**St. Francis Institute of Technology, Mumbai-400 103**

**Department Of Information Technology**

**A.Y. 2024-2025**

**Class: TE-ITA/B, Semester: VI**

**Subject: Business Intelligence Lab**

**Experiment – 5: a) To implement a classifier- Naïve Bayes using any one Language (JAVA/Python)**

**b) To implement a KNN classifier using any one Language (JAVA/Python) (Topic Beyond Syllabus)**

**1.** **Aim:** a) To implement a classifier- Naïve Bayes using any one Language (JAVA/Python)

b) To implement a KNN classifier using any one Language (JAVA/Python) (Topic Beyond Syllabus)

**2.**     **Objectives:** After study of this experiment, the students will be able to

                           implement Naïve based algorithm and Random forest/SVM algorithm

**3.**     **Outcomes:** After study of this experiment, the students will be able to

**CO 3:** Design and Implement various classification data mining techniques and apply metrics to measure its performance

**4.** **Prerequisite:** Introduction to all the classifiers through algorithms & Problem solving approach.

**5.** **Requirements:** Personal Computer, Windows XP operating system/Windows 7, Internet

Connection, Microsoft Word, WEKA tool, Java/R/Python

**6.** **Theory:**

a.Explain the Classification Algorithm (Naïve Bayes and KNN)

b. Applications of Classification Algorithms

c. Advantages and Disadvantages of Classification Algorithms

**7.**   **Laboratory Exercise:** Implementation of both (a&b) Classification Algorithm using JAVA/ R/ Python. Printout of implementation along with coding and Output.

**8.**   **Post-Experiments Exercise**

**a.** **Questions:**

**●** Compare and Contrast between Decision Tree & Naïve Bayes

* Compare and Contrast between Decision Tree and Random forest
* Solve a numerical on Naïve Bayes Algorithm

**b.**     **Conclusion:**

**●** Summary of Experiment

● Importance of Experiment

● Application of Experiment

**9.**     **Reference**: Data Mining: Concept & Techniques, 3rd Edition, Jiawei Han, Micheline Kamber, Jian Pei, Elsevier.

**Reference links:**

**·** [**https://scikit-learn.org/stable/modules/naive\_bayes.html**](https://scikit-learn.org/stable/modules/naive_bayes.html)

**·** [**https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn**](https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn)

**·** [**https://www.analyticsvidhya.com/blog/2021/11/implementation-of-gaussian-naive-bayes-in-python-sklearn/**](https://www.analyticsvidhya.com/blog/2021/11/implementation-of-gaussian-naive-bayes-in-python-sklearn/)

**·** [**https://github.com/2796gaurav/Naive-bayes-explained/blob/master/Naive%20bayes/Naive%20Bayes%20in%20scikit%20learn.ipynb**](https://github.com/2796gaurav/Naive-bayes-explained/blob/master/Naive%20bayes/Naive%20Bayes%20in%20scikit%20learn.ipynb)

**·** [**https://medium.com/analytics-vidhya/naive-bayes-classifier-a-beginners-guide-to-master-the-fastest-and-simplest-classification-d6a368e6b737**](https://medium.com/analytics-vidhya/naive-bayes-classifier-a-beginners-guide-to-master-the-fastest-and-simplest-classification-d6a368e6b737)

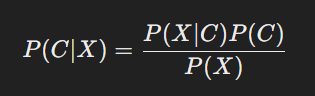
**·**

**Theory**

**Q.1 Explain the Classification Algorithm (Naïve Bayes and KNN)**

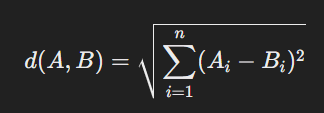
#### Naïve Bayes

Naïve Bayes is a probabilistic classification algorithm based on **Bayes’ Theorem**, assuming that features are independent (hence "naïve"). It calculates the probability of a class given certain features and selects the class with the highest probability.

* **Formula:**  
  Where:
  + P(C∣X)P(C|X)P(C∣X) is the probability of class CCC given features XXX.
  + P(X∣C)P(X|C)P(X∣C) is the likelihood of observing XXX in class CCC.
  + P(C)P(C)P(C) is the prior probability of class CCC.
  + P(X)P(X)P(X) is the probability of observing XXX.
* **Types of Naïve Bayes Classifiers:**
  + **Gaussian Naïve Bayes** (for continuous data).
  + **Multinomial Naïve Bayes** (for text classification).
  + **Bernoulli Naïve Bayes** (for binary features).
* **Use Case Example:** Spam email detection, sentiment analysis.

