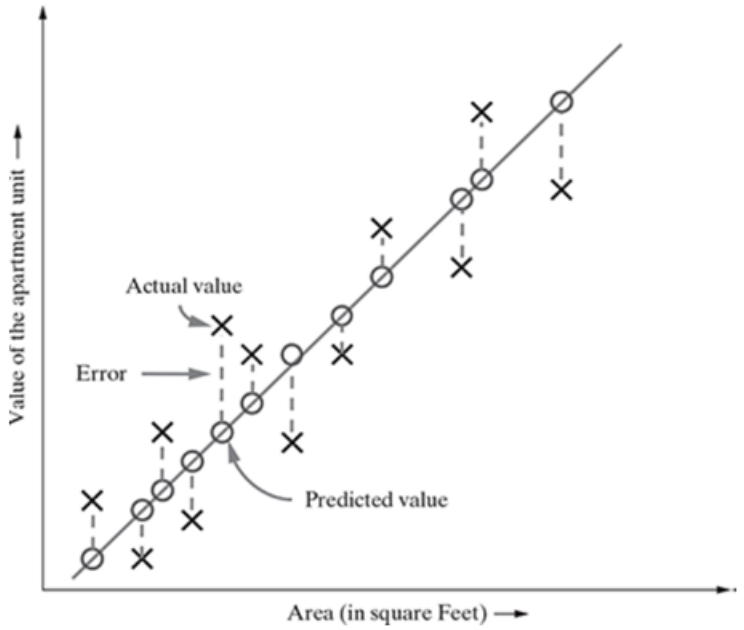
3.5.2 Evaluation for Regression Models

A well-fitted regression model produces predicted values close to actual values. Hence, a regression model which ensures that the difference between predicted and actual values is low can be considered as a good model. Figure 3.8 represents a very simple problem of real estate value prediction solved using linear regression model. If 'area' is the predictor variable (say x) and 'value' is the target variable (say y), the linear regression model can be represented in the form:

$$y = \alpha + \beta x.$$

For a certain value of x, say $\hat{x}$, the value of y is predicted as $\hat{y}$ whereas the actual value of y is Y (say).

Residual Error: The distance between the actual value Y and the fitted or predicted value ŷ is known as residual. The regression model can be considered to be fitted well if the difference between actual and predicted value, i.e. the residual value is less.



R-squared Error: R-squared is a good measure to evaluate the model fitness. It is also known as the coefficient of determination, or for multiple regression, the coefficient of multiple determination. The R-squared value lies between 0 to 1 (0%˘100%) with a larger value representing a better fit. It is calculated as:

$$R^2 = \frac{SST - SSE}{SST},$$

where **Sum of Squares Total (SST)** is squared differences of each observation from the overall mean

$$SST = \sum_{i=1}^{n} (y_i - \bar{y})^2,$$

where $\bar{y}$ is the mean and **Sum of Squared Errors (SSE)** (of prediction) is sum of the squared residuals

$$SSE = \sum_{i=1}^{n} (Y_i - \hat{y})^2,$$

where $\hat{y}_i$ is the predicted value of y_i and Y_i is the actual value of y_i.

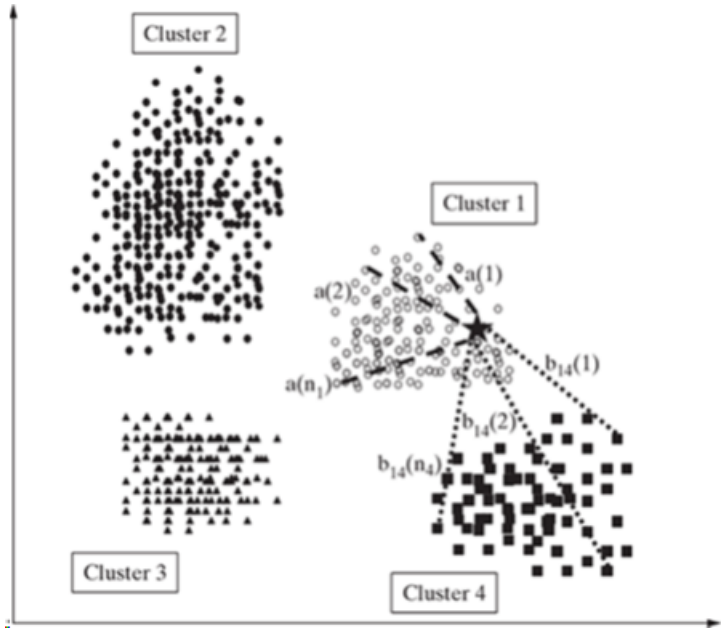
3.5.3 Evaluation for unsupervised learning - Clustering

1. **Internal evaluation:** The internal evaluation methods measure cluster quality based on homogeneity of data belonging to the same cluster and heterogeneity of data belonging to different clusters. **Silhouette width**, one of the most popular internal evaluation methods, uses distance (Euclidean or Manhattan distance) between data elements as a similarity measure. The value of silhouette width ranges between ˘1 and +1, with a high value indicating high intra-cluster homogeneity and inter-cluster heterogeneity. For a data set clustered into 'k' clusters, silhouette

width is calculated as:

$$\text{Silhouette width } = \frac{b(i) - ai}{\max\{a(i), b(i)\}},$$

where $a(i)$ is the average distance between the i th data and all other data in the same cluster, and $b(i)$ is the lowest average distance between the i th data and data of all other clusters.



Example: There are four clusters namely cluster $1, 2, 3$, and 4. Let's consider an arbitrary data element 'i' in cluster 1 (with n_1 data elements), resembled by the asterisk. Let $a(i)$ is the average of the distances $a_{i1}, a_{i2}, \ldots, a_{in_1}$ of the different data elements from the ith data element in cluster 1. Mathematically,

$$a(i) = \frac{a_{i1} + a_{i2} + \ldots + a_{in_1}}{n_1}.$$

Let b_{14}(average) is the average distance of an arbitrary data element 'i' in cluster 1 with the different data elements from cluster 4 (with n_4 elements)

$$b_{14}(average) = \frac{b_{14}(1) + b_{14}(2) + \cdots + b_{14}(n_4)}{n_4}.$$

Similarly, we can compute b_{12}(average), b_{13}(average). Finally

$$b(i) = \min\{b_{12}(average), b_{13}(average), b_{14}(average)\}.$$

2. **External evaluation:** In this approach, class label is known for the data set subjected to clustering. However, quite obviously, the known class labels are not a part of the data used in clustering. The cluster algorithm is assessed based on how close the results are compared to those known class labels. **Purity** evaluates the extent to which clusters contain a single class. For a data set having 'n' data instances and 'c' known class labels which generates 'k' clusters, purity is measured as:

$$Purity = \frac{1}{n} \sum_{k} \max(k \cap c).$$

3.6 Improving Performance of a Model

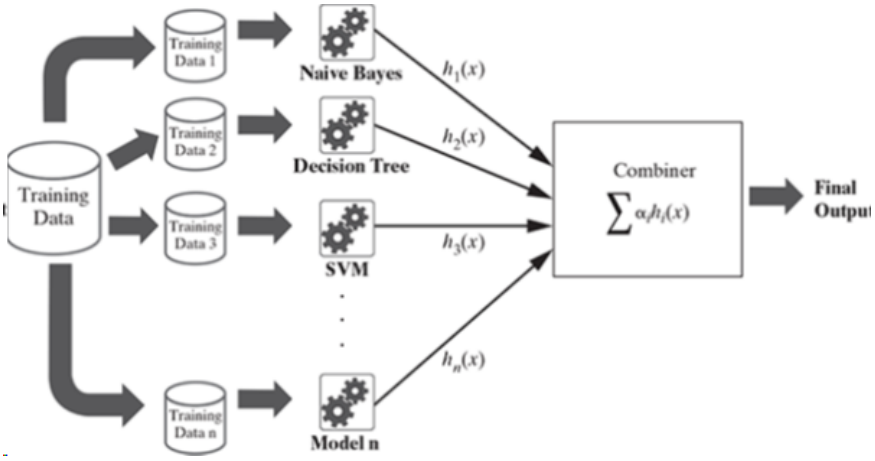
3.6.1 Model parameter tuning

One effective way to improve model performance is by tuning model parameter. Model parameter tuning is the process of adjusting the model fitting options. For example, in the popular classification model k-Nearest Neighbour (kNN), using different values of 'k' or the number of nearest neighbours to be considered, the model can be tuned. In the same way, a number of hidden layers can be adjusted to tune the performance in neural networks model. Most machine learning models have at least one parameter which can be tuned.

3.6.2 Ensemble of models

Ensemble learning is an approach where multiple models are combined to improve overall performance. Different models complement each other—where one struggles, another may perform well. By aggregating their predictions, ensembles reduce bias, minimize variance, and build stronger learners from weaker ones. Even models trained independently can deliver a performance boost when ensembled. The ensemble process typically involves:

1. Model Building – Train multiple models on the dataset.
2. Diversification – Vary training subsets using sampling methods like bootstrapping, or use different algorithms (e.g., SVM, neural networks, kNN).
3. Combination – Aggregate model outputs with a combination function.
4. Decision Making – For prediction tasks, a simple approach is majority voting (e.g., if 3 out of 5 models predict win and 2 predict loss, the final decision is win).



3.6.3 Bagging (Bootstrap Aggregating)

Bagging is one of the earliest and most popular ensemble methods. It creates multiple training datasets through bootstrap sampling and trains several models using the same algorithm. The final output is obtained by majority voting (for classification) or averaging (for regression). Bagging works particularly well with unstable learners like decision trees, where small changes in data can cause large variations in outcomes.

3.6.4 Boosting

Like bagging, boosting is a key ensemble technique that trains weak learners on resampled data and combines them using weighted voting, where stronger models get higher weight. A popular variant is AdaBoost, which iteratively improves weak learners to form a strong classifier.

3.6.5 Random forest

Random Forest is another widely used ensemble method that builds a "forest" of decision trees using bagging and feature randomness. By averaging their results, it reduces overfitting and improves prediction accuracy.

Chapter 4

Basics of Feature Engineering

4.1 Introduction

In this chapter, we study one of the most important aspects of machine learning - **Feature Engineering**. It focuses on three key components:

- Feature Construction
- Feature Selection
- Feature Transformation
- Machine Learning requires preparatory steps before modeling.
- One of the critical preparatory steps is **Feature Engineering**.
- It transforms raw input data into meaningful, well-aligned features ready to be used by ML models.

4.1.1 What is a Feature?

A feature is an attribute of a data set that is used in a machine learning process. There is a view amongst certain machine learning practitioners that only those attributes which are meaningful to a machine learning problem are to be called as features, but this view has to be taken with a pinch of salt. The features in a data set are also called its dimensions. So a data set having 'n' features is called an n-dimensional data set.

The famous **Iris dataset** has features: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width, and Species (class variable). It is a five-dimensional data set.

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
6.7	3.3	5.7	2.5	Virginica
4.9	3	1.4	0.2	Setosa
5.5	2.6	4.4	1.2	Versicolor
6.8	3.2	5.9	2.3	Virginica
5.5	2.5	4	1.3	Versicolor
5.1	3.5	1.4	0.2	Setosa
6.1	3	4.6	1.4	versicolor

FIG. 4.1 Data set features

4.1.2 What is Feature Engineering?

Feature engineering refers to the process of translating a data set into features such that these features are able to represent the data set more effectively and result in a better learning performance.
It has two major elements:

1. Feature Transformation
2. Feature Subset Selection

4.2 Feature Transformation

Feature transformation transforms the data – structured or unstructured, into a new set of features which can represent the underlying problem which machine learning is trying to solve. There are two variants of feature transformation:

1. Feature Construction
2. Feature Extraction

4.2.1 Feature construction

Feature construction process discovers missing information about the relationships between features and augments the feature space by creating additional features. Hence, if there are 'n' features or dimensions in a data set, after feature construction 'm' more features or dimensions may get added. So at the end, the data set will become '$n + m$' dimensional.

Let's take the example of a real estate data set having details of all apartments sold in a specific region.

The data set has three features – apartment length, apartment breadth, and price of the apartment. If it is used as an input to a regression problem, such data can be training data for the regression model. So given the training data, the model should be able to

predict the price of an apartment whose price is not known or which has just come up for sale. However, instead of using length and breadth of the apartment as a predictor, it is much convenient and makes more sense to use the area of the apartment, which is not an existing feature of the data set. So such a feature, namely apartment area, can be added to the data set. In other words, we transform the three dimensional data set to a four-dimensional data set, with the newly 'discovered' feature apartment area being added to the original data set. This is depicted in Figure 4.2.

apartment_ length	apartment_ breadth	apartment_ price
80	59	23,60,000
54	45	12,15,000
78	56	21,84,000
63	63	19,84,000
83	74	30,71,000
92	86	39,56,000

apartment_ length	apartment_ breadth	apartment_ area	apartment_ price
80	59	4,720	23,60,000
54	45	2,430	12,15,000
78	56	4,368	21,84,000
63	63	3,969	19,84,500
83	74	6,142	30,71,000
92	86	7912	39,56,000

FIG. 4.2 Feature construction (example 1)

There are certain situations where feature construction is an essential activity before we can start with the machine learning task. These situations are:

- when features have categorical value and machine learning needs numeric value inputs
- when features having numeric (continuous) values and need to be converted to ordinal values
- when text-specific feature construction needs to be done

Encoding categorical (nominal) variables

Athlete dataset as presented in Figure 4.3a. with features Age, City of origin, Parents athlete, Chance of win. The feature chance of a win is a class variable while the others are predictor variables. variables.

We know that any machine learning algorithm, whether it's a classification algorithm (like kNN) or a regression algorithm, requires numerical figures to learn from. So there are three features – City of origin, Parents athlete, and Chance of win, which are categorical in nature and cannot be used by any machine learning task.
In this case, feature construction can be used to create new dummy features which are usable by machine learning algorithms.

- City of origin (A, B, C) → origin_city_A, origin_city_B, origin_city_C.
- Parents athlete (Y/N) → parents_athlete_Y, parents_athlete_N.
- Chance of win (Y/N) → win_chance_Y, win_chance_N.

Each dummy variable gets 0/1 values. The entire set of transformation for athletes' data set is shown in Figure 4.3b.
However, examining closely, we see that the features 'Parents athlete' and 'Chance of win' in the original data set can have two values only. So creating two features from them is a kind of duplication, since the value of one feature can be decided from the value of the other. To avoid this duplication, we can just leave one feature and eliminate the other, as shown in Figure 4.3c.

Age (Years)	City of origin	Parents athlete	Chance of win
18	City A	Yes	Y
20	City B	No	Y
23	City B	Yes	Y
19	City A	No	N
18	City C	Yes	N
22	City B	Yes	Y

(a)

Age (Years)	origin_ city_A	origin_ city_B	origin_ city_C	parents_ athlete_Y	parents_ athlete_N	win_ chance_Y	win_ chance_N
18	1	0	0	1	0	1	0
20	0	1	0	0	1	1	0
23	0	1	0	1	0	1	0
19	1	0	0	0	1	0	1
18	0	0	1	1	0	0	1
22	0	1	0	1	0	1	0

(b)

Age (Years)	origin_city_A	origin_city_B	origin_city_C	parents_athlete_Y	win_chance_Y
18	1	0	0	1	1
20	0	1	0	0	1
23	0	1	0	1	1
19	1	0	0	0	0
18	0	0	1	1	0
22	0	1	0	1	1

(c)

FIG. 4.3 Feature construction (encoding nominal variables)

Encoding categorical (ordinal) variable

Let's assume that there are three variable – science marks, maths marks and grade as shown in Figure 4.4a.

