2020 Rotten Tomatoes Comments Sentiment Classification

Ruochong Fan
rfan35@wisc.edu

Yuzhao Jin
yjin89@wisc.edu

Abstract

The objective of the project is to compare and predict Rotten Tomatoes' comment reviews on movies posted in the year of 2020 using various machine learning methodologies and classifiers. The dataset contains 45746 reviews with sentiment scores of 0 or 1 based on Tomatometer's "Rotten" and "Fresh" indicator. After pre-processing using NLTK package, the dataset generates a (45746×32922) *matrix and is randomly separated into 80 percent for training data and 20 percent for testing data. Classifiers of KNN, Decision Tree, Random Forest, XGBoost, Logistic Regression, Bagging and Stacking Cross Validation are trained based on the previous separation benchmark to compare their outputs of testing accuracies and their credible intervals. Visualization on comparing the training and testing accuracy at different feature points is produced for all classifiers. Out-of-Bag and .632 bootstraps are also used to adjust the testing accuracies from bias. Further predictions on the performance of classifiers are also conducted using confusion matrix. The result of this project shows that the Random Forest classifier has the best .632 bootstrap testing accuracy and the performance on true prediction percentage while the Logistic Regression classifier has the best normal testing accuracy.*

1. Introduction

The rapid development in computer technologies and the internet enables people to generate a countless number of texts every single day. The abundant pop-up messages on smartphones and endless comments on social platforms are representations of the advancement. People can instantly write down their thoughts and comments on any hot-spot news or social attention; the comments, on the other hand, can serve as valuable data for researchers to study the attitudes and sentiments that fluctuate among the public.

On studying the sentimental influences upon movie industries, the project chooses to use comments in Rotten Tomatoes to predict the general sentiment level in the year 2020 through multiple machine learning classification methods. Rotten Tomatoes (`https://www.rottentomatoes.com/`) is one of the most influential social platforms for evaluating the qualities and productions of films[1]. It consists of various comments from journal editors to general users. Through identifying the frequencies of directional words within these reviews, the project wants to determine and validate whether the sentiment of the reviews on a particular movie is Fresh or Rotten (Positive or Negative).

2. Motivation

Movies have always been popular since their appearance in the late 19^{th} century, and they will continue to be for years to come. The movie industry generates billions of dollars each year, whereas North America had 11.4 billion dollars in sales in 2019 [2]. Considering the size and impact the movie industry has, the goal of this project is to create a fast and efficient way to tell whether a review is considered "fresh" or "rotten" in Rotten Tomato's comments.

The movie industry was hit hard during the 2020 global Covid-19 pandemic with sales only reaching about 2.2 billion dollars in North America and Canada [3]. As seen in figure 1, this is a drastic decline in revenue from the previous year. Through the model, the team to discover the sentiment that people felt about movies during 2020. By identifying keywords associated with positive feedback, the project can understand the sentiment changes among audiences and possibly acknowledge reasons for the ratings of a movie other than physical interference.

3. Literature Review

When training natural language processing models, including particular words may be one of the most common puzzles. However, experimenting on texts exclusion has been identified. Na, Thet, Nasution, and Hassan believe that using feature reduction based on customized stop words in movie reviews and features based on rebuttal and stemmed words can significantly improve the accuracy of sentiment analysis[4]. While acknowledging the importance of counting the occurrence of words in natural language processing, the paper points out that only recording the frequency of adjectives may not produce the best accuracy.

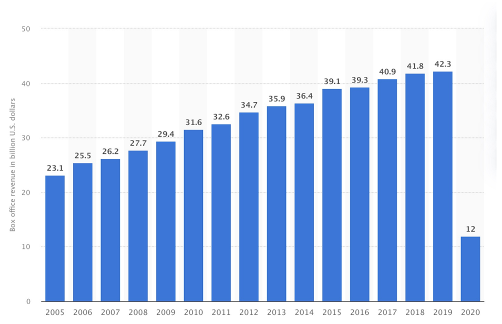


Figure 1. Box office revenue from 2005 to 2020. Image source: `https://www.statista.com/statistics/271856/global-box-office-revenue/`.

