TITLE-01

GRADIENT BOOSTING VS NAÏVE BAYES

import pandas as pd

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

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.ensemble import GradientBoostingClassifier  # Added for Gradient Boosting

from sklearn.metrics import accuracy\_score

# Load your dataset

data = pd.read\_csv('/content/student-por.csv')

# Define a target variable (label)

data['label'] = (data['higher'] == 'yes').astype(int)

# Select relevant columns for text processing

text\_data = data[['school', 'sex', 'address', 'Mjob', 'Fjob', 'reason', 'guardian']]

# Concatenate selected columns into a single text column

data['text'] = text\_data.apply(lambda row: ' '.join(row), axis=1)

# Feature Extraction

tfidf\_vectorizer = TfidfVectorizer(max\_features=1000)

X = tfidf\_vectorizer.fit\_transform(data['text'])

y = data['label']

# Initialize lists to store accuracy values

naive\_bayes\_accuracies = []

gradient\_boosting\_accuracies = []

# Perform 10 iterations

for iteration in range(10):

    # Split Data

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

    # Train and Evaluate Naïve Bayes

    naive\_bayes = MultinomialNB()

    naive\_bayes.fit(X\_train, y\_train)

    y\_pred\_nb = naive\_bayes.predict(X\_test)

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

    naive\_bayes\_accuracies.append(accuracy\_nb)

    # Train and Evaluate Gradient Boosting

    gradient\_boosting = GradientBoostingClassifier(n\_estimators=100, random\_state=42)

    gradient\_boosting.fit(X\_train, y\_train)

    y\_pred\_gb = gradient\_boosting.predict(X\_test)

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

    gradient\_boosting\_accuracies.append(accuracy\_gb)

# Calculate and print the average accuracy values

avg\_naive\_bayes\_accuracy = sum(naive\_bayes\_accuracies) / len(naive\_bayes\_accuracies)

avg\_gradient\_boosting\_accuracy = sum(gradient\_boosting\_accuracies) / len(gradient\_boosting\_accuracies)

print(f"Average Naïve Bayes Accuracy: {avg\_naive\_bayes\_accuracy:.4f}")

print(f"Average Gradient Boosting Accuracy: {avg\_gradient\_boosting\_accuracy:.4f}")

OUTPUT:

Average Naïve Bayes Accuracy: 0.9154

Average Gradient Boosting Accuracy: 0.9092

TITLE-02

RANDOM FOREST VS NAÏVE BAYES

import pandas as pd

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

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.ensemble import RandomForestClassifier  # Added for Random Forest

from sklearn.metrics import classification\_report

# Step 1: Load your dataset

# Assuming you have a dataset named 'social\_media\_data.csv'

data = pd.read\_csv('/content/student-por.csv')

# Step 2: Define a target variable (label)

# Assuming 'interested\_in\_education' column represents if a user is interested in education ('yes' or 'no')

data['label'] = (data['higher'] == 'yes').astype(int)

# Step 3: Select relevant columns for text processing

text\_data = data[['school', 'sex', 'address', 'Mjob', 'Fjob', 'reason', 'guardian']]

# Concatenate selected columns into a single text column

data['text'] = text\_data.apply(lambda row: ' '.join(row), axis=1)

# Step 4: Feature Extraction

tfidf\_vectorizer = TfidfVectorizer(max\_features=1000)  # Adjust max\_features as needed

X = tfidf\_vectorizer.fit\_transform(data['text'])

y = data['label']

# Step 5: Split Data

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

# Step 6: Train and Evaluate Naïve Bayes

naive\_bayes = MultinomialNB()

naive\_bayes.fit(X\_train, y\_train)

y\_pred\_nb = naive\_bayes.predict(X\_test)

print("Naïve Bayes Performance:")

print(classification\_report(y\_test, y\_pred\_nb))

