NOVA IMS
Information Management School

WIZARDRY SCHOOL ENROLLEMENT

MACHINE LEARNING I

AFONSO CADETE 20211519
LAURA MATIAS 20221836
MARTA ALMENDRA 20221878
MATILDE CASIMIRO 20221940
TERESA SIMÃO 20221873

GROUP 11
2023/2024

Abstract

In the realm of wizardry education, predicting student admissions is of paramount importance. This study employs supervised machine learning techniques to forecast admission into a revered wizard school. Leveraging a dataset of 1' attributes encompassing gender, experience level, family background, program preference, and other pertinent criteria, we aimed to construct a predictive model critical for evaluating student qualifications.

Our analysis utilized two distinct datasets: a training set comprising approximately 713 observations across 11 features, including admission status, and a test set with 176 observations lacking admission information. Our primary objective centered on predicting student acceptance using the model developed from the training dataset, with the ground truth value ("0" or "1").

Implementing various machine learning algorithms with diverse hyperparameter combinations, our study revealed the hypertuned KNeighbors Classifier as the optimal model. Model evaluation based on binary f1-scores indicated promising results, with the best model achieving 76% on the training set and nearly 74% on the validation set.

Despite challenges posed by an unbalanced problem and limited observations in the test dataset, our findings shed light on the predictive capabilities of our model. This research not only aids in admissions forecasting but also offers insights into the intricate criteria valued by wizarding institutions, paving the way for informed decision-making in evaluating prospective wizarding talents.

Keywords

Machine Learning; Supervised Learning; Feature Selection; F1-Score; K-Neighbors

Introduction

In a world of magic and power, every young wizard and witch desires to attend the most prestigious magical institutions to fully unlock the potential of their abilities.

As the school year begins and the doors to these esteemed schools open, our objective is to precisely identify the candidates worthy of studying there. In order to do so, our group is tasked with the development of a predictive machine learning model.

We were provided two distinct datasets, one with the purpose of training the model and the other of testing it. The train set contained information about 713 students who had previously been accepted or denied admission to the magic school, while the test set contained information about 176 students whose admittance to school was unknown. Both the data samples were composed of ten attributes for each students, which included details regarding their previous educational path, personal magical preferences and family situations, among others, and the train set also included a binary outcome feature.

In the course of this project, we applied a variety of machine learning techniques and methods to develop a predictive model, using the qualifications and magical potential of prospective witches and wizards to determine which students from the test dataset are eligible for admission and to ensure that only those most deserving pass through the gates of wisdom and power.

This report describes all the steps taken to create the model that best predicts whether a student will be accepted into a school or not.

Background

We intended to execute this project in the best and most efficient way possible, and so, in addition to the strategies and techniques provided throughout the Machine Learning I course, we implemented a few methods whose knowledge of was obtained through other sources.

Extra Trees Classifier

This feature selection technique is an ensemble learning method that belongs to the family of tree-based models, as it involves using decision tree models to assess the importance of features in a dataset.
The first step is to build multiple trees, where the splitting criteria for each feature is chosen randomly, without searching for the optimal split. This introduces additional randomness, which theoretically makes the trees more diverse. The importance score is then computed by measuring the average decrease in impurity caused by each feature when making splits across all the trees in the ensemble, and by examining these scores, it's possible to identify which features are more influential in making predictions.
This method allows defining the number of trees that are built and setting a threshold to select a subset of the most features (features with scores below the threshold can be filtered out). It's also a particularly useful strategy when working with large and complex datasets, since the randomness introduced in the tree-building process helps reduce overfitting.

Variance Inflation Factor (VIF)

The VIF is a statistical measure that assesses the degree of multicollinearity among predictor variables. Multicollinearity occurs when two or more features are highly correlated, making it difficult to separate their individual contributions and influence on the outcome.
The way this selection method works is by building a regression model for each variable in the dataset, where that one is defined as the dependent feature and all the remaining as the independent features. The VIF value of each features is then calculated as the ratio of the variance of the coefficient estimate in the full model to the variance of the coefficient estimate if that feature were fit alone.
A ratio of 1 indicates no multicollinearity and greater values indicate increasing levels of multicollinearity. If the VIF value of a variable is greater than 5, that indicated a significant level of multicollinearity, and removing it will probably improve the performance of the model.
Usually, the Variance Inflation Factor is used in conjunction with other feature selection techniques, rather than as a standalone method.

Principal Component Analysis (PCA)

PCA is an unsupervised approach for reducing the dimensionality of data comprised of a high number of inter-related variables while maintaining as much variation in the original data as possible. Each feature in the data is treated as a distinct dimension, and the PCA algorithm converts data features into a new collection of features known as Principal Components.
The first principal component is calculated to account for the largest amount of variance in the original features. The second component, which is orthogonal to the first, makes up the majority of the remaining variation.
The number of main components that PCA theoretically generates equals the number of features in the training dataset; however, in actual use, not all of them are retained. Taking just a few of the initial principle components is sufficient to approximate the original dataset without requiring extra features, as each consecutive principal component explains the variance that remains after its preceding component.

