COMP30027 Report
Anonymous

1. Brief Introduction of Data Set and Data Modelling Problem

The scope of this project aims to utilise and analyse surpervised machine learning techniques to predict IMDB movie ratings, based on predictor features related to directors, duration, movie title, actor names, facebook likes, country, genre etc. The target variable, imdb_score_binned, has 5 possible outcomes from 0 to 4, with 0 being the lowest score and 4 being the highest score. Classification-based machine learning models will be used in the analysis to categorise the movies into the distinct IMDB movie rating classes.

2. Exploratory Data Analysis

2.1 Examining Target Variable

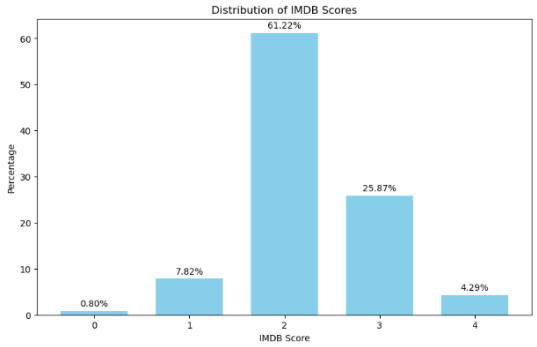


Figure 1 – Bar plot of distribution of target variable, imdb_score_binned

The distribution is significantly imbalanced, with imdb_score_binned = 2, 3 taking up more than 85% of the total number of samples. This is something that we should look into later, as a class balance may pose a challenge to machine learning models.

2.2 Examining Attributes In Relation To Target Variable

2.2.1 Categorical Feature Analysis

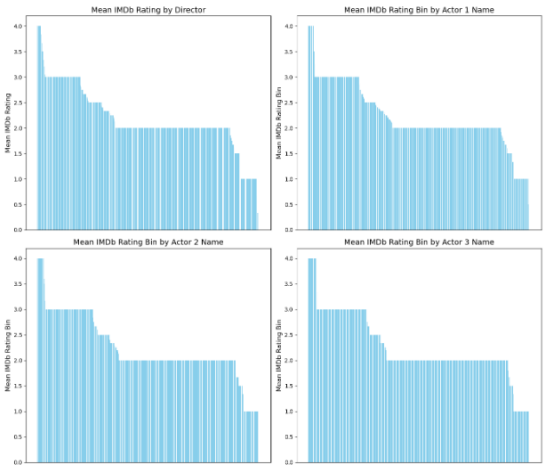


Figure 2 – Bar plot of mean IMDb Rating Bin by director_name, actor_1_name, actor_2_name, actor_3_name

Investigating the mean IMDb rating of movies associated with each individual. This initial exploration allowed us to uncover patterns in movie ratings based on the involvement of specific directors and actors. The bar plots shows a variable distribution of the abovementioned factors with respect to mean IMDB rating, hence we can keep these factors.

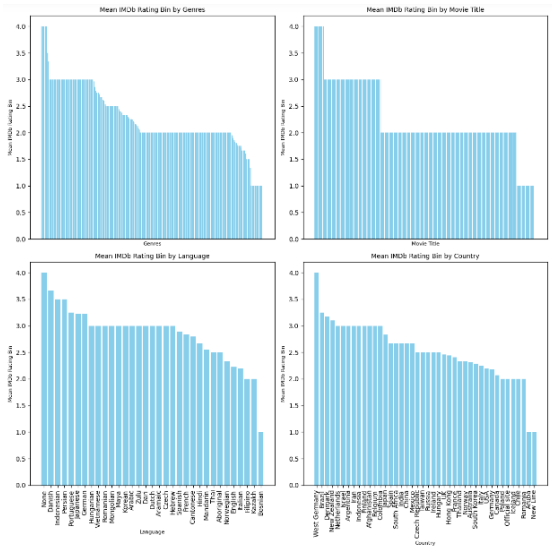


Figure 3 – Bar plot of mean IMDb Rating Bin by genre, movie_title, language, country

We look at genres, movie_title, language,

country, and investigated the mean IMDb rating of movies associated with each factor. The bar plots shows a variable distribution of the abovementioned factors with respect to mean IMDB rating.

