Predicting IMDb Score – Machine Learning COMP30027 Report

Anonymous
University of Melbourne
May 2024

1. Introduction

Predicting the success of a movie has always been an intriguing challenge for those in the entertainment industry, particularly in financial and marketing decision making, as well as recommender systems. The objective of the project is to obtain an optimised model to forecast the movie rating, and the dataset is modified from the opensource IMDb 5000 (Kaggle, 2018) dataset which guarantees reproducibility for future research. The predicted variable is the binned IMDb score (0-4).

Key features cover different aspects of movies such as director and casts, genres, and social media popularity, etc. Feature engineering and selection combined with other pre-processing steps such as standardisation transformed existing data into an optimal set for constructing models. Fundamental classification models including Support Vector Machine (SVM), and K-nearest Neighbours (KNN) and Logistic Regression (LR) were implemented. A Random Forrest (RF) was also used, with all these models then stacked using a Multi-Layer Perceptron (MLP) as the final estimator aiming to surpass the base models.

Models were optimised through cross validation hyper-parameter tuning. The final model was used to predict 752 held-out instances, submitted to Kaggle competition for a comprehensive validation and testing. The report aims to provide critical analysis for each model and evaluate the technical logic of their performance. Through analysing the errors each model makes, it finds that the success of models can vary due to things such as input data or the training process.

2. Methodology

The method for this project involves several steps. Firstly, general data exploration of the 3004 training instances to see what might be useful for this task, followed by more specific feature engineering and selection. Following this, we implement specific learners as described below.

2.1 Pre-processing: Feature Engineering and Selection

Figure 1 shows the target variable is imbalanced since most movies yielded average '2' performance. Hence, movies with labels '0', '1', and '4' are significantly less common than those with label '2' and '3'. Without proper manipulation, key fundamental statistical figures including mean are distorted and many models may struggle to predict these classes.

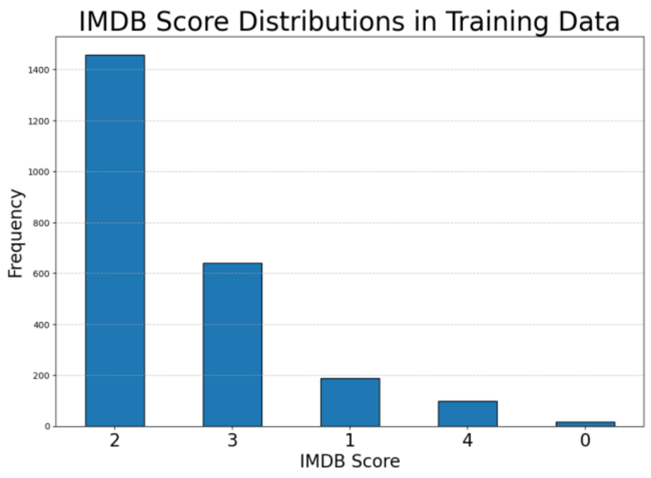


Figure 1 – Training Target Distribution

Some features also suffered from strong skewness in their distributions, of which 'country' (Figure 2) is a typical example. USA and UK together produced 87.78% percent of the movies, potentially leading to biased models. 'Country' is a categorical feature, while many common classifiers including SVM, and LR requires numerical inputs. Hence, we must transform this data into numerical values by one-hot encoding, in this case to 'USA', 'UK' and other, as we cannot make generalisable rules from countries with very few films.