Support Vector Machine (SVM)

The Support Vector Machine (SVM) is a well-known supervised learning algorithm that is widely used in machine learning for classification tasks. Its major purpose is to find the best hyperplane in an n-dimensional space, which will serve as a definitive border effectively separating different classes within the dataset.
In its pursuit of the best hyperplane, SVM identifies critical points known as support vectors, which have a substantial impact on the classification process. Using these support vectors allows the algorithm to accurately classify future data points.
Furthermore, SVM has a number of advantages. It performs effectively in high-dimensional areas, making it suitable for complex data. It achieves memory efficiency by exploiting a subset of training points known as support vectors in the decision-making process, and it demonstrates versatility by supporting various kernel functions.

Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP)

UMAP is a dimension reduction technique used in both machine learning and data analysis and it's particularly useful for visualizing high-dimensional data in a lower-dimensional space. This technique is based on a mathematical framework that involves constructing a low-dimensional representation of the data by modelling the relationships between data points in a way that preserves the underlying geometry of the data.

The algorithm is founded on three assumptions about the data

- The data is uniformly distributed on Riemannian manifold;
- The Riemannian metric is locally constant (or can be approximated as such);
- The manifold is locally connected.

METHODOLOGY

In this section, we outline the systematic and technical approach employed to achieve our research objectives. Our methodology encompasses data preprocessing, feature selection and the application of diverse machine learning algorithms, as well as the evaluation metrics chosen to assess model performance.
All of these phases were applied using python programming language and some of its available libraries.

1. Exploratory Data Analysis and Preprocessing

The foundation of any data-driven endeavor is preprocessing and exploratory data analysis. The raw data supplied was transformed and quality tested during this vital stage, which involved tasks such as dealing with missing values, outliers, and duplicates.
This approach also included examining the dataset's inherent patterns, correlations, and abnormalities. We employed visualizations such as histograms, bar charts, pie charts and correlation matrices to gain a better understanding of the data, which revealed insights that influenced later steps in the project.

1.1. Missing Values and Data Types

Following the initial stage of importing the dataset, we examined and evaluated the information in our dataset, with the Pandas built-in functions .info(), which returns each variable's data type and value count, and .describe(), which returns descriptive statistics on the numerical variables. The datatypes were verified and changed if the characteristics of the columns demanded so. In the case of columns with missing values, we calculated the percentage at which they occurred, since columns with more than half (>50%) of their data points missing should be removed to avoid any risk of incorrect influence when training the model later on.

1.2. Distribution and Outliers

Next, we analysed and built visualizations for our categorical and numerical features separately.
For the categorical variables, we created two bar charts per variable, comparing the frequency and proportion of each possible value and their admittance to school.
In the case of the numerical variables, we used mainly histograms, to better understand the distribution of the variables and possibly identify extreme univariate outliers.

1.3. Duplicates

The following step was to check for the existence of duplicate values in the dataset and if necessary delete them, as duplicate values may also influence the model incorrectly. This task was executed using the Pandas built-in function .duplicated().

1.4. Correlations

We employed Spearman's Correlation to built a correlation matrix. This helps, not only to discover which numerical variables hold greater power over the admittance of students to the school, but also to detect multicollinearity issues, which can impact the stability and interpretability of the model negatively.

1.5. Data Preparation

This preparation is the final step of the first phase of the project and the aim is to produce a clean and informative dataset to train the various models that will be employed and accurately predict the outcome of the students in the Test dataset provided.
This task began with a distinction between predictor and target features, where the target corresponds to the value we are trying to predict. At this stage, no partition method was applied to the dataset, since later on, to evaluate the models, the methods used included cross-validation, which provides a more reliable and comprehensive evaluation of a machine learning model compared to a simple hold-out method. Details regarding this technique are in the Model Evaluation section.
Furthermore, encoding our categorical variables was critical since the models can only deal with numerical data, and the method chosen to do so was OneHotEncoder from the scikit-learn python library, which transforms the categorical variables into binary columns representing the different categories.
Missing values were treated using KNN Imputer, a technique that takes multiple neighbour data points to estimate missing values, preserving the underlying relationships in the data. In this step, we decided to use MinMax technique to scale the data before imputing it in order to maintain the original distributions while applying the process.
Finally, the numerical parameters were then scaled using Standard Scaler, MinMax Scaler and Robust Scaler. All of these 3 techniques bring the numerical features to a comparable normalized size, preventing certain features from dominating simply because of their larger magnitude, which can improve algorithm performance. However, each scaler has distinct attributes and advantages, and might impact model performance differently, which is why we experimented and evaluated all instead of choosing one a priori.

2. Feature Selection

The next phase was the process of selecting the key parameters of our data and dropping those with least importance. If there's too many features, the model will collect irrelevant patterns and learn from variability, so it's essential to ensure that only the appropriate features are used to train an optimum model and avoid overfitting.
To ensure an efficient feature selection, we employed Filter Methods, Wrapper Methods and Embedded Methods.
Filter Methods don´t involve any machine learning algorithms and, instead, assess the relevance of individual features based on statistical or ranking criteria. The chosen Filter methods for this dataset were:

- **Constant Features:** features with no variability can be filtered out.
- **Chi-Square Score:** identifies the most relevant categorical features in the dataset by assessing the independence between each feature and the target variable through a chi-square test of independence. If a features has a low chi-square statistic, indicating independence, it may be filtered out.
- **Spearman Correlation:** selects features based on their spearman correlation with the target variable. Variables with low absolute values may be filtered out, as the relation between them and the outcome is weak.

