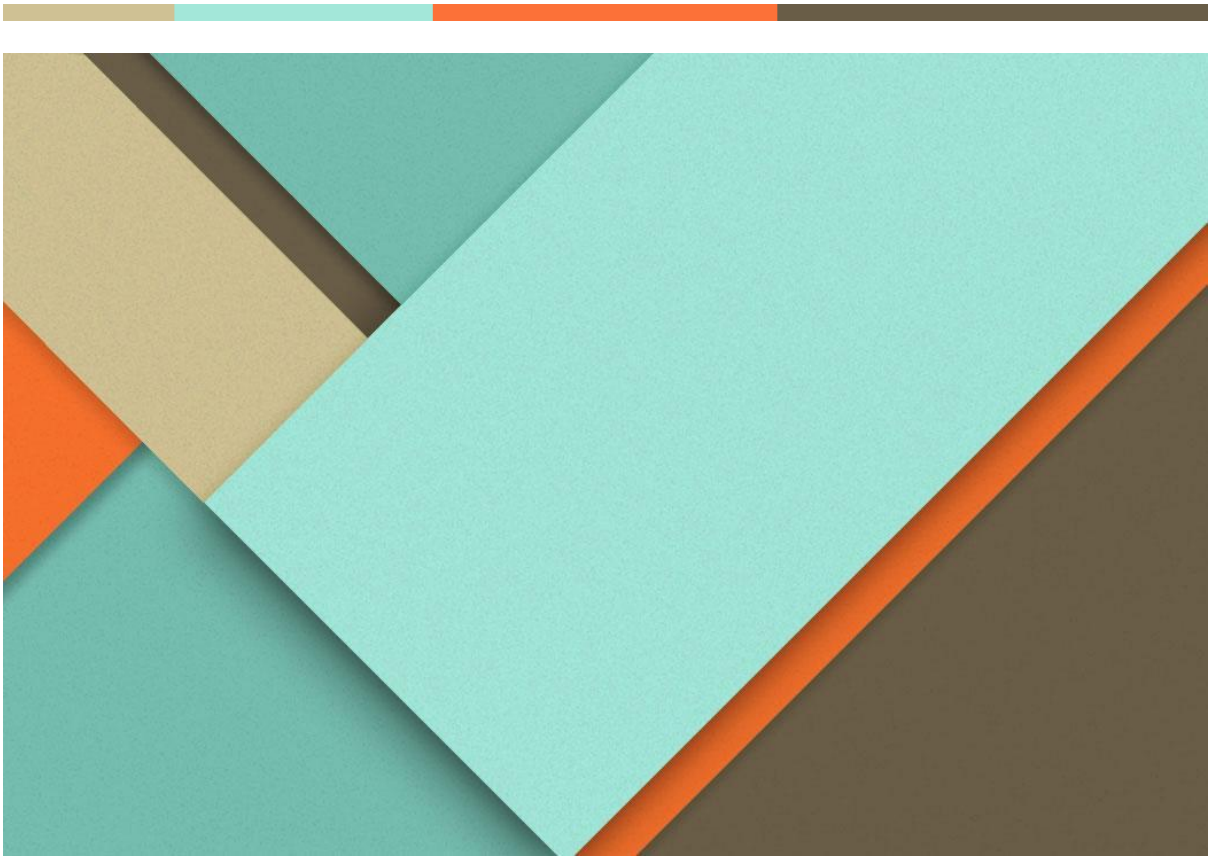


Advanced Data Science for Innovation - Assessment 1 Part D Report

03/03/2021

Link to Github repository - https://github.com/Reasmey/adsi_group2leftfeet

Team Members:
Reasmey Tith, 10845345
Charles Vanderbilt 11210325
Kevin Shao 12964235

Table of Contents

1 Business Understanding **2**
1.1 Objective 2
1.2 Project scope 3

2 Data Understanding **3**

3 Data Preparation **5**
3.1 Data cleaning 5
3.2 Data preprocessing 5
3.3 Feature engineering 6

4 Experimental Approach **7**
4.1 Addressing the imbalanced dataset 8
4.2 Choice of classifier 8
4.3 Hyperparameter tuning 8

5 Modelling **9**
5.1 Best models implemented 9
5.2 Results 9

6 Discussion and Issues **13**
6.1 Data quality issues 13
6.2 Modelling Issues 13

7 Evaluation **13**

8 Recommendations and Future Scope **14**

9 Appendix **15**

10 Member Contribution **18**

Predicting NBA Career longevity

1. Business Understanding

1.1 Objective

The objective is to build an accurate predictive model that will predict if a rookie player will last at least 5 years in the NBA league based on their prior game stats. If we can predict long term stability in a player, the investment in their career will carry less risk. This prediction

can be used by the business to make a more informed decision during the rookie selection process.

The approach to the investigation involved:

1. An exploration of the historic data provided by the NBA,
2. Finding the best algorithm to model the data with associated processes (i.e. data-preprocessing, selection of hyperparameters etc.), and
3. Evaluation of the model's performance by assessing the predicted outcomes from the model against the true outcomes.

