

Team 14 IMDb Movie Analysis

Alexander Buchholz
abuchho1@stevens.edu

Kristina Cheng
kcheng5@stevens.edu

Patrick Dalrymple
pdalrymp@stevens.edu

I. INTRODUCTION

Major film companies often spend inordinate amounts of money throughout a film's production process in hopes that their feature will become the next blockbuster and produce sizeable monetary returns. On the contrary, there are also a lot of smaller market film producers who are often constrained by tight budgets and are still looking to make a decent profit. Ideally, there would be a method to assist these production companies and small market movie makers in predicting at the outset whether all of their hard work and money will be worth it in the end. Machine learning is a useful technique in creating such a method where a movie's return on investment can be predicted before it is actually produced. As long as data is available and the features of interest of the film are chosen, a reasonable model should be able to be formed to help companies and creators decide whether to proceed with a production before it is too late.

II. PROBLEM STATEMENT

As suggested above, the goal of our project was to predict a movie's return on investment (ROI) given factors that include the movie's average rating, its top two starring actors, its director, and its genre. For the purpose of this project, the movie's ROI was defined as the ratio of its worldwide earnings to its budget. To predict ROI, three different classes of algorithms were implemented: Neural Networks, Decision Trees, and Regression Analysis. Techniques were also used to optimize each of the individual algorithms before analyzing the overall performance of each algorithm to select an algorithm believed to be most appropriate for the problem. As supplemental material, the problem was converted into a classification problem to predict whether a movie would be profitable or not based on the same features listed above.

A. Related Work

The problem of predicting a movie's return on investment or box office numbers has been attempted in the past. A few examples and the nature of their results are discussed here. Lash and Zhao predicted a movie's return on investment using a number of regression models that included LASSO, Support Vector Regression, Ridge Regression, CART, M5P Tree, and REP Tree. Based on the Root Mean Squared Error (RMSE) cost function, LASSO regression was determined to perform the best for the predictions. Shahrivar and Jernbacker predicted box office revenue of movies using Decision Trees, Support Vector Machine, and KNN. Respective success rates of 15%, 11%, and 12.2% were achieved showing that the decision tree

model seemed to be the best method Galvao and Henriques attempted to predict the profit of movies through implementation of neural networks, regression, and decision trees. The researchers used an interval approach to define success where if a prediction was within a desired interval of the actual, it was deemed a successful prediction. The researchers found that a neural network provided the best measure of success. Ryan Anderson attempted to predict a film's revenue based on budget, genre, release date, spoken languages, runtime, a list of production companies, cast members, crew members, and key words. A summary of Anderson's approach is provided on the website TowardsDataScience.com. Anderson implemented a regression model and was able to achieve an R2 value of 0.77.

III. DATA SET AND PREPROCESSING

In terms of the basis for the problem, the group used publicly available movie data from MovieLens, an online movie recommendation system. In particular, the group selected the "small" data set from the website that contains over 9,000 movies, 3,600 tag applications, and 100,000 ratings applied by 600 unique users. While larger data sets were available, the group felt this smaller data set would be much more manageable to work with and would be sufficient for the scope of the project. The MovieLens data set served as the starting point, but additional data was required to be extracted from the online International Movie Database (IMDb). As discussed in the problem statement, our machine learning problem was to predict the ROI based on the movie's average rating, genre, director, and top two starring actors. The rating and genre information was extracted from the MovieLens data, but the directors, top two starring actors, budget, and world wide earnings needed to be extracted from IMDb using python's IMDbPY library. A simple python script was used to call in this data. Once all of the data was collected, a good deal of preprocessing and cleaning was performed to prepare the data for algorithm implementation. The steps taken to preprocess and clean this data are described below:

- 1) Merge MovieLens data set file and compute average rating - the source data from MovieLens was actually a number of CSV files that needed to be combined into one. Furthermore, the movie ratings data was provided by individual users. Therefore a new rating field was created to get the average of the users' ratings for each movie.
- 2) Truncate genre field - many movies in the MovieLens data had multiple genres associated with them. Since the

group was concerned that multiple genres would over specify the genres field and call for a more complex model, the genre field was truncated to only include the first listed genre for each movie.