#### K-Nearest Neighbors (KNN)

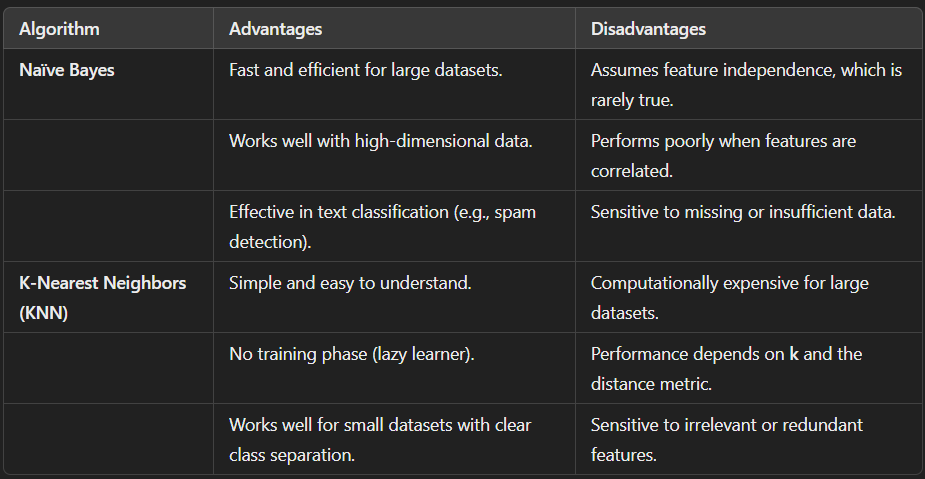
KNN is a **non-parametric** and **instance-based** learning algorithm that classifies a new data point based on the majority class of its **k-nearest neighbors** in the feature space.

* **How it Works:**
  + Choose a value for **k** (number of neighbors).
  + Measure the distance (e.g., **Euclidean distance**) between the new data point and existing points.
  + Identify the **k** nearest neighbors.
  + Assign the most common class among the neighbors to the new point.
* **Formula (Euclidean Distance):**
* **Use Case Example:** Recommendation systems, pattern recognition, medical diagnosis.

### Q.2 Applications of Classification Algorithms

1. **Spam Detection** – Naïve Bayes classifies emails as spam or not based on word probabilities.
2. **Sentiment Analysis** – Determines whether a review or comment is positive or negative.
3. **Medical Diagnosis** – KNN helps classify diseases based on patient symptoms.
4. **Image Recognition** – Identifies objects, faces, and handwriting patterns.
5. **Credit Risk Assessment** – Banks use classification models to determine loan approvals.
6. **Fraud Detection** – Detects fraudulent transactions in banking and e-commerce.
7. **Speech and Voice Recognition** – Used in virtual assistants like Siri and Google Assistant.

**Q.3 Advantages and Disadvantages of Classification Algorithms**



**Laboratory Exercise**

**Naiye bays**

import numpy as np

import pandas as pd

df = pd.read\_csv("weather\_data.csv")

play\_yes = len(df[df['PlayBall'] == 'Yes']) / len(df)

play\_no = len(df[df['PlayBall'] == 'No']) / len(df)

print(f"\nProbability of PlayBall='Yes': {play\_yes}")

print(f"Probability of PlayBall='No': {play\_no}")

outlook\_sunny\_yes = len(df[df['Outlook'] == 'Sunny']) / len(df[df['PlayBall'] == 'Yes'])

temperature\_cool\_yes = len(df[df['Temperature'] == 'Cool']) / len(df[df['PlayBall'] == 'Yes'])

humidity\_high\_yes = len(df[df['Humidity'] == 'High']) / len(df[df['PlayBall'] == 'Yes'])

wind\_strong\_yes = len(df[df['Wind'] == 'Strong']) / len(df[df['PlayBall'] == 'Yes'])

print("\n=============== Values of yes ===============")

print(f"outlook\_sunny\_yes : {outlook\_sunny\_yes}\ntemperature\_cool\_yes : {temperature\_cool\_yes }\nhumidity\_high\_yes : {humidity\_high\_yes}\nwind\_strong\_yes : {wind\_strong\_yes}")

outlook\_sunny\_no = len(df[df['Outlook'] == 'Sunny']) / len(df[df['PlayBall'] == 'No'])

temperature\_cool\_no = len(df[df['Temperature'] == 'Cool']) / len(df[df['PlayBall'] == 'No'])

humidity\_high\_no = len(df[df['Humidity'] == 'High']) / len(df[df['PlayBall'] == 'No'])

wind\_strong\_no = len(df[df['Wind'] == 'Strong']) / len(df[df['PlayBall'] == 'No'])

print("\n=============== Values of no ===============")

print(f"outlook\_sunny\_no : {outlook\_sunny\_no}\ntemperature\_cool\_no : {temperature\_cool\_no }\nhumidity\_high\_no : {humidity\_high\_no}\nwind\_strong\_no : {wind\_strong\_no}")

class\_yes = play\_yes \* ( outlook\_sunny\_yes \* temperature\_cool\_yes \* humidity\_high\_yes \* wind\_strong\_yes )

class\_no = play\_no \* ( outlook\_sunny\_no \* temperature\_cool\_no \* humidity\_high\_no \* wind\_strong\_no )

