Unit -3

National Forensic Sciences University

Knowledge | Wisdom | Fulfilment

An Institution of National Importance
(Ministry of Home Affairs, Government of India)

By Ms. Drashti Garadharia
National Forensic Science University, Gandhinagar

Overview

- Association Rule Mining
- Types of ML Algorithm
- Classification vs Regression
- Linear Regression & Logistic Regression
- K-Nearest Neighbors(k-NN)
- Decision Tree – Naïve Bayes
- Ensemble Method Random Forest
- Feature – Generation and Selection
- Filters, Wrappers

Association Rule Learning

Association rule learning is a type of unsupervised learning technique. It tries to find some interesting relations or associations among the variables of dataset. It is based on different rules to discover the interesting relations between variables in the database.

Application : Market Basket analysis,

Web usage mining, etc,

How does Association Rule Learning work?

Association rule learning works on the concept of If and Else Statement, such as if A then B.

Here the If element is called ANTECEDENT, and then statement is called as CONSEQUENT.

These types of relationships where we can find out some association or relation between two items is known as single cardinality.

To measure the associations between Multiple of data items following metrics used

B->C

1. Support

$$\text{Supp(X)} = \frac{Freq(X)}{T}$$

1. Confidence

$$\text{Confidence} = \frac{Freq(X,Y)}{Freq(X)}$$

1. Lift

$$\text{Lift} = \frac{Supp(X,Y)}{Supp(X) \times Supp(Y)}$$

Supp(x,y) = freq(x,y)/T

TID	ITEMSETS
T1	A, B
T2	B, D
T3	B, C
T4	A, B, D
T5	A, C
T6	B, C
T7	A, C
T8	A, B, C, E
T9	A, B, C

Lift Indicates

- **If Lift= 1: The probability of occurrence of antecedent and consequent is independent of each other.**
- **Lift>1: It determines the degree to which the two itemsets are dependent to each other.**
- **Lift<1: It tells us that one item is a substitute for other items, which means one item has a negative effect on another**

Basics of Machine Learning Algorithms

Machine Learning (ML) is a subset of artificial intelligence that involves training computers to learn from data without being explicitly programmed. This learning process allows machines to identify patterns, make predictions, and perform tasks that would typically require human intervention.

Key Components of a Machine Learning Algorithm

1. **Data:** The raw material for ML algorithms. It can be structured (e.g., CSV files) or unstructured (e.g., images, text).
2. **Features:** The relevant attributes or characteristics extracted from the data.
3. **Model:** A mathematical representation that learns a mapping function between the features and the target variable.
4. **Algorithm:** The procedure or method used to train the model.

