A red and black logo

Description automatically generatedA purple and white logo

Description automatically generated

**Bsc. (Hons) Artificial Intelligene & Data Science**

**Academic Year :** 2023/2024

**Semester :** 2

**Module Number :** CM2604

**Module Title :** Machine Learning

**Assessment Method Coursework :** Individual

**Deadline (time and date) :** 29th March 23.59 hr

**Submission Assessment :** Dropbox in the Module Study Area in CampusMoodle.

**Word Limit :** N/A

**Module Co-ordinator :** Sahan Priyanayana

**Student IIT ID :** 2021111

**Student RGU ID :** 2237949

**Student Name :** Manith Ratnayake

# Contents

[1. Contents 2](#_Toc162435155)

[List of Tables 2](#_Toc162435156)

[List of Figures 2](#_Toc162435157)

[Abstarct 3](#_Toc162435158)

[1. Corpus Preparation 3](#_Toc162435159)

[2. Implementation 3](#_Toc162435160)

[Naïve Bayes 3](#_Toc162435161)

[Random Forest 3](#_Toc162435162)

[2. Experiments 7](#_Toc162435163)

[3. Discussions 7](#_Toc162435164)

[4. Limitations 8](#_Toc162435165)

[5. Further Enchancements 8](#_Toc162435166)

# List of Tables

# List of Figures

# Abstarct

# Corpus Preparation

# Implementation

## Naïve Bayes

## Random Forest

from google.colab import drive

drive.mount('/content/drive')

file\_paths = [

    '/content/drive/My Drive/Machine Learning/old.adult.names',

    '/content/drive/My Drive/Machine Learning/Index',

    '/content/drive/My Drive/Machine Learning/adult.test',

    '/content/drive/My Drive/Machine Learning/adult.names',

    '/content/drive/My Drive/Machine Learning/adult.data'

]

for file\_path in file\_paths:

    with open(file\_path, 'r') as file:

        content = file.read()

        print(f"Content of {file\_path}: successful")

import pandas as pd

# Replace 'adult.data' with the actual path to your file if it's not in the working directory

file\_path = '/content/drive/My Drive/Machine Learning/adult.data'

file\_data = pd.read\_csv(file\_path, delimiter=',', header=None)  # Adjust delimiter as necessary

# Since the file may not contain headers, you might need to manually specify them

column\_names = ['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status',

                'occupation', 'relationship', 'race', 'sex', 'capital-gain', 'capital-loss',

                'hours-per-week', 'native-country', 'income']

file\_data.columns = column\_names

# Checking the first few rows

print(file\_data.head())

# Summarize statistics for numerical features

print(file\_data.describe())

# Distribution of the target variable

print(file\_data['income'].value\_counts())

# Replace '?' with NaN and then check for missing values

import numpy as np

file\_data = file\_data.replace('?', np.nan)

# Checking for missing values

print(file\_data.isnull().sum())

from sklearn.preprocessing import OneHotEncoder

# Assuming 'file\_data' is your DataFrame and 'income' is the target variable

categorical\_cols = file\_data.columns[file\_data.dtypes==object].tolist()

categorical\_cols.remove('income')

# Apply one-hot encoding with the updated parameter

# Use 'sparse\_output=False' if you're using scikit-learn 1.2 or newer.

# This will ensure your code is forward-compatible.

one\_hot\_encoder = OneHotEncoder(sparse\_output=False)

categorical\_data\_encoded = one\_hot\_encoder.fit\_transform(file\_data[categorical\_cols])

# Convert to DataFrame and add column names

encoded\_columns = one\_hot\_encoder.get\_feature\_names\_out(categorical\_cols)

categorical\_data\_encoded\_df = pd.DataFrame(categorical\_data\_encoded, columns=encoded\_columns)

# Drop original categorical columns and concat encoded columns

file\_data = file\_data.drop(categorical\_cols, axis=1).reset\_index(drop=True)

file\_data\_encoded = pd.concat([file\_data, categorical\_data\_encoded\_df], axis=1)

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

X = file\_data\_encoded.drop('income', axis=1)

y = file\_data\_encoded['income'].apply(lambda x: 1 if x == '>50K' else 0)  # Encoding the target variable

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

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, classification\_report

naive\_base\_model = GaussianNB()

naive\_base\_model.fit(X\_train, y\_train)

naive\_base\_predictions = naive\_base\_model.predict(X\_test)

print(f'Naïve Bayes Accuracy: {accuracy\_score(y\_test, naive\_base\_predictions)}')

print(classification\_report(y\_test, naive\_base\_predictions))

from sklearn.ensemble import RandomForestClassifier

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

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

random\_forest\_model\_predictions = random\_forest\_model.predict(X\_test)

print(f'Random Forest Accuracy: {accuracy\_score(y\_test, random\_forest\_model\_predictions)}')

print(classification\_report(y\_test, random\_forest\_model\_predictions))

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score

# Evaluation for Naïve Bayes

naive\_bayes\_accuracy = accuracy\_score(y\_test, naive\_base\_predictions)

naive\_bayes\_precision = precision\_score(y\_test, naive\_base\_predictions)

naive\_bayes\_recall = recall\_score(y\_test, naive\_base\_predictions)

naive\_bayes\_f1 = f1\_score(y\_test, naive\_base\_predictions)

# naive\_bayes\_auc = roc\_auc\_score(y\_test, naive\_base\_predictions)

print("Naïve Bayes model Performance \n")

print(f"Accuracy: {naive\_bayes\_accuracy:.4f}")

print(f"Precision: {naive\_bayes\_precision:.4f}")

print(f"Recall: {naive\_bayes\_recall:.4f}")

print(f"F1 Score: {naive\_bayes\_f1:.4f}")

# print(f"AUC: {naive\_bayes\_auc:.4f}\n")

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score

# Evaluation for Random Forest

random\_forest\_accuracy = accuracy\_score(y\_test, random\_forest\_model\_predictions)

random\_forest\_precision = precision\_score(y\_test, random\_forest\_model\_predictions)

random\_forest\_recall = recall\_score(y\_test, random\_forest\_model\_predictions)

random\_forest\_f1 = f1\_score(y\_test, random\_forest\_model\_predictions)

# random\_forest\_auc = roc\_auc\_score(y\_test, random\_forest\_model\_predictions)

print("Random Forest model Performance \n")

print(f"Accuracy: {random\_forest\_accuracy:.4f}")

print(f"Precision: {random\_forest\_precision:.4f}")

print(f"Recall: {random\_forest\_recall:.4f}")

print(f"F1 Score: {random\_forest\_f1:.4f}")

# print(f"AUC: {random\_forest\_auc:.4f}")

# Experiments

# Discussions

# Limitations

# Further Enchancements