Wrapper Methods, on the other hand, use the performance of a predictive model as the criterion for selecting the subset of key features and, consequently, delivers better results when the interaction between variables highly influences the outcome. Only one wrapper method was used to avoid excessive computational cost.

- **Exhaustive:** creates and evaluates all possible feature combinations subsets by building a model with each, then selects the subset with the best performance. This method can be computationally expensive, especially when we're working with a large number of features, but it guarantees that we find the best possible subset of features, and seeing as this dataset does not have a great quantity of variables, we decided we could afford it. To implement this method we used the LogisticRegression() and StratifiedKFold() functions from the sklearn.model_selection library.

Embedded Methods, unlike the prior two types, learn which features best contribute to the accuracy of the model while it is being created, which makes them computational efficient. Both these techniques were employed:

- **Lasso:** selects the most important features by introducing additional constraints into the optimization of a predictive algorithm and thus, biasing the model towards lower complexity. To implement this method we used the sklearn.linear_model.LassoCV() function.
- **Extra Trees:** uses decision tree models to identify and rank the importance of the features in the dataset. The process involves training multiple decision trees, and then selecting our most important features based on its contribution to the model's performance (detailed explanation in background). To implement this method we used the sklearn.ensemble.ExtraTreesClassifier() function.

Additionally, we applied a feature selection method that is not included in the previous groups of techniques:

- **Variance Inflation Factor:** measures the degree of multicollinearity among predictor variables and suggests removing features with high VIF values (detailed explanation in background).To implement this method we used the statsmodels.stats.outliers_influence.variance_inflation_factor() function.

Firstly, we separated the categorical features from the rest of the dataset and applied the chi-square method to select the most important ones. These results will be the same for all the different sets, since the scaling doesn't impact non-numerical features, and this way we manage to spare the computational expense of repeating this process for each set.
For each of the remaining methods, we developed a function to implement it and joined all of them in a bigger general function that returned the results of all the techniques applied to a dataset. Some functions required a more complex implementation, specifically the Exhaustive search. The model we chose to test on was Logistic Regression, which means we fit each possible combination of features to the model and tested its performance using stratified k-fold cross-validation, selecting as the best subset of features the one whose F1 score was greater. The only issue with this strategy is that each time it ran the results would be slightly different, since the cross-validation doesn't always divide the data into the same folds, so our function performs the exhaustive feature selection process multiple times (we decided on 100 times) and keeps track of how often each feature appears in the best subset across all iterations.

The idea is to identify which features are consistently selected as part of the best subsets. If a feature appears in less then 40 of the 100 subsets, the exhaustive method is discarding it. If it appears between 40 and 65 times, we classify it as a 'Keep(?)', which is a weak indication to keep the feature, and means the other methods will have more weight in the final call.
The general function was applied first to the non-scaled set and then to all the scaled sets (Standard Scaler, MinMax Scaler, Robust Scaler) and the results collected to make a final decision regarding each feature. The categorical features were included in this phase as Boolean datatype, which is also compatible with these selection strategies.
In order to make the final decision whether to keep or not each variable we considered the results from all the methods, but attributed a greater importance to VIF, Exhaustive and Lasso.

2.1. Principal Component Analysis

As mentioned in the project background, we tried Principal Component Analysis as an alternative preprocessing technique for our feature selection to see whether it performed better. This was achieved by using the sklearn.decomposition.PCA() method from the scikit-learn library on a fuction we created.
The major goal was to simplify the process of doing PCA on a dataset by encapsulating everything in a single function. It conducts PCA on the input dataset X_train, identifies and retains principle components that contribute to a variance threshold of 80%, and outputs a DataFrame containing these principal components for further analysis or modeling. It also provided visualizations to help determine the trade-off between reducing dimensionality and keeping a desired degree of information from the dataset.

3. Model Evaluation

Model evaluation is a critical aspect of our pipeline, encompassing techniques and metrics designed to assess the performance of the trained models through various aspects, such as accuracy, precision, recall, error rate and F1 score. The goal of this final phase is to select a leading model that strikes a balance between predictive power and generalization without overfitting to the train data.
To do so, a function was developed to incorporate all of the steps needed for model evaluation, which uses the sklearn GridSearchCV() method to print and provide the best hyperparameters and matching F1 score from the grid search applied to a model.
We assessed our models using F1 scoring since it is a metric that combines precision and recall, so it is critical in unbalanced cases, which, as previously stated, is the case with our dataset. This is an important aspect, as the goal of the project was to identify whether or not a student was admitted to the school, and false positives and negatives would have a detrimental influence on our predictions and therefore the core of the research.
In order to find out which model performed best with the given data and under what circumstances, we decided to evaluate different models: Logistic Regression, Decision Tree, K-Neighbors, Naive-Bayes, Multi-Layer Perceptron. The main motive for selecting these models was that each has different benefits and may be applied to various classification tasks based on the dataset and problem scenarios. More complex models, such as Random Forest, Support Vector Machine and Ada Boost, were also tested but they demand more computational power, so it was not viable to apply Grid Search and cross-validation techniques to them, and the main objective of the modelling part was to find an objective and generalizable solution.
The first model we evaluated was Logistic Regression, which was applied to all the differently scalled datasets with and without feature selection, with a combination of various parameters such as different estimator penalties, solvers, class weights, among others. With our model evaluating function, we were able to receive results that show the best parameters and matching binary F1-score for validayion of all these distinct logistic regressions.
For the subsequent models, we utilized the same concept of running the models with different parameters and use our function and grid search to get the optimum parameters and scoring results. For the complex models referred, the hypertuning was made based on simple models conclusions.
After determining which hyperparameters and data characteristics gave the greatest outcomes in each model, we compared the best solutions' training and validation binary f1-scores. We decided which of them was the best model based on the validation result, but also taking into account the overfitting present in the model.

