National College of Ireland

Predictive Analytics for Enhancing Student Success in the UK: A Machine Learning Approach

MSc Research Project
MSc Data Analytics

Oluwadamilare Adetuberu
Student ID: x18165125

School of Computing
National College of Ireland

Supervisor: Dr Anu Shani

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Oluwadamilare Adetuberu……

Student ID: x18165125……..……

Programme: MSc Data Analytics…… **Year:** 2023……..

Module: Research Project…….……

Supervisor: Dr Anu Shani…….……

Submission Due Date: 31 /1/24…….……

Project Title: Predictive Analytics for Enhancing Student Success in the UK: Machine Learning Approach…….……

Word Count: 798…… **Page Count**…8……..……..

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……

Date: …31/1/24……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the on-line project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Predictive Analytics for Enhancing Student Success in the UK: A Machine Learning Approach
Configuration Manual

Oluwadamilare Adetuberu
Student ID:x18165125

1 Introduction

The aim of this configuration manual is to enable the replication of the Predictive Analytics for Enhancing Student Success in the UK: A Machine Learning Approach project.
The project involves the building, test and evaluation of the following machine learning models (Decision Tees, Random Forest, Gradient Boosting Machine and Logistic Regression) to predict student results based on data from an online learning environment.

2 Computational Resources

HP Pavilion Laptop 15-eg0xxx
Processor: 11th Gen Intel(R) Core (TM) i7-1165G7 @ 2.80GHz, 2803 MHz, 4 Core(s), 8 Logical Processor(s)
Installed Physical Memory (RAM) 20.0 GB
System Type x64-based PC
OS Name: Microsoft Windows 11 Home
IDE: Google Colab and Jupyter notebook.
Programming language: Python
Python libraries required:

```
import pandas as pd
import numpy as np
import io
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.impute import SimpleImputer
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score,
recall_score,f1_score, roc_auc_score, log_loss, roc_curve
```

from sklearn.model_selection import cross_val_score

3 Dataset

The OULAD dataset contains records of 32,592 students, over 10 million rows for VLE data, 22 different courses, and the data available as 7 separate CSV files, it was obtained from the Open University Learning Analytics website (Kuzilek, et al., 2017).

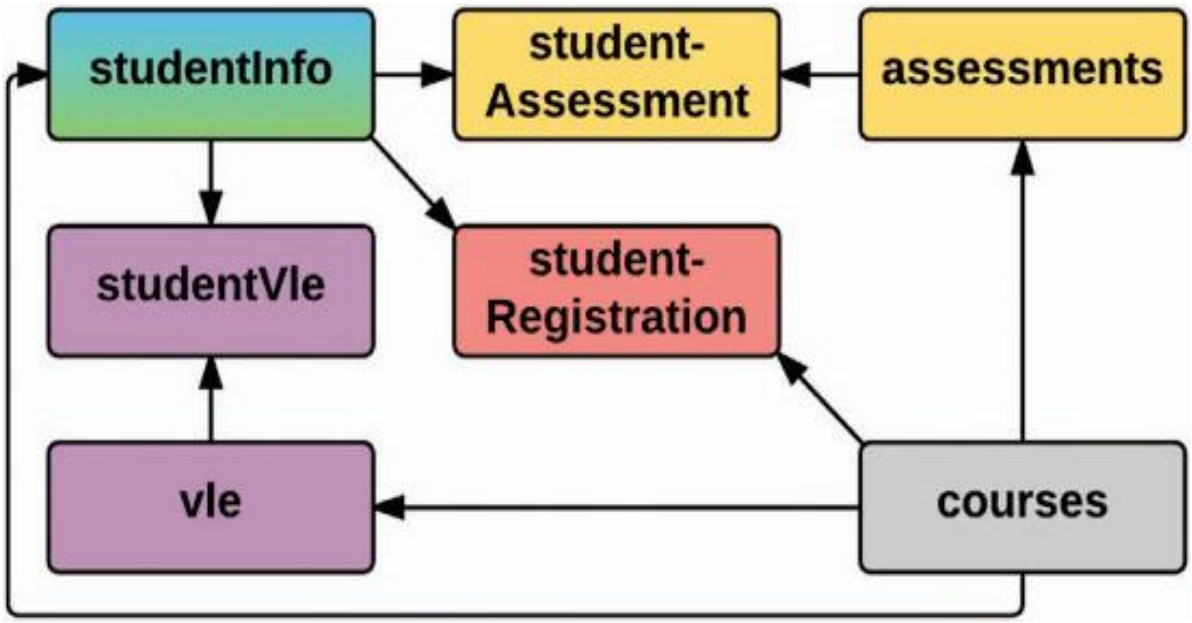

Fig1. An entity relationship diagram of the dataset (Kuzilek, et al., 2017)

