Introduction

This report investigates and compares 2 Machine Learning methods to analyse the data from the OULAD dataset to ***predict a student's final result***. Since the data provided is ***labelled*** and ***discrete***, I have implemented the following Supervised Learning techniques, ***Logistic Regression (LR)*** and ***Random Forests (RF)***

Data Preparation

From OULAD I used the '***studentInfo.csv***' dataset. Since we're predicting the student's final grade I sorted and grouped the data by '***final_result***'. This resulted in a total of 32,593 unique entries categorised into 4 grades:

Final Grade	Distinction	Pass	Fail	Withdrawn
Total Entries	3,024	12,361	7,052	10,156

My next step in preparing the data was to reduce the number of categories down to two. This is because LR predicts a ***binary dependant variable***. I choose to grade the students into ***Pass***/***Fail*** absorbing ***Distinction*** and ***Withdrawn*** into them respectively.

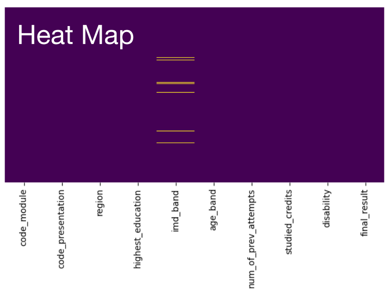
Final Grade	Pass (and above)	Fail (and below)
Total Entries	15,385	17,208

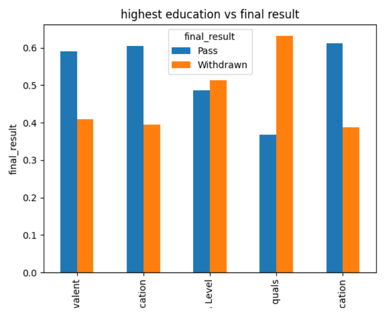
I removed the ***id_student column*** from the dataset since it was a ***unique-identifier*** and didn't have any relation to the students final grade.

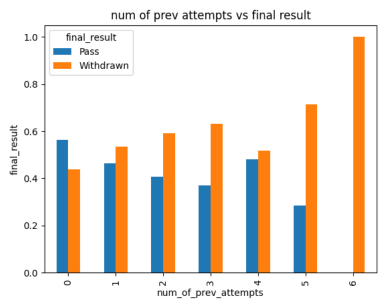
I produced a ***Heat Map*** of all the features in the dataset to highlight any missing data. From this graph, we can see there was only missing data in the ***imd_band*** category. I decided to eliminate the entries with missing data instead of removing the column.

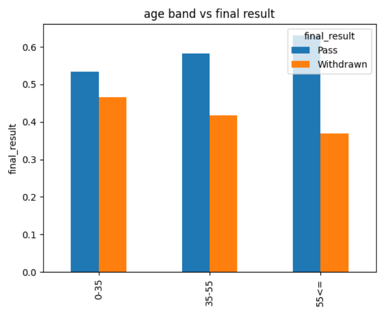
One-hot encoding was used to convert all categoric variables to a ***uniform numeric representation***. This included '***Pass***' and '***Fail***' being converted to ***1*** and ***0*** respectively.

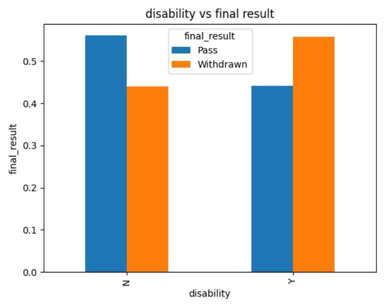
Below we can see some graphs which illustrate the data better. Some features have a more observable correlation to the student's final grade than others.

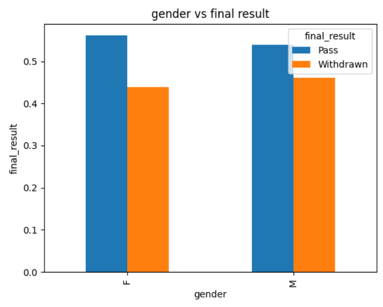












Logistic Regression

LR is a ***binary classifier*** that classifies categorical data with a ***binary dependant variable***. All data-points fit onto an 's' shaped "Logistic Function" which ***predicts the maximum likelihood*** of a student's final result.