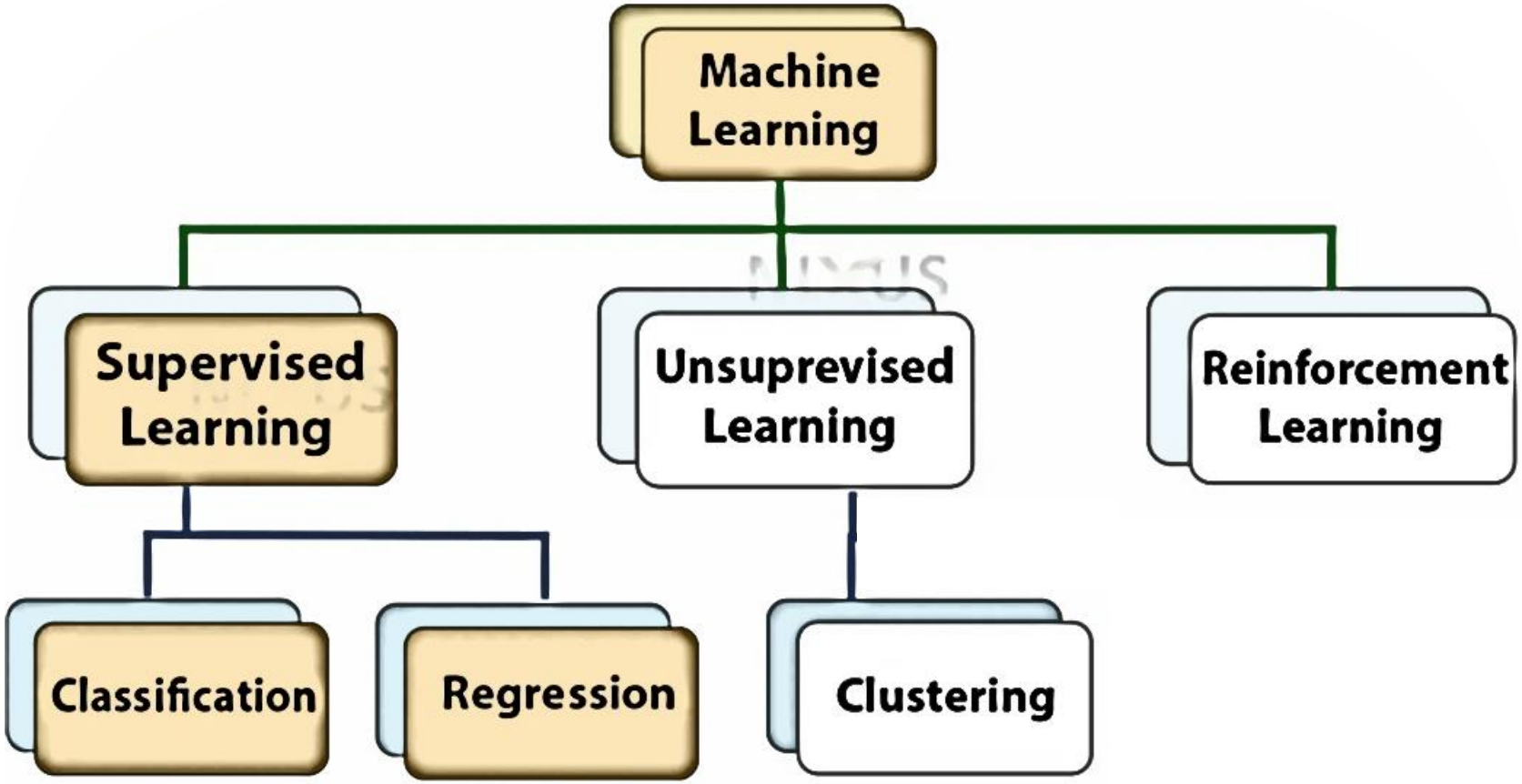
Types of Machine Learning Algorithms

1. **Supervised Learning:**
   - **Regression: Predicting a continuous numerical value (e.g., house prices).**
     Ex- Linear regression, logistic regression, decision trees, random forests.
   - **Classification: decision trees. random forests, neural networks.**

2. **Unsupervised Learning:**
   - **Clustering: Grouping data points into similar clusters.**
     Ex- K-means clustering, hierarchical clustering, DBSCAN.

3. **Reinforcement Learning: Learning through trial and error, interacting with an environment and receiving rewards or penalties.**

