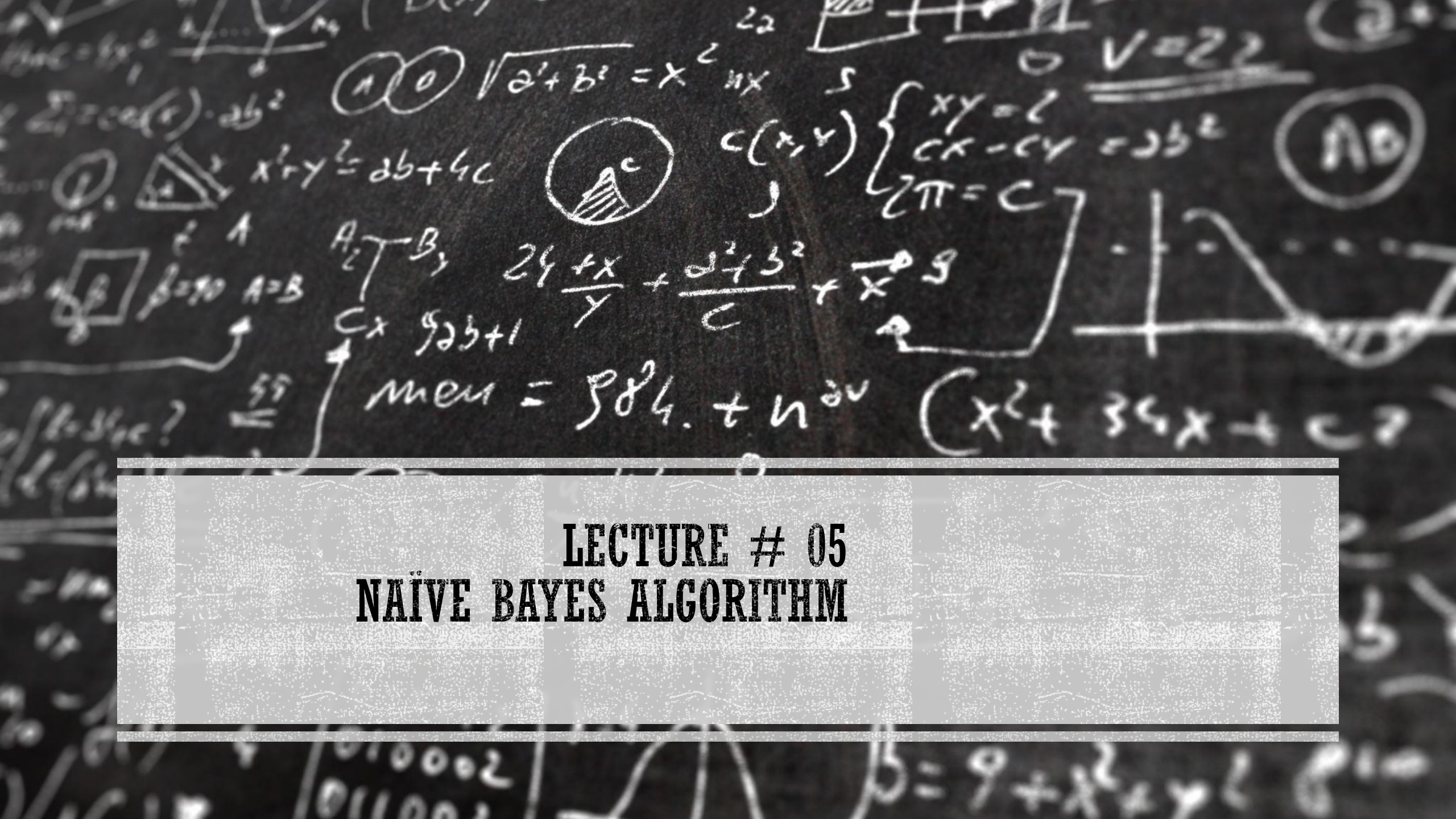