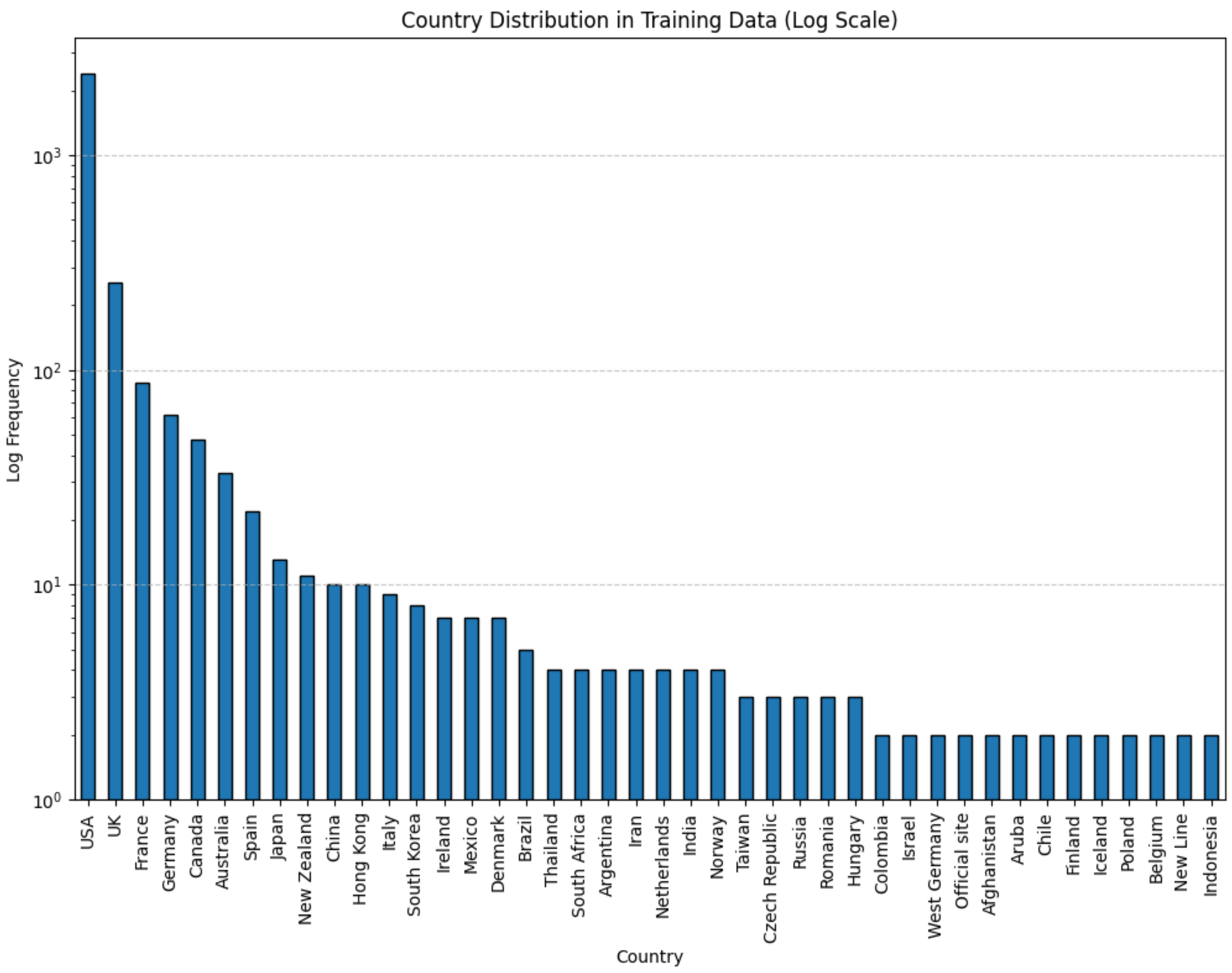


Figure 2 – Country Distribution (logged scale)

Many multi-class classifiers can generalise better when features have strong separability. Highly correlated features may lead to multicollinearity, and this should be avoided in linear models such as logistical regression. Also, to avoid the curse of dimensionality, these features should be dropped as they are potentially presenting redundant information. We compared the 3 highly correlated pairs with the IMDb score (see those with corr > 0.7 in Figure 5 p.3). 'Actor_1_facebook_likes', 'num_user_for_reviews' and 'movie_facebook_likes' were left out, having weaker correlation with the target variable than their counterparts.

Content rating experienced some updates and changes along with the evolution of movies. Online research reveals movies produced at different periods may be labelled with different ratings while meant to target at similar audiences. 'PG' replaced 'GP' and 'M', similarly 'NC-17' replaced 'X', so we grouped them into one label (The Classification & Rating Administration, 2018). 'Not Rated', 'Unrated', 'Approved' and 'Passed', were grouped together too.

For linear models, log transformation was applied to variables that exhibited significant skewness. 'num_voted_users' is an example of such a feature (Figure 3). Hence, we applied log transform to normalise the data, aiming to stabilise the variance and have a distribution closer to normal.

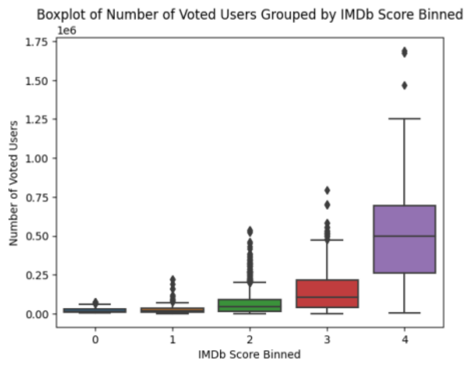


Figure 3 – num_voted_users before transformation

After transforming:

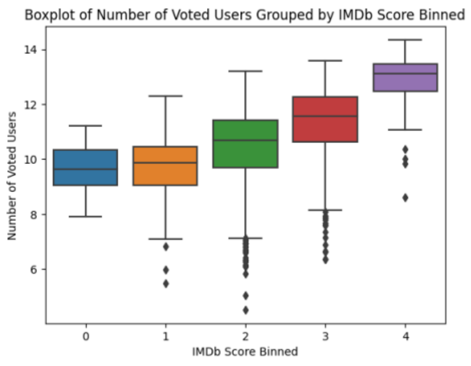


Figure 4 – num_voted_users after transformation

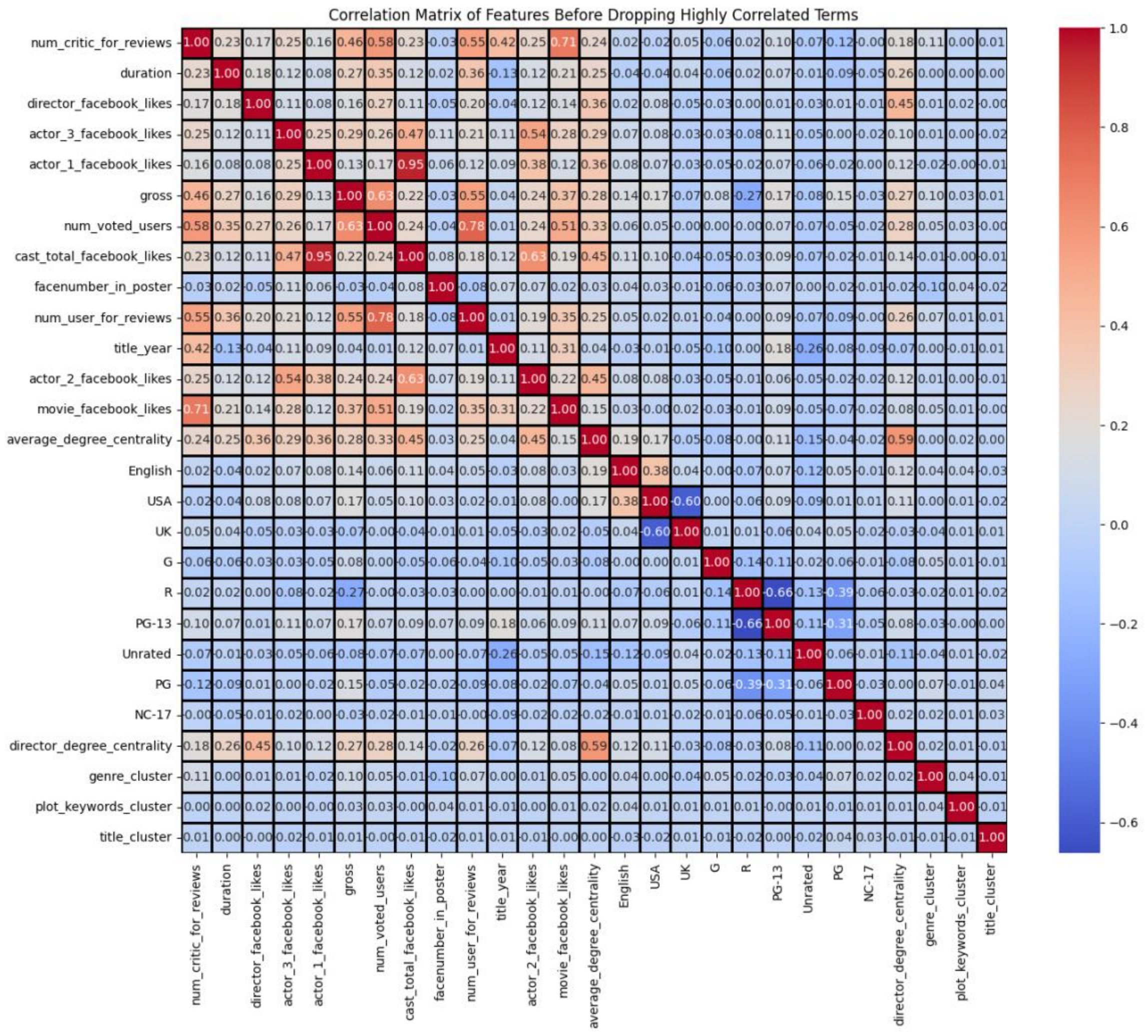


Figure 5 – Feature correlation

Similar to 'average_degree_centrality' measuring the degree of centrality among actors, we generated a new feature 'director_degree_centrality'. It measures the degree of 'connectedness' based on films that share the same director, (subtracting 1 to not measure self-connections):

$$[\text{Director Degree Centrality} = \frac{\text{Number of movies by director} - 1}{\text{Number of movies in the training set} - 1}]$$

We also performed 5-means clustering on the embeddings of 'genre', 'title' and 'plot_keywords' to categorise them for tree models.

2.2 Synthetic Minority Oversampling Technique (SMOTE)

In preliminary testing many models failed to classify a single instance of class 0. Attempting to mitigate this, we created synthetic samples based on the 5 nearest neighbours of existing class 0 instances for the models to train on using the method described in Chawla et al., (2002). We created as many as needed to match the second smallest minority '4'.

2.3 Data Splitting

To evaluate the models in a useful way (seeing more than just the Kaggle score), we split the final feature engineered data into multiple components.

1. An 80-20 test train split on the original **before** applying SMOTE. This is to leave a held out set to evaluate the effects of SMOTE on unseen data.
2. SMOTE is applied to the 80 component (resulting in 99 class '0' instances)
3. A 70-30 test train split on the larger 80 component above.

