# Machine Learning Coursework - OULAD Analysis

mbtj48

## 1 Introduction

Machine Learning and data gathering are paramount in order to produce great prediction models. This usually comes from analysis and evaluation of the dataset you are using & deciding how you are going to use the data. Then, building that data into a supervised learning algorithm, to make predictions for a given problem. Therefore I started through investigating the Open University Learning Analytics Dataset, trying to spot patterns and gather ideas for a model to develop.

## 2 Dataset Analysis

See below, the database schema for the OULAD.

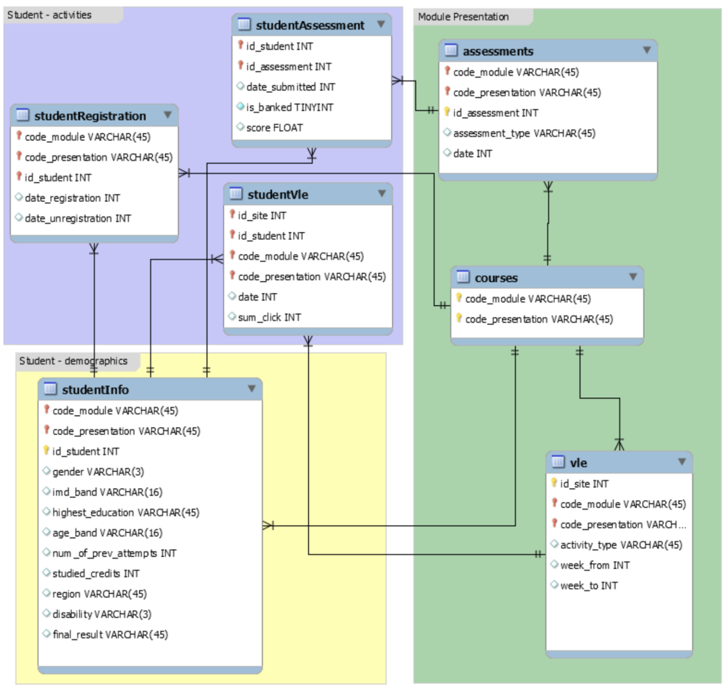

Figure 1: Dataset Schema

Firstly, I noticed a couple of very useful features, such as the score in the studentAssessment table, & the sum_click feature in the studentVle table. Therefore I started by grouping the sum_click and score attributs, thus finding the total number of times a person clicked within the portal through the year & their average mark. From this, I expected higher scores and click counts to perform higher as it is likely they put more effort in. N.b. This is shown in Table 1. From these, I plotted the data and noticed that it would be worth using a logistic regression model to test this. However, I noticed a poor performance from this data: 42% accuracy. Therefore I went back and added more data to my model, with the intention of using as much data as possible so the model can find patterns easier.

I noticed further useful data, like the date submitted feature; with this I can calculate how many days early a student submitted coursework. Ideally, one would expect more days before submission would result in a larger coursework mark as the student is more prepared and commited. In addition, I noticed, the assessments have a weighting, given in the assessments table, so I added data further through scaling the data based on the weight, ie their summative mark, as well as using their formative mark (where their weight is 0). I noticed this gradually improved the performance of my model, so I continued adding further data. I eventually included almost all of the data available, so I started to look at the data differently so I included the mean, mean absolute deviation, standard deviation, varience etc for the scores of students' coursework. This therefore can find patterns from extrapolated data to squeeze more patterns out of the schema. I then outputted the correlation matrix, the descriptions of the data & the pandas scatter matrix. These showed notable correlations:

| Feature | Correlation |
|---|---|
| assessment_var | -0.208126 |
| student_Std | -0.186952 |
| assessment_mad | -0.185294 |
| assessment_Std | -0.170608 |
| student_var | -0.169245 |
| is_banked | -0.163256 |
| studied_credits | -0.142413 |
| num_of_prev_attempts | -0.095372 |
| disability | -0.083723 |
| assessment_Meds | -0.070066 |
| student_mad | -0.069468 |
| gender | -0.028173 |
| age_band | 0.060998 |
| imd_band | 0.100306 |
| highest_education | 0.123420 |
| assessment_Mean | 0.145892 |
| formative | 0.162990 |
| daysEarly | 0.176952 |
| sum_click | 0.267279 |
| summative | 0.276203 |
| summativeNonScaled | 0.286314 |
| score | 0.356086 |
| student_Meds | 0.359332 |
| totalCoursework | 0.360867 |

