Data Science

Module 3: Data Analysis

Data Analysis: Introduction, Terminology, and Concepts

- **Introduction to Data Analysis:** This would typically cover:
  - **What is data analysis?** The process of inspecting, cleansing, transforming, and modeling data with the goal of discovering useful information, informing conclusions, and supporting decision-making.
  - **Why is it important?** Highlighting its role in various fields (business, science, healthcare, etc.) for gaining insights, making data-driven decisions, identifying trends, and solving problems.
  - **Types of data analysis:** Descriptive, diagnostic, predictive, and prescriptive analysis.
  - **The data analysis process:** Often depicted as a cycle involving data collection, cleaning, exploration, modeling, and interpretation.

Terminology and Concepts:

This part introduces fundamental vocabulary and ideas, such as:

- **Data:** Raw facts, figures, or symbols collected for analysis.
- **Information:** Data processed, organized, and structured in a given context to make it useful.
- **Knowledge:** The understanding gained from information, leading to insights and actionable intelligence.
- **Dataset:** A collection of related data.
- **Variable:** A characteristic or attribute that can be measured or observed.

Terminology and Concepts:

- **Observation/Record:** A set of values for all variables for a single entity.
- **Population:** The entire group of individuals or instances about which we want to draw conclusions.
- **Sample:** A subset of the population selected for analysis.
- **Qualitative vs. Quantitative Data:** Categorical vs. numerical data.
- **Discrete vs. Continuous Data:** Countable vs. measurable data.

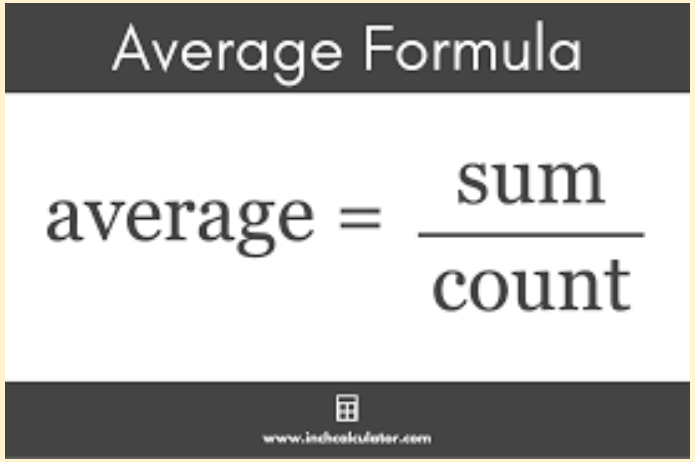
Introduction to Statistics

- **Introduction to Statistics**
- This section forms the backbone of data analysis, providing the tools for understanding and interpreting data.
- **What is Statistics?** The science of collecting, organizing, analyzing, interpreting, and presenting data.
- **Branches of Statistics:**
  - **Descriptive Statistics:** Summarizing and describing the main features of a dataset (e.g., mean, median, mode, standard deviation).
  - **Inferential Statistics:** Making inferences and drawing conclusions about a population based on a sample (e.g., hypothesis testing, confidence intervals).
- **Importance of Statistics in Data Analysis:** How statistical methods are used to draw valid conclusions, quantify uncertainty, and make predictions.

Central Tendencies and Distributions

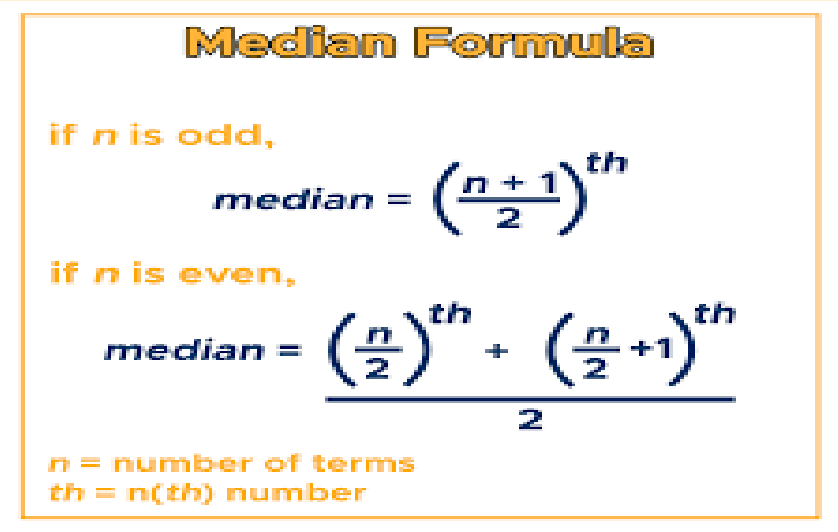
- These are fundamental concepts in descriptive statistics used to understand the typical value and spread of data.
- **Central Tendencies (Measures of Central Location):**

  - **Mean (Average):** The sum of all values divided by the number of values.



Central Tendencies and Distributions

- **Median:** The middle value in a sorted dataset. It's robust to outliers.



- **Mode:** The most frequently occurring value in a dataset.

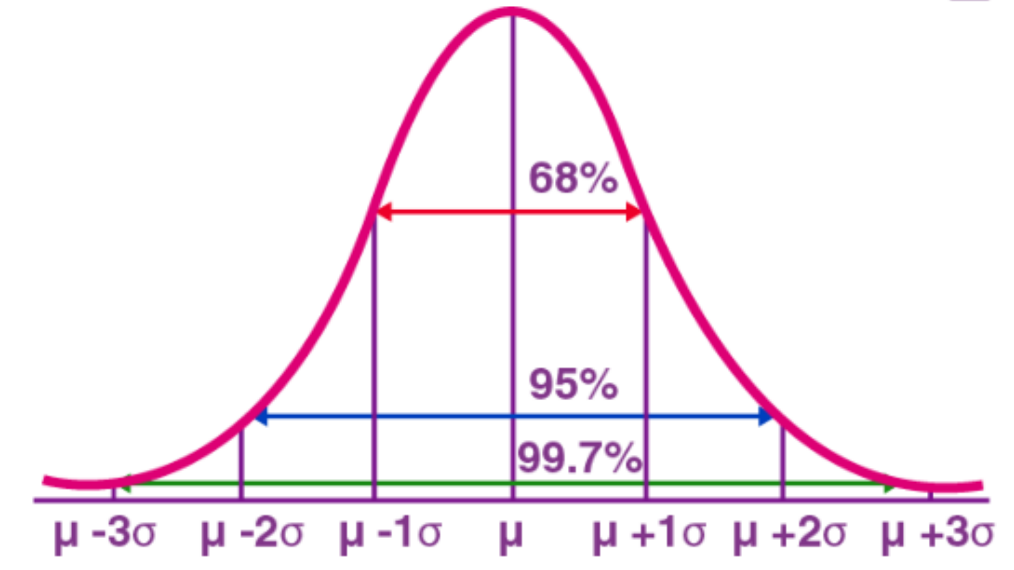
Central Tendencies and Distributions

- **Distributions:** Describe the pattern of values in a dataset.

1. **Frequency Distribution:** A table or graph that displays the frequency of various outcomes in a sample.
2. **Probability Distribution:** A mathematical function that describes the likelihood of obtaining the possible values that a random variable can take.

- **Common Distributions:**
  - **Normal Distribution (Bell Curve):** A symmetric, bell-shaped distribution that is very common in nature and statistics. Many statistical tests assume normality.

Normal Distribution

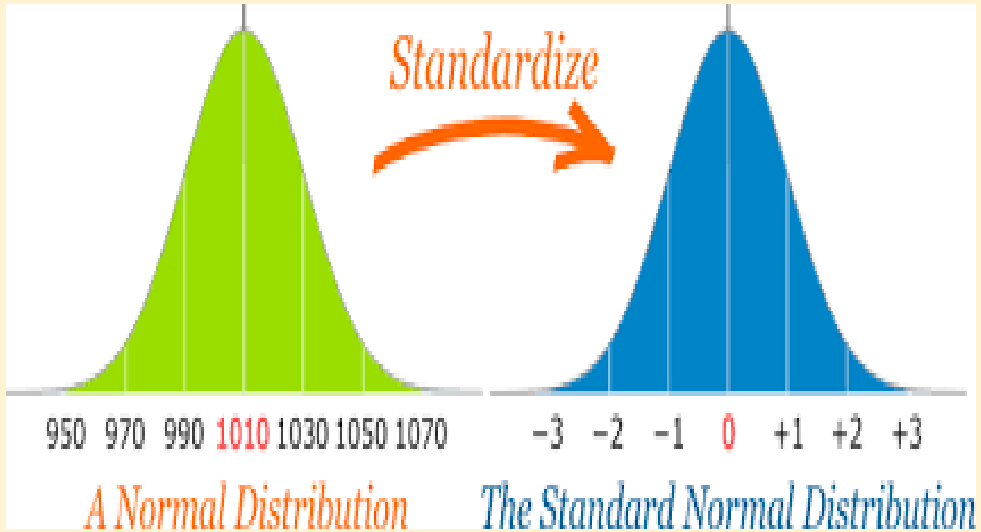


The probability density function of normal or gaussian distribution is given by;

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

Mean, Median, and Mode are Equal: In a Normal distribution, the mean (average), median (middle value), and mode (most frequent value) are all exactly the same and located at the very peak of the bell curve

Standard Normal Distribution



The probability density function of normal or gaussian distribution is given by;

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

- The normal distribution is the proper term for a probability bell curve.
- In a Standard normal distribution, the mean is zero and the standard deviation is 1. It has zero skew and a kurtosis of 3.
- Normal distributions are symmetrical, but not all symmetrical distributions are normal.