This is so we can evaluate: (1) training metrics, (2) test metrics on the data that **does** have the synthetic '0' samples, and (3) test metrics on the data that **doesn't** have the synthetic samples.

2.3 Classifiers Used

Random Forest combines the prediction power of multiple decision trees, which improves the robustness of the model. RF itself could be considered as an ensembled model. We chose the model due to its two major advantages: Bootstrapping sampling in each tree and random feature selection both improve RF's ability for generalisation and prevents overfitting. Scikit-learn automatically converted continuous features into discrete categories.

KNN predicts labels through calculating proximity of data points. We chose KNN for its supremacy in classifying complex and non-linear boundaries.

Both **Linear SVM** and **Logistic Regression** are suitable for separating linear decision boundaries. Moreover, SVM's ability to construct a hyperplane is maintained when dealing with high dimensional dataset.

Our **stacked model** extracts the outputs of the base models as inputs for the second layer of meta model. We experimented with a **Multi-Layer Perceptron** as the meta model because it is good for capitalising on the variety in models, capturing complex relationships. The model was trained on 100 epochs with a 20% held out. Once the accuracy on the holdout set stopped improving early call-back is triggered, stopping training.

2.4 Hyperparameter Tuning

Our method involved using a Grid Search 5-fold Cross Validation to choose the hyperparameters of each model that optimise accuracy. Figure 6 illustrates this process for a single hyperparameter c (regularisation strength in logistic regression). In this case there are diminishing marginal gains, where increasing c stops meaningfully increasing CV accuracy at around 5-7.

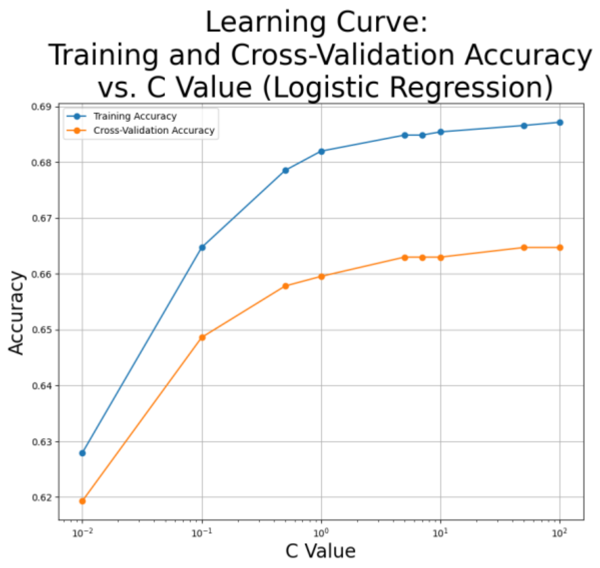


Figure 6 – C Hyperparameter Tuning (LR)

Unfortunately, the runtime of doing this on the MLP was very high, such that only some basic experimentation and testing of hyperparameters using the held-out set (with synthetic samples) was possible. The initial held-out set was still kept unseen to evaluate true performance.

3 Results

As described in our data splitting, for each model this report presents classification reports on (1) training data, (2) unseen test data including synthetic samples, (3) unseen test data excluding synthetic samples.

In terms of accuracy, models had varying success, with Random Forest performing the best on the non-synthetic test data (0.74) (Figure 7). In all cases, training accuracy was unsurprisingly higher than test accuracy. Further analysis of these results is discussed in **section 4,** with model specific classification reports below:

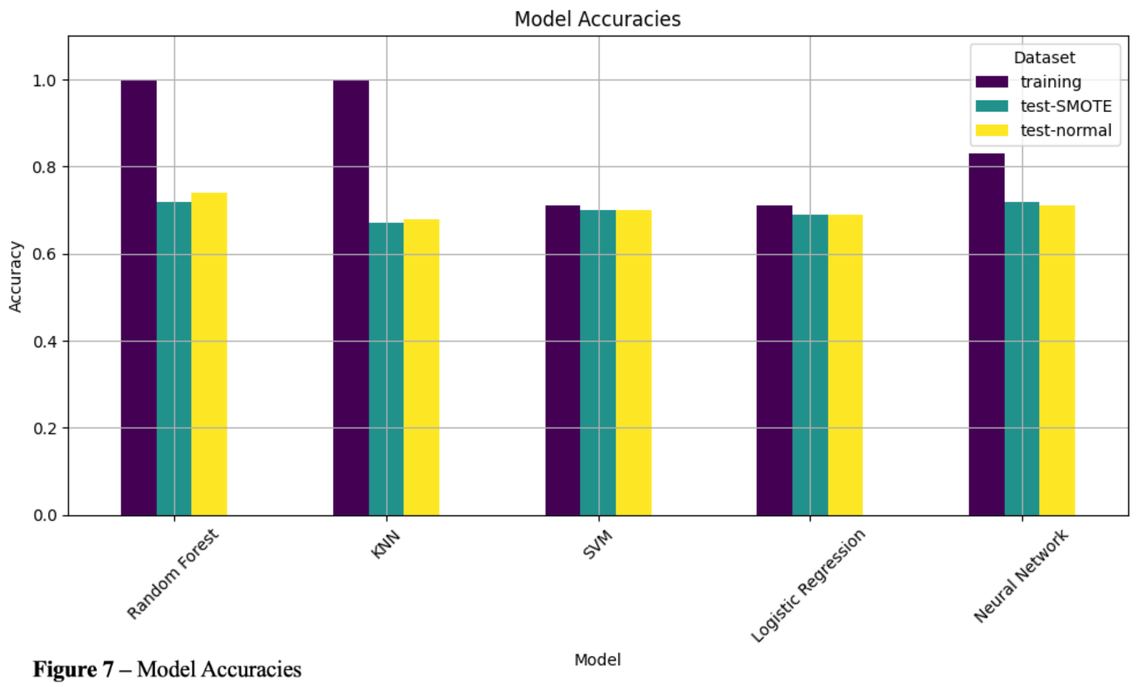


Figure 7 – Model Accuracies

Data Split	Metric	Precision	Recall	F1-Score	Support
Train	0	1.00	1.00	1.00	67
	1	1.00	1.00	1.00	137
	2	1.00	1.00	1.00	1020
	3	1.00	1.00	1.00	445
	4	1.00	1.00	1.00	70
	Macro Avg	1.00	1.00	1.00	1739
	Weighted Avg	1.00	1.00	1.00	1739
	Accuracy		**1.00**		1739
Heldout Test (Synthetic Samples)	0	0.92	0.75	0.83	32
	1	0.00	0.00	0.00	52
	2	0.72	0.92	0.81	438
	3	0.68	0.47	0.55	195
	4	0.79	0.66	0.72	29
	Macro Avg	0.62	0.56	0.58	746
	Weighted Avg	0.67	0.72	0.68	746
	Accuracy		**0.72**		746
Heldout Test (No Synthetic Samples)	0	0.00	0.00	0.00	7
	1	0.33	0.04	0.08	46
	2	0.74	0.94	0.83	381
	3	0.70	0.45	0.55	137
	4	0.90	0.63	0.75	30
	Macro Avg	0.54	0.41	0.44	601
	Weighted Avg	0.70	0.74	0.70	601
	Accuracy		**0.74**		601

Table 1- Random Forest Classification Reports

3.1 Random Forest (above)

The results of the Random Forest hyperparameter optimisation were: balanced class weight, using entropy, max depth of 20, with maximum 20 features per tree and 100 estimators per forest. Other models with fewer features and smaller depth produced accuracy scores as low as 0.43, while the most complicated (max depth) models produced accuracies around 0.69. These parameters maximised the 5-fold CV accuracy, at around 0.72, with the results on each set above.

3.2 KNN (below)

The results from the KNN hyperparameter optimisation suggested the optimal accuracy was achieved at 20 neighbours, using Manhattan Distance and distance weighting, producing a 5-fold CV accuracy of 0.66:

Data Split	Metric	Precision	Recall	F1-Score	Support
Train	0	1.00	1.00	1.00	67
	1	1.00	1.00	1.00	137
	2	1.00	1.00	1.00	1020
	3	1.00	1.00	1.00	445
	4	1.00	1.00	1.00	70
	Macro Avg	1.00	1.00	1.00	1739
	Weighted Avg	1.00	1.00	1.00	1739
	Accuracy		**1.00**		1739
Heldout Test (Synthetic Samples)	0	0.85	0.69	0.76	32
	1	0.00	0.00	0.00	52
	2	0.69	0.92	0.79	438
	3	0.60	0.37	0.46	195
	4	1.00	0.14	0.24	29
	Macro Avg	0.63	0.42	0.45	746
	Weighted Avg	0.63	0.67	0.62	746
	Accuracy		**0.67**		746
Heldout Test (No Synthetic Samples)	0	0.00	0.00	0.00	7
	1	0.14	0.02	0.04	46
	2	0.71	0.91	0.79	381
	3	0.58	0.39	0.46	137
	4	1.00	0.23	0.38	30
	Macro Avg	0.49	0.31	0.33	601
	Weighted Avg	0.64	0.68	0.63	601
	Accuracy		**0.68**		601

Table 2- KNN Classification Reports

Data Split	Metric	Precision	Recall	F1-Score	Support
Train	0	1.00	0.55	0.71	67
	1	0.00	0.00	0.00	137
	2	0.71	0.91	0.80	1020
	3	0.66	0.49	0.56	445
	4	0.96	0.61	0.75	70
	Macro Avg	0.66	0.51	0.56	1739
	Weighted Avg	0.66	0.71	0.67	1739
	Accuracy		**0.71**		1739
Heldout Test (Synthetic Samples)	0	1.00	0.41	0.58	32
	1	0.00	0.00	0.00	52
	2	0.70	0.91	0.79	438
	3	0.64	0.47	0.54	195
	4	0.89	0.55	0.68	29
	Macro Avg	0.65	0.47	0.52	746
	Weighted Avg	0.65	0.70	0.66	746
	Accuracy		**0.70**		746
Heldout Test (No Synthetic Samples)	0	0.00	0.00	0.00	7
	1	0.00	0.00	0.00	46
	2	0.73	0.92	0.81	381
	3	0.57	0.40	0.47	137
	4	0.79	0.50	0.61	30
	Macro Avg	0.42	0.37	0.38	601
	Weighted Avg	0.63	0.70	0.65	601
	Accuracy		**0.70**		601

Table 3- Linear SVM Classification Report

3.3 Linear SVM (above)

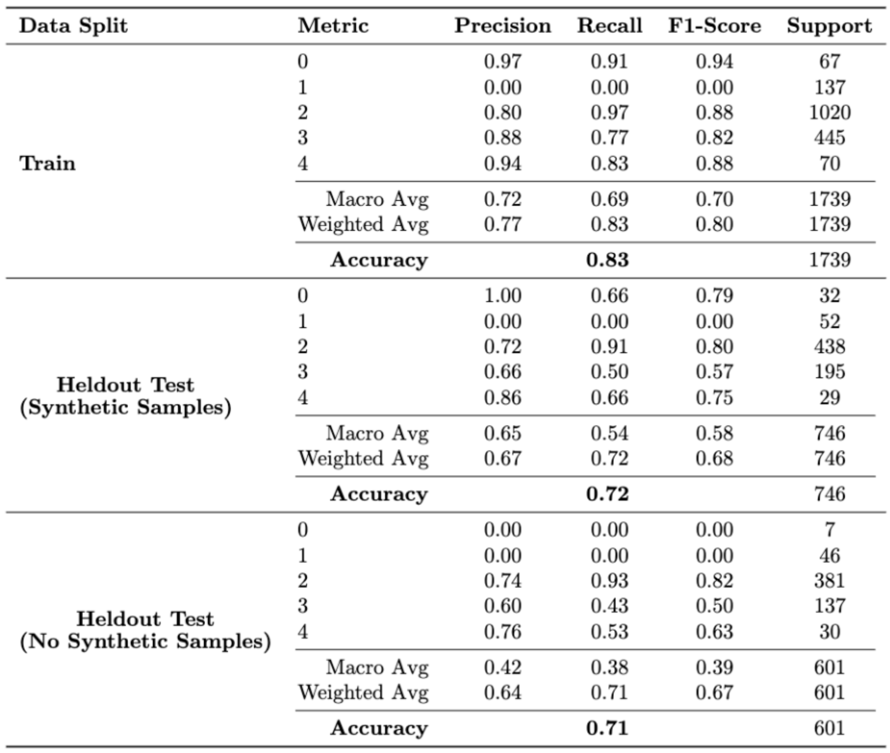
The optimal hyperparameters for this model were c=0.9 (penalty parameter) with no changes to class weights. It appears it could not create a linear decision boundary that separated class 1 from the others, with 0.00 precision and recall in all splits for this class. This will be discussed in section 4.

3.4 Logistic Regression (below)

The optimal parameters for the logistic regression were c=7, using a one-vs-all multiclass approach. The results are as follows:

Data Split	Metric	Precision	Recall	F1-Score	Support
Train	0	0.96	0.69	0.80	67
	1	0.31	0.04	0.07	137
	2	0.71	0.90	0.79	1020
	3	0.65	0.49	0.55	445
	4	0.89	0.67	0.76	70
	Macro Avg	0.70	0.56	0.60	1739
	Weighted Avg	0.68	0.71	0.67	1739
	Accuracy		**0.71**		1739
Heldout Test (Synthetic Samples)	0	0.80	0.50	0.62	32
	1	0.25	0.02	0.04	52
	2	0.70	0.90	0.79	438
	3	0.64	0.45	0.53	195
	4	0.78	0.72	0.75	29
	Macro Avg	0.63	0.52	0.54	746
	Weighted Avg	0.66	0.69	0.66	746
	Accuracy		**0.69**		746
Heldout Test (No Synthetic Samples)	0	0.00	0.00	0.00	7
	1	0.00	0.00	0.00	46
	2	0.73	0.90	0.81	381
	3	0.55	0.38	0.45	137
	4	0.74	0.67	0.70	30
	Macro Avg	0.40	0.39	0.39	601
	Weighted Avg	0.63	0.69	0.65	601
	Accuracy		**0.69**		601