Table 1: Correlations

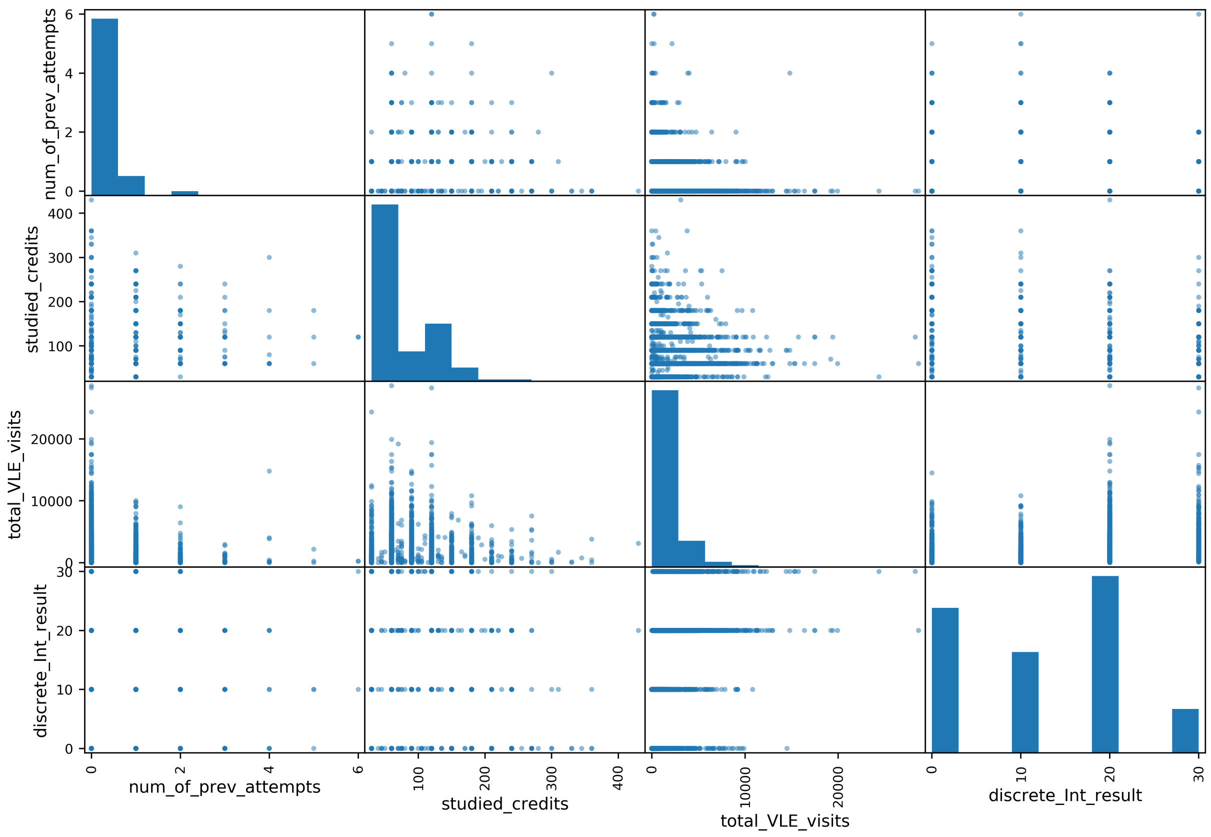

Figure 2: Data Scatter Matrix

The most surprising of these is the age feature; personally, I would have thought the correlation would have shown a strong negative correlation. However this may be because the age data values are estimated medians. To improve this correlation, I would need actual data rather than these estimates.

See the scatter figure in 2, which plots most data against itself, so people can spot patterns easier. Clearly, features such as score show a positive correlation. However it is hard to see the correlations against final result as it is a discrete value.

After data gathering, I looked at preprocessing the data, through using an imputer and normalisation. The imputer changes all NA values to the median of that feature. While the normaliser, changes the scale of all features to be within $0 - 1$, this prevents feature domination with large ranges and makes the features unit dependent Further, I exchanged certain features to their corresponding categorical codes, so the model can interpret the data.

## 3 Model Selection

The following phase was model selection. Here, I first split the data into train and test data this was a 75/25 split; then I tested the Linear, Logistic, SVR (Linear Kernel), SVR (Polynomial Kernel), Decision Tree, & Random Forrest Regression models, and compared how these performed in cross validation on the training data.

I noticed, the decision tree was the worst performing, with a mean accuracy of 0.15% between the cross validation folds, this therefore would be a poor model to pick. Next, the linear and SVR (linear kernel) regression models performed at 0.38% & 0.35% respectively. This is expected upon analysing the data, showing little features with a linear correlation. The SVR (polynomial kernel) performed a little better at 0.45% accuracy, but instead, I picked the best two performing models: the logistic & random forrest regression models. These had accuracies of 0.65% and 0.53% respectively.

## 4 Model A - Logistic Regression

After I picked logistic regression, I began my hyperparameter tuning. For this model, I decided to use a grid search to validate the best parameter within the permutations

I provided. I gave the model, two potential sets of permutations, the first, cycled through the C value, which is the regularisation strength; smaller values show a stronger strength, so I started with a strong logarithmic scale to check through, until after enough testing I used the range between 950 and 1125. It also cycled through the tolerance value with small values around the default of 0.0001. The other set of permutations loop through a similar set of values of C and adjusts the max number of iterations model uses to fit the data and find patterns.