Central Tendencies and Distributions

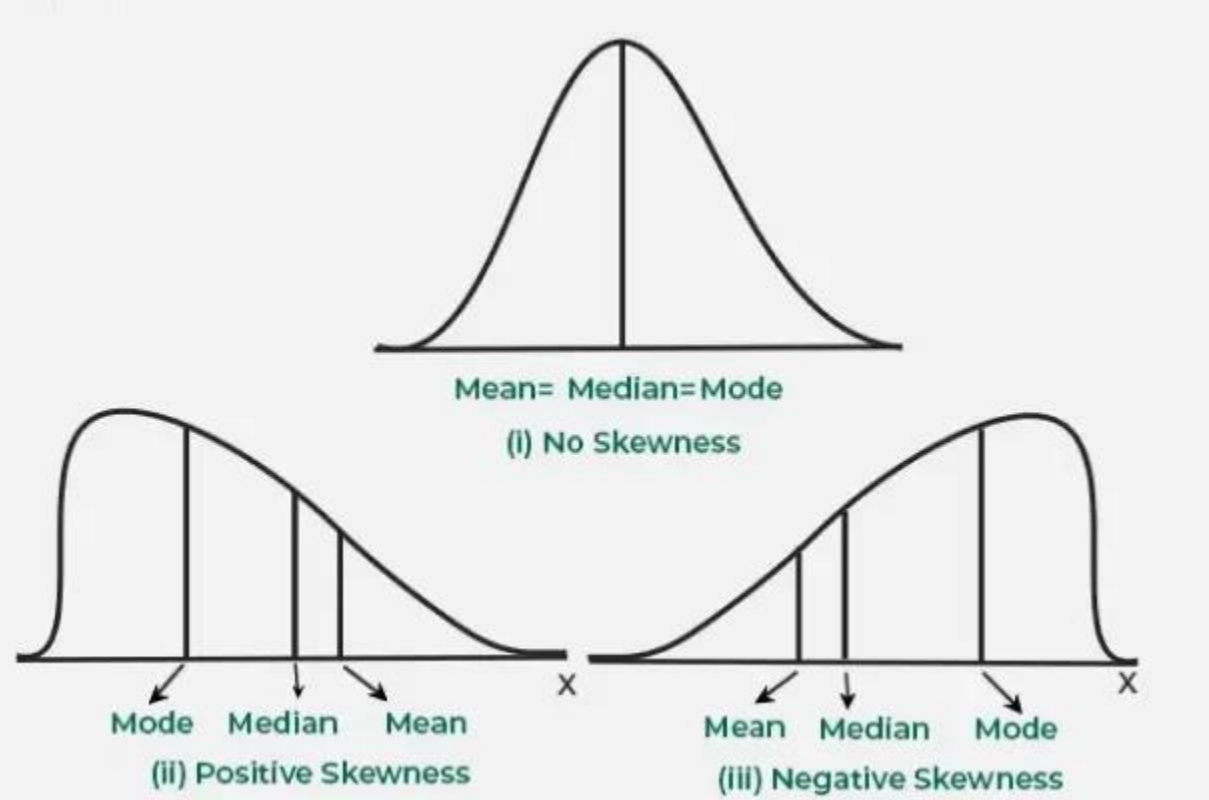
- **Skewness:** A measure of the asymmetry of the probability distribution of a real-valued random variable about its mean.

The Formula of Skewness is:

$$\text{Skewness} = \frac{\sum (x - \bar{x})^3}{(n - 1) \cdot S^3}$$

Where:
S: standard deviation
$\bar{X}$: Mean

Skewness


Mean= Median=Mode
(i) No Skewness
X
Mode
Median
Mean
(ii) Positive Skewness
X
Mean
Median
Mode
(iii) Negative Skewness

Central Tendencies and Distributions

- **Kurtosis:** A measure of the "tailedness" of the probability distribution of a real-valued random variable.
- Kurtosis is a statistical measure that defines how heavily the tails of a distribution differ from the tails of a normal distribution. In other words, kurtosis identifies whether the tails of a given distribution contain extreme values.

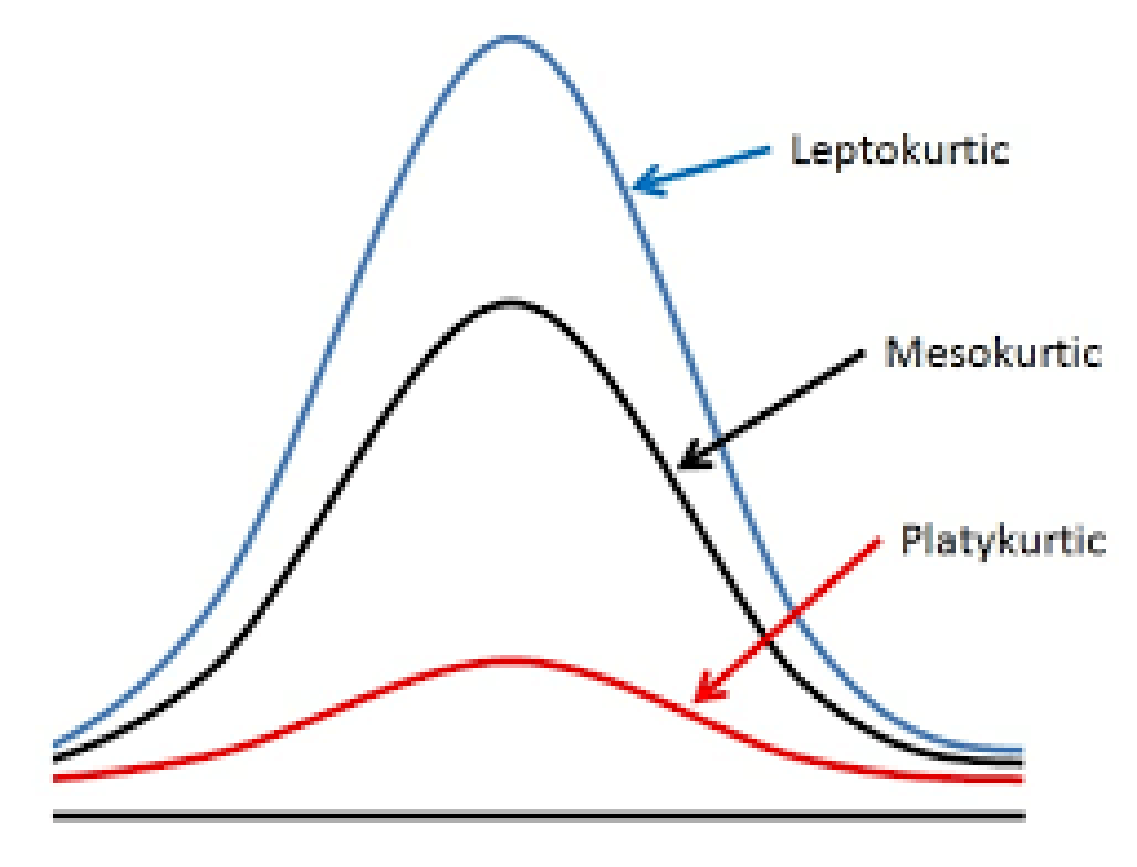
The Formula of kurtosis is:

$$\text{Kurtosis} = \frac{\sum \left(x - \bar{x}\right)^4}{(n - 1) \cdot S^4}$$

Where:

S: standard deviation $\quad \bar{X}$: Mean

Kurtosis



Variance

Variance is a key measure of data dispersion or spread.