As we can see, the grade is an ordinal variable with values A, B, C, and D. To transform this variable to a numeric variable, we can create a feature num_grade mapping a numeric

value against each ordinal value.
In the context of the current example, grades A, B, C, and D. in Figure 4.4a is mapped to values 1, 2, 3, and 4 in the transformed variable shown in Figure 4.4b.

marks_science	marks_maths	Grade
78	75	B
56	62	C
87	90	A
91	95	A
45	42	D
62	57	B

(a)

marks_science	marks_maths	num_grade
78	75	2
56	62	3
87	90	1
91	95	1
45	42	4
62	57	2

(b)

FIG. 4.4 Feature construction (encoding ordinal variables)

Transforming numeric (continuous) features to categorical features

Sometimes there is a need of transforming a continuous numerical variable into a categorical variable.

we may want to treat the real estate price prediction problem, which is a regression problem, as a real estate price category prediction, which is a classification problem.

In that case, we can 'bin' the numerical data into multiple categories based on the data range. In the context of the real estate price prediction example, the original data set has a numerical feature apartment_price as shown in Figure 4.5a. It can be transformed to a categorical variable price-grade either as shown in Figure 4.5b or as shown in Figure 4.5c.

apartment_ area	apartment_ price
4,720	23,60,000
2,430	12,15,000
4,368	21,84,000
3,969	19,84,500
6,142	30,71,000
7,912	39,56,000

(a)

apartment_ area	apartment_ grade
4,720	Medium
2,430	Low
4,368	Medium
3,969	Low
6,142	High
7,912	High

(b)

apartment_ area	apartment_ grade
4,720	2
2,430	1
4,368	2
3,969	1
6,142	3
7,912	3

(c)

FIG. 4.5 Feature construction (numeric to categorical)

Text-specific feature construction

In the current world, text is arguably the most predominant medium of communication. Whether we think about social networks like Facebook or micro-blogging channels like Twitter or emails or short messaging services such as Whatsapp, text plays a major role in the flow of information. Hence, text mining is an important area of research – not only for technology practitioners but also for industry practitioners. However, making sense of text data, due to the inherent unstructured nature of the data, is not so straightforward. In the first place, the text data chunks that we can think about do not have readily available features, like structured data sets, on which machine learning tasks can be executed. All machine learning models need numerical data as input. So the text data in the data sets need to be transformed into numerical features.

Text data, or corpus which is the more popular keyword, is converted to a numerical representation following a process is known as vectorization. In this process, word occurrences in all documents belonging to the corpus are consolidated in the form of bag-of-words. There are three major steps that are followed:

1. **Tokenize** → split into words/tokens.
2. **Count** → word occurrences.

3. **Normalize** → weight terms (reduce importance of common words).

Creates a document-term matrix (rows = documents, columns = tokens, values = counts).

This	House	Build	Feeling	Well	Theatre	Movie	Good	Lonely	...
2	1	1	0	0	1	1	1	0	
0	0	0	1	1	0	0	0	0	
1	0	0	2	1	1	0	0	1	
0	0	0	0	1	0	1	1	0	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	

FIG. 4.6 Feature construction (text-specific)

4.2.2 Feature extraction

In feature extraction, new features are created from a combination of original features. Some of the commonly used operators for combining the original features include

1. For Boolean features: Conjunctions, Disjunctions, Negation, etc.

2. For nominal features: Cartesian product, M of N, etc.

3. For numerical features: Min, Max, Addition, Subtraction, Multiplication, Division, Average, Equivalence, Inequality, etc.

Let's take an example and try to understand. Say, we have a

- Dataset feature set $F_i(F_1, F_2, \ldots, F_n)$.

- After extraction → new set $G_i(G_1, G_2, \ldots, G_m)$
  such that G_i=f (F_i) where $m < n$.

- Example: G_1=f(F_1,F_2).

$Feat_A$	$Feat_B$	$Feat_C$	$Feat_D$
34	34.5	23	233
44	45.56	11	3.44
78	22.59	21	4.5
22	65.22	11	322.3
22	33.8	355	45.2
11	122.32	63	23.2

$Feat_1$	$Feat_2$
41.25	185.80
54.20	53.12
43.73	35.79
65.30	264.10
37.02	238.42
113.39	167.74

$$Feat_1 = 0.3 \times Feat_A + 0.9 \times Feat_A$$
$$Feat_2 = Feat_A + 0.5\ Feat_B + 0.6 \times Feat_C$$

FIG. 4.7 Feature extraction

Let's discuss the most popular feature extraction algorithms used in machine learning:

Principal Component Analysis

Every data set, as we have seen, has multiple attributes or dimensions – many of which might have similarity with each other. For example, the height and weight of a person, in general, are quite related. If the height is more, generally weight is more and vice versa. So if a data set has height and weight as two of the attributes, obviously they are expected to be having quite a bit of similarity. In general, any machine learning algorithm performs better as the number of related attributes or features reduced. In other words, a key to the success of machine learning lies in the fact that the features are less in number as well as the similarity between each other is very less. This is the main guiding philosophy of principal component analysis (PCA) technique of feature extraction.
In PCA, a new set of features are extracted from the original features which are quite dissimilar in nature. So an ndimensional feature space gets transformed to an mdimensional feature space, where the dimensions are orthogonal to each other, i.e. completely independent of each other. To understand the concept of orthogonality, we have to step back and do a bit of dip dive into vector space concept in linear algebra.

The objective of PCA is to make the transformation in such a way that

1. The new features are distinct, i.e. the covariance between the new features, i.e. the principal components is 0.

2. The principal components are generated in order of the variability in the data that it captures. Hence, the first principal component should capture the maximum variability, the second principal component should capture the next highest variability etc.

3. The sum of variance of the new features or the principal components should be equal to the sum of variance of the original features.

PCA works based on a process called eigenvalue decomposition of a covariance matrix of a data set. Below are the steps to be followed:

1. First, calculate the covariance matrix of a data set.
2. Then, calculate the eigenvalues of the covariance matrix.
3. The eigenvector having highest eigenvalue represents the direction in which there is the highest variance. So this will help in identifying the first principal component.
4. The eigenvector having the next highest eigenvalue represents the direction in which data has the highest remaining variance and also orthogonal to the first direction. So this helps in identifying the second principal component.
5. Like this, identify the top 'k' eigenvectors having top 'k' eigenvalues so as to get the 'k' principal components.

Singular value decomposition

Singular value decomposition (SVD) is a matrix factorization technique commonly used in linear algebra. SVD of a matrix A (m × n) is a factorization of the form:

$$A = U \sum V^T$$

where, U and V are orthonormal matrices, U is an $m \times m$ unitary matrix, V is an $n \times n$ unitary matrix and $\sum$ is an $m \times n$ rectangular diagonal matrix. The diagonal entries of $\sum$ are known as singular values of matrix A. The columns of U and V are called the left-singular and right-singular vectors of matrix A, respectively.

SVD is generally used in PCA, once the mean of each variable has been removed. Since it is not always advisable to remove the mean of a data attribute, especially when the data set is sparse (as in case of text data), SVD is a good choice for dimensionality reduction in those situations.
SVD of a data matrix is expected to have the properties highlighted below:

1. Patterns in the attributes are captured by the right-singular vectors, i.e. the columns of V.
2. Patterns among the instances are captured by the left-singular, i.e. the columns of U.
3. Larger a singular value, larger is the part of the matrix A that it accounts for and its associated vectors.
4. New data matrix with 'k' attributes is obtained using the equation

$$D^{'} = D \times [v_1, v_2, \ldots, v_k].$$

   Thus, the dimensionality gets reduced to k SVD is often used in the context of text data.

Linear Discriminant Analysis

Linear discriminant analysis (LDA) is another commonly used feature extraction technique like PCA or SVD. The objective of LDA is similar to the sense that it intends to transform a data set into a lower dimensional feature space. However, unlike PCA, the focus of LDA is not to capture the data set variability. Instead, LDA focuses on class separability, i.e. separating the features based on class separability so as to avoid over-fitting of the machine learning model.

Unlike PCA that calculates eigenvalues of the covariance matrix of the data set, LDA calculates eigenvalues and eigenvectors within a class and inter-class scatter matrices. Below are the steps to be followed:

1. Calculate the mean vectors for the individual classes.
2. Calculate intra-class and inter-class scatter matrices.
3. Calculate eigenvalues and eigenvectors for S_W^{-1} and S_B , where S_W is the intra-class scatter matrix and S_B is the inter-class scatter matrix

$$S_W = \sum_{i=1}^{c} S_i,$$

$$S_i = \sum_{x \in D_i}^{n} (x - m_i)(x - m_i)^T$$

   where, m_i is the mean vector of the i-th class

$$S_B = \sum_{i=1}^{c} N_i (m_i - m)(m_i - m)^T$$

   where, m_i is the sample mean for each class, m is the overall mean of the data set, N_i is the sample size of each class
4. Identify the top 'k' eigenvectors having top 'k' eigenvalues

4.3 Feature Subset Selection

Feature selection is arguably the most critical pre-processing activity in any machine learning project. It intends to select a subset of system attributes or features which makes a most meaningful contribution in a machine learning activity. Let's quickly discuss a practical example to understand the philosophy behind feature selection. Say we are trying to predict the weight of students based on past information about similar students, which is captured in a 'student weight' data set. The student weight data set has features such as Roll Number, Age, Height, and Weight. We can well understand that roll number can have no bearing, whatsoever, in predicting student weight. So we can eliminate the feature roll number and build a feature subset to be considered in this machine learning problem. The subset of features is expected to give better results than the full set. The same has been depicted in Figure 4.8.

But before we go forward with more detailed discussion on feature selection, let's try to understand the issues which have made feature selection such a relevant problem to be solved.

Roll Number	Age	Height	Weight
12	12	1.1	23
14	11	1.05	21.6
19	13	1.2	24.7
32	11	1.07	21.3
38	14	1.24	25.2
45	12	1.12	23.4

Age	Height	Weight
12	1.1	23
11	1.05	21.6
13	1.2	24.7
11	1.07	21.3
14	1.24	25.2
12	1.12	23.4

FIG. 4.8 Feature selection

Issues in high-dimensional data

'High-dimensional' refers to the high number of variables or attributes or features present in certain data sets, more so in the domains like DNA analysis, geographic information systems (GIS), social networking, etc. The high-dimensional spaces often have hundreds or thousands of dimensions or attributes, e.g. DNA microarray data can have up to 450,000 variables (gene probes).

In a large document corpus having few thousand documents embedded, the number of unique word tokens which represent the feature of the text data set, can also be in the range of a few tens of thousands. To get insight from such high-dimensional data may be a big challenge for any machine learning algorithm. On one hand, very high quantity of computational resources and high amount of time will be required. On the other hand the performance of the model – both for supervised and unsupervised machine learning task, also degrades sharply due to unnecessary noise in the data. Also, a model built on an extremely high number of features may be very difficult to understand. For this reason, it is necessary to take a subset of the features instead of the full set.

The objective of feature selection is three-fold:

- Having faster and more cost-effective (i.e. less need for computational resources) learning model
- Improving the efficiency of the learning model
- Having a better understanding of the underlying model that generated the data

Key drivers of feature selection – feature relevance and redundancy

Feature relevance

In supervised learning, the input data set which is the training data set, has a class label attached. A model is inducted based on the training data set – so that the inducted model can assign class labels to new, unlabelled data. Each of the predictor variables, is expected to contribute information to decide the value of the class label. In case a variable is not contributing any information, it is said to be irrelevant. In case the information contribution for prediction is very little, the variable is said to be weakly relevant. Remaining variables, which make a significant contribution to the prediction task are said to be strongly relevant variables.

In unsupervised learning, there is no training data set or labelled data. Grouping of similar data instances are done and similarity of data instances are evaluated based on the value of different variables. Certain variables do not contribute any useful information for deciding the similarity of dissimilarity of data instances. Hence, those variables make no significant information contribution in the grouping process. These variables are marked as irrelevant variables in the context of the unsupervised machine learning task.

To get a perspective, we can think of the simple example of the student data set that we discussed at the beginning of this section. Roll number of a student doesn't contribute any significant information in predicting what the Weight of a student would be. Similarly, if we are trying to group together students with similar academic capabilities, Roll number can really not contribute any information whatsoever. So, in context of the supervised task of predicting student Weight or the unsupervised task of grouping students with similar academic merit, the variable Roll number is quite irrelevant.

Feature redundancy

A feature may contribute information which is similar to the information contributed by one or more other features. For example, in the weight prediction problem referred earlier in the section, both the features Age and Height contribute similar information. This is because with an increase in Age,Weight is expected to increase. Similarly, with the increase of Height also Weight is expected to increase. Also, Age and Height increase with each other. So, in context of the Weight prediction problem, Age and Height contribute similar information. In other words, irrespective of whether the feature height is present as a part of the feature subset, the learning model will give almost same results. In the same way, without age being part of the predictor variables, the outcome of the learning model will be more or less same. In this kind of a situation when one feature is similar to another feature, the feature is said to be potentially redundant in the context of the learning problem.

All features having potential redundancy are candidates for rejection in the final feature subset. Only a small number of representative features out of a set of potentially redundant features are considered for being a part of the final feature subset.

So, in a nutshell, the main objective of feature selection is to remove all features which are irrelevant and take a representative subset of the features which are potentially redundant. This leads to a meaningful feature subset in context of a specific learning task.

Now, the question is how to find out which of the features are irrelevant or which features have potential redundancy. For that multiple measures are being used, some of which have been covered in the next sub-section.

Measures of feature relevance and redundancy

Measures of feature relevance

For supervised learning, mutual information is considered as a good measure of information contribution of a feature to decide the value of the class label. That's why it is a good indicator of the relevance of a feature with respect to the class variable. Higher the value of mutual information of a feature, more relevant is that feature. Mutual information can be calculated as follows:

$$MI(C, f) = H(C) + H(f) - H(C, f)$$

where, marginal entropy of the class, $H(C) = -\sum_{i=1}^{k} p(C_i) \log_2 p(C_i)$

marginal entropy of the feature 'x', $H(f) = -\sum_{c} p(f = x) \log_2 p(f = x)$

and K = number of classes, C = class variable, f = feature set that take discrete values.

In case of unsupervised learning, there is no class variable. Hence, feature-to-class mutual information cannot be used to measure the information contribution of the features. In case of unsupervised learning, the entropy of the set of features without one feature at a time is calculated for all the features. Then, the features are ranked in a descending order of information gain from a feature and top 'β' percentage (value of 'β' is a design parameter of the algorithm) of features are selected as relevant features. The entropy of a feature f is calculated using Shannon's formula below:

$$H(f) = -\sum_{x} p(f = x) \log_2 p(f = x)$$

$\sum_x$ is used only for features that take discrete values. For continuous features, it should be replaced by discretization performed first to estimate probabilities $p(f = x)$.

Measures of Feature redundancy

Feature redundancy, as we have already discussed, is based on similar information contribution by multiple features. There are multiple measures of similarity of information contribution, salient ones being

1. Correlation-based measures
2. Distance-based measures, and
3. Other coefficient-based measure

1. Correlation-based similarity measure
Correlation is a measure of linear dependency between two random variables. Pearson's product moment correlation coefficient is one of the most popular and accepted measures of correlation between two random variables. For two random feature variables F_1 and F_2 , Pearson correlation coefficient is defined as:

Let α be the correlation coefficient between two datasets F_1 and F_2. Then:

$$\alpha = \frac{\text{cov}(F_1, F_2)}{\sqrt{\text{var}(F_1) \cdot \text{var}(F_2)}}$$

where the covariance is defined as:

$$\text{cov}(F_1, F_2) = \sum_{i=1}^{n} (F_{1i} - \bar{F_1})(F_{2i} - \bar{F_2})$$

and the variances are:

$$\text{var}(F_1) = \sum_{i=1}^{n} (F_{1i} - \bar{F_1})^2, \quad \text{where } \bar{F_1} = \frac{1}{n} \sum_{i=1}^{n} F_{1i}$$

$$\text{var}(F_2) = \sum_{i=1}^{n} (F_{2i} - \bar{F_2})^2, \quad \text{where } \bar{F_2} = \frac{1}{n} \sum_{i=1}^{n} F_{2i}$$

Correlation values range between -1 and $+1$. Specifically:

- A correlation of $+1$ implies a perfect positive linear relationship.
- A correlation of -1 implies a perfect negative linear relationship.
- A correlation of 0 suggests **no linear relationship** between the features.

