University of Southampton Malaysia

Machine Learning Coursework 1

Name: LIM SHI XUN

STUDENT ID: 33351341

DATE: 4th December

Contents

1. Introduction:

1.1. Problem statement

Build a predictive model that can accurately predict the quality of a movie by categorize it into 5 different categories.

- IMDB score 1-2: Categorized as "Poor" movies or 0 after label encoding.
- IMDB score 2-4: Categorized as "Below Average" movies or 1 after label encoding.
- IMDB score 4-6: Categorized as "Average" movies or 2 after label encoding.
- IMDB score 6-8: Categorized as "Good" movies or 3 after label encoding.
- IMDB score 8-10: Categorized as "Excellent" movies or 4 after label encoding.

Before implementing the algorithm, data cleaning processes such as handling missing values and outliers needed to be done so that the input data for the machine learning algorithm is reliable and robust.

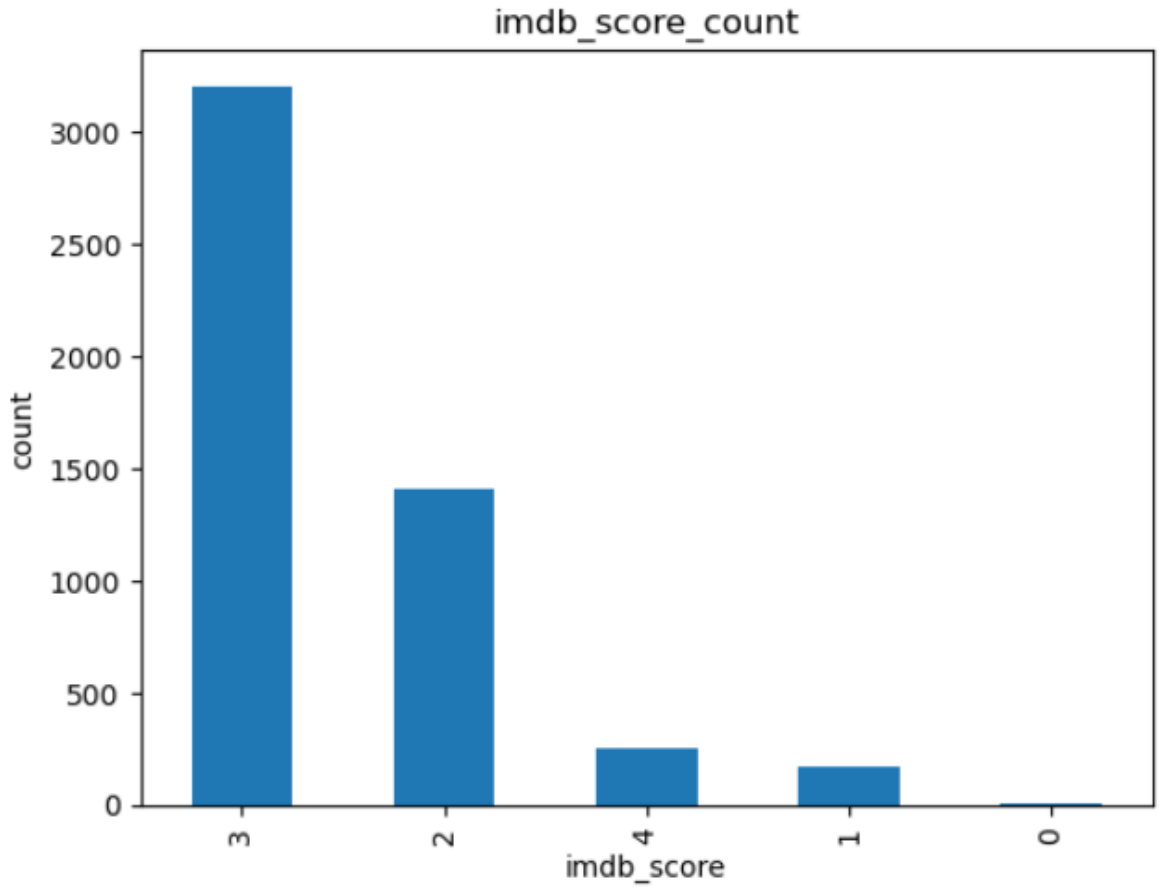
1.2. Dataset

This movie dataset consists of 5043 rows, 28 unique features across 100 years in 66 different countries. The target feature is imdb_score which is IMDB score of the movie on IMDB. Below is the meaning of each feature.

Feature name	**Meaning**
movie_title	Title of the Movie
duration	Duration of the movie in minutes
director_name	Name of the Director of the Movie
director_facebook_likes	Number of likes on the Director's Facebook Page
actor_1_name	Primary actor starring in the movie
actor_1_facebook_likes	Number of likes on Actor 1's Facebook Page
actor_2_name	Other actor starring in the movie
actor_2_facebook_likes	Number of likes on Actor 2's Facebook Page
actor_3_name	Other actor starring in the movie
actor_3_facebook_likes	Number of likes on Actor 3's Facebook Page
num_user_for_reviews	Number of users who gave a review
num_critic_for_reviews	Number of critical reviews on IMDb
num_voted_users	Number of people who voted for the movie
cast_total_facebook_likes	Total number of Facebook likes for the entire cast of the movie
movie_facebook_likes	Number of Facebook likes on the movie's page
plot_keywords	Keywords describing the movie plot
facenumber_in_poster	Number of actors featured on the movie poster
color	Film colorization (e.g., 'Black and White' or 'Color')
genres	Film categorization (e.g., 'Animation', 'Comedy', 'Romance')
title_year	The year in which the movie was released (from 1916 to 2016).
language	Language of the movie (e.g., English, Arabic, Chinese, French)

country	Country where the movie was produced
content_rating	Content rating of the movie
aspect_ratio	Aspect ratio in which the movie was made
movie_imdb_link	IMDb link of the movie
gross	Gross earnings of the movie in Dollars
budget	Budget of the movie in Dollars
imdb_score	IMDb Score of the movie on IMDb

Check for class imbalance in target feature.



We can see that the class in target feature is highly imbalance. The ratio between imdb_score of 4 (Good) and 1 (Poor) is 457.86. Because the target is imbalance, we need to be careful not to remove any of the minority classes such as 0(Poor), 1(Below Average) and 4(Excellent).

2. Methodology:

2.1. Data pre-processing

2.1.1. Duplicates

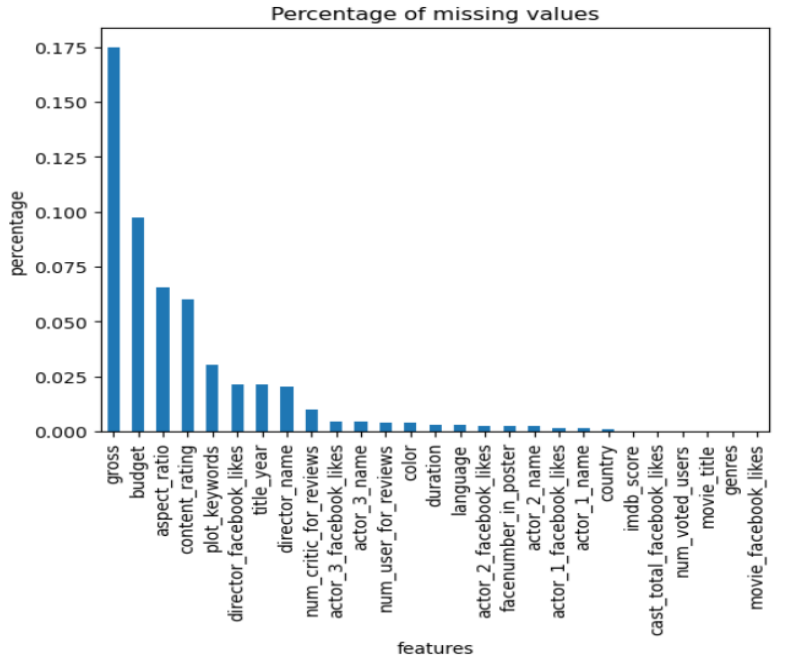
First step is to remove duplicated rows. Since it is impossible for 2 movies to have the exact same budget, gross and facebook likes, duplicated values might cause several problems such as overfitting and biases towards the most duplicated class. This is because multiple of the same point will shift the weight towards the point, making the point seem more important than others.

2.1.2. Check for miss input in categorical data.

\xa0 is at the end of all movie title, even though it does not directly improve the quality of the data, we remove all the \xa0 to improve the consistency and readability of the data.

2.1.3. Handling Missing Values

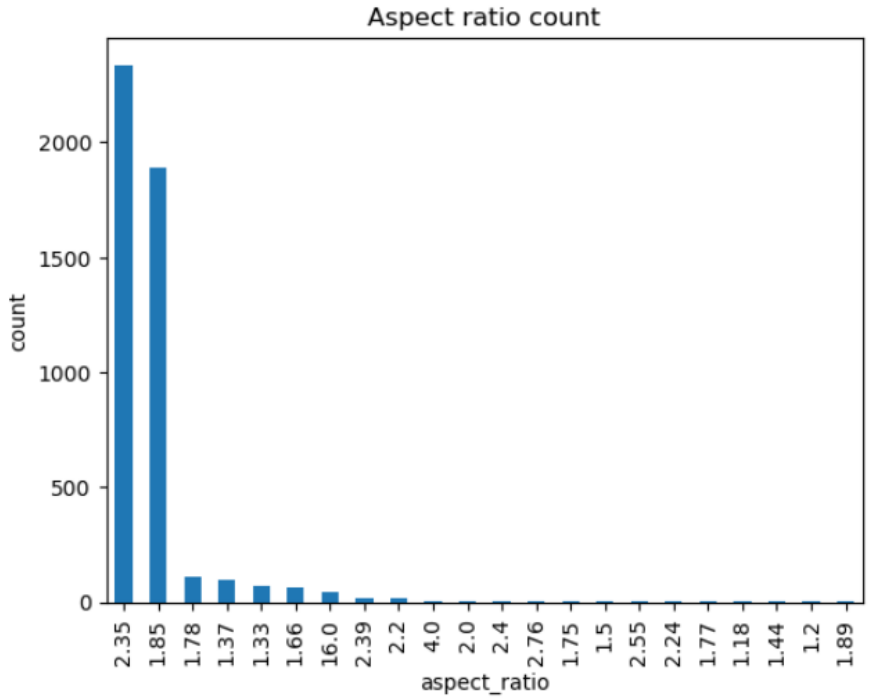
Before Handling any missing value, we should remove movie_imdb_link because it represents the link to each individual webpage. The high cardinality makes the feature useless in predicting imdb_score. So, it is safe to drop the column movie imdb link.



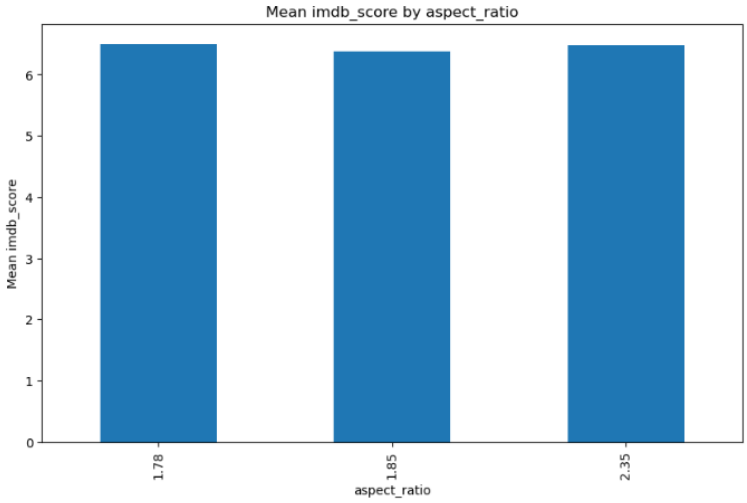
Here we can see **gross** has the highest missing value percentage of **17% Budget** has 10% missing value, **aspect ratio** has **7%**, content rating has **6%** The rest are less than 3% Most of the data are non-missing data, but gross and budget has way too much missing value, **> 10%**, filling all of them with imputation might introduce bias.

Aspect Ratio

Most of the aspect ratio are **2.35** and **1.85** we can group all the other aspect_ratio into 1.78 and evaluate the impact of aspect ratio on imdb_score



Most of the aspect ratio are **2.35** and **1.85** we can group all the other aspect ratio into 1.78 and evaluate the impact of aspect ratio on IMDB score.



Here we can see that aspect ratio is almost equally distributed, means that different aspect ratio has the same mean IMDB score, which might suggest that aspect ratio has little to no impact on IMDB score. So, no matter which aspect ratio we choose, it is going to give the same average IMDB score.

Content Rating

There is some TV rating in this movie dataset, we should remove them since they are not useful in predicting movie IMDB score and might introduce noise due to difference in budget, duration, etc.

Besides that, we can also group the some of the rating together since they are just different version of each other, for instance, NC-17 is the newer version of X rating.

Plot Keywords

Plot keyword seems to be similar to genre. So first, we calculate the correlation between plot keyword and genres using Cramer's V. The reason of using Cramer's V is because they are both nominal categorical feature which has no ordering and both columns have more than 2 unique values [1]. The Cramer's V value between them is 0.42, which is consider highly correlated according to [2]. Besides that, there is also way too many unique values in plot keyword which does not help in generalizing the model. Based on the high correlation and high unique values, it is reasonable to drop plot keywords. This can help reducing noise and reducing dimensionality.

Table 2

Interpretation of Phi and Cramer's V.

Phi and Cramer's V	Interpretation
>0.25	Very strong
>0.15	Strong
>0.10	Moderate
>0.05	Weak
>0	No or very weak

Table 1: Adapted from [2]

Categorical data with small missing value

When dealing with categorical with low unique value such as color, and language and with small number of missing value (less than 1%), we can fill them with mode. Mode is the most occurring item in the column, so filling small number of missing values with mode will not affect the overall distribution of the data and cause biases.

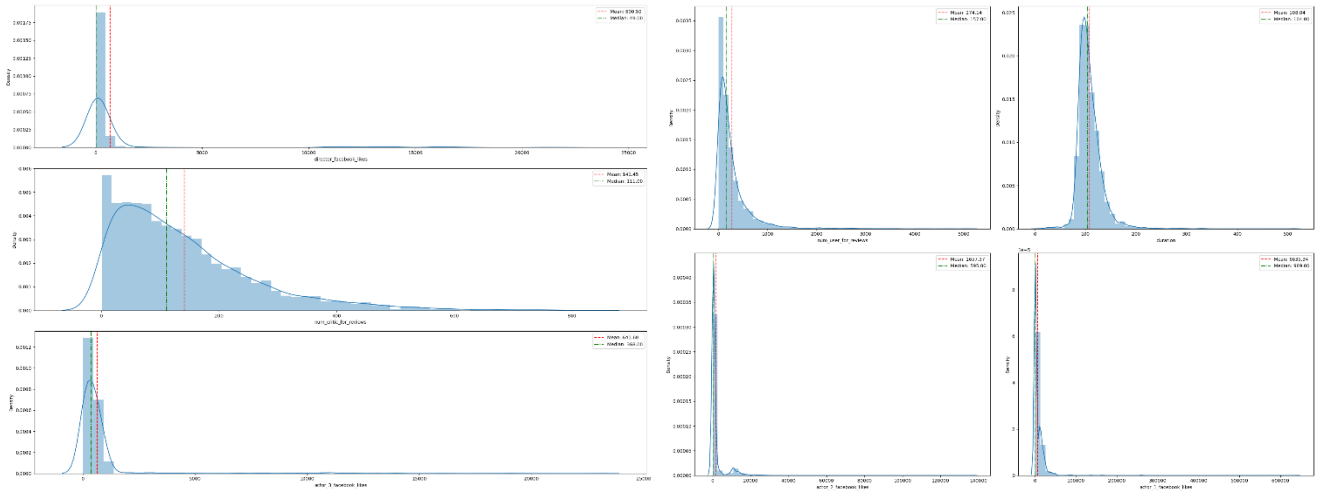
Beside categorical value, we imputate numerical value with small amount of unique value such as title year and number of faces in poster with mode to preserve the distribution of the data since the missing value is only around or less than 1%.

Names

We can't use mode to fill the missing value in names because of the high uniqueness.

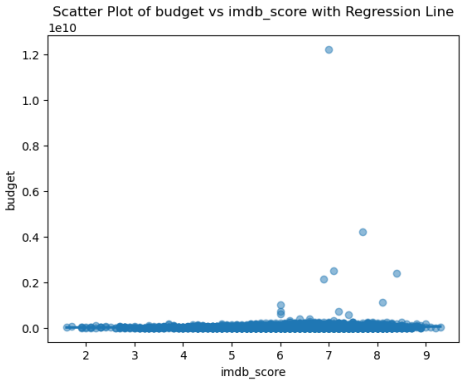
Missing names might indicate they are either anonymous, or not well known enough to be record down, or the credit of the movie does not have their name. Which might suggest the quality of the movie and the overall effort when into crediting the cast and crew. So, we will replace all the missing values in name features into a new class called Unknown.

Numerical data



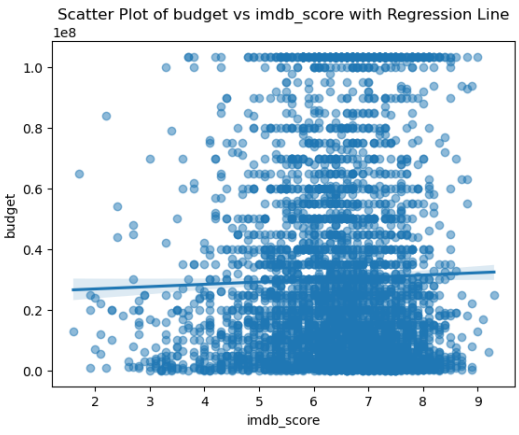
Before deciding numerical data with mean, or median, we first look at the distribution of the data. From the diagram above, we can see that all the data is skewed toward the left side which means that the data is more concentrated at the left side with lower value. We can also see the right side of each plot is extend over the x axis, which indicates the presences of outlier. With the presence of outlier and skewed distribution, it is better to use median than mean. Median is the middle value of the entire column while mean is the average of the entire column. Outlier with relatively large value will skew the mean calculation towards it while median does not affect by it.

Budget



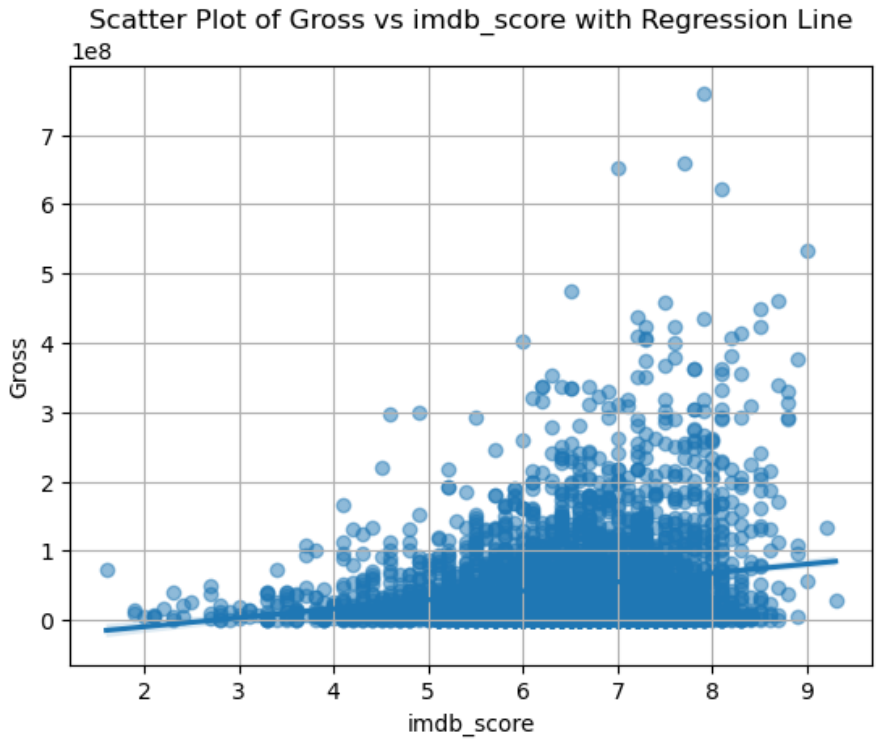
The diagram above shows the scatter plot between budget and IMDB score. But because of the extreme outliers with 11 billion budgets, we cannot see any correlation, so we cap the outlier first.

After capping the outliers, the diagram shows budget has very little correlation with IMDB score. Budget has 487 missing value which is around 10% of the total rows. We cannot imputate it with mean or median because it will introduce bias toward the imputated value. Based on the large amount of missing value and extremely low



correlation with target feature, it is reasonable to drop budget.

Gross



From the scatter plot above, we can see there is slight positive correlation between gross and IMDB score. This mean that as IMDB score increases, gross also tend to increase. But because gross has 874 missing values which is around 17% of the total dataset, we cannot simply imputate using mean or median. Dropping the column might not be the best choice here since there is some correlation between gross and target feature and based on general knowledge, even though not also, it is more likely for critically acclaimed movie to have high gross at the same time for example, Avatar and Avengers. Thus, we will use model-based imputation.

The model we will be using to imputate must be less sensitive to outlier. Random Forest and Gradient Boost was selected and tested.

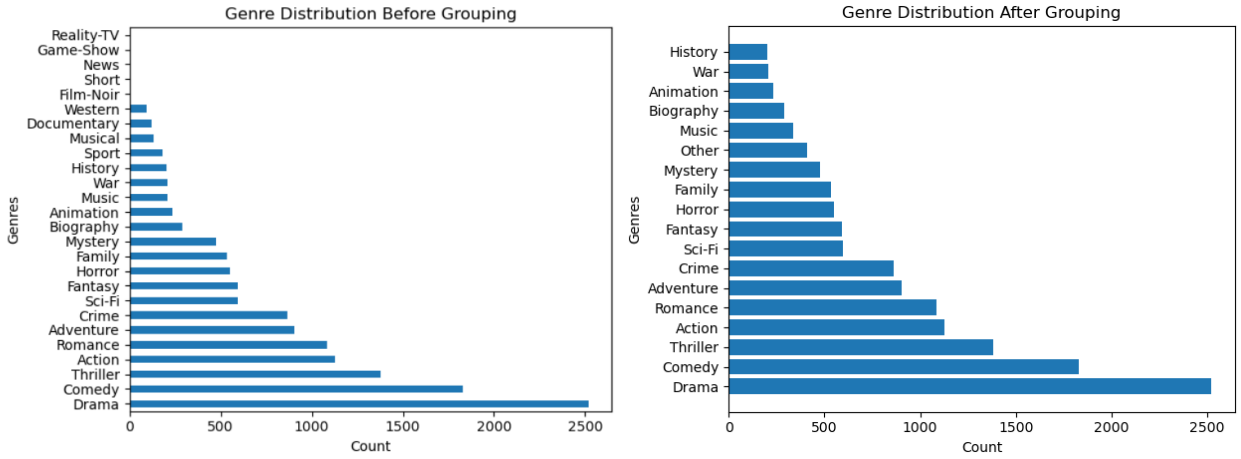
To imputate the data, we use other features that are moderately correlated to gross such as number of voted users, number of users for reviews and number of critics for reviews. Moderately correlated features have higher predictive power since as one increase for a certain amount another also tend to increase.

2.1.4. Feature Encoding

To fit features into a machine learning algorithm, we must convert every value in the dataset into numerical value because algorithm uses functions that only works on numbers and not strings or other object.

Genres

To encode genres, first we investigate the number of unique genres.

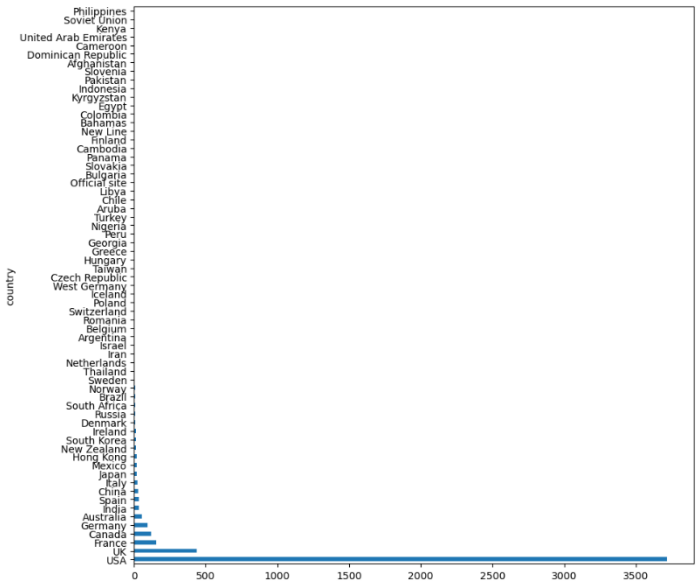


From the diagram above, we can see that there are 26 unique genres. We can combine music and musical into a single genre since they are very similar. We can combine the last genres into **others** because they have very few instances (< 200).

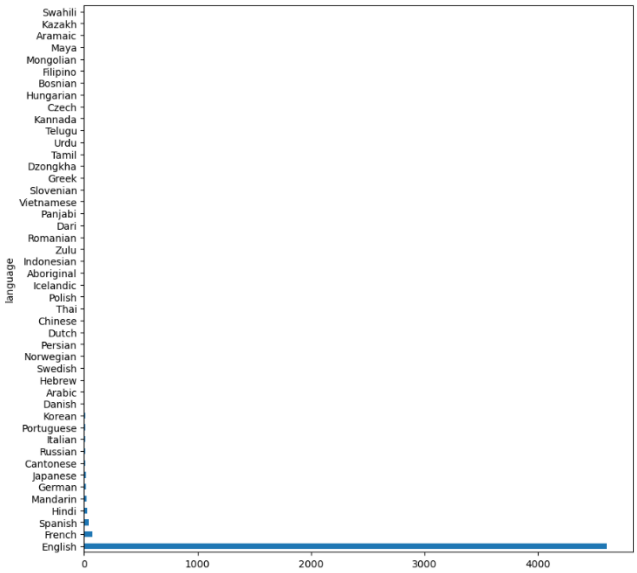
Because genres are a nominal categorical data, we cannot use label encoding since label encoding makes the label with higher value more important than other. We also do not use target encoding because it might cause data leakage due to direct usage of target value in predictor value which might cause overfitting. We use one-hot-encoding to encode genres because genres are nominal and the increase in dimensionality is reasonable.

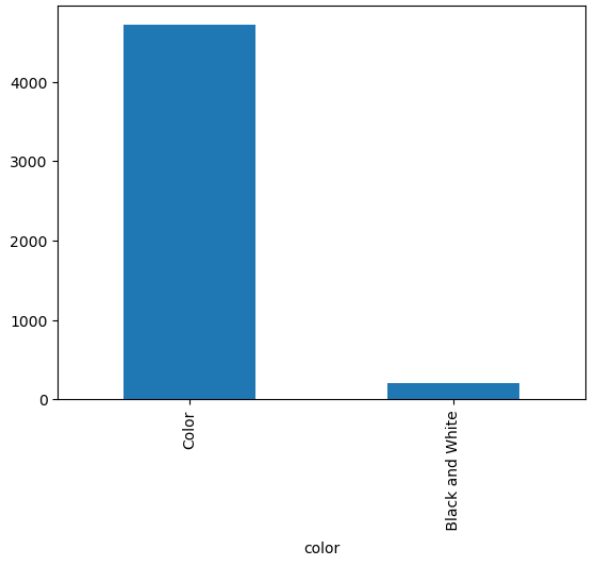
Country

First, we check the distribution of country. From the image, we can see that most of the movies are from USA (80%), UK (10%) and the rest of the country are less than 10%. We can first combine all other country into new class and then apply one-hot-encoding to country since they are also nominal categorical data.



Language & Color





From here we can see that more than 95% of the classes in language and color are just English and Color. Column with majority single class contributes very little to the predicting process because of the lack of pattern and variety in the column. Encoding them will not be a good idea because it will increase dimensionality of the data which can reduce performance and introduce noises that disturb the prediction. So, it is reasonable to drop language and color.

Names & Movie Title

```
actor_1_name :  2069
actor_2_name :  2989
actor_3_name :  3474
director_name :  2394
```

Names and movie title are both high cardinality features. Although we can use target encoding for high cardinality feature, but again data leakage is a very serious problem and might impact inference in different ways such as overfitting and inflated performance. Because of the high cardinality and being a nominal feature, we cannot use one-hot-encoding or label encoding which will cause curse of dimensionality and biases towards higher labeled class respectively. Besides that, names and movie title

are ambiguous where the same name does not necessary refer to the same person or movie. Thus, it is reasonable to drop all the names and movie title name columns.

2.1.5. Handling Outlier

Before handling outlier for numerical values, we need to identify the skewness of the feature by plotting distribution curves. If the distribution follows the normal / gaussian distribution, it is better to use z-score to identify the outlier because z-score make use of mean and standard deviation to identify the outlier. Generally, values that is 3 standard deviations away from the mean is consider an outlier. When the distribution a feature is skewed and contains outliers, it is better to use median and interquartile range (IQR) for identifying outliers because median is less sensitive to outliers, since calculation go median does not involve the extreme value of outlier. IQR on the other hand calculate the difference between 75 percentile and 25 percentiles (middle 50% of the data), which give us a range that helps identifying outlier (usually 1.5*IQR) that is less sensitive to outlier.

But before we investigate the distribution, we need to check if the max and min value of features make sense or not. We should not cap or remove outlier that are natural and not cause by entry error or sampling error because they are real data points and removing them might alter the nature of the data [3].

	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_1_facebook_likes	gross	num_voted_users	cast_total_facebook_likes	facenumber_in_poster	num_user_for_reviews	title_year	actor_2_facebook_likes	imdb_score	movie_facebook_likes
count	4924.000000	4924.000000	4924.000000	4924.000000	4924.000000	4.924000e+03	4.924000e+03	4924.000000	4924.000000	4924.000000	4924.000000	4924.000000	4924.000000	4924.000000
mean	141.168359	108.032088	684.896832	640.325751	6626.167953	4.259618e+07	8.410093e+04	9769.530869	1.359870	273.661251	2002.518887	1654.567628	6.425264	7487.784322
std	121.079929	24.185387	2813.753304	1644.679447	15147.163388	6.402622e+07	1.388952e+05	18273.593321	2.011965	379.122887	12.431582	4051.455763	1.115658	19328.844222
min	1.000000	7.000000	0.000000	0.000000	0.000000	1.620000e+02	5.000000e+00	0.000000	0.000000	1.000000	1916.000000	0.000000	1.600000	0.000000
25%	52.000000	94.000000	7.000000	133.000000	614.750000	3.446209e+06	8.560000e+03	1406.500000	0.000000	65.750000	1999.000000	280.750000	5.800000	0.000000
50%	111.000000	104.000000	49.000000	368.000000	989.000000	1.953418e+07	3.446000e+04	3104.000000	1.000000	157.000000	2005.000000	595.000000	6.600000	161.000000
75%	195.000000	118.000000	191.250000	635.000000	11000.000000	5.396449e+07	9.666300e+04	13921.250000	2.000000	326.250000	2011.000000	918.000000	7.200000	3000.000000
max	813.000000	511.000000	23000.000000	23000.000000	640000.000000	7.605058e+08	1.689764e+06	656730.000000	43.000000	5060.000000	2016.000000	137000.000000	9.500000	349000.000000

Features seem to have reasonable maximum and minimum value. But 511-minute movie duration seem to be unreasonable. So, we will cap the maximum duration to 300-minute max as there is only 3 extreme duration with > 300 minutes.

2.1.6. Scaling

Since we are trying to develop a model to categorize movie into 5 different predefined categories, we are going to use classifier model such as SVM, Logistic Regression, decision tree etc. But because of the presence of outlier in out predictor, we will be using model that are less sensitive to outlier, specifically any tree-based model such as XGBoost, Random Forest and more. Scaling in not needed as tree-based model does not benefit from scaled predictor because decision tree makes splitting decision by comparing the value with a threshold. For distance-based model like SVM would benefit greatly from scaling or normalization since the distance between 50 and 100 is different from 0.5 and 1.

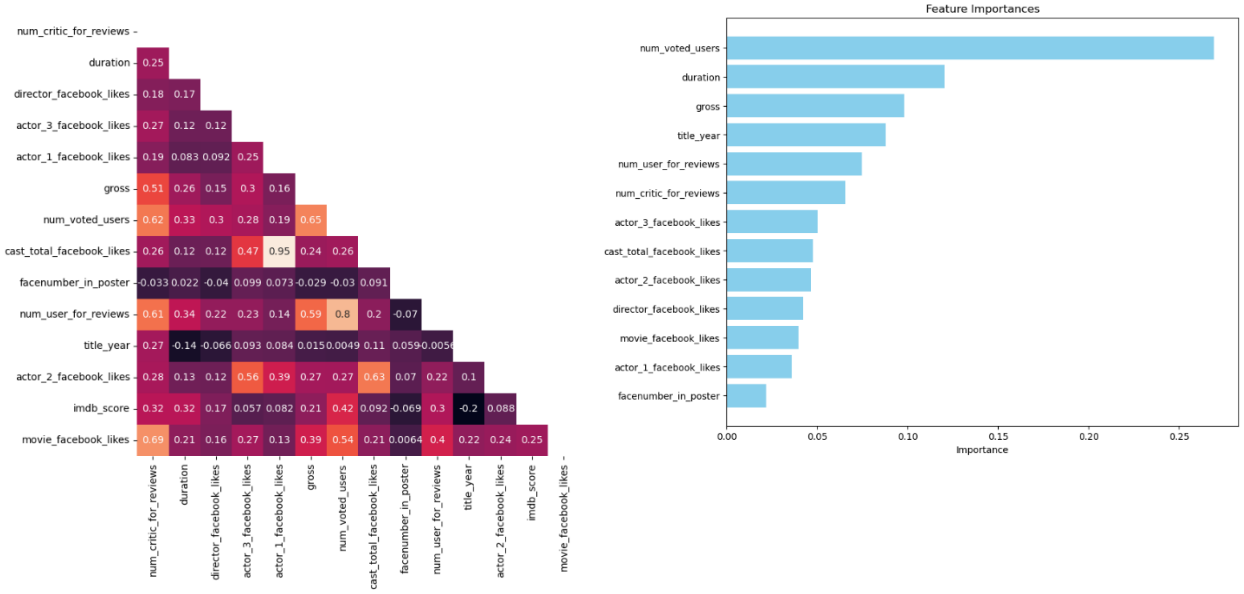
2.1.7. Feature Selection

We use feature selection rather than feature extraction method like PCA because we only want to remove the unimportant features. Feature selection allow us to do just that while retaining the

readability of the features and it is easy to implement. We use Random Forest to calculate the information gain for each feature.

From the information gain plot, we can see that face number in poster is the least informative feature. So, we will drop it to reduce dimensionality.

Besides that, cast total facebook likes also has extremely high correlation with actor 1 facebook likes (0.95), suggest that they might be redundant to each other. Thus, we will be removing cast total facebook likes.



2.2. Model Selection

To choose the right model for predicting IMSB score, first we need to identify what kind of data we have.

1. Our dataset has target feature (IMDB score), so we will be using supervised algorithm.
2. Our objective is to develop a classification model, so we will be using classification algorithm.
3. Our dataset consists of outliers, so we will be using a tree-based model, tree-based model is less sensitive to outlier, means that outlier will have less impact on the prediction compared to distance-based algorithm.
4. The final choices of algorithms are:
    a. Decision tree: A simple tree-based algorithm that make decision based on information gain.
    b. Random Forest: Ensemble algorithm that uses multiple decision tree, splitting training data by using bootstrapping across each decision tree and take the majority result. Help reduce overfitting.
    c. XGBoost: Ensemble algorithm that uses multiple weak learners, focus on reducing the error made by the previous model. A more effective version of gradient boosting

2.3. Splitting data

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)
```

70/30 split is used to split the data into training set (70%) and testing set (30%). This is because it provides 50% increase in testing set size compared to 80/20 split. And because our target feature suffers from class imbalance, a larger testing set also mean that there are more minority classes in the testing set. This is especially important because our objectives are to classify movies into 5 categories, which means that minority classes are just as important as the other. Besides that, **stratify=y** help preserves class distribution in both testing and training set.

2.4. Cross Validation