# Step 7: Train and Evaluate Random Forest

random\_forest = RandomForestClassifier(n\_estimators=100, random\_state=42)  # You can adjust parameters as needed

random\_forest.fit(X\_train, y\_train)

y\_pred\_rf = random\_forest.predict(X\_test)

print("Random Forest Performance:")

print(classification\_report(y\_test, y\_pred\_rf))

OUTPUT:

Naïve Bayes Performance:

precision recall f1-score support

0 0.00 0.00 0.00 12

1 0.91 1.00 0.95 118

accuracy 0.91 130

macro avg 0.45 0.50 0.48 130

weighted avg 0.82 0.91 0.86 130

Random Forest Performance:

precision recall f1-score support

0 1.00 0.08 0.15 12

1 0.91 1.00 0.96 118

accuracy 0.92 130

macro avg 0.96 0.54 0.55 130

weighted avg 0.92 0.92 0.88 130

**TITLE-03**

**DECISION TREE VS NAÏVE BAYES**

import pandas as pd

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

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification\_report

# Step 1: Load your dataset

# Replace 'your\_data.csv' with your actual data file path

data = pd.read\_csv('/content/student-por.csv')

# Step 2: Define a target variable (label)

# For example, let's say we want to predict if a student is interested in education based on their features.

# We'll assume a student is interested in education if their 'higher' column is 'yes' (want to pursue higher education).

data['label'] = (data['higher'] == 'yes').astype(int)

# Step 3: Select relevant columns for text processing

text\_data = data[['school', 'sex', 'address', 'Mjob', 'Fjob', 'reason', 'guardian']]

# Concatenate selected columns into a single text column

data['text'] = text\_data.apply(lambda row: ' '.join(row), axis=1)

# Step 4: Feature Extraction

tfidf\_vectorizer = TfidfVectorizer(max\_features=1000)  # Adjust max\_features as needed

X = tfidf\_vectorizer.fit\_transform(data['text'])

y = data['label']

# Step 5: Split Data

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

# Step 6: Train and Evaluate Naïve Bayes

naive\_bayes = MultinomialNB()

naive\_bayes.fit(X\_train, y\_train)

y\_pred\_nb = naive\_bayes.predict(X\_test)

print("Naïve Bayes Performance:")

print(classification\_report(y\_test, y\_pred\_nb))

# Step 7: Train and Evaluate Decision Tree

decision\_tree = DecisionTreeClassifier()

decision\_tree.fit(X\_train, y\_train)

y\_pred\_dt = decision\_tree.predict(X\_test)

print("Decision Tree Performance:")

print(classification\_report(y\_test, y\_pred\_dt))

**output:**

Naïve Bayes Performance:

precision recall f1-score support

0 0.00 0.00 0.00 12

1 0.91 1.00 0.95 118

accuracy 0.91 130

macro avg 0.45 0.50 0.48 130

weighted avg 0.82 0.91 0.86 130

Decision Tree Performance:

precision recall f1-score support

0 0.40 0.17 0.24 12

1 0.92 0.97 0.95 118

accuracy 0.90 130

macro avg 0.66 0.57 0.59 130

weighted avg 0.87 0.90 0.88 130

**TITLE-04**

**K-NN VS NAÏVE BAYES**

import pandas as pd

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

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import classification\_report

# Step 1: Load your dataset

# Replace 'your\_data.csv' with your actual data file path

data = pd.read\_csv('/content/student-por.csv')

# Step 2: Define a target variable (label)

# We'll assume a student is interested in education if their 'higher' column is 'yes'.

data['label'] = (data['higher'] == 'yes').astype(int)

# Step 3: Select relevant columns for text processing

text\_data = data[['school', 'sex', 'address', 'Mjob', 'Fjob', 'reason', 'guardian']]

# Concatenate selected columns into a single text column

data['text'] = text\_data.apply(lambda row: ' '.join(row), axis=1)

# Step 4: Feature Extraction

tfidf\_vectorizer = TfidfVectorizer(max\_features=1000)  # Adjust max\_features as needed

X = tfidf\_vectorizer.fit\_transform(data['text'])

y = data['label']

# Step 5: Split Data

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

# Step 6: Train and Evaluate Naïve Bayes

naive\_bayes = MultinomialNB()

naive\_bayes.fit(X\_train, y\_train)

y\_pred\_nb = naive\_bayes.predict(X\_test)

print("Naïve Bayes Performance:")

print(classification\_report(y\_test, y\_pred\_nb))

# Step 7: Train and Evaluate K-Nearest Neighbors

knn = KNeighborsClassifier(n\_neighbors=5)  # You can adjust the number of neighbors as needed

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

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

print("K-NN Performance:")

print(classification\_report(y\_test, y\_pred\_knn))

**output:**

Naïve Bayes Performance:

precision recall f1-score support

0 0.00 0.00 0.00 12

1 0.91 1.00 0.95 118

accuracy 0.91 130

macro avg 0.45 0.50 0.48 130

weighted avg 0.82 0.91 0.86 130

K-NN Performance:

precision recall f1-score support

0 0.00 0.00 0.00 12

1 0.91 0.99 0.95 118

accuracy 0.90 130

macro avg 0.45 0.50 0.47 130

weighted avg 0.82 0.90 0.86 130

WITH 10 ACCURACY VALUES:

TITLE-01

GRADIENT BOOSTING VS NAÏVE BAYES

import pandas as pd

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

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.metrics import accuracy\_score

# Load your dataset

data = pd.read\_csv('/content/student-por.csv')

# Define a target variable (label)

data['label'] = (data['higher'] == 'yes').astype(int)

# Select relevant columns for text processing

text\_data = data[['school', 'sex', 'address', 'Mjob', 'Fjob', 'reason', 'guardian']]

# Concatenate selected columns into a single text column

data['text'] = text\_data.apply(lambda row: ' '.join(row), axis=1)

# Feature Extraction

tfidf\_vectorizer = TfidfVectorizer(max\_features=1000)

X = tfidf\_vectorizer.fit\_transform(data['text'])

y = data['label']

# Initialize variables to store cumulative accuracy

total\_accuracy\_nb = 0

total\_accuracy\_gb = 0

# Perform 10 iterations

for iteration in range(10):

print(f"Iteration {iteration + 1}")

# Split Data

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

# Train and Evaluate Naïve Bayes

naive\_bayes = MultinomialNB()

naive\_bayes.fit(X\_train, y\_train)

y\_pred\_nb = naive\_bayes.predict(X\_test)

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

total\_accuracy\_nb += accuracy\_nb

print(f"Naïve Bayes Accuracy: {accuracy\_nb:.4f}")

# Train and Evaluate Gradient Boosting

gradient\_boosting = GradientBoostingClassifier(n\_estimators=100, random\_state=42)

gradient\_boosting.fit(X\_train, y\_train)

y\_pred\_gb = gradient\_boosting.predict(X\_test)

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

total\_accuracy\_gb += accuracy\_gb

print(f"Gradient Boosting Accuracy: {accuracy\_gb:.4f}")

print("=" \* 50) # Add a separator for clarity

# Calculate average accuracy

avg\_accuracy\_nb = total\_accuracy\_nb / 10

avg\_accuracy\_gb = total\_accuracy\_gb / 10

print(f"\nAverage Naïve Bayes Accuracy: {avg\_accuracy\_nb:.4f}")

print(f"Average Gradient Boosting Accuracy: {avg\_accuracy\_gb:.4f}")

output:

Iteration 1

Naïve Bayes Accuracy: 0.9231

Gradient Boosting Accuracy: 0.9308

==================================================

Iteration 2

Naïve Bayes Accuracy: 0.9462

Gradient Boosting Accuracy: 0.9231

==================================================

Iteration 3

Naïve Bayes Accuracy: 0.9077

Gradient Boosting Accuracy: 0.8769

==================================================

Iteration 4

Naïve Bayes Accuracy: 0.9231

Gradient Boosting Accuracy: 0.9154

==================================================

Iteration 5

Naïve Bayes Accuracy: 0.8923

Gradient Boosting Accuracy: 0.8846

==================================================

Iteration 6

Naïve Bayes Accuracy: 0.8769

Gradient Boosting Accuracy: 0.8923

==================================================

Iteration 7

Naïve Bayes Accuracy: 0.9462

Gradient Boosting Accuracy: 0.9385

==================================================

Iteration 8

Naïve Bayes Accuracy: 0.9462

Gradient Boosting Accuracy: 0.9462

==================================================

Iteration 9

Naïve Bayes Accuracy: 0.8692

Gradient Boosting Accuracy: 0.8615

==================================================

Iteration 10

Naïve Bayes Accuracy: 0.9231

Gradient Boosting Accuracy: 0.9231

==================================================

Average Naïve Bayes Accuracy: 0.9154

Average Gradient Boosting Accuracy: 0.9092

TITLE-02

RANDOM FOREST VS NAÏVE BAYES

import pandas as pd

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

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

# Load your dataset

data = pd.read\_csv('/content/student-por.csv')

# Define a target variable (label)

data['label'] = (data['higher'] == 'yes').astype(int)

# Select relevant columns for text processing

text\_data = data[['school', 'sex', 'address', 'Mjob', 'Fjob', 'reason', 'guardian']]

# Concatenate selected columns into a single text column

data['text'] = text\_data.apply(lambda row: ' '.join(row), axis=1)

# Feature Extraction

tfidf\_vectorizer = TfidfVectorizer(max\_features=1000)

X = tfidf\_vectorizer.fit\_transform(data['text'])

y = data['label']

# Initialize lists to store accuracy values

naive\_bayes\_accuracies = []

random\_forest\_accuracies = []

# Perform 10 iterations

for iteration in range(10):

    print(f"Iteration {iteration + 1}")

    # Split Data

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

    # Train and Evaluate Naive Bayes

    naive\_bayes = MultinomialNB()

    naive\_bayes.fit(X\_train, y\_train)

    y\_pred\_nb = naive\_bayes.predict(X\_test)

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

    naive\_bayes\_accuracies.append(accuracy\_nb)

    print(f"Naive Bayes Accuracy: {accuracy\_nb:.4f}")

    # Train and Evaluate Random Forest

    random\_forest = RandomForestClassifier(n\_estimators=100, random\_state=42)

    random\_forest.fit(X\_train, y\_train)

    y\_pred\_rf = random\_forest.predict(X\_test)

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

    random\_forest\_accuracies.append(accuracy\_rf)

    print(f"Random Forest Accuracy: {accuracy\_rf:.4f}")

    print("=" \* 50)  # Add a separator for clarity

# Calculate and print the average accuracy values

avg\_naive\_bayes\_accuracy = sum(naive\_bayes\_accuracies) / len(naive\_bayes\_accuracies)

avg\_random\_forest\_accuracy = sum(random\_forest\_accuracies) / len(random\_forest\_accuracies)

print(f"Average Naive Bayes Accuracy: {avg\_naive\_bayes\_accuracy:.4f}")

print(f"Average Random Forest Accuracy: {avg\_random\_forest\_accuracy:.4f}")

OUTPUT:

Iteration 1

Naive Bayes Accuracy: 0.9231

Random Forest Accuracy: 0.9231

==================================================

Iteration 2

Naive Bayes Accuracy: 0.9462

Random Forest Accuracy: 0.8923

==================================================

Iteration 3

Naive Bayes Accuracy: 0.9077

Random Forest Accuracy: 0.8846

==================================================

Iteration 4

Naive Bayes Accuracy: 0.9231

Random Forest Accuracy: 0.9077

==================================================

Iteration 5

Naive Bayes Accuracy: 0.8923

Random Forest Accuracy: 0.8923

==================================================

Iteration 6

Naive Bayes Accuracy: 0.8769

Random Forest Accuracy: 0.8923

==================================================

Iteration 7

Naive Bayes Accuracy: 0.9462

Random Forest Accuracy: 0.9462

==================================================

Iteration 8

Naive Bayes Accuracy: 0.9462

Random Forest Accuracy: 0.9308

==================================================

Iteration 9

Naive Bayes Accuracy: 0.8692

Random Forest Accuracy: 0.8692

==================================================

Iteration 10

Naive Bayes Accuracy: 0.9231

Random Forest Accuracy: 0.9231

==================================================

Average Naive Bayes Accuracy: 0.9154

Average Random Forest Accuracy: 0.9062

**TITLE-03**

**DECISION TREE VS NAÏVE BAYES**

import pandas as pd

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

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

# Load your dataset

data = pd.read\_csv('/content/student-por.csv')

# Define a target variable (label)

data['label'] = (data['higher'] == 'yes').astype(int)

# Select relevant columns for text processing

text\_data = data[['school', 'sex', 'address', 'Mjob', 'Fjob', 'reason', 'guardian']]

# Concatenate selected columns into a single text column

data['text'] = text\_data.apply(lambda row: ' '.join(row), axis=1)

# Feature Extraction

tfidf\_vectorizer = TfidfVectorizer(max\_features=1000)

X = tfidf\_vectorizer.fit\_transform(data['text'])

y = data['label']

# Initialize lists to store accuracy values

naive\_bayes\_accuracies = []

decision\_tree\_accuracies = []

# Perform 10 iterations

for iteration in range(10):

    print(f"Iteration {iteration + 1}")

    # Split Data

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

    # Train and Evaluate Naive Bayes

    naive\_bayes = MultinomialNB()

    naive\_bayes.fit(X\_train, y\_train)

    y\_pred\_nb = naive\_bayes.predict(X\_test)

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

    naive\_bayes\_accuracies.append(accuracy\_nb)

    print(f"Naive Bayes Accuracy: {accuracy\_nb:.4f}")

    # Train and Evaluate Decision Tree

    decision\_tree = DecisionTreeClassifier(random\_state=42)

    decision\_tree.fit(X\_train, y\_train)

    y\_pred\_dt = decision\_tree.predict(X\_test)

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

    decision\_tree\_accuracies.append(accuracy\_dt)

    print(f"Decision Tree Accuracy: {accuracy\_dt:.4f}")

    print("=" \* 50)  # Add a separator for clarity

# Calculate and print the average accuracy values

avg\_naive\_bayes\_accuracy = sum(naive\_bayes\_accuracies) / len(naive\_bayes\_accuracies)

avg\_decision\_tree\_accuracy = sum(decision\_tree\_accuracies) / len(decision\_tree\_accuracies)

print(f"Average Naive Bayes Accuracy: {avg\_naive\_bayes\_accuracy:.4f}")

print(f"Average Decision Tree Accuracy: {avg\_decision\_tree\_accuracy:.4f}")

OUTPUT:

Iteration 1

Naive Bayes Accuracy: 0.9231

Decision Tree Accuracy: 0.9308

==================================================

Iteration 2

Naive Bayes Accuracy: 0.9462

Decision Tree Accuracy: 0.8846

==================================================

Iteration 3

Naive Bayes Accuracy: 0.9077

Decision Tree Accuracy: 0.8769

==================================================

Iteration 4

Naive Bayes Accuracy: 0.9231

Decision Tree Accuracy: 0.9077

==================================================

Iteration 5

Naive Bayes Accuracy: 0.8923

Decision Tree Accuracy: 0.8769

==================================================

Iteration 6

Naive Bayes Accuracy: 0.8769

Decision Tree Accuracy: 0.8923

==================================================

Iteration 7

Naive Bayes Accuracy: 0.9462

Decision Tree Accuracy: 0.9231

==================================================

Iteration 8

Naive Bayes Accuracy: 0.9462

Decision Tree Accuracy: 0.9077

==================================================

Iteration 9

Naive Bayes Accuracy: 0.8692

Decision Tree Accuracy: 0.8692

==================================================

Iteration 10

Naive Bayes Accuracy: 0.9231

Decision Tree Accuracy: 0.8923

==================================================

Average Naive Bayes Accuracy: 0.9154

Average Decision Tree Accuracy: 0.8992

**TITLE-04**

**K-NN VS NAÏVE BAYES**

import pandas as pd

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

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.neighbors import KNeighborsClassifier  # Added for KNN