RESULTS

Exploratory Data Analysis and Preprocessing

As stated in the Methodology section, we started by thoroughly examining the data provided, assessing the data types, looking for missing values and constructing simple visualizations. In the table below are the initial conclusions derived regarding each variable and how they relate individually to the target variable ‘ Admitted in School’.

Variables	Observations
Program	• Categorical Variable - Sorcery School / Magi Academy / Witchcraft Institute • No missing values • Most frequent value: ‘Sorcery School’ • Students who are in the Magi Academy program have a higher proportion of admittance
Student Gender	• Categorical Variable - Female / Male • No missing values • Most frequent value: ‘Male’ (65.78%) • Higher proportion of female students admitted
Experience Level	• Numerical Variable - between 0.42 and 80 • Missing about 20.5% of data points • Spearman Correlation of -0.48 with ‘Admitted to School’
Student Siblings	• Numerical Variable - between 0 and 8 • No missing values • Spearman Correlation of 0.086 with ‘Admitted to School’
Student Family	• Numerical Variable - between 0 and 6 • No missing values • Spearman Correlation of 0.12 with ‘Admitted to School’
Financial Background	• Numerical Variable - between 0 and 512.3292 • No missing values • Spearman Correlation of 0.32 with ‘Admitted to School’
School Dormitory	• Categorical Variable • Missing about 78.5% of data points - if more than 50% of the column is missing, we should drop it to avoid impacting the model incorrectly
School of Origin	• Categorical Variable - Arcan Institute / Eldertree Enclave / Mystic Academy • No missing values • Most frequent value: Mystic Academy • Students who attended Eldertree Enclave have a higher proportion of admittance
Student Social Influence	• Numerical Variable - between 1 and 24 • No missing values • Spearman Correlation of 0.019 with ‘Admitted to School’
Favourite Study Element	• Categorical Variable - Air / Earth / Fire / Water • No missing values • Most frequent value: Earth • Does not impact significantly the students’ admittance

fig. 1 - Predictor variables’ characteristics

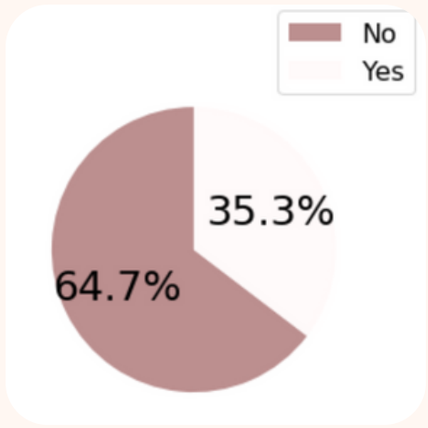

fig. 2 - ‘Admitted in School ‘ distribution

Regarding the target variable, ‘Admitted in School’, as we can see on the pie chart to the left (fig. 2), the quantity of students who weren’t admitted to the school outnumbers the quantity of students admitted, which means we are dealing with an unbalanced dataset.
This imbalance can pose a challenge, as the model may become biased toward the majority class (outcome= 0) and perform poorly on the minority class (outcome =1), which we want to avoid.

Through the visual representations of the predictors, it was also possible to detect the presence of outliers, mainly in the features ‘Financial Background’ and ‘Student Siblings’.
We found 3 outliers whose ‘Financial Background’ value was extremely high in comparison with the rest of the data points (512.3292). Upon further examination, we also found that these 3 students share the same values for the features ‘Program’ ('Magi Academy'), ‘School of Origin’ ('Eldertree Enclave') and ‘Student Siblings’ (0), they all have an high Experience Level (around the 3rd quartile) and were admitted to the school.
Additionally, we found 4 students that stand out for having 8 ‘Student Siblings’, when the closest data points have only 5 ‘Student Siblings’. Besides sharing the value of that feature, these students have the same ‘Program’ ('Sorcery School'), ‘School of Origin’ ('Mystic Academy'), ‘Student Family’ (2) and ‘Financial Background’ (69.55), which raises the possibility of them being are siblings. However, these outliers were not admitted in school.
We decided not to eliminate these outliers from the dataset, since they don´t seem to be erroneous values and might offer important information to the model (Bill Gates effect).
Subsequently, we concluded that there were no duplicate values in the dataset, and therefore, no need to treat them.