On machine learning algorithms and classification processes, Khan, Gul, Uddin, Shah, Ahmad, Firdausi, and Zaindi demonstrate the benefits of using basic Bayesian classifier. The basic Bayesian classifier – used in determining the negativity or positivity of movie comments – can be used on features with increased frequencies weighted by IDF (Inverse Document Frequency) and present better performance and better sentiment accuracy[5].

The cited works are good examples of determining both classification and model selection processes for movie reviews. The literature also presents their unique methodologies on choosing specific words as features. The project has utilized some key concepts from the listed literature and has provided corresponding analyses.

4. Software and Python Packages

The main coding language for this project is python and the computer hardware is each team member's laptop. The main operational platform for this project is Jupyter Lab and Jupyter Notebook. These items will serve as the foundation of project initiation.

The Python packages for this project are Pandas, NumPy, Scikit-Learn, MLXtend, Matplotlib, and Natural Language Tool Kit (NLTK). Pandas and NumPy are two fundamental packages that are widely used in multi-purpose Python projects. The two have saved much computational time for processing data and data cleaning. Scikit-Learn is the main package for using machine learning classifiers, and it serves as the fundamental procedure in model selections using the grid search cross-validation method. MLXtend is suitable for model evaluation through using Out-of-Bag and .632 bootstraps. Due to hardware memory insufficiency, the .632+ bootstrap method is dropped. Both Out-of-Bag and .632 bootstrap methods are used for updating testing accuracy and avoiding bias influence in classifiers. Matplotlib package serves as the visualization of model selections and accuracy results in testing and predictions. The NLTK is considered the core package in data cleaning for it can clean the review texts and tokenize each word to be prepared for training and testings [6]. Details towards the usage of these packages will be in the "Experiments" section.

5. Proposed Method

5.1. K Nearest Neighbors

Among all machine learning algorithms, K Nearest Neighbors (KNN) has the characteristic of being computationally efficient. In this algorithm, the project first finds the nearest neighbors in the training dataset and then makes a corresponding prediction based on k. The mathematical computation formula is expressed by

$$h(x^{[t]}) = \frac{1}{k}\sum_{i=1}^{k} f(x^{[i]})$$

where the input consists of the k-closest training set and the output is the average of k nearest neighbors for the object.

The project chooses KNN for its computational efficiency and its directness for tuning the hyperparameter k. Here KNN is performed not only as of the first attempt but also as the baseline for accuracy scores. For calculating the distance in KNN, the project favors Euclidean distance for brevity.

5.2. Decision Tree and Random Forest

The Decision Tree and Random Forest algorithm can be used for both classification and regression. The project grows the tree as a regression algorithms mainly by tuning the hyperparameters "criterion" and "max_depth", where criterion is expressed as entropy: $H(p) = \sum_i p_i \log_2(\frac{1}{p_i})$ represents the information gain and gini = $1 - \sum_i (p(c = i)^2)$ is the gini impurity. The hyperparameter "max_depth" is a numeric value (None for default) that bounds the expansion of nodes.

Here the team chooses both classifiers to see how the data perform with a binary response. While performing model selections, the team chooses parameters from number of estimators, entropy and gini, and numeric values of maximum depth to find the best accuracy score. Fitting on bootstrap samples is also performed on these two methods, and that random selection on feature subsets is also considered on each number of splits. Three hyperparameters is also used for training and testing: Number of estimator, maximal depth and information criterion.

5.3. XGBoost

Among the two broad categories of boosting, the project chooses to perform Gradient Boost classifier(specifically XGBoost) for our dataset. Here Gradient Boost classifier takes multiple weak learners and combines them into a stronger learner without adjusting the weight of the training set. Instead of changing the dataset to achieve a balanced response, here the team optimize the differentiable loss function. The team regulates the term ”learning rate” to tune the training process.

5.4. Logistic Regression

Since the response variable is binary, the Logistic Regression Model is preferred. It is a soft classifier that predicts the conditional probability of a binary label. In the Logistic Regression Classifier, the project predicts the label (y variable) using the formula

$$\text{logit}(\frac{P(y_i = 1)}{P(y_i = -1)}) = (\beta \times x_i)^T$$

where x is the feature vector and β is the coefficient vector of the feature. The probability of success (P1= 1) is defined by $\frac{e^{\beta x_i}}{1+e^{\beta x_i}}$. To tune the model and avoid potential overfit, the project includes the normalization constant ”C” representing the learning rate. By changing the values of ”C”, the project can regulate and optimize the training and fitting process.

