Comparison between Random Forests and Support-Vector Machines for Grade Prediction

Introduction

This report will compare the random forest and support-vector machine learning methods within the field of learning analytics using data from the OULAD [1] to predict students' final results by classification.

Chosen Methods

A random forest (RF) classifier produces a set of decision trees (DTs) using bootstrapped (randomly selected) data sets. When predictions are made, each DT *votes* on the outcome (where the outcome with the greatest number of *votes* wins), this has the advantage of smoothing out the inaccuracy produced by using only one DT.

A support-vector machine (SVM) generates a hyperplane to separate training data in hyperspace. While best suited to continuous data, as explained in Data Preparation, categorical features may be mapped to meaningful numerical features.

Experimental Procedure Plan

Once an initial feature set is selected, the dataset is split into a training and test set.

The processing, analysis and cross-validation of the methods are performed using Python with NumPy and Pandas for data manipulation, scikit-learn [2] for model application and matplotlib for data visualisation.

Finally, the models are tweaked by adjusting the feature set and hyperparameters to try decrease testing variance (and minimise training overfit).

Data Preparation and Feature Set

In order to use the models, the data must first be transformed into a set of input features and a corresponding set of input labels.

The OULAD dataset is provided as a set of *.csv* tables, the *studentInfo* table is the primary source of features (and prediction labels), the *assessments* and *studentAssessments* tables are processed and merged into *studentInfo* to produce a weighted score prediction per course-presentation-student.

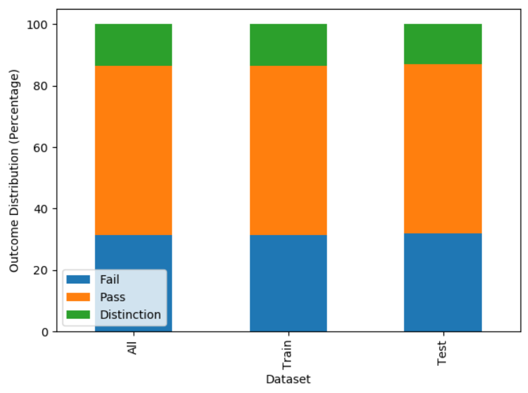
The feature pool (set of available features) $F = \{num\_of\_prev\_attempts, studied\_credits, predicted\_score, region, highest\_education, gender, disability, imd\_band, age\_band\}$.

To simplify data preparation and improve accuracy for both methods, only the valid *results* grade bands will be considered, thus the *Withdrawn* category as well as samples containing *null* or *NaN* values are dropped.

Since both models require all features to be numerical inputs, non-binary unordered categories (for example *region*) use one-hot encoding (one binary input feature per possible category, this can be computationally expensive for features with many categories). Ordered (and binary) categories (for example *gender*, *disability*, *highest_education*, *imd_band* and *age_band*) may use a direct numerical map (usually linear) to encode values.

To enable one round of cross-validation to be performed, the dataset must be split into training and testing subsets. Generally, the size of the training set is greater than that of the testing set. Here, an arbitrary fraction of 75% will be used for the size of the training set.

Graph 1



Graph 1 demonstrates the distribution of outcomes between the training and test datasets (of which there is less than one percent difference).

Performance of Methods

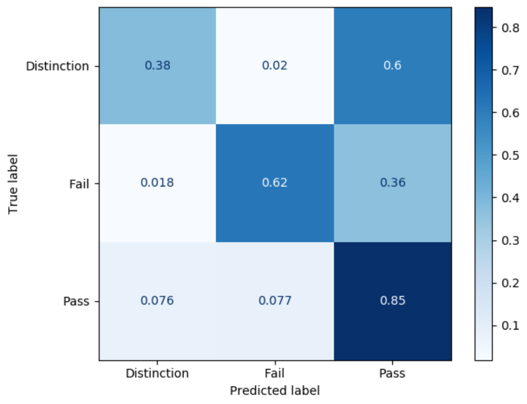
Classification accuracy score works out the fraction of correctly classified samples when a model is used to predict labels using the test dataset.

However, this score does not provide a full insight (as predicting a Fail outcome all of the time would produce a 30% accuracy for example) – normalised confusion matrices visualise the fraction of instances within the sample set where a true label is output as a given prediction label. The higher the values in the leading diagonal of the matrix (and the lower the values everywhere else), the greater the accuracy of the model.

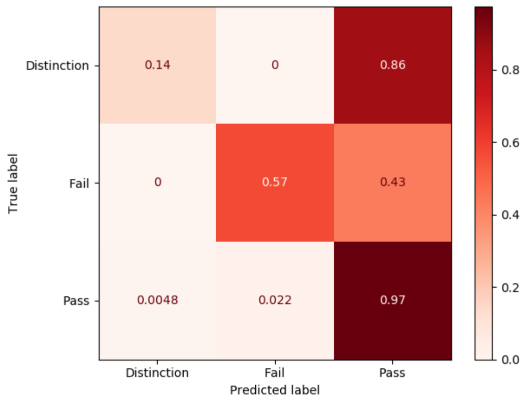
Here are the results for various subsets of *F* with default hyperparameters.

1. *{predicted_score}*

RF:
Accuracy: 0.7150

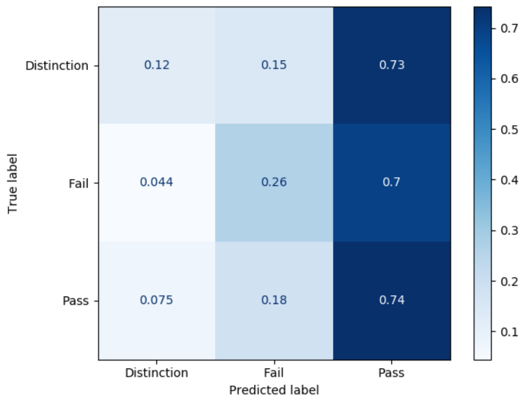


SVM:
Accuracy: 0.7382

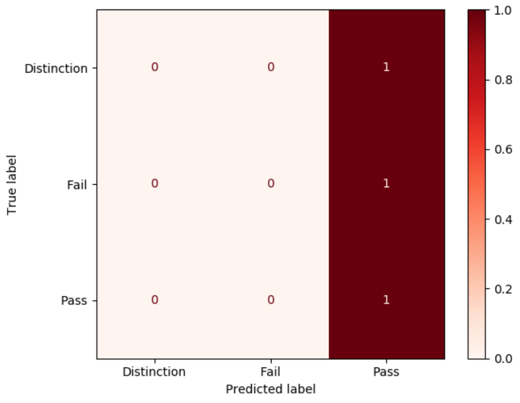


2. *F - {predicted_score}*

RF:
Accuracy: 0.5167

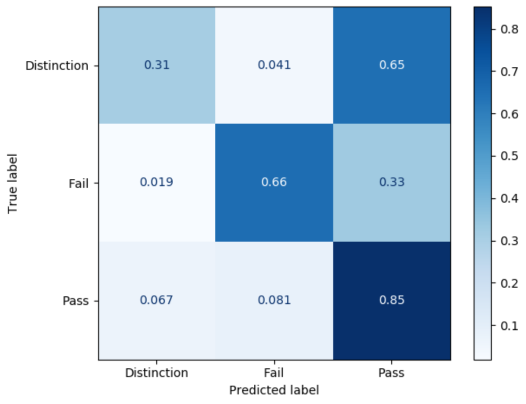


SVM:
Accuracy: 0.5738

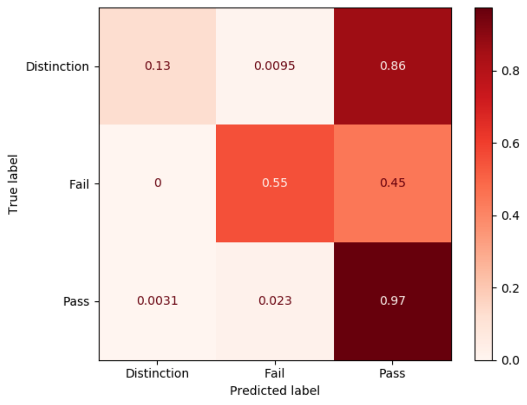


3. *F*

RF:
Accuracy: 0.7179



SVM:
Accuracy: 0.7329



The scikit-learn module provides a generic object-oriented interface for their machine learning methods, this means there is no apparent difference between the methods used when training and evaluating them (except for input parameters on instantiation).

Computationally, training the SVM takes noticeably longer (for the same training and feature set) than the RF.

This makes sense as, while both methods use polynomial-time training algorithms, for n_f features and n_s samples SVMs run in between $O(n_f n_s^2)$ and $O(n_f n_s^3)$ time [3] and balanced construction of a DT takes about $O(n_f n_s \log n_s)$ time [4] (implying a maximum time complexity for an RF with n_t trees of about $O(n_t n_f n_s \log n_s)$).

Hyperparameter Tuning

The main hyperparameter unique to RFs is the number of DTs, by default 100 are used, an increase yields diminishing returns and can decrease accuracy.

RFs also naturally inherit the hyperparameters from the underlying DTs. Maximum tree depth and minimum leaf samples are adjusted to keep generality and avoid over-fit.

Due to their random nature, the performance of RFs are heavily impacted by the random seed used, however, through testing on constant training and testing sets (with a random seed used during forest training), confusion matrix values tend to vary by about 1%.

SVMs also have several hyperparameters. The C value determines how strictly vectors must segregate the data (with diminishing returns). Multi-class data such as this can either be classified one-verses-rest or one-verses-one. The kernel projects the data into higher dimensions before classifying it allowing groups of a certain class *sandwiched* by that of another class to be correctly split.

Changing these hyperparameters discussed here from defaults did not result in any significant performance boost (but did negatively affect it in some situations).

Conclusions

Both RF and SVM tend to produce a significant number of false-positive passes. This is to be expected as a distinction grade is effectively a small sub-set of a pass (on the upper extremity).

While the SVM tends to have a higher accuracy score it generally has a worse distribution in the confusion matrix. The better accuracy score comes from the heavy weighting towards predicting a Pass (which is the most likely outcome).

In terms of the feature set, unsurprisingly the derived predicted score contributes significantly to producing the most accurate prediction results for both methods, notably the introduction categorical features have a greater effect on the RF than the SVM.

References

[1] J. Kuzilek, M. Hlosta and Z. Zdrahal, “Open University Learning Analytics dataset,” *Scientific Data*, vol. 4, no. 1, 2017.

[2] F. Pedregosa, et al., “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

[3] F. Pedregosa, et al., “scikit-learn SVM Documentation,” [Online]. Available: https://scikit-learn.org/stable/modules/svm.html.

[4] F. Pedregosa, et al., “scikit-learn DT Documentation,” [Online]. Available: https://scikit-learn.org/stable/modules/tree.html.