Another observation we could detect, by calculating and plotting the spearman correlation between variables, was a possible relation between Student Siblings , Student Family and Financial Background variables.
To validate and depict this relationship, we scaled those numerical variables using a MinMax Scaler and generated KDE plots of their distributions, which revealed that they were indeed similarly dispersed. (fig.3)

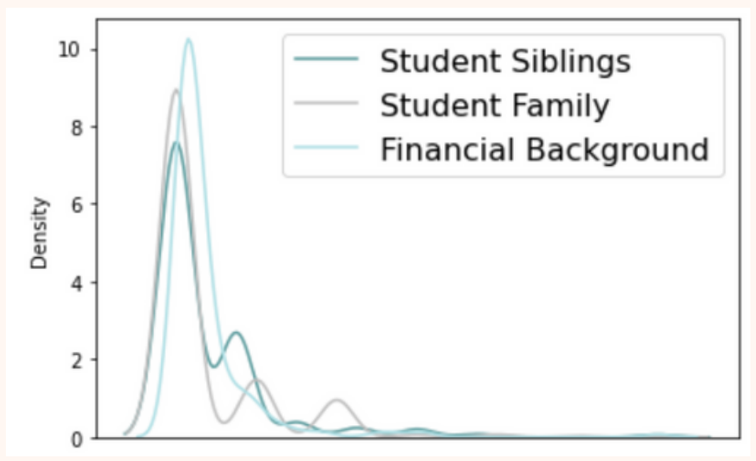

fig. 3 - Student Siblings x Student Family x Financial Background

Feature Selection

As mentioned in the Methodology, we employed 7 distinct techniques to evaluate feature importance and select only those who would add useful information to the model and train it correctly.
The conclusions were different for the various sets, but they are all displayed in the same format for each below.

- **Non scalled set:**

Predictor	Chi-Square	Univariate	Tree-Based	Spearman	VIF	Exhaustive	Lasso	What to do?
Experience Level	*Not applicable*	Keep	Keep	Discard	Discard	Keep	Keep(?)	Discard
Student Siblings	*Not applicable*	Keep	Keep	Discard	Keep	Keep	Keep	**Include in the model**
Student Family	*Not applicable*	Keep	Keep	Discard	Keep	Discard	Keep(?)	Discard
Financial Background	*Not applicable*	Keep	Keep	Discard	Keep	Discard	Keep(?)	Discard
Student Social Influence	*Not applicable*	Keep	Keep	Discard	Keep	Discard	Discard	Discard
Program_Sorcery School	Keep	Keep	Keep	Discard	Keep	Keep	Keep	**Include in the model**
Program_Witchcraft Institute	Keep	Keep	Discard	Discard	Keep	Keep	Keep	**Include in the model**
Student Gender_male	Keep	Keep	Keep	Keep(?)	Keep	Keep	Keep	**Include in the model**
School of Origin_Eldertree Enclave	Keep	Keep	Discard	Discard	Keep	Keep(?)	Keep(?)	**Include in the model**
School of Origin_Mystic Academy	Keep	Keep	Discard	Discard	Discard	Keep(?)	Keep(?)	Discard
Favourite Study Element_Earth	Discard	Keep	Discard	Discard	Keep	Discard	Discard	Discard
Favourite Study Element_Fire	Discard	Keep	Discard	Discard	Keep	Discard	Discard	Discard
Favourite Study Element_Water	Discard	Keep	Discard	Discard	Keep	Keep(?)	Discard	Discard

fig. 4 - Feature Selection results for the non-scaled set

- **Standard Scaler set:**

Predictor	Chi-Square	Univariate	Tree-Based	Spearman	VIF	Exhaustive	Lasso	What to do?
Experience Level	*Not applicable*	Keep	Keep	Discard	Discard	Keep	Keep	Discard
Student Siblings	*Not applicable*	Keep	Keep	Discard	Keep	Keep	Keep	**Include in the model**
Student Family	*Not applicable*	Keep	Keep	Discard	Keep	Discard	Keep(?)	Discard
Financial Background	*Not applicable*	Keep	Keep	Discard	Keep	Discard	Keep	Try with and without
Student Social Influence	*Not applicable*	Keep	Keep	Discard	Keep	Keep(?)	Discard	Discard
Program_Sorcery School	Keep	Keep	Keep	Discard	Keep	Keep	Keep	**Include in the model**
Program_Witchcraft Institute	Keep	Keep	Discard	Discard	Keep	Keep	Keep	**Include in the model**
Student Gender_male	Keep	Keep	Keep	Keep(?)	Keep	Keep	Keep	**Include in the model**
School of Origin_Eldertree Enclave	Keep	Keep	Discard	Discard	Keep	Keep(?)	Keep(?)	**Include in the model**
School of Origin_Mystic Academy	Keep	Keep	Discard	Discard	Discard	Keep(?)	Keep(?)	Discard
Favourite Study Element_Earth	Discard	Keep	Discard	Discard	Keep	Discard	Discard	Discard
Favourite Study Element_Fire	Discard	Keep	Discard	Discard	Keep	Discard	Discard	Discard
Favourite Study Element_Water	Discard	Keep	Discard	Discard	Keep	Discard	Keep(?)	Discard