5.5. Bagging

The Bagging Classifier relies on majority vote and can use other base estimator in model fitting. The project applies Bagging classifier with Decision Tree classifier and Logistic Regression classifier. This is because Decision Tree classifier is one of the most common learning algorithms used in Bagging and Logistic Regression seems to fit our data very well due to the binary response.

5.6. Stacking Cross Validation

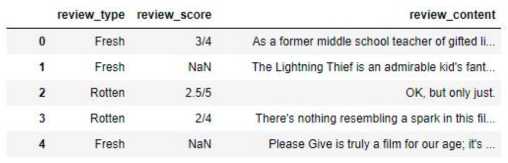
For Stacking Cross Validation (Stacking CV) classifier, the project chooses the classifiers with relatively high accuracy scores. Here the project chooses Random Forest as classifier and Logistic Regression as meta classifier. Here the project chooses the base learner Random Forest classifier and let it make predictions that serve as input features to the meta-classifier Logistic Regression classifier. The team is acknowledged that Stacking CV has a high tendency of suffering from extensive over-fitting; also for the sake of run-time, the team chooses not to tune hyperparameters in stacking.

6. Experiments

6.1. Dataset

The dataset is originally scraped from the Rotten Tomatoes database and posted by Stephano Leone in Kaggle with a pre-processed version [7]. The project uses the critic review file from Kaggle for sentiment analyses.

The Rotten Tomatoes critic reviews dataset has 8 columns and over one million rows of review types and review contents dated back to the website’s initiation in 1998. Because the critic review file is too large, this project extracts the latest year (2020) data from January to October (the latest update of this dataset) and trains machine learning models based on this data. After extraction, there are 45746 rows in the 2020 dataset. Since the sentiment analysis focuses on the review contents and their binary ratings, the project utilizes three columns: review type, review score, and review content for analyses (Figure 2). However, due to inconsistent formatting issues in column ”review score” and little relevance to the training and testing process later in this project, the column is eventually dropped.



	review_type	review_score	review_content
0	Fresh	3/4	As a former middle school teacher of gifted li...
1	Fresh	NaN	The Lightning Thief is an admirable kid's fant...
2	Rotten	2.5/5	OK, but only just.
3	Rotten	2/4	There's nothing resembling a spark in this fil...
4	Fresh	NaN	Please Give is truly a film for our age; it's ...

Figure 2. View of Selected Columns

Preliminary computation on this dataset shows that the ratio between ”Fresh” and ”Rotten is roughly about 71.4 percent versus 28.6 percent (Figure 3). This result indicates that our dataset is not balanced and may require balanced tuning when fitting the training and testing classifiers.

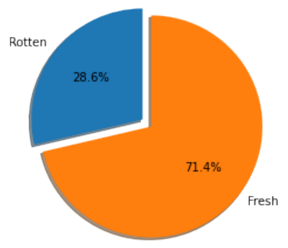


Figure 3. The distribution of the response variable

This project assumes that all samples in the 2020 dataset are proper representations of the population, which are rotten tomatoes’ critic reviews of all times. Since the global COVID-19 pandemic started from March 2020 has severely influenced the movie industries, this project thinks that online movie review websites such as Rotten Tomatoes may also reflect some of those influences and may demonstrate significant shifts in sentiment and review emotions. Thus, in this case, this project chooses to accept such possible shifts for the sake of coherence and assume that reviews emotions stay the same throughout 2020.

6.1.1 Removal of Punctuation Signals and Numbers

Since the Kaggle version of the Rotten Tomatoes data have already excluded HTML tags and non-word characters in the data, it provides a quicker path for this project in preprocessing the data. In order to avoid irrelevant word characters, this project decides to remove all punctuation signals and numbers in the data through the usage of ”str.replace” function so that the testing and training process does not contain them.

