Predicting Final Grades of Portuguese Mathematics students using Student Data

Namita Chhibba (s3631442) & Soumyashree Giri (s3640845)

10 June 2018

Contents

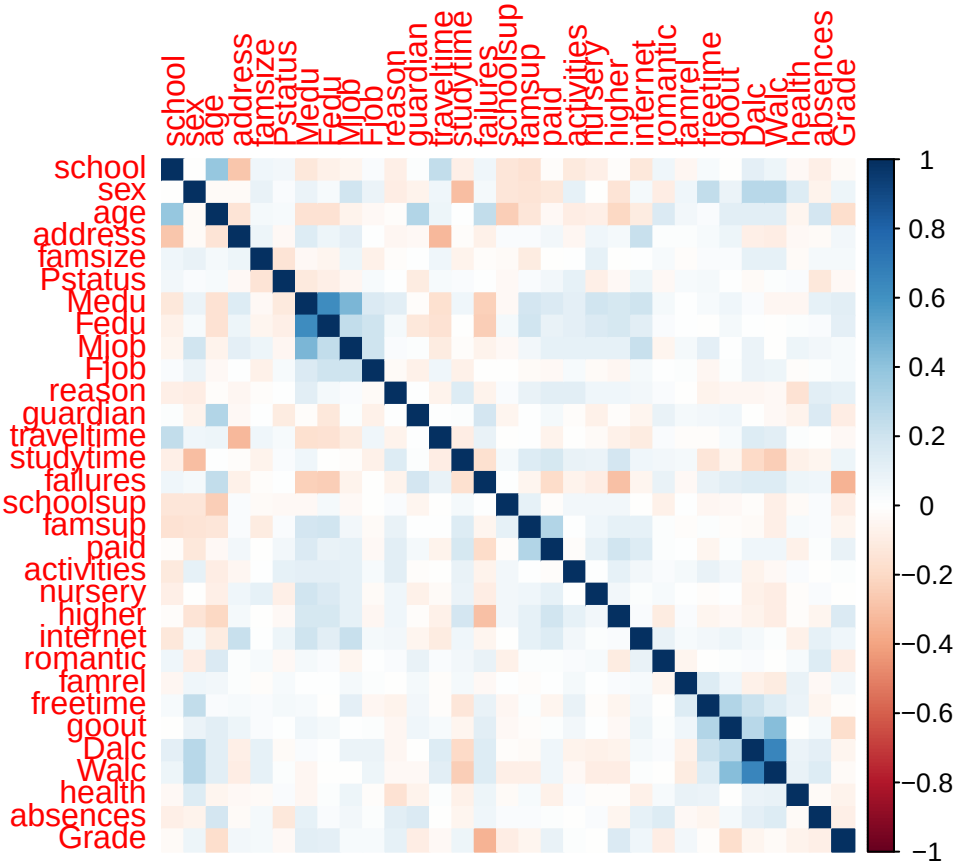
1 Introduction

The objective of this project was to build classifiers to predict the performance of students in mathematics in secondary education from the Student Performance dataset sourced from the UCI Machine Learning Repository.

Phase 1 of the project involved the cleaning of the data and analysing it to check which descriptive features have more impact on the target (Final Grade of student in mathematics). Further to get the clearer picture we built the correlation matrix to find the impact of different descriptive features on the target. And based on this analysis we categorized certain features as less granular than others. In Phase II of the project, we built three binary-classifiers viz. Naive Bayes, Random Forest and k-nearest neighbours, on the cleaned data.

Section 2 of the report gives an overview of our methodology. Section 3 describes fine-tuning process and detailed performance analysis of each classifier. Section 4 provides the comparison of the performance of the classifiers using the same resampling method. Section 5 critiques the methodology used. And in the final section we provide our conclusion.