from sklearn.metrics import accuracy\_score

# Load your dataset

data = pd.read\_csv('/content/student-por.csv')

# Define a target variable (label)

data['label'] = (data['higher'] == 'yes').astype(int)

# Select relevant columns for text processing

text\_data = data[['school', 'sex', 'address', 'Mjob', 'Fjob', 'reason', 'guardian']]

# Concatenate selected columns into a single text column

data['text'] = text\_data.apply(lambda row: ' '.join(row), axis=1)

# Feature Extraction

tfidf\_vectorizer = TfidfVectorizer(max\_features=1000)

X = tfidf\_vectorizer.fit\_transform(data['text'])

y = data['label']

# Initialize lists to store accuracy values

naive\_bayes\_accuracies = []

knn\_accuracies = []

# Perform 10 iterations

for iteration in range(10):

    print(f"Iteration {iteration + 1}")

    # Split Data

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

    # Train and Evaluate Naive Bayes

    naive\_bayes = MultinomialNB()

    naive\_bayes.fit(X\_train, y\_train)

    y\_pred\_nb = naive\_bayes.predict(X\_test)

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

    naive\_bayes\_accuracies.append(accuracy\_nb)

    print(f"Naive Bayes Accuracy: {accuracy\_nb:.4f}")

    # Train and Evaluate K-Nearest Neighbors (KNN)

    knn = KNeighborsClassifier(n\_neighbors=5)  # You can change the number of neighbors

    knn.fit(X\_train, y\_train)

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

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

    knn\_accuracies.append(accuracy\_knn)

    print(f"KNN Accuracy: {accuracy\_knn:.4f}")

    print("=" \* 50)  # Add a separator for clarity

# Calculate and print the average accuracy values

avg\_naive\_bayes\_accuracy = sum(naive\_bayes\_accuracies) / len(naive\_bayes\_accuracies)

avg\_knn\_accuracy = sum(knn\_accuracies) / len(knn\_accuracies)

print(f"Average Naive Bayes Accuracy: {avg\_naive\_bayes\_accuracy:.4f}")

print(f"Average KNN Accuracy: {avg\_knn\_accuracy:.4f}")

OUTPUT:

Iteration 1

Naive Bayes Accuracy: 0.9231

KNN Accuracy: 0.9154

==================================================

Iteration 2

Naive Bayes Accuracy: 0.9462

KNN Accuracy: 0.9077

==================================================

Iteration 3

Naive Bayes Accuracy: 0.9077

KNN Accuracy: 0.9000

==================================================

Iteration 4

Naive Bayes Accuracy: 0.9231

KNN Accuracy: 0.9154

==================================================

Iteration 5

Naive Bayes Accuracy: 0.8923

KNN Accuracy: 0.8923

==================================================

Iteration 6

Naive Bayes Accuracy: 0.8769

KNN Accuracy: 0.8769

==================================================

Iteration 7

Naive Bayes Accuracy: 0.9462

KNN Accuracy: 0.9308

==================================================

Iteration 8

Naive Bayes Accuracy: 0.9462

KNN Accuracy: 0.9308

==================================================

Iteration 9

Naive Bayes Accuracy: 0.8692

KNN Accuracy: 0.8231

==================================================

Iteration 10

Naive Bayes Accuracy: 0.9231

KNN Accuracy: 0.9154

==================================================

Average Naive Bayes Accuracy: 0.9154

Average KNN Accuracy: 0.9008