Movie Success Prediction

Yonatan Cipriani & Hamza Khanani

December 11, 2018

Abstract

The goal of the project was to accurately categorize a movie into three categories: “flop”, “hit”, or “blockbuster”. To implement this project we explored several methods that could help us predict the success of a movie with the highest accuracy. We split this project into two tasks. First, we analyzed an existing dataset for the movies to come up with any trends and patterns that could help us with the prediction. This included various techniques, such as preprocessing the data and feature reduction which improved our accuracy immensely. Feature reduction was key in overcoming the curse of dimensionality because there were over 25 features for each movie in our dataset. Once we were confident with the prediction algorithm that we implemented using a decision tree, we split up the dataset into train and test sets. After many attempts and low accuracy scores with various models, we were able to obtain an accuracy score of 84.6% and a precision score of 94% after the implementation of a decision tree model and with the use of Gini Index as the impurity measure. With this major improvement, we were able to accurately predict the category under which a movie will fall and thus achieved our goal. This prediction is of great significance to movie stakeholders and producers and with this, they will be able to take the necessary steps to make a movie more profitable or axe the movie if needed. Also, the movie industry can use our implementation to modify the movie criteria for obtaining likelihood of blockbusters. Movie audience can also use our model to know whether it is going to be worth their time and money to watch a certain movie.

Intro + Problem Statement

For this project, we analyzed movies from the past and identified the factors that established those movies as flops, hits, or blockbusters. Millions of dollars are continuously poured into Hollywood and the movie industry, therefore it is important to identify what makes these movies flourish at the box office. Successfully predicting the fortunes of an impending movie release would be beneficial to everyone who was involved in the movie creation process. If a movie is considered a flop, that means that is not profitable during its time in the theaters. Once a movie is predicted to be unprofitable, there are several options that may be taken. This would either lead to shifting strategies and creating marketing decisions or the movie may just be axed. It is important for movie executives to deliver movies that can maximize their profits at the box office. IMDB (Internet Movie Database) is an online database of information related to movies, television, and video games. As of 2018, the website had over 83 million registered users. This provided us with a vast amount of information that helped us in making our prediction and we can feel confident that the data is reliable. Movie stakeholders need to find a way to utilize all of the information that this website has to offer. By having growing, historical data available machine learning techniques can be applied in order for organizations to have a better chance of identifying what is profitable and what is a risk. The most important feature that we based off our project was the IMDB score. Our project can also be very useful for the viewers who are not willing to waste their money and time at the theatre as the algorithm we implemented will give them a rough idea of what to expect from the movie.

Literature Review:

There are many machine learning algorithms that have been implemented for the prediction of a movie's success. We got a chance to go through some literature and it looked like genre was one of the biggest factors of a movie being considered a flop, hit or a blockbuster. Another factor which affected the rating was the main actors. Most of the well-known actors have a reputation because of their past works, so viewers tended to give a good rating to movies which included high profile actors (Meenakshi et al. 1). The literature we read has also helped us consider the weight we gave to some specific features.

We found a similar project on IEEE which was very interesting (Ahmad et al. 1). In their project, the criteria they used to predict the success of a movie was very similar to how we made our predictions. Some of the attributes which we also took into account were budget and director. They also realized that you can't make a precise prediction only from one attribute/feature. As a result, they built a model based on the relation between features which we also did just in a different way. We implemented this differently since our model found stronger correlations between some of our other features and our imdb_score feature. Something that they did differently which we also considered trying to incorporate was to associate weights to the features (Ahmad et al. 1). For example, the budget was given a lower weight if the movie's budget was below 5 million dollars. We tried to implement something similar to this, but we were met with poor results so we scratched that idea.
Each project implementation that we researched chose specific features and defines the "success" a little differently. We analyzed some of the implementations results with ours and drew conclusions on the similarities and differences. Also, we looked at the other methods that were used for the prediction and tried applying this to our project implementation but were not met with good results. The literature present also gave us a high-level overview of the techniques used in the past and how successful those models were. By researching the different methods and techniques other people used we were able to explore many different ways of analyzing the data and making our prediction.

<u>Methods + Techniques</u>

This was a very challenging project but was very interesting since we had to figure out a strategy that would help us learn the data from our dataset. We used data mining techniques in order to extract patterns that would be beneficial to us predicting the success of a movie. We explored many different ways of analyzing the data and ways to evaluate what we learned. Before we could apply different data mining strategies, we needed to clean and preprocess the data. Feature selection is something we dealt with early on in the project in order to overcome the curse of dimensionality problem. The dataset which we got from data.world, was a very large dataset with twenty-eight columns ("IMDB 5000 Movie Dataset"). As the result, we had a lot of information about each of the movies in the dataset. There is a total of 5043 movies in our dataset, so we had a large number of samples that we were able to draw conclusions from. We read our data by importing pandas so that we could have our data in the form of
a dataframe (Bronshtein). Our initial approach was to create our own decision tree model. In order to do this, we pre-pruned some of the features to ensure that we were not overfitting the dataset. We chose imbd_score as the root feature because was the most impactful for the movie to be considered a hit, flop or blockbuster. We reduced the number of features to 9 by deleting all the attributes we rationally thought wouldn't be as important as the others. Some of the features we deleted were color, movie_imdb_link, aspect ratio, language, actor names, country it was released in, and plot keywords. Our reasoning for deleting these specific features was because of

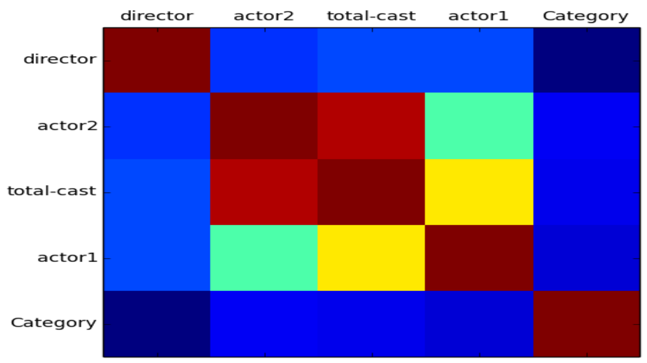
similar values throughout many of the different movies. For example, 95% of the movies in our dataset were in color, therefore the color feature was insignificant. Also, the language feature seemed redundant to include since English was the value in more than 93% of our dataset. We calculated these percentages by counting the number of times each value showed up in each of the feature columns.

Next, we performed data slicing which split out dataset into test and train sets using the sklearn.cross validation library ("Cross Validation"). Originally, we used 70% of the dataset for the train set and 30% for the test set. The test set only included the IMDB score column because this was the main feature that we wanted to test. The train set included all of the other columns which were used as our predictor values. Also, we imported the DecisionTreeClassifier from sklearn and chose Gini Index as the node impurity measure and fit our data to our model so it could calculate an accuracy score ("Sklearn.tree"). In order to fit the data, we needed to fill in any missing value (nans) and convert float values to ints. Once we accomplished this, we predicted the IMBD scores for the test set. We imported the sklearn.metrics library in order to use the accuracy_score function ("Sklearn.metrics"). We received a 41% accuracy_score which was lower than expected. This could have happened for a number of reasons. We could have been underfitting by removing too many attributes and thus the decision tree did not learn the data properly. Another attempt that we made was rounding the floats to the nearest integer before casting them to ints but that lowered the accuracy score to 37%. Also, we tried using Entropy as the node impurity measure instead of Gini Index but this gave us similar results. In order to overcome underfitting, we added some features back that we pre-pruned. This improved the accuracy to 46%. In addition, we tried splitting the train and test sets in a variety of ways. For example, we split the dataset into 75% for the train set and 25% for the test set, however that did not improve the accuracy.

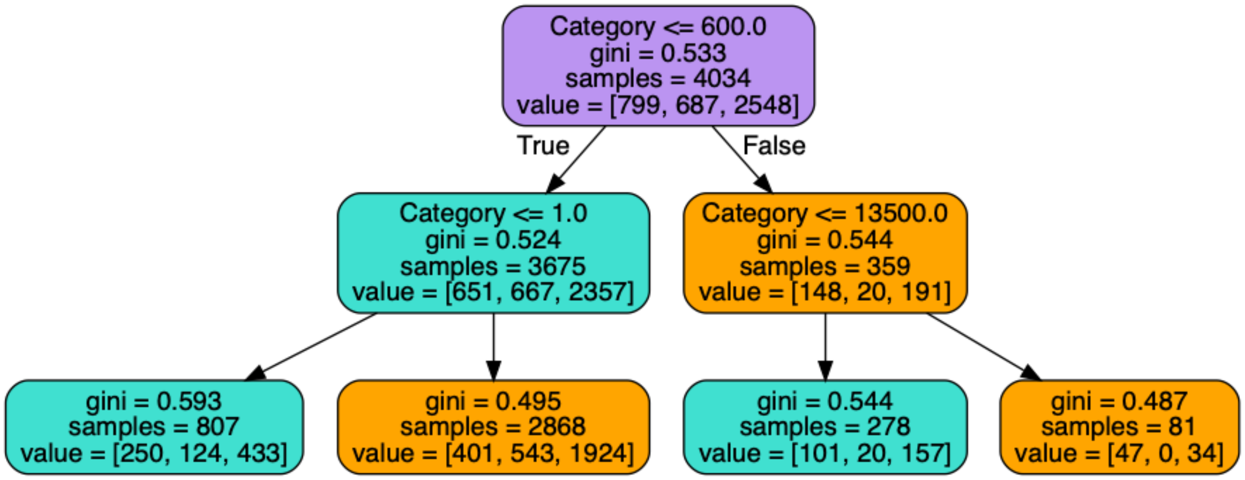
After getting these low accuracy scores, we decided to shift our approach. The next approach that we took was evaluate more attributes in order to have more information to make correct classification predictions. It was difficult to incorporate some of the features because their values were strings and our classifiers couldn't properly fit the data from our dataframe. We needed to clean our data from all of the null values and rethink our feature selection once again. First, we added back some features such as genre so that our classifier would learn the train set data better. We still had to deal with the string issue. To overcome this, we used sklearn's LabelEncoder which converted all of the string values in our dataframe to values from 0 to n_classes-1 (ints). Once all of our dataframe types were ints we could properly fit the data into various classifiers. We also tried using Kmeans Clustering classifier instead of a decision tree approach ("Sklearn.cluster"). Using the Kmeans classifier was a challenge, because this was the first time we were using this library. We had to do extensive research in order to choose our parameters. We were still met with poor results so we returned to our decision tree approach but tweaked it. We started by adding a column to our data set and dividing the IMDB ratings into three different categories blockbuster, hit and flop.

Pre-pruning the decision tree features did not give us good results, our decision tree was cut pre-maturely and was not yet fully grown. As a result, we decided that we wanted to post prune our decision tree to enhance our results. We wanted to figure out what features had the strongest correlation with the imdb_score feature column. Therefore, we imported the numpy library in order to use its corr (correlation) function and loop through each feature to find its correlation score with the imdb_score column. We were getting poor correlations for all of the string columns but this was due to the conversion of strings to ints. At that point, we

decided to ignore those attributes for the time being. The attributes that had the highest correlation with the imdb_score feature were num_voted_users
, num_critic_for_reviews, num_user_for_reviews , and movie_facebook_likes. Later, we realized that these features could not be used in our model because this data was obtained after a movie was released which defeated the purpose of this project. As a result, we picked the features with the highest correlation with the imdb_score and were available before movie production. Our top features were: director_facebook_likes (0.17), actor_1_facebook_likes (0.08), actor_2_facebook_likes (0.08), and total_cast_facebook_likes (0.09). We added the correlation matrix plot below which we obtained using the matplotlib library and the correlation scores.



Another approach that we had was to aggregate some of the features into one feature. This approach led us to adding a new column into our dataset. We added a new column called ‘Categories’, which was our classification feature that we were trying to predict. After, we divided our movies into three different categories. As soon as our dataset was ready, we preprocessed our data using the LabelEncoder. Our ‘Category’ values were assigned to 0 for flop, 1 for hit, and 2 for a blockbuster. We went ahead and made a decision tree classifier with Gini Index using the features with the highest correlation with the imdb_score and that were available before movie production. First, we chose the default parameters for our decision tree classifier and our accuracy rate was 74%. We experimented with the parameters of the classifier and also changed the splitting percentages of the dataset. We changed the max_depth parameter’s value so that the tree didn’t grow until all leaves were pure. We started with a value of 2 and increased until reaching 5. It was surprising that our best accuracy rate was 77% at value 2. Therefore, we continued with it and changed the min_sample_leaf parameter. This parameter restricted a node to become a leaf node until it had the required number of samples. Next, we began with setting the value to 2 until reaching 5. Surprisingly, 5 had the best results and our accuracy score increased to 85.1%. This major improvement was largely due to post pruning the data instead of pre pruning. By pre pruning our decision tree, it is very possible that we prematurely terminated our decision tree while it was in the tree-growing process. This gave us a low accuracy because our tree didn’t learn our dataset sufficiently. Post pruning allowed us to build a fully grown tree and then cut out features that were not as relevant to the imdb_score and our newly formulated feature; ‘Category’.



Finally, after getting sufficient results that we were pleased with, we tried optimizing our approach to achieve a better prediction accuracy. Next, we adjusted our cross validation techniques. We divided the dataset into two equal parts, the test set and train set were 50%. This slightly decreased our accuracy score to 81.01%. After getting these results, we wanted to use another cross validation technique to calculate our movie success prediction accuracy. As a result, we imported SVC (Support Vector Classification) and precision_score from the sklearn library to get the prediction precision scores ("Support Vector Machines"). SVC's implementation is based on libsvm, a SVM (support vector machines) library. We used this because SVMs are a set of supervised learning methods used for classification and regression. They are effective when it comes to high dimensional spaces and uses a five-fold cross validation. The precision_score function calculates the ratio of the number of true positives divided by the number of true positives plus the false positives. In the function, there are a number of parameters that we researched and experimented with in order to get a better accuracy. The 'average' parameter was initially set to 'micro' so that it could count the metrics globally and count the total true positives, false negatives, and false positives. As a result, we got a 83.2% accuracy which was very similar to the accuracy that we achieved using the accuracy_score function. We changed the value of that parameter to 'macro' which calculates the metrics for each label and finds its unweighted mean. This gave us a 94.7% precision score which was great.

By getting high accuracy scores using different cross validation techniques, we felt confident our model was successful. The last thing we did was modify our imdb_score intervals to allow a larger number of movies to fit into the less frequent categories; flop and blockbuster. There were too many movies in the hit category which we suspected to have a negative affect on our model. We wanted to experiment with different intervals, thus we started from 0-5.5 for a flop, 5.5-7.5 for a hit, and 7.5-10 for a blockbuster. This dropped the accuracy score to 63.4% and the precision score to 87.8%. Next, we changed the intervals again. This time 0-6 for a flop, 6-8 for a hit, and 8-10 for a blockbuster. The precision score was similar to the previous intervals, but the accuracy score dropped drastically to 61.5% . We continued observing the accuracy and the precision scores and it appeared that the more unbalanced our dataset got, the accuracy scores increased. Modifying the intervals helped us make our dataset more balanced than the previous attempts because less movies fell under the hit category.

Conclusion and Directions for Future Work:

Precisely predicting the success of a movie would benefit movie stakeholders along with movie viewers (Meenakshi et al. 1). Knowing if a movie is profitable is valuable and has the potential to impact investments throughout the movie industry. Also, movie viewers would be able to know in advance if it's worth their time and money to go see a specific movie. We conducted a large amount of research and strategized our plan before implementing our approach which helped us manage this project more effectively. Adjusting and optimizing the success of our model was something that we had to do frequently. We learned that it was important to first clean and preprocess the data before trying to make our predictions. Feature reduction was key in order to overcome the curse of dimensionality. We wanted to implement a decision tree algorithm to build our model but it was not as easy as we anticipated. As we progressed throughout the project, we found better ways to learn and find trends in our data which helped us make more accurate movie success predictions. At first, we were handpicking the features which we thought would be the best to include. These were met with poor results; 44% accuracy score. At this point, we knew that we had to try something new. Therefore, we decided to post prune our decision tree and only include features with the highest correlation with our imdb_score feature as long as they had data available prior to movie production. This approach significantly improved our prediction accuracy score. Another major factor that enhanced the accuracy score was the addition of a new column; 'Category'. This column classified movies into three different categories, blockbuster, hit and flop which was our goal. Once we had our model working we tried optimizing it by experimenting with different values for the parameters which changed the way our decision tree classified the movies. We used two different metrics for cross validation purposes; precision_score and accuracy_score. The highest score we obtained was a 94% in the precision_score, and 84% for the accuracy_score. This was after we split the dataset into two parts; a test set which was 20% of the dataset and a train set that had 80% of the data.

We felt confident in these results, therefore it could be used by movie stakeholders to properly classify if a move will be a flop, hit, or blockbuster. If we had more time on this project we would of tried to incorporate other machine learning techniques. This can be done by discovering and applying new combinations of features which have higher correlations to the IMDB rating/Categories. We had a difficult time including features that had string data type values. If we were able to incorporate these features, that might of enhanced our model. In addition, we would have tried to search for more similar datasets which we could apply our model to for cross validation purposes and making sure we weren't overfitting. Also, this would have given us a better indication of how our model would react in different environments. Even without the future work, we are confident in our model and think that it may benefit a large amount of people if it is applied. Overall, we feel like we have learned a substantial amount from this project and we will apply the knowledge we have gained in the future.

Works Cited

Ahmad, Javaria, et al. “Movie Success Prediction Using Data Mining.” *An Introduction to Biometric Recognition - IEEE Journals & Magazine*, Wiley-IEEE Press, ieeexplore.ieee.org/document/8204173.

Bronshtein, Adi. “A Quick Introduction to the ‘Pandas’ Python Library.” *Towards Data Science*, Towards Data Science, 18 Apr. 2017, towardsdatascience.com/a-quick-introduction-to-the-pandas-python-library-f1b678f34673.

“Cross-Validation: Evaluating Estimator Performance¶.” *1.4. Support Vector Machines - Scikit-Learn 0.19.2 Documentation*, scikit-learn.org/stable/modules/cross_validation.html.

“IMDB 5000 Movie Dataset - Dataset by Popculture.” *Data.world*, 7 Oct. 2016, data.world/popculture/imdb-5000-movie-dataset.

Meenakshi, K, et al. “A Data Mining Technique for Analyzing and Predicting the Success of Movie.” *Journal of Physics: Conference Series*, vol. 1000, 2018, p. 012100., doi:10.1088/1742-6596/1000/1/012100.
http://iopscience.iop.org/article/10.1088/1742-6596/1000/1/012100/pdf

“Sklearn.metrics.accuracy_score¶.” *1.4. Support Vector Machines - Scikit-Learn 0.19.2 Documentation*, scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html.

“Sklearn.tree.DecisionTreeClassifier¶.” *1.4. Support Vector Machines - Scikit-Learn 0.19.2 Documentation*, scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

“Sklearn.cluster.KMeans¶.” *1.4. Support Vector Machines - Scikit-Learn 0.19.2 Documentation*, scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

“Support Vector Machines¶.” *1.4. Support Vector Machines - Scikit-Learn 0.19.2 Documentation*, scikit-learn.org/stable/modules/svm.html.