4. Data Preprocessing

The datasets were loaded into the Python environment as follows:

```
# Load the studentInfo CSV file into a DataFrame
studentInfo = pd.read_csv("/Users/darea/Downloads/anonymisedData/studentInfo.csv")
print(studentInfo.info())

# Loading Student Registration file
studentRegistration = pd.read_csv("/Users/darea/Downloads/anonymisedData/studentRegistration.csv")
print(studentRegistration.info())

# Loading the Courses file
courses = pd.read_csv("/Users/darea/Downloads/anonymisedData/courses.csv")
print(courses.info())

# Loading the Assessment file
assessments = pd.read_csv("/Users/darea/Downloads/anonymisedData/assessments.csv")
print(assessments.info())

# Loading Student Assessment file
studentAssessment = pd.read_csv("/Users/darea/Downloads/anonymisedData/studentAssessment.csv")
print(studentAssessment.info())
```

Due to computing capacity limitations in processing the large dataset, a stratified sample of the largest datafile (StudentVle with over 10 million records was used to create a merge of all the seven datafiles as follows:

```
# Loading Student Vle file the largest with over 10 million records
studentVle = pd.read_csv("/Users/darea/Downloads/anonymisedData/studentVle.csv")
print(studentVle.info())

# Creating a stratified sample representative of the dataset by code_module
samplesize = 0.1

# creating stratifying column
stratifingcolumn = 'code_module'

# stratified sampling
stratifiedstudentVle = studentVle.groupby(stratifingcolumn, group_keys=False).apply(lambda x: x.sample(frac=samplesize))
print(stratifiedstudentVle.info())
```

A further stratified sample of the merged dataset containing over one million records was created and used for this analysis as follows:

```
# Creating a stratified sample representative of the Merged dataset by code_module
samplesize = 0.1

# creating stratifying column
stratifybycolumn = 'code_module'

# stratified sampling
stratifiedMergedata = Mergedata1.groupby(stratifybycolumn, group_keys=False).apply(lambda x: x.sample(frac=samplesize))
print(stratifiedMergedata.info())
# writing stratified merged data to file as Mergedata2
Mergedata2 = stratifiedMergedata
Mergedata2.to_csv('Mergedata2.csv', index=False)
```

Data Cleaning: missing numerical and categorical data were filling as follows. The categorical variables were converted to numerical for the purposes of classification analysis. The numerical values were transformed to scale. Finally, the dataset was split into training and testing data while stratifying the minority class to address class imbalance.

```
# Fill missing numerical data with median
num_imputer = SimpleImputer(strategy='median')
num_cols = X.select_dtypes(include=['int64', 'float64']).columns
X[num_cols] = num_imputer.fit_transform(X[num_cols])

# Fill missing categorical data with most frequent category
cat_imputer = SimpleImputer(strategy='most_frequent')
cat_cols = X.select_dtypes(include=['object']).columns
X[cat_cols] = cat_imputer.fit_transform(X[cat_cols])

# Convert categorical variables into numerical ones
encoder = LabelEncoder()
for col in cat_cols:
    X[col] = encoder.fit_transform(X[col])

# Scale numerical features
scaler = StandardScaler()
X[num_cols] = scaler.fit_transform(X[num_cols])

# Split the data into a training set and a test set, stratifying y to addres
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran

X_train.head()
```