- **Definition:** The average of the squared differences from the mean. It quantifies how far each number in the dataset is from the mean and therefore from every other number in the dataset.
- **Formula:** For a population, $\sigma^2 = \frac{\sum(x_i-\mu)^2}{N}$. For a sample, $s^2 = \frac{\sum(x_i-\bar{x})^2}{n-1}$.
- **Interpretation:** A high variance indicates that data points are widely spread out from the mean, while a low variance indicates that data points are clustered closely around the mean.
- **Relationship to Standard Deviation:** The standard deviation is the square root of the variance ($\sigma = \sqrt{\sigma^2}$ or $s = \sqrt{s^2}$), providing a measure of spread in the same units as the original data.

Distribution Properties and Arithmetic

- This likely delves deeper into the characteristics of distributions and how to perform calculations with them.
- **Properties of Distributions:**

  - **Shape:** Skewness, kurtosis, symmetry.
  - **Measures of Spread/Dispersion:** Range, interquartile range (IQR), variance, standard deviation.
  - **Percentiles, Quartiles:** Values that divide a dataset into specific proportions.

Samples/CLT (Central Limit Theorem)

- This section is crucial for inferential statistics.
- **Samples:**
  - **Sampling Techniques:** Methods for selecting a subset of a population for study (e.g., simple random sampling, stratified sampling, cluster sampling).
  - **Sampling Error:** The difference between a sample statistic and its corresponding population parameter.
  - **Bias in Sampling:** Systematic errors introduced during the sampling process.

Central Limit Theorem (CLT):

It states that, given a sufficiently large sample size from a population with a finite level of variance, the mean of all samples from the same population will be approximately equal to the mean of the population.

Furthermore, the distribution of the sample means will be approximately normal, regardless of the shape of the original population distribution.

- **Significance:** It is critical for hypothesis testing and constructing confidence intervals because it allows us to use the normal distribution to make inferences about population parameters even when the population distribution is not normal.
- **Conditions for CLT:** Independence of samples, sufficiently large sample size (often n > 30 is a rule of thumb).

Central Limit Theorem

- **Example 1. A distribution has a mean of 69 and a standard deviation of 420. Find the mean and standard deviation if a sample of 80 is drawn from the distribution.**

Central Limit Theorem

Given: $\mu = 69$, $\sigma = 420$, $n = 80$

As per the Central Limit Theorem, the sample mean is equal to the population mean.

Hence, $\mu_{\bar{x}} = \mu = 69$

Now, $\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$

$\Rightarrow \sigma_{\bar{x}} = 420/\sqrt{80}$

$\Rightarrow \sigma_{\bar{x}} = \mathbf{46.95}$

Example2:

The average weight of a water bottle is 30 kg, with a standard deviation of 1.5 kg. If a sample of 45 water bottles is selected at random from a consignment and their weights are measured, find the probability that the mean weight of the sample is less than 28 kg.

Example 2:

. Understand the Given Information:

- Population mean (μ) = 30 kg
- Population standard deviation (σ) = 1.5 kg
- Sample size (n) = 45
- We want to find the probability that the sample mean ($\bar{x}$) is less than 28 kg.

The standard error of the mean ($\sigma_{\bar{x}}$) is the standard deviation of the sampling distribution of the sample means. It's calculated as:

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$$

$$\sigma_{\bar{x}} = \frac{1.5}{\sqrt{45}}$$

Example2:

2. Z-score:

$$Z = \frac{\bar{x} - \mu}{\sigma_{\bar{x}}} = \frac{28 - 30}{0.2236} = \frac{-2}{0.2236} \approx -8.944$$

3. Probability:

$$P(\bar{x} < 28) = P(Z < -8.944) \approx 0.0000 \text{ (practically zero)}$$

Example3:

The average daily sales of a coffee shop are 500 cups, with a standard deviation of 40 cups. If a random sample of 30 days is selected, what is the probability that the mean daily sales for this sample will be greater than 515 cups?

Solution:

Standard Error of the Mean: $\sigma_{\bar{x}} = \frac{40}{\sqrt{30}} \approx 7.303$

Z-score: $Z = \frac{515-500}{7.303} = \frac{15}{7.303} \approx 2.054$

Probability: $P(\bar{x} > 515) = P(Z > 2.054) = 1 - P(Z \leq 2.054) \approx 1 - 0.9800 = 0.0200$

Z –Score Table

ztable.net

z	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
-0	.50000	.49601	.49202	.48803	.48405	.48006	.47608	.47210	.46812	.46414
-0.1	.46017	.45620	.45224	.44828	.44433	.44034	.43640	.43251	.42858	.42465
-0.2	.42074	.41683	.41294	.40905	.40517	.40129	.39743	.39358	.38974	.38591
-0.3	.38209	.37828	.37448	.37070	.36693	.36317	.35942	.35569	.35197	.34827
-0.4	.34458	.34090	.33724	.33360	.32997	.32636	.32276	.31918	.31561	.31207
-0.5	.30854	.30503	.30153	.29806	.29460	.29116	.28774	.28434	.28096	.27760
-0.6	.27425	.27093	.26763	.26435	.26109	.25785	.25463	.25143	.24825	.24510
-0.7	.24196	.23885	.23576	.23270	.22965	.22663	.22363	.22065	.21770	.21476
-0.8	.21186	.20897	.20611	.20327	.20045	.19766	.19489	.19215	.18943	.18673
-0.9	.18406	.18141	.17879	.17619	.17361	.17106	.16853	.16602	.16354	.16109
-1	.15866	.15625	.15386	.15151	.14917	.14686	.14457	.14231	.14007	.13786
-1.1	.13567	.13350	.13136	.12924	.12714	.12507	.12302	.12100	.11900	.11702
-1.2	.11507	.11314	.11123	.10935	.10749	.10565	.10383	.10204	.10027	.09853
-1.3	.09680	.09510	.09342	.09176	.09012	.08851	.08692	.08534	.08379	.08226
-1.4	.08076	.07927	.07780	.07636	.07493	.07353	.07215	.07078	.06944	.06811
-1.5	.06681	.06552	.06426	.06301	.06178	.06057	.05938	.05821	.05705	.05592

Basic Machine Learning Algorithms

This section transitions from purely statistical analysis to predictive modeling using computational algorithms. Machine learning algorithms enable systems to learn from data without explicit programming.

- **Supervised Learning:** Algorithms that learn from labeled data (input-output pairs) to make predictions.
  - **Regression:** Predicting a continuous output (e.g., house prices).
  - **Classification:** Predicting a categorical output (e.g., spam/not spam).
- **Unsupervised Learning:** Algorithms that find patterns in unlabeled data (e.g., clustering).

.