In the context of **feature selection**, correlation is often used to assess the similarity or redundancy between features. A threshold value is typically adopted to determine whether two features are sufficiently similar.

2. Distance-based similarity measure

The most common distance measure is the Euclidean distance, which, between two features F_1 and F_2 are calculated as:

$$d(F_1, F_2) = \sqrt{\sum_{i=1}^{n}(F_{1i} - F_{2i})^2}$$

where F_1 and F_1 are features of an n-dimensional data set.
Refer to the Figure 4.9. The data set has two features, aptitude (F_1) and communication (F_2) under consideration. The Euclidean distance between the features has been calculated using the formula provided above.

Aptitude (F_1)	Communication (F_2)	($F_1 - F_2$)	($F_1 - F_2$)^2
2	6	–4	16
3	5.5	–2.5	6.25
6	4	2	4
7	2.5	4.5	20.25
8	3	5	25
6	5.5	0.5	0.25
6	7	–1	1
7	6	1	1
8	6	2	4
9	7	2	4
			81.75

FIG. 4.9 Distance calculation between features

A more generalized form of the Euclidean distance is the **Minkowski** distance, measured as

$$d(F_1, F_2) = \sqrt{\sum_{i=1}^{n}(F_{1i} - F_{2i})^r}$$

- r=2 → Euclidean
- r=1 → Manhattan
- Special case: Hamming distance (binary vectors).

  For example, the Hamming distance between two vectors 01101011 and 11001001 is 3.

Overall feature selection process

Feature selection is the process of selecting a subset of features in a data set. As depicted in Figure 4.12, a typical feature selection process consists of four steps:

1. generation of possible subsets
2. subset evaluation
3. stop searching based on some stopping criterion
4. validation of the result

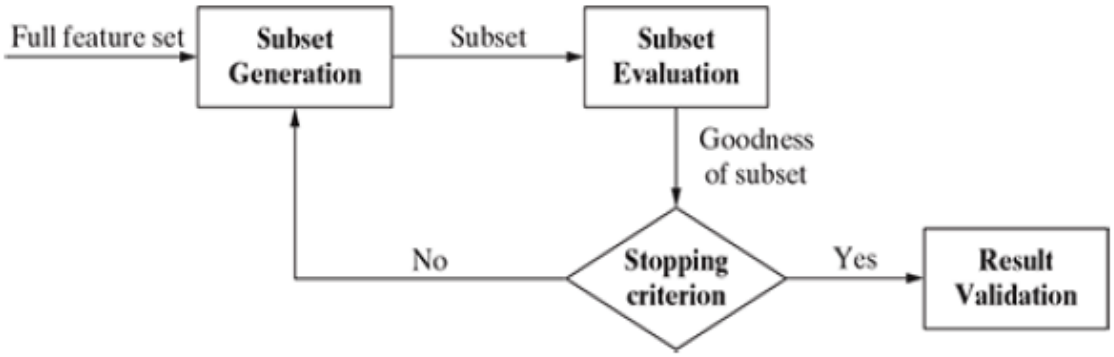


FIG. 4.12 Feature selection process

Subset generation, which is the first step of any feature selection algorithm, is a search procedure which ideally should produce all possible candidate subsets. However, for an n-dimensional data set, 2^n subsets can be generated. So, as the value of 'n' becomes high, finding an optimal subset from all the 2^n candidate subsets becomes intractable. For that reason, different approximate search strategies are employed to find candidate subsets for evaluation. On one hand, the search may start with an empty set and keep adding features. This search strategy is termed as a sequential forward selection. On the other hand, a search may start with a full set and successively remove features. This strategy is termed as sequential backward elimination. In certain cases, search start with both ends and add and remove features simultaneously. This strategy is termed as a bi-directional selection.

Each candidate subset is then evaluated and compared with the previous best performing subset based on certain **evaluation criterion**. If the new subset performs better, it replaces the previous one.

This cycle of subset generation and evaluation continues till a pre-defined stopping criterion is fulfilled. Some commonly used **stopping criteria** are

1. the search completes
2. some given bound (e.g. a specified number of iterations) is reached
3. subsequent addition (or deletion) of the feature is not producing a better subset
4. a sufficiently good subset (e.g. a subset having better classification accuracy than the existing benchmark) is selected

Then the selected best subset is **validated** either against prior benchmarks or by experiments using real-life or synthetic but authentic data sets. In case of supervised learning, the accuracy of the learning model may be the performance parameter considered for validation. The accuracy of the model using the subset derived is compared against the model accuracy of the subset derived using some other benchmark algorithm. In case of unsupervised, the cluster quality may be the parameter for validation.

Feature selection approaches

There are four types of approach for feature selection:

1. Filter approach
2. Wrapper approach
3. Hybrid approach
4. Embedded approach

In the **filter approach** (as depicted in Fig. 4.13), the feature subset is selected based on statistical measures done to assess the merits of the features from the data perspective. No learning algorithm is employed to evaluate the goodness of the feature selected. Some of the common statistical tests conducted on features as a part of filter approach are – Pearson's correlation, information gain, Fisher score, analysis of variance (ANOVA), Chi-Square, etc.

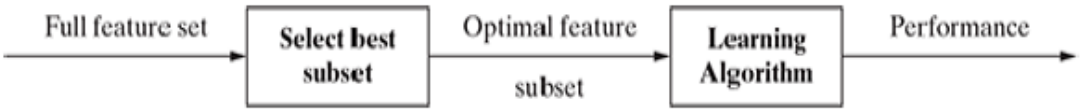


FIG. 4.13 Filter approach

In the **wrapper approach** (as depicted in Fig. 4.14), identification of best feature subset is done using the induction algorithm as a black box. The feature selection algorithm searches for a good feature subset using the induction algorithm itself as a part of the evaluation function. Since for every candidate subset, the learning model is trained and the result is evaluated by running the learning algorithm, wrapper approach is computationally very expensive. However, the performance is generally superior compared to filter approach.

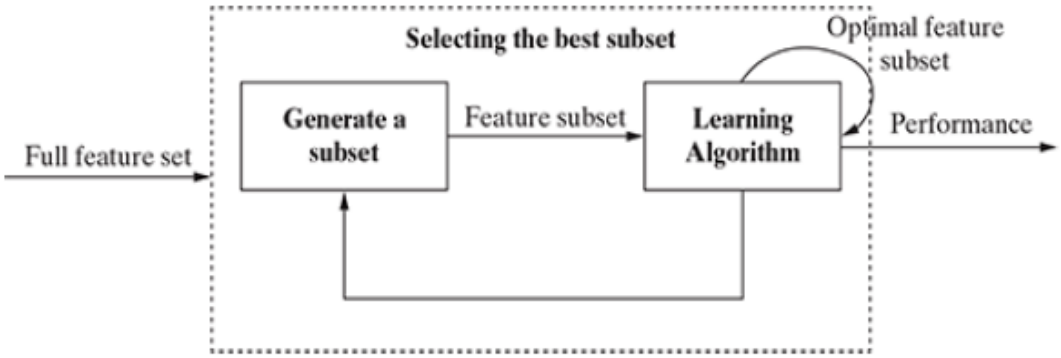


FIG. 4.14 Wrapper approach

Hybrid approach takes the advantage of both filter and wrapper approaches. A typical hybrid algorithm makes use of both the statistical tests as used in filter approach to decide the best subsets for a given cardinality and a learning algorithm to select the final best subset among the best subsets across different cardinalities.

Embedded approach (as depicted in Fig. 4.15) is quite similar to wrapper approach as it also uses and inductive algorithm to evaluate the generated feature subsets. However, the difference is it performs feature selection and classification simultaneously.

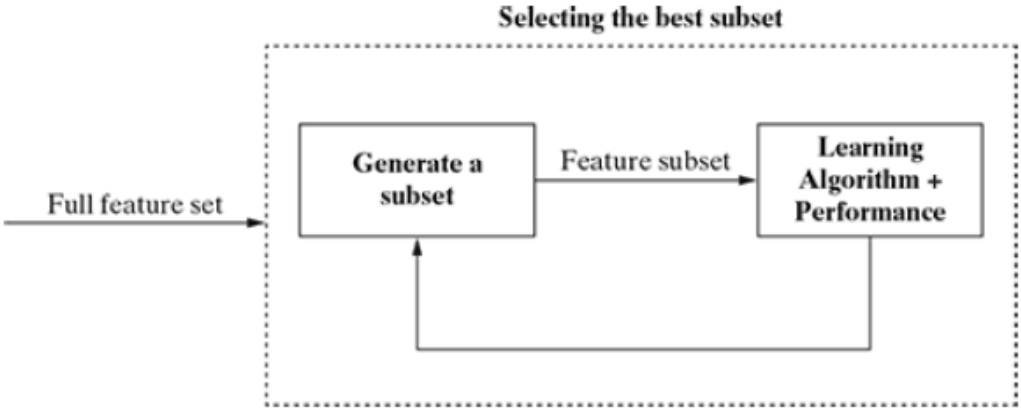


FIG. 4.15 Embedded approach

Chapter 5

A Brief Review of Probability Theory

5.1 Introduction

Sample Space (S): The set of all possible outcomes.

$$S = \{\text{all possible outcomes}\}$$

Event (A): Any subset of the sample space.

$$A \subseteq S$$

Probability Measure: $P : S \to [0, 1]$

$$P(S) = 1 \text{ (total certainty)}$$
$$0 \leq P(A) \leq 1$$

5.1.1 Fundamental Rules

Addition Rule:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

Complement Rule:

$$P(A^c) = 1 - P(A)$$

5.1.2 Joint Probability

Joint Probability: Probability that A and B both occur.

$$P(A \cap B)$$

Independence: If A and B are independent,

$$P(A \cap B) = P(A)P(B)$$

5.1.3 Conditional Probability

Definition: Probability of A given B.

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \qquad P(B) > 0$$

5.1.4 Bayes' Rule

Bayes' Theorem: Relates conditional probabilities.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Where:

- $P(A|B)$: Probability of A given B occurred.
- $P(B|A)$: Probability of B given A occurred.
- $P(A)$, $P(B)$: Prior probabilities.

5.2 Random Variables

A Random Variable (RV) assigns a numerical value to each outcome of a random experiment. A random variable X is a function from the sample space S to $\mathbb{R}$, X: S $\to \mathbb{R}$. Example: X(H) = 1, X(T) = 0

Types:
1.Discrete Random Variable:

A random variable X is said to be *discrete* if it takes only a finite or countably infinite set of values $x_1, x_2, x_3, \ldots$.

The probability distribution of X is defined by its **Probability Mass Function (PMF)**:

$$P(X = x_i) = p(x_i), \quad i = 1, 2, 3, \ldots$$

subject to the following conditions:

$$p(x_i) \geq 0 \quad \text{for all } i, \quad \text{and} \quad \sum_i p(x_i) = 1.$$

The corresponding **Cumulative Distribution Function (CDF)** is:

$$F(x) = P(X \leq x) = \sum_{t \leq x} p(t).$$

Example: If X denotes the number of heads in a single coin toss, then

$$X = \begin{cases} 1, & \text{if Head occurs,} \\ 0, & \text{if Tail occurs.} \end{cases}$$

and

$$P(X = 1) = 0.5, \quad P(X = 0) = 0.5.$$

2.Continuous Random Variable:

A random variable X is said to be *continuous* if it can take any value in an interval of real numbers.

It is described by a **Probability Density Function (PDF)** $f(x)$, such that:

$$P(a \leq X \leq b) = \int_a^b f(x)\, dx$$

and

$$\int_{-\infty}^{\infty} f(x)\, dx = 1.$$

The corresponding **Cumulative Distribution Function (CDF)** is defined as:

$$F(x) = P(X \leq x) = \int_{-\infty}^{x} f(t)\, dt.$$

Since $f(x) \geq 0$ for all x, the CDF is a non-decreasing function satisfying:

$$\lim_{x \to -\infty} F(x) = 0, \quad \lim_{x \to \infty} F(x) = 1.$$

Example: If $X \sim N(\mu, \sigma^2)$, then

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

5.3 Discrete Distributions

5.3.1 Bernoulli Distribution

When we have a situation where the outcome of a trial is withered 'success' or 'failure', then the behaviour of the random variable X can be represented by Bernoulli distribution. Such trials or experiments are termed as Bernoulli trials. A random variable X is called Bernoulli random variable with parameter p when its pmf takes the form of

$P(X = 1) = p$, $P(X = 0) = 1 - p$. The mean and variance of Bernoulli random variable X are $\mu = p$, $\sigma^2 = p(1 - p)$

5.3.2 Binomial Distribution

If n independent Bernoulli trials are performed and X represents the number of success in those n trials, then X is called a binomial random variable. That's the reason a Bernoulli random variable is a special case of binomial random variable with parameters (1, p). The pmf of X with parameters (n, p) is given by $P(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}$
This distribution has the following mean and variance: $\mu = np$, $\sigma^2 = np(1 - p)$.

5.3.3 Poisson Distribution

Poisson random variable has a wide range of application as it may be used as an approximation for binomial with parameter (n,p) when n is large and p is small and thus np is of moderate size. An example application area is, if a fax machine has a faulty transmission line, then the probability of receiving an erroneous digit within a certain page transmitted can be calculated using a Poisson random variable. So, a random variable X is called a Poisson random variable with parameter λ (> 0) when the pmf looks like $P(X = k) = \frac{e^{-\lambda}\lambda^k}{k!}$ where $k = 0, 1, 2, ...$
The Mean and variance of the distribution are λ.

5.4 Continuous Distributions

5.4.1 Uniform Distribution

The pdf of a uniform random variable is given by: $f(x) = \frac{1}{b-a}$, $a \leq x \leq b$
The mean and variance of a uniform random variable X are $\mu = (a+b)/2$, $\sigma^2 = (b-a)^2/12$

5.4.2 Normal Distribution

The most widely used distribution in statistics and machine learning is the Gaussian or normal distribution. Its pdf is given by $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
For a normal random variable, mean and variance are μ and σ^2.
A standard normal random variable is defined as the one whose mean is 0 and variance is 1 which means $Z = N(0; 1)$.

5.4.3 The Laplace Distribution

Another distribution with heavy tails is the Laplace distribution, which is also known as the double-sided exponential distribution. This has the following pdf: $f(x; \mu, b) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right)$, here μ is a location parameter and $b > 0$ is a scale parameter.

5.5 Multiple Random Variables

5.5.1 Bivariate random variables

Let us consider two random variables X and Y in the sample space of S of a random experiment. Then the pair (X, Y) is called s bivariate random variable or two-dimensional random vector where each of X and Y are associated with a real number for every element of S.

5.5.2 Joint distribution functions

The joint cumulative distribution function (or joint cdf) of X and Y is defined as: $F_{XY}(x, y) = P(X \leq x, Y \leq y)$. For certain values of x and y, if A and B are independent events of S, then $F_{XY}(x, y) = F_X(x)F_Y(y)$.
Few important properties of joint cdf of two random variables which are similar to that of the cdf of single random variable are

5.5.3 Joint probability mass functions

For the discrete bivariate random variable (X, Y) if it takes the values (x , y) for certain allowable integers i and j, then the joint probability mass function (joint pmf) of (X, Y) is given by $p_{XY}(x_i, y_j) = P(X = x_i, Y = y_j)$.