Types of ML Algorithms



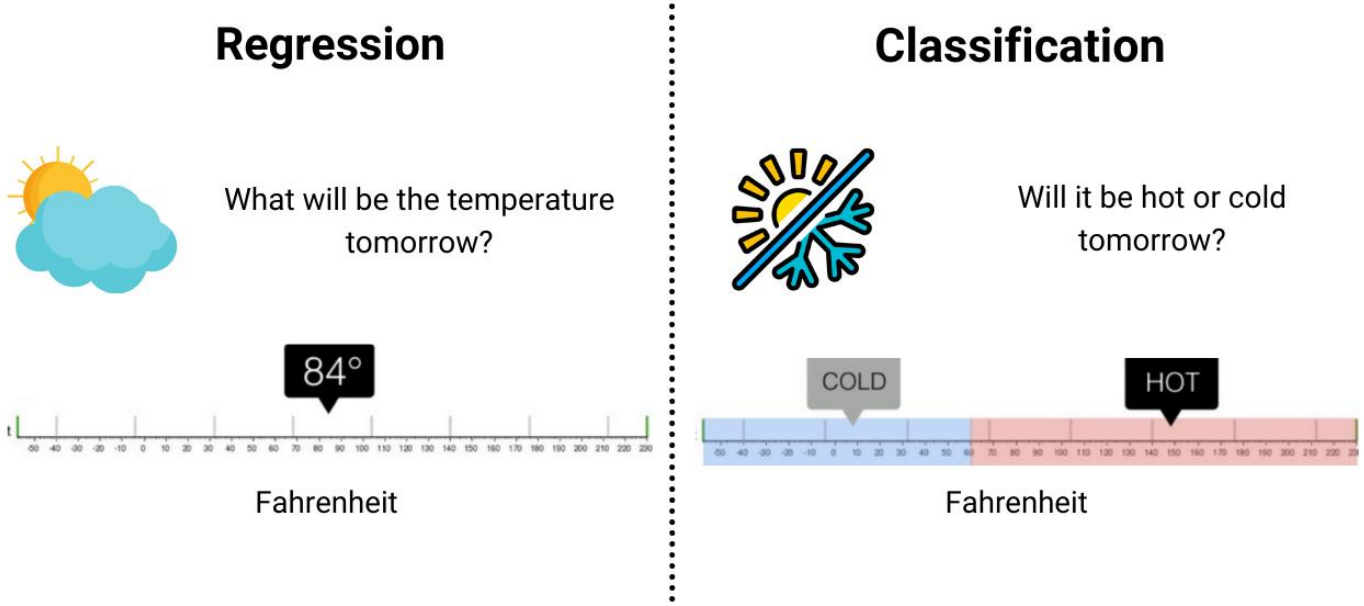
Regression vs Classification

Linear regression is used to predict the relationship between two variables by applying a linear equation to observed data. Linear regression is commonly used for predictive analysis.

The main idea of regression is to examine two things.

First, does a set of predictor variables do a good job in predicting an outcome?

Second, which variables are significant predictors of the outcome variable?



Regression vs Classification vs Clustering

Regression	Classification	Clustering
• Supervised Learning	• Supervised Learning	• Unsupervised Learning
• Output is a continuous quantity	• Output is a categorical quantity	• Assigns data points into clusters
• Main aim is to forecast or predict	• Main aim is to compute the category of the data	• Main aim is to group similar items clusters
• Eg: Predict stock market price	• Eg: Classify emails as spam or non-spam	• Eg: Find all transactions which are fraudulent in nature
• Algorithm: Linear Regression	• Algorithm: Logistic Regression	• Algorithm: K-means

Types of Regression Algorithms

- Linear regression
- Random Forest regression
- Decision Tree regression

Use cases of Regression

- Age prediction
- Weather prediction
- Market trend prediction, etc.

Linear Regression

Linear regression is used to predict the relationship between two variables by applying a linear equation to observed data. Linear regression is commonly used for predictive analysis.

The main idea of regression is to examine two things.

First, does a set of predictor variables do a good job in predicting an outcome?

Second, which variables are significant predictors of the outcome variable?

$$m = \frac{\sum_{i=1}^{n} (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \ldots\ldots\ldots\ldots\ldots (i),$$