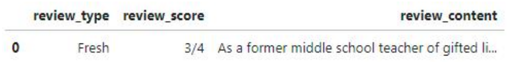
6.1.2 Adjustment of Characters to Lower Cases

To avoid words with same meaning to be counted separately, it seems essential to transform all words into same format. Since same words may have different cases due to their positions in the sentence, this project decides to use ”str.lower” function to transform all words into lower case.

6.1.3 Removal of Stopwords and Tokenization in Comments

Words like ”a”, ”an” and ”the” are generally considered as stopwords and they do not serve any meaningful reference to the sentiment of a comment. Although there is no unilateral definition on the categorization of stopwords, ”NLTK” and ”Scikit-Learn” packages still provide methodologies in using their attached dictionaries to rule out stopwords in sentences. Also, tokenization of the words is also essential in preparing dataset to be ready for lemmatization and vectorization and it is found that these steps can be finished in the same function. Output on this transformation is shown in the following figures (Figure 4 and 5).

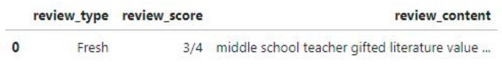
Before:



	review_type	review_score	review_content
0	Fresh	3/4	As a former middle school teacher of gifted li...

Figure 4. View of Prior Stopwords Removal and Tokenization

After:



	review_type	review_score	review_content
0	Fresh	3/4	middle school teacher gifted literature value ...

Figure 5. View of Post Stopwords Removal and Tokenization

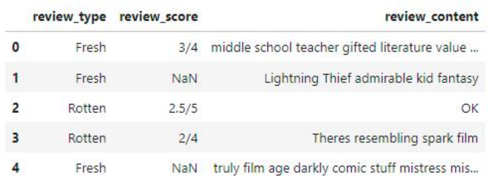
6.1.4 Lemmatization of Comments

After tokenization of the ”review content” section of the dataset, it can proceed to unify the words under the same meaning. Two common ways of achieving this goal are lemmatization and Stemming. According to Manning, Raghaven, and Schütze, the goal of both stemming and lemmatization is to ”reduce inflectionally and sometimes derivative forms” of words to a common base form [8]. However, the stemming method chooses to ”chop off” the extra ending part of a word and only remain its base while the lemmatization method prefers to categorize based on ”vocabulary and morphological analysis of words”[8]. Since the meaning of a word is more important than the base of a word in analyzing the sentiment of the text, this project chooses lemmatization over stemming and utilizing relevant functions from ”NLTK” packages so that there will not be excessive deletions over words that contain different meanings.

6.1.5 Quantification of the Texts Column

In order to fit the cleaned dataset in to the model, simply using the categorized labels in the ”review type” column is insufficient to demonstrate differences on sentiment. This project chooses to transform ”fresh” and ”rotten” labels in the ”review type” to quantitative difference of ”1” and ”-1” so that visualization can be demonstrated and each comment will serve as a significant input when training and testing different models (Figure 6 and Figure 7).

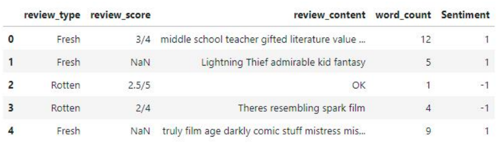
Before:



	review_type	review_score	review_content
0	Fresh	3/4	middle school teacher gifted literature value ...
1	Fresh	NaN	Lightning Thief admirable kid fantasy
2	Rotten	2.5/5	OK
3	Rotten	2/4	Theres resembling spark film
4	Fresh	NaN	truly film age darkly comic stuff mistress mis...

Figure 6. View of Prior Review Type Quantification

After:



	review_type	review_score	review_content	word_count	Sentiment
0	Fresh	3/4	middle school teacher gifted literature value ...	12	1
1	Fresh	NaN	Lightning Thief admirable kid fantasy	5	1
2	Rotten	2.5/5	OK	1	-1
3	Rotten	2/4	Theres resembling spark film	4	-1
4	Fresh	NaN	truly film age darkly comic stuff mistress mis...	9	1

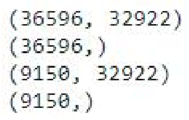
Figure 7. View of Post Review Type Quantification

6.1.6 Vectorization of Lemmatized Comments

To measure the association between movie reviews shown in ”review content” column and the corresponding quantified sentiment levels in ”sentiment” column, this project decide to count the words’ occurrences in each review contents and vectorize the counts. By doing this, the dataset obtains a feature of size (45746×32922) and a label of size of (45746×1). After vectorizing the feature of the data, benchmark of splitting the data into training and testing purposes can be made to prepare for fitting different classifiers.

6.2. Model Training

After cleaning and vectorizing the dataset, it is time to work towards the direction of training the data and computing the accuracy. This project decides that the benchmark for splitting the cleaned dataset is 80 percent to 20 percent, meaning that 80 percent of the dataset will be used on training purpose and 20 percent of the dataset will be used on testing purpose. After python randomized the dataset and splitting it based on the previous mentioned benchmark, four matrices are generated to show the span of matrices used in training and testing section. The order of the resulting four matrices in the following graph is: ”X training matrix”, ”y training matrix”, ”X testing matrix”, ”y testing matrix” (Figure 8).



```
(36596, 32922)
(36596,)
(9150, 32922)
(9150,)
```

Figure 8. View of Dimensions of Matrices

Classifiers from ”Proposed Method” section will be used to train and test data based on the benchmark percentage set in the previous paragraph. These classifiers will also be tuned by changing values on relevant parameters. To avoid randomized results while operating these classifiers, classifiers with available ”random state” parameter are all set to 1. Details to the usage of each classifier are shown below.

6.2.1 K Nearest Neighbors

To tune the parameter k for KNN classifier, this project chooses ”KNeighborsClassifier” and ”GridSearchCV” from Sci-kit Learn package to compare different testing accuracy generated by different values of parameter k. Since the optimal numeric range for parameter k is the odd integer between 3 and 15, the cross validation grid search will run [3,5,7,9,11,13,15] for parameter k in KNN classifier and generate relevant testing accuracy. Process to find optimal parameter k is shown in the following graph (Figure 9)

6.2.2 Decision Tree and Random Forest

The project decides to tune the Decision Tree and Random Forest classifiers based on hyperparameters maximal depth and information criterion. The grid search cross validation on the hyperparameters of the tree will also be utilized for

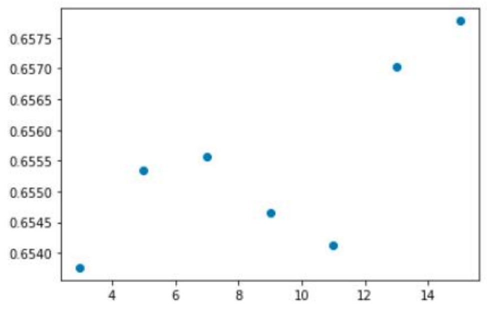


Figure 9. View of Obtaining Optimal Parameter k

tuning. In terms of maximal depth, Decision Tree classifier chooses $[1, 2, 3, 4, 5, 6, 10, \text{None}]$ while Random Forest classifier chooses $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, \text{None}]$, where ’None’ infers an unlimited upper bound in which the nodes are expanded until all leaves are pure. As for information criterion, both ”entropy” and ”gini” are considered.

6.2.3 XGBoost

The team uses the hyperparameter optimization method to tune the XGBoost classification. The team tunes alpha and lambda both ranging from 1e-8 to 1 and tune number of estimators from the list 10, 30, 50, 100. The learning rate is tuned by a factor of 0.01 to control overfitting. Since the data is a regression with vectorized features and labels, the XGBoost classifier is not expected to return very high accuracy score. However, the team still chooses to perform the classifier to see how well XGBoost performs with natural language processing problems.

6.2.4 Logistic Regression

On the subject of tuning Logistic Regression classifier (”LogisticRegression” in Sci-kit Learn Package), ”GridSearchCV” will also be used in determining the most optimal regularization parameter C. The range of C that will be applied in ”GridSearchCV” is from 0.0001 to 10 with step factor of 10. Process to find the optimal regularization parameter C is shown in the following graph (Figure 10).

6.2.5 Bagging

The project separately performs Bagging Decision Tree and Logistic Regression. The project chooses decision tree with the best hyperparameter tuned from grid search and apply the tree in bagging. In the end, the accuracy has increased about 5% compared to the decision tree classification. Although logistic regression returns a relatively high accuracy score, applying it with bagging does not seem to make significant improvement.

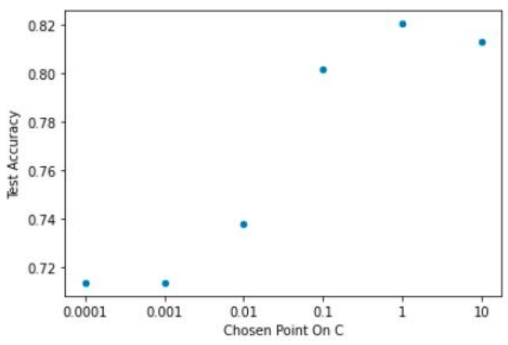


Figure 10. View of Obtaining Optimal Regularization Parameter C

6.2.6 Stacking Cross Validation

Although currently it is still uncertain about the decision on choosing which previously mentioned classifiers for Stacking CV classifier, considering the quantity and time length on Python computation, this project proposes that classifiers with the top highest and second highest test accuracy will be used in Stacking CV classifier. Relevant parameters that assist the output of the top two testing accuracy scores will be directly used in the Stacking CV classifier.

7. Results

As the team starts to analyze the results generated by the first six classifiers, the team starts to notice that that classifiers of Logistic Regression and Random Forest performs are outstanding in terms of the normal and bootstrap testing accuracies. Further testing on Stacking Cross Validation classifier is also performed based on these two classifiers and results are similar to the previous two classifiers. Throughout the project, the team has computed model selections upon all seven classifiers, where each classifier's most optimal parameters and available hyper-parameters are generated (Table 1).

Model	Best Hyperparameter
KNN	k=15
Decision Tree	criterion='entropy', max_depht=None
Random Forest	criterion='entropy', n_estimator=150, class_weight='balanced'
XGBoost	alpha=1.328, lambda=1.015, learning rate=0.01
Logistic Regression	C=1
Stacking CV	classifier=RF, meta_classifier=LR, cv=10

Table 1. Model and corresponding best hyperparameters

Model	Test accuracy	OOB accuracy	.632 accuracy
KNN	65.56	64.13	66.37
DecisionTree	75.672	73.636	79.763
RandomForest	81.95	79.83	84.52
XGBoost	71.038		
Logistic Regression	82.02	80.02	84.05
Bagging with Decision Tree	80.448		
Bagging with Logistic Regression	82.000	80.826	83.699
Stacking CV	82.055		

Table 2. Classifiers with test accuracy and bootstrap accuracy in percent

Accuracy scores computed by each algorithm are stored in the following table (Table 2). The test accuracy scores of all classifiers are computed from the model fitting process; the OOB and .632 bootstrap methods generate accuracy scores based on pessimistic and optimistic bias. As for XGBoost, the team chooses not to compute its bootstrap method since it provides a relatively low accuracy score but takes a lot of time to run. For Bagging with Decision Tree and Stacking CV, the OOB and .632 testing accuracy are not computed due to the circumstance that running the bootstrap methods on Bagging seems to be too time-consuming and Stacking Cross-Validation seems not available for bootstraps.

Further enhancement on the range of accuracy scores is generated using normal approximation for test accuracy scores and bootstrap methods in order to measure pessimistic and optimistic bias for each classifier. As expected, the OOB bootstrap methods generally show relatively lower intervals than the normal approximated test accuracy intervals while the .632 bootstrap shows higher ones than the normal intervals due to the existence of pessimistic and optimistic biases. The detailed percentages rounded to 3 decimal places are shown in the following Table (Table 3).

In order to demonstrate the performances of all classifiers for prediction purposes, the team chooses to generate the confusion matrix for each algorithm. The confusion matrix is composed of four cells: True positive, true negative, false positive, and false negative. To test their true valid prediction results, numbers of true positive and true negative are added together to be divided by the number of test data which is 9150. Taking the confusion matrix of Random Forest classifier as an example (Figure 11), its valid

Model	test accuracy	OOB	.632
KNN	(64.589, 66.536)	(58, 69)	(60, 71)
DecisionTree	(74.793, 76.551)	(72.538, 74.706)	(78.943, 80.598)
RandomForest	(81.191, 82.766)	(79, 81)	(84, 85)
XGBoost	(71.875, 73.699)		
Logistic Regression	(81.235, 82.809)	(80, 81)	(84, 84)
Bagging with Decision Tree	(79.635, 81.261)		
Bagging with Logistic Regression	(81.213, 82.787)	(80.235, 81.371)	(83.242, 84.075)
Stacking CV	(81.268, 82.841)		

Table 3. Classifiers with test accuracy confidence intervals and Bootstrap confidence intervals in percent

Model	True Prediction(%)
KNN	74.30
Decision Tree	75.67
Random Forest	81.05
XGBoost	72.79
Bagging with Decision Tree	80.45
Bagging with Logistic Regression	79.89
Logistic Regression	76.28
Stacking CV	82.05

Table 4. True prediction percent for used algorithms

true prediction percentage is $\frac{5805+1611}{9150} = 81.05\%$. Similar methods are applied to other confusion matrices. Relevant true prediction percentage for each classifier are listed in the above table (Table 4).

The team also generates visualized comparisons between training accuracy and normal approximated testing accuracy among all classifiers. Take the Random Forest model as an example (Figure 12). Training accuracy seems to be constant at all times. However, normal testing accuracy seems to be improving as more features from training sets start to compute under the classifier and try to fit in the testing data. It shows that the more the features, the better the models can be trained as well as the testing accuracies. Similar phenomenons are found across all classifiers, which can be said that the dataset is generally computational friendly to all classifiers.

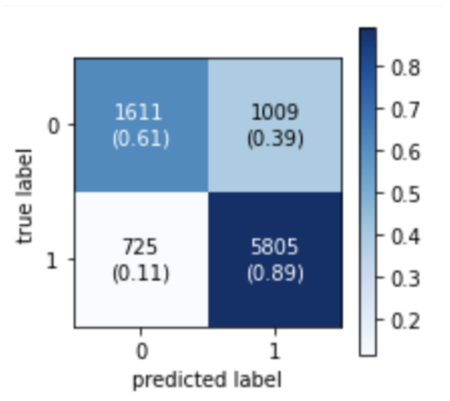


Figure 11. Confusion matrix for random forest classifier

Based on previous discussions, it seems that the Stacking CV classifier has the best outcomes in normal test accuracies and true prediction percentage. However, in light of the overfitting problem in the Stacking CV classifier and its inability in doing bootstraps, the team decides to drop since it is uncertain whether the result generated by the Stacking CV classifier suffers any biases given the set classifier and meta classifiers. Thus, This team stands that the Random Forest classifier can produce the best .632 bootstrap test accuracy and true prediction percentage while the Logistic Regression classifier can generate the best normal test accuracy.

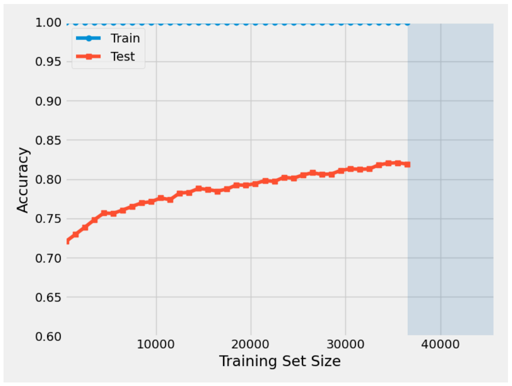


Figure 12. Accuracy scores with changing train set size on Random Forest Classifier

8. Discussions and Conclusions

Throughout the project, the team has cleaned the movie review texts, tokenized each word, and trained multiple models to fit the data. Among all classifiers, the Random

Forest classifier has the highest testing accuracy of 84.52 % after .632 bootstrap and the best true prediction percentage of 81.05 % with "entropy" as criterion and 150 as n_estimator while using "balanced" class weight. The Logistic Regression is also able to achieve the highest normal testing accuracy of 82.02 with parameter C equal to 1. In that sense, Random Forest classifier may better fit and predict the dataset as compared to the Logistic Regression classifier. Still, the Logistic Regression classifier can generate better results if further tuning mechanisms can be used on it.