The candidate ***log(odds)*** [1] values are calculated for each of the training data-points. Then we calculate the output candidate probability, ***p*** of the student passing by parsing the ***log(odds)*** as input into [2]. These are then projected onto a best-fit curve that can be used to extrapolate the student's final predicted grade.

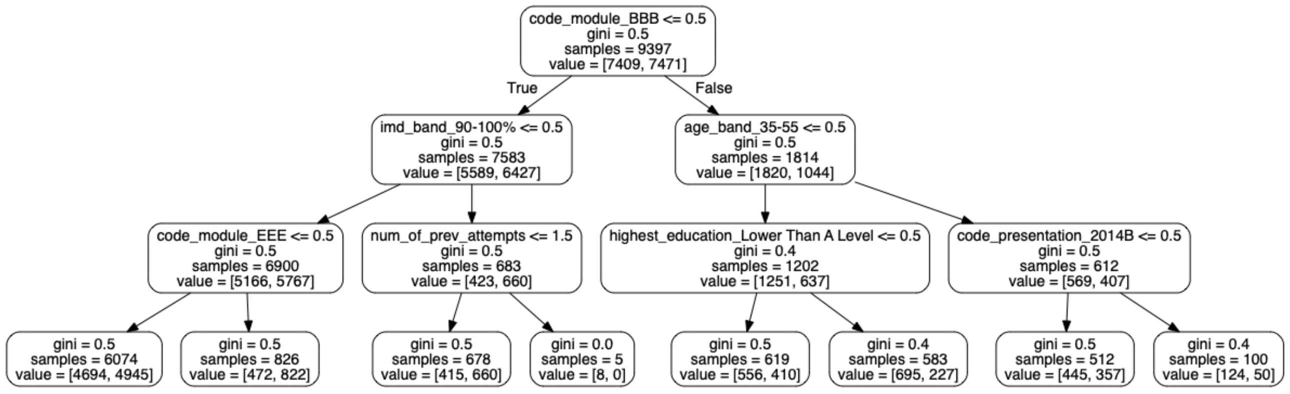
$$\log(\text{odds}) = \log(\frac{p}{1-p}) \quad p = \frac{e^{\log(odds)}}{1 + e^{\log(odds)}}$$

[1] [2]

Random Forests

RFs are based on decision trees that decide the best category to separate the data using the ***GINI Impurity***. With the difference being a variety of decision trees are produced opposed to a singular tree. This variety makes RFs far more effective than individual decision trees.

To create a random forest we use a ***bootstrapped dataset***. This is the same size as the original dataset but random non-distinct samples are selected. Then a decision tree is built using the bootstrapped dataset using a ***subset of variables***.



Decision Tree Depth=3

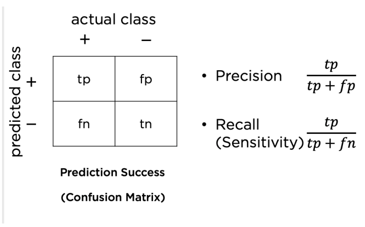
Comparison

I partitioned the data into a ***75/25 split*** of ***Training*** and ***Test data*** respectively. I discovered this provided the best results with enough data for the model to train and yet sufficient test data to be able to have confidence in the evaluation metrics. Testing with similar partitions i.e. 60/40 or 80/20 provided no significant improvement of performance.

Both models were trained on the training dataset partition with the label '***Final_result***' which held the student final grade data. With the rest of the columns being the features in the model.

I used two performance metrics to access the accuracy of each of my models and compare the effectiveness of my ***hyper-parameters***.

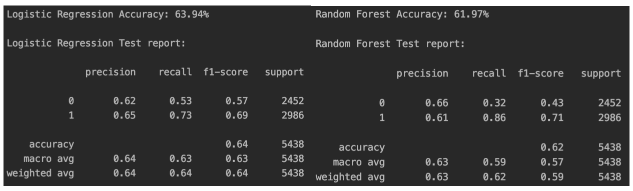
These metrics are ***precision*** and ***recall*** which are visualised using a ***confusion matrix*** which helps predict the success of the models. The closer these metrics are to ***1*** the better the model.


actual class
+
−
predicted class
+
−
tp
fp
fn
tn
• Precision $\frac{tp}{tp + fp}$
• Recall (Sensitivity) $\frac{tp}{tp + fn}$
Prediction Success
(Confusion Matrix)

Exploring different subsets of the dataset I found removing the ***gender*** category had a ***positive impact*** on the prediction ***increasing precision and recall.***

There were 1,910 more ***Pass*** data entries than ***Withdrawn*** which is a significant amount totalling a difference of 19.2%. I decided to experiment by ***equalising the amount of data entries by deleting excess data-points***. This was successful and ***provided more accurate predictions*** for both models and provided a more ***balanced dataset.***

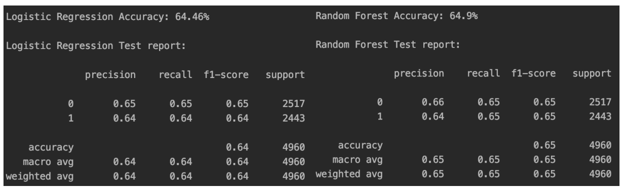
	Pass	Withdrawn
Entries before data-balancing	11,830	9,920
Entries after data-balancing	9,920	9,920



```
Logistic Regression Accuracy: 63.94%

Logistic Regression Test report:

              precision    recall  f1-score   support

           0       0.62      0.53      0.57      2452
           1       0.65      0.73      0.69      2986

    accuracy                           0.64      5438
   macro avg       0.64      0.63      0.63      5438
weighted avg       0.64      0.64      0.64      5438
```

```
Random Forest Accuracy: 61.97%

Random Forest Test report:

              precision    recall  f1-score   support

           0       0.66      0.32      0.43      2452
           1       0.61      0.86      0.71      2986

    accuracy                           0.62      5438
   macro avg       0.63      0.59      0.57      5438
weighted avg       0.63      0.62      0.59      5438
```

Before data-balancing



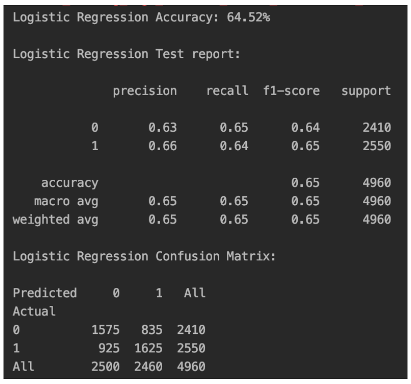
```
Logistic Regression Accuracy: 64.46%

Logistic Regression Test report:

              precision    recall  f1-score   support

           0       0.65      0.65      0.65      2517
           1       0.64      0.64      0.64      2443

    accuracy                           0.64      4960
   macro avg       0.64      0.64      0.64      4960
weighted avg       0.64      0.64      0.64      4960
```

```
Random Forest Accuracy: 64.9%

Random Forest Test report:

              precision    recall  f1-score   support

           0       0.66      0.65      0.65      2517
           1       0.64      0.65      0.65      2443

    accuracy                           0.65      4960
   macro avg       0.65      0.65      0.65      4960
weighted avg       0.65      0.65      0.65      4960
```

After data-balancing

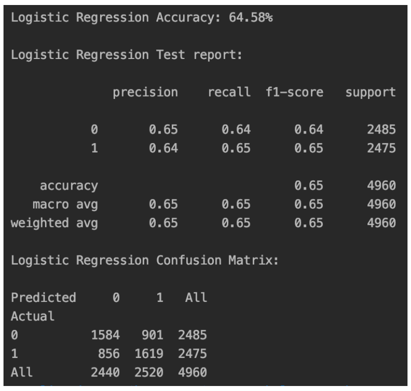
Here we can see the model accuracy is increased caroused the board for both models after data balancing. With a noticeable ***increase of 2.93% accuracy*** in regards to the RF model. Another noticeable difference is an increase of 0.33 in recall in ***withdrawn*** predictions, more than double the value before data balancing.

LR was fine-tuned with the ***hyper-parameter***: ‘iterations’. Here are 3 accuracy reports from 3 different parameters.



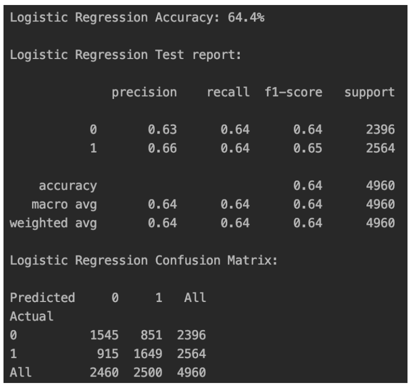
```
Logistic Regression Accuracy: 64.52%

Logistic Regression Test report:

              precision    recall  f1-score   support

           0       0.63      0.65      0.64      2410
           1       0.66      0.64      0.65      2550

    accuracy                           0.65      4960
   macro avg       0.65      0.65      0.65      4960
weighted avg       0.65      0.65      0.65      4960

Logistic Regression Confusion Matrix:

Predicted     0     1   All
Actual
0          1575   835  2410
1           925  1625  2550
All        2500  2460  4960
```

Iterations = 100



```
Logistic Regression Accuracy: 64.58%

Logistic Regression Test report:

              precision    recall  f1-score   support

           0       0.65      0.64      0.64      2485
           1       0.64      0.65      0.65      2475

    accuracy                           0.65      4960
   macro avg       0.65      0.65      0.65      4960
weighted avg       0.65      0.65      0.65      4960

Logistic Regression Confusion Matrix:

Predicted     0     1   All
Actual
0          1584   901  2485
1           856  1619  2475
All        2440  2520  4960
```

Iterations = 1000

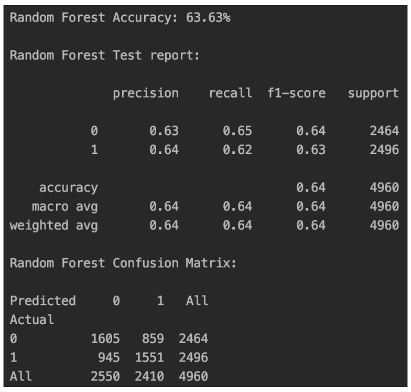


```
Logistic Regression Accuracy: 64.4%

Logistic Regression Test report:

              precision    recall  f1-score   support

           0       0.63      0.64      0.64      2396
           1       0.66      0.64      0.65      2564

    accuracy                           0.64      4960
   macro avg       0.64      0.64      0.64      4960
weighted avg       0.64      0.64      0.64      4960

Logistic Regression Confusion Matrix:

Predicted     0     1   All
Actual
0          1545   851  2396
1           915  1649  2564
All        2460  2500  4960
```

Iterations = 2500

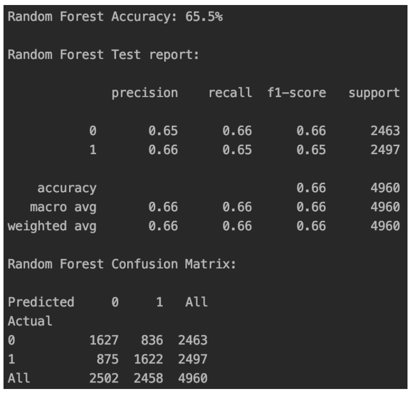
From my research, the most accurate predictions came from 1000 iterations. A larger increase from 100 however not much noticeable improvement of the ***execution-time trade-off*** from iterations above 1000.

RF models are initialised with 2 hyper-parameters: ***Number of trees*** and ***tree depth***.



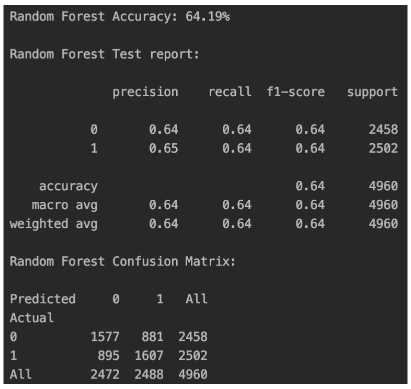
```
Random Forest Accuracy: 63.63%

Random Forest Test report:

              precision    recall  f1-score   support

           0       0.63      0.65      0.64      2464
           1       0.64      0.62      0.63      2496

    accuracy                           0.64      4960
   macro avg       0.64      0.64      0.64      4960
weighted avg       0.64      0.64      0.64      4960

Random Forest Confusion Matrix:

Predicted     0     1   All
Actual
0          1605   859  2464
1           945  1551  2496
All        2550  2410  4960
```

Trees=100 Depth=3



```
Random Forest Accuracy: 65.5%

Random Forest Test report:

              precision    recall  f1-score   support

           0       0.65      0.66      0.66      2463
           1       0.66      0.65      0.65      2497

    accuracy                           0.66      4960
   macro avg       0.66      0.66      0.66      4960
weighted avg       0.66      0.66      0.66      4960

Random Forest Confusion Matrix:

Predicted     0     1   All
Actual
0          1627   836  2463
1           875  1622  2497
All        2502  2458  4960
```

Trees=500 Depth=7

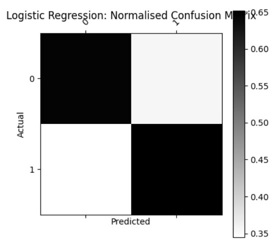


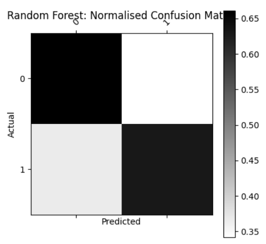
```
Random Forest Accuracy: 64.19%

Random Forest Test report:

              precision    recall  f1-score   support

           0       0.64      0.64      0.64      2458
           1       0.65      0.64      0.64      2502

    accuracy                           0.64      4960
   macro avg       0.64      0.64      0.64      4960
weighted avg       0.64      0.64      0.64      4960

Random Forest Confusion Matrix:

Predicted     0     1   All
Actual
0          1577   881  2458
1           895  1607  2502
All        2472  2488  4960
```

Trees=700 Depth=12

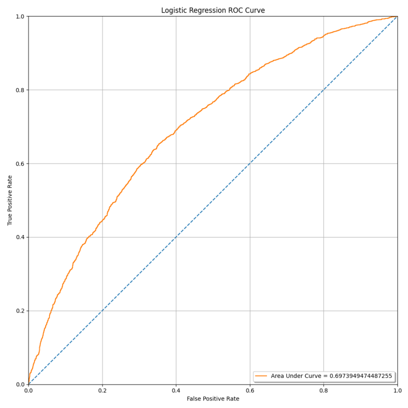
Here we can see in terms of ***precision*** and ***recall*** the trees=500 and depth=7 parameters is the best at predicting the student's final grade.

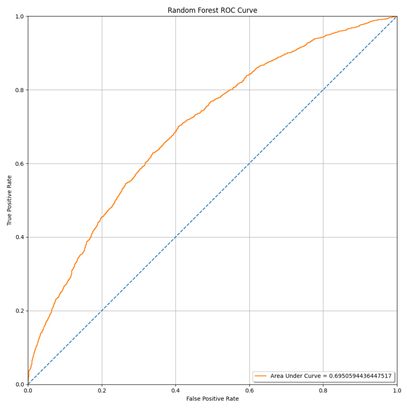
The ***normalised confusion matrix*** has a ***high number of true-negatives and true-positives*** which means the models are good at correctly predicting the correct grades to the true labels.





ROC Curves tells us how much the model is capable of distinguishing between classes. The greater the ***area under curve*** (AUC) the better a model is at predicting the correct final grade. Here both AUCs are ***above 0.5*** at ***around 0.7*** which is a reliable result.





Conclusion

Overall both methods use different approaches when classifying the dataset with LR using ***odds*** and RFs ***separating the data by categories***.

In conclusion, I found both methods to be very ***similar in terms of accuracy*** when used on the dataset. Yet RFs had the slight advantage of the data being slightly ***imbalanced*** and had ***multi-variable features***. Which when compared to LR that excels at ***linearly separable datasets*** with fewer features produced less correct predictions.