1.0.0.1 The scatter matrix which we used to support our analysis in 1st phase to check the granularity of the descriptive features is as follows:

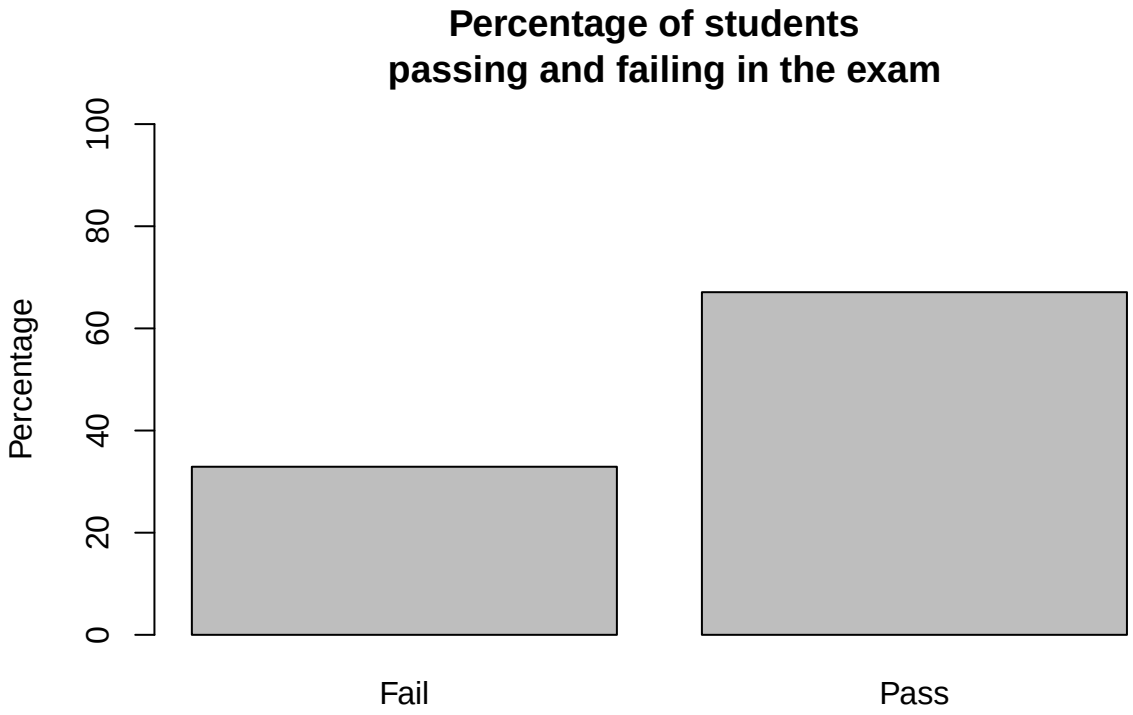


2 Methodology

We considered three classifiers - Naive Bayes (NB), Random Forest (RF), and K-Nearest Neighbour (KNN). It is critically important to develop a baseline of performance when working on a machine learning problem. A baseline provides a point of reference from which to compare other machine learning algorithms. For our

problem we had NB was the baseline classifier. Each classifier was trained to make probability predictions to enable adjustment of prediction threshold to improve the performance. The data was split into 80 % training set and 20 % test set.

Our target feature was not balanced as seen below so, we conducted five-fold cross-validation stratified sampling on each classifier. Stratified sampling was used to counter the imbalance in the class of the target feature. It ensures that every split reflects the distribution of target labels in equal proportion.



So our each set resembled the full data by having the same proportion of target classes i.e. approximately 67 % of students passing and 33 % failing.

Further, the optimal probability threshold for each classifier was determined. Predictions on the test data were made using the tuned hyperparameters and the optimal thresholds. We relied on mean misclassification error rate (mmce) for model training. Besides mmce, we used ROC curve and the confusion matrix on the test data for evaluation of classifiers' performance. The model implementation was done in R using mlr package.

3 Hyperparameter Tune-Fining

3.1 Naive Bayes

The NB classifier could produce some fitted zero probabilities as predictions because ofsome unsuspected excluded rare instances in the training set. So, to prevent this we smoothened the probabilities using Laplacian smoothing parameter. Weexperimented with values ranging from 0 to 30, using the stratified sampling. The optimal value for Laplacian parameter was 23.3 with a mean test error of 0.275.

```
## Tune result:
```

```
## Op. pars: laplace=23.3
## mmce.test.mean=0.2752976
```