- 3) Clean budget and world wide earnings fields - most of the budget and world wide earnings values included commas, dollar symbols, other currency designations, and words such as “estimated”. Therefore, the fields were cleaned to remove these unwanted features.
- 4) Compute ROI field - since ROI was not a data field in the MovieLens or IMDb data, it was computed by dividing the world wide earnings by the budget and converting to a percentage for each movie.
- 5) Create “Is profitable?” field - this field is a binary field that includes a 1 for a movie if its ROI was greater than 1 and 0 if not.
- 6) Delete movies that did not contain 1 or more of the targeted features mentioned in the problem statement - the group decided to delete movies that had missing data. Filling in missing data via averaging was not possible for categorical features like director, actors, and genre, and filling in box office data with averages for other movies did not seem like an accurate approach, so the group opted against it.

After the data preprocessing and cleaning phase was complete, the data set was reduced from over 9,000 movies to 2,760 movies.

IV. ALGORITHM IMPLEMENTATIONS

A. Neural Networks

Two different types of Neural Network algorithms were implemented: a regression algorithm and a classification algorithm. In order to compare the two implementations fairly and eliminate as much bias as possible, both algorithms were implemented using the same dataset, which was the 2760 IMDb movie database preprocessed using the methods described above. Both algorithms were also built on top of the same toolset, Keras. Keras is an open-source Python library running on top of TensorFlow, which is an open-source library developed for machine learning applications. Both algorithms also applied an additional processing step to clean the data further for Keras: non-numeric categories (genre, director, and actors) were encoded as integers and data was standardized. Finally, both algorithms were also validated with the same technique, K-Fold validation. The implementations used for the project were derived as a combination of techniques learned in the class, as well as Keras regression and classification examples from Machine Learning Mastery [6].

Regression was implemented through KerasRegressor, which is a backpropagation algorithm. Rectified Linear Unit (‘relu’) activation was used for all testing in the regression problem. First, a variety of loss functions were tested to find a neural network that would converge quickly enough to produce an acceptable result. Mean Squared Error (MSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error

(MAPE), and Mean Squared Logarithmic Error (MSLE) were all tested against the same data set to find an option that would converge. Of those loss functions, only MSLE actually produced a functional output, so MSLE was used for all further regression testing. The MSLE loss function likely performed better because it was less punishing towards large ROI values, which were common in the data set. The algorithm was then tested with a variety of different epoch numbers, batch sizes, hidden layers, and nodes per layer, to find a combination of inputs that would produce a small error without sacrificing processing time. Since processing time was such a limiting factor, the following simple inputs in Fig. 1 were used, with the resulting adjusted MSE and standard deviation shown.

An example of a regression output is shown below. As the epoch number increases, the loss value gradually decreases through gradient descent. Fig. 2 shows the results, with the final MSE at only 0.23% of ROI, meaning the regressive neural network is an incredibly attractive option.

To treat the problem as a classification problem ROI was numerically encoded into two categories: $\text{ROI} \geq 100$ is encoded to 1, while $\text{ROI} < 100$ is encoded to 0. Classification was implemented through KerasClassifier, which is another backpropagation-based algorithm. Rectified Linear Unit (‘relu’) activation was used for all input testing in the regression problem, with softmax activation used for the absolute layer. Since the classification problem was inherently simpler, only one loss function was tested, which was categorical cross-entropy. Categorical cross-entropy produced satisfactory convergence so no longer loss functions were used. Again, the algorithm was then tested with a variety of different epoch numbers, batch sizes, hidden layers, and nodes per layer, to find a combination of inputs that would produce a small error without sacrificing processing time. The following inputs in Fig. 3 were used, with the accuracy and standard deviation shown.

An example of classification output is shown below. As the epoch number increases, the loss value gradually decreases through gradient descent. Additionally, accuracy also increases with epoch number. Fig. 4 shows the final results, with the final accuracy at 76.85% success with a standard deviation of 14.17%.

While these numbers show that the classification network had some success, they are surprisingly less accurate than the metrics shown in the regression implementation. This loss in accuracy can likely be attributed to the nature of the data used and the way that data was classified: since both large and small positive ROI values (for example, 110% and 1100%) would be classified with the same weight, a large amount of information is lost during classification, which hinders the performance of the algorithm.