fig. 5 - Feature Selection results for the Standard Scaler set

- **MinMax set:**

Predictor	Chi-Square	Univariate	Tree-Based	Spearman	VIF	Exhaustive	Lasso	What to do?
Experience Level	*Not applicable*	Keep	Keep	Discard	Discard	Keep	Keep	Discard
Student Siblings	*Not applicable*	Keep	Keep	Discard	Keep	Keep	Keep	**Include in the model**
Student Family	*Not applicable*	Keep	Keep	Discard	Keep	Keep(?)	Keep(?)	**Include in the model**
Financial Background	*Not applicable*	Keep	Keep	Discard	Keep	Keep	Keep	**Include in the model**
Student Social Influence	*Not applicable*	Keep	Keep	Discard	Keep	Keep(?)	Discard	Discard
Program_Sorcery School	Keep	Keep	Keep	Discard	Keep	Keep	Keep	**Include in the model**
Program_Witchcraft Institute	Keep	Keep	Discard	Discard	Keep	Keep	Keep	**Include in the model**
Student Gender_male	Keep	Keep	Keep	Keep(?)	Keep	Keep	Keep	**Include in the model**
School of Origin_Eldertree Enclave	Keep	Keep	Discard	Discard	Keep	Keep	Keep(?)	**Include in the model**
School of Origin_Mystic Academy	Keep	Keep	Discard	Discard	Discard	Keep(?)	Keep(?)	Discard
Favourite Study Element_Earth	Discard	Keep	Discard	Discard	Keep	Keep(?)	Discard	Discard
Favourite Study Element_Fire	Discard	Keep	Discard	Discard	Keep	Discard	Keep(?)	Discard
Favourite Study Element_Water	Discard	Keep	Discard	Discard	Keep	Discard	Keep(?)	Discard

fig. 6 - Feature Selection results for the MinMax Scaler set

- **Robust Scaler set:**

Predictor	Chi-Square	Univariate	Tree-Based	Spearman	VIF	Exhaustive	Lasso	What to do?
Experience Level	*Not applicable*	Keep	Keep	Discard	Discard	Keep	Keep	Discard
Student Siblings	*Not applicable*	Keep	Keep	Discard	Keep	Keep	Keep	**Include in the model**
Student Family	*Not applicable*	Keep	Keep	Discard	Keep	Discard	Keep(?)	Discard
Financial Background	*Not applicable*	Keep	Keep	Discard	Keep	Discard	Keep(?)	Discard
Student Social Influence	*Not applicable*	Keep	Keep	Discard	Keep	Keep(?)	Discard	Discard
Program_Sorcery School	Keep	Keep	Keep	Discard	Keep	Keep	Keep	**Include in the model**
Program_Witchcraft Institute	Keep	Keep	Discard	Discard	Keep	Keep	Keep	**Include in the model**
Student Gender_male	Keep	Keep	Keep	Keep(?)	Keep	Keep	Keep	**Include in the model**
School of Origin_Eldertree Enclave	Keep	Keep	Discard	Discard	Keep	Keep(?)	Keep(?)	**Include in the model**
School of Origin_Mystic Academy	Keep	Keep	Discard	Discard	Discard	Keep	Keep(?)	Discard
Favourite Study Element_Earth	Discard	Keep	Discard	Discard	Keep	Discard	Discard	Discard
Favourite Study Element_Fire	Discard	Keep	Discard	Discard	Keep	Discard	Discard	Discard
Favourite Study Element_Water	Discard	Keep	Discard	Discard	Keep	Keep(?)	Keep(?)	Discard

fig. 7 - Feature Selection results for the Robust Scaler set

As for the **Principal Component Analysis**, a separate function was defined to perform dimensionality reduction on the input and then determine the number of principal components needed to meet or exceed the specified variance threshold, whose default we set to 0.8. This function's output included not only a DataFrame containing the selected features, but also a plot of the cumulative explained variance ratio to help visualize how much variance is explained by each principal component (fig.8). The red line represents the specified threshold and the markers above it represent the discarded features.

fig. 8 - PCA results for the non-scaled set, Standard Scaler set, MinMax Scaler set and Robust Scaler set respectively

However, after testing this function on our several scalled datasets, we noticed that it did not perform as well as the approach described above and hence didn't use it to train our model. The underperformance was attributed to the technique's dependency on the model's linearity, resulting in its inability to effectively capture non-linear patterns. Nonetheless, we thought it was important to discuss every stage of the study, including the strategies that were not employed, because they led to a more thorough examination of the best approaches to apply for our model.