5.5.4 Joint probability density functions

In case (X, Y) is a continuous bivariate random variable with cdf $F_{XY}(x, y)$ and the function

$$f_{XY}(x, y) = \frac{\partial^2 F_{XY}(x, y)}{\partial x \, \partial y}$$

is called the joint probability density function (joint pdf) of (X, Y). Thus, integrating, we get

$$F_{XY}(x, y) = \int_{-\infty}^{x} \int_{-\infty}^{y} f_{XY}(\xi, \eta) \, d\eta \, d\xi$$

A few important properties of $f_{XY}(x, y)$ are:

1. $f_{XY}(x, y) \geq 0$
2. $$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f_{XY}(x, y) \, dx \, dy = 1$$
3. $$P\big((X, Y) \in A\big) = \int_{R_A} f_{XY}(x, y) \, dx \, dy$$

5.5.5 Conditional distributions

While working with a discrete bivariate random variable, it is important to deduce the conditional probability function as X and Y are related in the finite space. Based on the joint pmf of (X, Y), the conditional pmf of Y when $X = x_i$ is defined as

$$P_{Y|X}(y_j|x_i) = \frac{P_{XY}(x_i, y_j)}{P_X(x_i)} \quad \text{when } P_X(x_i) > 0$$

and

$$P_{X|Y}(x_i|y_j) = \frac{P_{XY}(x_i, y_j)}{P_Y(y_j)} \quad \text{when } P_Y(y_j) > 0$$

In the same way, when (X, Y) is a continuous bivariate random variable and the joint pdf is $f_{XY}(x, y)$, then the conditional pdf of Y in case $X = x$ is defined as

$$f_{Y|X}(y|x) = \frac{f_{XY}(x, y)}{f_X(x)} \quad \text{when } f_X(x) > 0$$

and

$$f_{X|Y}(x|y) = \frac{f_{XY}(x, y)}{f_Y(y)} \quad \text{when } f_Y(y) > 0$$

5.5.6 Covariance and Correlation

The covariance between two random variables X and Y measure the degree to which X and Y are (linearly) related, which means how X varies with Y and vice versa. $\text{Cov}(X, Y) = E[(X - E[X])(Y - E[Y])] = E(XY) - E(X)E(Y)$. So, if the variance is the measure of how a random variable varies with itself, then the covariance is the measure of how two random variables vary with each other.
The correlation $\rho_{XY} = \frac{\text{Cov}(X,Y)}{\sigma_X \sigma_Y}$ where σ_X^2 and σ_Y^2 are variance of X and Y.

- Few important properties of correlation are:
  1. $-1 \leq \text{corr}[X, Y] \leq 1$. Hence, in a correlation matrix, each entry on the diagonal is 1, and the other entries are between -1 and 1.
  2. $\text{corr}[X, Y] = 1$ if and only if $Y = aX + b$ for some parameters a and b, i.e., if there is a *linear* relationship between X and Y.
  3. From property (2), it may seem that the correlation coefficient is related to the slope of the regression line, i.e., the coefficient a in the expression

     $$Y = aX + b$$

     However, the regression coefficient is in fact given by

     $$a = \frac{\text{Cov}[X, Y]}{\text{Var}[X]}.$$

A better way to interpret the correlation coefficient is as a degree of linearity.

5.6 Central Limit Theorem (CLT)

Let $X_1, X_2, X_3, \ldots, X_n$ be a sequence of independent and identically distributed (i.i.d.) random variables with

$$E[X_i] = \mu \quad \text{and} \quad Var(X_i) = \sigma^2 < \infty.$$

Define the sample mean as

$$\bar{X}_n = \frac{1}{n} \sum_{i=1}^{n} X_i.$$

Then, as $n \to \infty$, the standardized form of $\bar{X}_n$ converges in distribution to the standard normal distribution:

$$Z = \frac{\bar{X}_n - \mu}{\sigma/\sqrt{n}} \xrightarrow{d} N(0,1).$$

Equivalently, for any real number z,

$$\lim_{n\to\infty} P\left(\frac{\bar{X}_n - \mu}{\sigma/\sqrt{n}} \leq z\right) = \Phi(z),$$

where $\Phi(z)$ is the cumulative distribution function (CDF) of the standard normal distribution.

Chapter 6

Bayesian Concept Learning

In this chapter, we discuss the fundamentals of Bayesian learning and explain how Bayes' theorem forms the theoretical foundation for several machine learning algorithms. Bayesian learning methods provide a principled way of reasoning under uncertainty and enable systematic updating of beliefs as new data becomes available.

The technique is derived from the work of the 18th-century mathematician **Thomas Bayes**, who developed the mathematical framework that describes how probabilities should be revised in the presence of new evidence.

6.1 Why Bayesian Methods are Important

Bayesian learning algorithms, such as the Naïve Bayes classifier, are highly practical for many real-world learning problems because they compute *explicit probabilities* for hypotheses.

- Bayesian classifiers estimate the probability of each class using observed feature values from training data.
- For unseen instances, these probabilities are used to predict the most likely class.
- The use of prior knowledge allows Bayesian models to align better with real-life uncertainty.

Applications of Bayesian Learning

- Text classification (spam filtering, author identification, topic categorization)
- Medical diagnosis based on observed symptoms
- Network security and anomaly detection

6.2 Bayes' Theorem

Bayes' theorem describes the relationship between conditional probabilities:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{6.1}$$

where A and B are events such that B has already occurred.

In machine learning, Bayes' theorem is used to determine the best hypothesis h from a hypothesis space H, given training data T.

6.2.1 Prior, Likelihood and Posterior

Prior Probability: $P(h)$ represents the initial belief about hypothesis h before observing data.

Likelihood: $P(T|h)$ denotes the probability of observing the data T assuming hypothesis h is true.

Posterior Probability: $P(h|T)$ represents the probability that hypothesis h is true after observing the data T.

According to Bayes' theorem:

$$P(h|T) = \frac{P(T|h)P(h)}{P(T)} \tag{6.2}$$

6.2.2 Maximum A Posteriori (MAP) Hypothesis

The hypothesis with the highest posterior probability is called the **Maximum A Posteriori (MAP)** hypothesis:

$$h_{MAP} = \arg\max_{h \in H} P(h|T) \tag{6.3}$$

$$= \arg\max_{h \in H} P(T|h)P(h) \tag{6.4}$$

If all hypotheses have equal prior probability, then MAP reduces to **Maximum Likelihood (ML)** estimation:

$$h_{ML} = \arg\max_{h \in H} P(T|h) \tag{6.5}$$

6.3 Illustrative Example: Tumour Diagnosis

Let:

- MT: tumour is malignant
- PT: positive laboratory test

Given:

$$P(MT) = 0.005, \quad P(!MT) = 0.995$$
$$P(PT|MT) = 0.98, \quad P(PT|!MT) = 0.02$$

Posterior probabilities:

$$P(MT|PT) = 0.98 \times 0.005 = 0.0049$$
$$P(!MT|PT) = 0.02 \times 0.995 = 0.0199$$

Since $P(!MT|PT) > P(MT|PT)$, the MAP hypothesis predicts a **non-malignant tumour**.

6.4 Bayes Optimal Classifier

The Bayes optimal classifier predicts the class with the highest probability averaged over all hypotheses:

$$P(c_i|T) = \sum_{h \in H} P(c_i|h)P(h|T) \tag{6.6}$$

$$\text{Bayes Optimal Classifier} = \arg\max_{c_i \in C} \sum_{h \in H} P(c_i|h)P(h|T) \tag{6.7}$$

6.4.1 Example

If hypotheses have posterior probabilities $0.4, 0.3, 0.3$ and only h_1 predicts True, then:

$$P(True|T) = 0.4$$
$$P(False|T) = 0.6$$

Thus, the Bayes optimal prediction is **False**.

6.5 Naïve Bayes Classifier

The Naïve Bayes classifier is a supervised learning algorithm based on Bayes' theorem with a strong assumption of conditional independence among features.

6.5.1 Key Characteristics

- Requires very little training data
- Assumes independence among attributes
- Computationally efficient
- Performs well in text classification

6.5.2 Naïve Bayes Classification Steps

1. Construct frequency tables for attributes
2. Compute posterior probabilities using Bayes' theorem
3. Normalize probabilities

6.6 Bayesian Belief Networks

Bayesian Belief Networks relax the strong independence assumption of Naïve Bayes by allowing conditional dependencies among variables.

- Represented using directed acyclic graphs (DAGs)
- Nodes represent random variables
- Edges represent causal or probabilistic dependencies

6.6.1 Independence and Conditional Independence

Variables A and B are independent if:

$$P(A|B) = P(A)$$

The generalized chain rule for joint probabilities is:

$$P(A_1, \ldots, A_n) = \prod_{i=1}^{n} P(A_i|A_{i+1}, \ldots, A_n) \tag{6.8}$$

6.7 Applications of Bayesian Belief Networks

Bayesian networks are widely used in:

- Bioinformatics and gene regulation
- Medical diagnosis
- Image processing
- Decision support systems
- Information retrieval and document classification

Summary

Bayesian learning provides a mathematically sound framework for handling uncertainty in machine learning. Techniques such as MAP estimation, Naïve Bayes, and Bayesian Belief Networks remain essential tools in probabilistic modeling and inference.

Chapter 7

Supervised Learning Classification

7.1 Classification vs Regression

Classification: predicts a categorical label. **Regression:** predicts a continuous numeric value.

Key differences:

- Output: classes vs numbers.
- Metrics: accuracy/F1 vs MSE/MAE.
- Models: logistic/softmax for classification, linear regression for regression.
- Classification handles class imbalance; regression focuses on error patterns.

Choose: category ⇒ classification; number ⇒ regression.

7.1.1 Applications of Classification

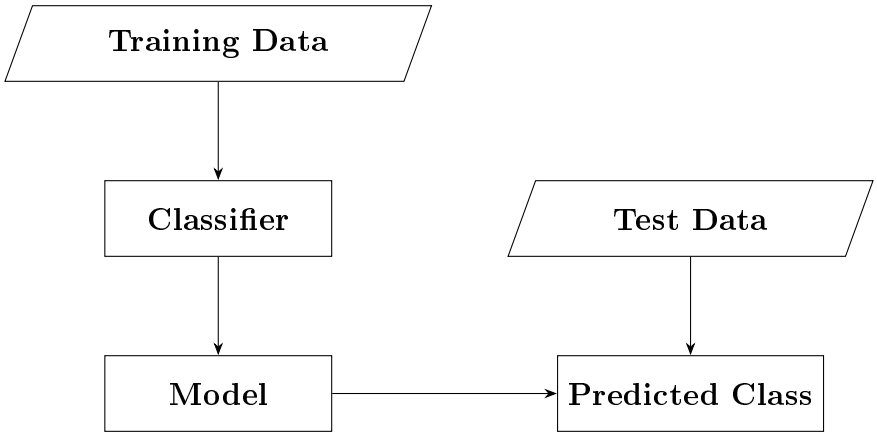
- Healthcare: disease diagnosis.
- Finance: loan approval, fraud detection.
- NLP: spam, sentiment analysis.
- Vision: image/object classification.
- Cybersecurity: intrusion detection.

7.1.2 Classification Learning Steps

1. Define problem and classes.
2. Collect required data.
3. Preprocess (clean, encode, scale).
4. Split data (train/validation/test).

5. Select algorithm.
6. Train + tune hyperparameters.
7. Evaluate and deploy.

7.1.3 Classification Model



7.2 k-Nearest Neighbours (kNN)

What is kNN?

kNN is a simple, non-parametric, instance-based algorithm. It predicts the class or value of a new point by finding its k nearest neighbours and using **majority vote** (classification) or **average** (regression). No model is built; the training data itself is the model.

Algorithm (Step-by-Step)

1. Choose k and a distance metric.
2. For each test point:
   (a) Compute distance to all training samples.
   (b) Select the k closest points.
   (c) Use majority vote or averaging to predict.

Pseudocode

```
function KNN_Predict(test_point, training_set, k):
    distances = []
    for each (x_i, y_i) in training_set:
        d = distance(test_point, x_i)
        append (d, y_i) to distances
```

```
sort distances by d ascending
take first k entries -> neighbors
prediction = majority_vote(neighbors)
return prediction
```

Why kNN is a "Lazy Learner"

- **No training model**: kNN stores the full dataset without building a model.
- **Computation delayed**: distance calculations happen only during prediction.
- **Result:** low training cost, but slow prediction as data size grows.

7.2.1 Worked mathematical example (step-by-step arithmetic)

Problem: 2D training points (features: Aptitude, Communication) with class labels are:

$$\begin{array}{ll} A(1.0,\ 4.0) & \text{class} = \text{Leader} \\ B(2.0,\ 3.5) & \text{class} = \text{Leader} \\ C(3.5,\ 1.0) & \text{class} = \text{Intel} \\ D(4.0,\ 1.5) & \text{class} = \text{Intel} \end{array}$$

Test point: $T(2.5,\ 2.5)$. Use Euclidean distance and $k = 3$.

We compute Euclidean distance $d((x, y), (u, v)) = \sqrt{(x-u)^2 + (y-v)^2}$.

Distance from T to A:

$$d(T, A) = \sqrt{(2.5-1.0)^2 + (2.5-4.0)^2} = \sqrt{(1.5)^2 + (-1.5)^2} = \sqrt{2.25 + 2.25} = \sqrt{4.50} \approx 2.121$$

Distance from T to B:

$$d(T, B) = \sqrt{(2.5-2.0)^2 + (2.5-3.5)^2} = \sqrt{(0.5)^2 + (-1.0)^2} = \sqrt{0.25 + 1.00} = \sqrt{1.25} \approx 1.118$$

Distance from T to C:

$$d(T, C) = \sqrt{(2.5-3.5)^2 + (2.5-1.0)^2} = \sqrt{(-1.0)^2 + (1.5)^2} = \sqrt{1.00 + 2.25} = \sqrt{3.25} \approx 1.803$$

Distance from T to D:

$$d(T, D) = \sqrt{(2.5-4.0)^2 + (2.5-1.5)^2} = \sqrt{(-1.5)^2 + (1.0)^2} = \sqrt{2.25 + 1.00} = \sqrt{3.25} \approx 1.803$$

Sort distances (ascending):

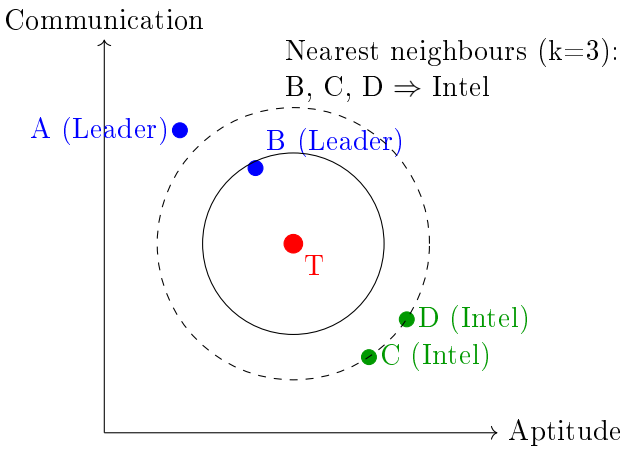
$$B : 1.118, \quad C : 1.803, \quad D : 1.803, \quad A : 2.121$$

Select $k = 3$ nearest neighbours: $\{B, C, D\}$. Their classes: $\{\text{Leader}, \text{Intel}, \text{Intel}\}$.

Majority vote ⇒ predicted class for T is Intel.

(If we used $k = 1$ the prediction would be **Leader** since B is the nearest.)

