**Mini Project Report :Defaulter Prediction**

**BATCH : B1**

**Student Name:**

**Prajakta Jadhav-UCE2022516**

**Synopsis :**

**1)Title:** Attendance-Based Defaulter Prediction System

**2)Problem Statement:**

Our project aims to develop an efficient system for predicting student defaulters based on their attendance records. By analyzing attendance patterns, we seek to forecast whether a student is likely to default on academic obligations.

# Introduction:

Our project focuses on developing a predictive system to identify student defaulters based on their attendance records using supervised learning techniques, specifically employing the Random Forest classification model. The dataset provided is in CSV format, containing labels denoted as YES or NO, indicating whether a student is likely to

default.

The core aim of our project is to provide educators with a tool that can predict student defaulters for any given month by analyzing their attendance data. By inputting the attendance for the upcoming month, our system generates predictions for all students, aiding teachers in proactive intervention and support.

**Key Features:**

**1)Random Forest Model:** Our system employs a Random Forest classification model to predict student defaulters. Random Forest is a powerful algorithm that works by creating multiple decision trees and combining their predictions. It's known for its ability to handle complex datasets effectively and produce accurate results.

**2)Real-time Prediction:** Educators can input the attendance data for the next month, and our system generates predictions for all students promptly, enabling timely intervention.

**3)Data Integrity Check:** Our system includes functionality to identify missing values in the CSV file, ensuring data accuracy and completeness.Which helps teacher to modify CSV file easily.

**4)Outlier Detection:** Using outlier detection techniques, our system flags instances where incorrect attendance entries have been made by teachers, such as attendance exceeding the total number of lectures.

**5)Visualization:** Our system offers visualizations depicting student defaulter rates, allowing educators to gain insights into attendance patterns and potential risk factors.

**6)Model Comparison:** Additionally, our project analyzes the performance of different classification models, including K-Nearest Neighbors (KNN) and Logistic Regression, evaluating their accuracy and potential for overfitting. Among these models, Random Forest emerges as the most effective choice for predicting student defaulters.

By combining machine learning with data visualization and model analysis, our project aims to provide educators with a comprehensive tool for proactive student support, ultimately enhancing academic success and retention rates.

# 3)Data set information (link,few data samples etc)

The dataset contains the following attributes:

**CNumber:** Unique identification number assigned to each student.

**Name of the Student:** Name of the student.

**Attendance for each subject:** IoT , SE, HS-OB, OE1-SC, OEI-ICCF, IOTL.

**Defaulter Status for Each Subject:** Indicates whether the student is a defaulter for each subject.

**Overall Defaulter Status:** Indicates whether the student is an overall defaulter, based on being a defaulter in at least 3 subjects.

[Google](https://colab.research.google.com/drive/1mXR4spjOWx_Rrb3e1MYjH_Vzx_IBPoQz?usp=sharing) [Collab](https://colab.research.google.com/drive/1mXR4spjOWx_Rrb3e1MYjH_Vzx_IBPoQz?usp=sharing)

[Datasets](https://drive.google.com/drive/folders/1NH18kvZf5uq9hM4YCHFLETdBqcHbJR3l?usp=sharing)

# Code and output:

Mount Google drive

from

google.colab

import

drive

drive.mount(

'/content/drive'

)

o/p:

Mounted at /content/drive

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Import the necessary libraries | | | | | | | |
| import pandas as pd |  | | | | | | |
|  |
| from sklearn.model\_selection import train\_test\_split | | | | | | |  |
|  | | | | | | |
| from sklearn.preprocessing import LabelEncoder | | | |  | | | |
|  | | | |
| from sklearn.neighbors import KNeighborsClassifier | | | | | |  | |
|  | | | | | |
| from sklearn.metrics import classification\_report | | | | |  | | |
|  | | | | |
| from scipy import stats | |  | | | | | |
|  | |
| import matplotlib.pyplot as plt | | |  | | | | |
|  | | |

K

NN

-

algorithm

It

giving

wr

ong

output

#

Load

the

dataset

df

=

pd.read\_csv(

"/content/drive/MyDrive/ML

datasets/DefaulterList

Sheet1

(1)

.csv"

)

#

Encode

the

target

variable

le

=

LabelEncoder()

df[

'Defaulter'

]

=

le.fit\_transform(df[

'Defaulter'

])

print

(

"Columns

in

the

DataFrame:"

)

print

df.columns

)

(

#

Selecting

only

the

specified

columns

for

features

selected\_columns

=

[

'IoT'

,

'SE'

,

'HS-OB'

,

'OE1-

SC'

,

'OEI-ICCF'

,

'IOTL'

]

X

=

df[selected\_columns]

#

Target

variable

y

=

df[

'Defaulter'

]

O/P:

Columns

in

the

DataFrame:

Index(['CNumber', 'Name of the Student', 'IoT', 'SE', 'HS-OB', 'OE1- SC',

'OEI-ICCF', 'IOTL', 'Defaulter(IoT)', 'Defaulter(SE)',

'Defaulter(HS-OB)', 'Defaulter(OE1- SC)', 'Defaulter(OEI-ICCF)',

'Defaulter(IOTL)', 'Defaulter'],

dtype='object')

T

r

aining

and

T

esting

#

Splitting

the

dataset

into

the

Training

set

and

Test

set

X\_train,

X\_test,

y\_train,

y\_test

=

train\_test\_split(X,

y,

test\_size=

0.3

,

random\_state=

42

)

print

(

"Missing

values

in

X\_test:"

,

X\_test.isnull().

sum

())

O/P:

Missing

values

in

X\_test:

IoT

0

|  |  |
| --- | --- |
| SE | 0 |
| HS-OB | 0 |
| OE1- SC | 0 |
| OEI-ICCF | 0 |
| IOTL | 0 |

dtype: int64

Remo

v

e

the

outliers

outlier\_indices

=

[]

for

column

in

selected\_columns:

outlier\_index

=

df[df[column]

>

30

]

.index

outlier\_indices.extend(outlier\_index)

if

len

(

)

outlier\_index

>

0

:

print

(

"Outliers

for"

,

column,

":"

)

for

idx

in

outlier\_index:

print

(

"CNumber:"

,

df.at[idx,

'CNumber'

]

,

",

Name:"

,

df.at[idx,

'Name

of

the

Student'

]

,

",

Subject:"

,

column,

",

Value:"

,

df.at[idx,

column])

#

Remove

outliers

from

the

dataset

df.drop(outlier\_indices,

inplace=

True

)

O/P:

Outliers

for

IoT

:

CNumber: C22018221430 , Name: AGARWAL MUSKAN , Subject: IoT , Value: 100 CNumber: C22018221431 , Name: JOSHI ADITI ANANT , Subject: IoT , Value: 100

Implementing Random Forest

|  |  |  |
| --- | --- | --- |
| from sklearn.ensemble import RandomForestClassifier | | |
| # Initialize the Random Forest classifier |  | |
|  |
| rf\_classifier = RandomForestClassifier(n\_estimators=100, random\_state=42) | |  |
|  | |
|  | | |

#

Train

the

classifier

rf\_classifier.fit(X\_train,

y\_train)

#

Predictions

on

the

test

set

y\_pred

=

rf\_classifier.predict(X\_test)

#

Evaluate

the

classifier

print

(

"Classification

Report:"

)

print

(

classification\_report(y\_test,

y\_pred))

O/P:

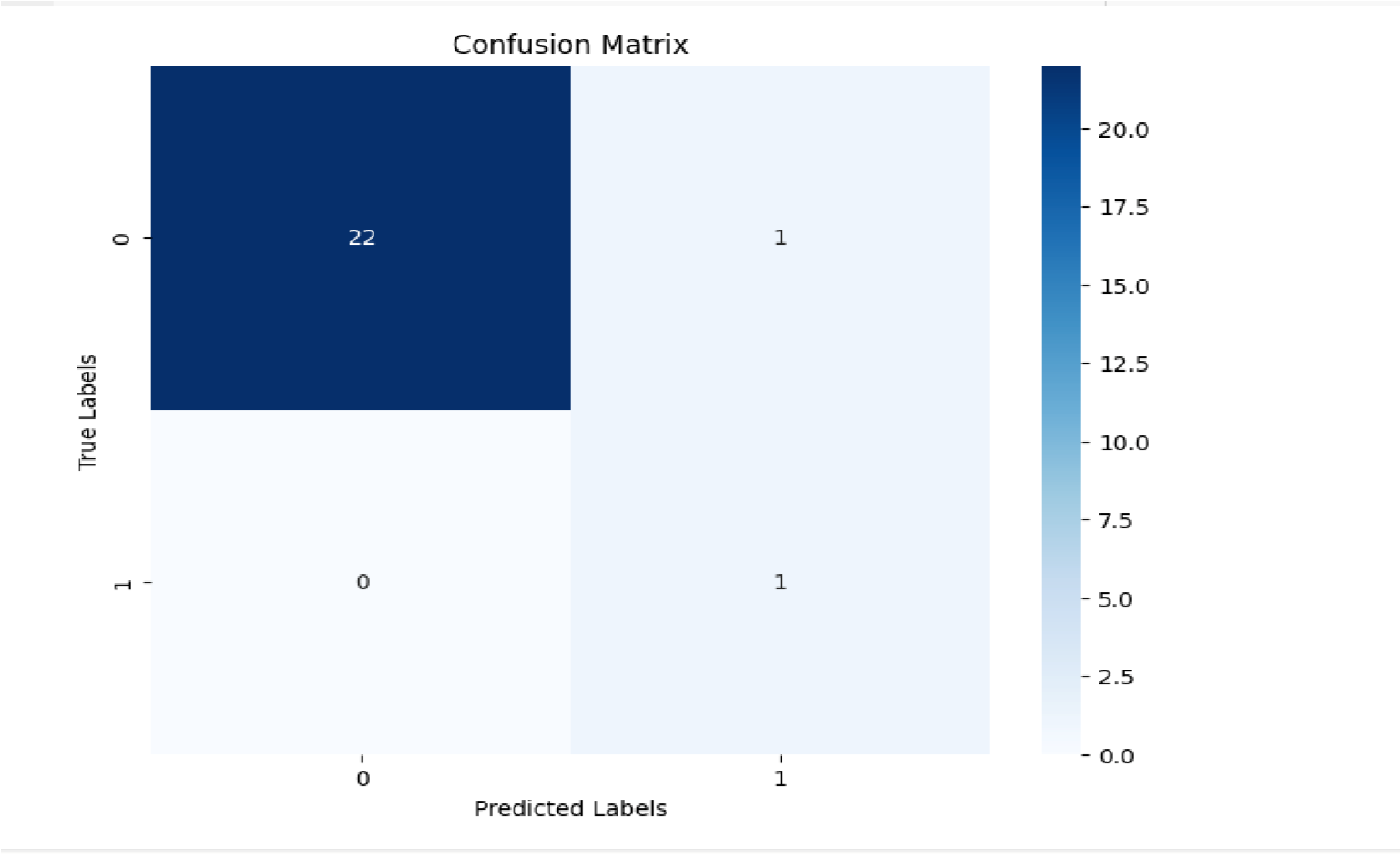
Classification

Report:

|  |  |  |  |
| --- | --- | --- | --- |
| precision | recall | f1-score | support |
| 0 1.00 | 0.96 | 0.98 | 23 |
| 1 0.50 | 1.00 | 0.67 | 1 |
| accuracy |  | 0.96 | 24 |
| macro avg 0.75 | 0.98 | 0.82 | 24 |
| weighted avg 0.98 | 0.96 | 0.96 | 24 |

Confusion Matrix

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| import seaborn as sns | | | | | | | | | | | | |
| import matplotlib.pyplot as plt | | | | | |  | | | | | | |
|  | | | | | |
| from sklearn.metrics import confusion\_matrix | | | | | | | |  | | | | |
|  | | | | | | | |
|  | | | | | | | | | | | | |
| # Compute confusion matrix | | |  | | | | | | | | | |
|  | | |
| conf\_matrix = confusion\_matrix(y\_test, y\_pred) | | | | | | | | | |  | | |
|  | | | | | | | | | |
|  | | | | | | | | | | | | |
| # Plot confusion matrix using seaborn heatmap | | | | | | | | |  | | | |
|  | | | | | | | | |
| plt.figure(figsize=(8, 6)) | | |  | | | | | | | | | |
|  | | |
| sns.heatmap(conf\_matrix, annot=True, fmt="d", cmap="Blues", | | | | | | | | | | | |  |
|  | | | | | | | | | | | |
| xticklabels=rf\_classifier.classes\_, | | | | | | | | | | |  | |
|  | | | | | | | | | | |
| yticklabels=rf\_classifier.classes\_) | | | | | | |  | | | | | |
|  | | | | | | |
| plt.title("Confusion Matrix") | | | |  | | | | | | | | |
|  | | | |
| plt.xlabel("Predicted Labels") | | | | |  | | | | | | | |
|  | | | | |
| plt.ylabel("True Labels") | |  | | | | | | | | | | |
|  | |
| plt.show() |  | | | | | | | | | | | |
|  |
|  | | | | | | | | | | | | |



**P**

**i**

**e**

**c**

**h**

**a**

**r**

**t**

#

Map

the

label

values

to

'YES'

and

'NO'

df[

'Defaulter'

]

=

df[

'Defaulter'

].

map

({

1

:

'YES'

,

0

:

'NO'

})

#

Count

the

occurrences

of

each

label

in

the

target

variable

label\_counts

=

df[

'Defaulter'

]

.value\_counts

()

#

Plotting

the

pie

chart

plt.figure(figsize=(

8

,

6

))

plt.pie(label\_counts,

labels=label\_counts.index,

autopct=

'%1.1f%%'

,

colors=[

'lightblue'

,

'lightgreen'

])

plt.title(

'Defaulter

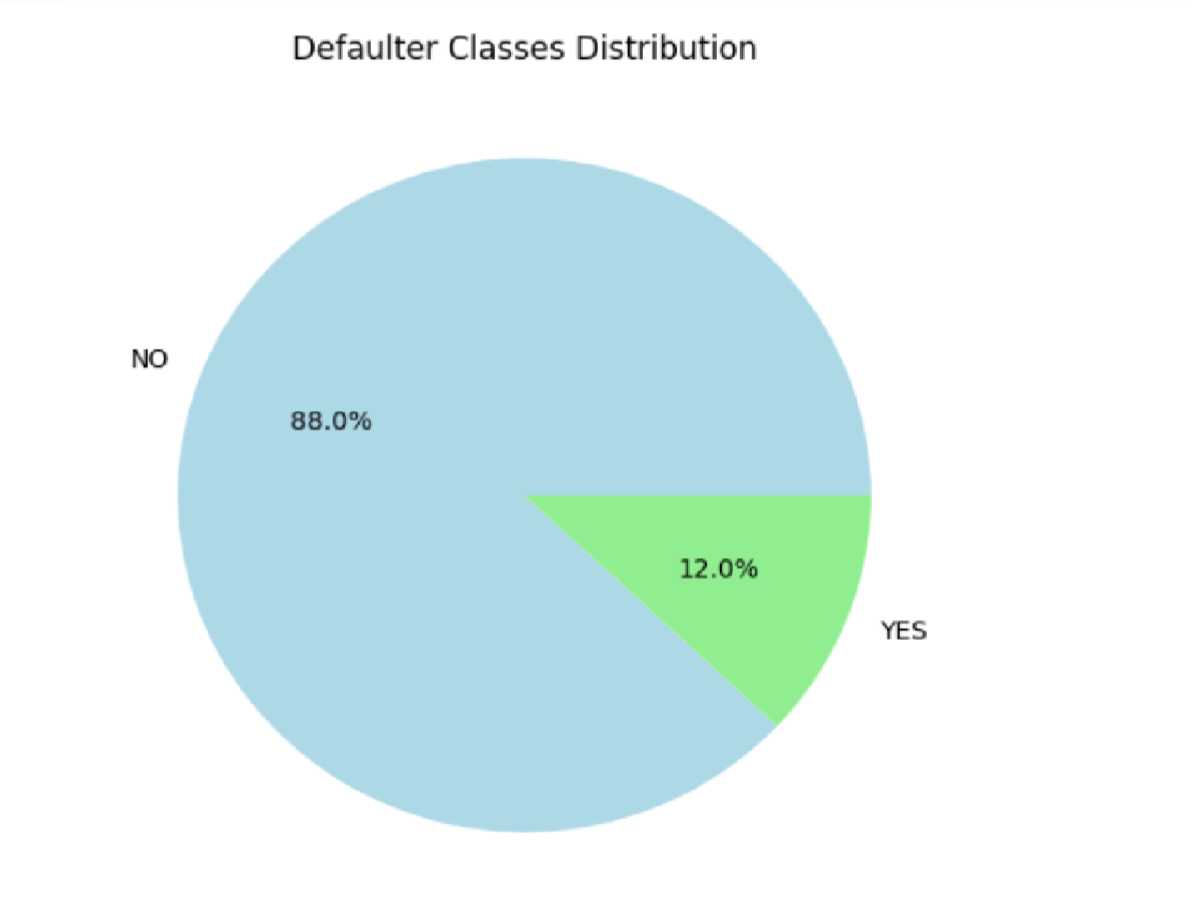
Classes

Distribution'

)

plt.show()

O/P



Side-by-side chart

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| import matplotlib.pyplot as plt | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| # Count the occurrences of "YES" and "NO" for each subject column | | | | | | | | | | | | | |  | | |
|  | | | | | | | | | | | | | |
| defaulter\_counts = {} | | | | | |  | | | | | | | | | | |
|  | | | | | |
| subjects = ['IoT', 'SE', 'HS-OB', 'OE1- SC', 'OEI-ICCF', 'IOTL'] | | | | | | | | | | | | |  | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| for subject in subjects: | | | | | | |  | | | | | | | | | |
|  | | | | | | |
| counts = df['Defaulter(' + subject + ')'] .value\_counts () | | | | | | | | | | | |  | | | | |
|  | | | | | | | | | | | |
| defaulter\_counts[subject] = counts | | | | | | | | | |  | | | | | | |
|  | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| # Extract counts for YES and NO | | | | | | | |  | | | | | | | | |
|  | | | | | | | |
| yes\_counts = [defaulter\_counts[subject].get('YES', 0) for subject in | | | | | | | | | | | | | | | |  |
|  | | | | | | | | | | | | | | | |
| subjects] |  | | | | | | | | | | | | | | | |
|  |
| no\_counts = [defaulter\_counts[subject].get('NO', 0) for subject in | | | | | | | | | | | | | | |  | |
|  | | | | | | | | | | | | | | |
| subjects] |  | | | | | | | | | | | | | | | |
|  |
|  | | | | | | | | | | | | | | | | |
| # Plotting | |  | | | | | | | | | | | | | | |
|  | |
| fig, ax = plt.subplots(figsize=(10, 6)) | | | | | | | | | | |  | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| # Bar width | | |  | | | | | | | | | | | | | |
|  | | |
| bar\_width = 0.35 | | | | |  | | | | | | | | | | | |
|  | | | | |
|  | | | | | | | | | | | | | | | | |
| # Bar positions | | | |  | | | | | | | | | | | | |
|  | | | |
| bar\_positions = range(len( subjects )) | | | | | | | | |  | | | | | | | |
|  | | | | | | | | |

#

Plotting

"NO"

counts

ax.bar([pos

-

bar\_width/

2

for

pos

in

bar\_positions],

no\_counts,

bar\_width,

label=

'NO'

,

color=

'lightgreen'

)

#

Plotting

"YES"

counts

ax.bar([pos

+

bar\_width/

2

for

pos

in

bar\_positions],

yes\_counts,

bar\_width,

label=

'YES'

,

color=

'lightblue'

)

#

Adding

labels

and

title

ax.set\_ylabel(

'Count'

)

ax.set\_title(

'Defaulter

Classes

Distribution

for

Each

Subject'

)

ax.set\_xticks(bar\_positions)

ax.set\_xticklabels(subjects)

ax.legend()

plt.xticks(rotation=

45

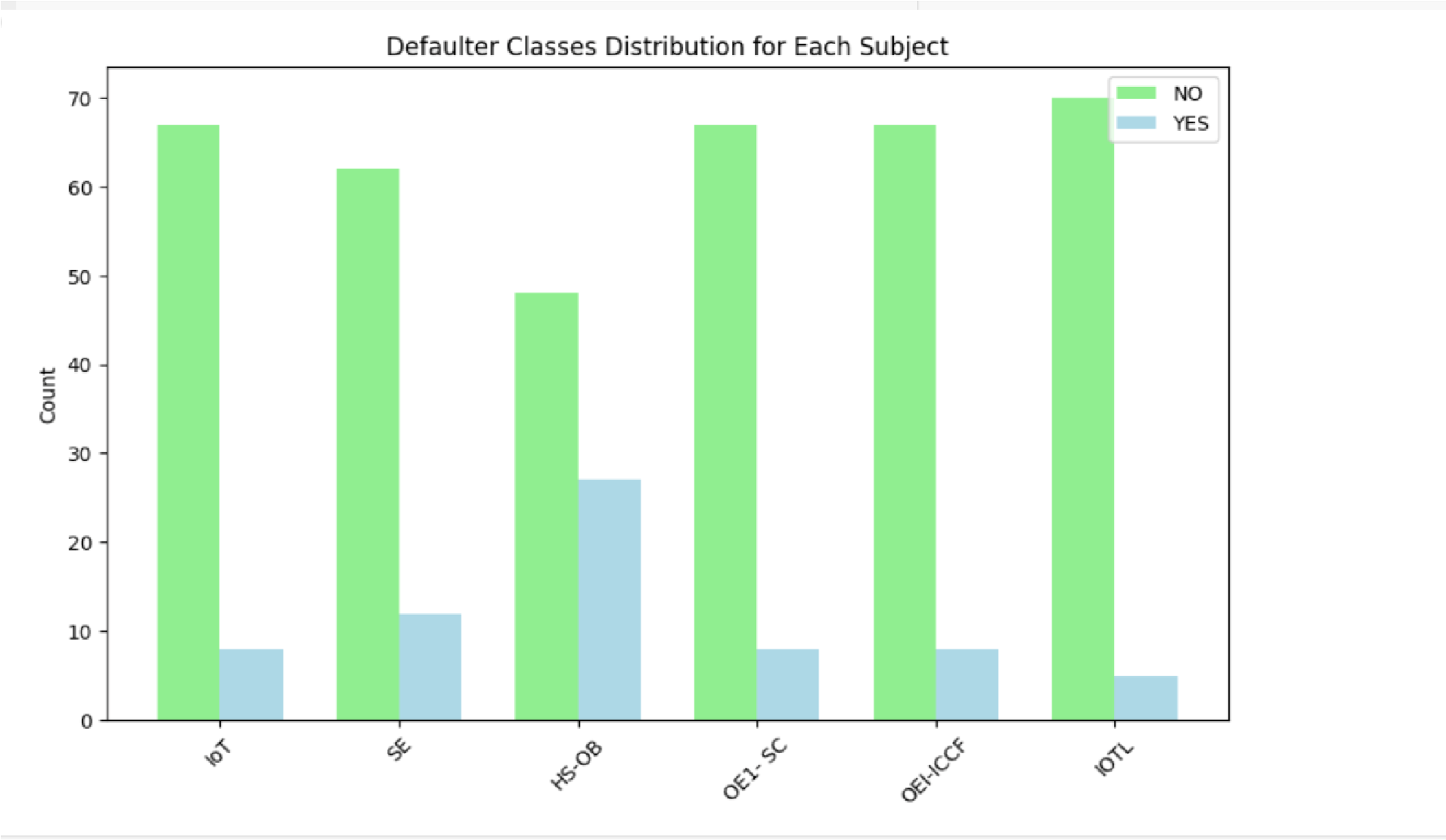
)

plt.show()

O/P:

T

esting



|  |  |
| --- | --- |
| #USER INPUT TESTING | |
| student\_attendance = { |  |
|  |

'IoT'

:

1

,

'SE'

:

1

,

'HS-OB'

:

10

,

'OE1-

SC'

:

12

,

'OEI-ICCF'

:

0

,

'IOTL'

:

0

}

#

Convert

the

student's

attendance

into

DataFrame

student\_df

=

pd.DataFrame([student\_attendance])

#

Predict

whether

the

student

is

a

defaulter

or

not

using

Random

Forest

classifier

defaulter\_prediction

=

rf\_classifier.predict(student\_df)

defaulter\_prediction\_label

=

le.inverse\_transform(defaulter\_prediction)

print

(

"Predicted

Defaulter

Label:"

,

defaulter\_prediction\_label[

0

])

O/P:

Predicted

Defaulter

Label:

YES

Predict Defaulter for next month data

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| # Load the dataset |  | | | | | | | | | | | |
|  |
| df1 = pd.read\_csv("/content/Defaulter\_NextMonth.csv") | | | | | | | | | |  | | |
|  | | | | | | | | | |
|  | | | | | | | | | | | | |
| # Initialize the LabelEncoder | | | | |  | | | | | | | |
|  | | | | |
| le = LabelEncoder() | |  | | | | | | | | | | |
|  | |
|  | | | | | | | | | | | | |
| # Fit the LabelEncoder on the target variable 'Defaulter' | | | | | | | | | | |  | |
|  | | | | | | | | | | |
| le.fit(df1['Defaulter']) | | |  | | | | | | | | | |
|  | | |
|  | | | | | | | | | | | | |
| # Selecting only the specified columns for features | | | | | | | | |  | | | |
|  | | | | | | | | |
| selected\_columns = ['IoT', 'SE', 'HS-OB', 'OE1- SC', 'OEI-ICCF', 'IOTL'] | | | | | | | | | | | |  |
|  | | | | | | | | | | | |
| X1 = df1[selected\_columns] | | | |  | | | | | | | | |
|  | | | |
|  | | | | | | | | | | | | |
| # Predict using the Random Forest classifier | | | | | | | |  | | | | |
|  | | | | | | | |
| y\_pred = rf\_classifier.predict(X1) | | | | | |  | | | | | | |
|  | | | | | |
|  | | | | | | | | | | | | |
| # Map predictions to 'Yes' and 'No' labels | | | | | | |  | | | | | |
|  | | | | | | |

y\_pred\_labels

=

[

'YES'

if

label

==

1

else

'NO'

for

label

in

y\_pred]

#

Assign

predicted

labels

to

the

'Defaulter'

column

df1[

'Defaulter'

]

=

y\_pred\_labels

#

Save

the

DataFrame

back

to

the

CSV

file

df1.to\_csv(

"/content/Defaulter\_NextMonth.csv"

,

index=

False

)

print

(

df

1[

"Defaulter"

])

O/P:

0

NO

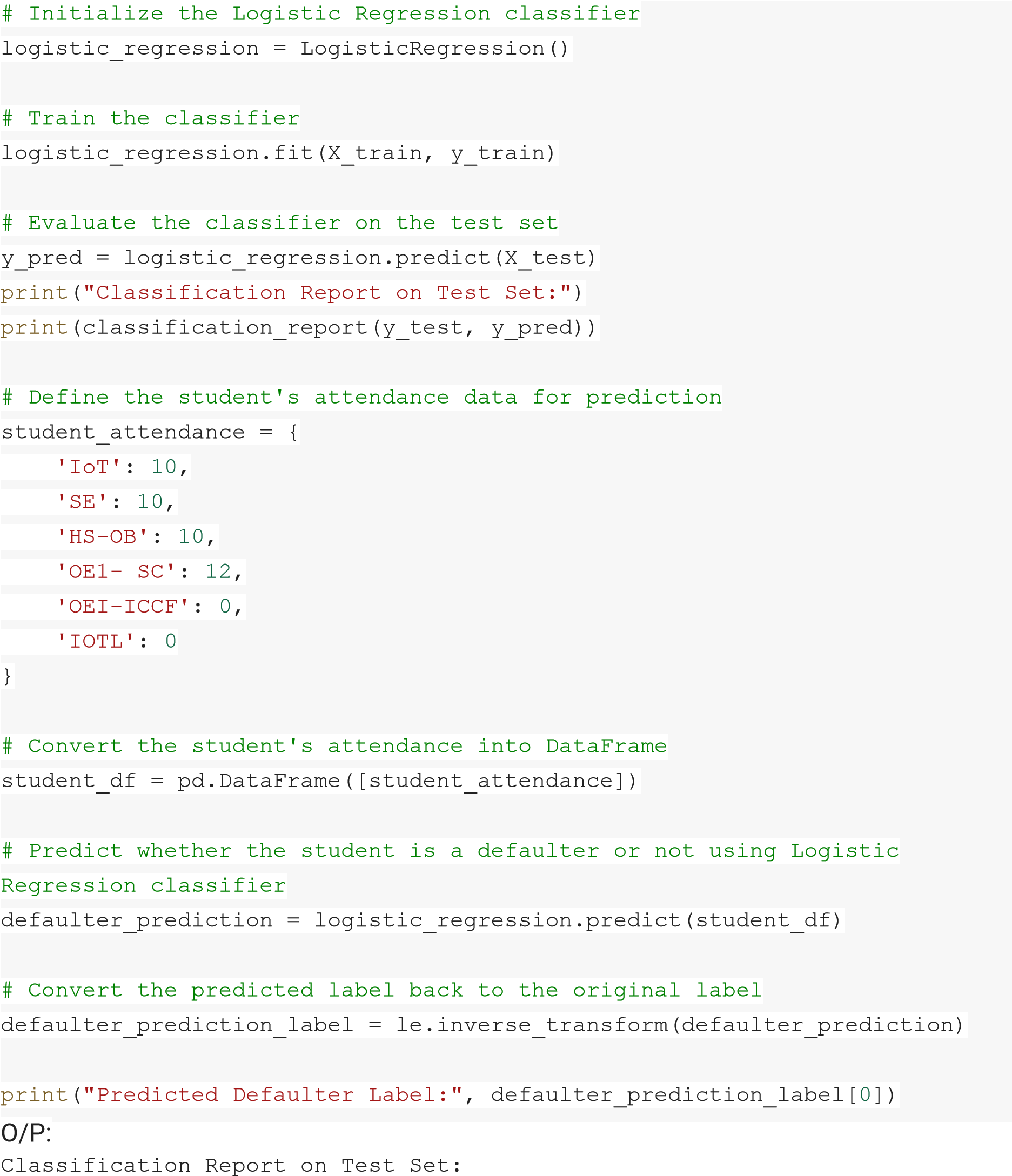
1. NO
2. NO
3. NO
4. NO

...

1. NO
2. NO
3. NO
4. YES 76 NO

Name: Defaulter, Length: 77, dtype: object Logistic regression is also giving wrong output

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| from sklearn.linear\_model import LogisticRegression | | | | | | | | | | | |
|  | | | | | | | | | | | |
| # Load the dataset | |  | | | | | | | | | |
|  | |
| df2 = pd.read\_csv("/content/drive/MyDrive/ML datasets/DefaulterList - | | | | | | | | | |  | |
|  | | | | | | | | | |
| Sheet1 (1).csv") |  | | | | | | | | | | |
|  |
|  | | | | | | | | | | | |
| # Encode the tar get variable | | | | | |  | | | | | |
|  | | | | | |
| le = LabelEncode r() | | |  | | | | | | | | |
|  | | |
| df2['Defaulter'] = le.fit\_transform(df2['Defaulter']) | | | | | | | | |  | | |
|  | | | | | | | | |
|  | | | | | | | | | | | |
| # Selecting only the specified columns for features | | | | | | | |  | | | |
|  | | | | | | | |
| selected\_columns = ['IoT', 'SE', 'HS-OB', 'OE1- SC', 'OEI-ICCF', 'IOTL'] | | | | | | | | | | |  |
|  | | | | | | | | | | |
| X = df2[selected \_columns] | | | | |  | | | | | | |
|  | | | | |
| y = df2['Default er'] | | | |  | | | | | | | |
|  | | | |
|  | | | | | | | | | | | |
| # Split the data 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) |  | | | | | | | | | | |
|  |



precision recall f1-score support

* 1. 1.00 0.88 0.93 16
  2. 0.00 0.00 0.00 0 accuracy 0.88 16

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| macro avg | 0.50 | 0.44 | 0.47 | 16 |
| weighted avg | 1.00 | 0.88 | 0.93 | 16 |

Predicted Defaulter Label: YES

Implementing

the

KNN

model

from

sklearn.neighbors

import

KNeighborsClassifier

from

sklearn.metrics

import

classification\_report

#

Initialize

the

KNN

classifier

knn\_classifier

=

KNeighborsClassifier(n\_neighbors=

5

)

#

Train

the

classifier

knn\_classifier.fit(X\_train,

y\_train)

#

Predictions

on

the

test

set

y\_pred

=

knn\_classifier.predict(X\_test)

#

Evaluate

the

classifier

print

(

"Classification

Report:"

)

print

(

classification\_report(y\_test,

y\_pred))

O/P:

Classification

Report:

|  |  |  |  |
| --- | --- | --- | --- |
| precision | recall | f1-score | support |
| 0 1.00 | 1.00 | 1.00 | 23 |
| 1 1.00 | 1.00 | 1.00 | 1 |
| accuracy |  | 1.00 | 24 |
| macro avg 1.00 | 1.00 | 1.00 | 24 |
| weighted avg 1.00 | 1.00 | 1.00 | 24 |

# Conclusion:

By analyzing attendance data across multiple subjects, our predictive model successfully identifies students at risk of defaulting on academic obligations. Leveraging a threshold of defaulting in at least 3 subjects to determine overall defaulter status, our system provides actionable insights for educators to intervene and support at-risk students effectively, ultimately promoting academic success and retention.

# References:

[Random](https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/) [Forest](https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/) [Algorithm](https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/) [in](https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/) [Machine](https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/) [Learning](https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/)

[Random](https://www.geeksforgeeks.org/random-forest-regression-in-python/) [Forest](https://www.geeksforgeeks.org/random-forest-regression-in-python/) [Regression](https://www.geeksforgeeks.org/random-forest-regression-in-python/) [in](https://www.geeksforgeeks.org/random-forest-regression-in-python/) [Python](https://www.geeksforgeeks.org/random-forest-regression-in-python/)

[K](https://www.geeksforgeeks.org/k-nearest-neighbors-with-python-ml/) [Nearest](https://www.geeksforgeeks.org/k-nearest-neighbors-with-python-ml/) [Neighbors](https://www.geeksforgeeks.org/k-nearest-neighbors-with-python-ml/) [with](https://www.geeksforgeeks.org/k-nearest-neighbors-with-python-ml/) [Python](https://www.geeksforgeeks.org/k-nearest-neighbors-with-python-ml/) [|](https://www.geeksforgeeks.org/k-nearest-neighbors-with-python-ml/) [ML](https://www.geeksforgeeks.org/k-nearest-neighbors-with-python-ml/)

[Logistic](https://www.geeksforgeeks.org/understanding-logistic-regression/) [Regression](https://www.geeksforgeeks.org/understanding-logistic-regression/) [in](https://www.geeksforgeeks.org/understanding-logistic-regression/) [Machine](https://www.geeksforgeeks.org/understanding-logistic-regression/) [Learning](https://www.geeksforgeeks.org/understanding-logistic-regression/)

[Matplotlib](https://www.geeksforgeeks.org/matplotlib-tutorial/) [Tutorial](https://www.geeksforgeeks.org/matplotlib-tutorial/)