Predicting academic success of undergraduate students

by Catherine Meng, Jinxun Cheng, Jingyuan Wang, Siddarth Subrahmanian 2024/12/23

```
In [1]: import pandas as pd
        import altair as alt
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import StandardScaler, OneHotEncoder
        from sklearn.compose import make_column_transformer
        from sklearn.pipeline import make_pipeline
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.model_selection import RandomizedSearchCV
        from scipy.stats import loguniform
        from sklearn import set_config
        set_config(display='text')
```

Summary

In this analysis, we attempt to build a classification model using the k-nearest neighbors algorithm to predict student dropout and academic success based on information available at enrollment (including academic path, demographics, and socio-economic factors). Our final classifier performed consistently on unseen test data, achieving a cross-validation training score of 0.71, with a similar test score. Although the model's accuracy is moderate, it performs consistently. Given that the data was collected from a single institution, a larger dataset may be necessary to generalize predictions to other institutions or countries. We believe this model is close to supporting dropout prediction for the institution from which the data was collected, though further research to improve performance and better understand characteristics of incorrectly predicted students would still be beneficial.

Introduction

Higher education institutions worldwide face the ongoing challenge of academic dropout and student failure, which affect not only individual students' futures but also the institution's reputation and resources. The ability to predict and anticipate students' potential difficulties is valuable not only for supporting individual students in achieving their goals but also for institutions aiming to implement strategies that support and guide students who may be at risk of academic failure or dropout.

The goal of this analysis is to help reduce academic dropout and failure in higher education by applying machine learning techniques to identify at-risk students early in their academic journey, enabling institutions to implement targeted support interventions.

Methods

Data

The data set is created by Mónica Vieira Martins, Jorge Machado, Luís Baptista and Valentim Realinho at the Instituto Politécnico de Portalegre (MV,Martins, D. Tolledo, J. Machado, L. M.T. Baptista, V.Realinho. 2021). It is sourced from UC Irvine's Machine Learning Repository and can be found here. The data contains demographic, enrollment and academic (1st and 2nd semesters) information on the students. Each row in the data set represents a student record. Using these data, a model would be built to predict the academic outcome of the student. There are 36 columns in total.

Analysis

The k-nearest neighbors (k-nn) algorithm was used to build a classification model to predict whether a student is at risk of dropping out. All variables included in the original data set, with the exception of the Course, Nacionality, Gender, Unemployment rate, Inflation rate, GDP, Previous qualification, Mother qualification Mother occupation, Father qualification, Father occupation columns were used to fit the model. Data was split with 80% being partitioned into the training set and 20% being partitioned into the test set. The hyperparameter K was chosen using 5-fold cross validation. All numeric features were standardized just prior to model fitting. We knew the range of features as they are because they all have integer data type.

Results & Discussion

To look at whether each of the predictors might be useful to predict the tumour class, we plotted the distributions of each predictor from the training data set and coloured the distribution by class (Dropout, Stay, Enrolled, orange, and Graduate, red).

```
In [2]: # Import and split data
        df = pd.read_csv('../data/raw/data.csv', delimiter = ';')
        train, test = train_test_split(df, train_size = 0.8, random_state = 123)
```

```
In [3]: train.nunique()
```

```
Out[3]: Marital status                                 6
        Application mode                              17
        Application order                              7
        Course                                        17
        Daytime/evening attendance\t                   2
        Previous qualification                        17
        Previous qualification (grade)               101
        Nacionality                                   21
        Mother's qualification                        29
        Father's qualification                        34
        Mother's occupation                           31
        Father's occupation                           44
        Admission grade                              597
        Displaced                                      2
        Educational special needs                      2
        Debtor                                         2
        Tuition fees up to date                        2
        Gender                                         2
        Scholarship holder                             2
        Age at enrollment                             46
        International                                  2
        Curricular units 1st sem (credited)           22
        Curricular units 1st sem (enrolled)           22
        Curricular units 1st sem (evaluations)        36
        Curricular units 1st sem (approved)           20
        Curricular units 1st sem (grade)             655
        Curricular units 1st sem (without evaluations)  11
        Curricular units 2nd sem (credited)           23
        Curricular units 2nd sem (enrolled)           25
        Curricular units 2nd sem (evaluations)        29
        Curricular units 2nd sem (approved)           21
        Curricular units 2nd sem (grade)             605
        Curricular units 2nd sem (without evaluations)  10
        Unemployment rate                              9
        Inflation rate                                 8
        GDP                                           10
        Target                                         3
        dtype: int64
```

```
In [4]: # Remove extra '\t' from the column name
        train.rename(columns = {"Daytime/evening attendance\t" : "Daytime/evening attendance"}, inplace = True)
        test.rename(columns = {"Daytime/evening attendance\t" : "Daytime/evening attendance"}, inplace = True)

        # Remove / from column name to prevent issues with Altair errors
        train.columns = train.columns.str.replace("/", "_", regex=False)
        test.columns = test.columns.str.replace("/", "_", regex=False)
```

```
In [5]: # Group feature types based on feature description from source data
        categorical_features = ["Application order", "Course", "Nacionality", "Gender",
                                "Marital status", "Application mode", "Daytime/evening attendance",
                                "Previous qualification", "Mother's qualification", "Father's qualification",
                                "Mother's occupation", "Father's occupation", "Displaced",
                                "Educational special needs", "Debtor", "Tuition fees up to date",
                                "Scholarship holder", "International"]

        numeric_features = ["Previous qualification (grade)", "Admission grade", "Age at enrollment",
                            "Curricular units 1st sem (credited)", "Curricular units 1st sem (enrolled)",
                            "Curricular units 1st sem (evaluations)", "Curricular units 1st sem (approved)",
                            "Curricular units 1st sem (grade)", "Curricular units 1st sem (without evaluations)",
                            "Curricular units 2nd sem (credited)", "Curricular units 2nd sem (enrolled)",
                            "Curricular units 2nd sem (evaluations)", "Curricular units 2nd sem (approved)",
                            "Curricular units 2nd sem (grade)", "Curricular units 2nd sem (without evaluations)",
                            "Unemployment rate", "Inflation rate", "GDP"]
```

```
In [6]: # Plot numerical features
        def plot_numerical(col, col_encode):
            p = alt.Chart(train).encode(
                x = alt.X(col, title=col, bin=alt.Bin(maxbins=30)),
                y = alt.Y('Target', title=None),
                color = alt.Color('Target', legend=alt.Legend(orient="top")
            ).properties(
                width = 130,
                height = 100
            ).repeat(
                numeric_features,
                columns = 2
            )
        plot_numerical('Target', 'Target')
```

```
Out[6]:
```


Figure 1. Distribution of Numerical Variable per Academic Outcome

```
In [7]: # Plot categorical and boolean features
        alt.Chart(train).mark_bar().encode(
            x = alt.X(column, type='quantitative'),
            y = 'count()',
            column = alt.Column('Target', title = None),
            row = alt.Row(column, legend = None)
        ).properties(
            width = 120,
            height = 100
        ).resolve_scale(
            y = 'independent'
        ).repeat(
            categorical_features,
            columns = 1
        )
```

```
Out[7]:
```


Figure 2. Distribution of Categorical Variable per Academic Outcome

```
In [8]: # Features to drop from model
        drop_features = ['Nacionality', "Gender", "Unemployment rate",
                         "Inflation rate", "GDP", "Previous qualification", "Mother qualification",
                         "Mother occupation", "Father qualification", "Father occupation"]
```

Reason for dropping:

- Course: only captures 17 different courses and does not generalize the real world well.
- Nacionality and Gender are removed to avoid racial and gender bias
- Unemployment rate, Inflation rate, and GDP are continuous values, they each have less than 10 unique values out of 3000+ rows. This doesn't provide enough range to generalize the problem.
- In Figure 3, Previous qualification, Mother qualification, Mother occupation, Father qualification and Father occupation have cluster patterns. This cluster pattern represents since the ranking of education levels are arbitrary. E.g. 1- Higher Education - Doctorate' is ranked higher than '1- Secondary Education' but lower than '10 - 11th Year of Schooling - Not Completed'. The source data website provides

```
In [9]: X_train = train.drop(columns=['Target'])
        y_train = train['Target']
        X_test = test.drop(columns=['Target'])
        y_test = test['Target']

        # Make preprocessor
        preprocessor = make_column_transformer(
            (StandardScaler(), numeric_features),
            ('drop', drop_features)
        )
```

```
In [10]: # Build the pipeline, use knn to train the model

         my_pipeline = make_pipeline(
             preprocessor,
             KNeighborsClassifier(n_neighbors=5)
         )

         my_pipeline.fit(X_train, y_train)
```

```
Out[10]: Pipeline(steps=[('columntransformer',
                          ColumnTransformer(transformers=[('standardscaler',
                                                           StandardScaler(),
                                                           ['Previous qualification '
                                                            '(grade)',
                                                            'Admission grade',
                                                            'Age at enrollment',
                                                            'Curricular units 1st sem '
                                                            '(credited)',
                                                            'Curricular units 1st sem '
                                                            '(enrolled)',
                                                            'Curricular units 1st sem '
                                                            '(evaluations)',
                                                            'Curricular units 1st sem '
                                                            '(approved)',
                                                            'Curricular units 1st sem '
                                                            '(grade)',
                                                            'Curricular units 1st sem '
                                                            '(without evaluations)',
                                                            'Curricular units 2nd sem '
                                                            '(credited)',
                                                            'Curricular units 2nd sem '
                                                            '(enrolled)',
                                                            'Curricular units 2nd sem '
                                                            '(evaluations)',
                                                            'Curricular units 2nd sem '
                                                            '(approved)',
                                                            'Curricular units 2nd sem '
                                                            '(grade)',
                                                            'Curricular units 2nd sem '
                                                            '(without evaluations)',
                                                            'Unemployment rate',
                                                            'Inflation rate', 'GDP'],
                                                           ['Application order',
                                                            'Course', 'Nacionality',
                                                            'Gender',
                                                            'Unemployment rate',
                                                            'Inflation rate', 'GDP',
                                                            'Previous qualification',
                                                            'Mother qualification',
                                                            'Mother occupation',
                                                            'Father qualification',
                                                            'Father occupation']),
                                                          ('drop', 'drop',
                                                           ['Application order',
                                                            'Course', 'Nacionality',
                                                            'Gender',
                                                            'Unemployment rate',
                                                            'Inflation rate', 'GDP',
                                                            'Previous qualification',
                                                            'Mother qualification',
                                                            'Mother occupation',
                                                            'Father qualification',
                                                            'Father occupation']),
                                                          ('remainder', 'drop',
                                                           ['Marital status',
                                                            'Application mode',
                                                            'Daytime/evening '
                                                            'attendance',
                                                            'Displaced',
                                                            'Educational special '
                                                            'needs',
                                                            'Debtor',
                                                            'Tuition fees up to '
                                                            'date',
                                                            'Scholarship holder',
                                                            'International'])])),
                         ('kneighborsclassifier',
                          KNeighborsClassifier())])
```

```
In [11]: # Use RandomizedSearchCV to tune hyperparameters

         parameter_distributions = {
             'kneighborsclassifier__n_neighbors': randint(1, 30)
         }

         random_search = RandomizedSearchCV(
             estimator=my_pipeline,
             param_distributions=parameter_distributions,
             n_iter=10,
             cv=5,
             scoring='accuracy',
             random_state=2,
             n_jobs=-1
         )

         random_search.fit(X_train, y_train)

         print("Best Parameters:", random_search.best_params_)
         print("Best CV Accuracy:", random_search.best_score_)
```

```
Best Parameters: {'kneighborsclassifier__n_neighbors': 17}
Best CV Accuracy: 0.7141346153846154
```

We utilized the KNN to train the dataset and employed RandomizedSearchCV to fine-tune the hyperparameters. Based on the results, the optimal hyperparameter value is k=12, achieving a best cross-validation score of 0.71. Using this value, we retrained the model and evaluated its performance on the test set, obtaining a final test score of 0.71.

```
In [12]: my_pipeline_best = make_pipeline(
             preprocessor,
             KNeighborsClassifier(n_neighbors=12)
         )

         my_pipeline_best.fit(X_train, y_train)

         test_score = my_pipeline_best.score(X_test, y_test)
         print("Test accuracy: {test_score}")
```

```
Test accuracy: 0.7139417431424519
```

References

1. Martins, M. V., D. Tolledo, J. Machado, L. M. T. Baptista, and V. Realinho. "Early Prediction of Student's Performance in Higher Education: A Case Study." In Trends and Applications in Information Systems and Technologies, vol. 1, Advances in Intelligent Systems and Computing series. Springer, 2021. https://archive.ics.uci.edu/dataset/697/predict+students+dropout+and+academic+success. DOI: 10.1007/978-3-030-72657-7_16.

2. Scikit-learn Developers. "Scikit-learn: Machine Learning in Python – API Reference Documentation." Accessed November 23, 2024. https://scikit-learn.org/dev/api/index.html.

3. Vega-Altair Developers. "Altair User Guide." Accessed November 23, 2024. https://altair-viz.github.io/user_guide/data.html#.

4. Timbers, T., J. Cleblond, and M. Lee. "Predicting Breast Cancer from Digitized Images of Breast Mass." Accessed November 21, 2024. https://github.com/ttimbers/breast-cancer-predictor/blob/0.0.1/notebooks/breast_cancer_predictor_report.ipynb.