```
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)
```

For our cross-validation technique, we use StratifiedKFold, which is a K-Fold cross validator that preserve the ratio of each sample for each class in every fold. And the reason why we use n_split 5 is because the least populated class has only 7 elements. Using a higher number of splits might cause some fold lacking samples from the class.

2.5. Evaluation metric

Our target feature has imbalance class, when calculating the performance of the model, Accuracy might not be the suitable performance metric since it calculates the percentage of correctly predicted values to the overall values (TP+TN/TP +TN+FP+FN). Accuracy might be misleading because it is bias towards the dominant value, and the minor value has little to no impact on the accuracy cause by the low percentage of minority class in TP + TN.

We will be considering 3 evaluation metrics:

1. Weighted-Recall: Also known as True Positive Rate calculate how many of the actual positive value are predicted as positive. Recall is good at minimizing False Negative, the probability of falsely categorizing an actual positive case [5].
2. Weighted-Precision: Calculate how many of the predicted positive are correct. Precision is good at minimizing False Positive, the probability of falsely categorizing an actual Negative case [5].
3. Weighted-F1-score: The balance between precision and recall. F1-score is used when there is imbalance class where both precision and recall are required.

In our case, Weighted-F1-score will be the best since our target feature is highly imbalance.

2.6. Model Training Process

Categorizing IMDB score

Before any training, we need to categorize the imdb_score into 5 categories first. We can directly alter the numerical value by thresholding the value into the label encoded result. 0-4 for poor to excellent.

XGBoost

```
xgb = XGBClassifier(device = "cuda", n_estimators=200, learning_rate=0.01, objective='multi:softmax', num_class=5)
```

The parameter objective='multi:softmax' set XGBoost to do multiclass classification with num_class amount of classed. A low learning rate is used here to decrease the step of each iteration which makes the model less likely to overfit or underfit.

```
xgb.fit(X_train, y_train)
```

The training set is then fit to the model.

```
# Define the scoring metrics as a list
scoring = ['f1_weighted', 'recall_weighted', 'precision_weighted']

# Perform cross-validation and compute multiple metrics simultaneously
scores = cross_validate(xgb, X_train, y_train, cv=skf, scoring=scoring)
```

In the cross-validation process, the evaluation metrics will be evaluated using the StratifiedKFold we defined earlier.

```
Mean F1: 0.7152516213665987
Mean Recall: 0.7356366084011695
Mean Precision: 0.7103584711742381
CPU times: total: 24.3 s
Wall time: 13.3 s
```

Random Forest

```
rfc = RandomForestClassifier(random_state=1, n_jobs=-1, class_weight = 'balanced' )
```

The parameter class_weight = 'balanced' is used, it adjust the weight of each class according to the inversely proportional to class frequencies in the input data as n_samples / (n_classes * np.bincount(y)). This balance out the weight in an imbalance class and prevent the model from being bias toward the majority class.

The cross-validation process is similar with XGBoost.

```
Mean F1: 0.7174786382684398
Mean Recall: 0.7463755495256728
Mean Precision: 0.7215636652947718
CPU times: total: 2.84 s
Wall time: 3.52 s
```

Decision Tree

```
dtc = DecisionTreeClassifier(random_state=1,class_weight = 'balanced' )
```

Similar to random forest, class_weight = 'balanced' is used to balance out the weight of each class.

The cross-validation process is similar with XGBoost.

```
Mean F1: 0.6551087509832544
Mean Recall: 0.6540960434151575
Mean Precision: 0.6568569730558043
CPU times: total: 531 ms
Wall time: 524 ms
```

2.7. Hyperparameter tuning

Random Forest

```
param_grid = {
    'n_estimators': [100, 200, 400],
    # Number of trees in the forest. More trees can increase model performance,
    # but too many trees might lead to overfitting or increased computation time.

    'max_depth': [None, 5, 10, 20],
    # Maximum depth of each tree. Higher depths allow the model to capture more complex
    # relationships in the data but might lead to overfitting if too high.

    'min_samples_split': [2, 5, 10],
    # Minimum number of samples required to split an internal node.
    # Higher values might prevent overfitting by requiring more samples for a split.

    'min_samples_leaf': [1, 2, 4],
    # Minimum number of samples required to be at a leaf node.
    # Similar to min_samples_split, higher values can prevent overfitting.

    'max_features': ['auto', 'sqrt'],
    # The number of features to consider when looking for the best split.
    # 'auto' uses all features, 'sqrt' uses the square root of the total features.

    'bootstrap': [True, False],
    # Whether bootstrap samples are used when building trees.
    # False might lead to slightly more random trees but might reduce overfitting.
    'class_weight' : ['balanced']
}
```

```
{'bootstrap': False,
 'class_weight': 'balanced',
 'max_depth': 20,
 'max_features': 'sqrt',
 'min_samples_leaf': 1,
 'min_samples_split': 5,
 'n_estimators': 400}
```

XGBoost

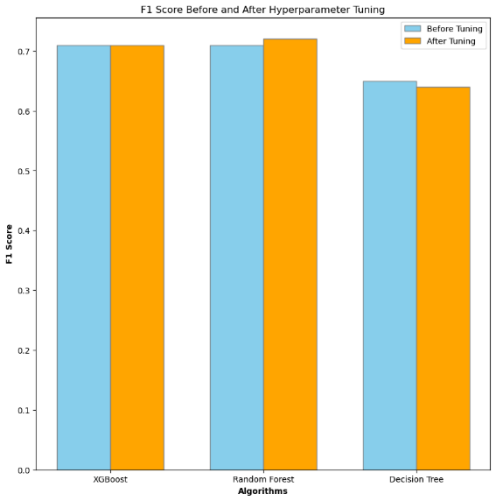
```
xgb_param_grid = {
    # How much the step size shrink in each iteration,smaller learning rate cost more computational power,but more robust
    'learning_rate' : [0.01,0.1],