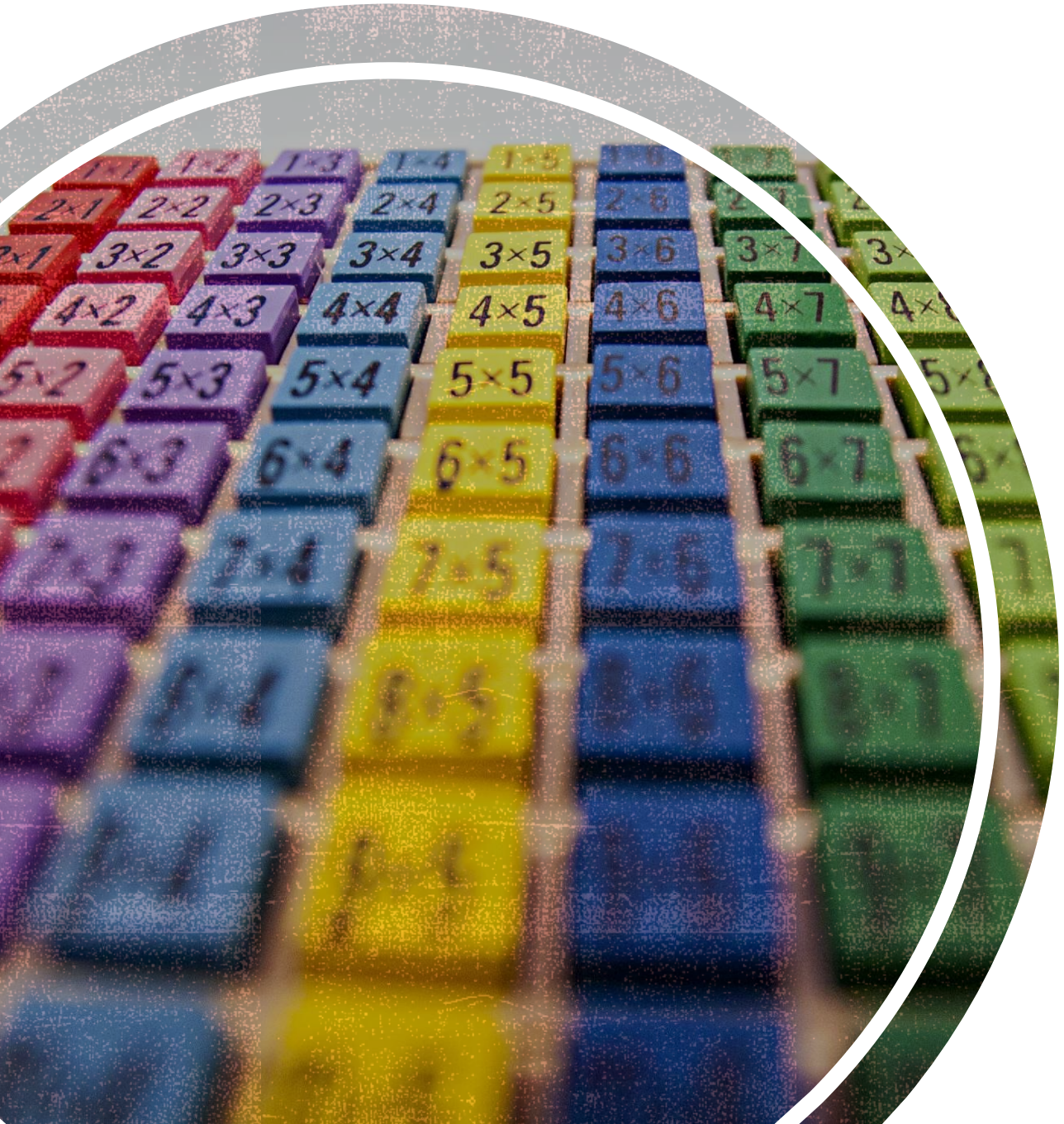