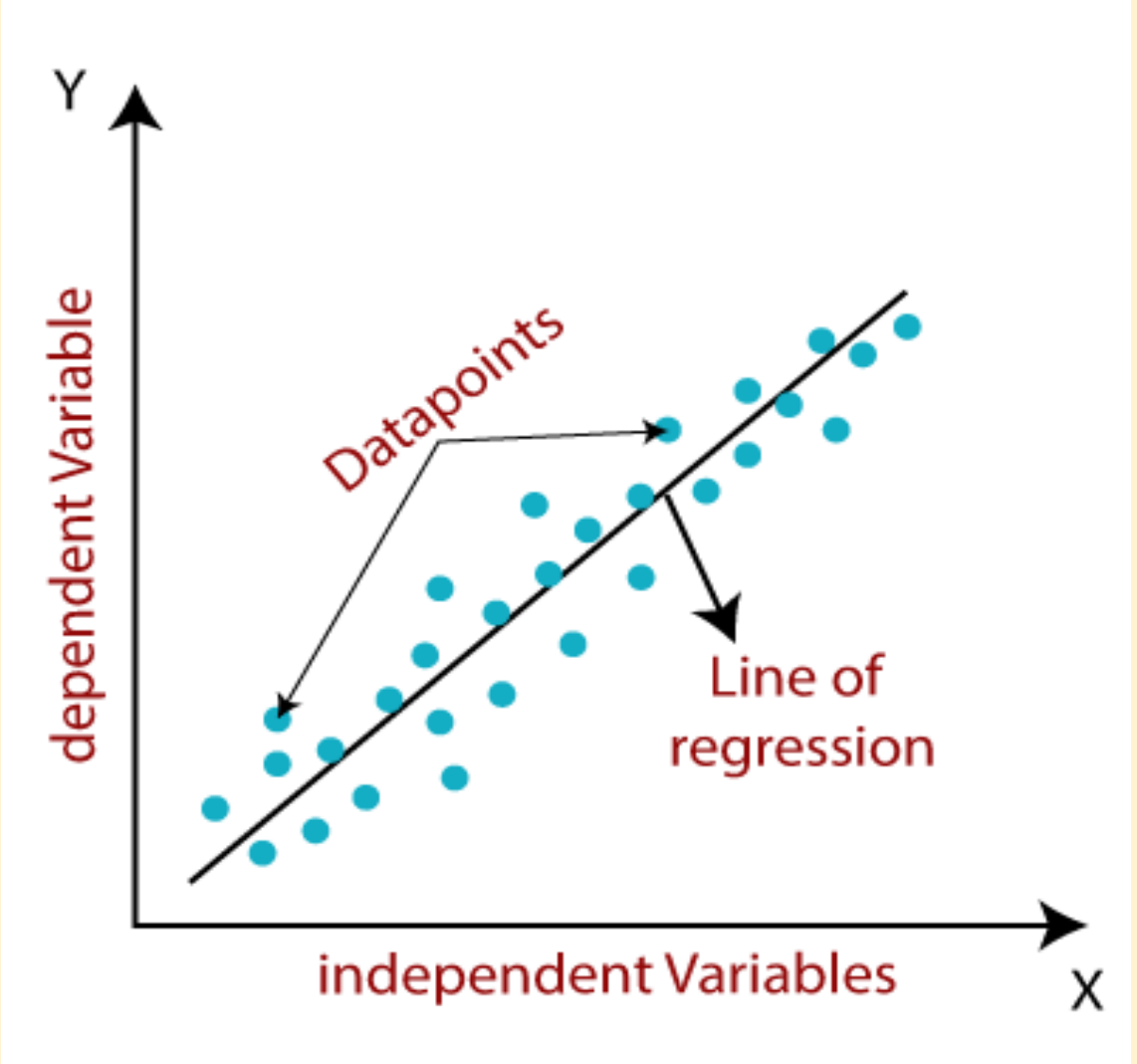
Basic Machine Learning Algorithms

- **Model Evaluation/Validation:** Metrics to assess the performance of a machine learning model (e.g., accuracy, precision, recall, F1-score for classification; MSE, R-squared for regression).
- **Overfitting and Underfitting:** Common problems in machine learning and techniques to address them

Linear Regression

- A fundamental and widely used supervised learning algorithm for regression tasks.
- **Concept:** Models the relationship between a dependent variable (target) and one or more independent variables (features) by fitting a linear equation to the observed data.
- **Simple Linear Regression:** One independent variable ($Y=\beta0+\beta1X$).
- **Multiple Linear Regression:** Multiple independent variables ($Y=\beta0+\beta1X1+\cdots+\beta nXn$).
- **Assumptions:** Linearity, independence of errors, homoscedasticity, normality of residuals.
- **Interpretation of Coefficients:** How much the dependent variable changes for a one-unit change in an independent variable.
- **Applications:** Sales forecasting, predicting prices, understanding relationships between variables.

Linear Regression



Linear Regression

Estimated Regression Model

$$\hat{y} = b_o + b_1 x$$

$\hat{y}$ = Estimated, or predicted y value

b_0 = unbiased estimate of the regression INTERCEPT

b_1 = unbiased estimate of the regression SLOPE

x = value of the independent variable

$$b_0 = \bar{y} - b_1 \bar{x}$$

$$b_1 = \frac{n\sum xy - (\sum x)(\sum y)}{n\sum x^2 - (\sum x)^2}$$

Linear Regression

Example:

The following data represents the number of hours 12 different students watched television during the weekend and the scores of each student who took a test the following Monday.

a.) Find the equation of the regression line.

b.) Use the equation to find the expected test score for a student who watches 9 hours of TV

Hours, x	0	1	2	3	3	5	5	5	6	7	7	10
Test score, y	96	85	82	74	95	68	76	84	58	65	75	50

Example1

Hours, x	0	1	2	3	3	5	5	5	6	7	7	10
Test score, y	96	85	82	74	95	68	76	84	58	65	75	50

$$\sum y = 908 \qquad \sum xy = 3724 \qquad \sum x^2 = 332 \qquad \sum y^2 = 70836$$

$$b_1 = \frac{n\sum xy - (\sum x)(\sum y)}{n\sum x^2 - (\sum x)^2}$$

$$= \frac{12(3724) - (54)(908)}{12(332) - (54)^2} \approx -4.067$$

Example1:Linear Regression

$$b_0 = \bar{y} - b_1\bar{x}$$

$$= \frac{908}{12} - (-4.067)\frac{54}{12}$$

$$\approx 93.97$$

$$\hat{y} = -4.07x + 93.97$$

Validation

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Root Mean Square Error (RMSE) is a commonly used measure of the differences between values predicted by a model and the values actually observed. It is a way to quantify the error in a model's predictions, by calculating the square root of the average of the squared differences between predicted and observed values. A smaller RMSE indicates a better fit of the model to the data

Simple Linear Regression: Loading the Data

In [1]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

In [2]:
```
df = pd.read_csv('placement.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	cgpa	package
0	6.89	3.26
1	5.12	1.98
2	7.82	3.25
3	7.42	3.67
4	6.94	3.57

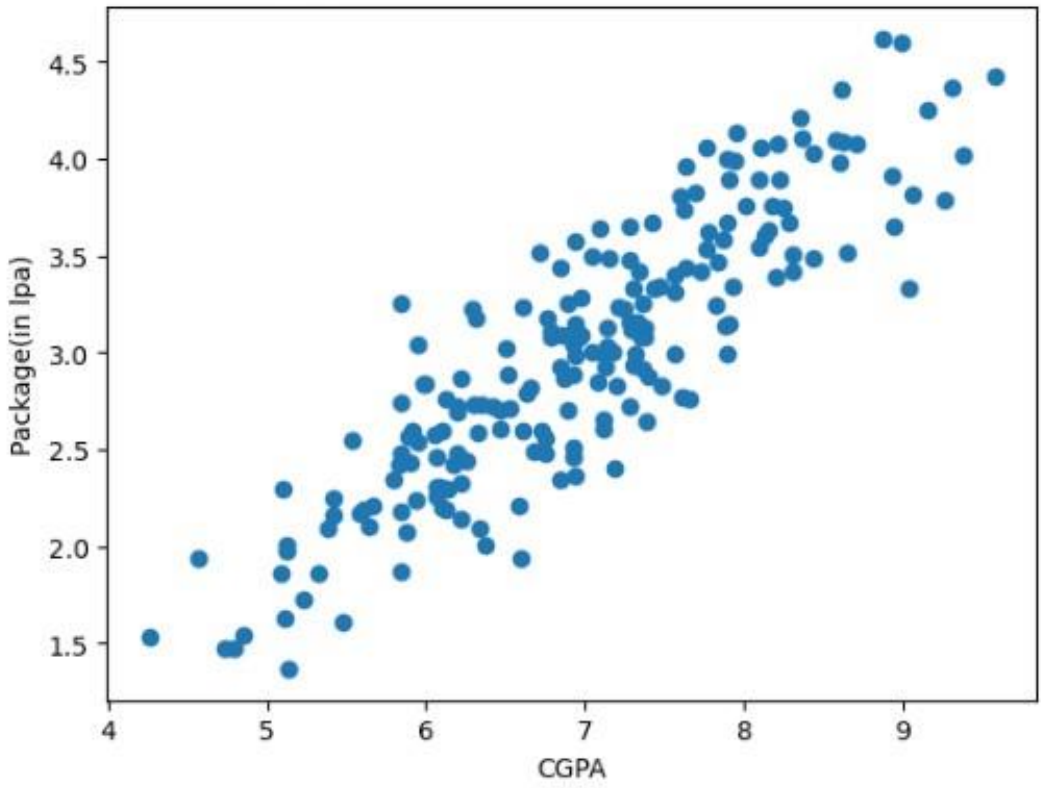
In [4]:
```
plt.scatter(df['cgpa'],df['package'])
plt.xlabel('CGPA')
plt.ylabel('Package(in lpa)')
```

Out[4]:
```
Text(0, 0.5, 'Package(in lpa)')
```

Simple Linear Regression: Scatter Plot of Data

```
In [4]: plt.scatter(df['cgpa'],df['package'])
        plt.xlabel('CGPA')
        plt.ylabel('Package(in lpa)')
```

```
Out[4]: Text(0, 0.5, 'Package(in lpa)')
```



Simple Linear Regression: Splitting the Data

```
In [5]: X = df.iloc[:,0:1]
        y = df.iloc[:,-1]