## 5 Model B - Random Forest Regressor

Following my logistic hyperparameter tuning, I tuned my random forrest regressor, using a random search. This randomly picks n permutations to validate the model against based on the given parameters. I decided to check the number of estimators (trees in forrest), the max depth of each tree in the forrest, the minimum number of samples required at a leaf node, & the minimum number of samples required to split an internal node. As the random search prefers ranges, all of these values are provided with the python range type so it can pick any value within the range. This lead to picking 20 out of 244,800 permutations of parameters.

## 6 Conclusion

| | Logistic | Random Forrest |
|---|---|---|
| Max Err | 3.000 | 2.682 |
| Explained Var | 0.361 | 0.604 |
| Mean Abs Err | 0.393 | 0.405 |
| Mean Square Err | 0.513 | 0.309 |
| RMSE | 0.716 | 0.556 |
| Med Abs Err | 0.000 | 0.318 |
| r2 Score | 0.343 | 0.604 |
| Best Score | 0.678 | 0.590 |

Table 2: Metrics of Final Models

In conclusion, the logistic regression model finished with an accuracy of 0.68%, whereas the Random Forrest Model finished with an accuracy of 0.59%, therefore making the logistic model initially more desirable. Upon further inspection and validation on the testing data set, the maxiumum error was 0.3 lower for the random forrest, & the r2 score was 0.25 higher for random forrest, potentially making random forrest overall a better choice.

## Appendix

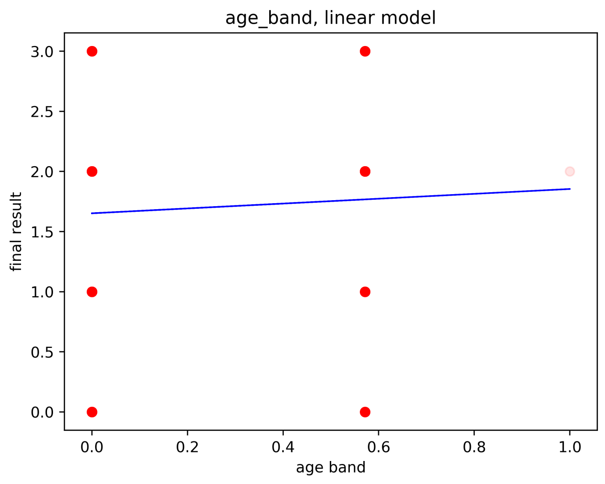

Figure 3: Age band against Final Result Linear Regression Model

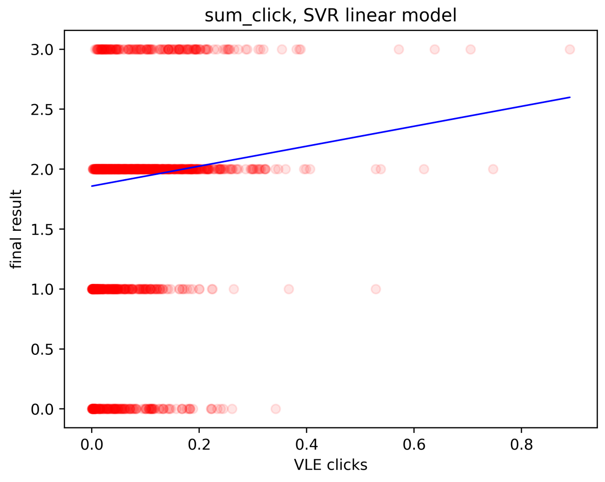

Figure 4: Total VLE clicks against Final Result SVR (Linear Kernel) Regression Model

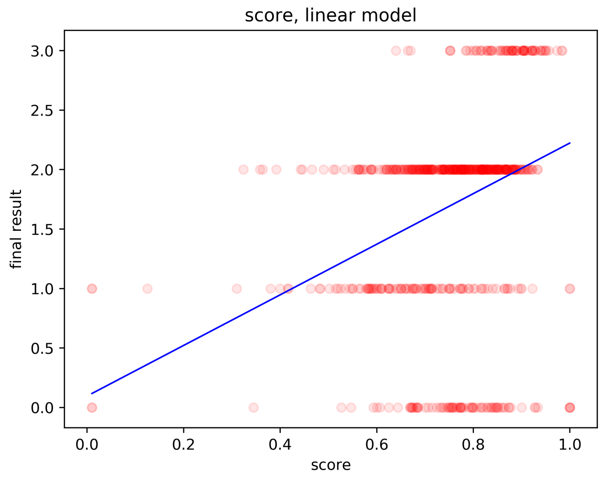

Figure 5: Score against Final Result Linear Regression Model