AARHUS UNIVERSITY

AARHUS BSS

Summer University in Python Fundamentals for Machine Learning

Predicting Student Dropouts:

An evaluation of machine learning methods

Authors

Rufus Ewings

Mark Hellsten

Alicja Agata Siudak

Monika Maja Górka

Jakob Thusgaard Olsen

Advisors

Balu Mohandas Menon,

Tharsika Pakeerathan Srirajan

Academic Year 2021/2022

Abstract

This project evaluates different supervised machine learning algorithms in the context of predicting student drop-outs which classifies as a multinomial classification problem. The chosen dataset consists of information on performance and traits of students from Instituto Politécnico De Portalegre in Portugal. Four different methods are employed and it is found that a logistic regression makes the most accurate prediction in this case. To optimize the prediction, different methods of date preprocessing is used, where the accuracy of each one dictate which sample is appropriate for the given situation.

Contents

1. Problem introduction

Mark, Jakob

A problem facing the educational system is student drop-outs. Student dropouts are costly both to society, and the student. Predicting and identifying the student at risk of dropping out may not allow for any direct action, as one likely cannot use the predictions as grounds for admission. However, it could provide useful information about the traits of drop-outs, which could help the institution in building a support network for enrolled students.

Investigating student dropouts has been done before, both using traditional methods and machine learning (Serra *et al.*, 2018, Aulck *et al.*, 2016, Martins *et al.*, 2021). As a sample of these, Serra *et al.* (2018) was able to predict dropouts with a 67 percent accuracy with a logistic regression, and 65 percent using k-nearest neighbours. Aulck *et al.* (2016) found got a 72 percent accuracy with a random forest classifier, and 65 percent ordinary decision tree. This suggests that machine learning may be able to increase the prediction accuracy in certain cases, although not substantially so far.

Another consideration is that while traditional methods may be able to accurately predict which students will drop out using "obvious" factors such as GPA and parents' income, this knowledge may be useless in the eyes of the policymaker (the income of a student's parents can't be changed, after all.). Machine learning may contribute by identifying less obvious factors influencing students' decisions to drop out. One of the most common issues in "success or dropout" classification is the class imbalance issue. Class imbalance happens when when one of the classification values is significantly lower than the other one, resulting in bad prediction of the rows the minor category. In this case the aim is to identify the dropout students but they represent only a small fraction of the rows.

This project aims to analyze the "Students' dropout and academic success" data set and explore different machine learning approaches to find which methods may be appropriate

when predicting student dropouts. The data set contains information about students and their backgrounds for a number of factors, as well as details on their academic performance. Ranging from details of their parents' income to their admission grades, these categories paint a picture of each student and the elements which may or may not have a substantial impact on their chances of success, or failure. Being able to identify patterns through those experiments could be crucial to developing mitigation strategies to prevent dropouts.

This paper seeks to analyse and evaluate the use of different machine learning methods in their application to a classification problem: predicting university dropouts based on a variety of factors.

2. Data collection and description

Alicja, Maja

Data set used for this project was obtained from an online platform. It is a student data that was originally collected and processed by researchers from Instituto Politécnico De Portalegre (IPP) Martins *et al.* (2021). The collection of data took place across a span of four years; 2008-2009, and 2018-2019 and resulted in 4424 entries (rows). The full data set can be found in the Attachment

As can be seen in Figure 1 the data contains wide range of information accumulating to a total of 37 columns of different data types. The columns contain, among other, data about student's performance, their and theirs parents' background, some time-based control variables, as well as student's current enrollment status. The enrollment status can be either one of three labeled classes: graduated, enrolled or dropout.

FIGURE 1
Feature list

1. Marital status
2. Application mode
3. Application order
4. Course
5. Daytime/evening attendance
6. Previous qualification
7. Previous qualification (grade)
8. Nationality
9. Mother's qualification
10. Father's qualification
11. Mother's occupation
12. Admission grade
13. Displaced
14. Educational special needs
15. Debtor
16. Tuition fees up to date
17. Gender
18. Scholarship holder
19. Age at enrollment
20. International
21. Curricular units 1st sem (credited)
22. Curricular units 1st sem (enrolled)
23. Curricular units 1st sem (evaluations)
24. Curricular units 1st sem (approved)
25. Curricular units 1st sem (grade)
26. Curricular units 1st sem (without evaluations)
27. Curricular units 2nd sem (credited)
28. Curricular units 2nd sem (enrolled)
29. Curricular units 2nd sem (evaluations)
30. Curricular units 2nd sem (approved)
31. Curricular units 2nd sem (grade)
32. Curricular units 2nd sem (without
33. evaluations)
34. Unemployment rate
35. Inflation rate
36. GDP
37. Target

Most of the acquired data was represented as numbers and described in a legend as in an example below. The full description of features can be found in the Attachment.