In [6]: y

Out[6]: 0      3.26
        1      1.98
        2      3.25
        3      3.67
        4      3.57
               ...
        195    2.46
        196    2.57
        197    3.24
        198    3.96
        199    2.33
        Name: package, Length: 200, dtype: float64
```

Simple Linear Regression: Fitting the Regression Line on the Training Data

```
In [7]: from sklearn.model_selection import train_test_split
        X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)

In [8]: from sklearn.linear_model import LinearRegression

In [9]: lr = LinearRegression()

In [10]: lr.fit(X_train,y_train)

Out[10]: LinearRegression
         LinearRegression()
```

Simple Linear Regression: X_Test and Y_Test

```
In [11]: X_test
```

Out[11]:

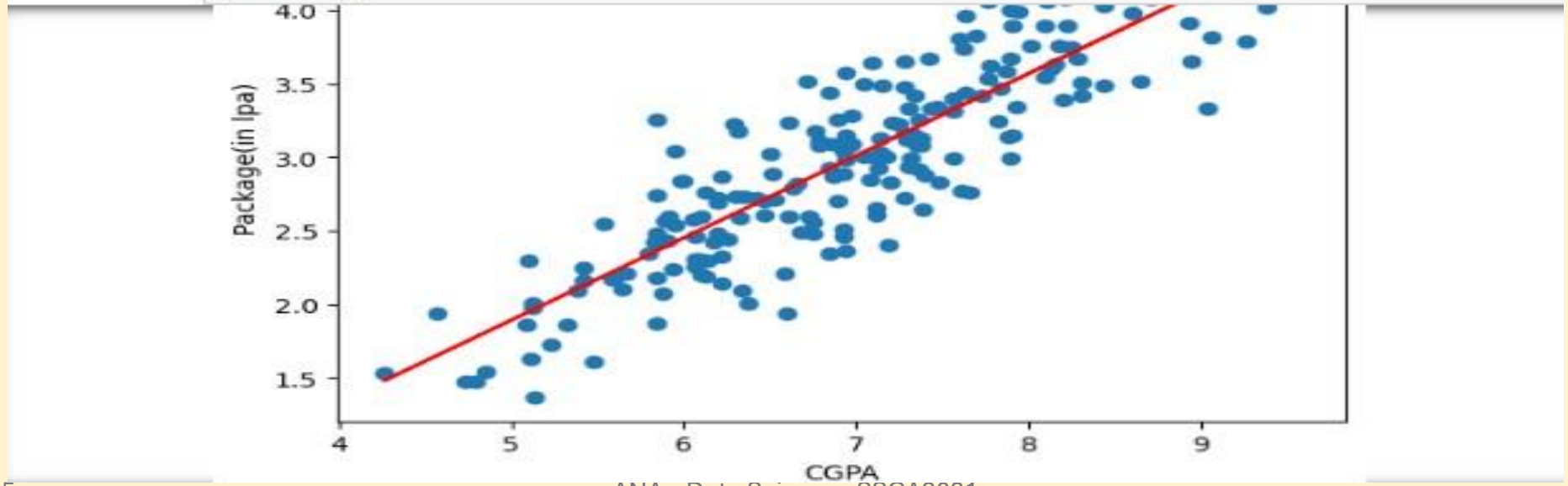
	cgpa
112	8.58
29	7.15
182	5.88
199	6.22
193	4.57
85	4.79
10	5.32
54	6.86
115	8.35

```
In [12]: y_test
```

```
Out[12]: 112    4.10
         29     3.49
         182    2.08
         199    2.33
         193    1.94
         85     1.48
         10     1.86
         54     3.09
         115    4.21
         35     2.87
         12     3.65
         92     4.00
         13     2.89
         126    2.60
         174    2.99
```

Simple Linear Regression Prediction for the Train Data

```
In [13]: X_train.shape
Out[13]: (160, 1)

In [14]: Y_train_Pred=lr.predict(X_train)

In [38]:
plt.scatter(df['cgpa'],df['package'])
plt.plot(X_train,Y_train_Pred,color='red')
plt.xlabel('CGPA')
plt.ylabel('Package(in lpa)')
plt.show()
```



Simple Linear Regrssion :Obtaining the Best Fit Line

```
In [44]: m = lr.coef_
         print(m)

         [0.55795197]

In [45]: b = lr.intercept_
         print(b)

         -0.8961119222429144

In [17]: print(f"The equation of the line is: y = {m}x + {b}")

         The equation of the line is: y = [0.55795197]x + -0.8961119222429144
```

Simple Linear Regression : Prediction from Best Fit Regression Line

```
In [18]: # y = mx + b

         m * 8.58 + b

Out[18]: array([3.89111601])

In [19]: m * 9.5 + b

Out[19]: array([4.40443183])

In [20]: m * 100 + b

Out[20]: array([54.89908542])
```

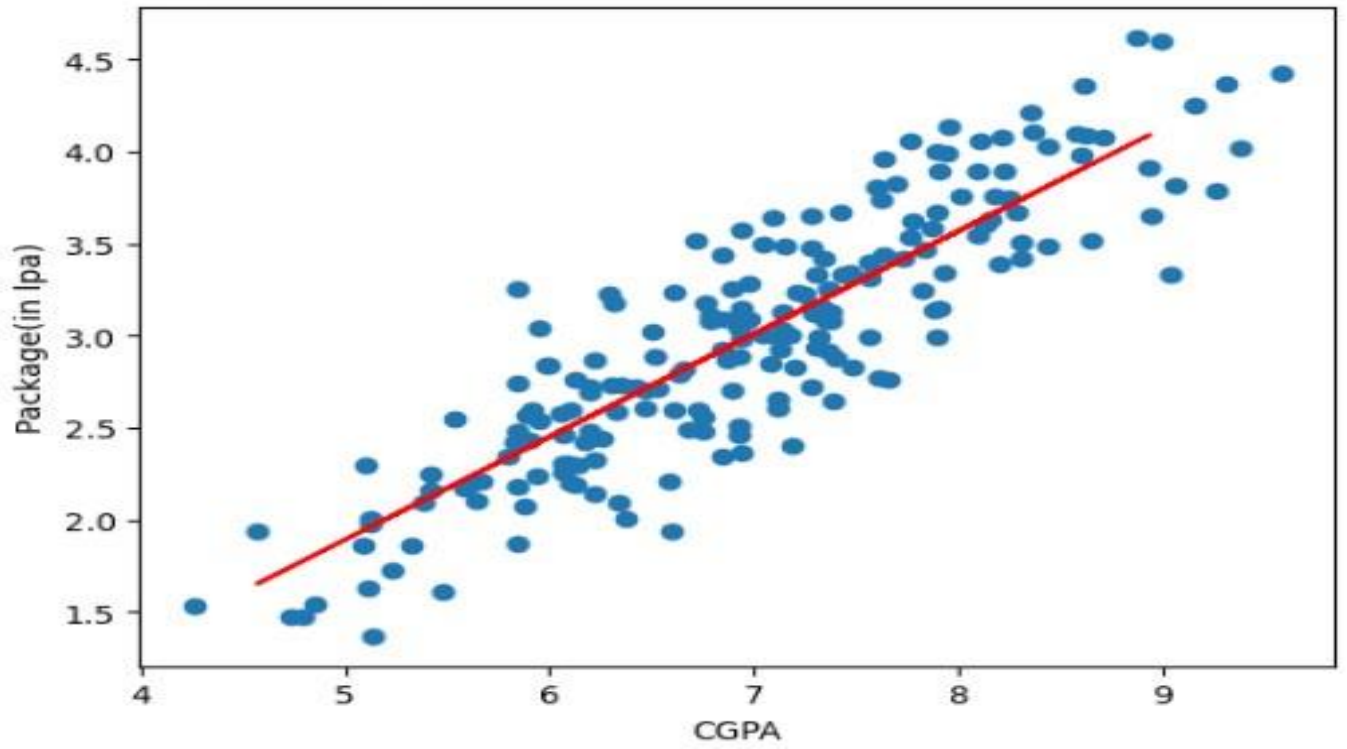
Simple Linear Regression :Prediction from the model on the Test Data

```
In [32]: X_test.shape

Out[32]: (40, 1)

In [36]: Y_test_Pred=lr.predict(X_test)

In [37]: plt.scatter(df['cgpa'],df['package'])
         plt.plot(X_test,Y_test_Pred,color='red')
         plt.xlabel('CGPA')
         plt.ylabel('Package(in lpa)')

Out[37]: Text(0, 0.5, 'Package(in lpa)')
```

Simple Linear Regression Scatter Plot for Test Data