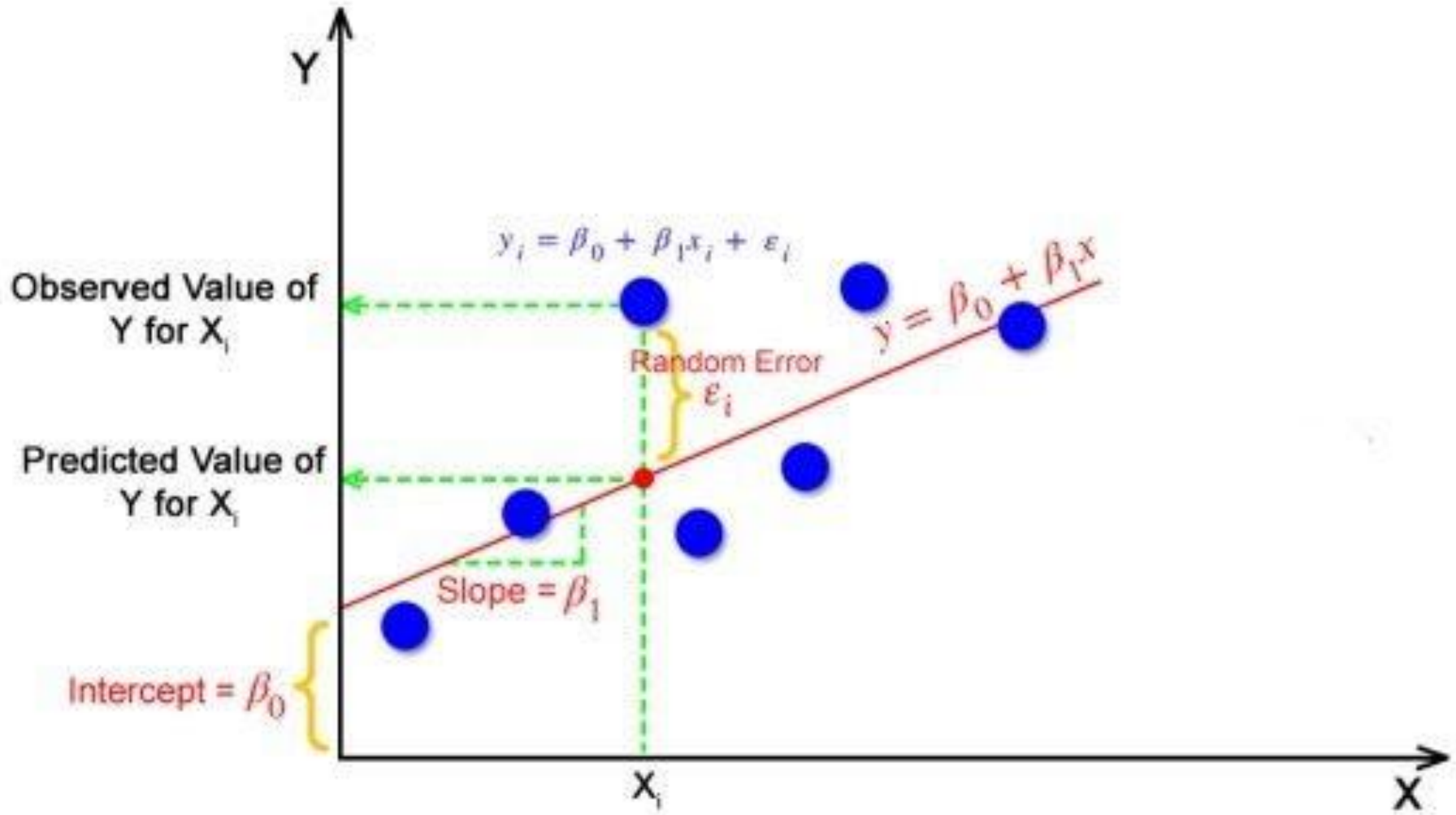
and

$$c = \bar{y} - m\bar{x} \ldots\ldots\ldots\ldots\ldots\ldots (ii)$$

$B_0 = C$

$B_1 = m$

$y=mx+c$


Y
$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$
Observed Value of Y for X_i
Random Error ε_i
$y = \beta_0 + \beta_1 x$
Predicted Value of Y for X_i
Slope = β_1
Intercept = β_0
X_i
X

LR – Practical Manual

```
import numpy as np
import matplotlib.pyplot as plt

# Given data
x = np.array([1500, 1600, 1700, 1800, 1900])
# Square Footage
y = np.array([245, 312, 279, 308, 335])
 # Prices in thousands of dollars

# Calculating the means of x and y
mean_x = np.mean(x)
mean_y = np.mean(y)

# Calculating slope (m) and intercept (c)
m = np.sum((x - mean_x) * (y - mean_y)) / np.sum((x - mean_x) ** 2)
c = mean_y - m * mean_x

print(f"Slope (m): {m}")
print(f"Intercept (c): {c}")

# Predicting prices using the linear regression model
y_pred = m * x + c

# Calculating Mean Squared Error (MSE)
mse = np.mean((y - y_pred) ** 2)
print(f"Mean Squared Error (MSE): {mse}")

plt.scatter(x,y)
plt.plot(x,y_pred)
plt.show()
```