Model Evaluation

As mentioned in the Methodology section, GridSearchCV() was used to test a number of estimators and parameter combinations. Among the models examined, the **Decision Tree** followed by **K-Nearest Neighbors**, **Multi-Layer Percepton, Logistic Regression** and **Naive Bayes**, from sklearn produced some of the best outcomes. It is important to remember that these outcomes are the result of several evaluations of which dataset would perform better with the models, as we have different types of scalled datasets and feature selections. In the following tables are the the characteristics of the data that perfomed best in each model, and their scoring results on both train and validation data:

Model	Dataset Characteristics
Logistic Regression	• Scalling- StandardScaler • No Feature Selection
Decision Tree Classifier	• No Scaling • No Feature Selection
KNeighbors Classifier	• Scalling- StandardScaler • Feature Selection: included variables - Student Siblings, Financial Background,Program_Sorcery School, Program_Witchcraft Institute, Student Gender_male, School of Origin_Eldertree Enclave
Naive Bayes	• Scalling- StandardScaler • Feature Selection: included variables - Student Siblings,Program_Sorcery School, Program_Witchcraft Institute, Student Gender_male, School of Origin_Eldertree Enclave
Multi-layer Perceptron	• Scalling- StandardScaler • No Feature Selection

fig. 9 - Highest performing models

	Train	Validation
Logistic Regression	0.74+/-0.01	0.72+/-0.05
Decision Tree	0.899+/-0.01	0.753+/-0.06
K-Nearest Neighbors	0.761+/-0.01	0.737+/-0.05
Naive Bayes	0.698+/-0.01	0.698+/-0.05
Multi-Layer Perceptron	0.742+/-0.01	0.698+/-0.07

fig. 10 - F1 Score results on train and validation data of each model

However, we had to asses the possibility of overfitting, and because there is a considerable difference between the Decision Tree results on the training versus the validation data, we consider some overfitting to be occurring.

As a result, we found the K-Nearest Neighbors model to be the best overall model given that it performs strongly in both datasets, excluding likelihood of overfitting or underfitting. Furthermore, Standard scaler is applied to this model, so there is no bias from variables with larger magnitude, and it utilizes a feature-selected dataset, which reduces complexity.

Finally, the plots below, illustrate the reduction in dimensionality of the train and test data differentiated by target. By viewing both graphs, we can clearly detect certain patterns in the data, as most data points with a negative target outcome are grouped together, whereas positive outcomes, although also tending to stay near, are also in close distance with several negative target data points. This illustrates how difficult it is to accurately predict a positive target, or in other words, if a student was addmited.

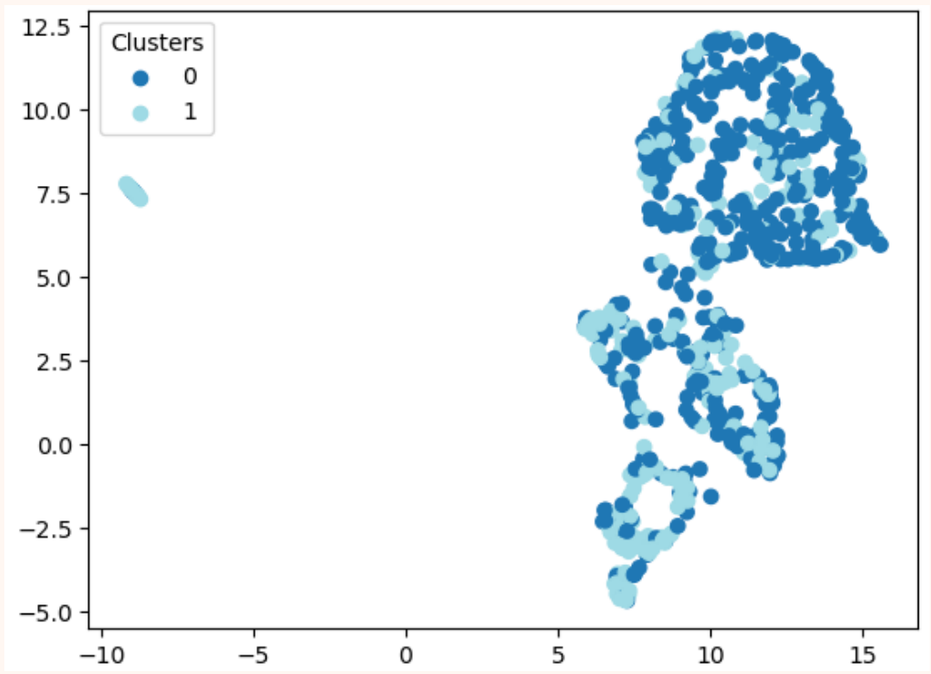

fig. 11 - Dimensionality Reduction of train data distinguished by the target

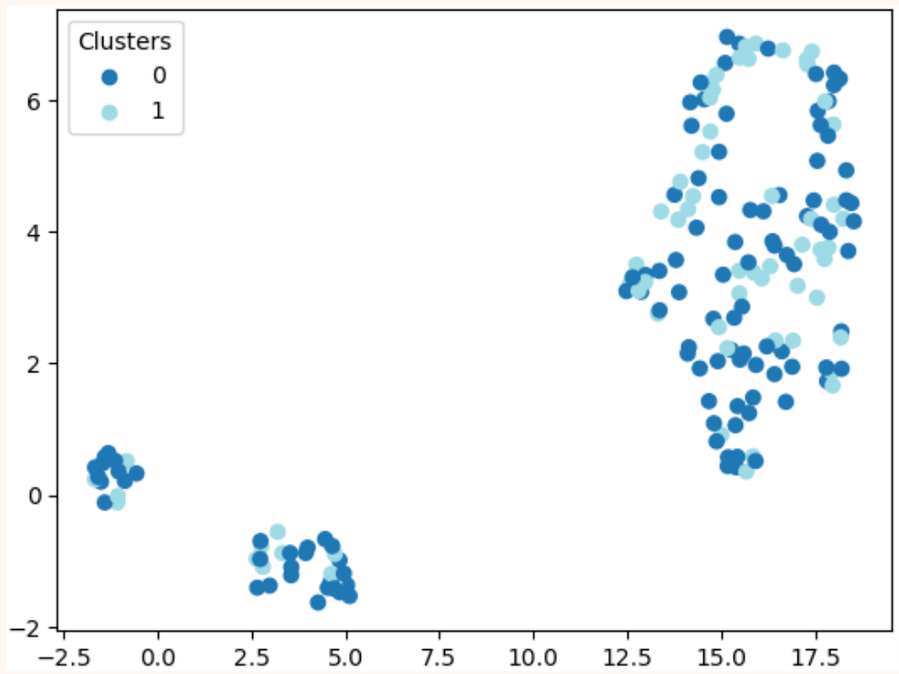

fig. 12 - Dimensionality Reduction of test data distinguished by the target

Discussion

The development of this predictive model allowed us to take a more practical approach to building a machine learning program for a real-world categorization assignment. However, while conducting our investigation, we encountered some obstacles.

Firstly, during our data exploration and preprocessing phase, we found a class imbalance problem among the target variable, which was a main challenge in creating an efficient predictive model free of biases and overfitting. This was primarily addressed by using assessment measures that take these conditions into account, like the F1-Score. Also, determining the proper techniques for choosing the attributes that would best serve the model was another crucial and notably difficult stage. It was possible to assess each feature's relevance and determine which set of predictors would be most useful to include in the modeling process by using the Filtering, Wrapper, and Embedded methods. This led to the conclusion that the variables pertaining to the student siblings, their program at the sorcery school and witchcraft institute, the male students, and their origins from Eldertree Enclave school were typically the most beneficial in predicting their addmitance.

Furthermore, several of the project's tasks involved highly repetitive processes, which made it difficult for us to organize our code and come up with creative yet efficient solutions to complete all the tasks at hand. This problem mostly surfaced during the feature selection and model evaluation phases.

Moreover, we believe that using a dataset with more observations for this same analysis would be extremely advantageous. This could result in a greater amount of information available for defining the influence of the dataset's features and, as a result, a better understanding of their predictive value. Lastly, while not every model that was tested yielded impressive results, the majority of them were able to forecast with noteworthy accuracy levels, which suggests that student addmitance is, in fact, predictable.

Conclusion

Our team was tasked with developing a a Supervised Learning model capable of predicting whether or not students are qualified to enroll in a specific wizardry school. Given that the goal of this institutions is to only accept the best students, a suitable and refined model is required, which is why special emphasis was placed on defining the best criteria to evaluate if the predictive model built has an optimal performance.

We came to the conclusion that K-Nearest Neighbors model, which is described in our Results section, wielded the best results, satisfying the key criterion of accuratly predicting which students are admitted.

Employing this model alongside comprehensive demographic and behavioral data allowed us to discern distinct patterns across various student levels. This implementation yielded a powerful tool to evaluate and pinpoint students well-suited for admission to magical institutions.

From a wider perspective, the ideas from this investigation might be applied to investigate how predictive algorithms could help in education. For instance, we may modify the approach to investigate how Machine Learning could benefit students in general. We may classify kids based on shared features using strategies like Unsupervised Learning, allowing schools to better understand and serve their students' needs. This could lead to better educational experiences and student satisfaction.
Its adaptability extends to predicting admissions across diverse educational institutions, offering a versatile framework applicable to a wide array of schooling scenarios.

References

[1] 1.4. Support Vector Machines. scikit-learn. Published 2023. Accessed December 18, 2023. https://scikit-learn.org/stable/modules/svm.html

[2] 1.13. Feature selection. scikit-learn. Published 2023. Accessed December 18, 2023. https://scikit-learn.org/stable/modules/feature_selection.html

[3] Bro R, Smilde AK. Principal component analysis. Analytical Methods. 2014;6(9):2812-2831. doi:https://doi.org/10.1039/c3ay41907j

[4] Diaz-Papkovich A, Anderson-Trocmé L, Gravel S. A review of UMAP in population genetics. Journal of Human Genetics. 2020;66(1):85-91. doi:https://doi.org/10.1038/s10038-020-00851-4

[5] Ketan Rajshekhar Shahapure, Nicholas C. Cluster Quality Analysis Using Silhouette Score. Maryland Shared Open Access Repository (USMAI Consortium). Published online October 1, 2020. doi:https://doi.org/10.1109/dsaa49011.2020.00096

[6] Jamal Ibrahim Daoud. Multicollinearity and Regression Analysis. Journal of physics. 2017;949:012009-012009. doi:https://doi.org/10.1088/1742-6596/949/1/012009

[7] UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction — umap 0.5 documentation. Readthedocs.io. Published 2018. Accessed December 18, 2023. https://umap-learn.readthedocs.io/en/latest/