```
In [37]: plt.scatter(df['cgpa'],df['package'])
         plt.plot(X_test,Y_test_Pred,color='red')
         plt.xlabel('CGPA')
         plt.ylabel('Package(in lpa)')
```

```
Out[37]: Text(0, 0.5, 'Package(in lpa)')
```



Simple Linear Regrssion :Validation

```
In [27]: from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score

In [30]: print("MAE",mean_absolute_error(y_test,Y_test_Pred))
         MAE 0.2884710931878175

In [31]: print("MSE",mean_squared_error(y_test,Y_test_Pred))
         MSE 0.1212923531349527

In [33]: print("RMSE",np.sqrt(mean_squared_error(y_test,Y_test_Pred)))
         RMSE 0.34827051717731616
```

Simple Linear Regression: Validation

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

The R-squared (R^2) or coefficient of determination is a statistical measure that represents the proportion of variance in the dependent variable that can be explained by the independent variables in a regression model.

The R-squared (R^2) score, also known as the coefficient of determination, ranges from 0 to 1. A value of 0 indicates that the model explains none of the variability in the dependent variable, while a value of 1 signifies that the model perfectly explains all the variability.

SVM (Support Vector Machine)

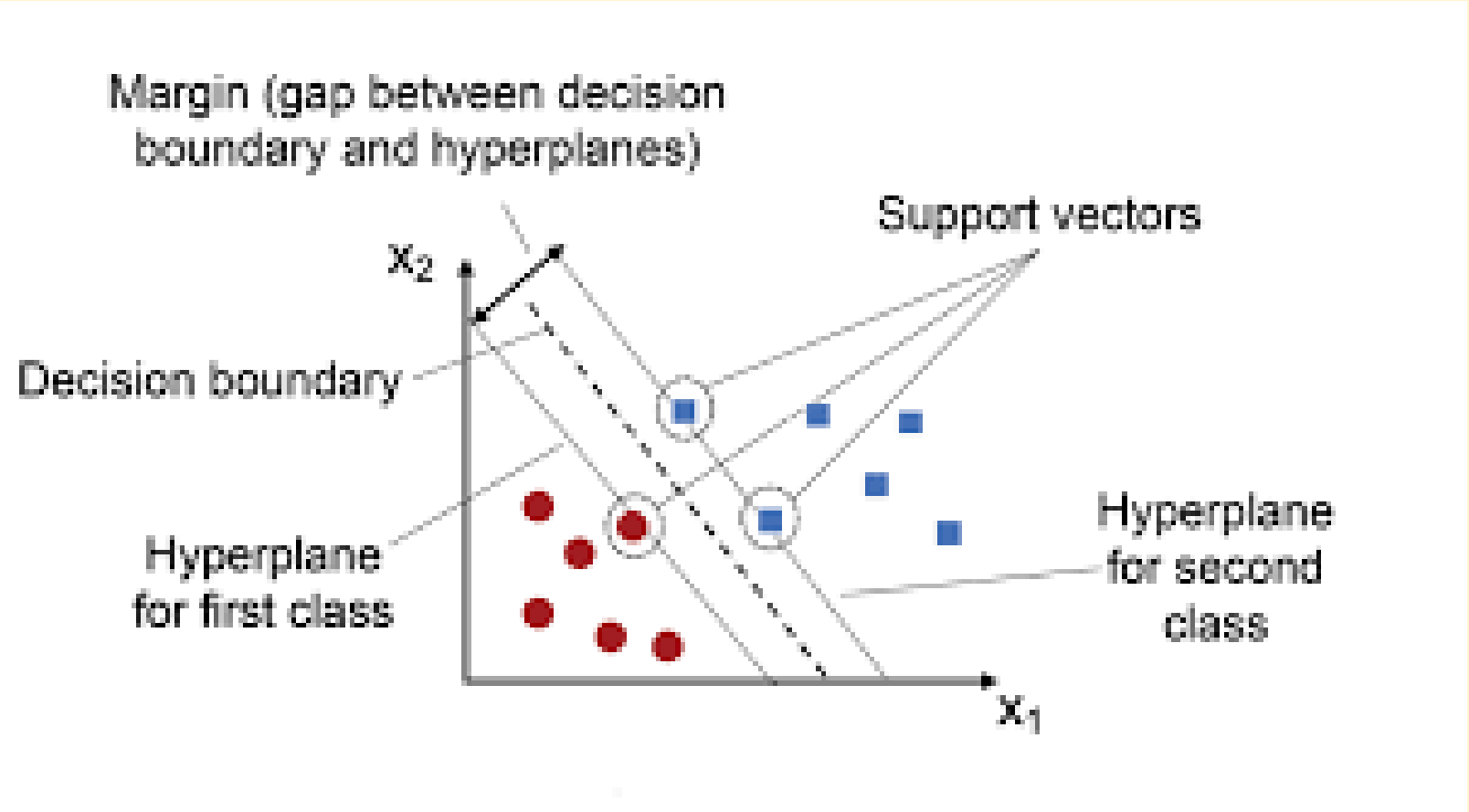
A powerful supervised learning algorithm primarily used for classification, but also for regression.

- **Concept:** SVM works by finding the optimal hyperplane that best separates data points of different classes in a high-dimensional space. The "support vectors" are the data points closest to the hyperplane, which play a crucial role in defining it.
- **Support Vectors:** These are the data points that are closest to the decision boundary (hyperplane) and are crucial in defining the margin. They are the most influential training records and determine the position and orientation of the optimal hyperplane.
- **Maximal Margin Classifier:** The goal is to maximize the margin between the decision boundary and the support vectors, leading to better generalization.

SVM (Support Vector Machine)

- **Kernel :** Allows SVM to classify non-linearly separable data by mapping it into a higher-dimensional feature space where a linear separation is possible (e.g., polynomial, radial basis function (RBF) kernels).
- **Applications:** Image classification, text categorization, bioinformatics.

SVM (Support Vector Machine)



SVM (Support Vector Machine): Loading the Data

In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
from sklearn.model_selection import train_test_split
```

In [3]:
```
from sklearn.svm import SVC
```

In [4]:
```
df=pd.read_csv('Iris.csv')
df.head()
```

Out[4]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

SVM (Support Vector Machine): Exploring the Data

```
In [6]: df.shape
Out[6]: (150, 6)

In [7]: df.nunique()
Out[7]: Id               150
        SepalLengthCm     35
        SepalWidthCm      23
        PetalLengthCm     43
        PetalWidthCm      22
        Species            3
        dtype: int64

In [8]: df.Species.unique()
Out[8]: array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)

In [9]: df.isna().sum()
Out[9]: Id               0
        SepalLengthCm    0
        SepalWidthCm     0
        PetalLengthCm    0
        PetalWidthCm     0
        Species          0
        dtype: int64
```

SVM (Support Vector Machine): Encoding

```
In [10]: df.Species.replace(('Iris-setosa', 'Iris-versicolor', 'Iris-virginica'), (1,2,3), inplace=True)
```

```
In [11]: df.head()
```

Out[11]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	1
1	2	4.9	3.0	1.4	0.2	1
2	3	4.7	3.2	1.3	0.2	1
3	4	4.6	3.1	1.5	0.2	1
4	5	5.0	3.6	1.4	0.2	1

SVM (Support Vector Machine): Exploring the Data

```
In [12]: df.describe()
```

Out[12]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
count	150.000000	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667	2.000000
std	43.445368	0.828066	0.433594	1.764420	0.763161	0.819232
min	1.000000	4.300000	2.000000	1.000000	0.100000	1.000000
25%	38.250000	5.100000	2.800000	1.600000	0.300000	1.000000
50%	75.500000	5.800000	3.000000	4.350000	1.300000	2.000000
75%	112.750000	6.400000	3.300000	5.100000	1.800000	3.000000
max	150.000000	7.900000	4.400000	6.900000	2.500000	3.000000