While the techniques used in each neural network implementation offered satisfactory results, additional work can be done to optimize the performance of each algorithm. Metrics like the number of nodes per layer, the number of hidden layers, number of epochs, the value of K in k-fold validation, and batch size were all picked at values that would keep

Regression Results

Loss Function	Epochs	Batch Size	# Hidden Layers	# Nodes per Hidden Layer	K-Fold	MSE	Standard Deviation
MSLE	50	5	1	10	10	0.23%	0.11%

Fig. 1. Regression Neural Network Inputs and Metrics

```

Epoch 1/50
2484/2484 [=====] - 1s 370us/step - loss: 25.6786
Epoch 2/50
2484/2484 [=====] - 1s 343us/step - loss: 15.1120
Epoch 3/50
2484/2484 [=====] - 1s 297us/step - loss: 10.8615
Epoch 4/50
2484/2484 [=====] - 1s 258us/step - loss: 8.4868
Epoch 5/50
2484/2484 [=====] - 1s 273us/step - loss: 6.9140
Epoch 6/50
2484/2484 [=====] - 1s 265us/step - loss: 5.7725
Epoch 7/50
2484/2484 [=====] - 1s 261us/step - loss: 4.9055
Epoch 8/50
2484/2484 [=====] - 1s 259us/step - loss: 4.2277
...
Epoch 47/50
2484/2484 [=====] - 1s 262us/step - loss: 1.6339
Epoch 48/50
2484/2484 [=====] - 1s 233us/step - loss: 1.6339
Epoch 49/50
2484/2484 [=====] - 1s 242us/step - loss: 1.6339
Epoch 50/50
2484/2484 [=====] - 1s 261us/step - loss: 1.6339
276/276 [=====] - 0s 163us/step
Standardized: 0.23 (0.11) MSE

```

Fig. 2. Regression Neural Network Output

computation time low. Additional work could be done to significantly slow computation time, but might produce a lower error or higher accuracy. Additional analysis could also be performed on the various input parameters to determine what features are the most valuable to the tree. Any additional complexity would likely require either more time or a stronger GPU to test the network in a sufficient amount of time, however.

B. Regression Tree Implementation

The decision tree implementation was fully performed using python in a Jupyter notebook environment. Python libraries such as pandas, numpy, sklearn, and matplotlib were utilized to aid in the construction, visualization, and optimization of trees. Furthermore, the cleaned data set containing 2,760 movies was used in all tree implementations. The methodology behind the regression tree implementation was to identify a baseline tree using default parameters. Then the tree was to be optimized via a pruning approach that consisted of varying specific parameters of the tree for optimization. In particular, three trees were created: a tree optimized for maximum depth, a tree optimized for maximum leaf nodes, and a tree optimized for the best combination of maximum depth and maximum leaf nodes. To assess the performance of each tree, a number of performance metrics were computed and included Mean Squared Error (MSE), Mean Squared Logarithmic Error (MSLE), Mean

Absolute Error (MAE), and R2 value. Before the different trees were implemented, some final data preprocessing was required. First, each of the categorical features needed to be converted into numerical data. These features included genre, director, actor1, and actor2. To do so, the pandas function `get_dummies()` was utilized to encode each of these features into a series of 1s and 0s. For example, if a movie was listed as “comedy” for genre, a 1 would show for the comedy column while 0s would show for the remaining genre type columns. By taking this approach, the number of features for each movie unfortunately increased by a wide margin as the data was now formatted to show columns for all of the unique genres, unique directors, and unique actors as opposed to one column for each. Second, the data needed to be separated into a training set and a testing set. The `train_test_split()` function was used to take care of this, and the testing set size was chosen to be 30% of the data.

1) *BASELINE TREE*: To begin the algorithm implementation, a baseline regression tree was created using default parameters. The performance metrics for this tree are shown in Fig. 5.

As expected, the tree exhibits clear overfitting as shown by the perfect scores for the training set data and the extremely poor results for the testing set data. Since there were no parameters specified, the tree was built until all features were used and all nodes could no longer be branched out. As a result, a very specific tree was built that fits to the training data perfectly and cannot generalize well with the testing data.