7.2.2 Intuition diagram (2D) and explanation



Explanation: The test point T lies roughly halfway between the blue (Leader) and green (Intel) groups. With $k = 3$, two of the three nearest neighbours are `Intel`, so kNN predicts `Intel`. Changing k or weighting by inverse distance can change the outcome.

7.2.3 Strengths and Weaknesses

Strengths:

- Very simple and intuitive.
- Non-parametric: makes no strong assumptions about data distribution.
- Naturally handles multi-class problems.

Weaknesses:

- Prediction time can be slow for large datasets (distance computation to all points).
- Requires feature scaling; sensitive to irrelevant features.
- Memory intensive: must store full training set.
- Choice of k critical; susceptible to class imbalance.

7.2.4 Applications of k-Nearest Neighbours (kNN)

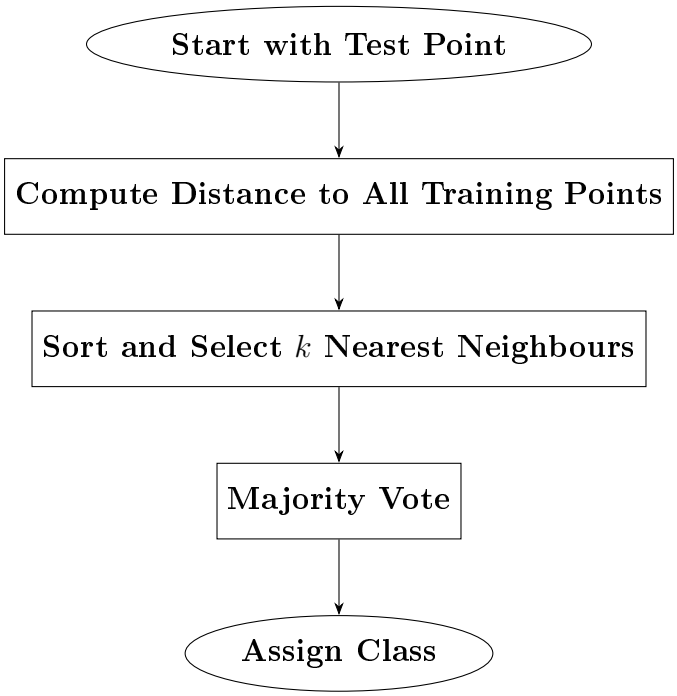
kNN is a simple yet effective algorithm widely used for:

- **Classification:** disease detection, handwriting/face recognition.
- **Recommender Systems:** movie and product recommendations.

- **Pattern Recognition:** gesture, voice, and speech classification.
- **Finance:** credit scoring, fraud detection.
- **Anomaly Detection:** network intrusion and fault detection.
- **Text Classification:** spam filtering, sentiment analysis.

kNN works best for small/medium datasets, irregular decision boundaries, and interpretable neighbour-based decisions.

7.2.5 kNN Flowchart



7.3 Decision Trees

What is a Decision Tree?

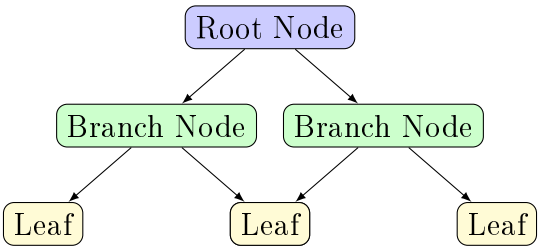
A **Decision Tree** is a supervised learning model that predicts outcomes using a hierarchical structure of **nodes** and **branches**. Each internal node represents a **test on a feature**, each branch represents a **possible outcome of the test**, and each leaf node corresponds to a **final class label or predicted value**. Decision Trees work by recursively splitting the dataset into smaller subsets in a way that **maximizes class separation** using criteria such as Gini impurity or information gain.

Key properties of a decision tree:

- Simple and interpretable (tree-like structure).
- Handles categorical and numerical attributes.

- Non-parametric (makes no assumption about data distribution).

General Structure



Each path from the root to a leaf node represents a classification rule.

Example: Car Driving Decision Tree

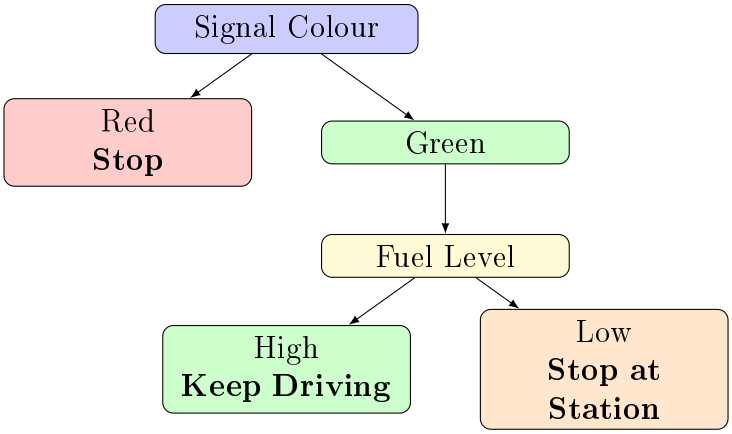
We create a simple decision tree to answer the question:

"Should I continue driving or stop?"

Attributes:

- Signal Colour (Red / Green)
- Fuel Level (Low / High)
- Road Condition (Clear / Blocked)

Decision Tree Diagram (Car Driving Example)



7.3.1 Steps in Building a Decision Tree

- **1. Select Best Attribute:** Choose feature with highest Information Gain. *Car example: Signal Colour → Red implies Stop (root).*
- **2. Split Data:** Divide samples by attribute values. *Red → Stop, Green → check next feature.*

- **3. Check Purity:** If all samples are same class → leaf. *Red branch becomes pure.*
- **4. Recurse:** Repeat for impure branches. *Green → evaluate Fuel Level.*
- **5. Stop Splitting:** No improvement or attributes left. *Fuel Level gives clear split.*
- **6. Assign Labels:** Label leaves by majority class. *Green + High Fuel → Drive; Green + Low Fuel → Stop at Station.*

7.3.2 Strengths of Decision Trees

- Easy to interpret and visualize.
- Works with numerical and categorical data.
- Requires little preprocessing.
- Fast during prediction.

7.3.3 Weaknesses of Decision Trees

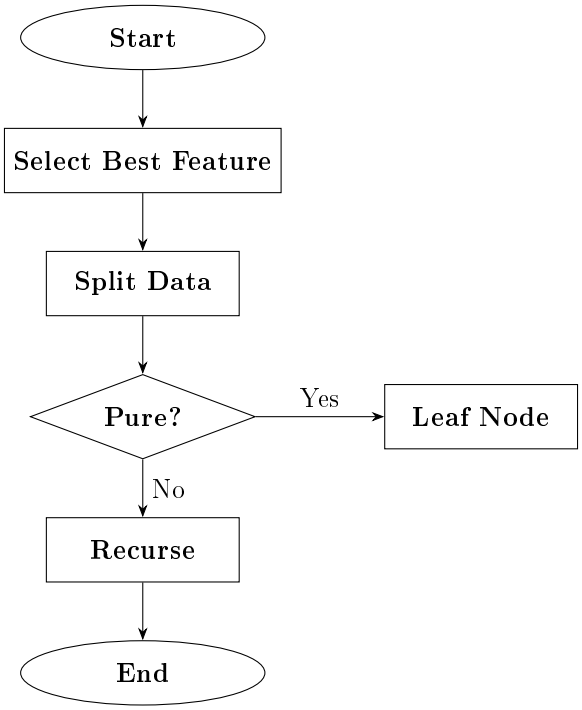
- Prone to overfitting.
- Small changes in data can drastically change the tree.
- Greedy splitting may not yield globally optimal trees.

7.3.4 Applications of Decision Trees

Decision trees are widely used because they are simple, interpretable, and rule-based. Key applications include:

- **Healthcare:** disease prediction, risk assessment.
- **Business:** credit scoring, marketing, fraud detection.
- **Engineering:** fault detection, predictive maintenance.
- **Autonomous Systems:** obstacle and traffic sign classification.
- **Environment:** weather and disaster risk prediction.
- **Agriculture:** crop disease and yield prediction.
- **Cybersecurity:** intrusion and malware detection.
- **Education:** student performance and dropout prediction.
- **Text Analytics:** sentiment and document classification.

7.3.5 Decision Tree Building Flowchart



7.3.6 Building a Decision Tree(Entropy and Information Gain)

Step-by-Step Procedure with Term Explanations

1. Compute the **Entropy** of the target attribute:

$$E(S) = -\sum_{i=1}^{c} p_i \log_2 p_i$$

   - **E(S)**: Entropy of dataset S, measuring impurity.
   - $\mathbf{p_i}$: Proportion of class i in dataset S.
   - c: Number of classes.

2. For each attribute A, compute the **Information Gain (IG)**:

$$IG(A) = E(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} E(S_v)$$

   - **IG(A)**: Reduction in entropy after splitting on A.
   - **Values(A)**: Possible values of attribute A.
   - $\mathbf{S_v}$: Subset of S with attribute value v.
   - $|\mathbf{S_v}|$: Number of tuples in S_v.
   - $|\mathbf{S}|$: Total samples in dataset S.

3. Choose the attribute with the **highest Information Gain**.
4. Split the dataset based on the chosen attribute.
5. Repeat the process recursively for each branch until:

- All tuples belong to the same class, or
- There are no remaining attributes.

7.3.7 Avoiding Overfitting (Pruning)

Overfitting in Decision Trees

A decision tree may become too complex and capture noise or random fluctuations in the training data. This leads to poor generalization on new data.

Pruning Techniques:

i. **Pre-pruning (Early Stopping):** Stop growing the tree before it perfectly classifies the training data. Example: stop when information gain becomes too small or when minimum node size is reached.

ii. **Post-pruning:** Grow the full tree first, then remove branches that do not improve validation accuracy. This simplifies the model and reduces overfitting.

7.3.8 Strengths and Weaknesses

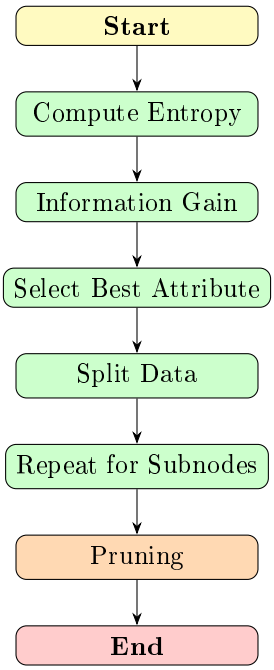
Strengths

- Easy to interpret and visualize.
- Handles both numerical and categorical data.
- Requires little data preprocessing.
- Nonlinear relationships between features do not affect performance.

Weaknesses

- Prone to overfitting if not pruned.
- Small changes in data can result in a different tree.
- Biased toward attributes with more levels.

7.3.9 Summary Diagram



7.3.10 Applications of Decision Trees

- **Medical diagnosis** (disease prediction, treatment recommendation)
- **Credit scoring and loan approval** (risk assessment)
- **Fraud detection** (anomaly detection)
- **Customer segmentation** (targeted marketing)
- **Weather prediction** (rainfall, storm classification)
- **Manufacturing fault detection** (quality control)
- **Supply chain optimization** (inventory decisions)
- **Retail analytics** (product recommendation)
- **Human resource analytics** (attrition prediction)
- **Agriculture** (crop yield, soil classification)
- **Cybersecurity** (intrusion detection)
- **Energy sector** (load forecasting)
- **Education** (student performance prediction)
- **Real estate** (property price prediction)

7.3.11 Example: Building a Decision Tree (Two Levels)

Objective: Construct a two-level decision tree using the PlayTennis dataset.

7.3.12 Training Dataset

Outlook	Temp	Humidity	Wind	Play
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Step 1: Entropy of Target

There are 9 Yes and 5 No.

$$E(S) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940$$

Step 2: Information Gain

(i) Outlook (Sunny, Overcast, Rain)

Sunny: 2 Yes, 3 No

$$E = 0.971$$

Overcast: 4 Yes $\rightarrow E = 0$

Rain: 3 Yes, 2 No

$$E = 0.971$$

$$E(\text{Outlook}) = \frac{5}{14}(0.971) + \frac{4}{14}(0) + \frac{5}{14}(0.971) = 0.693$$

$$IG(\text{Outlook}) = 0.940 - 0.693 = 0.247$$

(ii) Temperature

Hot: 2Y/2N, Mild: 4Y/2N, Cool: 3Y/1N

$$E(\text{Temp}) = 0.911, \quad IG = 0.029$$

(iii) Humidity

High: 3Y/4N, Normal: 6Y/1N

$$E(\text{Humidity}) = 0.789, \quad IG = 0.151$$

(iv) Wind

Weak: 6Y/2N, Strong: 3Y/3N

$$E(\text{Wind}) = 0.892, \quad IG = 0.048$$

Step 3: Root Selection

$$IG(\text{Outlook}) = 0.247 = \max$$

Root attribute: Outlook

Step 4: Level-2 Splits

1. Outlook = Overcast All 4 = Yes → pure leaf.

2. Outlook = Sunny

Sunny subset:

Temp	Humidity	Wind	Play
Hot	High	Weak	No
Hot	High	Strong	No
Mild	High	Weak	No
Cool	Normal	Weak	Yes
Mild	Normal	Strong	Yes

Entropy(Sunny) = 0.971

Humidity check: High → 3 No → $E = 0$, Normal → 2 Yes → $E = 0$

$$IG(\text{Humidity}) = 0.971$$

Final split: High → No , Normal → Yes

3. Outlook = Rain

Rain subset:

Temp	Humidity	Wind	Play
Mild	High	Weak	Yes
Cool	Normal	Weak	Yes
Cool	Normal	Strong	No
Mild	Normal	Weak	Yes
Mild	High	Strong	No

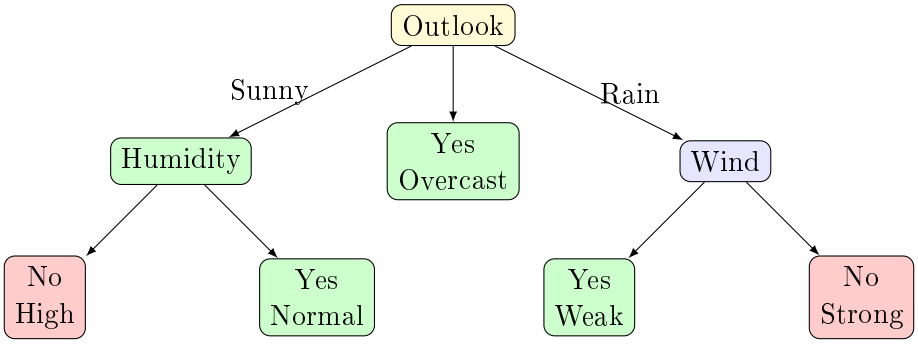
Entropy(Rain) = 0.971

Wind: Weak → 3 Yes → $E = 0$, Strong → 2 No → $E = 0$

$$IG(\text{Wind}) = 0.971$$

Final split: Weak → Yes , Strong → No

Final Two-Level Decision Tree



Result: Outlook is the root. Sunny → Humidity, Overcast → Yes, Rain → Wind. All leaf nodes are pure (entropy = 0).

7.3.13 Algorithm (Simplified Entropy-Based)

Input: Training data set, test data set (or data points)

Steps:

1. Compute total entropy $E(S)$ of the dataset.
2. For each attribute A_i:
   - Compute entropy E_i for each possible value of A_i.
   - Compute information gain $IG(A_i) = E(S) - E_i$.
   - If $E_i < E_{\min}$, set A_i as the best attribute.
3. Select the attribute with the maximum $IG(A_i)$ as the root node.
4. Partition the dataset accordingly and repeat for sub-nodes until pure or stopping criteria are met.

7.4 Random Forest: Ensemble of Decision Trees

What is Random Forest?

