ReadME File for Student Dropout Prediction Challenge F23

HUDK 4054 Individual Assignment #2

Project Information

- **Title:** Student Dropout Prediction Challenge F23
- **Creator:** Minxi Wang
- **ORCID iD:** 0009-0004-1363-187X
- **Affiliation:** Teachers College, Columbia University
- **Research Date:** 10-30-2023

Project Overview

The goal of this project is to use machine learning algorithms to predict undergraduate student outcomes through the analysis of a large dataset that include financial data, academic progress data, and static data. Our methodical approach to data integration, cleaning, and analysis seeks to identify the main factors that influence student performance and dropout risk. The F-Beta score, a performance metric that strikes a compromise between recall and precision, will be used to assess the initiative's progress and determine how accurate our dropout predictions were.

Deep Dive into the Dataset

The dataset is a comprehensive collection of data points covering aspects of student progress, financial aid, and static background information across multiple academic years:

- **Progress Data:** Tracks students' academic progress over terms by reflecting academic activities such as course enrollments, grades, credits, and learning objectives.
- **Financial Data:** Details on scholarships, loans, grants, and work-study programs.
- **Static Data:** includes demographic data such age, gender, race/ethnicity, educational history, GPA from high school, and previous college enrollment. Because it remains constant over time, this data set is static.

Python Libraries Used

In [1]:
```
import os
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, mean_squared_error
from sklearn.preprocessing import LabelEncoder, StandardScaler, PolynomialFeatures
from sklearn.impute import KNNImputer
from sklearn.feature_selection import VarianceThreshold, SelectKBest, chi2, mutual_
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier, BaggingClassifier, GradientBoo
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression, LinearRegression, Ridge
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from xgboost import XGBClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb
import matplotlib.pyplot as plt
from scipy import stats
```

Data Cleaning and Preparation

- Data cleansing was conducted on three distinct datasets involving the removal of columns with over 40% missing values, the substitution of -1 values with NaN, and the exclusion of rows with negative values in the value columns. The column named ['With Leading ID'] was renamed to ['StudentID']. Additionally, the variables for ['Marital Status', 'gradelevel', 'housing'] were converted into numerical codes to transform them into categorical variables, facilitating better analysis. Financial data, previously segmented by academic years and categories, was consolidated by summing the values within each category, resulting in new variables that represent the total amounts for loans, scholarships, work-study programs, and grants for each academic year. These new variables were then integrated back into the original dataset, enhancing its structure for further analysis.
- Create a new column named ['TermID'] to store both the term and year information. This DataFrame will retain the first column, which represents unique ['StudentID'], along with the newly created ['TermID'] and the values from the ['TermGPA'] column.For each specified column in the list ['CompleteDevMath', 'CompleteDevEnglish', 'Complete1', 'Complete2', 'CompleteCIP1', 'CompleteCIP2', 'DegreeTypeSought'], the function will calculate both the total GPA and the count of terms. The resulting total and count for each column will be added to the new columns. These new columns will be named ['Total_ColumnName', 'Count_ColumnName'], respectively, where ['ColumnName'] is the name of the column being processed.

Data Labels

Data Label	Significance/Interpretation
TotalGPA	The cumulative grade point average across all terms for a student.

Data Label	Significance/Interpretation
GPATerms	The number of terms or semesters included in the TotalGPA calculation.
Total_CompleteCIP1	The total number of completed courses within a specific Classification of Instructional Programs (CIP) code.
Count_CompleteCIP1	The count of instances a student completed courses under the first CIP code.
Total_CompleteDevMath	The total number of developmental or remedial math courses completed by a student.
Count_CompleteDevMath	The number of terms in which the student completed at least one developmental math course.
Total_CompleteDevEnglish	The total number of developmental or remedial English courses completed.
Count_CompleteDevEnglish	The number of terms in which the student completed at least one developmental English course.
Total_Complete1	The total number of a specific set of courses or requirements completed, possibly the first in a sequence.
Count_Complete1	The count of terms in which the student completed courses or requirements from the set referred to as "Complete1".
Total_Complete2	Similar to Total_Complete1, but for a second set or sequence of courses or requirements.
Count_Complete2	The count of terms in which courses or requirements from the "Complete2" set were completed.
Total_CompleteCIP2	The total number of completed courses in a second CIP code category.
Count_CompleteCIP2	The count of instances a student completed courses under the second CIP code.
Total_DegreeTypeSought	The total number of degrees the student is pursuing, if applicable.
Count_DegreeTypeSought	The number of terms in which the student was actively pursuing a degree.
TotalLoan	The total amount of loan funding received by the student for educational purposes.
TotalScholarship	The cumulative amount of scholarship money awarded to the student.
TotalWork_Study	The total amount earned by the student through work-study programs during their education.
TotalGrant	The total amount of grant money received, which, unlike loans, does not need to be repaid.