5. **Training and Testing the model the four models.**

The models were initialised and ran through a loop to generate the prediction evaluation metrics for each model as follows:

```
#Training and Evaluating the Model
# Initialize the models
log_reg = LogisticRegression(solver='saga', max_iter=1000, random_state=42)
dec_tree = DecisionTreeClassifier(max_depth=5, min_samples_leaf=4, random_state=42)
rand_forest = RandomForestClassifier(class_weight='balanced', n_estimators=10,
                                     max_features='auto', max_depth=3,
                                     min_samples_leaf=1, random_state=42)
gbm_model = GradientBoostingClassifier(learning_rate=0.1, n_estimators=10,
                                       max_depth=3, min_samples_leaf=1,
                                       subsample=1.0, random_state=42)

# Creating a list of models
models = [log_reg, dec_tree, rand_forest, gbm_model]

# Now iterating over the list of models
for model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)  # Make predictions
    y_pred_proba = model.predict_proba(X_test)[:, 1]  # Probability for ROC AUC

    # Checking the classes in y_test
    if len(np.unique(y_test)) > 1:
        roc_auc = roc_auc_score(y_test, y_pred_proba)
    else:
        roc_auc = "Not Defined"

    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred, zero_division=0)
    f1 = f1_score(y_test, y_pred)
    logloss = log_loss(y_test, y_pred_proba) if 'roc_auc' != "Not Defined" else "Not Defined"

    print(f"Results for {model.__class__.__name__}:")
    print("Accuracy:", accuracy)
    print("Precision:", precision)
    print("Recall:", recall)
    print("F1 Score:", f1)
    print("ROC AUC Score:", roc_auc)
    print("Log Loss:", logloss)
    print("\n")
```

6. **Model Evaluation**

The visualisations of each model result and the final comparative result table is as follows:

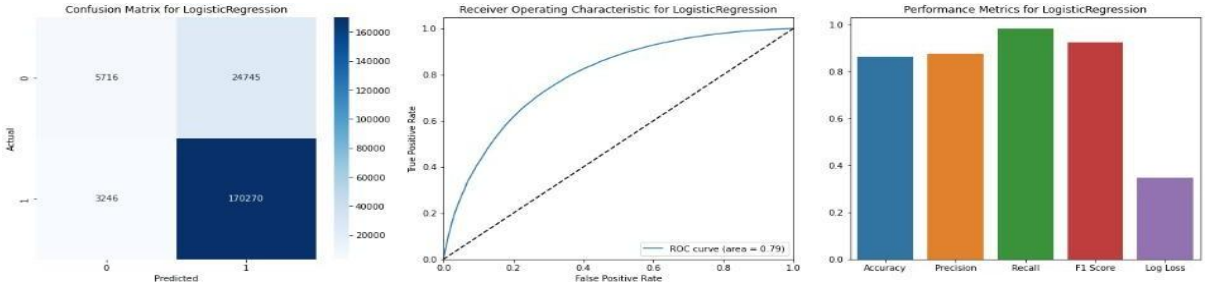

Fig 2. Logistic regression classifier evaluation visualization.