Random Forest is an **ensemble method** that combines multiple decision trees to improve accuracy and reduce overfitting. Each tree predicts, and the final output is determined by **majority vote** (classification) or **average** (regression).

7.4.1 How Random Forest Works

1. **Bootstrap Sampling (Bagging):** Train each tree on a random sample with replacement.
2. **Feature Randomness:** Split nodes using a random subset of features to decorrelate trees.
3. **Prediction Aggregation:**
   - Classification: majority vote.
   - Regression: average prediction.

7.4.2 Out-of-Bag (OOB) Error

- About 1/3 of data is not used in a tree's training → **OOB samples**.
- OOB samples test the tree; averaging errors gives the **OOB error estimate**.

Acts as an internal cross-validation without a separate validation set.

Example

Predict diabetes using features like *age, glucose, BMI, blood pressure*:

- 100 trees trained on random patient and feature subsets.
- Each tree predicts Yes/No.
- Final prediction: majority vote across all trees.

7.4.3 Strengths of Random Forest

- Handles both classification and regression tasks.
- Reduces overfitting compared to individual decision trees.
- Works well on large datasets and high-dimensional data.
- Provides feature importance scores.
- Robust against noise and missing values.

7.4.4 Weaknesses of Random Forest

- Computationally intensive (especially with many trees).
- Harder to interpret compared to a single decision tree.
- May perform poorly on very sparse data.
- Memory usage is high due to multiple trees.

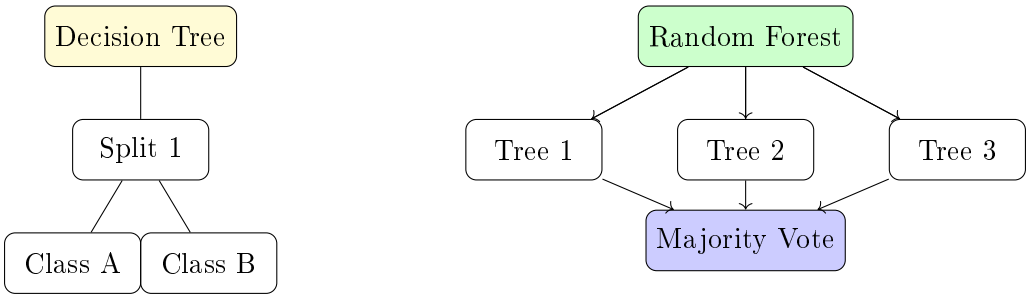
7.4.5 Applications of Random Forest

Random Forests are widely used due to their robustness and strong predictive performance. Key applications include:

- **Finance:** credit scoring, fraud detection.
- **Healthcare:** disease prediction, medical diagnosis.
- **Marketing:** customer segmentation, churn prediction.
- **Remote Sensing:** land-cover classification, change detection.
- **Bioinformatics:** gene expression and biomarker identification.
- **Cybersecurity:** intrusion and malware detection.
- **Manufacturing:** predictive maintenance, fault detection.
- **Environment:** pollution prediction, disaster-risk analysis.
- **Education:** student performance and dropout prediction.
- **NLP:** sentiment analysis, text classification.

7.4.6 Comparison of Decision Tree and Random Forest

Decision Tree	Random Forest
Simple model consisting of a single tree.	Ensemble model combining many decision trees.
Fast to train and easy to understand.	More computationally expensive.
High risk of overfitting.	Significantly reduces overfitting.
High variance: small data changes can produce different trees.	Low variance: averaging multiple trees stabilizes predictions.
Interpretable and easy to visualize.	Difficult to interpret due to many trees.
Works well on small datasets.	Performs better on large and complex datasets.
Uses all features for splitting.	Each tree uses a random subset of features (feature bagging).
Prediction comes from a single model.	Prediction is based on majority vote or averaging across all trees.



7.5 Support Vector Machines (SVMs)

What are Support Vector Machines (SVMs)?

Support Vector Machines (SVMs) are supervised learning models primarily used for classification tasks. They work by finding an optimal **hyperplane** that separates different classes with the **maximum possible margin**. The data points closest to the hyperplane, called **support vectors**, play a crucial role in defining the decision boundary. SVMs can also handle nonlinearly separable data using the **kernel trick**, which maps data into a higher-dimensional space where separation becomes easier.

7.5.1 Classification Using Hyperplanes

A **hyperplane** is a line in 2D, a plane in 3D, or a hyperplane in higher dimensions.

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

Definition of w:

$$\mathbf{w} \in \mathbb{R}^d \text{ is the normal vector to the hyperplane } \mathbf{w} \cdot \mathbf{x} + b = 0.$$

It defines the orientation of the hyperplane. The SVM optimization problem aims to find

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{s.t.} \quad y_i(\mathbf{w} \cdot x_i + b) \geq 1 \ \forall i.$$

From the dual form:

$$\boxed{\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i x_i}$$

where only support vectors have $\alpha_i > 0$.

Classification rule:

$$y = \begin{cases} +1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b \geq 0 \\ -1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b < 0 \end{cases}$$

$$\hat{y}(x) = \text{sign}(\mathbf{w} \cdot x + b).$$

Margin: When support vectors satisfy $y_i(\mathbf{w} \cdot x_i + b) = 1$, the margin width between two classes is:

$$\boxed{\text{Margin} = \frac{2}{\|\mathbf{w}\|}}.$$

7.5.2 Worked Numeric Example

Training Points:

$$x^{(1)} = (1,1),\ y^{(1)} = +1; \quad x^{(2)} = (-1,-1),\ y^{(2)} = -1.$$

Assume $\mathbf{w} = (a, a)$ (symmetry). Support vectors satisfy:

$$\begin{cases} \mathbf{w} \cdot (1,1) + b = 1, \\ \mathbf{w} \cdot (-1,-1) + b = -1. \end{cases}$$

Subtracting gives $2(\mathbf{w} \cdot (1,1)) = 2 \Rightarrow \mathbf{w} \cdot (1,1) = 1$.

Thus $2a = 1 \Rightarrow a = 0.5$, giving:

$$\boxed{\mathbf{w} = (0.5, 0.5)}, \quad b = 0.$$

So the hyperplane equation is:

$$(0.5, 0.5) \cdot x = 0 \ \Rightarrow \ x_1 + x_2 = 0.$$

Norm and Margin:

$$\|\mathbf{w}\| = \sqrt{0.5^2 + 0.5^2} = \sqrt{0.5} \approx 0.7071, \quad \text{Margin} = \frac{2}{0.7071} \approx 2.828.$$

Prediction Example: For $x = (2, 0)$,

$$\mathbf{w} \cdot x + b = 0.5(2) + 0.5(0) = 1 \Rightarrow \hat{y} = +1.$$

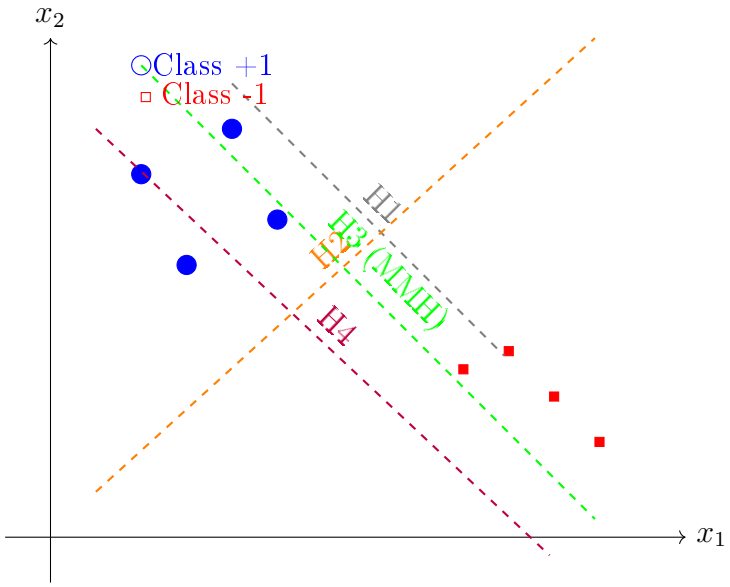
7.5.3 Support Vectors, Hyperplane, and Margin

- **Support Vectors:** Points closest to the hyperplane.
- **Margin:** Distance between hyperplane and support vectors:

$$\text{Margin} = \frac{2}{\|\mathbf{w}\|}$$

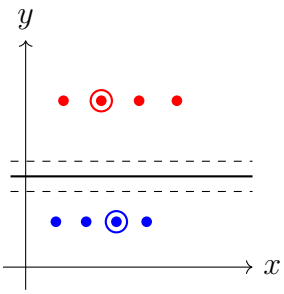
- **Maximum Margin Hyperplane (MMH):** Hyperplane that maximizes the margin.

7.5.4 Identifying the Correct Hyperplane

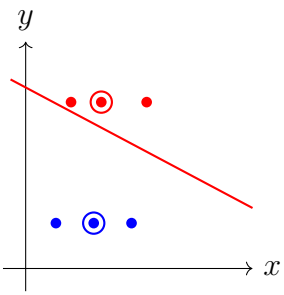


Explanation of Scenarios:

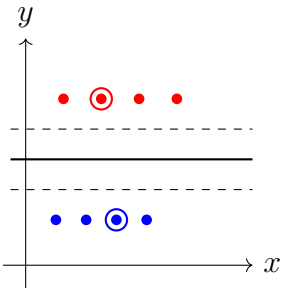
- **Scenario 1:** Hyperplane too close to one class ⇒ small margin ⇒ Not optimal



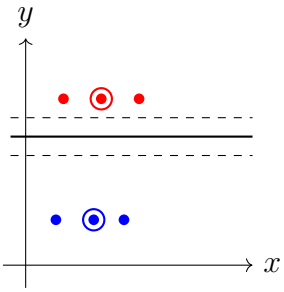
- **Scenario 2:** Hyperplane intersects data ⇒ misclassification ⇒ Invalid



- **Scenario 3:** Hyperplane maximizes margin ⇒ Ideal ⇒ **Maximum Margin Hyperplane (MMH)**

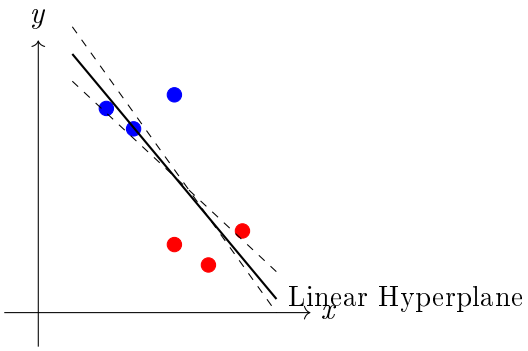


- **Scenario 4:** Hyperplane far but margin smaller than max ⇒ Suboptimal

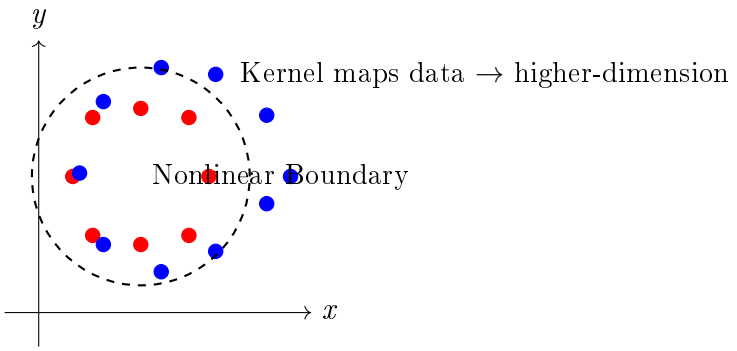


7.5.5 Linearly vs Nonlinearly Separable Data

- **Linearly Separable Data:** A single straight hyperplane can separate the two classes. SVM finds the Maximum Margin Hyperplane (MMH).



- **Nonlinearly Separable Data:** No straight line can separate the classes. A kernel maps data into higher-dimensional space to obtain linear separability there.



7.5.6 Kernel Trick

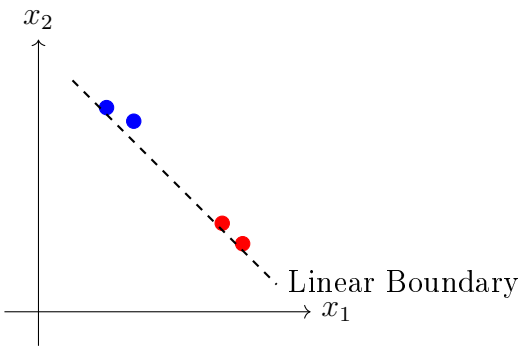
Maps data to higher dimensions to make it linearly separable. Common kernels:

- **Linear kernel:** $K(x_i, x_j) = x_i \cdot x_j$
- **Polynomial kernel:** $K(x_i, x_j) = (x_i \cdot x_j + c)^d$
- **Sigmoid kernel:** $K(x_i, x_j) = \tanh(\alpha x_i \cdot x_j + c)$
- **Gaussian / RBF kernel:** $K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$

Kernel Trick: Diagram

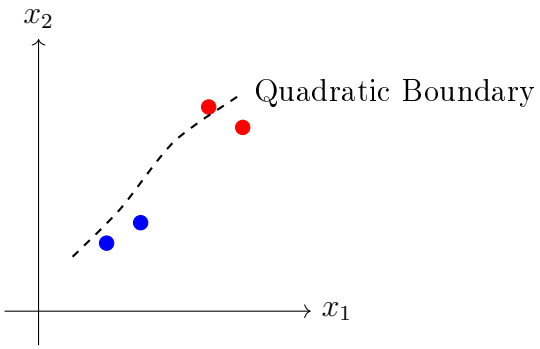
Maps data to higher dimensions to make it linearly separable. Common kernels:

- **Linear kernel:** $K(x_i, x_j) = x_i \cdot x_j$



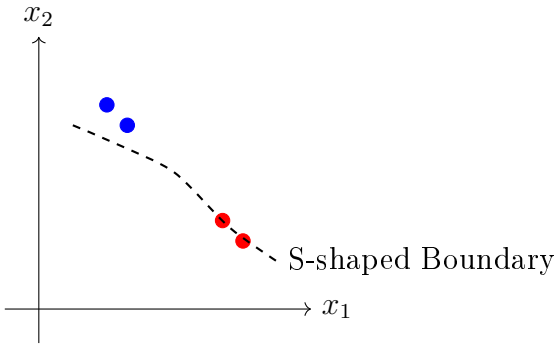
Linear kernel → no transformation.

- **Polynomial kernel:** $K(x_i, x_j) = (x_i \cdot x_j + c)^d$



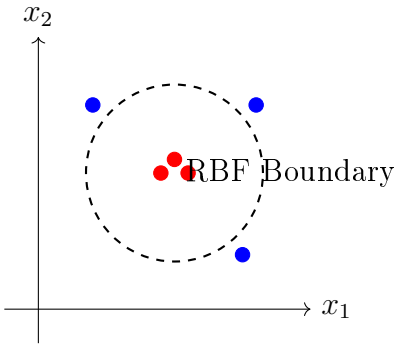
Polynomial kernel → creates curved decision boundaries.

- **Sigmoid kernel:** $K(x_i, x_j) = \tanh(\alpha x_i \cdot x_j + c)$



Sigmoid kernel → neural-network-like S-curve boundary.

- **Gaussian / RBF kernel:** $K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$



RBF kernel → circular / radial decision boundaries.