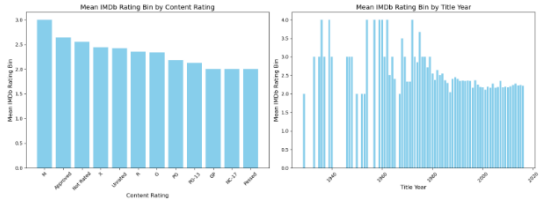


Figure 4 – Bar plot of mean IMDb Rating Bin by content_rating, title_year

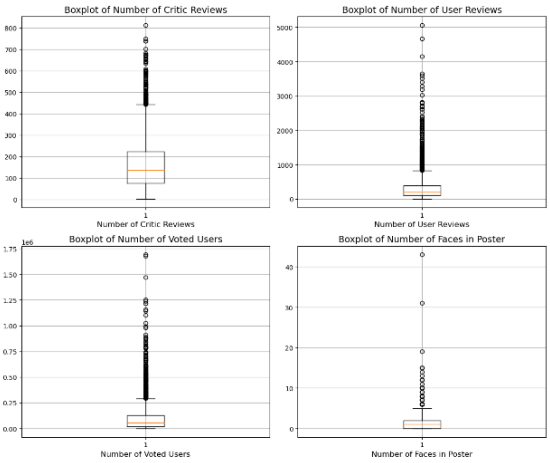
content_rating, title_year also show a variable distribution, hence by retaining these 2 factors in our model.

	Variable	Number of Unique Categories
0	Director Name	1460
1	Actor 1 Name	1265
2	Actor 2 Name	1903
3	Actor 3 Name	2198
4	Genres	675
5	Movie Title	2942
6	Language	33
7	Country	42
8	Content Rating	12
9	Title Year	72

Table 1 – Number of unique categories for categorical variables

Some features were also found to have a large number of unique categories, as shown in the table as well.

2.2.2 Checking For Outliers (Numerical Variables)



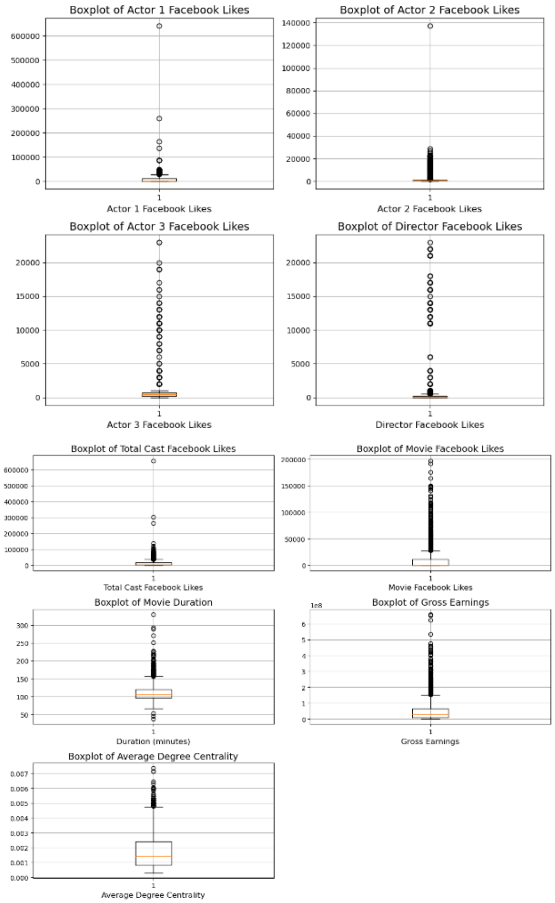
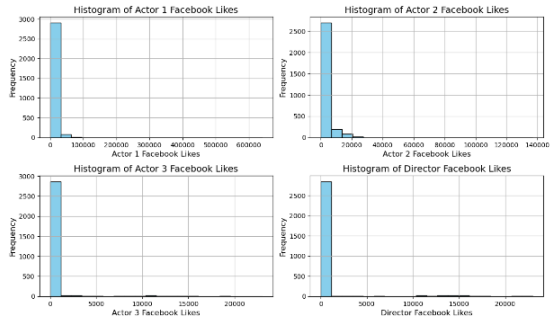


Figure 5 – Box plots of distribution of numerical variables

From the boxplots of all the numerical variables, there is quite a significant number of outliers (outside the interquartile range) present in the numerical variables in the dataset. However, removing a large number of data points which can contain valuable information for our machine learning models may not be feasible, and there is insufficient evidence that these data points are actually anomalous observations.