LECTURE # 05
NAÏVE BAYES ALGORITHM



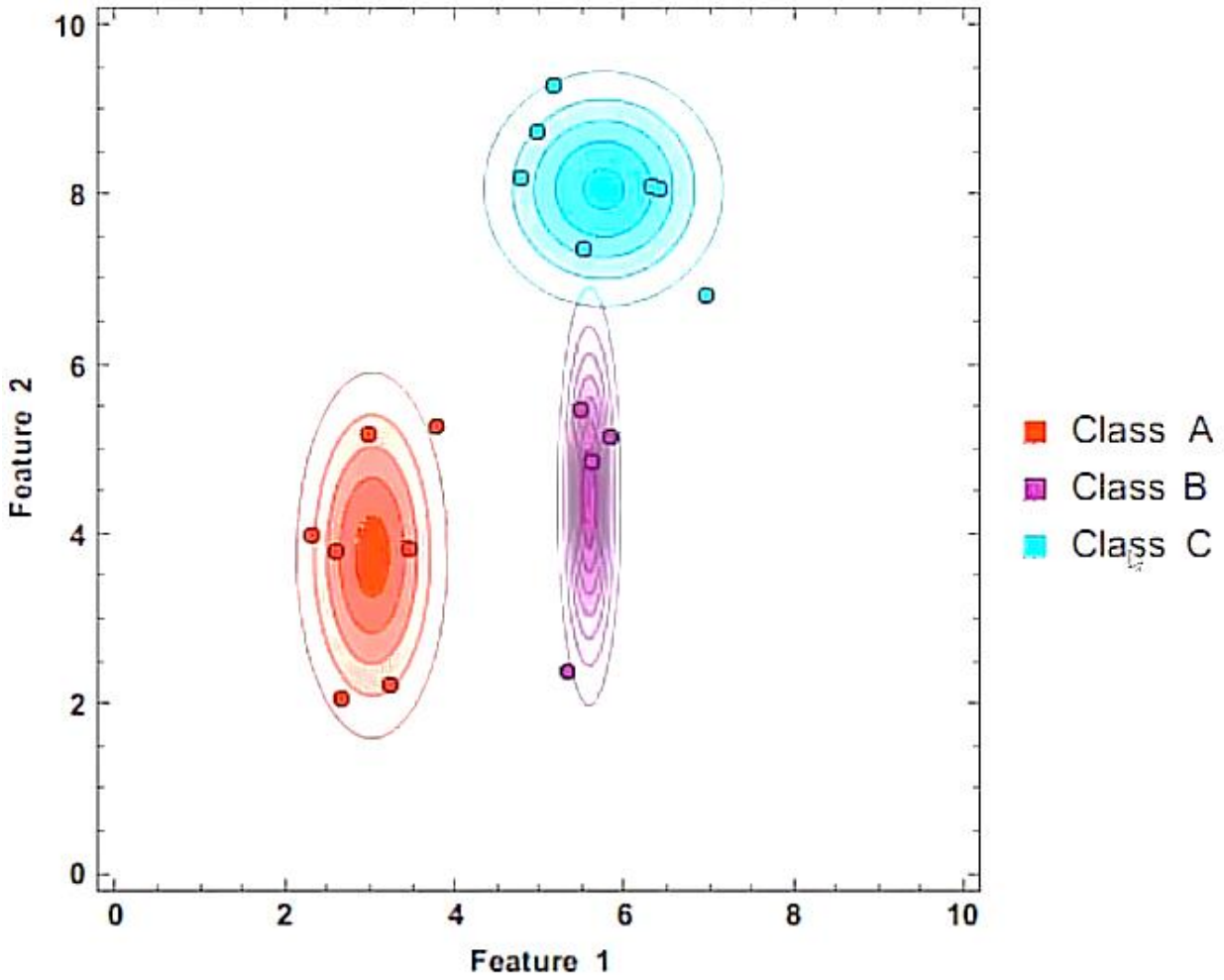
INTRODUCTION TO NAÏVE BAYES:

- **Introduction to Naive Bayes Algorithm:**
- Naive Bayes is a simple, yet powerful algorithm used for classification tasks in machine learning. It's based on Bayes' Theorem, named after the Reverend Thomas Bayes. At its core, Naive Bayes relies on probability theory to make predictions.

SCENARIO IN WHICH NAÏVE BAYES IS USED:

- Naive Bayes is primarily used for classification tasks in machine learning. It's particularly well-suited for classification because it calculates the probability of an instance belonging to each class based on its features, and then assigns the instance to the class with the highest probability. This makes Naive Bayes a powerful tool for tasks such as spam detection, sentiment analysis, document categorization, and many others where we need to classify data into different categories or classes.