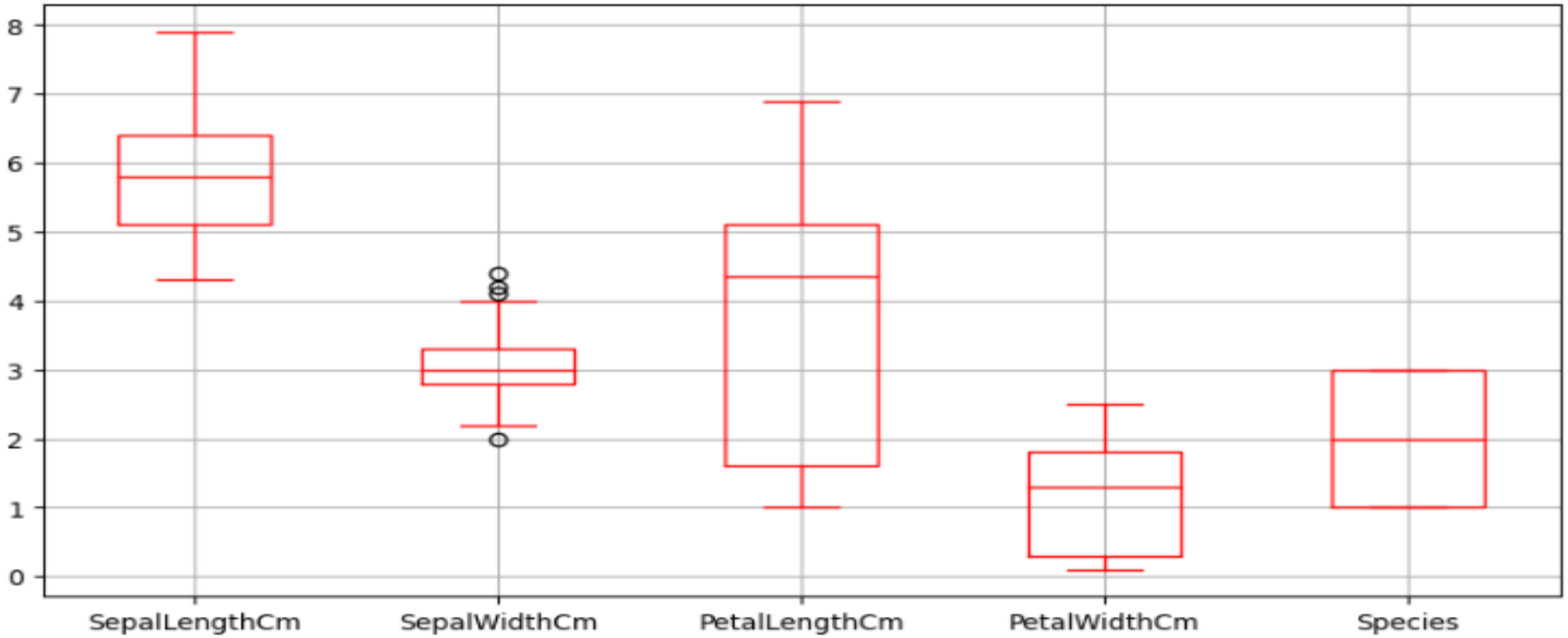
```
In [13]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Id              150 non-null    int64
 1   SepalLengthCm   150 non-null    float64
 2   SepalWidthCm    150 non-null    float64
 3   PetalLengthCm   150 non-null    float64
 4   PetalWidthCm    150 non-null    float64
 5   Species         150 non-null    int64
dtypes: float64(4), int64(2)
memory usage: 7.2 KB
```

SVM (Support Vector Machine): Box Plot of the Data

```
In [14]: plt.figure(figsize = (10, 5))
         df_modified=df.drop(columns=['Id'])
         df_modified.boxplot(color='red')
```

```
Out[14]: <AxesSubplot:>
```

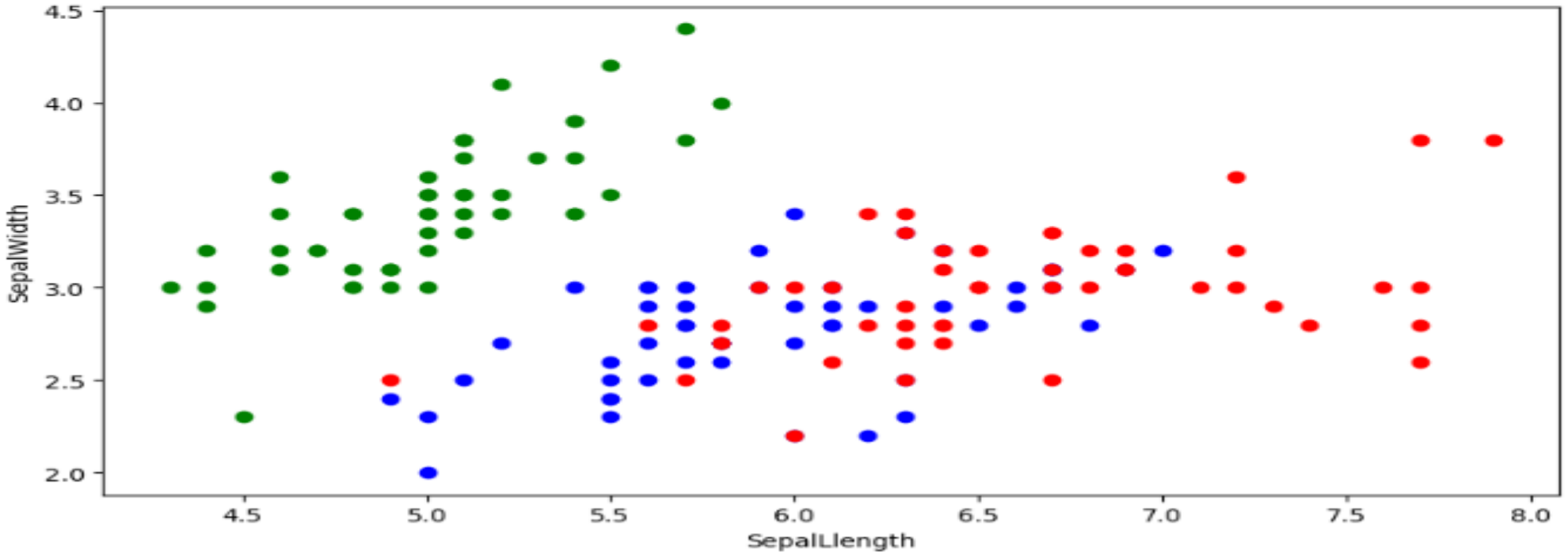


SVM (Support Vector Machine): Scatter Plot from the Data

```
In [15]: df1=df[df.Species==1]
         df2=df[df.Species==2]
         df3=df[df.Species==3]
```

```
In [19]: plt.figure(figsize = (10, 5))
         plt.xlabel('SepalLength')
         plt.ylabel('SepalWidth')
         plt.scatter(x=df1['SepalLengthCm'],y=df1['SepalWidthCm'], color='green')
         plt.scatter(x=df2['SepalLengthCm'],y=df2['SepalWidthCm'], color='blue')
         plt.scatter(x=df3['SepalLengthCm'],y=df3['SepalWidthCm'], color='red')
```

```
Out[19]: <matplotlib.collections.PathCollection at 0x1591c229850>
```



SVM (Support Vector Machine): Splitting the Data

```
In [20]: x=df.drop(['Species'],axis='columns')
         x.head()
```

Out[20]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	1	5.1	3.5	1.4	0.2
1	2	4.9	3.0	1.4	0.2
2	3	4.7	3.2	1.3	0.2
3	4	4.6	3.1	1.5	0.2
4	5	5.0	3.6	1.4	0.2

```
In [21]: y=df.Species
         y.head()
```

```
Out[21]: 0    1
         1    1
         2    1
         3    1
         4    1
         Name: Species, dtype: int64
```

SVM (Support Vector Machine): Splitting the Data

```
In [22]: x_train,x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2, random_state = 2)
```

```
In [23]: print(len(x_train))
         print(len(y_train))
```

```
120
120
```

```
In [24]: print(len(x_test))
         print(len(y_test))
```

```
30
30
```

SVM (Support Vector Machine): Fitting the classifier model on the Data

```
In [25]: model = SVC()

In [26]: model.fit(x_train,y_train)

Out[26]: SVC
         SVC()
```

SVM (Support Vector Machine): Prediction from the model ,Accuracy, Confusion Matrix

```
In [27]: y_test_pred= model.predict(x_test)
```

```
In [39]: accuracy_score(y_test,y_test_pred )
```

```
Out[39]: 0.9666666666666667
```