    # The maximum depth that is too high can cause overfitting while too low cause underfitting
    'max_depth': [6,10,20],

    # L2 regularization term on weights. Increasing this value will make model more conservative.
    'reg_lambda': [0,100],

    # L1 regularization term on weights. Increasing this value will make model more conservative.
    'reg_alpha': [0,100],

    # Control the balance of positive and negative weights, useful for unbalanced classes.
    'scale_pos_weight': [1,3,5],

    #  is the subsample ratio of columns when constructing each tree. Subsampling occurs once for every tree constructed.
    'colsample_bytree': [0.5,0.75],

    # Subsample ratio of the training instances. Setting it to 0.5 means that XGBoost would randomly sample half of the
    # training data prior to growing trees. and this will prevent overfitting.
    # Subsampling will occur once in every boosting iteration.
    'subsample':[0.5,0.75]
}
```

```
{'colsample_bytree': 0.75,
 'learning_rate': 0.1,
 'max_depth': 10,
 'reg_alpha': 0,
 'reg_lambda': 0,
 'scale_pos_weight': 1,
 'subsample': 0.5}
```

3. Results and Interpretation:

Model Comparison and visualization

From the bar plot, we can see the comparison of weighted F1-score between the 3 chosen algorithm. XGBoost and Random Forest have similar weighted F1-score (0.71) before hyperparameter tuning. But after hyperparameter tuning, XGBoost's F1-score (0.72) overtakes Random Forest (0.71). This might be because XGBoost allow more customization and parameter tuning of the algorithm. Decision tree's Weighted F1-score is around 6% lower than the other (0.65). The weighted F1-score of decision tree decrease after hyperparameter tuning, which indicates overfitting.



Overall, the F1-score of all 3 model increases after hyperparameter tuning because of the optimized parameter. For instance, adjusting the max depth prevent overfitting, which improve the model's ability to generalize and predict unseen data.

Confusion Matrix Before and after Hyperparameter tuning.

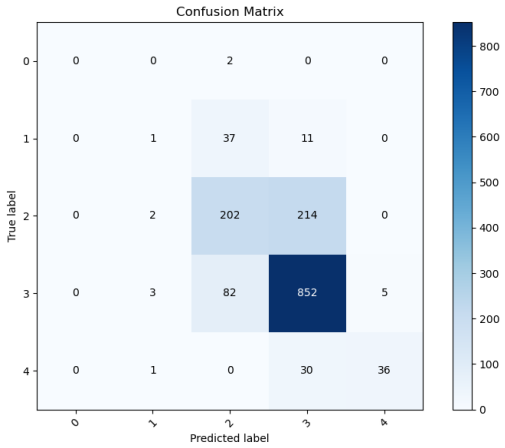


Figure 1: XGBoost Before hyperparameter tuning

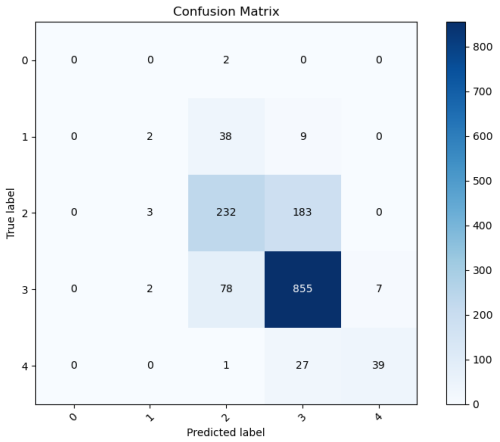


Figure 3: XGBoost After hyperparameter tuning.

From the confusion matrix above, we can see that the True Positive of each class has increase except for class 0 (Poor), which only has 2 test sample. The TP value of class 1 and 4 only increases by a few. This show us that it is hard for model to capture the pattern for extreme minority class because of the extremely small class size for testing and training. Hyperparameter tuning has limited effect on the minority class.

From here we can also see that the model is bias toward the majority class as majority of the False Negative values (wrongly classify as Negative) of each class are from class 2 (Average) and class 3 (Good) which are both majority classes.

Pros and limitation

XGBoost

Pros: Better performance compared to other 2 especially after hyperparameter tuning. Robust against overfitting, better handle of complex relationship between features

Cons: Computationally expensive and require careful tuning of hyperparameter.

Random Forest

Pros: Robust against overfitting, has good performance and computationally less expensive compared to XGBoost

Cons:

Decision Tree

Pros: Simple and fast run time but with terrible scoring in weighted F1-score.

Cons: Prone to overfitting, does not handle imbalance class as well as other 2 ensemble algorithm.

4. Summary

Throughout this entire project, we have begun with data pre-processing, which includes steps such as handling missing values, feature encoding, identifying, and handling outliers and feature selection. These steps improve the overall data robustness and reliability. After that, we have selected 3 models for this classification task which are Decision Tree, Random Forest and XGBoost. Various considerations are made before choosing these models such as presence of outliers and the type of task needed to be done. In the training process, we split the data into 70/30 split to increase data robustness and uses StratifiedKFold cross validation technique to retain the sample of minority class in each iteration of cross validation. Results are compared and analysed using bar plot and confusion matrix to compare between algorithm before and after hyperparameter tuning. Finally, this summary concludes by identifying XGBoost as the best model for predicting IMDB score as it has the highest F1-score compare to the other models.

Link to notebook:
https://colab.research.google.com/drive/1gHbqtmdIQnagdkLz7QXxciqWs33kTkW8?usp=sharing

5. Bibliography

[1] "Cramer's V - StatsTest.com," *StatsTest.com*, Apr. 07, 2020. https://www.statstest.com/cramers-v-2/

[2] H. Akoğlu, "User's guide to correlation coefficients," *Turkish Journal of Emergency Medicine*, vol. 18, no. 3, pp. 91–93, Sep. 2018, doi: 10.1016/j.tjem.2018.08.001.

[3] J. Frost, "Guidelines for removing and handling outliers in data," *Statistics by Jim*, Apr. 05, 2021. https://statisticsbyjim.com/basics/remove-outliers/

[4] "Relative information loss in the PCA," *IEEE Conference Publication | IEEE Xplore*, Sep. 01, 2012. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6404738

[5] T. Kanstrén, "A look at precision, recall, and F1-Score - towards data science," *Medium*, Sep. 27, 2023. [Online]. Available: https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec