COMP30027 REPORT

Project 2: IMDB Movie Rating Prediction

Student numbers: 1266382, 1309191

Kaggle team: “夺命十三枪😠”

1. Introduction:

This project aims to predict IMDB movie ratings using a variety of movie features. Five predictive models were developed based on the training dataset to predict a test dataset. The report is structured into six sections. The "Methodology" section details the data preprocessing steps and outlines the conceptual framework of the models. The "Results" section presents the performance outcomes of these models. "Discussion and Critical Analysis," delves into the limitations of the current models and explores potential improvements that could enhance their predictive accuracy.

2. Methodology

2.1 Data preprocessing

Before doing anything, inspect the dataset first.

The dataset comprises 26 features including the 'label', IMDb score, with a mix of categorical data (e.g., names, country) and numerical data (e.g., gross income, Facebook likes). While inspecting the labels, it was observed that the distribution is highly skewed, with the majority of labels classified as '2'. This imbalance could potentially affect the performance of certain models, necessitating adjustments or specific methodologies to address the issue.

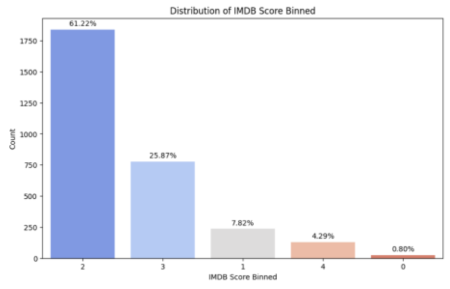
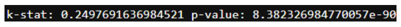


Figure 1: Score Distribution

- Deal with missing values

The training data contains missing values and some zeros that may represent missing data at random. For missing values in the 'language' feature, we fill them by making an educated guess based on the corresponding country. For zero values, we conducted a Kolmogorov-Smirnov test to assess normality and replaced non-normally distributed zeros with the median.



```
k-stat: 0.2497691636984521 p-value: 8.382326984770057e-90
```

- Full dataset

Given the encodings available, the most straightforward approach is to replace all categorical data with its corresponding encoding, resulting in a dataset of significant dimensions. Despite its large size, certain models may be effectively utilized to handle this complexity.

- Feature selection

With the extensive number of features, the training process can become overly complex for many models to manage efficiently. Therefore, feature selection is necessary to streamline the model training and enhance performance.

Names:

With director and actor names, we observed a vast array of unique entries. The excessive variety may not significantly aid predictive modeling due to their abundance. Additionally, the average scores of directors with the most films do not vary significantly.

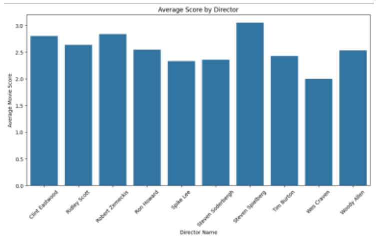
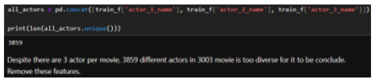


Figure 2: Average Score Director



```
all_actors = pd.concat([train_f['actor_1_name'], train_f['actor_2_name'], train_f['actor_3_name']])

print(len(all_actors.unique()))
```

```
3859
```

Despite there are 3 actor per movie, 3859 different actors in 3003 movie is too diverse for it to be conclude. Remove these features.

Genres:

The variability in genres is minimal and not significantly influential, suggesting it might be useful but is currently average. We will remove it for now.

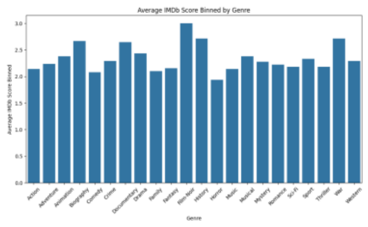
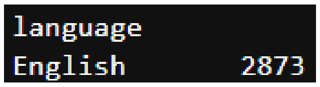


Figure 3:Average Score Genre

Plot keyword:

Observed 5892 different keywords, too diverse, remove.

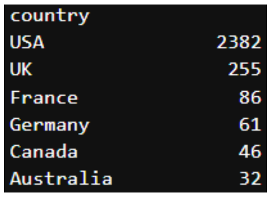
Language:



```
language
English       2873
```

2873/3003 English, remove.