Analytical Approach

- **Data Preprocessing:** I approached categorical and numerical data in distinct ways: Mean imputation and standard scaling were used to numerical characteristics, while one-hot encoding was used to fill categorical variables with the most common values. After that, I used TestIDs and Dropout labels to select datasets for Kaggle testing and training. After this configuration, operations were combined and the inputs (X) and outputs (Y) for the models were specified. I separated the data after preprocessing to produce separate training and test sets. The best-performing model was subsequently chosen after we tested several classifiers on the validation set. This model was then used to forecast results on the Kaggle test data, resulting in an output of a structured DataFrame.

- **Feature Engineering:** I designed features based on my investigation to represent the students' socioeconomic and academic attributes. Using significance scores, I trained a Decision Tree with a fixed seed to discover important predictive variables and ensure reproducibility. The dataset was refined by using a Random Forest for feature selection and LDA to reduce dimensionality. I also added Polynomial Features and Interactive Feature Generation to the data, which improved the models' predictive accuracy (as indicated by the F-Beta score) for student dropout predictions.
- **Model Selection and Evaluation:** Multiple models were tested for their predictive accuracy, including Logistic Regression, Random Forest, XGBoost, KNN, and LDA. The models were evaluated based on accuracy, precision, recall, and F1 scores to select the best performer for predicting student outcomes. The analysis revealed XGBoost, augmented by Polynomial Features, as the standout performer, achieving an impressive accuracy of 0.9604565837749695, showcasing its superior predictive capability.

```
DecisionTree XGBoost Accuracy: 0.9429270281288219
LogisticsRegression XGBoost Accuracy: 0.9584182633509988
LDA XGBoost Accuracy: 0.9368120668569099
RandomForest XGBoost Accuracy: 0.9584182633509988
PCA XGBoost Accuracy: 0.7594781899714635
SelectFromModel XGBoost Accuracy: 0.9551569506726457
PolynomialFeatures XGBoost Accuracy: 0.9604565837749695
InteractiveFeatureGeneration XGBoost Accuracy: 0.9584182633509988
Best Feature Set for XGBoost: PolynomialFeatures with Accuracy: 0.9604565837749695
```

Challenges and Solutions

- **Data Integration:** The challenge of merging datasets from various sources was met by developing a consistent schema and using Python scripts for automated merging and cleaning.
- **Model Selection:** The diverse nature of the data necessitated the testing of multiple models. Cross-validation and grid search techniques were employed to identify the models that best fit our data.
- **Model Accuracy Improvement:** Addressing low model accuracy was a key challenge. Initially, hyperparameter adjustments provided some improvement, but the real breakthrough came from using cross-validation to reveal overestimations of model performance. This led to a deeper dive into data processing and cleaning. Analyzing variable relationships and enhancing data relevance through thorough cleaning significantly improved model accuracy. Exploring multiple solutions, such as data transformation and optimizing variable correlations, was crucial in solving this problem.

Data Storage and Access

- **Local and Cloud Storage:** Data is finally stored both locally and on the cloud platform as .csv files to ensure accessibility and security. Cloud storage solutions provide a centralized location for team members to access and collaborate on data.
- **Data Access:** Access to sensitive data was controlled through secure authentication mechanisms, ensuring that only authorized team members could access the information

for analysis purposes.

Conclusion

The value of meticulous data preparation and the potential of machine learning in educational research are demonstrated by this undertaking. Our analysis yields actionable insights that can guide actions aimed at improving student retention rates by identifying critical factors influencing student outcomes.

Final Oberservation

- **Which metadata standard did you choose and why?** The primary reason I selected the Data Documentation Initiative (DDI) as the metadata standard for ReadME files is because it offers an extensive metadata management framework that is necessary to streamline and standardize data description efforts.
- **Which template/software did you use?** I chose to use Python to generate the Markdown files because I found the operation of Python easier and more familiar to me compared to Rstudio. However, in the process of converting Markdown to PDF, I ran into some typographical issues. The process seems to first convert Markdown to HTML and then convert it to PDF, which results in a less-than-ideal final PDF layout.
- **What was the most challenging part of creating a ReadME file? How did you overcome these obstacles?** Enhancing the visual appeal of a ReadME file can indeed be challenging, especially when incorporating images and tables. The issue with images in Jupyter Notebooks is a common one; they often won't display correctly until converted to an HTML format. For tables, Markdown requires manual formatting which can be cumbersome and prone to errors. My solution was to convert the files to HTML to view the images and adjust the Markdown table formatting by adding colons in the syntax to align the text as desired. These adjustments helped me achieve a clear and visually engaging presentation in the ReadME file.