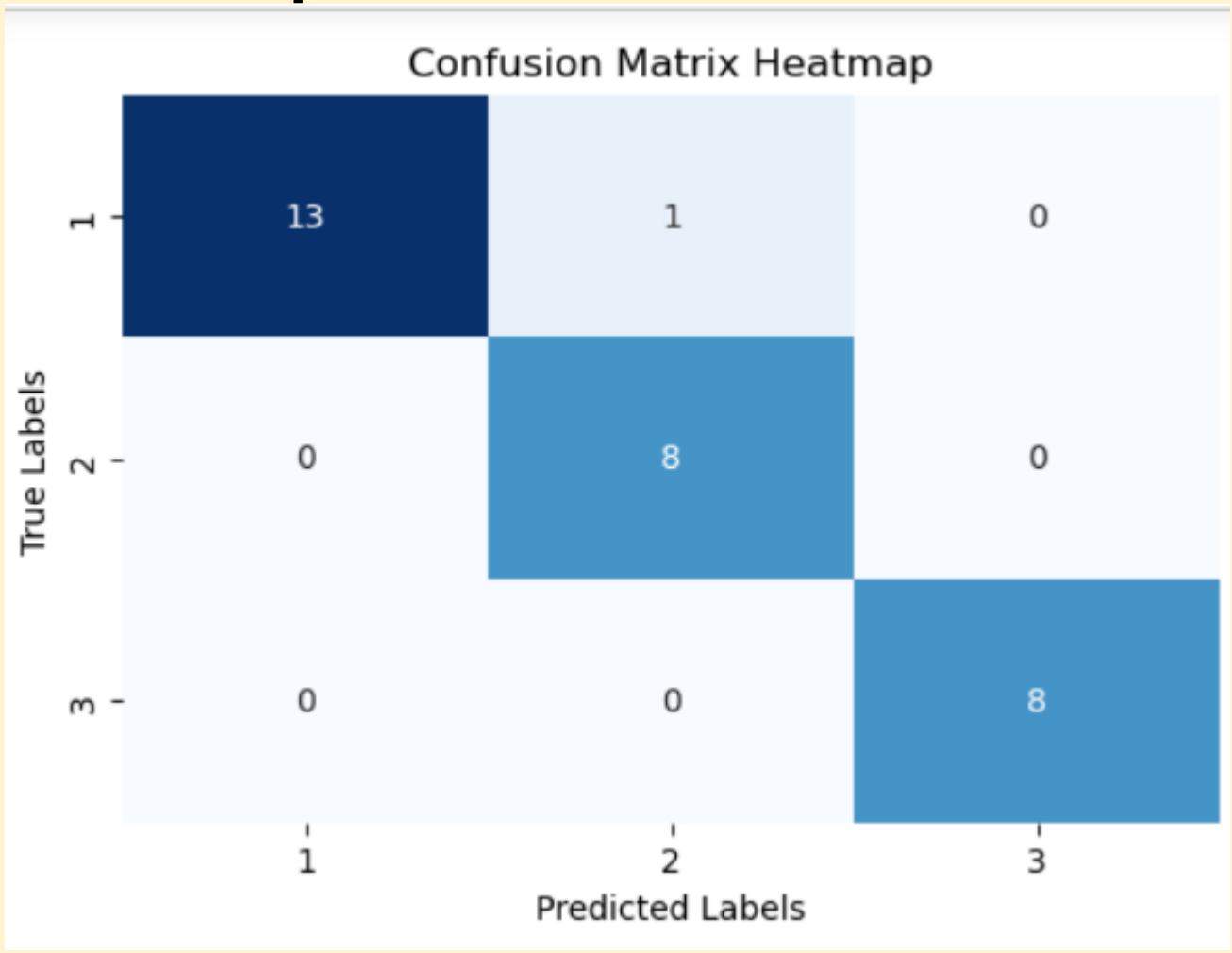
```
In [29]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
         print("\nConfusion Matrix:")
         print(confusion_matrix(y_test, y_test_pred))
```

```
Confusion Matrix:
[[13  1  0]
 [ 0  8  0]
 [ 0  0  8]]
```

SVM (Support Vector Machine): Confusion Matrix Heatmap

In [28]:
```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

# Compute confusion matrix
cm = confusion_matrix(y_test, y_test_pred, labels=[1, 2, 3])

# Plot heatmap
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False, xticklabels=[1,2,3], yticklabels=[1,2,3]
plt.title('Confusion Matrix Heatmap')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```

SVM (Support Vector Machine): Confusion Matrix Heatmap



SVM (Support Vector Machine): Validation

In [35]:
```
print("\nClassification Report:")
print(classification_report(y_test, y_test_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           1       1.00      0.93      0.96        14
           2       0.89      1.00      0.94         8
           3       1.00      1.00      1.00         8

    accuracy                           0.97        30
   macro avg       0.96      0.98      0.97        30
weighted avg       0.97      0.97      0.97        30
```

Naive Bayes

- A probabilistic supervised learning algorithm for classification tasks, based on Bayes' theorem.
- **Concept:** It assumes that the features are conditionally independent of each other given the class label (hence "naive"). Despite this strong assumption, it often performs surprisingly well, especially with large datasets.
- **Bayes' Theorem:** $P(A|B)=P(A)P(B|A)/P(B)$. In Naive Bayes, this is used to calculate the probability of a class given the observed features.
- **Types:** Gaussian Naive Bayes (for continuous data), Multinomial Naive Bayes (for count data, e.g., text classification), Bernoulli Naive Bayes (for binary features).
- **Advantages:** Simple, fast to train, works well with high-dimensional data, performs well with small training datasets.
- **Applications:** Spam filtering, sentiment analysis, document classification.
- This comprehensive set of topics provides a strong foundation in both statistical principles and practical machine learning techniques essential for effective data analysis.

Naïve Bayes

Naive Bayes

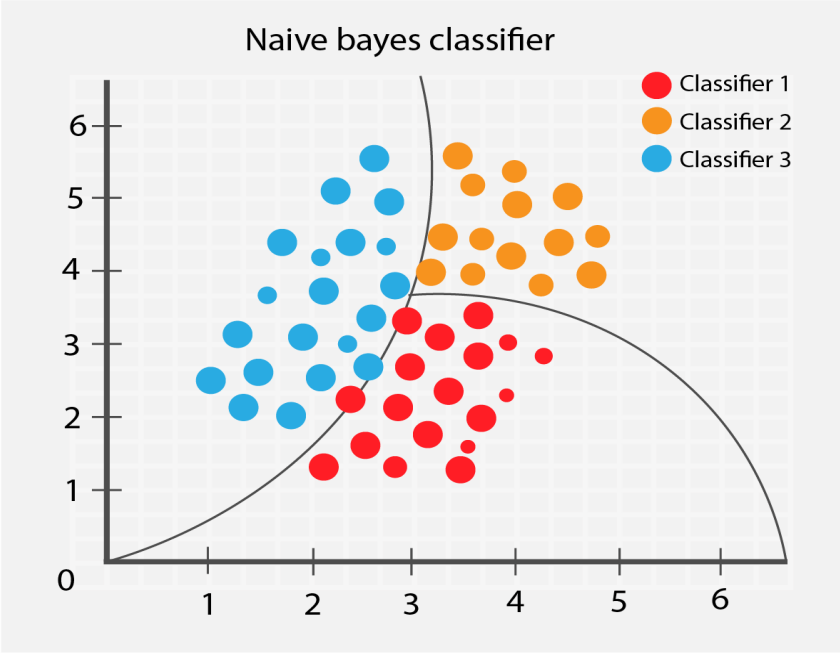
thatware.co

In machine learning, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

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

using Bayesian probability terminology, the above equation can be written as

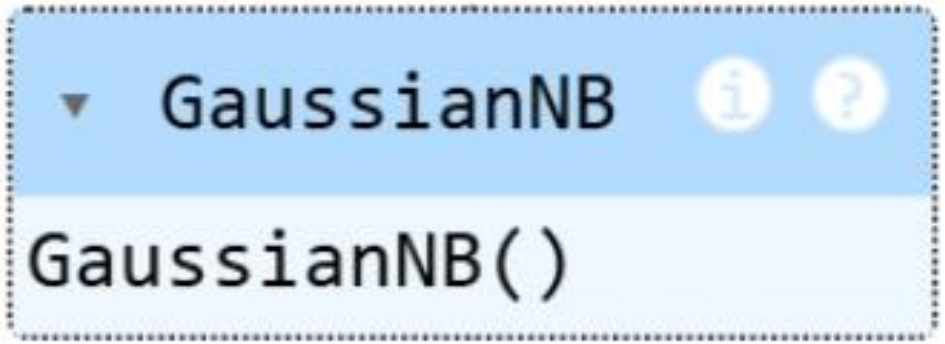
$$\text{Posterior} = \frac{\text{prior x likelihood}}{\text{evidence}}$$



Model Training And Prediction –Gaussian Naïve Bayes

```
gnb = GaussianNB()

gnb.fit(X_train, y_train)
```

Output:



GaussianNB

Model Training And Prediction –Gaussian Naïve Bayes

```
y_pred = gnb.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)
print(f"The Accuracy of Prediction on Iris Flower is:
{accuracy}")
```

Output:

The Accuracy of Prediction on Iris Flower is: 0.9777777777777777