3.2 Random Forest

For hyperparametric tuning in Random Forest we tuned mtry i.e. the number of variables randomly sampled as candidates at each split. For a classification problem, Breiman (2001) suggests mtry = p where p is the number of descriptive features. In our case, sqrt(p) = sqrt (21) = 4.58. We experimented mtry = 2, 3, 4 , 5 and 6. The result was 3 with a mean test error of 0.269. Other hyperparameters, such as the number of trees to grow were left set at the default value.

```
## Tune result:
## Op. pars: mtry=3
## mmce.test.mean=0.2692956
```

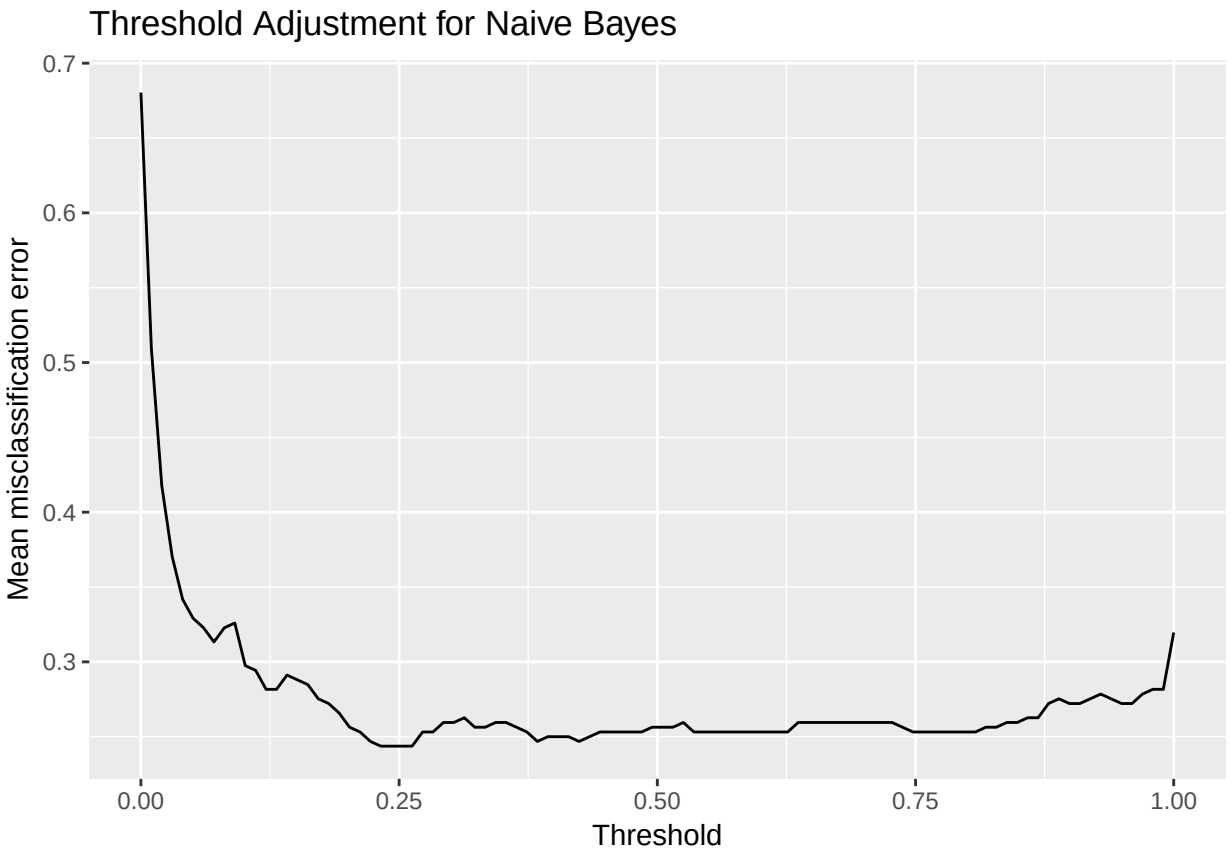
3.3 K-Nearest Neighbour

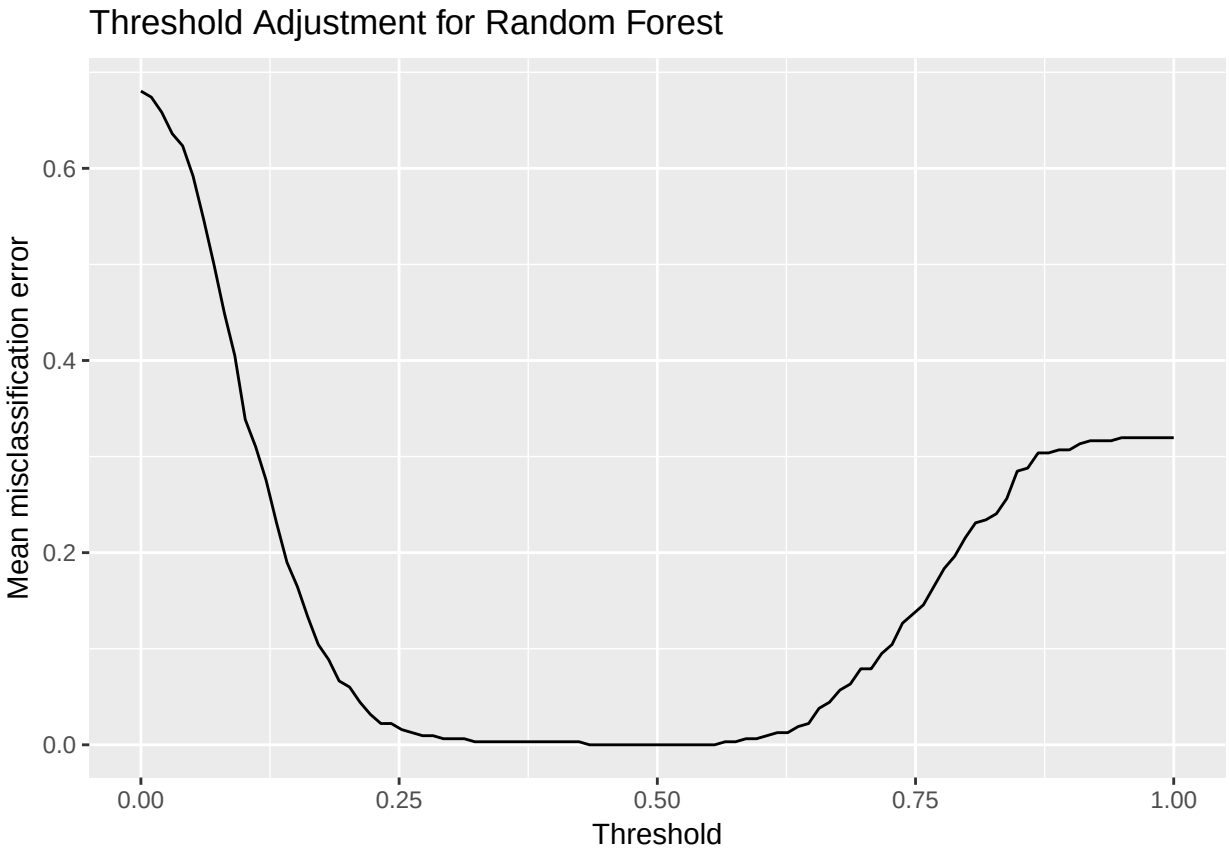
To determine the best value for 'k' in knn we ran a grid search fork = 2 to 20. The result was 20 and corresponding mean testerror was 0.278.