This modelling can be viewed as a binary classification problem, where we are interested in predicting one of two outcomes (classes):

- A player has a career in the NBA of greater than or equal to 5 years (Class = 1).
- A player has a career in the NBA of less than 5 years (Class = 0).

1.2 Project scope

A team of three data scientists were involved in the project. All analyses and modelling was conducted in Python 3.7, in conjunction with other cost-free tools such as Github and Docker for project artifact management. Each data scientist worked simultaneously over a three week period to find the most accurate model by trying and testing different algorithms, data pre-processing techniques and hyperparameter tuning where necessary.

The best model from all experiments was determined by the highest achievable Area Under the Receiver Operating Characteristic Curve (AUC) score. This score is a measure of the model's ability to predict the correct class whilst minimising incorrect predictions, and the higher the score, the more accurate the model. The success of the project is determined by achieving a sufficiently high AUC score, which we expect to be around 0.8.

If we cannot obtain a model with an acceptable AUC score, we can still estimate a probability that a player will last 5 years, and a rank ordering of players can be made.

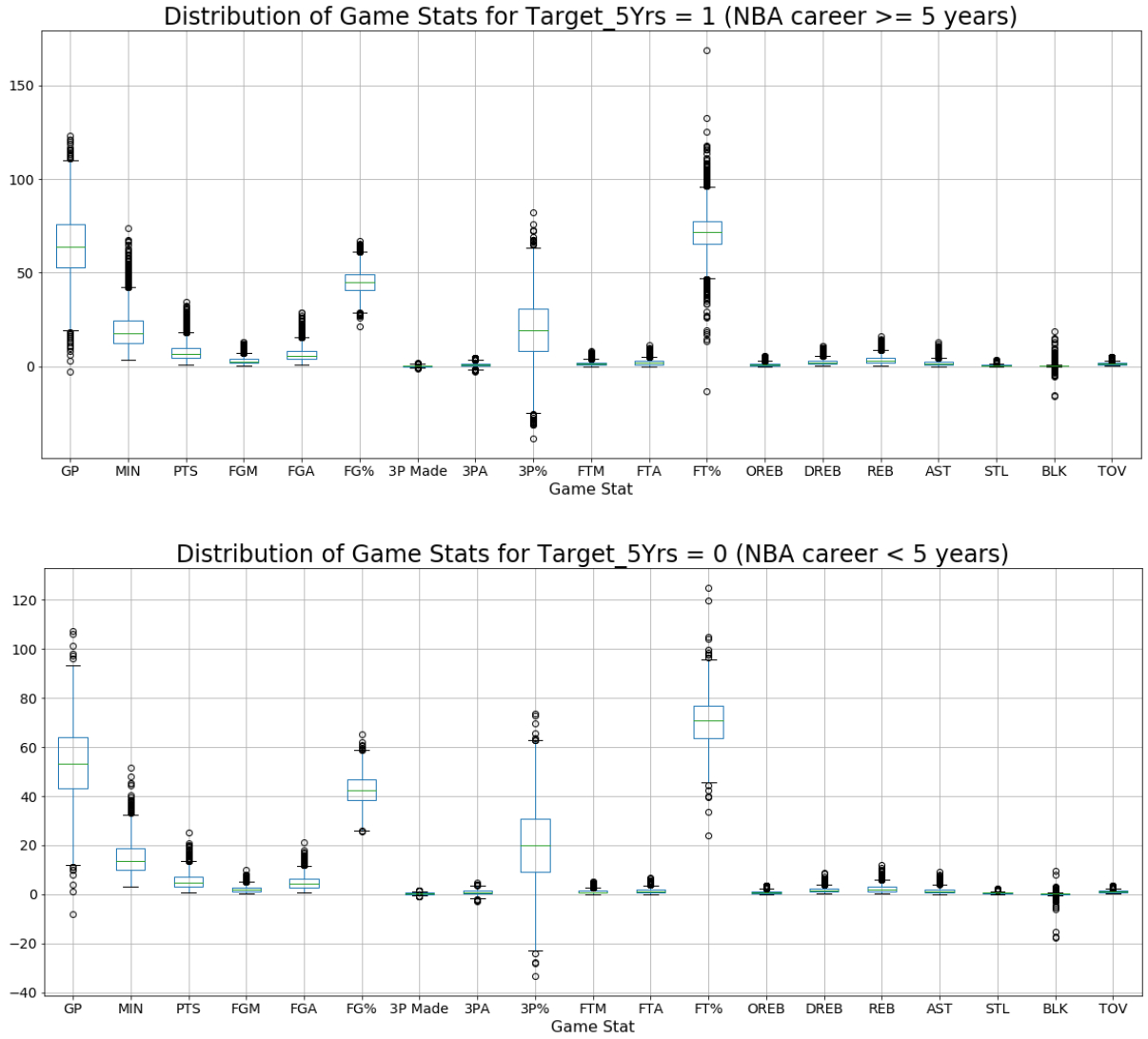
2. Data Understanding

The dataset of historical game stats consisted of 11,799 unique player identification numbers, 19 numeric variables representing a player's performance in various elements of the game, and 8000 players had a class label indicating whether they had a career length of >= 5 years (1) or less than 5 years (0). A full data dictionary can be found in Table A in the appendix.

When looking at the distribution of game stats for each class (Figure 1), we can see several variables had values below zero. Contextually, this is not a plausible observation as it is impossible to have negative games played (GP) for instance, or a negative average free throw attempt rate (FT%). There was also some inconsistency where we expected the values of the "%" variables to be the ratio of "made" variables to "attempts" variables, but upon a sense

check they were not calculated in that manner. This indicated that there were issues with the validity of the data, most of which were able to be addressed before modeling (described in section 3.2).

Figure 1 - Boxplots of game stats variables by class

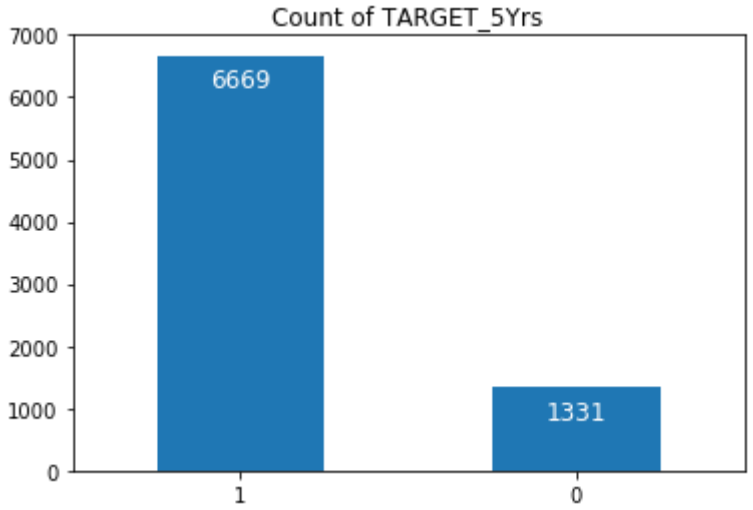


Another other notable trend is the similarity of the distributions of variables between classes. The closeness of values here made it more difficult for a model to distinguish between the two classes, which was confirmed by the large amount of experiments conducted in order to ascertain a single, best performing model. The average (arithmetic mean) across all stats was slightly higher for class 1 than for class 0, hence this was exploited when creating new features.The data preparation and modelling experiments section elaborates on this further.

There was also an imbalance of classes, with the majority of players belonging to class 1, having lasted at least 5 years in the NBA (Figure 2 below). Usually this imbalance would impact the sensitivity of a model to correctly predict the minority class as it is “overpowered”

by information from the majority class, therefore we implemented some data resampling techniques as part of the experimentation.

Figure 2 - Class imbalance in the training set



3. Data Preparation

3.1 Data cleaning

The raw data was in a relatively clean state upon receipt and consisted of one table with 8000 observations reserved for training, and another table of 3799 observations for testing. There were no missing values in the data, and since the variables were all numeric there was no one-hot-encoding necessary.

For some experiments, the negative observations were treated by either taking the absolute value, or converting to zero. The former was performed as a trivial way of removing negative observations, and the later was done as an alternative so as to not artificially inflate the stats of a player.

3.2 Data preprocessing

Various techniques were used to pre-process the raw training data, depending on the line of experimentation chosen. A list of all techniques used are summarised in the table below.

Table 1 - Summary of data preprocessing

Technique	**Transformation**	**Justification**
Standardisation	Standardise features by removing the mean and scaling to unit variance	Some classifiers required standardised values in order to lessen the effect of outliers
Adjusting Weights	Setting a weight parameter to scale the model weights by the ratio of negative to positive class	Balances the weights of each class e.g. in xgboost to counteract the bias towards the majority class
Upsampling	Increase the observations of the minority class by copying with replacement	To provide a balanced dataset of equal amounts of each class
Downsampling	Decrease the observations of the majority class by removing them	To provide a balanced dataset of equal amounts of each class
Synthetic Minority Oversampling Techniques (SMOTE)	Create synthetic observations based on existing observations	To provide a balanced dataset of equal classes, and/or oversampled minority class to a predefined ratio

Once a technique was applied, the training data was further split into two sets of 80% training and 20% set aside for validation. The same preprocessing was then applied to the testing set before making predictions.

3.3 Feature engineering

As we mentioned in Section 2, there were some data validity issues which we sought to rectify by creating new variables, or features, that may be better indicators of a player's performance. By recalculating the percentage variables, and creating additional combinations of stats, we hoped to further improve the separation of observations between classes, hence improving model performance. Many of the new features were based on logical combinations of stats that would give a more holistic view of a player, or represent other elements of the game that were not already present such as offensive ratings and defensive ratings, by combining these stats to create a more powerful or predictive feature which can reduce autocorrelation that causes models to overfit. There was also a trend of the mean of all stats for class 1 being slightly higher than class 0, so a combined average for each Id over all stats was computed (after standardising all values first).

Table 2 - Summary of new features created during experimentation

Feature	Data Type	Description
mean_stats	numeric	Row wise mean (after standardisation) of all stats per player
FG Rate	numeric	FG% x GP x MIN. A rate of field goals by number of games and minutes played
3P Rate	numeric	FG% x GP x MIN. A rate of 3 pointers by number of games and minutes played
FT Rate	numeric	FG% x GP x MIN. A rate of free throws by number of games and minutes played
Offensive	numeric	1. OREB+REB+AST. An additive score of offensive moves 2. FGM+3PM+FTM+OREB+AST+STL. An additive score of offensive moves
Defensive	numeric	1. DREB+STL+BLK. An additive score of defensive moves 2. DREB+BLK. An additive score of offensive moves
FG%	numeric	Recalculated values derived from dividing FGM / FGA
3P%	numeric	Recalculated values derived from dividing 3P Made / 3PA
FT%	numeric	Recalculated values derived from dividing FTM / FTA
Total Minutes on Court	numeric	GP x MPG. Multiplying total games played with average minutes on court gives us total minutes on court

Although some of these features were found to be more influential than the original variables in some model experiments (see Appendix Figure A), the overall performance of those models were fairly poor with AUC scores of less than 0.6. Possibilities to further extend on this topic are suggested in Section 8.

4. Experimental Approach

The selection of modelling techniques are based on the continual feedback of the AUC score upon submission. Once we see improvements of the AUC score on a particular experimentation, we iterate organically to focus on the model variations that give us better results.

The approach of our modelling revolves around the CRISP-DM guide where after we evaluate a particular model through the AUC score, we go back to the business objectives, data understanding, and preparation where every step we analyse and incorporate our insights from previous iterations. Then, we form the hypothesis we want to test for current iterations, put it into modelling, evaluate, and the cycle begins again.

4.1 Addressing the imbalanced dataset

As mentioned in Section 3.2, a number of resampling techniques were applied to the training dataset in order to better balance the amount of observations for each class. We know that if a dataset is not balanced, there is a bias towards the majority class when implementing a model. We experimented with three methods mentioned in the above table, Upsampling, Downsampling, and SMOTE.

Even with our findings from the EDA, we believed we could gain some improvement to a model by resampling class 0 observations. For a majority of the experiments we implemented, we found marginal improvement for some classifiers over using the raw dataset. We suspect that the replication of class 0 observations, or generation of similar observations just increased the noise around the barrier between classes, rather than provide a clearer distinction for the model to learn.

4.2 Choice of classifier

We used a variety of models in our approach such as LogisticRegression, SVM and XGBoost. These models were chosen as a starting point to explore some out of the box settings to compare to a null classifier[1]. They were also easy to implement, and we could quickly gauge a baseline for what performance to expect. We also used autoML methods such as the HyperOpt Estimator from the HyperOpt-Sklearn package to search a large number of predefined classifiers to see if we could find a better base model.

4.3 Hyperparameter tuning

To improve the performance of any base classifier we experimented with, we applied hyperparameter tuning. This was particularly important for the xgboost models where we tried different values for hyperparameters such as tree depth, number of trees, learning rate etc.. We performed both manual tuning, as well as automated tuning such as grid search and random search using the Sklearn and HyperOpt packages. The improvements to most models were again minimal, achieving only small improvements in AUC if any, but mostly driving moderately performing models to overfit. We suspect that the minimal improvements were due to the relatively small size of the training data and the unnecessary complexity of xgboost models for this problem, where hyperparameter tuning could not improve significantly from the base model.

[1] Null classifier refers to a model which predicts only one class

5. Modelling

Table B in appendix consists of the list of models that we tried categorised by week in which they were submitted. During the first week, we experimented with various models to tackle the classification problems.

In general, the models didn't do so well in week one due to the data imbalance. In the following weeks, we treated the data imbalance and the score improved significantly, notably Logistic Regression improved from 0.50080 to 0.63828. This is further stressed with the confusion matrix for Logistic Regression in section 6.2.

5.1 Best models implemented

Out of all the experiments tried, the best performing models as determined by the highest achievable AUC from predictions on the testing data are listed below.

Table 3 - Summary of best performing models

Model	**Data Preprocessing**	**Model parameters**	**Public Score**	**Private Score**
Logistic Regression	Upsampling	Solver = Liblinear	0.70683	0.69711
Logistic Regression CV	Upsampling	Solver = Liblinear	0.70760	0.69692

The method for these models involved cleaning the training data according to Section 3.1, and then checking the existence of imbalance data. We treated the imbalanced data using resample functions of sklearn package and upsampling the minority target data. After confirming the datasets were balanced, we proceeded to split the training data into training and validation sets in an 80% to 20% ratio. We instantiated the LogisticRegression model, fit it to the training dataset, and then predicted the probability and label for the training and validation datasets.

5.2 Results

We evaluate the model with an AUROC score before exporting the prediction into a csv. The AUROC score was 0.70667 for the training set and 0.70478 for the validation set.

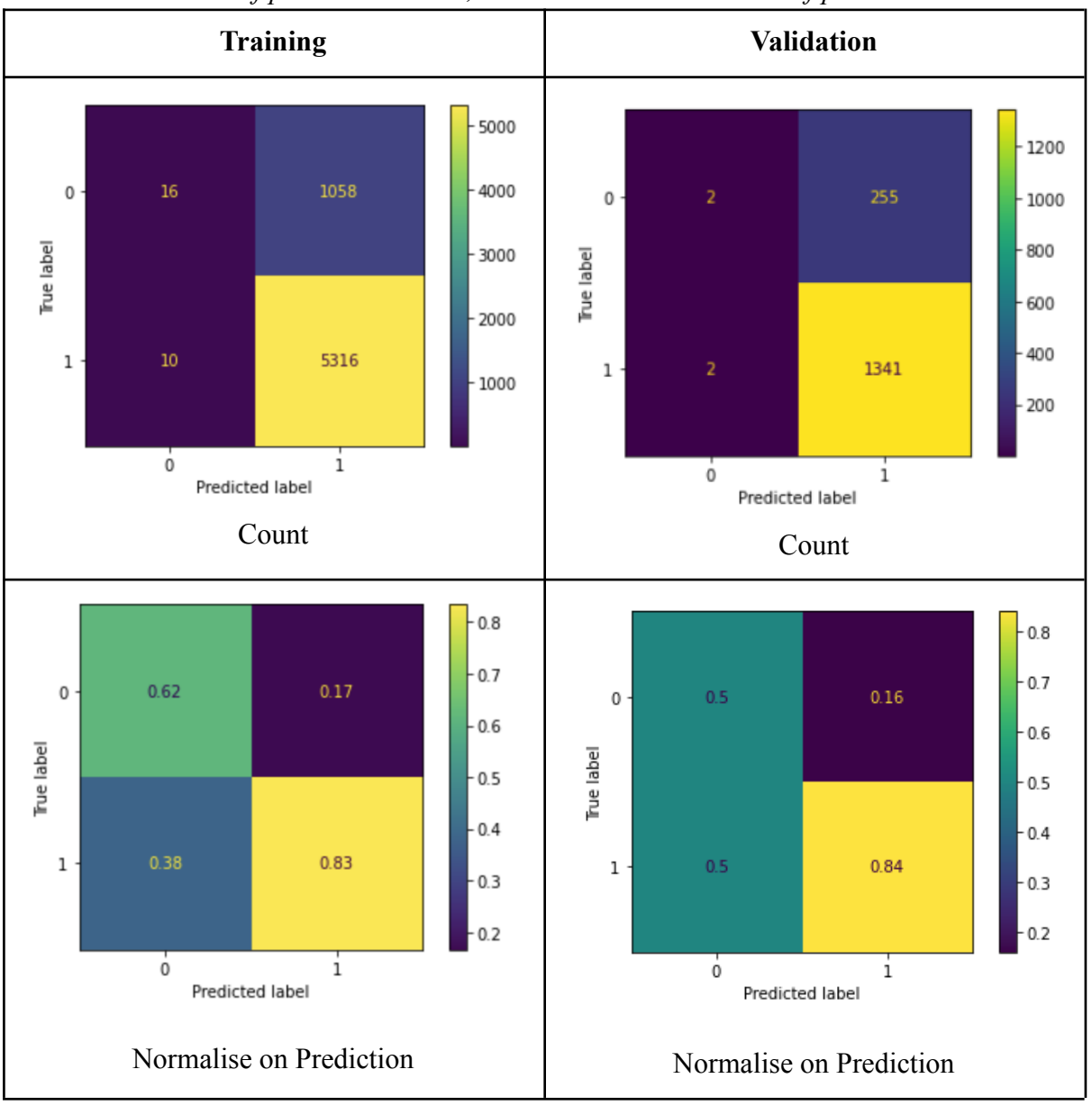
Comparatively, the second best method returned 0.70615 for training and 0.70433 for the validation AUROC score. This model uses the same data preparation and treatment of

imbalance data, however, the model is a variation of Logistic Regression that uses cross-validation. Using the public score as a guide, the cross-validation variation scored better with 0.70760 vs 0.70683 with the non cross-validation. At that time, we concluded that LogisticRegressionCV was the best model based on the public score.

However, in the private score the non CV also fared slightly better with the best model being Logistic Regression scored 0.69711 and the model Logistic Regression CV scored 0.69692.

Next, let's explore the confusion matrix of the Logistic Regression model without treatment of imbalanced data.