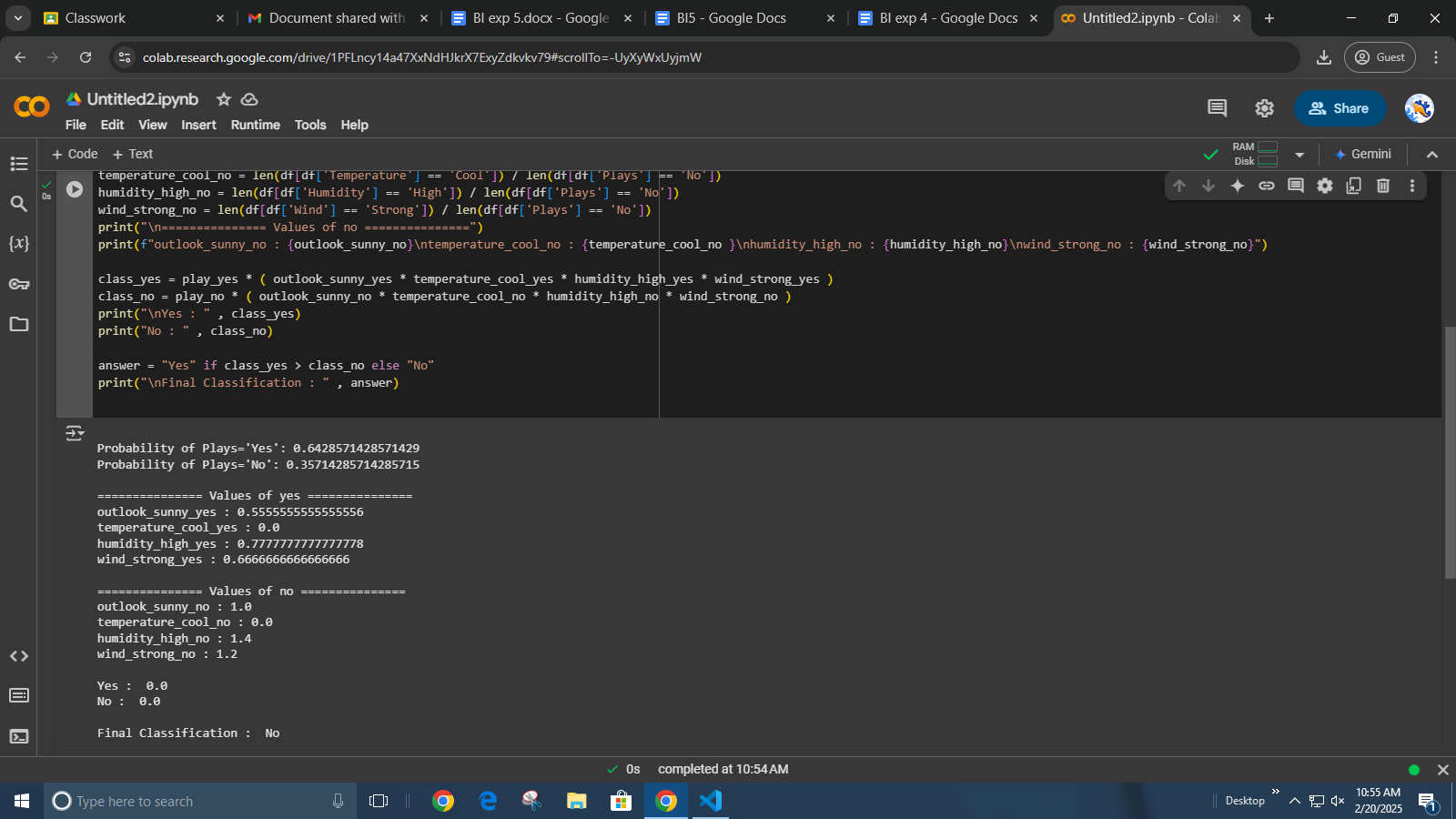
print("\nYes : " , class\_yes)

print("No : " , class\_no)

answer = "Yes" if class\_yes > class\_no else "No"

print("\nFinal Classification : " , answer)

Output:

****

**KNN:**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import accuracy\_score

data = pd.read\_csv("weather\_data.csv")

# Separate features (X) and target variable (y)

X = data.iloc[:, :-1].copy()

y = data.iloc[:, -1].copy()

# Convert categorical features to numerical values

label\_encoders = {}

for column in X.columns:

if X[column].dtype == 'object':

le = LabelEncoder()

X[column] = le.fit\_transform(X[column])

label\_encoders[column] = le

# Convert target variable if it's categorical

if y.dtype == 'object':

y\_le = LabelEncoder()

y = y\_le.fit\_transform(y)

# Split dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train KNN classifier

knn\_model = KNeighborsClassifier(n\_neighbors=3)

knn\_model.fit(X\_train, y\_train)

# Predict and evaluate

y\_knn\_pred = knn\_model.predict(X\_test)

accuracy\_knn = accuracy\_score(y\_test, y\_knn\_pred)

print("KNN Accuracy:", accuracy\_knn)

output:

