# Comparison between Decision Trees and Support-Vector Machines for Grade Prediction

## Introduction

This report will compare the decision tree and logistic regression machine learning methods within the field of learning analytics using data from the OULAD [1] to predict (by classifying) students’ final results.

## Chosen Methods

A decision tree classifier generates a tree structure when training, this is primarily useful when the input and output is categorical, which in this case it mostly is (dependant on the feature selection).

A support-vector machine 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.

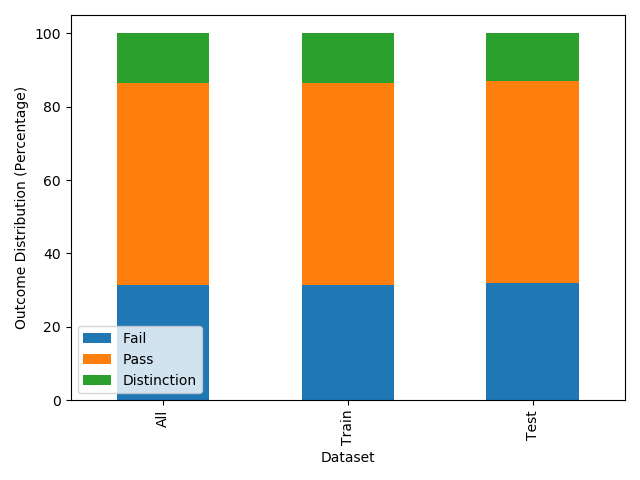
The OULAD dataset is provided as a set of *.csv* tables,

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preparation and improve accuracy for both methods, only the valid *results* grade bands will be considered, thus the *Withdrawn* category as well as samples containing *null* or *NaN* values are dropped.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 for one round of cross-validation, 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 will be used for the size of the training set.

Graph 1

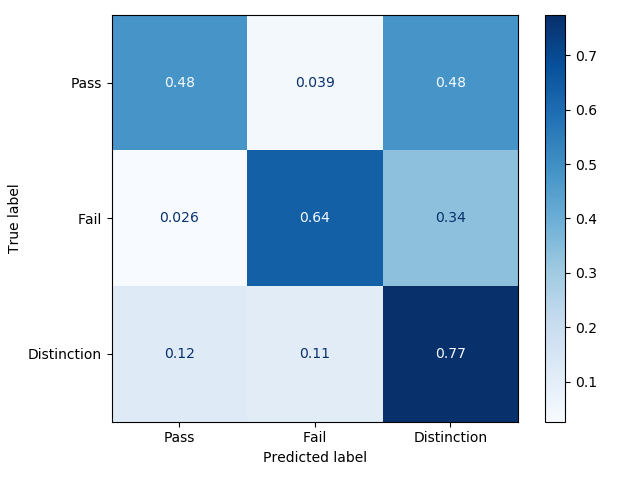
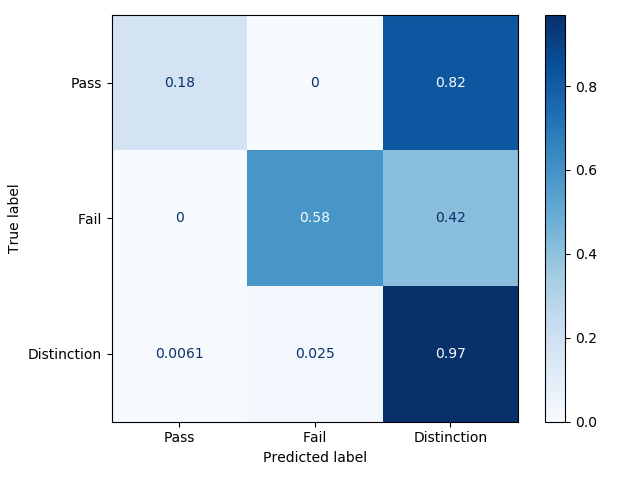


Graph 1 demonstrates the distribution of outcomes between the training and test datasets (of which there is less than one percent difference).

## 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. For this, decision tree yielded a score of (about accuracy) and support-vector attained (about accuracy).

However, this score does not show the whole story – 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.

## Hyperparameter Tuning

Support vector classifiers have several parameters. The value determines how strictly vectors must segregate the data. 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.

## Conclusions

There are many ways both the ML methods and experimental procedure can be tweaked to improve the results, both with a trade-off between computational time, bias, and variance.

Comparisons: Training, evaluation, prediction

# 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. |