7.5.7 Strengths of SVM

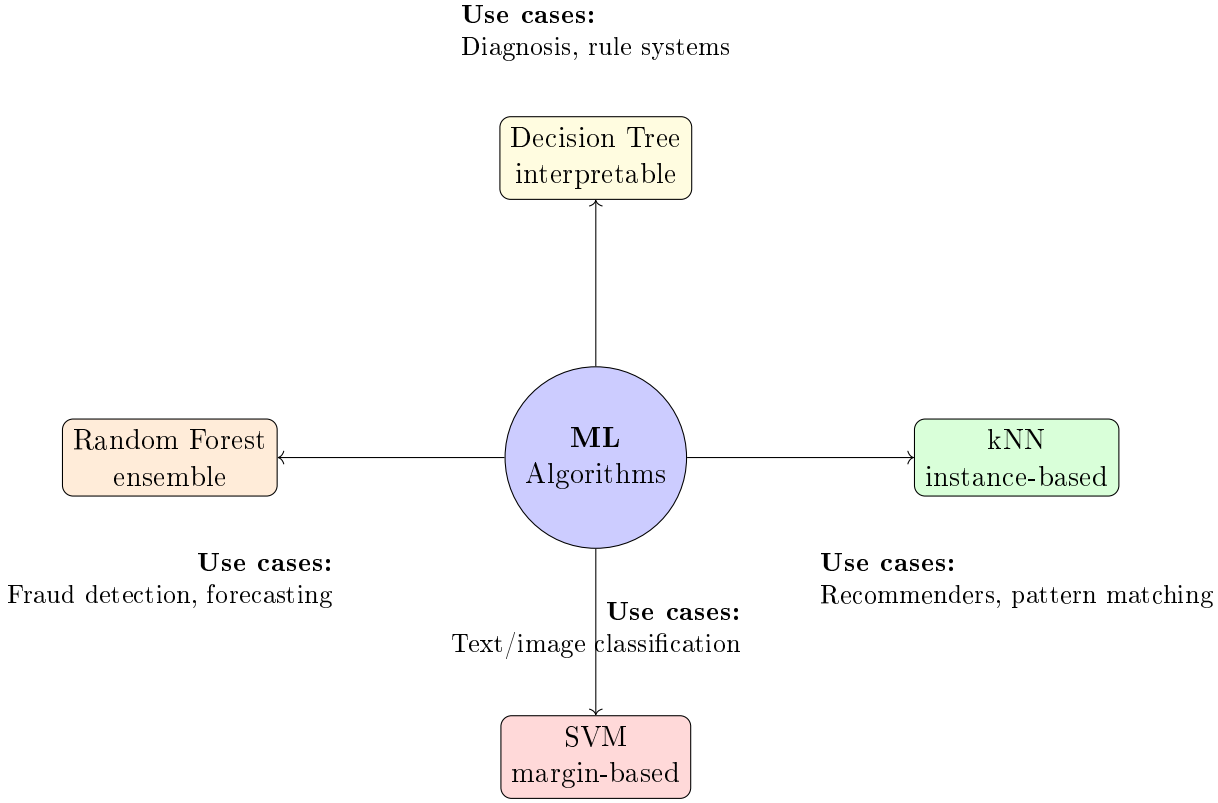
- Works well in high-dimensional spaces
- Effective with small/medium datasets
- Robust to overfitting with large margin
- Can handle nonlinear boundaries via kernels

7.5.8 Weaknesses of SVM

- Computationally expensive for large datasets
- Choosing the right kernel is challenging
- Sensitive to noise and overlapping classes
- Less interpretable than linear models

7.5.9 Applications of SVM

- **Text**: spam filtering, sentiment analysis.
- **Images**: face/digit recognition.
- **Bioinformatics**: gene/protein classification.
- **Medical**: cancer and disease detection.
- **Finance**: fraud and risk prediction.
- **Speech/Audio**: speaker and emotion recognition.
- **Industry**: fault detection, quality control.
- **Remote sensing**: land-cover classification.
- **Cybersecurity**: intrusion and malware detection.



7.6 Comparison of kNN, Decision Tree, Random Forest, and SVM

Criteria	kNN	Decision Tree	Random Forest	SVM
Model Type	Lazy learner, instance-based	Tree-based rule model	Ensemble of trees (bagging)	Margin-based classifier
Training Time	Very low (no model built)	Fast	Moderate to high	High (large datasets)
Prediction Time	Slow (search over all samples)	Very fast	Moderate (many trees)	Fast after training
Overfitting Risk	Low–Medium (depends on k)	High	Low (bagging reduces variance)	Low
Interpretability	Poor	Excellent	Poor	Poor–Moderate
Works Well For	Small datasets, distance-based classification	Simple, rule-based tasks	Large, complex, high-dimensional tasks	High-dimensional, separable/non-separable tasks
Sensitivity to Noise	High	High	Low	Medium
Handles Non-linear Data	Yes (locally)	Yes	Yes	Yes (with kernels)
Applications	Recommendation, pattern recognition	Medical diagnosis, segmentation	Fraud detection, forecasting	Text classification, image recognition

Chapter 8

Supervised Learning: Regression

8.1 Introduction

Supervised learning involves learning a mapping from input features (X) to an output variable (Y). It includes two major categories: classification and regression.

- Classification - Output is categorical (e.g., spam/not spam, disease/no disease).
- Regression- Output is continuous (e.g., sales amount, house price, rainfall, temperature).

Regression is a technique which builds a mathematical relationship:

$$Y = f(X_1, X_2, \ldots, X_n) + \text{error}$$

where Y is the target and $X_1, X_2, \ldots, X_n$ are predictors.
Example:
A real-world example involving Karen, a real estate expert. As her business grows, she needs a reliable method to estimate house prices. The factors (independent variables X) influencing price:

- Area of flat
- Location score
- Floor number
- Age of property
- Amenities

Target (dependent variable Y):

- Selling price

Frank, a data scientist, suggests using regression. Using historical sales data, the model learns the relationship between property features and prices. Thus, regression becomes:

$$Price = f(Area, Location, Floor, Age, Amenities)$$

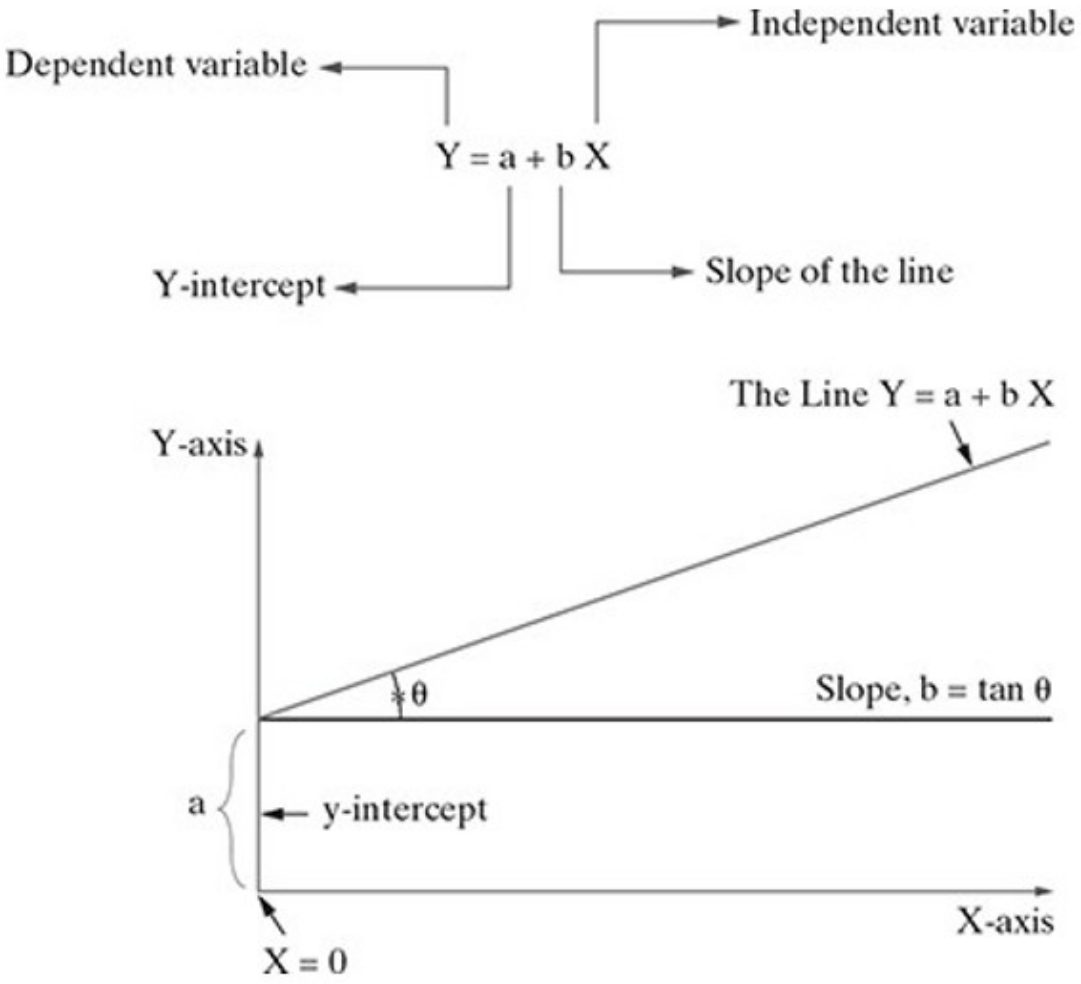
In the context of regression, dependent variable (Y) is the one whose value is to be predicted, e.g. the price quote of the real estate in the context of Karen's problem. This variable is presumed to be functionally related to one (say, X) or more independent variables called predictors. In the context of Karen's problem,Frank used area of the property, location, floor, etc. as predictors of the model that he built. In other words, the dependent variable depends on independent variable(s) or predictor(s). Regression is essentially finding a relationship (or) association between the dependent variable (Y) and the independent variable(s) (X), i.e. to find the function $'f'$ for the association $Y = f(X)$.

COMMON REGRESSION ALGORITHMS:

1. Simple linear regression
2. Multiple linear regression
3. Polynomial regression
4. Logistic regression
5. Maximum likelihood estimation (least squares)

8.2 Simple Linear Regression

Simple linear regression is the simplest regression model which involves only one predictor. This model assumes a linear relationship between the dependent variable and the predictor variable as shown in the following Figure:



The Mathematical form:

$$Y = a + bX + \epsilon$$

Where:

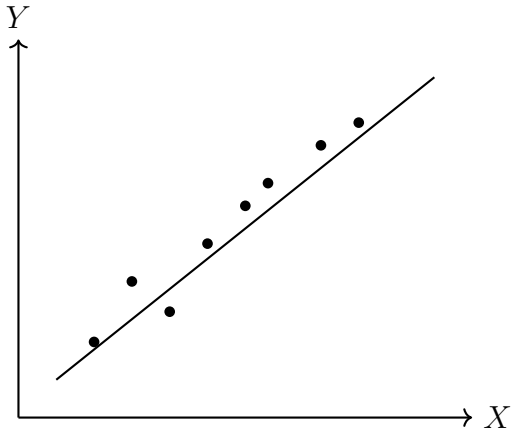
- a = intercept
- b = slope
- ϵ = error term

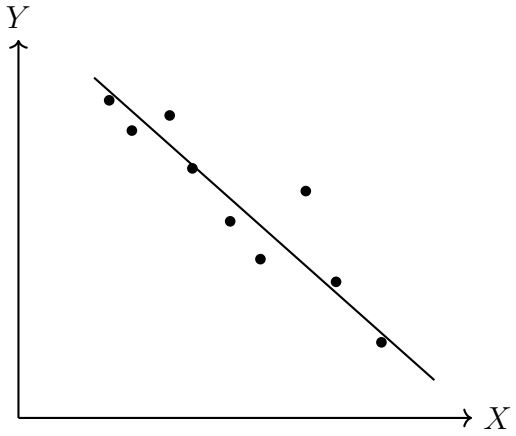
Interpretation:

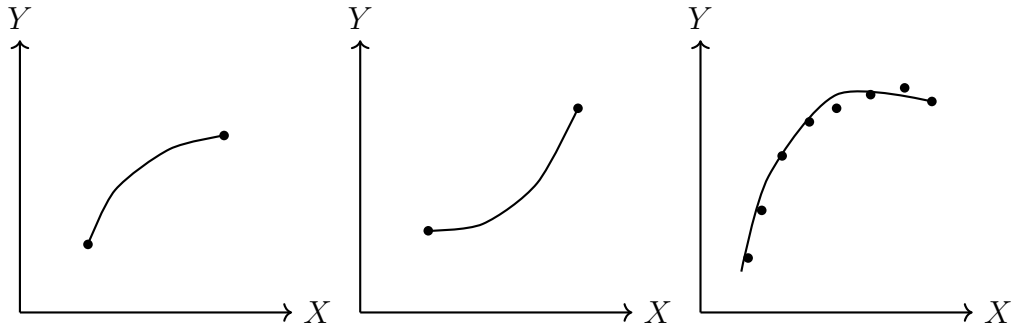
- Slope (b) shows change in Y for a $1-$unit change in X.

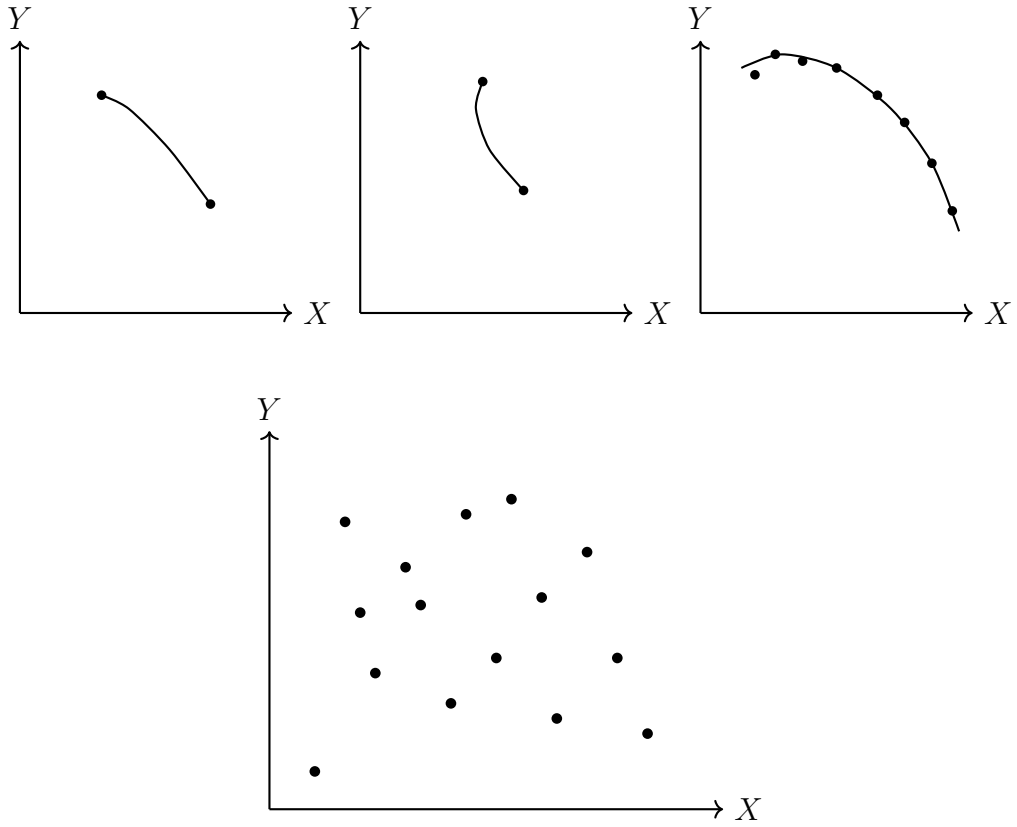
- Intercept (a) shows expected Y when $X = 0$.