Country:



```
country
USA          2382
UK            255
France         86
Germany        61
Canada         46
Australia      32
```

The majority of movies are from the USA and UK. Comparing the average ratings of countries with the most movies shows similar results, so we will remove this feature.

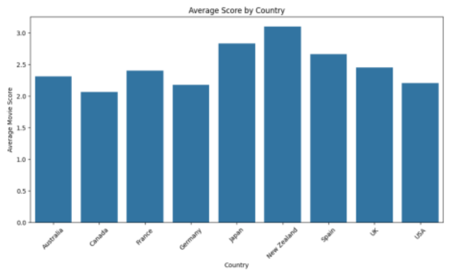
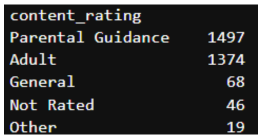


Figure 4:Average Score Country

Content rating:

Too many content rating category, combine and examine.



```
content_rating
Parental Guidance    1497
Adult                1374
General                68
Not Rated              46
Other                  19
```

Average score similar after examine, remove this feature.

Movie title:

As movie title is unique to each movie but does not determine the movie's quality, it might be related to movie's watch count but does not directly affect movie's rating. Hence, remove.

Since all that is left is numerical, check correlation and remove the highly correlated ones.

With these selections, we have a second dataset that can compare with, and default for models involving distance measurements due to the first dataset’s massive dimensionality.

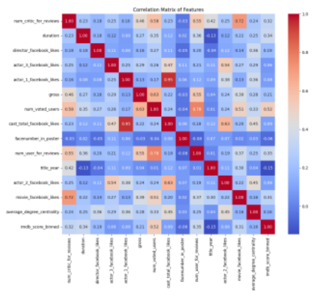


Figure 5:Correlation Matrix of Features

2.2 K Nearest Neighbor

The K-Nearest Neighbors (KNN) algorithm is straightforward and easily understood, often effective in capturing complex interactions among variables. However, its performance may degrade with high-dimensional data and large datasets.

2.3 Support Vector Machine

Support Vector Machines (SVMs) are powerful for handling high-dimensional datasets, which is advantageous given the extensive number of features derived from the IMDB dataset.

2.4 Random Forest

Random Forest is not sensitive to normalization and can provide insights into feature importance, helping to identify the most influential features for predicting IMDB scores.

2.5 Stacking Model

Stacking combines multiple models to leverage their individual strength, potentially leading to better performance.

2.6 Neural Networks

Neural networks are computational models that mimic the human brain, effectively recognizing patterns in complex datasets like the IMDB dataset.

3. Result

3.1 K Nearest Neighbor

By normalizing the selected dataset using 'MinMaxScaler' and 'StandardScaler', and used to model KNN, a plot of k-value versus accuracy was created to select the best-performing k-value. According to the figure, the highest accuracy using the normalized method is 0.664, while for the Min-Max normalized method it is 0.650.

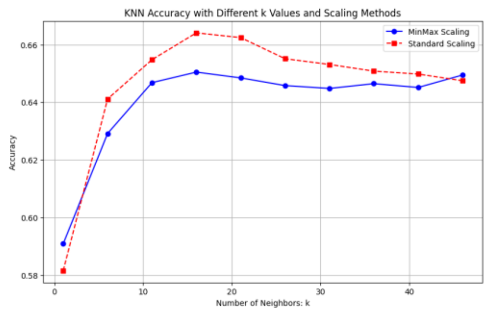


Figure 6:KNN Accuracy with Range of K

A confusion matrix was plotted to show how the model's predictions compare to the true labels. Most samples are predicted to be 2. This indicates the skewed pattern in the label caused KNN to include other labels into the majority ‘neighborhood’ of ‘2’, providing a reference and benchmark for the subsequent development of more complex models.

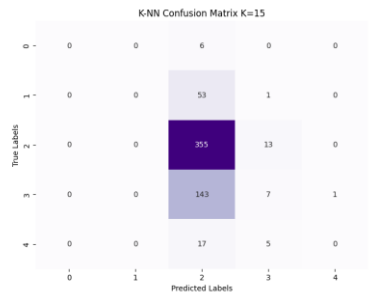


Figure 7:KNN Confusion Matrix

3.2 Support Vector Machine

The SVM model was also trained using selected normalized data, and its performance metrics were tabulated. Like the KNN model, SVM performs better when predicting the label 2 but struggles with other labels due to the skewed label. Both SVM and KNN exhibit low accuracy and high label errors, this might be due to the distance measure models on dataset with unevenly distributed label.

Accuracy:	0.611
Precision:	0.463
Recall:	0.611

Table 1:Classification report of SVM

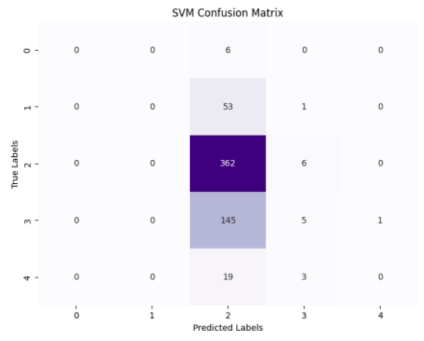


Figure 8: SVM Confusion Matrix

3.3 Random Forest

Based on a tree model, random forest is capable of handling massive dimensionality of full dataset with embedding, we modeled random forest on dataset 1 first:

Accuracy:	0.636
Precision:	0.553
Recall:	0.636

Table 2:Classification report of RF on full

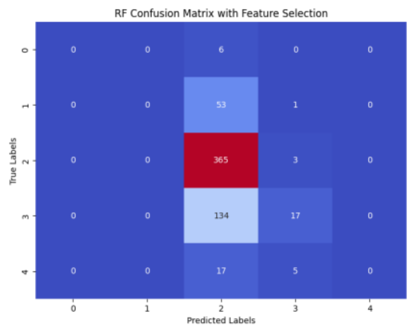


Figure 9:RF data1 Confusion Matrix

Comparing to RF on dataset with feature selection:

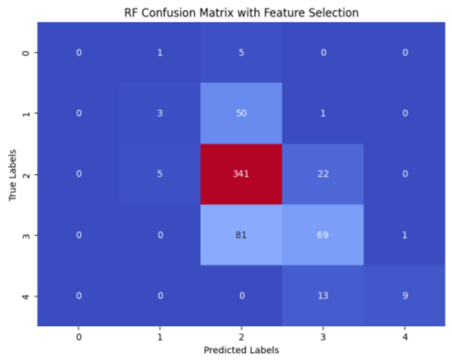


Figure 10: RF data2 Confusion Matrix

With accuracy of 0.702, feature selection does prove to be useful, from the confusion matrix, it shown that with feature selection, random forest is better at predicting label '3' correctly.

3.4 Stacking model

As tree model appears more suitable for this dataset, we are stacking Logistic, Decision Tree, Random Forest, Gradient Boosting Machine together and model on the dataset with feature selection.

Normalization is used for consistency.

The stacking model achieve a slightly better result at accuracy of 0.705, and from the confusion matrix we can see that it does a better job at identifying label '3' and label '4'

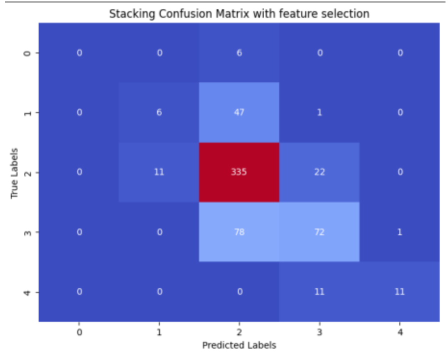


Figure 11: Stacking Confusion Matrix

3.5 Neural Networks

With neural network, we implemented a simple dense neural network, with a fast convergence optimizer 'nadam' accompanied by early stop mechanism to prevent overfitting, this results a similar outcome of accuracy 0.691

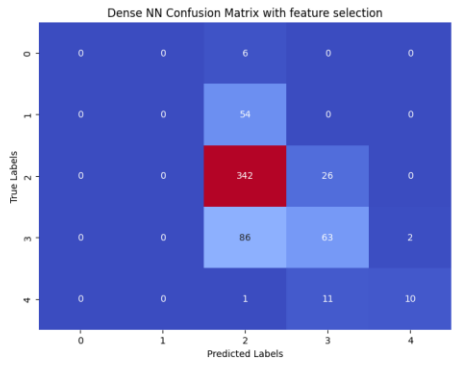


Figure 12:NN confusion matrix

However, with the introduction of embedding layer technique, we can incorporate some of the features that got removed.
When add genre into the embedded layer, the performance increases with accuracy of 0.734, and as confusing matrix suggest, it does a better job at predicting label '1' and '3'.

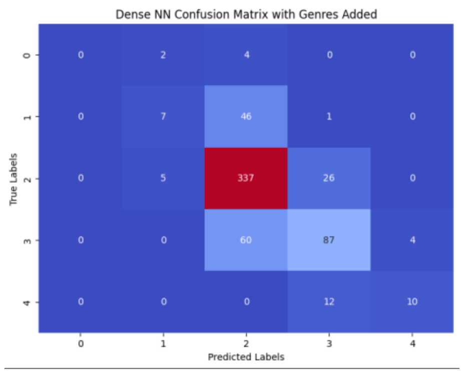


Figure 13:NN Confusion Matrix with Genres

4. Discussion and Critical Analysis

- KNN and SVM

These two models exhibit the poorest performance, primarily because both involve distance measurements in their algorithms.
In this project, KNN is used as the baseline model due to its simplicity. However, its performance is suboptimal because its distance-based approach can be biased towards the majority group, often misclassifying instances into that group. Another limitation of KNN is its poor handling of high-dimensional data. Despite feature selection reducing the dataset to 20+ features, this is still considerable, and normalization does not always equalize the importance of features effectively. For instance, gross income might have a greater predictive weight than Facebook likes, but KNN treats all features with equal importance.
On the other hand, SVM does not directly compute distances between instances but rather optimizes the margin between classes. Despite this, SVM still struggles with class

imbalance as it may fail to set an appropriate margin for minority groups. Furthermore, SVM's binary nature complicates its application in multi-class settings. The need to aggregate decisions from multiple SVMs can lead to misalignments in decision boundaries, resulting in decreased accuracy and higher error rates in multi-class classification.

- Tree based models

Random forest, was selected as the subsequent model due to its ability to manage large datasets with high dimensionality through its ensemble approach, employing multiple trees each utilizing a subset of instances. This configuration enhances the model's capability to handle diverse data characteristics effectively.
Initially, the model was tested on a dataset containing all encoded features; however, performance was suboptimal, likely attributable to excessive noise from the high-dimensional data.
Subsequently, employing a dataset refined through feature selection, Random Forest outperformed both KNN and SVM. This improvement was due to its reduced susceptibility to noisy or irrelevant variables. By concentrating on the most pertinent features, Random Forest constructed more accurate and universally applicable decision trees.
This focused methodology significantly elevated the model's accuracy over KNN and SVM, which tend to struggle with feature redundancy and are more sensitive to dataset noise.
Additionally, Random Forest's ability to average the outcomes from multiple decision trees aids in mitigating overfitting, thereby enhancing the robustness of the model for predictions across diverse data points.

- Stacking model

To further enhance performance, we decided to implement a stacking model, predominantly utilizing tree-based models due to their proven effectiveness.
The stack consisted of Logistic Regression, Decision Tree, Random Forest, and Gradient Boosting Machine as base models, with Logistic Regression serving as the final estimator.
This arrangement resulted in only a slight improvement in performance. The incremental gains could be ascribed to the increased complexity of the model and the already high performance of the individual base models, indicating that additional complexity does not necessarily lead to substantial performance improvements.
Additionally, the learning curve (Figure 14) reveals a discrepancy between the high accuracy on the training set and the lower accuracy on the validation set, indicating an overfitting issue in the model.
Moreover, employing Logistic Regression as the final estimator may not effectively utilize the collective strengths of the base models.

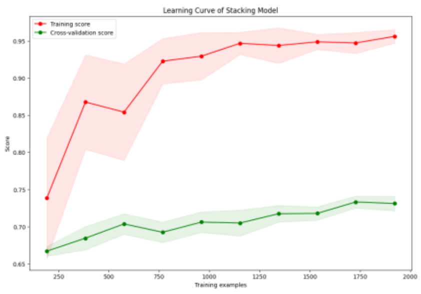


Figure 14:Learning Curve of Stacking Model

- **Neural Network**

After observing only a minimal performance increase with the stacking model, we deployed another powerful model: Neural Networks. Specifically, we opted for a Dense Neural Network (DNN) because its fully connected structure can capture all possible interactions between features. This model showed decent performance on the dataset post-feature selection. For model optimization, we chose the 'Nadam' optimizer due to its rapid convergence properties, attempting to counteract noise from the uneven dataset. We also implemented an early stopping mechanism to mitigate potential overfitting associated with the 'Nadam' optimizer.
Regarding other parameters, we maintained the default settings, as we did not find a more optimal way to adjust them. Another significant advantage we leveraged is the embedding layer in neural networks. During feature selection, we removed many features, some of which might contain useful information. Incorporating the embedding layer substantially enhanced the model's performance, leading to our final predictions, which we uploaded to Kaggle.

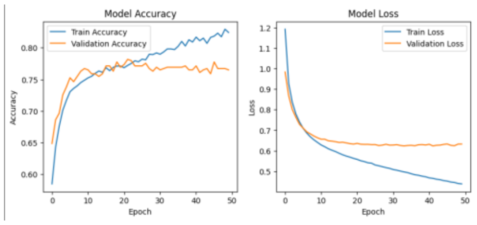


Figure 15:Learning Curve of Dense NN

The learning curve depicted in the figure above suggests that the issue of overfitting persists, as evidenced by the disparity between validation accuracy and test accuracy, and the validation loss ceases to decrease beyond a certain point.

- Limitations

This dataset contains numerous names, including movie titles and names of directors and actors. Intuitively, these names could convey information about movie ratings, but given the vast number of unique names and the complexity involved in processing these names into meaningful and interpretable data, it is challenging to extract useful information from these features. Another issue is the uneven distribution of labels. After examining the original dataset available online, we noted that the scores range from 0 to 10, with most scores concentrated between 4 and 8. Employing a method that bins the data equally across this range can obscure finer distinctions, such as differentiating between movies rated 4.5 and 4.6.

5. Conclusion

This project aimed to predict IMDB movie ratings using various machine learning models from a set of features from the dataset.

Throughout the course of the study, we developed and evaluated five different models: K-Nearest Neighbors, Support Vector Machine, Random Forest, a Stacking model, and a Dense Neural Network. Each model was tailored to maximize its strengths in handling the complexities inherent in the predictive task.

Our findings indicate that while models like KNN and SVM provided a foundational understanding of the dataset, they were hampered by issues such as high dimensionality and class imbalance, leading to suboptimal performance. In contrast, tree-based models, especially Random Forest, showed a remarkable ability to manage high-dimensional data, yielding more accurate predictions after feature selection was applied. This reinforces the importance of feature engineering in improving model performance.

The stacking model, while slightly improving performance, highlighted the limitations of increasing complexity without corresponding gains in predictive accuracy. This was further evidenced by the modest enhancements seen with the Dense Neural Network, which, despite its advanced capabilities in handling complex interactions between features, also faced challenges such as overfitting, as indicated by the learning curves.

The project underscored several critical aspects of predictive modeling: the importance of feature selection, the challenges of handling class imbalances and high-dimensional data, and the nuanced balance between model complexity and generalization. Moving forward, these insights will guide further refinements in our modeling approaches and suggest a continued focus on feature engineering and model tuning to enhance the predictive accuracy of our algorithms.

6. Reference

-English, M. L. in P. (2023, June 3). Deep Learning Course — Lesson 7.5: Nadam (Nesterov-accelerated Adaptive Moment Estimation). Medium. https://medium.com/@nerdjock/deep-learning-course-lesson-7-5-nadam-nesterov-accelerated-adaptive-moment-estimation-efe9050d5b9b

- M, M. (2020, October 31). How to Use Stacking to Choose the Best Possible Algorithm? Analytics Vidhya https://www.analyticsvidhya.com/blog/2020/10/how-to-use-stacking-to-choose-the-best-possible-algorithm/#:~:text=Stacking%20refers%20to%20a%20method,performance%20of%20each%20individual%20model.

- Thakur, A. (2023, April 28). Keras Dense Layer: How to Use It Correctly. W&B. https://wandb.ai/ayush-thakur/keras-dense/reports/Keras-Dense-Layer-How-to-Use-It-Correctly--Vmlldzo0MjAzNDY1