TaskMachineLearningLast

April 21, 2024

#Kaggle competition by Tuwaiq Academy #Task 5 |week 5

#1- Competition Registration: a. Register for the competition on the Kaggle platform if you haven't already done so.

b. Access the competition page using the provided link.

c. Create your Team of 3 and give your team a name under KSA vision 2030. Ex: Tuwaiq, Hemah… etc

#mulhum | teem #Names of trainees:Safa nasser, lujain, Anwar

#2- Data Exploration and Preprocessing: a. Download the competition datasets provided on the Kaggle competition page.

b. Explore the datasets to understand the features and target variables.

c. Perform data preprocessing tasks such as handling missing values, encoding categorical variables, and scaling numerical features.

- Introduction

In an increasingly competitive world, the need for elite training programs is more important than ever. These programs aim to discover and develop exceptional talent, supporting them to achieve their full potential and make a positive impact in their fields. We are pleased to offer you the opportunity to participate in the Kaggle competition entitled "Unlocking the Potential of Elite Training Programs". This competition aims to provide a platform for trainees in elite programs to test their skills and gain practical experience in the field of machine learning using the famous Kaggle platform.

[ ]:
```
#Import library
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
```

[ ]:
```
pip install colorama
```

```
Requirement already satisfied: colorama in /usr/local/lib/python3.10/dist-
packages (0.4.6)
```

a. Download the competition datasets provided on the Kaggle competition page.

First, we'll load the data from the files you've uploaded using the pandas library, which is a powerful tool in Python for data manipulation and analysis

[ ]:
```
test_data = pd.read_csv('test.csv')
train_data = pd.read_csv('train.csv')
```

b. Explore the datasets to understand the features and target variables.

[ ]:
```
print(train_data.shape[0])
print(test_data.shape[0])
```

```
6548
818
```

- display Type data

[ ]:
```
from tabulate import tabulate
from termcolor import colored
from colorama import Fore, Back, Style, init
#
train_dtypes = train_data.dtypes
test_dtypes = test_data.dtypes

#     DataFrame
data_types_df = pd.DataFrame({
    'Train Data Types': train_dtypes,
    'Test Data Types': test_dtypes
})

table = tabulate(data_types_df, headers='keys', tablefmt='grid')

#
colored_table = table.replace('-', Fore.BLUE + '-')
print(colored_table)
```

```
+-------------
-------------
------+-------
-------------
+-------------
------+
|                                | Train Data Types    | Test Data Types    |
+================================+=====================+====================+
| Age                            | float64             | float64            |
+-------------
-------------
------+-------
-------------
+-------------
------+
| College                        | object              | object             |
+-------------
-------------
------+-------
-------------
+-------------
------+
| Completed Degree               | object              | object             |
+-------------
-------------
------+-------
-------------
+-------------
------+
| Education Speaciality          | object              | object             |
+-------------
-------------
------+-------
-------------
+-------------
------+
```

Identifying Numerical and Categorical Columns

After loading the data, we need to identify which columns are numerical and which are categorical. This is crucial because it determines how we'll process these columns moving forward

[ ]:
```
numerical_cols = train_data.select_dtypes(include=['int64', 'float64']).columns.
 ↪tolist()
categorical_cols = train_data.select_dtypes(include=['object']).columns.tolist()
numerical_cols = test_data.select_dtypes(include=['int64', 'float64']).columns.
 ↪tolist()
categorical_cols = test_data.select_dtypes(include=['object']).columns.tolist()
```

- display first 5 student

[ ]:
```
train_data.head()
```

[ ]:
```
                             Student ID   Age Gender   Home Region  Home City  \
0  4f14c50d-162e-4a15-9cf0-ec129c33bcf0  37.0
1  0599d409-876b-41a5-af05-749ef0e77d32  21.0
2  38a11c0e-4afc-4261-9c64-e94cc0a272fb  24.0
3  1693e85b-f80e-40ce-846f-395ddcece6d3  23.0
4  98a0e8d0-5f80-4634-afd8-322aa0902863  23.0

                             Program ID Program Main Category Code  \
0  453686d8-4023-4506-b2df-fac8b059ac26                       PCRF
1  cc8e4e42-65d5-4fa1-82f9-6c6c2d508b60                       APMR
2  e006900d-05a9-4c2b-a36f-0ffb9fce44cd                       APMR
3  2ec15f6b-233b-428a-b9f5-e40bc8d14cf9                       TOSL
4  d32da0e9-1aed-48c3-992d-a22f9ccc741e                       CAUF

  Program Sub Category Code Technology Type Program Skill Level  …  \
0                      PCRF             NaN                 NaN  …
1                      SWPS             NaN                    …
2                       NaN             NaN                    …
3                      TOSL             NaN                 NaN  …
4                      SWPS                                 …

  Completed Degree Level of Education                Education Speaciality  \
0
1
2                                              Information Technology
3                                         -    )     (
4

                        College University Degree Score  \
0                            NaN                   2.44
1                                        5.00
2                            NaN                   3.50
```

```
3                             NaN                   3.55
4                                      4.00

  University Degree Score System Employment Status Job Type  Still Working  Y
0                            4.0                       NaN            NaN  0
1                            5.0                        NaN            NaN  0
2                            5.0                        NaN            NaN  0
3                            5.0                        NaN            NaN  0
4                            5.0               NaN       NaN            NaN  0

[5 rows x 24 columns]
```

c. Perform data preprocessing tasks

- change Type data

[ ]:
```python
#Change table Train data:
train_data['Program Presentation Method'] = train_data['Program Presentation Method'].replace({'    ': 0, '    ': 1})
train_data['Program Presentation Method'] = train_data['Program Presentation Method'].astype(int)
#train_data['Program Start Date'] = pd.to_datetime(train_data['Program Start Date'])
#train_data['Program End Date'] = pd.to_datetime(train_data['Program End Date'])
train_data['Program Presentation Method'] = train_data['Program Presentation Method'].replace({'    ': 0, '    ': 1})
train_data['Program Presentation Method'] = train_data['Program Presentation Method'].astype(int)
```

```
train_data['Education Speaciality'] = train_data['Education Speaciality'].
↪replace({'Product Design':'       ','Business Administration':'  ␣
↪   ','Computer science and engineering':'             ','Human resources':
↪'        ','ENGINEER IN MARINE TECHNOLOGY':'                ','management␣
↪information system':'           ','Civil and Environmental engineering':
↪'             ','Computer Networking and Cyber security':'            ␣
↪     ','product design':'       ','Software Engineering':'  ␣
↪     ','Software engineering':'         ','Information technology':'    ␣
↪     ','Information Technology': '           ','information technology':
↪'          ','computer science':'        ','Computer Science':'  ␣
↪     ','Information System':'       ','Cybersecurity engineering':'      ␣
↪     ','Cybersecurity':'         ','cybersecurity':'         ','cyber␣
↪security':'         ','Is':'         ','Information Security':'  ␣
↪     ','Biology': '    ','Chemistry':'   ','Physics and astronomy':'       ␣
↪   ','Computer Network and Communication':'              ','IT security':
↪'        ','Art and design':'      ','Electrical and Electronics␣
↪Engineering':'                  ', 'IT': '          ','It':'     ␣
↪     ','Civil and Environmental engineering':'             ','Software␣
↪Engineer':'        ','Software Engineering' :'        ','Software␣
↪engineering':'        ','Software engineering':'        ','Computer␣
↪Networking and Cyber security':'                  ','Computer Network':'  ␣
↪     ','Computer Information Systems':'            ','Telecommunication and␣
```

```
                                                                                ␣
↪ 'Computer Science and Engineering':'              ','art education':'    ␣
↪  ','Dental surgery':'       ','libraries and Information':'    ␣
↪    ','Data Management':'       ','software engineer':'       ','Health␣
↪Informatics':'         ','Management Information Systmes':'       ␣
↪   ','PMP':'           ','language and translation':'       ','BSc␣
↪Geology':'    ','EMBA':'                 ','E-Business':'    ␣
↪     ','Mobile communication and security':'            ','Actuarial␣
↪Science':'      ','Information Tehnology':'          ','Information␣
↪Analytics':'         ','Graphic Design & Digital Media':'            ␣
↪   ','computer security':'       ','Advanced security and digital forensic':
↪'                 ','Information technology &amp; computing':'          ␣
↪   ','Statistics':'    ','COMPUTER ENGINEER':'       ','Mathematical':
↪'    ','Mathematics':'    ','Applied Computing-Computer Networks Track':'  ␣
↪        -         ','Electrical Engineering':'          ','IS':'   ␣
↪    ','Products design':'        ','Applied computing cyber security track':
↪'                    ','Computer':'   ','Multimedia':'         ','Computer␣
↪sciences':'       ','Engineering Management':'         ','information␣
↪System':'       ','Networking and telecommunication system':'            ␣
↪          ','Masters in Business Administration':'              ␣
↪   ','Health Information Management and Technology':'              ␣
↪  ','Cyber forensics and information security':'              ␣
↪    ','Human resources management':'            ','Physics':
↪'    ','Computer Information System ':'             ','Software system␣
↪engineering':'           ','Computer and information technology':'     ␣
↪           ','web devolper':'     ','accounting':'    ','Business':
↪'   ','Management information systems':'             ','Finance':
↪'   ','Computer Engineering Technology':'             ','Cybersecurity and␣
↪Info Assurance':'                 ','Computer Engineer':'   ␣
```

```
#Change table Test data:
test_data['Program Presentation Method'] = test_data['Program Presentation Method'].replace({'   ': 0, '    ': 1})
test_data['Program Presentation Method'] = test_data['Program Presentation Method'].astype(int)
test_data['Education Speaciality'] = test_data['Education Speaciality'].replace({'Biology': '    ','Chemistry':'   ','Physics and astronomy':'      ','Computer Network and Communication':'              ','IT security':'         ','Art and design':'      ','Electrical and Electronics Engineering':'                 ', 'IT': '          ','It':'       ','Civil and Environmental engineering':'             ', 'Software Engineering' :'        ','Software engineering':'        ','Software engineering':'        ','Computer Networking and Cyber security':'           ','Computer Network':'        ',  'Computer information systems':'           ','network security engineering':'         ','Data Science':'      ','Long Island University-New York':'         -     ','Civil and Environmental engineering':'             ','BS in Computer Science':'          ','Network and communication system':'            ','Computer programming':'          ','computer science':'        ','Computer sicence':'         ','Computer Science':'        ','MIS':'            ','Arabic NLP':'              ','product design':'       ','Product design':'        ','management information system':'           ','Construction Management':'         ',
'applied network systems engineering':'                ','Cybersecurity':'          ','Business Administration':'       ','E-Businesses':'        ','CS':'      ','Insurance and Risk management':'          ','Management Information System':'           ','Cloud computing architect':'            ','Educational Technology':'         ','Network telecommunication systems':'            ','Management information system':'             ','Information technology':'          ','Information Technology':'          ','University of Dubuque':'      ','Applied linguistics':'           ','Information Systems Security':'          ','Information Systems':'       ','Information System':'          ','information systems':'       ','English':'    ','Management':'   ','Businesses IT':'              ','Computer Engineering':'          ','Software engineer':'       ','Management Information Systems':'            ','Computer science':'        ','computer sicence':'           ','Information system':'       ','       - Computer Science':'          ','computer sciences':'       ','Data science':'       ',})
```

- handling missing values

Next, we'll address any missing values in the data. We'll use SimpleImputer from sklearn to fill missing values in numerical columns with the median and categorical columns with the most frequent value

[ ]:

```
train_isnull = train_data.isnull().sum()
test_isnull = test_data.isnull().sum()

#     DataFrame
data_isnull = pd.DataFrame({
    'Train Data Null': train_isnull,
    'Test Data Null': test_isnull
})

tableMissing = tabulate(data_isnull , headers='keys', tablefmt='grid')

#
colored_tableM = tableMissing.replace('-', Fore.BLUE + '-')
print(colored_tableM)
```

```
+-------------
-------------
------+-------
------------+-
-------------
----+
|                                |   Train Data Null |   Test Data Null |
+================================+===================+==================+
| Age                            |                92 |               14 |
+-------------
-------------
------+-------
------------+-
-------------
----+
| College                        |              3890 |              492 |
+-------------
-------------
------+-------
------------+-
-------------
----+
| Completed Degree               |                 0 |                0 |
+-------------
-------------
------+-------
------------+-
-------------
----+
| Education Speaciality          |               277 |               37 |
+-------------
-------------
------+-------
------------+-
-------------
----+
```

[ ]:

```
#handel with train table
'''train_data['Age'].fillna(train_data['Age'].median(), inplace=True)

mode_value = train_data['Level of Education'].mode()[0]
train_data['Level of Education'].fillna(value=mode_value, inplace=True)
mode_valueC = train_data['Home Region'].mode()[0]
train_data['Home City'].fillna(value=mode_value, inplace=True)
mode_valueC = train_data['Education Speaciality'].mode()[0]
train_data['Education Speaciality'].fillna(value=mode_valueC, inplace=True)
train_data['Program Sub Category Code'].fillna(train_data['Program Sub Category␣
↪Code'].mode(), inplace=True)
train_data['Technology Type'].fillna(train_data['Technology Type'].mode()[0],␣
↪inplace=True)
train_data['Program Skill Level'].fillna(train_data['Program Skill Level'].
↪mode()[0], inplace=True)
train_data['College'].fillna('Unknown', inplace=True)
train_data['University Degree Score'].fillna(train_data['University Degree␣
↪Score'].median(), inplace=True)
train_data['University Degree Score System'].fillna(train_data['University␣
↪Degree Score System'].mode()[0], inplace=True)
train_data['Employment Status'].fillna('Unknown', inplace=True)
train_data['Job Type'].fillna('Unknown', inplace=True)
train_data['Still Working'].fillna('Unknown', inplace=True)
mode_value = train_data['Program Sub Category Code'].mode()[0]
train_data['Program Sub Category Code'].fillna(value=mode_value, inplace=True)
#handel with test table
test_data['Age'].fillna(test_data['Age'].median(), inplace=True)
mode_value = test_data['Home Region'].mode()[0]
test_data['Home Region'].fillna(value=mode_value, inplace=True)
mode_value = test_data['Home Region'].mode()[0]
test_data['Home Region'].fillna(value=mode_value, inplace=True)
mode_valueC = test_data['Home Region'].mode()[0]
test_data['Home City'].fillna(value=mode_valueC, inplace=True)
test_data['Program Sub Category Code'].fillna(test_data['Program Sub Category␣
↪Code'].mode(), inplace=True)
test_data['Technology Type'].fillna(test_data['Technology Type'].mode()[0],␣
↪inplace=True)
test_data['Program Skill Level'].fillna(test_data['Program Skill Level'].
↪mode()[0], inplace=True)
test_data['College'].fillna('Unknown', inplace=True)
test_data['University Degree Score'].fillna(test_data['University Degree␣
↪Score'].median(), inplace=True)
test_data['University Degree Score System'].fillna(test_data['University Degree␣
↪Score System'].mode()[0], inplace=True)
test_data['Employment Status'].fillna('Unknown', inplace=True)
test_data['Job Type'].fillna('Unknown', inplace=True)
```

```
test_data['Still Working'].fillna('Unknown', inplace=True)
test_data['Education Speaciality'].fillna('Unknown', inplace=True)
mode_value = test_data['Level of Education'].mode()[0]
test_data['Level of Education'].fillna(value=mode_value, inplace=True)
mode_value = test_data['Program Sub Category Code'].mode()[0]
test_data['Program Sub Category Code'].fillna(value=mode_value, inplace=True)'''
```

[ ]:

```
"train_data['Age'].fillna(train_data['Age'].median(),
inplace=True)\n\nmode_value = train_data['Level of
Education'].mode()[0]\ntrain_data['Level of Education'].fillna(value=mode_value,
inplace=True)\nmode_valueC = train_data['Home
Region'].mode()[0]\ntrain_data['Home City'].fillna(value=mode_value,
inplace=True)\nmode_valueC = train_data['Education
Speaciality'].mode()[0]\ntrain_data['Education
Speaciality'].fillna(value=mode_valueC, inplace=True)\ntrain_data['Program Sub
Category Code'].fillna(train_data['Program Sub Category Code'].mode(),
inplace=True)\ntrain_data['Technology Type'].fillna(train_data['Technology
Type'].mode()[0], inplace=True)\ntrain_data['Program Skill
Level'].fillna(train_data['Program Skill Level'].mode()[0],
inplace=True)\ntrain_data['College'].fillna('Unknown',
inplace=True)\ntrain_data['University Degree
Score'].fillna(train_data['University Degree Score'].median(),
inplace=True)\ntrain_data['University Degree Score
System'].fillna(train_data['University Degree Score System'].mode()[0],
inplace=True)\ntrain_data['Employment Status'].fillna('Unknown',
inplace=True)\ntrain_data['Job Type'].fillna('Unknown',
inplace=True)\ntrain_data['Still Working'].fillna('Unknown',
inplace=True)\nmode_value = train_data['Program Sub Category
Code'].mode()[0]\ntrain_data['Program Sub Category
Code'].fillna(value=mode_value, inplace=True)\n#handel with test
table\ntest_data['Age'].fillna(test_data['Age'].median(),
inplace=True)\nmode_value = test_data['Home Region'].mode()[0]\ntest_data['Home
Region'].fillna(value=mode_value, inplace=True)\nmode_value = test_data['Home
Region'].mode()[0]\ntest_data['Home Region'].fillna(value=mode_value,
inplace=True)\nmode_valueC = test_data['Home Region'].mode()[0]\ntest_data['Home
City'].fillna(value=mode_valueC, inplace=True)\ntest_data['Program Sub Category
Code'].fillna(test_data['Program Sub Category Code'].mode(),
inplace=True)\ntest_data['Technology Type'].fillna(test_data['Technology
Type'].mode()[0], inplace=True)\ntest_data['Program Skill
Level'].fillna(test_data['Program Skill Level'].mode()[0],
inplace=True)\ntest_data['College'].fillna('Unknown',
inplace=True)\ntest_data['University Degree Score'].fillna(test_data['University
Degree Score'].median(), inplace=True)\ntest_data['University Degree Score
System'].fillna(test_data['University Degree Score System'].mode()[0],
inplace=True)\ntest_data['Employment Status'].fillna('Unknown',
inplace=True)\ntest_data['Job Type'].fillna('Unknown',
inplace=True)\ntest_data['Still Working'].fillna('Unknown',
```

```
inplace=True)\ntest_data['Education Speaciality'].fillna('Unknown',
inplace=True)\nmode_value = test_data['Level of
Education'].mode()[0]\ntest_data['Level of Education'].fillna(value=mode_value,
inplace=True)\nmode_value = test_data['Program Sub Category
Code'].mode()[0]\ntest_data['Program Sub Category
Code'].fillna(value=mode_value, inplace=True)"
```

[ ]:
```python
from sklearn.impute import SimpleImputer

# Imputer for numerical data
numerical_imputer = SimpleImputer(strategy='median')
# Imputer for categorical data
categorical_imputer = SimpleImputer(strategy='most_frequent')

# Apply imputation to numerical columns
train_data[numerical_cols] = numerical_imputer.
 ↪fit_transform(train_data[numerical_cols])
test_data[numerical_cols] = numerical_imputer.
 ↪transform(test_data[numerical_cols])

# Apply imputation to categorical columns
train_data[categorical_cols] = categorical_imputer.
 ↪fit_transform(train_data[categorical_cols])
test_data[categorical_cols] = categorical_imputer.
 ↪transform(test_data[categorical_cols])
```

[ ]:
```python
train_isnull = train_data.isnull().sum()
test_isnull = test_data.isnull().sum()

#     DataFrame
data_isnull = pd.DataFrame({
    'Train Data Null': train_isnull,
    'Test Data Null': test_isnull
})

tableMissing = tabulate(data_isnull , headers='keys', tablefmt='grid')

#
colored_tableM = tableMissing.replace('-', Fore.BLUE + '-')
print(colored_tableM)
```

```
+-------------
-------------
------+-------
-------------+-
-------------
----+
|                                 |   Train Data Null |   Test Data Null |
+=================================+===================+==================+
| Age                             |                 0 |                0 |
+-------------
-------------
------+-------
-------------+-
-------------
----+
| College                         |                 0 |                0 |
+-------------
-------------
------+-------
-------------+-
-------------
----+
| Completed Degree                |                 0 |                0 |
+-------------
-------------
------+-------
-------------+-
-------------
----+
| Education Speaciality           |                 0 |                0 |
+-------------
-------------
------+-------
-------------+-
-------------
----+
```

- Scaling numerical features and encoding categorical variables

Finally, we'll encode categorical data and scale numerical data to prepare it for modeling

[ ]:
```
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer

# Set up the data transformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols),
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_cols)
    ])

# Apply transformations to the data
train_data_preprocessed = preprocessor.fit_transform(train_data)
test_data_preprocessed = preprocessor.transform(test_data)
```

Summary:

Encoding transforms categorical data into a format that is easier for ML models to understand.

Scaling adjusts the range of numerical features so that different scales do not distort the learning process of the model.

Both steps help in enhancing model performance and are crucial for achieving accurate predictions.

With these steps completed, the data is now ready to be used for building models. This setup ensures that the data fed into the model is clean, well-structured, and appropriate for the algorithms you might want to use

1 ____

2 3-Model Building

a. Split the preprocessed data into training and testing sets.

b. Implement machine learning models to predict student persistence and completion rates. You can start with simple models like logistic regression or decision trees.

c. Evaluate the performance of each model using appropriate evaluation metrics such as accuracy, precision, recall, or F1-score

A.Build Models

Start by implementing simpler models. Simpler models are easier to interpret and can often provide baseline results quickly.

Logistic Regression: Good for binary classification tasks.

Decision Trees: Useful for handling nonlinear data with a hierarchical structure of decisions

Evaluate Models

To determine the best model, we evaluate them using metrics such as accuracy, precision, recall, and F1-score.

- Accuracy: Measures the overall correctness of the model.
- Precision: Measures the accuracy of positive predictions.
- Recall: Measures the model's ability to detect positive samples.
- F1-Score: Harmonic mean of precision and recall

- Split the preprocessed data into training and testing sets.

[ ]:
```
X_train = train_data_preprocessed
y_train = train_data["Y"]
X_test = test_data_preprocessed
```

- Implement machine learning models to predict student persistence and completion

[24]:
```
try:
    modelL = LogisticRegression(C=1.0, max_iter=1000)
    modelL.fit(X_train, y_train)
    predictions = modelL.predict(X_test)
    print("The model was trained successfully.")

    scoring = ['accuracy', 'precision', 'recall', 'f1']
    scores = cross_validate(modelL, X_train, y_train, scoring=scoring, cv=5)

    print("Cross-Validation Scores:")
    cv_results = pd.DataFrame(scores)
    table = tabulate(cv_results, headers=['Fit_time', 'Score_time',␣
↪'Test_accuracy', 'Test_precision', 'Test_recall', 'Test_f1'],␣
↪tablefmt='grid')
    colored_tableM = table.replace('-', Fore.BLUE + '-')
    print(colored_tableM)

    avg_accuracy = scores['test_accuracy'].mean()
    avg_f1 = scores['test_f1'].mean()  # Calculate average F1 score
    print("Average test accuracy:", avg_accuracy)
    print("Average F1 score:", avg_f1)

except Exception as e:
    print("Failure to train:", e)
```

```
The model was trained successfully.
Cross-Validation Scores:
```

```
+----+---------
---+----------
----+---------
--------+-----
-------------
+-------------
--+-----------+
|    |   Fit_time |   Score_time |   Test_accuracy |   Test_precision |
Test_recall |   Test_f1 |
+====+===========+=============+================+================+=========
======+===========+
|  0 |   1.43223  |    0.0257907 |        0.89542  |         0.741497 |
0.524038 |  0.614085 |
+----+---------
---+----------
----+---------
--------+-----
-------------
+-------------
--+-----------+
|  1 |   0.826078 |    0.0348432 |        0.905344 |         0.7625   |
0.586538 |  0.663043 |
+----+---------
---+----------
----+---------
--------+-----
-------------
+-------------
--+-----------+
|  2 |   0.844897 |    0.0157795 |        0.885496 |         0.683544 |
0.519231 |  0.590164 |
+----+---------
---+----------
----+---------
--------+-----
-------------
```

[ ]:

[25]:

```
try:
    modelT = DecisionTreeClassifier(max_depth=2, random_state=42)

    modelT.fit(X_train, y_train)
    dtree_pred=modelT.predict(X_test)
    print("The model was trained successfully.")
    scoring = ['accuracy', 'precision', 'recall', 'f1']
    scores = cross_validate(modelT, X_train, y_train, scoring=scoring, cv=5)
    print("Cross-Validation Scores:")


    cv_results = pd.DataFrame(scores)
    table = tabulate(cv_results, headers=['Fit_time', 'Score_time',␣
 ↪'Test_accuracy', 'Test_precision', 'Test_recall', 'Test_f1'],␣
 ↪tablefmt='grid')
    colored_tableM = table.replace('-', Fore.BLUE + '-')
    print(colored_tableM)
    avg_accuracy = scores['test_accuracy'].mean()
    avg_f1 = scores['test_f1'].mean()  # Calculate average F1 score
    print("Average test accuracy:", avg_accuracy)
    print("Average F1 score:", avg_f1)



except Exception as e:
    print("Failure to train:", e)
```

```
The model was trained successfully.
Cross-Validation Scores:
```

```
+----+---------
---+----------
----+---------
--------+-----
-------------
+-------------
--+-----------+
|    |   Fit_time |   Score_time |   Test_accuracy |   Test_precision |
Test_recall |   Test_f1 |
+====+============+==============+=================+==================+=========
======+===========+
|  0 |  0.207944  |    0.0189064 |        0.893893 |         0.660465 |
0.682692 |  0.671395 |
+----+---------
---+----------
----+---------
--------+-----
-------------
+-------------
--+-----------+
|  1 |  0.182964  |    0.0284693 |        0.883969 |         0.625    |
0.673077 |  0.648148 |
+----+---------
---+----------
----+---------
--------+-----
-------------
+-------------
--+-----------+
|  2 |  0.134637  |    0.01314   |        0.861832 |         0.566502 |
0.552885 |  0.559611 |
+----+---------
---+----------
----+---------
--------+-----
-------------
```

3 4- Model Improvement

a. Experiment with different machine learning algorithms and techniques to improve model performance.

b. Fine-tune hyperparameters of the selected models to optimize performance.

c. Explore feature engineering techniques to extract more meaningful insights from the data.

A.Experimenting with Different Algorithms

- Random Forest: Provides an improvement over decision trees by reducing the risk of overfitting.
- Support Vector Machines (SVM): Effective for high-dimensional problems.
- Gradient Boosting: A powerful model that focuses on correcting the errors made by previous estimators

Use GridSearchCV or RandomizedSearchCV to find the optimal set of parameters

- Experiment with different machine learning algorithms and techniques to improve model performance.

[ ]:
```
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
# Standardize the data
scaler = StandardScaler(with_mean=False)

X_scaled = scaler.fit_transform(X_train)

# Support Vector Machine
svm_model = SVC(kernel='linear')

# Random Forest
rf_model = RandomForestClassifier(n_estimators=100)

# Gradient Boosting
gb_model = GradientBoostingClassifier(n_estimators=100)

# Evaluate models using cross-validation
svm_scores = cross_val_score(svm_model, X_scaled,y_train, cv=5)
rf_scores = cross_val_score(rf_model, X_scaled, y_train, cv=5)
gb_scores = cross_val_score(gb_model, X_scaled, y_train, cv=5)

print("SVM Accuracy: %0.2f (+/- %0.2f)" % (svm_scores.mean(), svm_scores.std()␣
↪* 2))
print("Random Forest Accuracy: %0.2f (+/- %0.2f)" % (rf_scores.mean(),␣
↪rf_scores.std() * 2))
```

```python
print("Gradient Boosting Accuracy: %0.2f (+/- %0.2f)" % (gb_scores.mean(),␣
↪gb_scores.std() * 2))
# create Dictionary content all accuracy
model_scores = {'SVM': (svm_scores.mean(), svm_scores.std()),'Random Forest':␣
↪(rf_scores.mean(), rf_scores.std()) ,'Gradient Boosting': (gb_scores.mean(),␣
↪gb_scores.std())}
# find max accuracy
best_model = max(model_scores, key=lambda k: model_scores[k][0])
print('---------------------------------------------------------------')
print(f"The best performing model is {best_model} with an accuracy of␣
↪{model_scores[best_model][0]:.2f} ")
```

```
SVM Accuracy: 0.80 (+/- 0.01)
Random Forest Accuracy: 0.89 (+/- 0.02)
Gradient Boosting Accuracy: 0.89 (+/- 0.02)
----------------------------------------------------------------
The best performing model is Gradient Boosting with an accuracy of 0.89
```

- Fine-tune hyperparameters of the selected models to optimize performance.

[21]:
```python
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GradientBoostingClassifier

# Create Gradient Boosting Classifier
gb_clf = GradientBoostingClassifier()

# Hyperparameters to tune
param_grid = {
    'n_estimators': [100, 200],
    'learning_rate': [0.05, 0.1],
    'max_depth': [3, 5]
}
# Create the GridSearchCV object
grid_search = GridSearchCV(gb_clf, param_grid, cv=5, verbose=0)
grid_search.fit(X_train, y_train)

best_rf_scores = cross_val_score(gb_clf, X_scaled, y_train, cv=5)
b=np.mean(best_rf_scores)
print(f"Tuned model Accuracy: {np.mean(best_rf_scores):.2f} (+/- {np.
↪std(best_rf_scores) * 2:.2f})")
print("Best parameters:", grid_search.best_params_)
```

```
Tuned model Accuracy: 0.89 (+/- 0.02)
Best parameters: {'learning_rate': 0.05, 'max_depth': 5, 'n_estimators': 100}
```

[28]:
```python
scoring = ['accuracy', 'precision', 'recall', 'f1']
    scores = cross_validate(grid_search, X_train, y_train, scoring=scoring,␣
↪cv=5)
```

```
    print("Cross-Validation Scores:")


    cv_results = pd.DataFrame(scores)
    table = tabulate(cv_results, headers=['Fit_time', 'Score_time',␣
 ↪'Test_accuracy', 'Test_precision', 'Test_recall', 'Test_f1'],␣
 ↪tablefmt='grid')
    colored_tableM = table.replace('-', Fore.BLUE + '-')
    print(colored_tableM)
    avg_accuracy = scores['test_accuracy'].mean()
    avg_f1 = scores['test_f1'].mean()  # Calculate average F1 score
    print("Average test accuracy:", avg_accuracy)
    print("Average F1 score:", avg_f1)
```

```
Cross-Validation Scores:
```

```
+----+---------
---+----------
----+---------
--------+-----
-------------
+-------------
--+-----------+
|    |   Fit_time |    Score_time |    Test_accuracy |    Test_precision |
Test_recall |    Test_f1 |
+====+============+==============+=================+=================+=========
======+===========+
|  0 |    281.158 |     0.0111201 |         0.89771  |          0.737179 |
0.552885 |  0.631868 |
+----+---------
---+----------
----+---------
--------+-----
-------------
+-------------
--+-----------+
|  1 |    284.504 |     0.0109675 |         0.903817 |          0.715789 |
0.653846 |  0.683417 |
+----+---------
---+----------
----+---------
--------+-----
-------------
+-------------
--+-----------+
|  2 |    287.037 |     0.0115414 |         0.880916 |          0.638298 |
0.576923 |  0.606061 |
+----+---------
---+----------
----+---------
--------+-----
-------------
```

- Explore feature engineering techniques to extract more meaningful insights from the data.

```
[ ]: train_data.drop(['Student ID', 'Program ID'], axis=1)
     test_data.drop(['Student ID', 'Program ID'], axis=1)
```

```
[ ]:      Age Gender   Home Region Home City Program Main Category Code  \
     0    23.0                                                  CAUF
     1    31.0                                                  PCRF
     2    29.0                                                  CAUF
     3    23.0                                                  PCRF
     4    30.0                                                  TOSL
     ..    …     …           …        …                           …
     813  36.0                                                  GRST
     814  29.0                                                  CAUF
     815  32.0                                                  GRST
     816  28.0                                                  PCRF
     817  23.0                                                  PCRF

         Program Sub Category Code Technology Type Program Skill Level  \
     0                        SWPS
     1                        PCRF
     2                        SWPS
     3                        PCRF
     4                        SWPS
     ..                         …              …                  …
     813                      INFA
     814                      CRDP
     815                      INFA
     816                      PCRF
     817                      PCRF

         Program Presentation Method Program Start Date  … Program Days  \
     0                             1         2023-10-08  …          5.0
     1                             1         2023-07-16  …         19.0
     2                             1         2022-12-25  …         12.0
     3                             0         2023-03-19  …          5.0
     4                             0         2023-11-12  …         33.0
     ..                          …                  …  …          …
     813                           0         2023-08-13  …          5.0
     814                           0         2023-11-26  …         47.0
     815                           0         2023-07-23  …          5.0
     816                           1         2023-05-14  …        173.0
     817                           0         2023-07-16  …         12.0

          Completed Degree Level of Education Education Speaciality  \
     0
     1
```

```
2
3
4
..                 …                 …                    …
813
814
815
816
817

                            College University Degree Score  \
0                                         3.72
1                                         2.00
2                                         3.72
3                                         4.47
4                                         4.46
..                               …                      …
813                                       2.55
814                                          3.00
815                                       3.00
816                                       4.12
817                                       4.55

     University Degree Score System  Employment Status   Job Type  \
0                              4.0
1                              4.0
2                              5.0
3                              5.0
4                              5.0
..                             …                 …        …
813                            5.0
814                            4.0
815                            5.0
816                            5.0
817                            5.0

    Still Working
0             Yes
1             Yes
2             Yes
3             Yes
4              No
..            …
813           Yes
814           Yes
815           Yes
816           Yes
```

```
817          Yes

[818 rows x 21 columns]
```

4 Conclusion

we conclude that the best algorithm is:

logistical and Gradient Boosting.One of the difficulties we faced was that the data had an Arabic part and an English part, and this was solved by converting the English parts to Arabic and the missing data is large .

- the best model is Gradient Boosting Classifier
- Average test accuracy: 0.8901936680293213
- Average F1 score: 0.6313645083518844