Types of relationships:








Y
X
Y
X
Y
X
Y
X

8.2.1 Error in simple regression

The regression equation model in machine learning uses the above slope–intercept format in algorithms. X and Y values are provided to the machine, and it identifies the values of a (intercept) and b (slope) by relating the values of X and Y. However, identifying the exact match of values for a and b is not always possible. There will be some error value (ϵ) associated with it. This error is called marginal or residual error.
Residual Error: It is the distance between the predicted point (on the regression line) and the actual point.

$$Y = (a + bX) + \epsilon$$

The aim here is to calculate the values of a and b. For this a straight line is drawn as close as possible over the points on the scatter plot. Ordinary Least Squares (OLS) is the technique used to estimate a line that will minimize the error (ϵ), which is the difference between the predicted and the actual values of Y. This means summing the errors of each prediction or, more appropriately, the Sum of the Squares of the Errors (SSE) $\left(\sum_i \epsilon_i^2\right)$.

The SSE is least when b takes the value

$$b = \frac{\sum_i (X_i - \overline{X})(Y_i - \overline{Y})}{\sum_i (X_i - \overline{X})^2} = \frac{cov(X, Y)}{Var(X)}$$

and

$$a = \overline{Y} - b\overline{X}$$

. **Example:** Given data: 15 students with internal marks (X) and external marks (Y). Method:

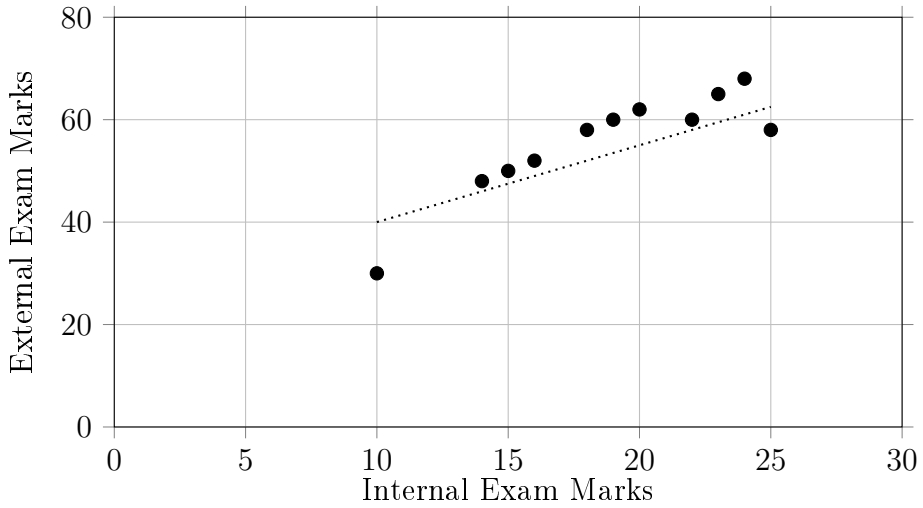
Internal Exam	15	23	18	23	24	22	22	19	19	16	24	11	24	16	23
External Exam	49	63	58	60	58	61	60	63	60	52	62	30	59	49	68

- Step 1: Calculate the mean of X and Y ($\overline{X}$ and $\overline{Y}$)
- Step 2: Compute deviations $(X - \overline{X})$ and $(Y_i - \overline{Y})$.
- Step 3: Get the product $SP = \sum (X - \overline{X})(Y - \overline{Y})$
- Step 4: Compute $SS_X = \sum (X - \overline{X})^2$
- Step 5: Compute slope: $b = \frac{SP}{SS_X}$
- Step 6: Compute intercept: $a = \overline{Y} - b\overline{X}$

For this example, $a = 19.0473$ and $b = 1.89395$.
Interpretation:

- Every 1−point increase in internal marks increases predicted external marks by about 1.89 points.
- Baseline external mark, even with zero internal, is 19.04.

A scatter plot was drawn to explore the relationship between the independent variable (internal marks) mapped to $X-$axis and dependent variable (external marks) mapped to $Y-$axis as depicted in the Figure below:



8.3 Multiple Linear Regression

Multiple Linear Regression (MLR) includes more than one predictor variable. For $'n'$ predictor variables:
Model:

$$Y = a + b_1X_1 + b_2X_2 + \cdots + b_nX_n$$

. Graphically, a multiple linear regression becomes a plane in $3D$ space.

8.3.1 Assumptions of Regression

Regression relies on several mathematical assumptions:

1. Linearity: Relation between X and Y is linear.
2. $n > k$: Number of observations must exceed number of parameters.
3. Regression explains association, not causation.
4. Valid for interpolation only.
5. Stable environment: Model validity depends on consistent conditions.
6. Homoskedasticity: Constant variance of errors.
7. Errors follow a normal distribution with mean$= 0$.
8. Errors are independent (no autocorrelation).

These assumptions ensure OLS provides BLUE (Best Linear Unbiased Estimators).

8.4 Problems in Regression

In multiple regressions, there are two primary problems: multicollinearity and heteroskedasticity.

8.4.1 Multicollinearity

- **Multicollinearity** occurs when two or more independent variables (predictors) are highly correlated with each other, i.e., one predictor can be approximately expressed as a linear combination of other predictors.
- Perfect multicollinearity: exact linear relationship
- Imperfect multicollinearity: strong (but not exact) correlation

Mathematically, multicollinearity violates the assumption that there should be **no exact linear relationship among the independent variables**.

Why Multicollinearity is a Problem?
Although a regression model with multicollinearity may still predict Y well, it causes serious issues in interpretation and inference:

- Unstable coefficient estimates: Small changes in data can lead to large changes in regression coefficients.
- Inflated standard errors: Variance of estimated coefficients increases.
- Statistical insignificance of important variables: A variable that truly affects Y may appear insignificant due to large standard errors.
- Difficulty in isolating individual effects: It becomes hard to determine how much Y changes when one predictor changes while others are held constant.

Variance Inflation Factor (VIF)
The most common measure to detect multicollinearity:

$$VIF_j = \frac{1}{1 - R_j^2}$$

where:

- $R_j^2 = R-$squared value obtained by regressing predictor X_j on all other predictors.

Interpretation:

- $VIF = 1-$ No multicollinearity
- $1 < VIF < 5-$ Moderate multicollinearity
- $VIF \geq 10-$ Severe multicollinearity (problematic)

Consequences in OLS Estimation

- OLS estimators remain unbiased
- But they are no longer efficient (large variance)
- Confidence intervals become wider
- Hypothesis tests become unreliable

Remedies for Multicollinearity

- Remove one of the correlated variables
- Combine variables (e.g., average, index, ratio)
- Increase sample size (if possible)
- Regularization methods
  - Ridge regression ($L2$)
  - Lasso regression ($L1$)
- Dimensionality reduction
  - Principal Component Analysis (PCA)

8.4.2 Heteroskedasticity

- **Heteroskedasticity** refers to the situation where the variance of the error term is not constant across observations.
  Formally, the homoskedasticity assumption is: $Var(u_i|X_i) = \sigma^2$.
  If instead: $Var(u_i|X_i) neq \sigma^2$, then heteroskedasticity is present.

Why Heteroskedasticity is a Problem? Heteroskedasticity does not bias coefficient estimates, but it affects:

- Standard errors (they become incorrect)
- t-tests and F-tests (invalid)
- Confidence intervals (misleading)
- Efficiency of OLS estimators

Hence, inference based on OLS becomes unreliable **Remedies to Improve Regression Accuracy**

- **Ridge Regression (L2):** $min(SSE + \lambda \sum b_j^2)$
- Shrinks coefficients
- Shrinks coefficients

- Reduces variance

- **Lasso Regression (L1):** $min(SSE + \lambda \sum |b_j|)$

- Shrinks coefficients

- Sets some coefficients to zero

- Performs variable selection

Subset Selection

- Best Subset Selection (2^k models)

- Forward Stepwise Selection

- Backward Stepwise Selection

Dimensionality Reduction

- Principal Component Analysis (PCA)

- Converts correlated predictors into uncorrelated components

- Reduces multicollinearity

8.5 Polynomial Regression

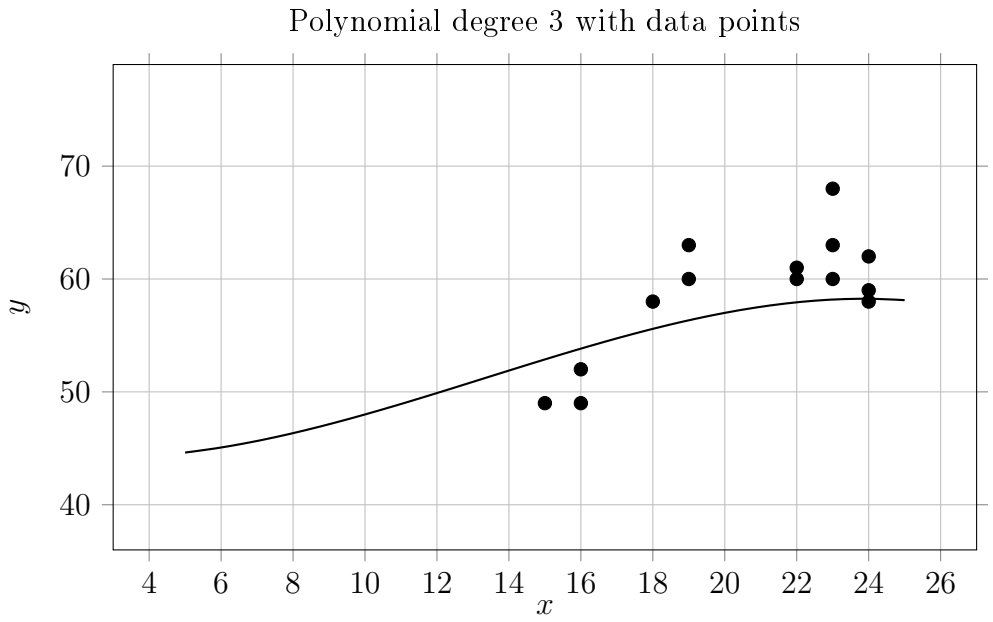
Polynomial regression model is the extension of the simple linear model by adding extra predictors obtained by raising (squaring) each of the original predictors to a power. For example, if there are three variables, X, X^2, and X^3 are used as predictors. This approach provides a simple way to yield a non-linear fit to data.

$$f(X) = c_0 + c_1X^1 + c_2X^2 + c_3X^3$$

In the above equation, c_0, c_1, c_2, and c_3 are the coefficients. **Example:** Let us use the below data set of (X, Y) for degree 3 polynomial. The regression line below is slightly curved for

Internal Exam	15	23	18	23	24	22	22	19	19	16	24	11	24	16	23
External Exam	49	63	58	60	58	61	60	63	60	52	62	30	59	49	68

polynomial degree 3 with the above 15 data points.



8.6 Logistic Regression

Logistic Regression is a supervised machine learning algorithm used for classification problems. Unlike linear regression which predicts continuous values it predicts the probability that an input belongs to a specific class. It is used for binary classification where the output can be one of two possible categories such as Yes/No, True/False or 0/1. It uses sigmoid function to convert inputs into a probability value between 0 and 1.

8.6.1 Assumptions of Logistic Regression

Understanding the assumptions behind logistic regression is important to ensure the model is applied correctly, main assumptions are:

1. **Independent observations:** Each data point is assumed to be independent of the others means there should be no correlation or dependence between the input samples.
2. **Binary dependent variables:** It takes the assumption that the dependent variable must be binary, means it can take only two values.
3. **Linearity relationship between independent variables and log odds:** The model assumes a linear relationship between the independent variables and the log odds of the dependent variable which means the predictors affect the log odds in a linear way.
4. **No outliers:** The dataset should not contain extreme outliers as they can distort the estimation of the logistic regression coefficients.
5. **Large sample size:** It requires a sufficiently large sample size to produce reliable and stable results.

8.6.2 Understanding Sigmoid Function

1. The sigmoid function is an important part of logistic regression, which is used to convert the raw output of the model into a probability value between 0 and 1.

2. This function takes any real number and maps it into the range 0 to 1, forming an "S" shaped curve called the sigmoid curve or logistic curve. Because probabilities must lie between 0 and 1, the sigmoid function is perfect for this purpose.

3. In logistic regression, we use a threshold value, usually 0.5, to decide the class label.
   - If the sigmoid output is the same or above the threshold, the input is classified as Class 1.
   - If it is below the threshold, the input is classified as Class 0.

   This approach helps to transform continuous input values into meaningful class predictions.

8.6.3 How does Logistic Regression work?

Logistic regression model transforms the linear regression function's continuous value output into categorical value output using a sigmoid function, which maps any real-valued set of independent variables input into a value between 0 and 1. This function is known as the logistic function.
Suppose we have input features represented as a matrix X and the dependent variable is Y having only binary value i.e 0 or 1.

$$Y = \begin{cases} 0, \text{if} \quad \text{Class 1} \\ 1, \text{if} \quad \text{Class 2} \end{cases}$$

then, apply the multi-linear function to the input variables X.

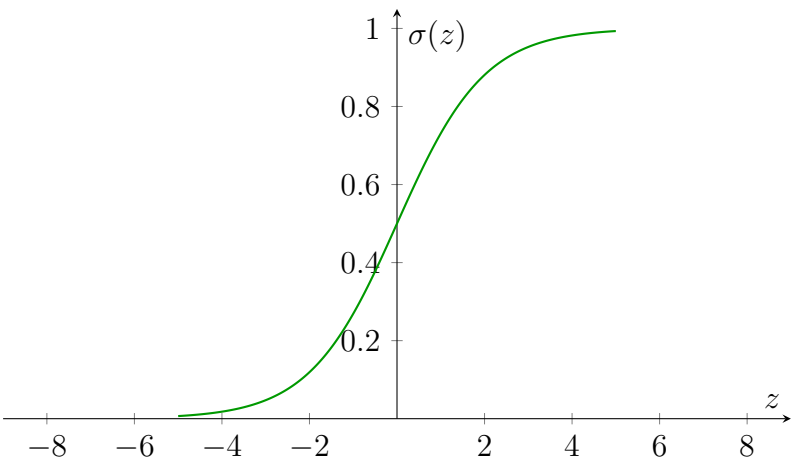
$$z = \left(\sum_{i=1}^{n} \omega_i x_i\right) + b$$

Here x_i is the i^{th} observation of X, $\omega_i = [\omega_1, \omega_2, \omega_3 \cdots \omega_n]$ is the weights or the Coefficient and $b_i s$ the bias term, also known as the intercept. Simply, this can be represented as the dot product of weight and bias.

$$z = \omega . X + b$$

At this stage, z_i s a continuous value from the linear regression. Logistic regression then applies the sigmoid function to z to convert it into a probability between 0 and 1 which can be used to predict the class.
Now we use the sigmoid function where the input will be z and we find the probability between 0 and 1. i.e. predicted y.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



Sigmoid function

- $\sigma(z)$ tends towards 1 as $z \to \infty$.
- $\sigma(z)$ tends towards 0 as $z \to -\infty$.
- $\sigma(z)$ is always bounded between 0 and 1.

where the probability of being a class can be measured as:

$$P(y = 1) = \sigma(z)$$
$$P(y = 0) = 1 - \sigma(z)$$

8.7 Maximum Likelihood Estimation

The coefficients in a logistic regression are estimated using a process called Maximum Likelihood Estimation (MLE). The goal is to find weights ω and bias b that maximises the likelihood of observing the data.
For each data point i

- for $y = 1$, predicted probabilities will be: $p(X; b, \omega) = p(x)$
- for $y = 0$, predicted probabilities will be: $p(X; b, \omega) = 1 - p(x)$

$$L(b, \omega) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}$$

Taking natural logs on both sides we have:

$$Log(L(b, w)) = Log(L(b, \omega)) = \sum_{i=1}^{n} -log1 + e^{\omega x_i + b} + \sum_{i=1}^{n} y_i(\omega.x_i + b)$$

This is known as the log-likelihood function. MLE is about predicting the value for the parameters that maximizes the likelihood function.