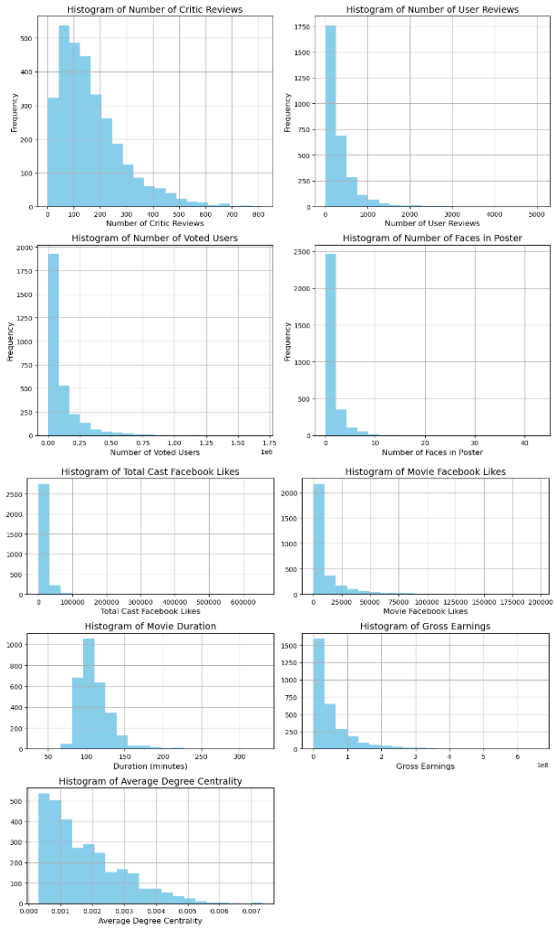


Figure 6 – Histograms of distribution of numerical variables

While outliers may initially appear concerning from the boxplots, the distribution of the numerical variables as seen from the histogram appears to be within an acceptable range, and does not necessarily indicate anomalous observations, hence does not require action.

2.2.2 Correlation Analysis

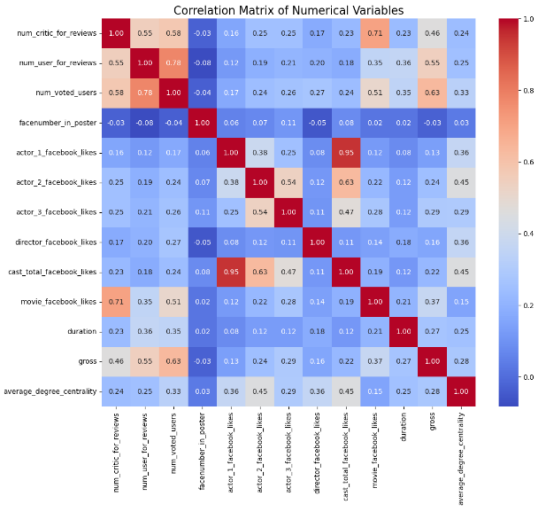


Figure 7 – Correlation Matrix of numerical variables

High correlation between feature variables could lead to multicollinearity, which might affect the performance and interpretability of certain machine learning models.

From the correlation matrix, we can see that actor_1_facebook_likes and cast_total_facebook_likes are highly correlated (r = 0.95). Thus, we will remove cast_total_facebook_likes.

3. Data-Preprocessing

3.1 Removing NA Columns

From investigation there are no missing data.

```
No missing values in any column.
```

Figure 8 – Number of missing data

3.2 Aggregation of Colums

Some categorical variable had a large number of unique categories, which might lead to the curse of dimensionality and lead to challenges for the machine learning models.

3.2.1 'title_year' column

We will first handle the 'Title Year' column which has 72 unique years, into bins of 10 years.

```
Counts for each category in Title_Year_Binned:
2000-2009    1399
2010-2019     769
1990-1999     573
1980-1989     180
1970-1979      43
1960-1969      21
1950-1959       8
1930-1939       5
1940-1949       5
1920-1929       1
Name: Title Year Binned, dtype: int64
```

Table 2 – Counts of each category for Title_Year_Binned

3.2.2 'country' column

We aggregated the 42 unique countries into the 7 continents. The broad geographical information provided by continent is still likely to capture the specific country data. Values that are not country related will be labelled as 'Others'.

```
Counts for each category in continent:
North America    2435
Europe            462
Asia               51
Australia          42
South America       9
Africa              3
Others              2
Name: continent, dtype: int64
```

Table 3 – Counts of each category in continent

3.3 Text Processing

3.3.1 Vectorising 'actor_3_name' Column using CountVectorizer

For actor_3_name, we will use CountVectorizer.

3.3.2 Replacing Textual Columns with Vectorised Columns

We replaced the text columns with their respective provided vectorized columns.

3.4 Removing Rows With Irrelevant or Ambiguous Information

3.4.1 'continent' column

During the aggregation of the country column into continents above, data which irrelavant to country were classified as 'Others'. From above, we can see that there are 2 rows are classified as 'Others' in train_dataset, and will be removed since these 2 data points are not likely to contribute to any meaningful feature.

3.4.2 'id' column

The 'id' column from the train_dataset and test_dataset is removed because it serves as an identifier and does not provide valuable information for the prediction.

3.5 Train-Test Split

We will be further splitting the train data, dividing it with a 70-30 ratio. As the test set does not have labels, we created a validation set from the training data to allow us to estimate how well the model might generalize to unseen data.

The train-test split here is also done before the text preprocessing, encoding (one-hot-encoding and target encoding) and feature scaling to prevent data leakage.

3.6 Encoding Categorical Variables

We will be encoding the remaining categorical variables to make it suitable to for input into machine learning models. One-hot-encoding will be applied to features with fewer distinct categories (continent, content_rating, title_year) while target encoding will be applied to features with more distinct categories (language).

One-hot-encoding creates binary columns for each respective category, and hence is not likely to work well for features having many distinct categories because it will lead to an increase in dimensionality.

Target encoding meanwhile replaces the categorical variables with the mean of the target variable for each category, hence allowing the models to capture meaningful relationship between the predictor and target variable and reducing dimensionality.

3.7 Feature Scaling (Numerical Data)

Feature scaling for numerical data will be applied. To prevent features with larger units dominating features with smaller units during distance calculation, we will be using Min-Max scaling for scaling the numerical features in our dataset. Numerical features will be scaled to a range of 0-1.

4. Feature Selection

4.1 Undersampling & Oversampling

Undersampling and oversampling techniques will be used to address the class imbalance of imdb_score_binned, as mentioned in EDA. A large class balance may cause models to struggle to effectively learn patterns from minority classes and lead to biased predictions and reduced model performance.

Therefore, undersampling and oversampling techniques will be used, where undersampling would reduce the size of the classes which are overrepresented, and oversampling will increase the size of the classes that are underrepresented, overall balancing out the distribution.

However, synthetic samples will only be generated for the train_dataset, not test_dataset, because synthetic samples do not belong to the target distribution.

5. Model Selection

We will be running 4 machine learning models, and thereafter test and evaluate the performance of the models. The best model will be selected. We will also be running hyperparameter tuning in the next section.

After which, we will be evaluating each respective model with the test set.

5.1 Gaussian Naïve Bayes Classifier

Gaussian Naive Bayes (GNB) is a supervised algorithm based on the Bayes Theorem, with an assumption of independence between predictor features. Computation is hence

tractable and requires a small amount of training data in estimating the classification parameters. GNB is easy to implement and a good choice for classifications with features that approximately follow a gaussian distribution.

5.2 K-Nearest-Neighbours Classifier

K-nearest-neighbors (KNN) is a supervised machine learning algorithm based on the principle of identifying the k data points nearest to a given query point in the training set, and using their labels in determining the label of the query point. KNN does not make assumptions about the underlying data distribution.

5.3 Random Forest Classifier

Random Forest is a supervised machine learning algorithm based on the construction of a series of decision trees during training, and outputting the mode of the classes of the individual trees for classification tasks. Majority voting will determine the final prediction for classification tasks. Overfitting is reduced and generalisation performance of the random forest classifier is improved through building each tree from a random subset of training data.

5.4 AdaBoost Classifier

AdaBoost iteratively trains a sequence of weak classifiers using weighted versions of the train set, and each subsequent classifier will focus on training instances that are misclassified by previous classifiers. A higher weight will be assigned to misclassified instances, thereby building a strong classifier in the process. AdaBoost handles complex datasets particularly well as it focuses on the instances that are most informative and is less prone to overfitting.

	Model	Dataset	Accuracy	Precision	Recall	F1 Score
0	Gaussian Naive Bayes	Original	0.533851	0.507173	0.533851	0.519288
1	Gaussian Naive Bayes	Resampled	0.533851	0.507173	0.533851	0.519288
2	K-nearest Neighbors	Original	0.591565	0.520376	0.591565	0.521957
3	K-nearest Neighbors	Resampled	0.231964	0.130896	0.231964	0.158108
4	Random Forest	Original	0.614872	0.646614	0.614872	0.510035
5	Random Forest	Resampled	0.609323	0.571528	0.609323	0.561079
6	AdaBoost	Original	0.631521	0.559973	0.631521	0.579363
7	AdaBoost	Resampled	0.267481	0.563797	0.267481	0.294738

Table 4 – Performance of models before hyperparameter tuning

6. Hyperparameter Tuning

The models are generally performing better on the original trainset as compared to the original trainset with oversampling and undersampling techniques applied. Therefore, we have decided to conduct hyperparameter tuning on the original trainsets only.

We then used random search which selects and evaluates random combinations of hyperparameters.

The scoring metric used for evaluating the performance of the hyperparameter tuning chosen is F1-Score. F1-Score is chosen because it provides a harmonic mean between precision and recall, accurately predicting imbd_score_binned across all classes while at the same time being particularly useful for imbalanced datasets.

Cross-validation will repeatedly run by split the dataset into 5 folds, training the model on k-1 folds and evaluating it on the last fold. By averaging the evaluating metrics across the 5 folds, it allows us to obtain a reliable estimate of the machine learning model's performance.

	Model	Dataset	Accuracy	Precision	Recall	F1 Score
0	Gaussian Naive Bayes	Original	0.533851	0.507173	0.533851	0.519288
1	K-nearest Neighbors	Original	0.559378	0.545129	0.559378	0.550012
2	Random Forest	Original	0.615982	0.653642	0.615982	0.504224
3	AdaBoost	Original	0.621532	0.573719	0.621532	0.582526

Table 5 – Performance of models after hyperparameter tuning

7. Model Evaluation

To determine the best performing model, we will be using Accuracy and F1-Score, since F1-Score already considers the harmonic mean of precision and recall. We can see that AdaBoost with the original trainset is the best performing model (Accuracy = 0.621532, F1-Score = 0.582526)

8. Limitations

Although oversampling and undersampling techniques may address class imbalance, applying oversampling and undersampling to address class balance generates synthetic samples, which inadvertently introduces noise and cause overfitting. This is because synthetic samples may not represent the underlying data distribution, hence leading to inaccuracies especially if samples generated are too similar to existing samples.

9. Areas Of Improvement

9.1 Grid Search

Grid search would offer a more exhaustive search through the hyperparameter space to choose the most optimal set of hyperparameters for each model more effectively.

9.2 Check For Overfitting/Underfitting

We could have evaluated train-test loss of the models to assess the generalisation performance of the models and detect signs of overfitting or underfitting.

If the model is underfitting, the training loss will stagnate till the end of training. If the model is overfitting, training loss will continue to decrease while validation loss decreases till a point and increases again, suggesting that the model is becoming less able to generalise with new data.