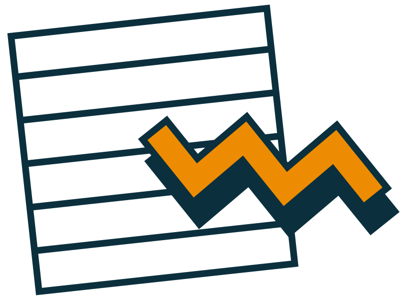
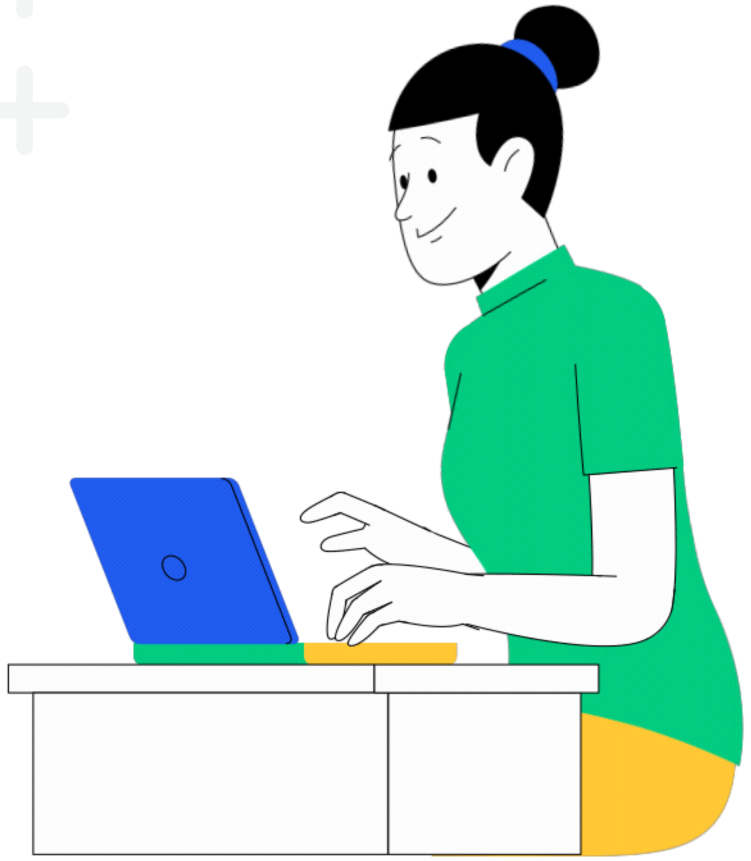


KAGGLE COMPETITION BY TUWAIQ ACADEMY

TASK 5 ..



أكاديمية طويق
TUWAIQ ACADEMY

“Mulhum | مُلهم Team”

Names of trainees : Safa Nasser, Anwar , lujain

أكاديمية سدايا
SDAIA Academy

SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority

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.

About Data set..

Student ID Age Gender Home Region Home City Program ID Program Main Category Code Program Sub Category Code Technology Type Program Skill Level ... Completed Degree Level of Education Education Speaciality College University Degree Score University Degree Score System Employment Status Job Type Still Working

What we did

01 DATA PREPROCESSING

Clean the data and convert it into a format suitable for analysis

02 BUILDING THE MODEL

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque non elit mauris. Cras euismod, metus ac finibus finibus, felis dui suscipit purus, a maximus leo ligula

03 MODEL EVALUATION

Measure model accuracy and improve it using different techniques.

Download the competition datasets provided on the Kaggle competition page.

```
[1]  #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
```

```
[2]  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

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

Explore the datasets to understand the features and target variables.

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

[4]
```python
print(train_data.shape[0])
print(test_data.shape[0])
```

```
6548
818
```

- display Type data

[5]
```python
from tabulate import tabulate
from termcolor import colored
from colorama import Fore, Back, Style, init
# استخراج أنواع البيانات
train_dtypes = train_data.dtypes
test_dtypes = test_data.dtypes

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

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

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

[5]

	Train Data Types	Test Data Types
Age	float64	float64
College	object	object
Completed Degree	object	object
Education Speaciality	object	object
Employment Status	object	object
Gender	object	object
Home City	object	object
Home Region	object	object
Job Type	object	object
Level of Education	object	object
Program Days	int64	int64
Program End Date	object	object
Program ID	object	object
Program Main Category Code	object	object
Program Presentation Method	object	object
Program Skill Level	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

```
[6] 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()
```

c. Perform data preprocessing tasks

- change Type data

```
[8] #Change table Train data:
    train_data['Program Presentation Method'] = train_data['Program Presentation Method'].replace({'حضوري': 0, 'عن بعد': 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({'حضوري': 0, 'عن بعد': 1})
    train_data['Program Presentation Method'] = train_data['Program Presentation Method'].astype(int)
    train_data['Education Speaciality'] = train_data['Education Speaciality'].replace({'Product Design':'تصميم المنتج','Business Administration':'إدارة الأعمال','Computer science and eng
                                                                                      'Computer Science and Engineering':'علوم الكومبيوتر و الهندسة','art education':'التربية الفنية','[

    #Change table Test data:
    test_data['Program Presentation Method'] = test_data['Program Presentation Method'].replace({'حضوري': 0, 'عن بعد': 1})
    test_data['Program Presentation Method'] = test_data['Program Presentation Method'].astype(int)
    test_data['Education Speaciality'] = test_data['Education Speaciality'].replace({'Biology': 'الاحياء','Chemistry':'كيمياء','Physics and astronomy':'الفيزياء وعلم الفلك','Computer Ne
                                                                                    'applied network systems engineering':'هندسة أنظمة الشبكات التطبيقية','Cybersecurity':'أمن الإلكتروني
```