FIGURE 2
Feature information

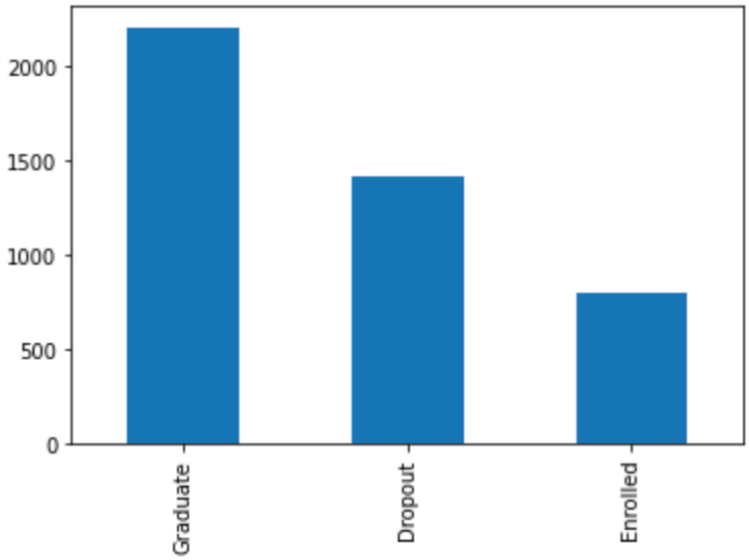
Feature	Type	Description
Marital status	Numeric/discrete	1 – single 2 – married 3 – widower 4 – divorced 5 – facto union 6 – legally separated
Application mode	Numeric/discrete	1 - 1st phase - general contingent 2 - Ordinance No. 612/93 5 - 1st phase - special contingent (Azores Island) … 44 - Technological specialization diploma holders 51 - Change of institution/course 53 - Short cycle diploma holders 57 - Change of institution/course (International)

Source: https://www.kaggle.com/datasets/ankanhore545/dropout-or-academic-success/discussion/333138

A potential worry for the data is how representative it is. It only covers one small school in Portugal. The topic at hand would open up the door for using large data sets (for example, registered data), with information from all regions of the country. However, this would require both access to such data, and a substantial amount of work cleaning the data. As the purpose of this examination is mostly related to the application and evaluation of machine learning methods, it has been decided to leave this as a limitation in this project.

After looking closer into the set it was discovered that the data set is imbalanced. Meaning, for each of the classes that the predictions are going to be made, there is a significant difference between the number of entries, as can be seen on the Figure 3.

FIGURE 3
Entries of labeled data



3. THEORETICAL BACKGROUND

Alicja, Jakob

Following section is a theoretical background on the chosen algorithms as well as the evaluation methods.

3.1. KNN - K-nearest neighbor

Maja, Jakob

K-nearest neighbor (KNN) is a widely used classification method that is popular for its simplicity. The KNN classifier is regularly based on the Euclidean distance between a test sample and the specified training samples. Keller *et al.* (1985), Peterson (2009)

3.2. Support vector machine

Maja, Jakob

A Support Vector Machine (SVM) is an extremely strong and flexible Machine Learning model that can perform regression, outlier identification, and linear or nonlinear classification. It is one of the most prominent models in Machine Learning and is particularly well suited for classification of complex but small or medium-sized datasets.Geron (2019)

3.3. Logistic regression

Maja, Jakob

Estimating the likelihood that an instance belongs to a specific class is frequently done using logistic regression, also known as logit regression. The model either predicts that the instance belongs to that class (known as the positive class, labeled "1") if the estimated probability is greater than 50%, or that it does not (i.e., that it belongs to the negative class, labeled "0") if the estimated probability is less than 50%. As a result being a binary classifier.

Without the need to train and merge numerous binary classifiers, the Logistic Regression model may be readily adapted to handle multiple classes. Geron (2019)

3.4. Random Forest Classification

Alicja, Maja

The Random Forests algorithm is a supervised machine learning algorithm that has been utilized in numerous studies to address the classification problem. The classifier is an ensemble algorithm that creates several decision trees and combines them to create a more powerful classifier. The most discriminant characteristic is chosen using the decision tree algorithm, which employs entropy and information gain to branch from it. Overfitting is one of the drawbacks of decision trees, especially if the tree is deep. If there are enough trees, an RF classifier can limit overfitting,

3.5. Evaluation Metrics

Alicja, Maja

As previously mentioned, the data set used is imbalanced. Accuracy, the most commonly used measure for the classification performance, by itself could be misleading. As the accuracy is a ratio of correctly classified samples to the total number of samples, it might be high even if the model does not predict minority class correctly. In consequence, to evaluate the experiments and select a model with the best predictive capability multiple factors must be taken into consideration. Geron (2019)

Confusion matrix allows to analyse results in a more holistic way, by providing information, such as:

- True positives (TP): Target variable labeled as positive that are actually positive
- False positives (FP): Target variable labeled as positive that are actually negative
- True negatives (TN): Target variable labeled as negative that are actually negative
- False negatives (FN): Target variable labeled as negative that are actually positive

Based on the above following metrics, along accuracy, can be calculated:

$$Precision = \frac{TP}{TP + FP}$$

Precision is a metric that represents how many of the values predicted to belong to a class were actually correct. It is used to reduce false positives.

$$Recall = \frac{TP}{TP + FN}$$

Recall measures correct predictions out of sum of correct ones and items which were not labeled as belonging to the positive class but should have been. It is used to reduce false negatives.

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

The F1-score combines recall and precision of a model into a single metric by taking their harmonic mean. It is primarily used to compare the performance of two models.

4. Data cleaning and feature selection

Jakob, Alicja

Data cleaning is the process of identifying and correcting (or removing) inaccurate or corrupt records from a record set, table, or database. In order to do that special attention must be paid to missing (null values), irregular (outliers), unnecessary (duplicates, etc.) and inconsistent data (capitalization).

