Supplementary Material

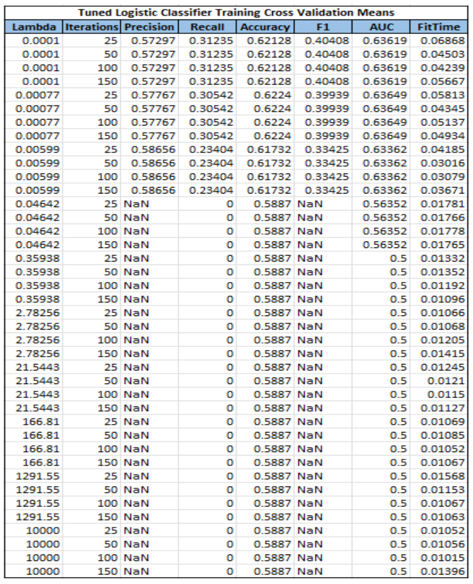
Glossary

Term	Definition
is_popular	A variable created through the "popularity". What is the popularity index based off? It is a spotify algorithm that looks at the number of streams, the recency of streams and other things like number of adds to a playlist etc. This makes time an important component as songs could have been popular and are no longer popular because others have taken their place. See the documentation here https://community.spotify.com/t5/Content-Questions/Artist-popularity/td-p/4415259)
Multicollinearity	When multiple variables are correlated
Binary Classification	Classifying a variable into 1 of 2 categories
Imbalanced Class	When the target class is under represented
Stratified	Maintains the proportion of features across sets
Cross Validation	Creates multiple train and test sets to validate a model
Grid Search	A method that searches through all specified combinations of hyperparameters
Hyperparameter	External configurations of a model to manage performance
Dummy Variable	A variable created by converting a categorical column into 1s and 0s for each category eg one column of Gender would turn into two columns of Male (1 or 0) and Female (1 or 0)
Lambda	A hyperparameter in logistic regression to control the importance of the weights of variables. It is a penalizing term in the equation that reduces weights down to zero if they do not add value. The higher the lambda the harsher the penalty
Iteration Value	The number of times a model will work through the dataset to find the optimal weights for coefficients
Number of Trees	The number of decision trees created by a Random Forest
Minimum Leaf Size	The minimum number of samples required in a leaf node of a tree
Out of Bag (OOB)	This is the sample that remains unseen when a random forest creates trees through bootstrapping. Functions similar to cross validation.
Precision	Tells us that out of all the predictions made, how many were actually correct. True Positive / (True Positive + False Positive). The higher the better
Accuracy	It tells us the overall performance. It is the number of correct predictions divided by the total number of predictions. The higher the better
Recall	Gives us a ratio of correct positive predictions to the total true positives in the sample. True Positive / (True Positive + False Negative). The higher the better

F1	Combines precision and recall and gives us a harmonic mean. The higher the better
Area Under Curve (AUC)	Quantifies the overall ability to identify a positive class from a negative class across multiple thresholds. The higher the better
OOB Error	The wrong predictions on unseen samples from the random forest bootstrapping
Fit Time	The time it took for the model to fit to the data
Receiver Operating Characteristic (ROC)	A curve that shows us the trade-off between the True Positive Rate and False Positive Rate. The area under the ROC is the AUC. A right angle type shape towards the top left of the plot signifies a better ROC
Confusion Matrix	Provides a summary of the correct and incorrect predictions by a model

Intermediate Results

- Initially the modelling was carried out on all 1.2 million rows but that quickly began to prove troublesome by the time I got to the Random Forest model because it was taking too long to train. The grid search with cross validation I did not attempt and just did grid search. Eventually I moved down to taking a smaller sample but even on the 1.2 million observations I was getting similar results to what was finally shown in the poster which suggest that more of the same data will not fix the problem. It is likely going to be a combination of different data and different hyperparameter optimisation that will improve the performance of the models.
- For the table on the right it was also interesting to see that of the defined hyperparameter space after a certain value of Lambda the Precision and F1 values started to return NAN and the AUC became no better than random guessing. This leads me to believe that the value of Lambda for the Lasso technique was adding too harsh of a penalty that made none of the variables predictive (no positive classes were returned). I am unsure if this would be the case for other projects but it seems intuitive that if you raise Lambda high enough it would reduce the importance of all the variables, Additionally the number of iterations did not matter at any level of Lambda giving the same score across iterations meaning that the model is quick at finding the right weights.



Tuned Logistic Classifier Training Cross Validation Means

Lambda	Iterations	Precision	Recall	Accuracy	F1	AUC	FitTime
0.0001	25	0.57297	0.31235	0.62128	0.40408	0.63619	0.06868
0.0001	50	0.57297	0.31235	0.62128	0.40408	0.63619	0.04503
0.0001	100	0.57297	0.31235	0.62128	0.40408	0.63619	0.04239
0.0001	150	0.57297	0.31235	0.62128	0.40408	0.63619	0.05667
0.00077	25	0.57767	0.30542	0.6224	0.39939	0.63649	0.05813
0.00077	50	0.57767	0.30542	0.6224	0.39939	0.63649	0.04345
0.00077	100	0.57767	0.30542	0.6224	0.39939	0.63649	0.05137
0.00077	150	0.57767	0.30542	0.6224	0.39939	0.63649	0.04934
0.00599	25	0.58656	0.23404	0.61732	0.33425	0.63362	0.04185
0.00599	50	0.58656	0.23404	0.61732	0.33425	0.63362	0.03016
0.00599	100	0.58656	0.23404	0.61732	0.33425	0.63362	0.03079
0.00599	150	0.58656	0.23404	0.61732	0.33425	0.63362	0.03671
0.04642	25	NaN	0	0.5887	NaN	0.56352	0.01781
0.04642	50	NaN	0	0.5887	NaN	0.56352	0.01766
0.04642	100	NaN	0	0.5887	NaN	0.56352	0.01778
0.04642	150	NaN	0	0.5887	NaN	0.56352	0.01765
0.35938	25	NaN	0	0.5887	NaN	0.5	0.01332
0.35938	50	NaN	0	0.5887	NaN	0.5	0.01352
0.35938	100	NaN	0	0.5887	NaN	0.5	0.01192
0.35938	150	NaN	0	0.5887	NaN	0.5	0.01096
2.78256	25	NaN	0	0.5887	NaN	0.5	0.01066
2.78256	50	NaN	0	0.5887	NaN	0.5	0.01068
2.78256	100	NaN	0	0.5887	NaN	0.5	0.01205
2.78256	150	NaN	0	0.5887	NaN	0.5	0.01415
21.5443	25	NaN	0	0.5887	NaN	0.5	0.01245
21.5443	50	NaN	0	0.5887	NaN	0.5	0.0121
21.5443	100	NaN	0	0.5887	NaN	0.5	0.0115
21.5443	150	NaN	0	0.5887	NaN	0.5	0.01127
166.81	25	NaN	0	0.5887	NaN	0.5	0.01069
166.81	50	NaN	0	0.5887	NaN	0.5	0.01085
166.81	100	NaN	0	0.5887	NaN	0.5	0.01052
166.81	150	NaN	0	0.5887	NaN	0.5	0.01067
1291.55	25	NaN	0	0.5887	NaN	0.5	0.01568
1291.55	50	NaN	0	0.5887	NaN	0.5	0.01153
1291.55	100	NaN	0	0.5887	NaN	0.5	0.01067
1291.55	150	NaN	0	0.5887	NaN	0.5	0.01063
10000	25	NaN	0	0.5887	NaN	0.5	0.01052
10000	50	NaN	0	0.5887	NaN	0.5	0.01056
10000	100	NaN	0	0.5887	NaN	0.5	0.01015
10000	150	NaN	0	0.5887	NaN	0.5	0.01396

Implementation Details

Why Pareek et al (2022) was chosen:

- They used a dataset that was also sourced from Kaggle. While it had a different sample size (which would mean different songs although this was not something I checked) it used the same variables as the Kaggle data set that I was using
    - This makes it an easier source to compare against
    - It should also be noted that in their paper, for classification they do not specify a threshold for popularity so it adds further fuzziness when comparing the results
    - Another similar paper
        - https://ijaem.net/issue_dcp/Spotify%20Data%20Analysis%20and%20Song%20Popularity%20Prediction.pdf

Why Pham et al (2015 was not chosen:

- James Pham, Edric Kyauk and Edwin Park (2015) used a dataset that was not entirely the same as the one available on Kaggle. There target measure was also different
    - This is a popular dataset coming from the Echo Nest API and is featured in the following paper
        - "(PDF) The Million Song Dataset.," www.researchgate.net. https://www.researchgate.net/publication/220723656_The_Million_Song_Dataset
    - The paper was part of their inspiration
    - While I could have used the same dataset as them, my reason for working with the Kaggle dataset was to reduce time sent preparing data
        - The Kaggle dataset is completely clean with no missing values
        - Spotify has documentation available about the variables
    - Another similar paper
        - https://cs229.stanford.edu/proj2018/report/16.pdf

Binary logistic regression relies on three underlying assumptions to be true: (https://fmch.bmj.com/content/9/Suppl_1/e001290)

- The observations must be independent
    - This was assumed
- There must be no perfect multicollinearity among independent variables
    - Checked this mostly through correlation plot
- Continuous predictors are linearly related to a transformed version of the outcome (linearity)
    - Did not check for this. Can be done for future work

Chose to optimise for precision for logistic regression because I wanted the predictions to actually be correct much like how you would want to correctly predict a default loan

Chose to optimise for the minimum oob error for random forests since the base model of the random forest was already out preforming the tuned logistic model on precision

Had a lot of trouble with the matlab documentation in certain cases. Simple things like calling the ooberror on the model were not working even though the code was written as described in the documentation. In the logistic regression implementation I chose to not use weights as a hyperparameter because I was unable to get it to work with the matlab function. But I feel the inclusion of Lasso mitigates this exclusion a little given it penalizes the weights if they don't add value.

Some of the categorical variables such as mode and key were already converted into a number from their string input. For logistic regression did not make a difference as they were converted into dummies and the original string input was not required. For random forests however, because it can natively handle categories and these categories were shown as numbers, I did not want the model to assume an order (the categories were marked 1 2 3 etc) so I explicitly casted them to a category in matlab for the random forest.

Used spearman correlation (instead of pearson) as the variables are not normally distributed, see distribution_plot_numeric.

The last dummy variable is dropped for each categorical variable to avoid multicollinearity

In logistic regression, when scaling the data with robust scaling, the data was split into train and test sets and then only the training data was scaled. This was done to avoid any bleed ie so that the models get no information about the test sets. The scaling parameters were saved and when the time for testing came we scaled the test data using the same parameters to replicate as if the model is seeing similar data.

When tuning for precision or OOB error, there were ties ie there were other combinations that got the same precision or OOB error. In those cases for simplicity I took the first row of the table that was produced although a more robust way would be to write a loop and filter out the rows on a hierarchy ie if precision the same then check for highest auc etc. If at the end there is still a tie then pick the whichever.

The column year was removed for simplicity. Songs could have been popular before but are no longer popular now but it might not reflect in the most recently captured score.

Variables such as artist name and song name were removed to as there was no way to account for the clickbait or artist popularity that would influence the popularity score as it is modelling on streams. https://rpubs.com/annabauer/940476 - this paper built artist popularity but I do not agree with the approach as it will cause data bleed as it uses song popularity to create artist popularity.

Genre was dropped as well as its dummy variables would give us too many dimensions, so it was left out for simplicity.