Linear Regression using sklearn lib

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
# load titaninc dataset
df = df.dropna()
x=df.Age
y=df.Fare
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=2)
lr = LinearRegression()
lr.fit(x_train.values.reshape(146,1),y_train.values.reshape(146,1)) #numer of row : 146
y_pred=lr.predict(x_test.values.reshape(37,1))
plt.scatter(x_test,y_test)
plt.plot(x_test,y_pred , c='red')
plt.title("Actual vs. Predicted Fares")
plt.show()
```

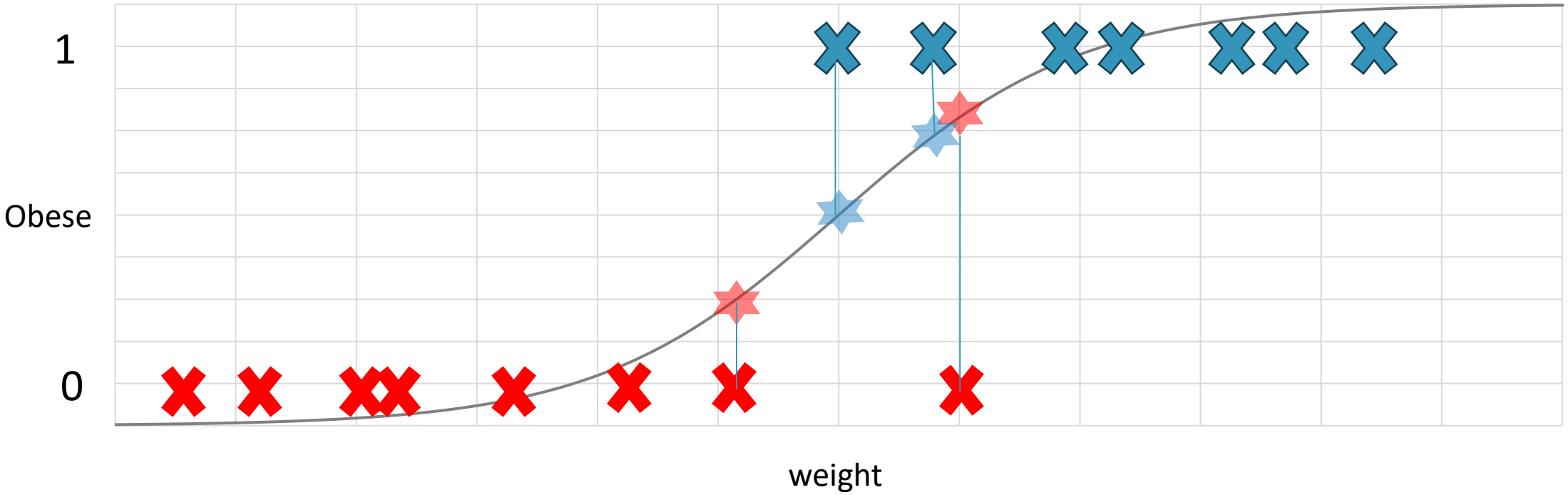
Types of classification Algorithms

- Logistic Regression
- K-Nearest Neighbours
- Naive Bayes
- Decision Tree classification
- Random Forest classification

Use cases of Classification

- Email spam detection
- Disease Detection
- Speech Recognition
- Face recognition, etc

Classification Problem : Logistic Regression

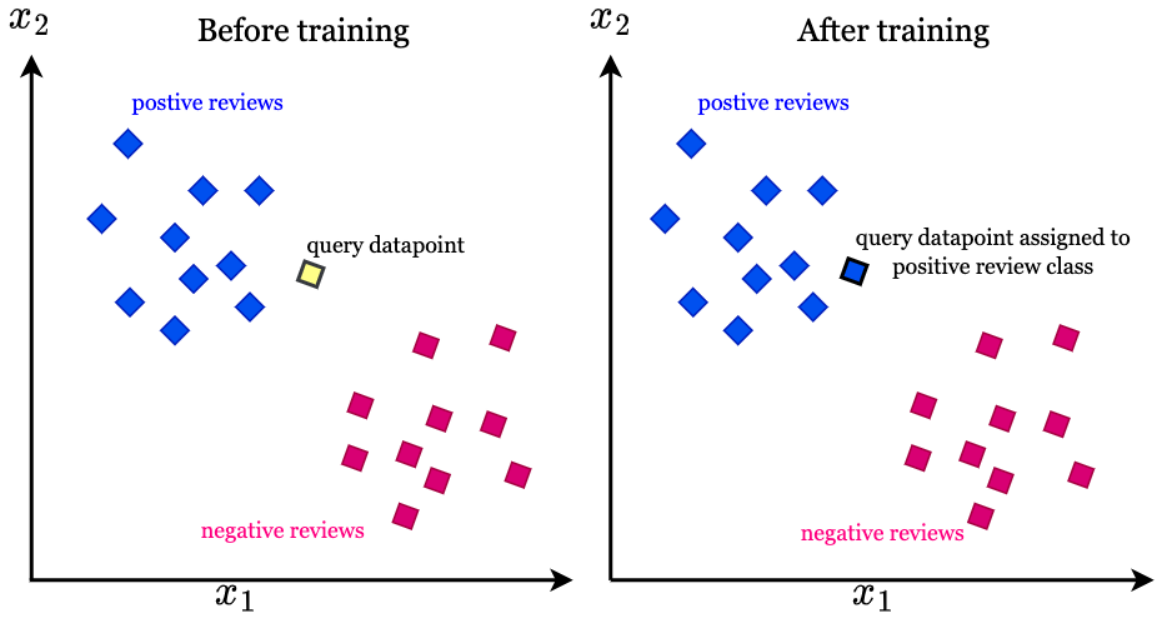


LogisticRegression – using Sklearn