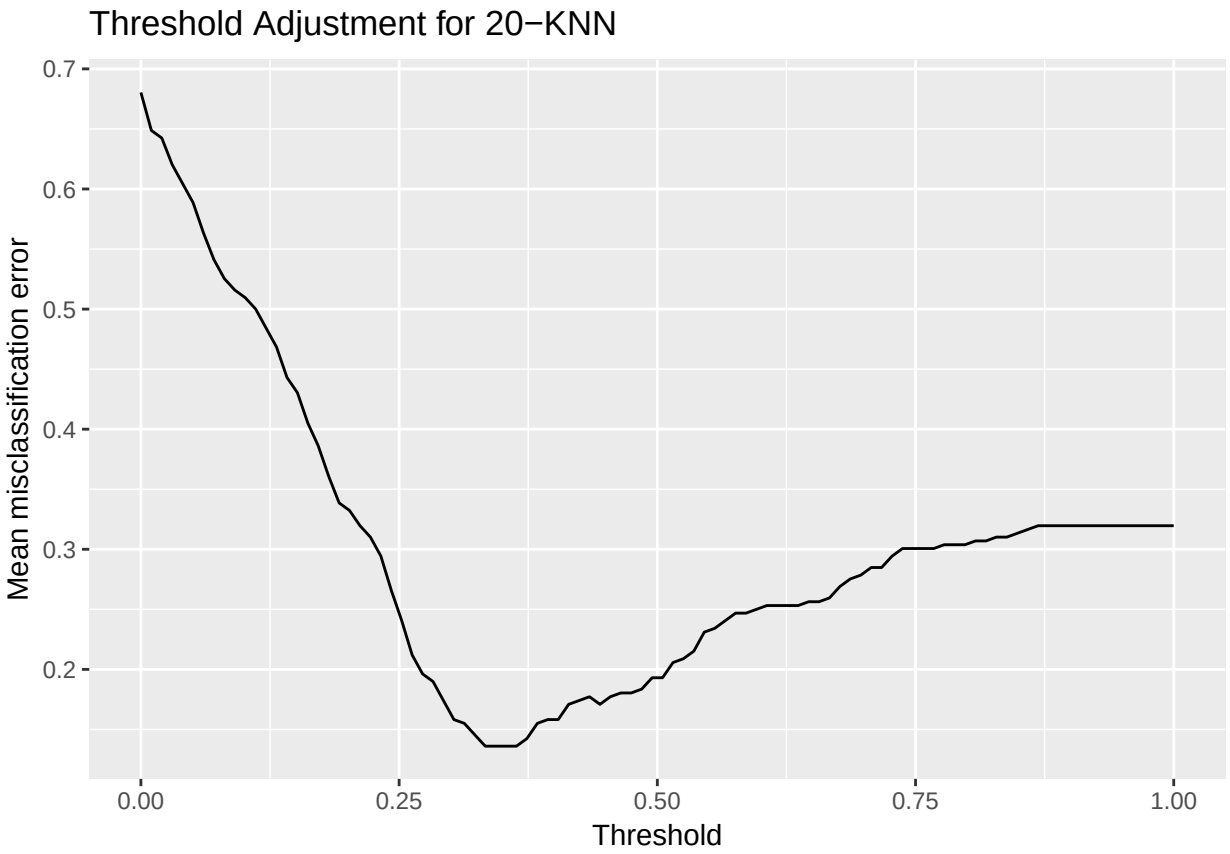
```
## Tune result:
## Op. pars: k=20
## mmce.test.mean=0.2785714
```

3.4 Threshold Adjustment

The plots depicting the value of mmce versus the range of probability thresholds are as follows. The thresholds values were approximately 0.232, 0.434, and 0.333 for NB, RF, and 20-KNN classifiers respectively. These thresholds were used to determine the probability of whether the student passes or fails in the mathematics exam.


Threshold Adjustment for Naive Bayes
Mean misclassification error
0.7
0.6
0.5
0.4
0.3
0.00
0.25
0.50
0.75
1.00
Threshold


Threshold Adjustment for Random Forest
Mean misclassification error
0.6
0.4
0.2
0.0
0.00
0.25
0.50
0.75
1.00
Threshold



We can generate the threshold value and determine the optimal threshold value which yields the best performance.

4 Evaluation

```
## Warning in predict.WrappedModel(tunedMod1, newdata = test_data): Provided
## data for prediction is not a pure data.frame but from class tbl_df, hence
## it will be converted.
```