2) *OPTIMAL MAXIMUM DEPTH TREE*: The next step was to build a tree based on the optimal maximum depth. To do so, a number of regression trees were created where the maximum depth was varied from 25 to 325 at step sizes of 25. A tree with max depth of 339 was also included as that was the upper bound for the depth. A function was created to plot the training and testing set errors as a function of max depth for the 4 different metrics. These plots are shown in Fig. 6.

While it is hard to discern any trends in the MSE and R2 plots due to the magnitude differences between the training errors and the testing errors, the MSLE and MAE seem to indicate the trends a little more clearly. Focusing on the MSLE specifically, the plot clearly shows that the tree tends to overfit as the tree depth increases. However, there does seem to be an optimal tree depth where the test set error decreases with the training set error and reaches an optimal minimum value. To pinpoint this optimal value, trees with a more focused range of depths around the estimated optimal value were created, and their training and testing set errors were plotted again and shown in Fig. 7.

Based on the MSLE plot, the testing set error appears to

Classification Results

Loss Function	Epochs	Batch Size	# Hidden Layers	# Nodes per Hidden Layer	K-Fold	Accuracy	Standard Deviation
Categorical Cross Entropy	50	5	2	10	10	76.85%	14.17%

Fig. 3. Classification Neural Network Inputs and Metrics

```
Epoch 1/50
2484/2484 [=====] - 1s 300us/step - loss: 61.5772 - accuracy: 0.5390
Epoch 2/50
2484/2484 [=====] - 1s 227us/step - loss: 2.7794 - accuracy: 0.7359
Epoch 3/50
2484/2484 [=====] - 1s 235us/step - loss: 1.1283 - accuracy: 0.7705
Epoch 4/50
2484/2484 [=====] - 1s 242us/step - loss: 0.7554 - accuracy: 0.7810
Epoch 5/50
2484/2484 [=====] - 1s 262us/step - loss: 0.5941 - accuracy: 0.7935
Epoch 6/50
2484/2484 [=====] - 1s 250us/step - loss: 0.5712 - accuracy: 0.8064
Epoch 7/50
2484/2484 [=====] - 1s 258us/step - loss: 0.5534 - accuracy: 0.8108
Epoch 8/50
2484/2484 [=====] - 1s 250us/step - loss: 0.5525 - accuracy: 0.8100
...
Epoch 47/50
2484/2484 [=====] - 1s 237us/step - loss: 0.4487 - accuracy: 0.8333
Epoch 48/50
2484/2484 [=====] - 1s 238us/step - loss: 0.4492 - accuracy: 0.8333
Epoch 49/50
2484/2484 [=====] - 1s 236us/step - loss: 0.4502 - accuracy: 0.8333
Epoch 50/50
2484/2484 [=====] - 1s 241us/step - loss: 0.4493 - accuracy: 0.8333
276/276 [=====] - 0s 185us/step
Baseline: 76.85% (14.17%)
```

Fig. 4. Classification Neural Network Output

```
Performance Measures - Training Data:
MSE: 0.0
MSLE: 0.0
MAE: 0.0
R^2: 1.0
Performance Measures - Testing Data:
MSE: 209128933.7841421
MSLE: 2.8642676564703358
MAE: 947.1437077294686
R^2: -0.004670408107555035
```

Fig. 5. Baseline Performance Metrics

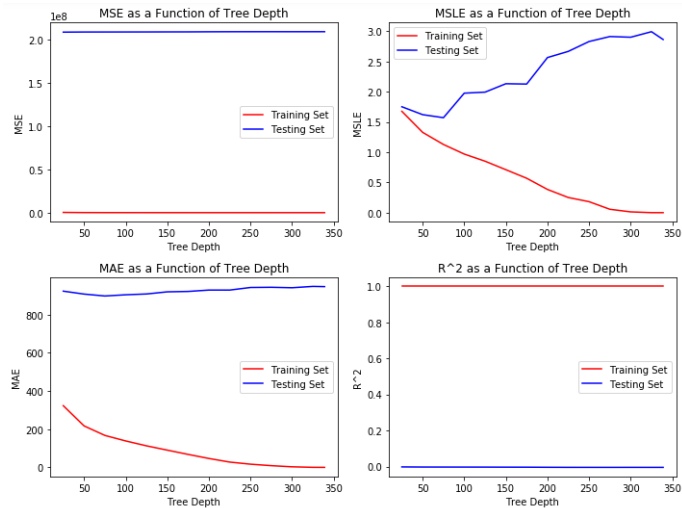


Fig. 6. Tree Depth Metrics

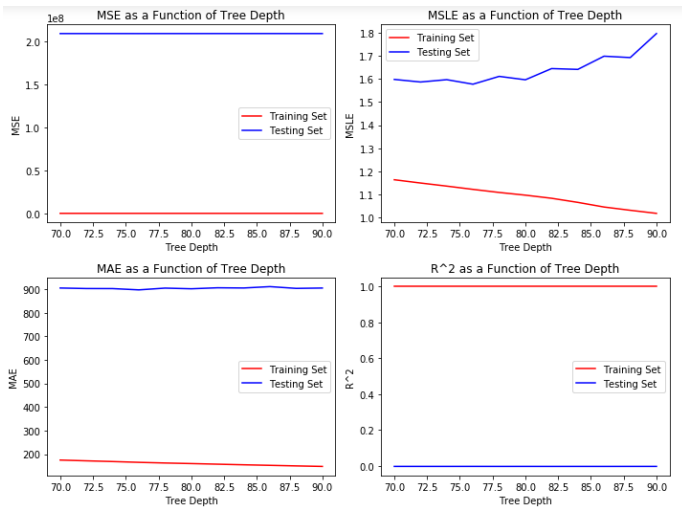


Fig. 7. Optimized Tree Depth Metrics

reach a minimum value at a max depth of 76. A regression tree with this depth size was created, and the performance results were computed and are shown in Fig. 8.

3) *OPTIMAL MAXIMUM LEAF NODES TREE*: The next step was to build a tree based on the optimal maximum leaf nodes. A similar approach as the max depth tree was taken where a number of regression trees with varying sizes of max leaf nodes were created and the training and testing set errors were plotted. In particular, the maximum leaf nodes parameter was varied from 100 to 1800 with step sizes of 100. A tree with maximum leaf nodes of 1,931 was also included as this

Performance Measures - Training Data:
MSE: 52607.38154142203
MSLE: 1.1220242618342415
MAE: 165.49975862997013
R²: 0.9999999997337697
Performance Measures - Testing Data:
MSE: 208778102.4751493
MSLE: 1.5775915529393403
MAE: 897.2124337364525
R²: -0.0029849893182711806

Fig. 8. Tree Depth Performance Metrics

was the upper bound. The errors were plotted and are depicted in Fig. 9.



Fig. 9. Tree Leaf Metrics

Once again, the MSLE plot seemed to be the best indicator of error trends, and an optimal value can be detected where the training set error reaches a minimum. A group of trees with a more focused max leaf node range around the optimal value was plotted to pinpoint the optimal number. This plot is shown in Fig. 10.

Considering the MSLE plot again, the testing set error appears to reach a minimum at 274 leaf nodes. A tree was then built with 274 leaf nodes, and the performance metrics are shown in Fig. 11.

4) **TREE WITH OPTIMAL COMBINATION OF MAX DEPTH AND MAX LEAD NODES:** A final tree was implemented to consider whether there was an optimal combination of max depth and max leaf nodes that could exceed the performance of the other trees. To determine the optimal combination of these parameters, grid search with cross validation was used to test a number of trees with varying max depth and max leaf nodes values. For the cross validation, the number

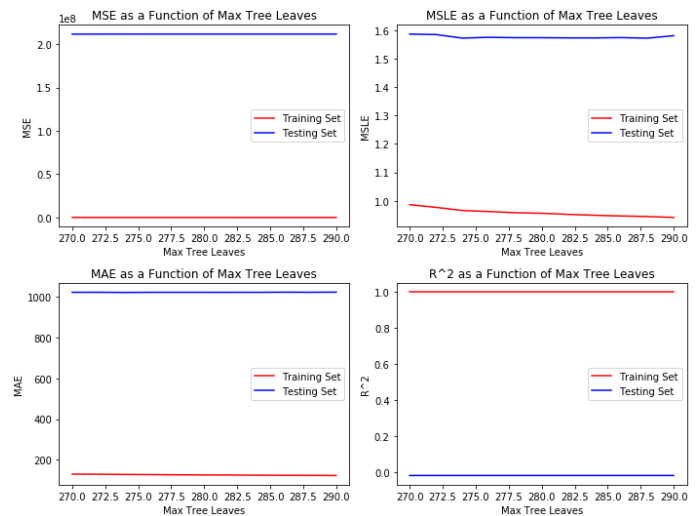


Fig. 10. Optimized Tree Leaf Metrics

Performance Measures - Training Data:
MSE: 28584.530146538178
MSLE: 0.9659258576320708
MAE: 128.34770262689946
R²: 0.9999999998553422
Performance Measures - Testing Data:
MSE: 211713272.85378292
MSLE: 1.57221916600109
MAE: 1023.0052110196295
R²: -0.01708575848879068

Fig. 11. Tree Leaf Performance Metrics

for folds was set to 3. To narrow down the optimal ranges of the two parameters, trial and error was used until the GridSearchCV method output a tree with values in the middle of the ranges that were being tested. Once these ranges were narrowed down, a final focused set of lists were set as the parameter grid. The final list of potential max depth values turned out to be [118, 120, 122, 124], while the final list of max leaf nodes values turned out to be [228, 230, 232]. Through execution of the test script, the optimal combination of max depth and max leaf nodes turned out to be 122 and 230 respectively. A tree was created with these parameters, and the performance metrics were computed and are shown in Fig. 12.

5) **COMPARISON OF DIFFERENT TREES:** A table was created to compare the performance of the three regression trees and is shown in Fig. 13.

Considering the above performance numbers, all of the trees ended up with extremely poor results. However, if one had to be chosen, the tree with the optimal max leaf nodes seemed to perform best. This tree produces the lowest training set error. Furthermore its testing set error is lower than the tree that was

Performance Measures - Training Data:

MSE: 35631.9213363672
 MSLE: 1.0447913211477717
 MAE: 142.7075263811434
 R²: 0.999999998196774

Performance Measures - Testing Data:

MSE: 211681972.98064604
 MSLE: 1.5884447809803033
 MAE: 1020.7559393908376
 R²: -0.016935391651695886

Fig. 12. Optimal Combination Performance Metrics

Training Set Error			
	Tree_OptimalDepth	Tree_OptimalMaxLeafs	Tree_OptimalCombination
MSE	52607.381541	28584.530147	35631.921336
MSLE	1.122024	0.965926	1.044791
MAE	165.499759	128.347703	142.707526
R ²	1.000000	1.000000	1.000000
Testing Set Error			
	Tree_OptimalDepth	Tree_OptimalMaxLeafs	Tree_OptimalCombination
MSE	2.087781e+08	2.117133e+08	2.116820e+08
MSLE	1.577592e+00	1.572219e+00	1.588445e+00
MAE	8.972124e+02	1.023005e+03	1.020756e+03
R ²	-2.984989e-03	-1.708576e-02	-1.693539e-02

Fig. 13. Performance Metrics Summary

optimized for max depth. The tree's testing set performance metrics are very comparable to those for the tree with the optimal combination of parameters, so the training set error metrics act as the tiebreaker. Therefore, it is logical to choose the tree optimized for max leaf nodes.

6) **CLASSIFICATION PROBLEM:** Due to the poor results obtained from the regression problem, the group experimented with a classification problem. Although a completely different problem, the group was curious to see if more reasonable results could be achieved. For this problem, a model is used to predict whether a movie is profitable or not based on the same features. To carry out this problem for classification trees, the same approach was taken as the regression tree problem – a baseline tree was created and then pruned via optimization of parameters that included max depth, max leaf nodes, and the best combination of the two. A similar analysis was carried out in python and can be viewed in detail in the source code. In terms of the results, the performance metrics of each tree were computed and are compared in Fig. 14.