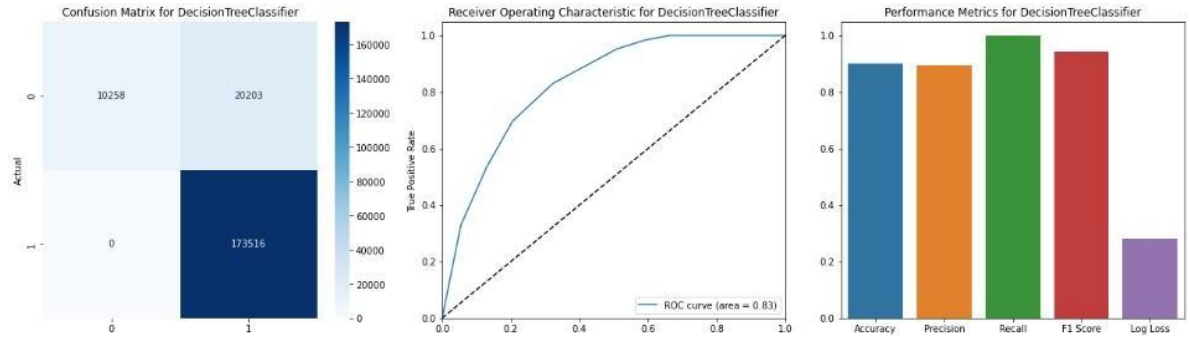

Fig3. Decision Tree Classifier evaluation visualization

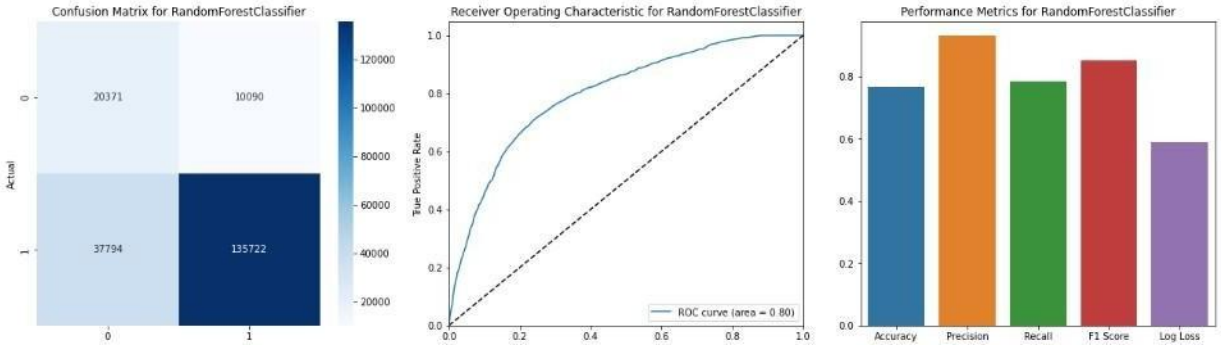

Fig 4. Random Forest Classifier evaluation visualization

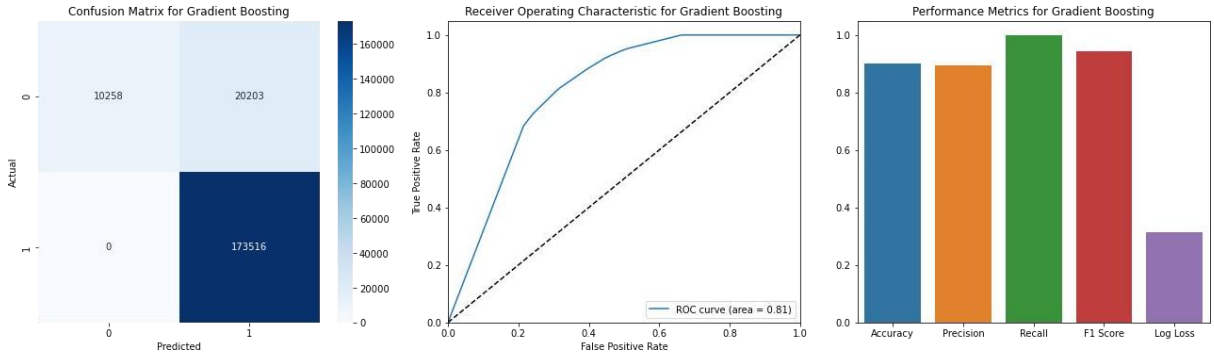

Fig 5. Gradient Boosting Classifier evaluation visualization

	Logistic Regression	Decision Trees	Random Forest	Gradient Boosting Classifier
Accuracy	0.863	0.901	0.765	0.901
Five-fold Cross validation average score	0.863	0.901	0.770	0.901
Precision	0.873	0.895	0.930	0.895
F1- Score	0.924	0.944	0.850	0.944
Recall	0.981	1.0	0.782	1.0

ROC AUC Score	0.790	0.831	0.798	0.808
Log Loss	0.345	0.280	0.589	0.312

Fig 6. Model Performance Result Table

7. Conclusion

In conclusion, the steps and code snippets outlined in this configuration manual can be applied to replicate the research.

References

Kuzilek, J., Hlosta, M. & Zdrahal, Z., 2017. *Kuzilek, J., Hlosta, M. and Zdrahal, Z. (2017) 'Data Descriptor: Open University Learning Analytics dataset',.* [Online]
Available at: https://analyse.kmi.open.ac.uk/open_dataset
[Accessed 2 August 2023].