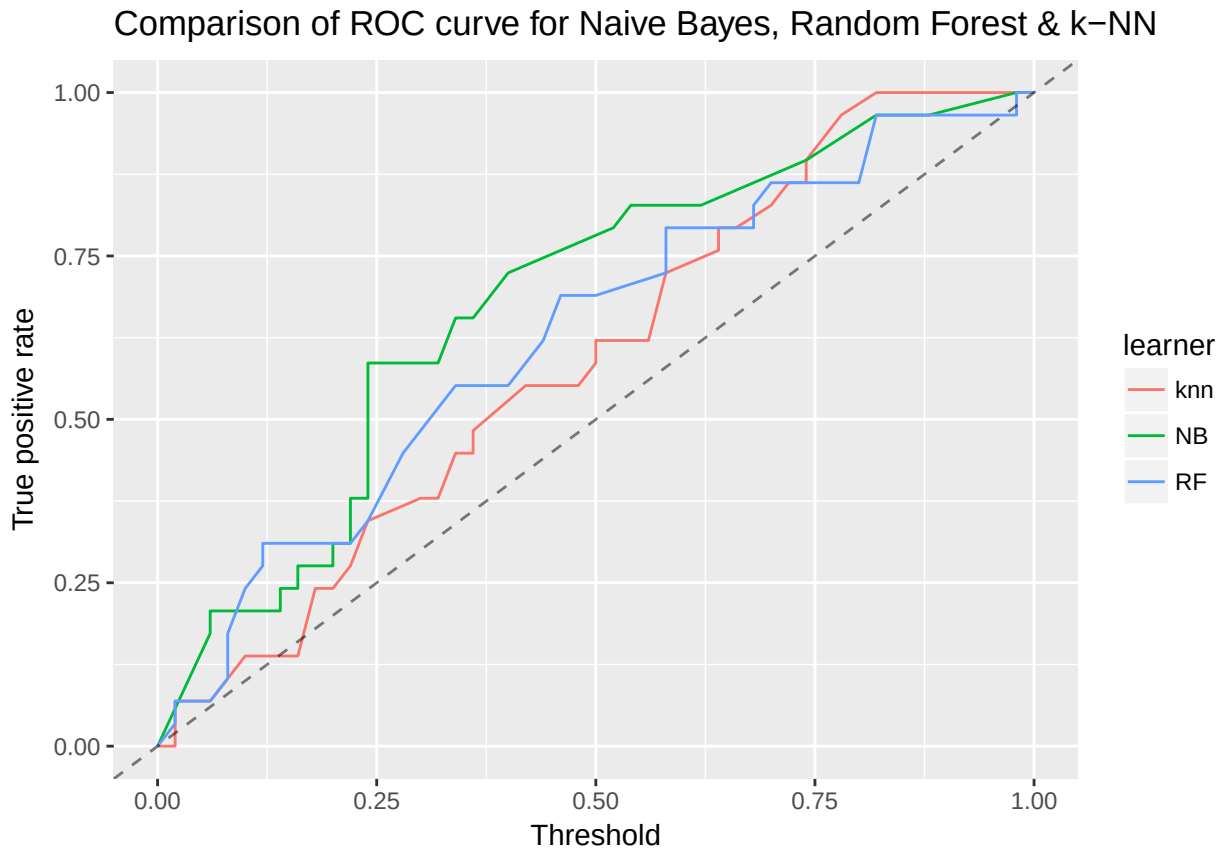
```
## Warning in predict.WrappedModel(tunedMod2, newdata = test_data): Provided
## data for prediction is not a pure data.frame but from class tbl_df, hence
## it will be converted.
```

```
## Warning in predict.WrappedModel(tunedMod3, newdata = test_data): Provided
## data for prediction is not a pure data.frame but from class tbl_df, hence
## it will be converted.
```

4.0.1 a) ROC Analysis

We shall utilise the threshold to perform ROC curve. Since we want to plot an ROC curve, we need to calculate the false and true positive rates (`fpr` and `tpr`).

```
d <- mlr::generateThreshVsPerfData(list(NB = testPred1, RF=testPred2, knn = testPred3 ),
                                   measures = list(mlr::fpr, mlr::tpr))
mlr::plotROCCurves(d)+ ggplot2::labs(title = 'Comparison of ROC curve for Naive Bayes, Random Forest & K
```



We see that the AUC is maximum for Naive Bayes classifier & minimum for K-Nearest Neighbours classifier. So as per AUC Naive bayes should be the suitable model.

4.0.2 b) Confusion Matrix

We calculated the confusion matrix using the parameters and threshold levels, for each classifier.

4.0.2.1 Confusion matrix of NB classifier:

```
## Relative confusion matrix (normalized by row/column):
##         predicted
## true     Fail      Pass      -err.-
##   Fail   0.38/0.48 0.62/0.32 0.62
##   Pass   0.24/0.52 0.76/0.68 0.24
##   -err.-      0.52      0.32 0.38
##
##
## Absolute confusion matrix:
##         predicted
## true     Fail Pass -err.-
##   Fail     11   18     18
##   Pass     12   38     12
##   -err.-   12   18     30
```

```
##      mmce        f1       acc       auc       tpr
## 0.3797468 0.4230769 0.6202532 0.6751724 0.3793103
```

4.0.2.2 Confusion matrix of RF classifer:

```
## Relative confusion matrix (normalized by row/column):
##         predicted
## true     Fail      Pass      -err.-
##   Fail   0.31/0.56 0.69/0.32 0.69
##   Pass   0.14/0.44 0.86/0.68 0.14
##   -err.-      0.44      0.32 0.34
##
##
## Absolute confusion matrix:
##         predicted
## true     Fail Pass -err.-
##   Fail      9   20     20
##   Pass      7   43      7
##   -err.-    7   20     27
```

```
##      mmce        f1       acc       auc       tpr
## 0.3417722 0.4000000 0.6582278 0.6255172 0.3103448
```

4.0.2.3 Confusion matrix of 20-KNN classifer:

```
## Relative confusion matrix (normalized by row/column):
##         predicted
## true     Fail      Pass      -err.-
##   Fail   0.48/0.44 0.52/0.32 0.52
##   Pass   0.36/0.56 0.64/0.68 0.36
##   -err.-      0.56      0.32 0.42
##
##
## Absolute confusion matrix:
##         predicted
## true     Fail Pass -err.-
##   Fail     14   15     15
##   Pass     18   32     18
##   -err.-   18   15     33
```

```
##      mmce        f1       acc       auc       tpr
## 0.4177215 0.4590164 0.5822785 0.5917241 0.4827586
```

Based on AUC NB is the best model and based on accuracy score Random Forest is the most suitable model. Apart from that, all the three classifiers could more accurately predict the students who passed better than the students who failed in the examination. The class accuracy difference was considerable. Based on mmce and class accuracy, we identified RF classifier as the best model.

5 Discussion

We saw that none of the classifiers performed accurately in predicting the failing students equally well as passing students. This indicates that the imbalance class problem existed even after stratified sampling.Alternative approach would be under- or oversampling to adjust the class balance, despite the risk of inducing biases.

The RF came up as the best model because of the bagging mechanism (i.e. 500 trees) which provides it a better accuracy. Multiple bagged models were run at each iteration during the resampling.

Naïve Bayes was the second best model based on AUC. The assumption for NB model is that the descriptive features follow normality, but this is not always true. The Transforming on numerical features would solve this issue. High AUC and decent accuracy attained by NB can be associated with the fact that our data is labelled and limited, so using high bias and low variance classifiers like NB works well over low bias and high variance classifiers like example k-NN.

KNN classifier might not be appropriate given that there were many categorical features in the data.

Since, in our problem it's more imortant to identify the students who fail (True positives) at all thresholds, so AUC is the better measure of total accuracy. And as per that Naive Bayes is the best model for the given problem.

6 Conclusion

For our classification problem, Naive Bayes performed best because of its ability to get higher true positive rates at all threshold besides being able predict the Grade of students with high accuracy. To begin we split our data into training and test sets. We figured the optimal value of the selected hyperparameter of each classifier and the probability threshold using stratified sampling. Even after this the class imbalance persisted and minimized the class accuracy of the students failing, more than that for those passing. For future improvements, we recommend considering under sampling or over sampling methods to counter the class imbalance.

7 References

Bischl, Bernd, Michel Lang, Lars Kotthoff, Julia Schiffner, Jakob Richter, Erich Studerus, Giuseppe Casalicchio, and Zachary M. Jones. 2016. "mlr: Machine Learning in R." Journal of Machine Learning Research. Breiman, L. 2001. "Random Forests." Machine Learning.

https://machinelearningmastery.com/estimate-baseline-performance-machine-learning-models-weka/ https://rstudio-pubs-static.s3.amazonaws.com/240657_5157ff98e8204c358b2118fa69162e18.html https://www.researchgate.net/post/What_are_the_factors_that_must_be_taken_in_consideration_when_selecting_a_machine_learning_classifier_algorithm https://stackoverflow.com/questions/34074859/can-knn-be-better-than-other-classifiers http://blog.echen.me/2011/04/27/choosing-a-machine-learning-classifier/ https://www.quora.com/Which-is-more-robust-to-noisy-data-a-Decision-Tree-or-Naive-Bayes