Table 4- Logistic Regression Classification Report



Data Split	Metric	Precision	Recall	F1-Score	Support
Train	0	0.97	0.91	0.94	67
	1	0.00	0.00	0.00	137
	2	0.80	0.97	0.88	1020
	3	0.88	0.77	0.82	445
	4	0.94	0.83	0.88	70
	Macro Avg	0.72	0.69	0.70	1739
	Weighted Avg	0.77	0.83	0.80	1739
	Accuracy		**0.83**		1739
Heldout Test (Synthetic Samples)	0	1.00	0.66	0.79	32
	1	0.00	0.00	0.00	52
	2	0.72	0.91	0.80	438
	3	0.66	0.50	0.57	195
	4	0.86	0.66	0.75	29
	Macro Avg	0.65	0.54	0.58	746
	Weighted Avg	0.67	0.72	0.68	746
	Accuracy		**0.72**		746
Heldout Test (No Synthetic Samples)	0	0.00	0.00	0.00	7
	1	0.00	0.00	0.00	46
	2	0.74	0.93	0.82	381
	3	0.60	0.43	0.50	137
	4	0.76	0.53	0.63	30
	Macro Avg	0.42	0.38	0.39	601
	Weighted Avg	0.64	0.71	0.67	601
	Accuracy		**0.71**		601

Table 5- Stacked MLP Classification Report

3.5 Stacked Model (Multi-Layer Perceptron final) (above)

Training time constraints prevented multi-fold cv hyperparameter tuning, so the hyper-parameters had to be loosely tuned on the first hold-out. The final model had one 32 neuron layer (with ReLU activation, l2 regularisation and 40% dropout to prevent overfitting), followed by a 16-neuron layer with the same paramters, followed by a 5 neuron (for 5 classes) layer with softmax activation from which final classification are made. Increasing complexity (more neurons, layers etc.) showed little to no gain. This model had only similar accuracy to the base models.

4 Discussion and Critical Analysis

This section aims to analyse and understand the behaviour and performance dynamics on the dataset, referencing the results in section 3. It comprises of higher-level evaluation, followed by a detailed analysis of each model.

4.1 Bias in results

Bias is the tendency of the model to produce systematically wrong predictions (Azam, 2024). High recall for the majority class, combined with low recall for the minority classes indicates the models are biased to the majority class. As an example, take the KNN model's metrics in Table 2. It has 0.91 recall on class 2 (majority), while the recall for the remaining classes range between 0.00 and 0.39 -the model managed to classify most of the true '2' instances but could not classify the true instances of the other classes as well. This phenomenon is consistent across all models.

Attempts to mitigate this bias synthetically oversampling the minority class, or using balanced class weights, as with the Random Forest model seem to also have been ineffective, to which we must ask why?

One significant reason to explain this is in the choice for **hyperparameter tuning** -aiming to maximise the accuracy score. This highly favours models that are good at predicting the majority class '2', as being good at predicting this class maximises accuracy. Perhaps a better approach would have been to tune hyperparameters on how they affect the recall or precision of classifications of minority classes.

4.2 Did synthetic sampling fail?

Synthetic data augmentation brings with it a series of disadvantages, including lacking interpretability and transparency, with no guaranteed benefit to performance (Azam, 2024).

In the held-out sets without synthetic data *every* model had a 0.00 recall and precision on class '0', meaning no model predicted a single true positive of this class on the held out data. This is even though each model did predict true positive '0's for training and test (with synthetic data) sets. This largely indicates that synthetic sampling was unsuccessful.

There may have simply been not enough instances of class 0 to make generalisable rules by taking a subsection. In fact, there were only 24 instances of class 0 in the initial set. After taking our split there were only 7 instances in the held-out test set, and 17 instances used to generate the synthetic samples, which we suspect are just not similar enough to create generalisable rules to compare the two on.

To verify this, we compared descriptive statistics of the synthetic sample and the held-out sample. Indeed, many features were largely different – the synthetic sample had half the mean 'num_critic_for_reviews', 'director_facebook_likes' and frequency of 'PG-13' films than the held-out samples among other large differences in features. These examples highlight how rules based on these features may be misleading, because synthetic instances are misrepresenting the actual 'average' class 0 film.

4.3 Model Analysis

The **Random Forest** achieves perfect 1.00 metrics on training data (Table 1), while only achieving 0.72 and 0.74 accuracy on the held out sets respectively. This is not necessarily a sign of overfitting in a Random Forest model (Google for Developers, 2024), as with sufficient trees this is often possible.

However, it fails to classify any instances of class '1' on the dataset with synthetic samples, while classes '0' and '2' had higher recall (0.75, 0.92). This indicates that the features relevant in classifying class '1' are potentially overshadowed by features relevant to nearby classes.

An interesting metric is the 0.94 recall on class '2' (non-synthetic holdout). This supports the reasoning that the hyperparameters were tuned to maximise the success of predicting the majority class (maximising accuracy at the expense of other classes).

The **KNN** model also achieves perfect 1.00 accuracy on the training set (Table 2). This is unsurprising, considering a small enough k (20).

It also fails to classify class '1' in the held out set while performing better on class '0' (0.69 recall). This supports the idea that the synthetic '0' instances had feature values very similar to class '1', which overshadowed the true instances of class '1'.

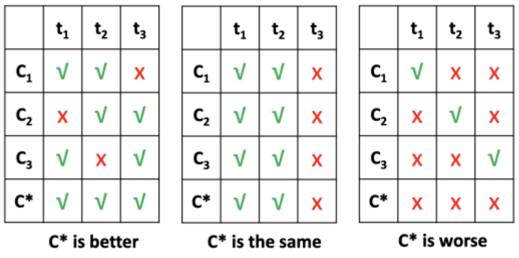
The **Linear SVM** produced some interesting results: in every case (training and both holdouts) it also failed to classify class '1'. This implies that the classes may not be linearly separable, and that a linear boundary may not be appropriate for what seems to be a more complex separation.

Perhaps a model that can produce more complex decision boundaries may have been more appropriate to capture the nuances between minority classes better.

The **Logistic Regression** also produced noteworthy results: it failed to classify a single instance of class '0' or '1' in the test (non-synthetic) set (despite doing okay on the training and synthetic test sets). This is explainable by the properties of the model -since the logistic regression produces a linear decision boundary (Ehinger, 2024), it is likely this boundary is overfit to the instances in the training data that are heavily influenced by the synthetic samples. Thus, it does not generalise well on unseen data.

This is not a blanket effect for all minority classes though with some success in class ‘4’. In fact, it seems that ‘4’ may simply be more separable than the other minority classes. Consider class ‘4’ having more distinct ‘num_user_reviews’ in Figure 4 (section 2).

Finally, the **Stacked** model using a **Multi-Layer Perceptron** failed to improve on the results of the underlying models. Weighting each class equally, the macro-average precision (0.42) and recall (0.38) were considerably worse than their instance weighted counterparts (0.64, 0.71), showing poor performance on minority classes. In fact, it failed to classify a single class ‘0’ or ‘1’ on the final held-out set. This highlights the fact that ensemble models such as stacked models will not always improve results given their input data. Consider Figure 8 below:



	t_1	t_2	t_3
C_1	√	√	x
C_2	x	√	√
C_3	√	x	√
C*	√	√	√

C* is better

	t_1	t_2	t_3
C_1	√	√	x
C_2	√	√	x
C_3	√	√	x
C*	√	√	x

C* is the same

	t_1	t_2	t_3
C_1	√	x	x
C_2	x	√	x
C_3	x	x	√
C*	x	x	x

C* is worse

Figure 8 – Ensemble Performance Based on Input, (t - instance, C -base classifier, C* - ensemble classifier) (Azam, 2024)

Seeing as all the base models made similar mistakes (especially on classes ‘0’ and ‘1’) while all performed similarly okay on class ‘2’, this is likely a case of **C* is the same** or even **worse**.

In other words, there was simply not enough variety across classifications in the base models for the final classifier to learn anything new.

5 Conclusions

This project provides a comprehensive analysis of various machine learning models on the IMDb 5000 movie dataset, highlighting disparities in performance, particularly among minority classes. Results underline the critical importance of thoughtful hyperparameter tuning, showing that optimising based on accuracy alone may come at the cost of successful prediction of minority classes, while also exemplifying the potential pitfalls of synthetic data augmentation in achieving robust performance.

The impact of these decisions become apparent in the limits on the effectiveness of the final ensemble model -with stacked performance failing to noticeably outperform base models. Ultimately, our work demonstrates the need for careful consideration of all evaluation metrics in model selection to address bias and improve outcomes in machine learning applications. Doing so will allow for better prediction of film success, influencing decision making around marketing, finances, and film recommendations.

6 References

Azam, B. (2024). *“Big Data,” Data Augmentation, and Ethics - COMP30027* [Lecture Slides]. https://canvas.lms.unimelb.edu.au/courses/181617/pages/week-11-semi-supervised-learning-and-llms?module_item_id=5169938

Azam, B. (2024). *Classifier Combination - COMP30027* [Lecture Slides]. https://canvas.lms.unimelb.edu.au/courses/181617/pages/week-7-classifier-combination-and-feature-selection?module_item_id=5169934

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*(16), 321–357. https://doi.org/10.1613/jair.953

Ehinger, K. (2024). *Logistic Regression - COMP30027* [Lecture Slides]. https://canvas.lms.unimelb.edu.au/courses/181617/pages/week-6-linear-and-logistic-regression?module_item_id=5169933

Google for Developers. (2024). *Machine Learning – Random Forests*. Google Developers. https://developers.google.com/machine-learning/decision-forests/random-forests

Kaggle. (2018). *IMDB 5000 Movie Dataset*. Www.kaggle.com. https://www.kaggle.com/datasets/carolzhangdc/imdb-5000-movie-dataset

The Classification & Rating Administration. (2018). *History*. Filmratings.com. https://www.filmratings.com/History

2596 words (excl. Bibliography, headings, tables etc.)