```
import numpy as np

from sklearn.linear_model import
LogisticRegression

import matplotlib.pyplot as plt


# Data: Hours studied (X) and whether the
student passed (Y)

X = np.array([[1], [2], [3], [4], [5], [6],
[7]])

# Hours studied

Y = np.array([0, 0, 0, 1, 1, 1, 1])

# Pass (1) or Fail (0)


# Initialize the logistic regression model

model = LogisticRegression()

# Fit the model

model.fit(X, Y)

x_test=np.linspace(0,8,10).reshape(10,1)

Prob_passing = model.predict_proba(x_test)[:,1]

plt.scatter(X, Y, color='red', label='Data
points')

plt.plot(x_test, Prob_passing, color='blue',
label='Logistic Regression Curve')

plt.xlabel('Hours Studied')

plt.ylabel('Probability of Passing')

plt.title('Logistic Regression: Probability of
Passing vs. Hours Studied')

plt.grid(True)

plt.show()
```

K-Nearest Neighbour

The k-nearest neighbors (KNN) algorithm is a supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point




```
import matplotlib.pyplot as plt


x = [4, 5, 10, 4, 3, 11, 14 , 8, 10, 12]
y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]
classes = [0, 0, 1, 0, 0, 1, 1, 0, 1, 1]


plt.scatter(x, y, c=classes)
plt.show()
```

K-Nearest Neighbour

The value of k affects the model's error rate and performance:

- Overfitting

  A small value of k can cause overfitting, which means the model performs well on training data but poorly on new data.

- Underfitting

  A large value of k can cause underfitting, which means the model doesn't perform well on training data and can't be generalized to new data.

- Ties

  Choosing an **odd value of k** can help avoid ties in classification.

Here are some tips for choosing the value of k:

- Square root of n: Choose k as the square root of the total number of data points.
- Bootstrap method: In binary classification problems, the bootstrap method is a popular way to choose the optimal k.
- Cross-validation: Use cross-validation to select the best value of k and improve performance.

K-Nearest Neighbour

USED For:

Data Preprocessing (As simple Data Imputer – to handle null value)

Forecasting (Finance / Healthcare)

Advantage:

Easy to implement

Few Hyperparameter (Distance Matrix, k-value)

Adaptable

Disadvantage:

Scale (require whole data so time taking process)

Curse of Dimensionality (add unnecessary column which might add noise)

Overfitting (on lower k values)

Underfitting (on higher k values)

K-Nearest Neighbour

```
from sklearn.neighbors import KNeighborsClassifier

data = list(zip(x, y))      // convert like [(4,5),(8,9)]
knn = KNeighborsClassifier(n_neighbors=1)

knn.fit(data, classes)

new_x = 8
new_y = 21
new_point = [(new_x, new_y)]

prediction = knn.predict(new_point)   //[0] or [1] sklearn provide result in array