VISUAL EXPLANATION:



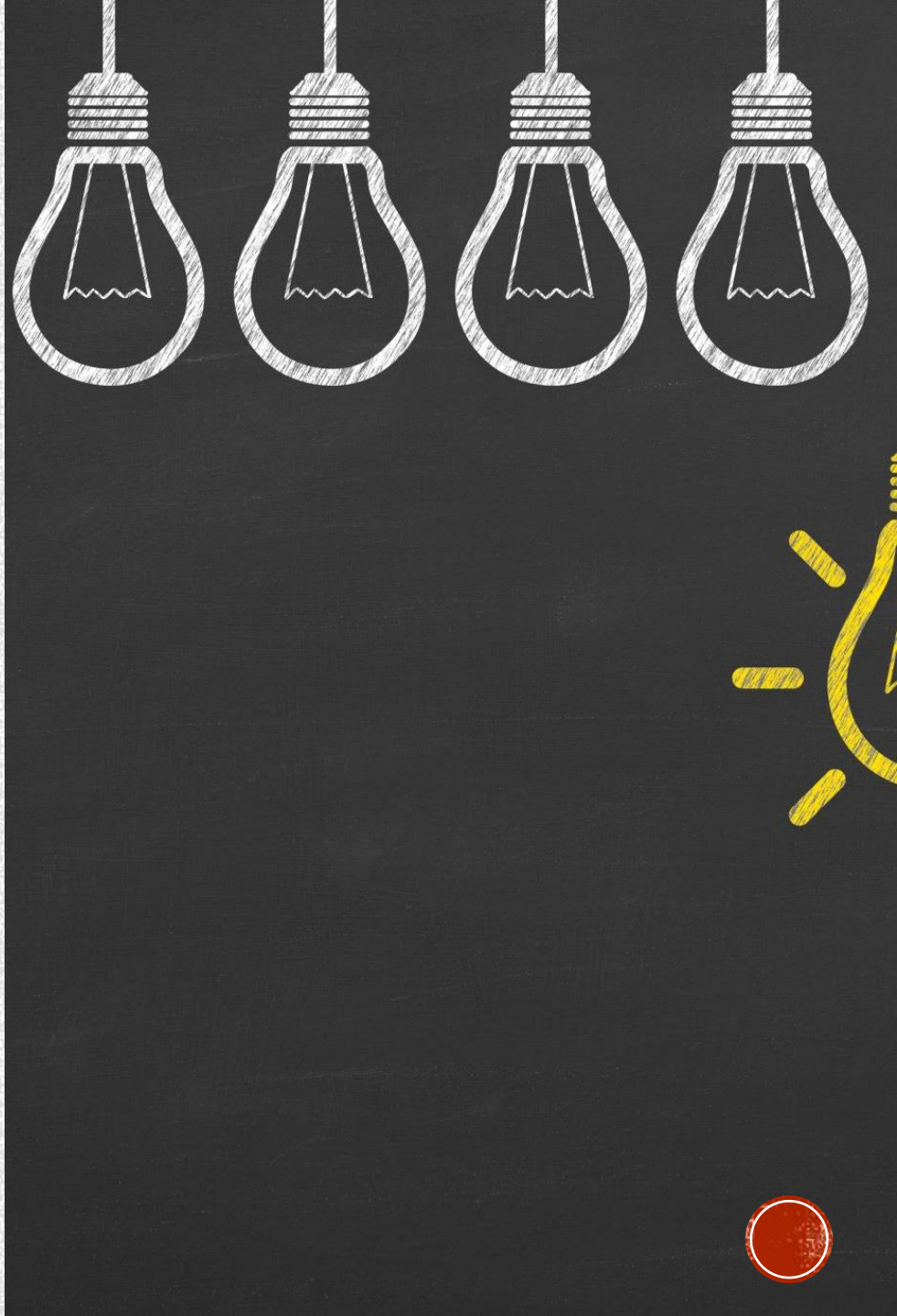
NAÏVE BAYES ASSUMPTION:

- **The Naive Assumption:**
- The "naive" in Naive Bayes comes from the assumption that features are independent of each other, given the class label. This simplifies the computation and makes the algorithm efficient, although it might not hold true in all real-world scenarios.

WORKING PRINCIPLE OF NAÏVE BAYES:

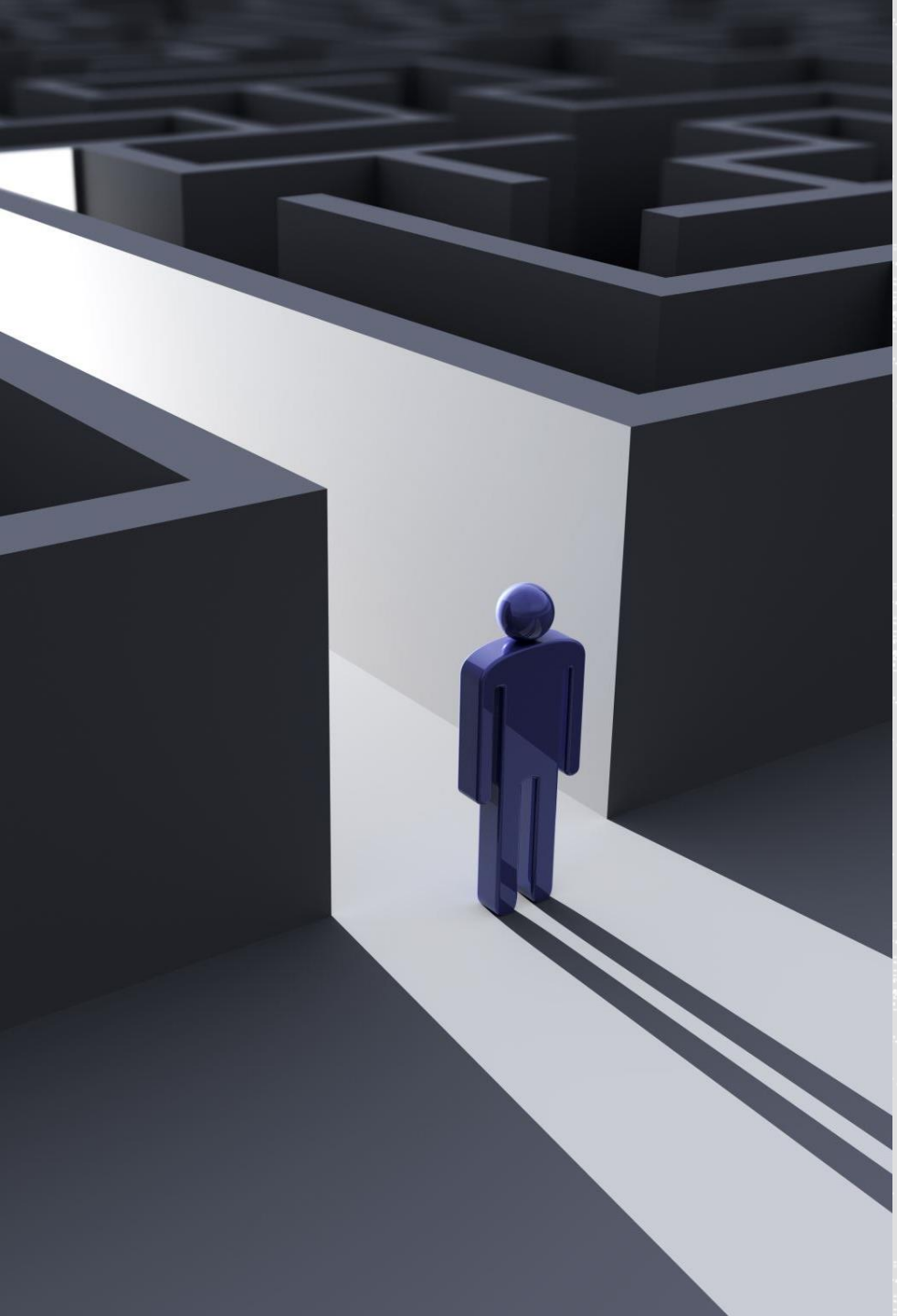
Working Principle:

1. **Training Phase:** Naive Bayes learns the probability distribution of the features for each class from the training data.
2. **Prediction Phase:** Given a new instance, Naive Bayes calculates the probability of it belonging to each class using Bayes' Theorem and selects the class with the highest probability.



ADVANTAGES:

- **Advantages of Naive Bayes:**
- **Simple and Fast:** Naive Bayes is easy to implement and computationally efficient.
- **Handles Large Feature Spaces:** It performs well even with a large number of features.
- **Works Well with Categorical Data:** It can handle both numerical and categorical data.



LIMITATIONS:

- **Limitations of Naive Bayes:**
- **Strong Independence Assumption:** The assumption of feature independence might not hold true in all cases.
- **Sensitive to Feature Correlations:** It may not perform well when features are highly correlated.

LIMITATIONS:

p(Dear | N) = 0.47

p(Friend | N) = 0.29

p(Lunch| N) = 0.18

p(Money| N) = 0.06

p(N) = 0.67

Lunch Money Money Money Money

α

p(S) = 0.33

p(Dear | S) = 0.29

p(Friend | S) = 0.14

p(Lunch| S) = 0

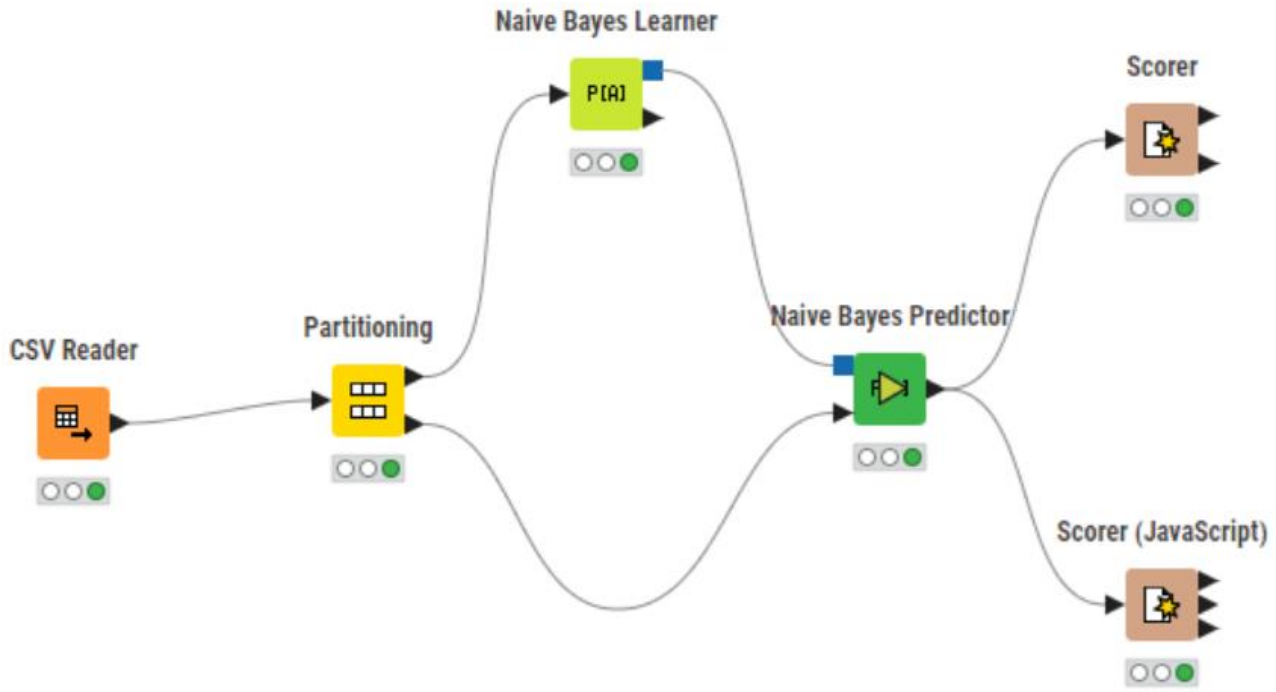
p(Money| S) = 0.57

REAL WORLD APPLICATIONS:

- **Real-World Applications:**
- Naive Bayes finds applications in various fields, including:
- **Email Spam Detection:** Classifying emails as spam or non-spam.
- **Document Classification:** Categorizing documents based on their content.
- **Sentiment Analysis:** Analyzing text to determine the sentiment expressed.

EMAIL SPAM CHECKING:

NAIVE BAYES IN KNIME:



NAÏVE BAYES CODE:

[3]
```
# Importing necessary libraries
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
```

[4]
```
df = pd.read_csv("/content/heart_failure_clinical_records_dataset.csv")
```

```
# Select the specified columns
selected_columns = ['age', 'anaemia', 'creatinine_phosphokinase', 'diabetes',
                    'ejection_fraction', 'high_blood_pressure', 'platelets',
                    'serum_creatinine', 'serum_sodium', 'sex', 'smoking', 'time',
                    'DEATH_EVENT']
df = df[selected_columns]

# Split the dataset into features (X) and target variable (y)
X = df.drop(columns=["DEATH_EVENT"])
y = df["DEATH_EVENT"]

# Split the 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)

# Initialize the Naive Bayes classifier
clf = GaussianNB()


# Train the classifier on the training data
clf.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = clf.predict(X_test)

# Evaluate the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7333333333333333
```

NAÏVE BAYES EXPLANATION:

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

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

NAÏVE BAYES ALGORITHM:

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

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

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

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

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

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

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

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

NAÏVE BAYES ALGORITHM:

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

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

$$= \underset{v_j \in \{yes,no\}}{\text{argmax}} \; P(v_j) \quad P(Outlook = sunny|v_j) P(Temperature = cool|v_j)$$

$$\cdot \; P(Humidity = high|v_j) P(Wind = strong|v_j)$$

$$v_{NB}(yes) = P(yes) \; P(sunny|yes) \; P(cool|yes) \; P(high|yes) \; P(strong|yes) = .0053$$

$$v_{NB}(no) = P(no) \; P(sunny|no) \; P(cool|no) \; P(high|no) \; P(strong|no) = .0206$$

$$v_{NB}(yes) = \frac{v_{NB}(yes)}{v_{NB}(yes)+v_{NB}(no)} = 0.205 \qquad v_{NB}(no) = \frac{v_{NB}(no)}{v_{NB}(yes)+v_{NB}(no)} = 0.795$$

As Probability for No is more than the Probability for Yes so,
According to Naïve Bayes the person will not play tennis given that

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

LET'S SOLVE THIS USING PYTHON (NAÏVE BAYES PYTHON IMPLEMENTATION):

```python
import pandas as pd
from collections import Counter

# Load the dataset
data = pd.read_csv('tennis dataset.csv')

# Split the data into two classes
yes_data = data[data['Play _Tennis'] == 'Yes']
no_data = data[data['Play _Tennis'] == 'No']

# Calculate the prior probabilities
total_instances = len(data)
prior_yes = len(yes_data) / total_instances
prior_no = len(no_data) / total_instances

# Calculate the likelihood probabilities
features = ['Outlook', 'Temperature', 'Humidity', 'Wind']

likelihood_yes = {}
likelihood_no = {}

for feature in features:
    likelihood_yes[feature] = Counter(yes_data[feature])
    likelihood_no[feature] = Counter(no_data[feature])

    total_yes = sum(likelihood_yes[feature].values())
    total_no = sum(likelihood_no[feature].values())

    for value in likelihood_yes[feature]:
        likelihood_yes[feature][value] /= total_yes

    for value in likelihood_no[feature]:
        likelihood_no[feature][value] /= total_no

# Define the evidence
evidence = {
    'Outlook': 'Sunny',
    'Temperature': 'Cool',
    'Humidity': 'High',
    'Wind': 'Strong'
}
```

```python
# Calculate the posterior probabilities
posterior_yes = prior_yes
posterior_no = prior_no

for feature, value in evidence.items():
    posterior_yes *= likelihood_yes[feature][value]
    posterior_no *= likelihood_no[feature][value]

# Make the prediction
if posterior_yes > posterior_no:
    prediction = 'Yes'
else:
    prediction = 'No'

print(f'Given the conditions: {evidence}')
print(f'The prediction is: {prediction}')
```

```
Given the conditions: {'Outlook': 'Sunny', 'Temperature': 'Cool', 'Humidity': 'High', 'Wind': 'Strong'}
The prediction is: No
```

TASKS:

Using python implement Naïve Bayes with two different splitting ratios on Heart Attack Analysis & prediction dataset to predict the chances of heart failure in a person and performed the following steps:

- Data Pre-processing step
- Fitting Naive Bayes to the Training set
- Predicting the test result
- Test accuracy of the result(Creation of Confusion matrix)
- Visualizing the test set result.
- Compare the accuracies

TASK 2:

- 2. Design a workflow with the help of Knime to predict whether a user buys a product by clicking the ad on the site based on their salary, age, and gender dataset provided in the lab (i.e. Social network ad dataset).