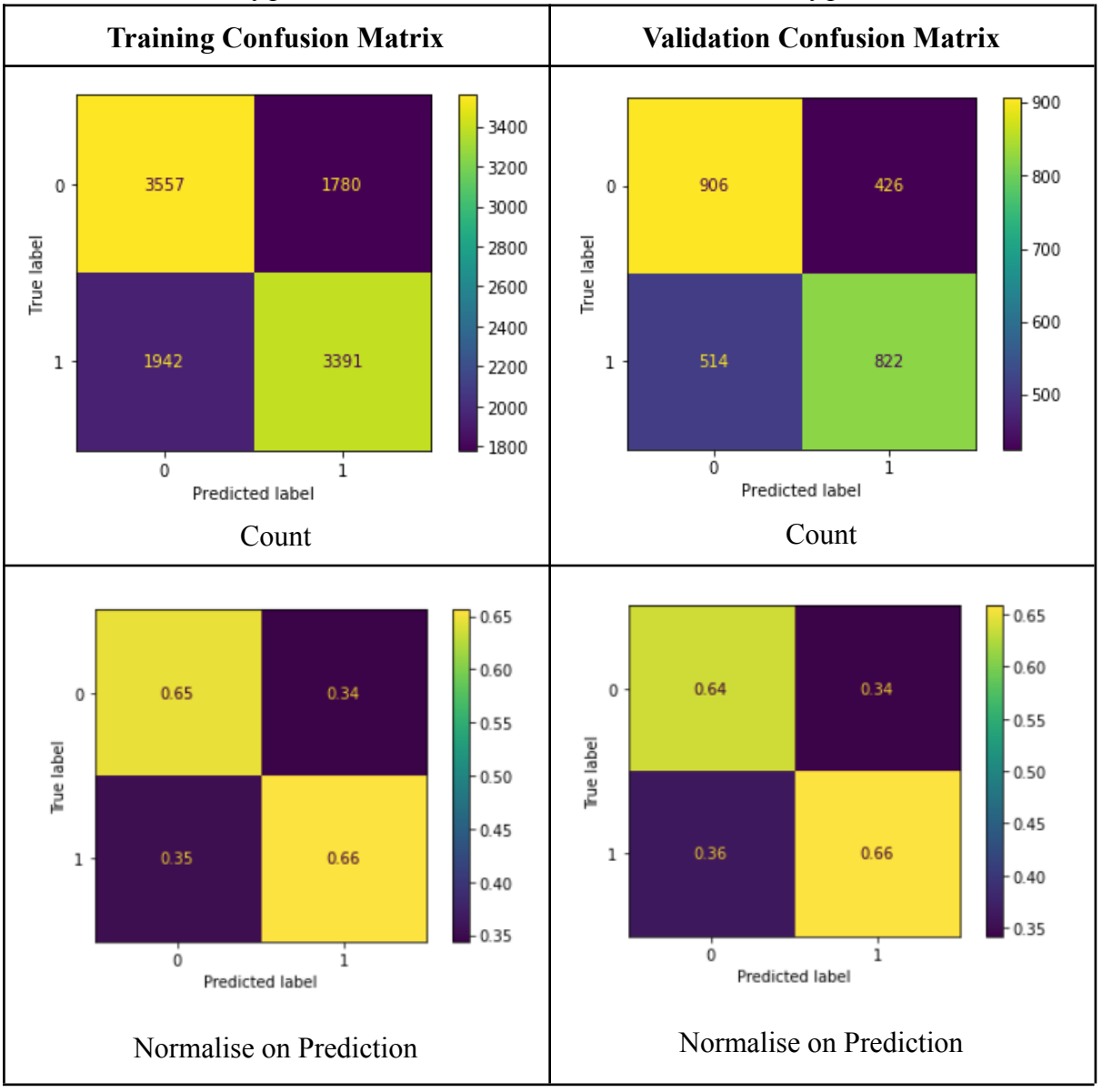
Figure 3 - Confusion matrix of training predictions without treatment of imbalanced data. Above: Counts of predicted classes, Below: Normalised values of predicted classes.



From the confusion matrix above, we can see that logistic regression has a good prediction for true positive for predicting 1, albeit still about 17% of the time it has got it wrong (false positive). Moreover, it has worse true negative results, stressing the importance of balancing the data.

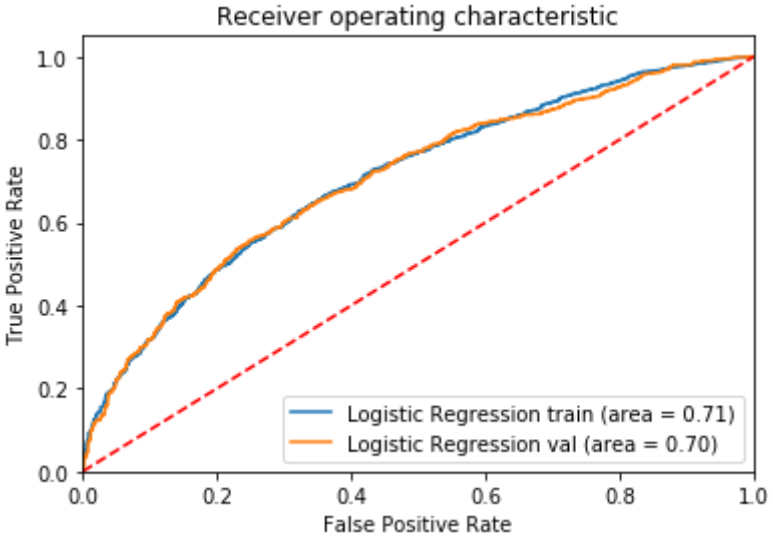
Comparing the results with the treatment for the imbalance data with upsampling, the resulting confusion matrix is below.

Figure 4 - Confusion matrix of training predictions with treatment of imbalanced data. Above: Counts of predicted classes, Below: Normalised values of predicted classes.



As we can see above in the validation sets, the True Positives value has gone down, though the True Negatives value has improved significantly from 0.5 to 0.64. Combining these findings with the improvement of the AUC score upon submission of 0.63828, it further supports our hypothesis that treating the imbalance data will help the model to generalise and avoid overfitting when testing against unknown datasets.

Figure 5 - ROC Curve for training and validation for best model - with balanced data



As we see from the ROC curve, the model was better at generalising to new data as the training and validation curves were very close. This is also supported by the f1-score being virtually the same for both training and validation as shown in figure 6.

Figure 6 - Classification Report for best model

Training report

	precision	recall	f1-score	support
0	0.65	0.67	0.66	5337
1	0.66	0.64	0.65	5333
accuracy			0.65	10670
macro avg	0.65	0.65	0.65	10670
weighted avg	0.65	0.65	0.65	10670

Validation report

	precision	recall	f1-score	support
0	0.64	0.68	0.66	1332
1	0.66	0.61	0.64	1336
accuracy			0.65	2668
macro avg	0.65	0.65	0.65	2668
weighted avg	0.65	0.65	0.65	2668

6. Discussion and Issues

6.1 Data quality issues

- Not all the values in the % column in 3P, FT, FG were accurate for e.g. the expectation is that when 3PM over 3PA should equal to 3P% but this is not always the case.
- There were negative values for some columns, this does not make sense as NBA stats cannot be negative.
- No features had an obvious distinction between the 0s and 1s, when running average of all features the most standout is offensive rebound skewing towards 1s. This is problematic when fitting models such as Random Forest, as the model favours standout features and learns mostly by its dominance.

6.2 Modelling Issues

While running the first iteration of logistic regression there were some negative coefficients in the feature such as 'stl', 'blk', 'Ast', this is problematic as intuitively these values should have a positive impact, the more steals or blocks the better the player. Regularization such as L1 (Lasso) as penalty, although it is negative but the value is very small therefore applying the L1 regularization essentially shrinks the less important features coefficient to zero.

In addition to that, the best model mentioned in section 5 includes the ID columns and the presence of negative values while fitting the model. This suggests that the model might have an overpowering predictor considering the uniqueness of ID columns, hence a good score.

The AUC observed in both the training and validation set were in the high 70s but submission AUC was in the low 60s, the cause of this problem could be model overfitting, i.e. too much information is fed to the model (Xgboost, Random Forest etc). Reducing the value of hyper parameters such as max depth, max features, number of trees helped to make the AUC a bit more consistent in training and validation set.

7. Evaluation

Unfortunately, the model does not meet the expected business success criteria of an AUC score of 0.8 as determined in the project scope, only reaching 0.69. With this score, we cannot be confident in the model's ability to accurately predict whether a player had less than a 5 year career as 34% were wrongly classified as >= 5 years, and similarly 36% were wrongly classified as < 5 years when in fact they lasted 5 years or more.

8. Recommendations and Future Scope

With the difficulties we encountered with the quality of the data, we suggest that more experiments should be performed. Some future directions are provided below:

- Additional data such as player efficiency and position of player, e.g. from the existing data the variable BLK (blocks) tend to rank high in variable importance therefore the assumption is that the higher the number of blocks the more likely to succeed in NBA but this might not be true for some positions, a point guard or shooting guard will be low on blocks as it's not a defensive position but still perform very well in NBA, therefore having a categorical variable on positions played will definitely improve the dataset.

- Principal Component Analysis - Exploring this technique to reduce dimensionality of the dataset. This could help by removing error value features mentioned above. It increases interpretability of the dataset at a minimum information loss. This feature engineering technique also creates new uncorrelated variables that maximise variance, as a lot of features in the dataset have autocorrelation.

We don't recommend that this model be used for decision making at this current stage. We suggest that more investigations be performed to improve the performance of the model by way of additional data, resolving the validity issues and then re-applying the experimental process performed here.

9. Appendix

Table A. Basketball stats of players prior to joining the NBA

Variable	Description	Data type
Id	Unique Player ID	Integer
GP	Games played prior to joining NBA	numeric
MIN	Average Minutes played per game	numeric
PTS	Average points scored per game	numeric
FGM	Average field goals made per game	numeric
FGA	Average field goals attempted per game	numeric
FG%	Field Goal percentage total field goals made divided by total field goals attempted	numeric
3P Made	Average 3 points made per game	numeric
3PA	Average 3 points attempted per game	numeric
3P%	Percentage of 3 points made	numeric
FTM	Average free throws made per game	numeric
FTA	Average free throws attempted per game	numeric
FT%	Percentage of free throws made	numeric
OREB	Average Offensive rebounds per game	numeric
DREB	Average Defensive rebounds per game	numeric
REB	Average total rebounds per game	numeric
AST	Average assist per game	numeric
STL	Average steals per game	numeric
BLK	Average blocks per game	numeric
TOV	Average turnovers per game	numeric
TARGET_5yrs	Outcome: 1 if career length >= 5 years, 0 otherwise	numeric

Table B. Basketball stats of players prior to joining the NBA

Week 1

Model Used	Public AUC score
ElasticNet Base	*0.69116*
Linear Regression without polynomial transformation	*0.70129*
RandomForest Base	0.51019
K Nearest Neighbour	0.50550
LogisticRegression Base	0.50080
Lasso Base*	0.68453*
Decision Tree Classifier	0.50993
NaiveBaiseGaussianNB	0.63341
LocalOutlierFactor	0.49969

Week 2

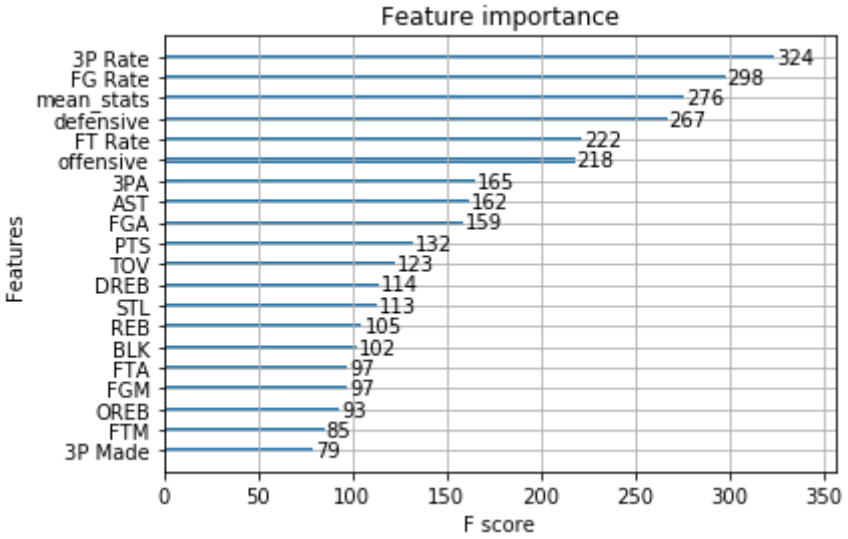
Model Used	Public AUC score
Lasso	0.68453
RidgeClassifier	0.50111
RidgeClassifierCV	0.50111
LogisticRegression Base	0.50080
LogisticRegressionCV	0.50142
Lasso with upsampling	0.68982
Lasso with downsampling	0.68852
LogisticRegression upsampling	0.63828
LogisticRegression upsampling - predict_proba1	0.70683
LogisticRegression downsampling - predict_proba1	0.70556
LogisticRegressionCV upsampling - predict_proba1	0.70760
LogisticRegressionCV SMOTE - predict_proba1	0.70895

LogisticRegressionCV with l1 penalty using SMOTE - predict_proba1	0.70655
LogisticRegression base with predict_proba	0.70707

Week 3

Model used and its parameters	Public AUC Score
Logistic Regression CV with SMOTE and liblinear. Dropped players with negative values.	0.70868
Logistic Regression CV with SMOTE and liblinear. Negative values = zero	0.71070
Logistic Regression CV with SMOTE and newton-cg with l1 penalty. Negative values = zero	0.70494
Logistic Regression CV with SMOTE and saga. Negative values = zero	0.69406
Logistic Regression CV with SMOTE and saga with elasticnet. Negative values = zero	0.69341
Logistic Regression CV with SMOTE and liblinear with higher Cs value (100). Negative values = zero	0.70862
Adaboost base with SMOTE and negative values = zero	0.67302
Adaboost base with SMOTE and negative values = zero. Learning rate 1e-1	0.68362
Logistic Regression CV with SMOTE and liblinear. Negative values = zero. Drop id_old column	0.70781
Logistic Regression CV with SMOTE and liblinear. Negative values = zero. Drop id_old column and recalculate % columns - FG%, 3P%, and FT% by dividing its Made column with its Attempts column.	0.70832
Logistic Regression CV with SMOTE and liblinear. Negative values = zero. Drop id_old column and recalculate % columns - FG%, 3P%, and FT% by dividing its Made column with its Attempts column and fixing errors. If Attempts value is lower than Made value, Attempts value is equal to Made.	0.70822

Figure A - Variable Importance plots of engineered features trialled in experiments



Here we see that the top 6 most important features are engineered features, although the result was a heavily overfitted model. There could have been data leakage here, or other spurious interactions.

10. Member Contribution

Reasmey - Setup and maintained the central project github repo. Contributed to sections 1-5 and 7 of the report, and editing and formatting the whole report. Cleaned notebook submissions and integrated with helper functions, zipped assignment artifacts. Planned group discussions and provided high level project management guidance.

Charles - Supplied the most experiments in the kaggle submission, as well as supplied the best models. Supported the team in their experiments by having regular meetings. Contribution in the report mainly from sections 5 onwards, and some on previous sections. Actively comment on the report overall and do hot-fixes (editing and formatting).

Kevin - Created kaggle team for the group. Contributed to group discussions and strategy meetings. Provided domain knowledge for feature engineering and related data questions. Initiated report scaffold, contributed to sections 6-8 and final editing.