Although this project can generate all relevant results given sufficient time and hardware, there is still room for further improvement. While the team is programming and conducting computations, the run-time issue is very significant, especially on computing relevant results for the Random Forest classifier. It may be due to the problems from hardware, oversized dataset, and repeated iterations. Also, the pre-processed dataset file seems to have certain rows of data updated, which may lead to missing values when conducting extractions. What's more, this project is unable to compute the .632+ bootstrap due to memory limitation on the hardwares. Since .632+ bootstrap performs better than OOB and .632 because of its immunity from both pessimistic and optimistic biases, results from .632+ may be different but they are more accurate and objective for inter-comparison. If the above problems can be further confirmed and fixed for future studies, then the results on testing accuracies and true prediction percentage can have better improvement.

9. Acknowledgements

This project team appreciates and thanks Rotten Tomatoes and user Stephano Leone from Kaggle for providing information on review comments and the organized dataset.

10. Contributions

The entire project is divided into four parts: Finding and preparing data, model fittings and programmings, writing project report, and preparing for final presentation. For finding and cleaning data, Ruochong finds the dataset online from Kaggle and performs vectorization of the cleaned data; Yuzhao performs the initial data cleaning. On model fittings and programmings, Ruochong is responsible for fitting the data to Decision Tree, XGBoost and Bagging (Decicion Tree and Logistic Regression) classifiers as well as all their relevant outputs on accuracies and predictions; Yuzhao is responsible for KNN, Random Forest, Logistic Regression and Stacking Cross Validation classifiers as well as their relevant outputs on accuracies and predictions. In terms of writing the project report, Ruochng writes sections of Literature Review, Software and Python Packages, and Proposed Methods; Yuzhao writes sections of Introduction, Motivation, Conclusion and the subsection of Dataset in the Experiment section. The subsection of Model Training in the Experiment section are written based on the distribution of work in model fittings and programmings. Sections of Result, Abstract and Acknowledgement of the report are written and conducted by both members. As for the final presentation, PowerPoint slides are created based on the distribution of work in the project report and both members are going to present together in front of the class.

11. Codes

All codes and plots are in this link (`https://github.com/SamuelFan1215/Stat451-Group23`)

References

[1] Rotten Tomatoes. About Rotten Tomatoes. *Fandango*, 2020. https://www.rottentomatoes.com/about

[2] M. Pamela. 2019 Global Box Office Revenue Hit Record $42.5B Despite 4 Percent Dip in U.S.*Billboard*, 2020. https://www.billboard.com/articles/news/8547827/2019-global-box-office-revenue-hit-record-425b-despite-4-percent-dip-in-us/.

[3] J. Navbarro. Global box office revenue from 2005 to 2020. *Statista*, 2021. https://www.statista.com/statistics/271856/global-box-office-revenue/

[4] J. Na, T. T. Thet, A. H. Nasution, and F. M. Hassan. A Sentiment-Based Digital Library of Movie Review Documents Using Fedora. *Canadian Journal of Information Library Sciences*, 35(3): 307–337, 2020. doi:10.1353/ils.2011.0018

[5] A. Khan, M. A. Gul, M. I. Uddin, S. A. Ali Shah, S. Ahmad, N. D. Al Firdausi and M. Zaindin. Summarizing Online Movie Reviews: A Machine Learning Approach to Big Data Analytics. *Scientific Programming*, 2020: 1–14, 2020. https://doi.org/10.1155/2020/5812715

[6] S. Bird, E. Klein, and E. Loper. Natural Language Processing with Python. *O'Reilly Media Inc*, 2009. https://www.nltk.org/book

[7] S. Leone, Rotten Tomatoes movies and critic reviews dataset. *Kaggle*, 2020. https://www.kaggle.com/stefanoleone992/rotten-tomatoes-movies-and-critic-reviews-dataset?select=rotten_tomatoes_critic_reviews.csv/.

[8] C. Manning, P. Raghavan, and H. Schütze. Introduction to Information Retrieval. *Cambridge University Press*, 2008. http://nlp.stanford.edu/IR-book/information-retrieval-book.html