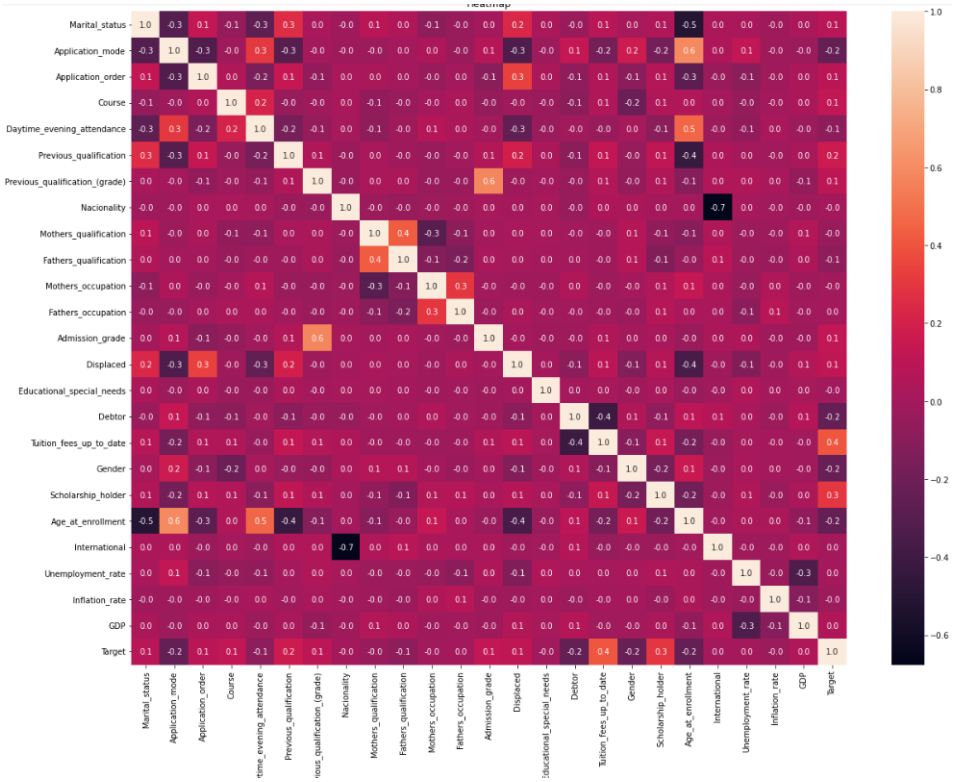
A heat map of null values was created to identify missing data. It was clear that the data wasn't missing any values.

On initial use of a 'Random Forest' with size 100 (further elaborated in section 4.2., it was revealed that the most correlated features were those which described a students grades, particularly in the second semester. This led to the decision, based on intuition, to ex-

clude categories of student grades: as the grades that a student achieves directly causes the pass/fail outcome. A student with nil marks on the final semester is highly likely to have dropped out and does not reveal anything about the impact of outside factors on a students likelihood for success.

It was decided to use Seaborn's heat-mapping functions to visualise the correlation between the features. The information provided on figure 4 can be used as arguments for eliminating some of the features manually, thus reducing the dimensionality.

FIGURE 4
Heatmap on Feature Correlation



As can be seen on the Heatmap, there are no alarmingly high correlation between features so removing features based on this wouldn't be optimal. Instead the information will be used

to validate the feature importance scoring that the Random Forest model suggests. The lowest row in the heatmap shows the correlation with "Target" which is the chosen output variable. From this figure it can be conlcuded that the features with the highest importance are "Tuition fees up to date" and "Scholarship holder".

4.1. Encoding

Rufus, Jakob

One of the greatest issues faced initially was the format of the data being used. Some were strings, some were categories (represented by numbers) and some purely numeric. The categorical variables would need to be encoded in a way that meant they were readable by algorithms (which cannot process strings) and also didn't imply the data had a hierarchy, or was ordered in any way.

Categorical values can be either nominal or ordinal. Nominal is when the numbering of the values does not have any inherent ordering between them. The algorithm should not misuse the values by treating them as hierarchical. Ordinal is when there is a clear ordering between the categories. For example for a category 'wealth' (lower class, middle class, upper class and so fourth). As mentioned, the dataset contains a combination of numerical and categorical features. Although the categories were already label encoded into nominal values, the decision was made to apply OneHot Encoding method on the categorical features to avoid issues with algorithms mistaking the data for ordinal. (Vanooteghem, 2020)

The categorical data was first converted to its corresponding category in text through use of dictionaries and for loops, this would enable use of the names as categories and also for later ease of translating data from number to its description - without having to manually check what each number meant. In hindsight this was unnecessary as the data would later be encoded back into labels, but the initial understanding was that correlation would be found based on the hierarchy of the data without prior knowledge of encoding.

Converting entire columns into binary data was tested, for example, converting 'Nationality' into a 1 or 0 for Portuguese or others. This was scrapped in favour of more encompassing encoding. Following this, binary encoding was considered for the categoricals. This would enable the splitting of categoricals into a small number of columns (unlike one-hot encoding which creates a column for each category), by assigning each binary number of the columns to a category - for example, two binary columns could hold four categories with 00, 01, 10 and 11.

Problems with this were encountered upon attempting to reverse and manipulate the encoding, which was in fact a hash function and thus one way. Once again this was before the revelation that data wouldn't require reversing, but regardless this increase in complexity was unnecessary for the data which spanned just fewer than 40 columns.

FIGURE 5
One-hot Encoding

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer

df_hot = df
for column in columns:

    column_transformer = make_column_transformer(\
                      (OneHotEncoder(), [column]),remainder='passthrough')

    df_hot = column_transformer.fit_transform(df_hot)
    df_hot = pd.DataFrame(data=df_hot, \
                          columns=column_transformer.get_feature_names_out())
    df_hot.columns = df_hot.columns.str.lstrip("mainder__")
    df_hot.columns = df_hot.columns.str.lstrip("onehotencoder__")
```

OneHot Encoding technique is when every nominal categorical feature is split into several binary features, so that all values are represented as new features. One of the disadvantages is the increase of dimensionality in the data set which can increase the error in the prediction

that the algorithm suggests if not handled properly. Another issue that can arise is when the number of features is increased a lot, there might not be enough observations for each feature combination. Algorithms such as Random Forest with Feature Importance Selection takes care of both of these issues by adding adequate amount of trees and selecting only the most important features.

Label encoding was considered for the purpose of normalising the categorical data into numbers. While this works for ordinal data it could be misconstrued by algorithms to be hierarchical, to prevent this OneHotEncoding would need to be utilised. This would be applied on top of the label encoding to split each different category into its own class, using GetDummies() 'Prefix' parameter to pass in the names for those columns. Difficulty with column naming led to the decision to ditch LabelEncoding, however, as it wasn't necessary with all implementations of OneHot, now with a little string manipulation the columns fit the categories as planned.

FIGURE 6
Label Encoding

```
[ ]  from sklearn.preprocessing import LabelEncoder
     l_df = df.copy()
     # Label encoding normalises labels into numbers, maintaining column name
     encoders = {}
     for col in l_df.columns:
      # If column in list of categorical columns
      if col in columns or col == 'Target' :
        # Encode by label
        le = LabelEncoder().fit(df[col])
        l_df[col] = le.transform(df[col])
        encoders[col] = le
```

Label encoding was re-implemented, next to one-hot after a shift in understanding of which algorithms would be applied and thus the lack of issues caused by the data nominal data being misconstrued as ordered. The reduction in dimensionality was hoped to improve performance

but each encoding performed better in different algorithms. For example, K-Nearest Neighbour (KNN) in particular excelled on data-sets with fewer columns. Nevertheless, one-hot was retained to allow the data to dictate what should be used.

4.2. Dimensionality Reduction

Mark, Rufus

Due to the number of categorical variables, what happens to the prediction when using different ways of reducing dimensionality have been checked. Two methods of doing this were identified, those being feature importance score, and Principal Component Analysis (PCA). These have been saved as three distinct datasets, and performed all the analyses on all of them to compare the results

FIGURE 7
Random Forest

```
[19] '''
     Dimension reductions wiith Random Forest

     Run a random forest, then rank features by importance and remove the
     least important from the data.

     Split training and testing sets.
     '''

     # Hyperparameters include the number of decision trees in the forest and the
     #   number of features considered by each tree when splitting a node

     l_df_top = pd.DataFrame()
     df_hot_top = pd.DataFrame()
     sample_list = [[l_df, l_df_top], [df_hot, df_hot_top]]


     for sample in sample_list:

       y = sample[0]['Target']
       X = sample[0].drop(['Target'], axis=1)
       X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                           test_size=0.25,
                                                           random_state=10)

       # Select model and fit data
       rfc = RandomForestClassifier(n_estimators=500, random_state=0)
       rfc.fit(X_train, y_train)

       # Create feature score list
       feature_scores = pd.Series(rfc.feature_importances_, index=X_train.columns)\
                                                        .sort_values(ascending=False)
       feature_scores.head(20)

       # Save top 20 features into ew dataset
       top_features = []
       for i in range(0, 20):
           top_features.append(list(feature_scores.index.values)[i])

       # Re add target column
       sample[1]['Target'] = sample[0]['Target']

       # Assign encodings top columns to top_features
       for column in top_features:
           sample[1][column] = sample[0][column]

       sample[1].head()

       # Plot graph

       feature_scores = feature_scores.head(20)
       sns.barplot(x=feature_scores, y=feature_scores.index)
       plt.xlabel('feature importance score')
       plt.show()
```

For the feature score method, Random Forest has been run with the same parameters, and random test-train split as the main tests. Based on this, the feature scores have been extracted from the model, i.e. the importance of each feature in the creation of the random forest. Then the 20 most important columns have been chosen to be kept.

FIGURE 8
Feature scores

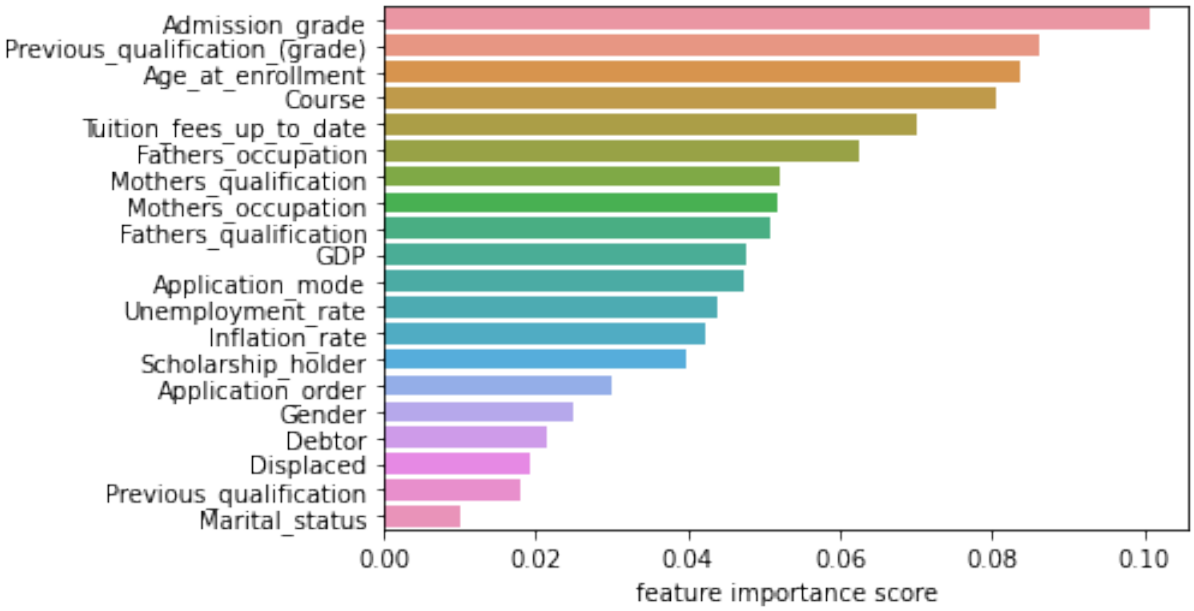


Figure 8 illustrates the features deemed most important by the Random Forest. As can be seen, Admission grade is the most important feature in predicting drop outs. This is unsurprising, considering a top-student would be unlikely to produce a result bad enough to consider dropping out. Most of the top features make common sense, and likely capture effects related to academic performance, economic situation of the student, as well as some personal traits. The GDP and unemployment rate are time based, and so probably just captures all relevant yearly effects related to the year the course was taken.

FIGURE 9
PCA Code

```
def PCA_Reduction(df):
  # dimensionality reduction
  from sklearn.decomposition import PCA
  from sklearn.preprocessing import MinMaxScaler

  pca = PCA(n_components=0.9)
  p = df
  p = p.drop(['Target'], axis=1)

  # Standardise and normalise the data
  p = (p - p.mean()) / p.std()

  scaler = MinMaxScaler()
  data_rescaled = scaler.fit_transform(p)

  components = pca.fit_transform(data_rescaled)
  p = pd.DataFrame(pca.transform(data_rescaled))
  p['Target']=df['Target']
  plt.plot(pca.explained_variance_ratio_)
  plt.ylabel('Explained Variance')
  plt.xlabel('Components')
  plt.show()

  return p
```

PCA was then utilised for the purpose of reducing the dimensionality of the data set, bringing the number of columns from 36 to only 18, whilst maintaining 90% of the variance found in the original data. As the key information in a table lies in the variance between different values, PCA is highly effective at compressing data with very little loss. Redundant variables are also removed, something which could potentially be improved upon by greater levels of preprocessing using intuition, for example: the incorporation of GDP and inflation, however the variance in these is likely to be minimal due to the amount of duplicate data and thus PCA was sufficient.

FIGURE 10
Explained Variance Ratio

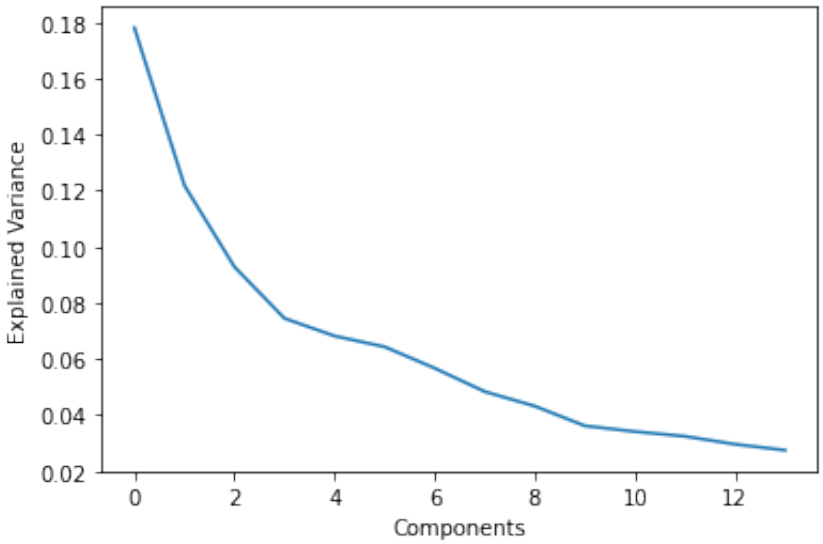


Figure 10 represents the variance ratio of each item in the PCA reduced dataframe. This is presented as a line graph to demonstrate the gradient and the rate at which the variance drops off - improved greatly by standardisation and normalisation of the data before applying PCA.

4.3. SELECTION OF DATA PROCESSING

Mark, Maja

As it would not be feasible to present the results of every data format(label- or one-hot encoding) and sample (full sample, random forest sample, or PCA sample), it was decided to evaluate the methods in advance, and to select the ones with the best prediction accuracy for each of the methods.

TABLE 1
Accuracy scores of different samples

	Full sample	PCA reduction	Random forest top-20
Label encoding			
Random forest	0.616	0.608	0.62
Logistic regression	0.601	0.604	0.603
KNN	0.533	0.605	0.532
SVC	0.611	0.59	0.61
One-hot			
Random forest	0.636	0.621	0.602
Logistic regression	0.648	0.628	0.607
KNN	0.528	0.587	0.53
SVC	0.637	0.632	0.61

Table 1 shows the accuracy scores of all the methods on both encoding types and every sample. The one-hot encoding seems to perform better for all methods but the KNN. The same is true for the full sample, where all methods aside from KNN perform best. This is likely because those methods are not vulnerable to the large number of features contained within the

That the KNN performs best in the label encoded PCA reduction is no surprise. KNN is vulnerable to data with a large number of dimensions. The PCA reduction of label encoding contains far fewer columns than its One-hot counterpart, while containing the same share of explained variance. Based on this, the One-hot has been chosen to be used, full sample for the Random forest, Logistic regression, and SVM, while the PCA reduction of the label encoded data for the KNN model was used.

5. Model Training

This section will discuss the considerations that needs to be made in order to run the machine

learning algorithms i.e. the libraries needed, the chosen datasplit, this algorithms them self as well as the chosen parameters within each of them.

5.1. Libraries

Jakob, Mark

The Scikit-learn is a open source library that is based on Numpy, Scipy and Matplotlib. It is considered the most robust library for machine learning and predictive data analysis. The package contains all the models used in this project. It also contains methods for evaluating the methods in question. This project primarily relied on accuracy scores, and data extracted from confusion matrices. For visualising and manipulating data, Seaborn and Pandas were used.

5.2. Splitting the dataset

Jakob, Alicja

The train/test split is a procedure in which the dataset is split into multiple portions so that the algorithm can be trained and evaluated within the same dataset. It is a typical procedure that can be used on any supervised learning algorithms.

It was chosen to use 80 percent of the data for training and the rest for testing the algorithms. Thís split seemed reasonable because of the relatively big number of rows in the dataset.

If the dataset had been even bigger, it could also be a good idea to split the dataset into three parts eg. 60/20/20, where the last 20 percent would function as a validation set.

Since it was chosen to split the data into two parts, thus disregarding the validating split, Cross validation can be very handy as a validation tool. Cross validation is a technique in which the dataset is divided into k subsets of data and then the model is run k times. Finally the average of the values from each iteration is found, ensuring a better use of the data since all observations is used as both test and training. Due to time limitations it was chosen to

not utilize this method.

5.3. KNN

Mark, Alicja

For the KNN model, the following code to define the model was used. The only setting chosen for this model was the number of neighbours to be included. This setting was chosen on the simple grounds that a low or very high value gave poor accuracy.

KNN_model = KNeighborsClassifier(n_neighbors=100)

TABLE 2
KNN

	precision	recall	f1-score
Dropout	0.67	0.53	0.59
Enrolled	0.0	0.0	0.0
Graduate	0.58	0.91	0.71
Accuracy:	0.6		

Table 2 contains the scores from the KNN model. The accuracy is 60 percent, and it performed well in predicting Dropouts, with a precision score of 67 percent. For dropouts, the model seems slightly more likely to commit false negatives than false positives. For Graduates, however, very few false negatives are recorded.

There was one unexpected result with prediction, recall, and f1-scores for enrolled students, i.e. students with delayed graduation generated nil for each of these values. As the numerators of the precision- and recall scores are the number of true positives, this implies that the reason for the zero is that there is not a single true positive in the test. The cause was traced back to how the PCA reduction is scaled, as the KNN model is the only one using the PCA

sample and the PCA model changes when the scaling is adjusted. If one were to investigate this further, the sample could be split using a cross-validation, as opposed to the test-train split. This could find whether this is caused by an unlikely random distribution.

Scaling is utilised within the PCA algorithm to maximise the variance, to enable the resulting data frame to best represent the data. During this scaling the ability to find the already difficult to predict 'Enrolled' outcome becomes impossible. One solution to this, albeit a somewhat heavy handed one, would be to drop all rows with an outcome of 'Enrolled'. This solution sounds more logical upon examination of the mission statement - attempting to predict whether a student is at risk of failing. Rows containing information on students still attending university do not help to predict what the outcome of their time at university will be and thus doesn't support the studies goal. Shifting the dependent variable to a binary outcome would have a positive impact on the clarity of much of the data, suit the situation being modelled and is likely to improve the success of many of the algorithms.

In future studies, it would be worth strong consideration to maintain the all data entries but combine graduates and enrolled into a single 'NOT FAILED' categorical. The overall purpose of predicting students at risk of failing is retained.

5.4. SVM

Rufus, Alicja

SVC is given the argument 'kernel' and being set to linear rather than the default RBF. This yielded a better accuracy result with the given set due to the nature of the data. It also takes the argument C, representing regularisation. This was left as 1, the default.

SVC_model = SVC(kernel='linear', C=1)

Table 3
SVM

	precision	recall	f1-score
Dropout	0.65	0.65	0.65
Enrolled	0.55	0.08	0.14
Graduate	0.63	0.86	0.73
Accuracy:	0.64		

The classification report from the SVM can be seen in table 3. Here, the accuracy is 64 percent. The values that stand out is a large number of false negatives for the enrolled students, as well as a low number of false negatives for the graduate prediction.

5.5. Logistic Regression

Rufus, Maja

Logistic regression takes the maximum number of iterations as an argument, this was set to 10,000 meaning the regression will run 10,000 times before coming to a conclusion on the likelihood/ratio of something occurring.

$$\text{logreg} = \text{LogisticRegression}(\text{max\_iter}{=}10000)$$

Table 4
Logistic regression

	precision	recall	f1-score
Dropout	0.66	0.65	0.66
Enrolled	0.53	0.18	0.26
Figure Graduate	0.65	0.84	0.74
Accuracy:	0.65		

The Logistic regression performed well with an overall accuracy of 65 percent, although the share of false negatives for the enrolled students seems quite high.

5.6. Random Forest

Mark, Jakob

The random forest was based on the following model. The setting of n_estimators is the number of trees in the forest. Again, a trial and error approach has been chosen, where a low number of trees did not give us good accuracy, while more trees did not affect the accuracy significantly. The random_state setting makes the random forest make the same random decision every time, here zero was picked to ensure the results are the same every time tests are run, all else equal.

rfc = RandomForestClassifier(n_estimators=500, random_state=0)

Table 5
Random forest

	precision	recall	f1-score
Dropout	0.65	0.65	0.65
Enrolled	0.5	0.05	0.09
Graduate	0.63	0.87	0.73
Accuracy:	0.64		

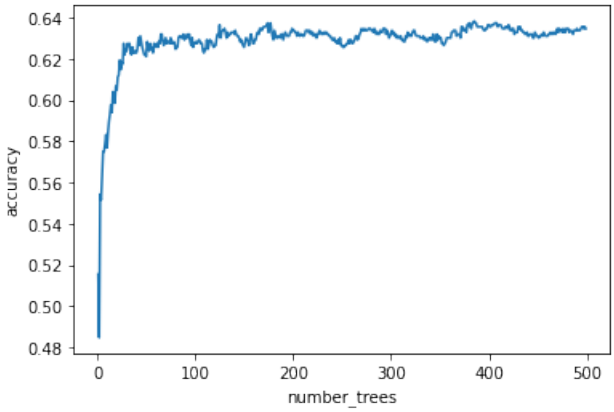
The random forest managed an accuracy of 64 percent. It does seem particularly weak when it comes to false negatives in the prediction of the enrolled students.

5.7. Random forest tweaking

Mark

To see if it could be improved on the random forest, some tweaking of the setting has been done. It was done by iterating through the settings, and capturing the setting with the highest accuracy score.

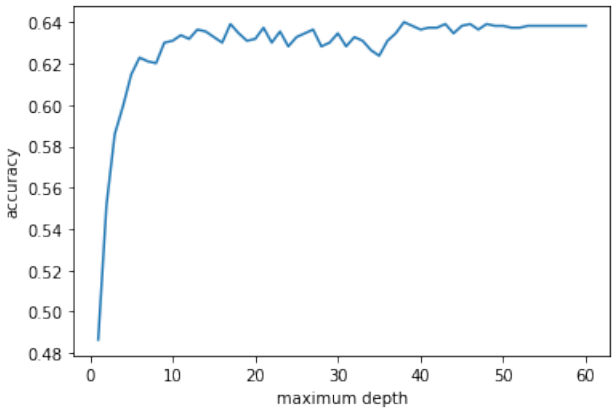
Figure 11
Accuracy scores with different number of trees



Variations to the number of trees. figure 11 illustrates the accuracy in the interval of 1-500 trees. Increasing numbers of trees were iterated over, though with greater intervals due to the considerable time required for processing. Diminishing returns come into force swiftly with additional trees. Accuracy rises sharply up until about 40 trees. At which point it becomes more like a random walk, possibly with a very slight positive trend. The top accuracy identified will likely vary with the data, and so the most accurate point is likely just a random effect. In the extended interval of 5000 trees, the pattern stayed the same. The peak accuracy was at 2000 trees, but the accuracy was actually slightly lower than that of tree number 385. Based on this, 385 trees were used.

Adjustments to the parameter 'criteria' were tested, such as setting it to entropy, but accuracy was reduced and as such the default arguments retained. .

FIGURE 12
Accuracy scores with different maximum tree depths



Further, alterations to the maximum tree depth were tested. It seems the deepest tree without any limits was 52. The accuracy results in table 12 are similar to when changing the number of trees. It rises sharply, and then bottoms out at around a depth of 15. The maximum accuracy recorded (although again seemingly random) is slightly above 64 percent, at maximum depth 38.

The resulting score was raised marginally, although so little that this could be attributed to a number of variables. Packages for tuning the model could have been utilised in order to tests many combinations of settings and as a result the model further. The best combination of settings is likely to vary from the best settings selected individually.

6. Evaluation of the models

Alicja, Mark

A summary of results showing the accuracy and mean f1-scores is shown in table 6. As shown in the table, the logistic regression got the highest accuracy score, followed by the random forest classifier. The KNN performed worst in this setting, although only five percent less than the best model. Every model seemed to struggle with a large share of false negatives of the enrolled students, perhaps due to their characteristics being hard to distinguish from both graduates, and dropouts. All models also seems to find it easy to spot the actual graduates, based on the high recall score. However, this probably mostly reflects the fact that this data is imbalanced, where most of the sample did, in fact, graduate. All models aside from the KNN produced very similar shares of type one and type two errors for dropouts, so the difference in the predictions actually seem to come down to the ability to predict graduates and enrolled students.

Table 6
Summary of results

	Accuracy	Mean of f1-scores
KNN	0.60	0.43
SVM	0.64	0.69
Logistic regression	0.65	0.70
Random forest	0.64	0.69

As the data used was also used by Martins *et al.* (2021), it is logical to compare outcomes. The logistic regression was most accurate model, at 65 percent, followed by random forest, at 64 percent. Martins *et al.* (2021) reached 68 percent accuracy with logistic regression, and 72 percent accuracy with their random forest algorithm. Based on this the model used by this project performs somewhat admirably with logistic regression, though accuracy could

be improved by tuning the random forest by altering hyper-parameters. The SVM model performed better than that of Martins *et al.* (2021), at four percent higher accuracy than their 60 percent, although this could potentially be attributed to random seeds which cannot be emulated here. A similarity to Martins *et al.* (2021) is the methods being prone to false negative predictions of the enrolled students. The models used seem to suffer more from this than those of Martins *et al.* (2021), although it is present in their methods. This could be a source for the lower prediction accuracy in this project, which could be considered upon making improvements to the model.

7. Conclusion

Rufus, Jakob

Overall the project could be considered a moderate success. Although improvements could be made to fine tune the performance of the algorithms used, the factors such as age, course and whether tuition fees are up to date could be utilised to provide targeted support in future to students at risk of dropping out of their course. A variety of algorithms were used to evaluate the effectiveness of each on the data set and the outcomes enabled more effective analysis of the data, meanwhile raising intriguing questions on the factors which might influence which learning tools might best suit a given problem. It can be concluded that the Logistic regression with the OneHot encoding and no feature selection makes the best predictions in regards to accuracy and mean of f1-score. The project has demonstrated that, despite some flaws, the use of Machine Learning to provide support for universities in this setting is both sound in theory and, with the gathering of a greater volume of data, holds great potential for improvements to education, personal fulfillment and wider society. Overall, it may be possible to predict student dropout with some success, although it is far fetched to imagine very high predictability of student decisions, based on data about performance and traits, as unobserved heterogeneity likely plays a major role in student

drop-outs.

8. Limitations and future work

Jakob, Rufus

To increase the accuracy of the predictions further, one could consider incorporating an ensemble technique, where multiple models are evaluated in unison. This will usually increase the flexibility and reduce the data-sensitivity of the model. Bagging and boosting are two of the more popular ones. The fundamental idea of them is same; Bagging is training and testing different models with different samples of the dataset and then uniting them so that the error will be the mean of the errors of the methods. Boosting however will connect the models subsequently, so that each model will learn from the previous one.

Acquiring and consolidating more data is a way to strengthen the algorithm further. By adding data from other universities in Portugal or other countries, the data would get more precise and representative for more people. Acquiring more data on the underrepresented data would also improve the accuracy on predicting the minority classes. Another solution to the imbalance is Re-sampling the dataset. This can be done either by under-sampling where some samples from the over-represented classes are removed or Over-sampling where samples from the underrepresented classes are added. The choice between the two depends on the size of the dataset, though oversampling is the most frequently used.

References

Aulck, L,. Velagapudi, N,. Blumenstock, J., and West, J.(2016). 'Predicting Student Dropout in Higher Education'. https://arxiv.org/pdf/1606.06364v4.pdf

Geron, A. (2019) 'Hands-on machine learning with scikit-learn, keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems'. 2nd ed. Sebastopol, CA: O'Reilly Media.

Martins, M,V., Tolledo, D., Machado, J., Baptista, L.M.T., and Realinho, V. (2021). 'Early Prediction of student's Performance in Higher Education: A Case Study.'

Keller, J. M., Gray, M. R., and Givens, J. A. (1985). 'A fuzzy K-nearest neighbor algorithm'. IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-15, no. 4, pp. 580-585, July-Aug. 1985, doi: 10.1109/TSMC.1985.6313426.

Peterson, L. (2009) 'K-nearest neighbor'. Scholarpedia, 4 (2) :1883

Serra, A., Perchinunno, P., Bilancia, M. (2018). 'Predicting Student Dropouts in Higher Education Using Supervised Classification Algorithms. In: , et al. Computational Science and Its Applications'. ICCSA 2018. Lecture Notes in Computer Science(), vol 10962. Springer, Cham. https://doi.org/10.1007/978-3-319-95168-3_2

Vanooteghem, J. (2020). 'Handling Categorical Data, The Right Way'. https://towardsdatascience.com/handling-categorical-data-the-right-way-9d1279956fc6)