Before and after data cleaning

	Train Data Null	Test Data Null
Age	92	14
College	3890	492
Completed Degree	0	0
Education Speaciality	277	37
Employment Status	566	70
Gender	0	0
Home City	2	1
Home Region	2	1
Job Type	4567	581
Level of Education	26	3
Program Days	0	0
Program End Date	0	0
Program ID	0	0
Program Main Category Code	0	0
Program Presentation Method	0	0

	Train Data Null	Test Data Null
Age	0	0
College	0	0
Completed Degree	0	0
Education Speaciality	0	0
Employment Status	0	0
Gender	0	0
Home City	0	0
Home Region	0	0
Job Type	0	0
Level of Education	0	0
Program Days	0	0
Program End Date	0	0
Program ID	0	0
Program Main Category Code	0	0
Program Presentation Method	0	0
Program Skill Level	0	0

- Scaling numerical features and encoding categorical variables

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

```
[13] 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.

We transformed the data using ColumnTransformer
from the Scikit-learn library.
This allows us to apply different transformations to different columns in the data set.

- Split the preprocessed data into training and testing sets.

```
[14]
X_train = train_data_preprocessed
y_train = train_data["Y"]
X_test = test_data_preprocessed
```

In the preprocessing process: We identified the independent variables and dependent variables for both the training data set and the test data set.

- Implement machine learning models to predict student persistence and completion

```python
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
3	0.630645	0.0294232	0.879297	0.658065	0.492754	0.563536
4	1.00152	0.0194485	0.884645	0.681529	0.514423	0.586301

```
Average test accuracy: 0.8900402964794523
Average F1 score: 0.6034258402389152
```

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

```
[17] 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))
     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]

```
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}
```

```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
3	285.547	0.0166445	0.877005	0.629213	0.541063	0.581818
4	290.524	0.0160437	0.89152	0.663366	0.644231	0.653659

Average test accuracy: 0.8901936680293213
Average F1 score: 0.6313645083518844

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

```python
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	Program Sub Category Code	Technology Type	Program Skill Level	Program Presentation Method	Program Start Date	...	Program Days	Completed Degree	Level of Education	Education Speaciality	College	University Degree Score	University Degree Score System	Employment Status	Job Type	Still Working
0	23.0	أنثى	منطقة الرياض	الرياض	CAUF	SWPS	تقليدية	متوسط	1	2023-10-08	...	5.0	لا	البكالوريوس	علوم الحاسب الالي	تكنولوجيا الاتصالات والمعلومات	3.72	4.0	خريج	دوام كامل	Yes
1	31.0	أنثى	منطقة الرياض	الرياض	PCRF	PCRF	تقليدية	مبتدئ	1	2023-07-16	...	19.0	لا	البكالوريوس	تقنية المعلومات	تكنولوجيا الاتصالات والمعلومات	2.00	4.0	موظف	دوام كامل	Yes
2	29.0	أنثى	منطقة الرياض	الرياض	CAUF	SWPS	تقليدية	متوسط	1	2022-12-25	...	12.0	نعم	البكالوريوس	شبكات الحاسب الآلي	تكنولوجيا الاتصالات والمعلومات	3.72	5.0	موظف	دوام كامل	Yes
3	23.0	أنثى	منطقة الرياض	الرياض	PCRF	PCRF	تقليدية	متقدم	0	2023-03-19	...	5.0	نعم	البكالوريوس	الاحياء	تكنولوجيا الاتصالات والمعلومات	4.47	5.0	غير موظف	دوام كامل	Yes
4	30.0	أنثى	منطقة الرياض	الرياض	TOSL	SWPS	داعمة	متقدم	0	2023-11-12	...	33.0	نعم	الدبلوم	تقنية المعلومات	تكنولوجيا الاتصالات والمعلومات	4.46	5.0	غير موظف	دوام كامل	No
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
813	36.0	ذكر	منطقة الرياض	الرياض	GRST	INFA	تقليدية	متوسط	0	2023-08-13	...	5.0	نعم	البكالوريوس	علوم حاسب	تكنولوجيا الاتصالات والمعلومات	2.55	5.0	موظف	دوام كامل	Yes

Conclusion

The Conclusion Of This Thesis

In 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 Conclusion Of This Thesis

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

THANKS