plt.scatter(x + [new_x], y + [new_y], c=classes + [prediction[0]])
plt.show()
```

Naïve Bayes

Email	Word1	Word2	Word3	Class
Email1	spam	offer	money	Spam
Email2	free	gift	win	Spam
Email3	meeting	project	deadline	Not Spam
Email4	sale	discount	buy	Spam
Email5	work	task	complete	Not Spam

Naïve Bayes

Step 1: Calculate Probabilities

Prior probabilities:

P(Spam) = 3/5 = 0.6

P(Not Spam) = 2/5 = 0.4

Step 2: Classify a New Email

New email: "free offer buy"

Calculate probabilities:

P(Class=Spam | Word1=free, Word2=offer, Word3=busy) =
P(Spam) * P(Word1=free | Spam) * P(Word2=offer | Spam) * P(Word3=buy| Spam)

P(Class=Not Spam | Word1=free, Word2=offer, Word3=busy) =
P(Not Spam) * P(Word1=free | Not Spam) * P(Word2=offer | Not Spam) * P(Word3=buy | Not Spam)

Compare probabilities and assign the class with the highest probability.

Naïve Bayes

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

Naive Bayes is a simple probabilistic classifier based on Bayes' theorem.

It assumes that the features of a given data point are **independent** of each other, which is often not the case in reality.

Used in :

- text classification. In text classification tasks, data contains high dimension (as each word represent one feature in the data).
- spam filtering, sentiment detection, rating classification etc.

Naïve Bayes

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

Steps Involved in Naive Bayes

Data Preparation:

- **Collect and clean your data:** Ensure the data is relevant and free from errors.
- **Split the data into training and testing sets:** This helps evaluate the model's performance.

Calculate Probabilities:

- **Calculate the prior probability:** This is the probability of each class occurring in the training set.
- **Calculate the conditional probability:** This is the probability of a feature occurring given a particular class.

Make Predictions:

- For a new instance, calculate the probability of it belonging to each class.
- Assign the instance to the class with the highest probability.

Naïve Bayes

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

Steps Involved in Naive Bayes

Data Preparation:

- **Collect and clean your data:** Ensure the data is relevant and free from errors.
- **Split the data into training and testing sets:** This helps evaluate the model's performance.

Calculate Probabilities:

- **Calculate the prior probability:** This is the probability of each class occurring in the training set.
- **Calculate the conditional probability:** This is the probability of a feature occurring given a particular class.

Make Predictions:

- For a new instance, calculate the probability of it belonging to each class.
- Assign the instance to the class with the highest probability.

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

NAIVE BAYES CLASSIFIER

Example - 1

$$P(PlayTennis = yes) = 9/14 = .64$$

$$P(PlayTennis = no) = 5/14 = .36$$

Outlook	Y	N
sunny	2/9	3/5
overcast	4/9	0
rain	3/9	2/5

Humidity	Y	N
high	3/9	4/5
normal	6/9	1/5

Tempreature		
hot	2/9	2/5
mild	4/9	2/5
cool	3/9	1/5

Windy		
Strong	3/9	3/5
Weak	6/9	2/5

$$\langle Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong \rangle$$

$$v_{NB} = \underset{v_j \in \{yes, no\}}{\text{argmax}} \; P(v_j) \prod_i \; P(a_i | v_j)$$

$$P(Outlook = sunny | v_j) P(Temperature = cool | v_j)$$

$$P(Humidity = high | v_j) P(Wind = strong | v_j)$$

$$v_{NB}(yes) = \; P(yes) \; P(sunny|yes) \; P(cool|yes) \; P(high|yes) \; P(strong|yes)$$

$$v_{NB}(no) = \; P(no) \; P(sunny|no) \; P(cool|no) \; P(high|no) \; P(strong|no)$$

Example No.	Color	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

$$\textit{New Instance} = (\textit{Red}, \textit{SUV}, \textit{Domestic})$$

Naïve Bayes

Advantage:

- **Less complex:** considered as simpler classifier since the parameters are easier to estimate.
- **Scales well**: fairly accurate when the conditional independence assumption holds. It also has low storage requirements.
- **Can handle high-dimensional data**: Use cases, such document classification, can have a high number of dimensions, which can be difficult for other classifiers to manage.
- **Overfitting** is generally not a major concern with Naive Bayes**.** This is primarily due to its strong independence assumption.

Disadvantage:

- Subject to **Zero frequency**: If categorical variable has a category (in test set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction.
- **Unrealistic core assumption**: In real life, it is almost impossible that we get a set of predictors which are completely independent.