7) **FUTURE WORK AND LESSONS LEARNED:** Based on the extremely poor performance of the trees, it was evident that the data set used in this project probably wasn't best suited for this specific type of problem and algorithm implementation. Reflecting on how the group approached this project, the problem of predicting a movie's ROI was essentially created based on choosing features that the group found interesting. In hindsight and by learning throughout the process of this project, it would have made more sense to do a features

	Tree_OptimalDepth	Tree_OptimalMaxLeafs	\
Training Prof. Acc.	100.000000	99.627098	
Training Not Prof. Acc.	86.377709	94.427245	
Testing Prof. Acc.	96.561605	95.988539	
Testing Not Prof. Acc.	9.230769	8.461538	
	Tree_OptimalCombination		
Training Prof. Acc.	98.943443		
Training Not Prof. Acc.	60.990712		
Testing Prof. Acc.	95.845272		
Testing Not Prof. Acc.	6.153846		

Fig. 14. Classification Performance Metrics

analysis to determine which features most likely determine a movie's success. Furthermore, given that the dataset was only 2,760 movies long with over 1,000 unique actors and 1,000 unique directors, the data set was way too specific from the start and did not have enough commonality to create a reasonable regression tree model. Given that a regression tree model computes its outputs based on the average value at each node, the plethora of unique actors and directors most likely made this a very difficult task, hence the extremely large error computations. Lastly, while the categorical features such as director and actors were converted to binary data and thereby greatly increased the number of features for each movie, a better approach might have been to map actor and director names to some sort of performance metric. For example, since a potential theory is that a-list actors appearing in a film would be a good indicator of the film's ROI, the actor names could be mapped to their career earnings. In this case, the more well known actors would have a higher metric compared to lesser known actors who have not made a lot of money in their careers. The current method of converting the actor and director names to binary coding does not help the model much when working with a data set of this size and so many unique actor names. In other words, the model can only really work off of the frequency that the actor and director names appear, and considering that the majority of names in the data set only appeared once, the model was almost set up for failure from the start.

C. Linear and Logistic Regression

We decided to implement linear and logistic regression to predict movie ROI because of the power of the two techniques to identify which variables influence the outcome with statistical significance and because of the ease with which the estimates of the magnitudes of those effects can be interpreted. We use linear regression to predict ROI as a continuous value and logistic regression to predict whether a movie would be profitable (ROI > 1).

1) **Data Preparation and Motivation:** For the regression portion, I used the Full Movie Lens dataset, which includes more than 45,000 different titles. After cleaning and filtering the data, however, we ended up with only about 5,400 unique movies. I removed rows with null values in important columns (such as lead actor), and I also chose to remove movies with revenues less than \$1,000. I chose this threshold because lower revenues are strong indicators that the people

who produced the movies did not intend for them to be commercialized or seen by a wide audience. Such movies are likely to have a completely different set of associated circumstances and set of factors which influence their ROI. After data cleaning, I added a few features to the dataset that I thought might influence ROI based on outside knowledge. I added the following key-word related flags to explore whether their relationships with the target variable might be significant: *Is_independent_film*: whether or not the movie was an independent film *Based_on_novel*: whether or not the movie was based on a novel *Christmas_release_flag*: whether or not the movie was released in December, a peak time for people to visit theaters due to children being out of school *Summer_release_flag*: whether or not the movie was released in the summer, for the same reasons as the Christmas release flag *Has_female_lead*: whether or not the movie has a female lead *Big_six*: whether the movie was produced by one of the “big six” companies (Fox, Warner Brothers, Paramount, Columbia, Universal, or Disney) In addition to the above, I also included flags for whether the lead actors and directors were considered “top tier.” I defined this metric as whether or not the person’s name appeared more than ten times in the dataset since the majority of names only appeared 1-3 times. The remaining predictors included the number of languages spoken, the cast and crew size, the log of the budget, and dummy variables for each genre.

2) *Ridge and Lasso Linear Regression Model Implementation*: I first tried using regression with ROI as a continuous variable. I used both l1 and l2 regularization techniques and tested values of C in the set [0.001,0.01,1,1,5,10,25]. I also used the default $k = 5$ cross validation. The performance of the models in terms of mean squared error and the r-squared values was extremely poor, which is likely due to the underlying poor quality of the predictive variables themselves. In fact, the r-squared value for every model iteration tested never exceeded 3%. The predictors simply do not adequately explain the variation in the outcome and are unable to distinguish between small differences in ROI. In addition to having poor performance metrics, none of the regression coefficients were statistically significant.

3) *Logistic Regression Model Implementation*: : I then tried re-defining the target variable as a binary classification denoting whether the movie was profitable. Again, I used k-fold cross validation with $k = 5$ and tried both l1 and l2 penalties with $C = [0.001,0.01,1,1,5,10,25]$. F1 score results with the different hyperparameter settings are indicated in Fig. 15—they changed only slightly with different regularization terms.

It turned out that the coefficient with the highest magnitude was the independent film flag—apparently independent films are more than two times less likely to be profitable. And then the highest magnitude positive coefficients all corresponded with features I predicted might be strongly related to ROI—the big six flag, the Christmas and summer flags, the critic vote average and the top lead actor flag. The regression coefficients in order of descending importance are listed below. The values

Value of C	Type of Penalty	F1 Score
.001	L1	0.82673943
.001	L2	0.83648393
.01	L1	0.83411876
.01	L2	0.83178937
.1	L1	0.83125933
.1	L2	0.83650953
1	L1	0.83493185
1	L2	0.83282675
5	L1	0.83028455
5	L2	0.83028455
10	L1	0.83028455
10	L2	0.82968988
25	L1	0.83028455
25	L2	0.83087862

Fig. 15. F1 Scores

of the coefficients are consistent with what someone might expect to be true about the movie industry given cultural heuristics. Also, the results indicate that factors which are typically associated with high budgets (big production company, famous actors, etc.) also tend to be associated with positive ROIs despite the risks and high costs. Results are shown in Fig. 16.

V. COMPARISON OF ALGORITHM RESULTS

Based on the earlier discussion of the performance of the regression tree models, these have been ruled out as a best






Variable	Coefficient	
is_independent_film	-1.1560	
big_six_flag	0.8907	
vote_average	0.4221	
christmas_release_flag	0.4155	
is_top_lead_actor	0.3906	
summer_release_flag	0.3740	

Fig. 16. Coefficient Results

method for predicting a movie's return on investment. The regression tree models may have suffered from the specificities of the data set and the implementation of the categorical feature encoding. A more refined data set that includes only movies with directors and actors that are listed more than a set threshold value might have better served these models. For example movies that included a director name or an actor name that appeared only once in the entire data set would be deleted. The Ridge and Lasso regression models also suffered from poor results and likely were due to the poor quality of the predictive variables. Ultimately, it was the neural network implementation that produced the most reasonable results having a final MSE of 0.23. Given that the other two implementations performed so poorly, there is not much reason for further comparison as the neural network was the best predictor by a long shot. In terms of the classification problem, the decision trees offered good performance with predicting when a movie was profitable, but poor performance for predicting when a movie was not profitable. Although an attempt was made to optimize the trees, the end results still hinted at overfitting as the models seemed to pick up on the fact that the majority of movies were profitable. In terms of the regression classification model, the logistic regression model produced solid results with a high precision, recall, and F1 scores. The neural network implementation also produced favorable results by producing an overall accuracy of 77%. Since the models used different metrics to assess their performance, additional work would be required for an even comparison, but for this project's sake, both produce very favorable results.

VI. FUTURE RESEARCH AND CONCLUSION

After comparing the results of each of the algorithms implemented, the neural network was the best predictor of success in determining a film's return on investment. For determining whether a movie was profitable or not, the neural network and logistic regression models both produced favorable results.

One interesting observation we noted during this process was that despite the fact that classification is in theory a simpler task than regression (since there are only two possible values), classification did not always perform better than regression for each of the three tested models. In fact, for the neural network model, classification performed a bit worse. The group learned a lot over the course of the project, and all of the group members agreed that things would most likely have been done differently knowing what they know now. It was clear that the nature of the underlying predictor variables had significant impacts on the quality of the predictions. At the outset of the project, the grouped formed the problem statement and chose predictor variables based on what sounded interesting. In hindsight, a more thorough features analysis would have been conducted to determine best predictors of movie success, which could have had a much more positive impact on the algorithm results. Additional experimentation could also be done to optimize the efficacy of the different algorithms. The neural network algorithms could use additional research to experiment with numbers of nodes, layers, and batch sizes in order to optimize the results, rather than processing time. Ideally, any future work would take the lessons learned from the successes and failures described here in order to implement more effective solutions at solving the movie problem.

VII. TASK ALLOCATION

Alex Buchholz - Neural Network Implementation, Report Writing, LaTeX formatting Kristina Cheng - Regression Implementation, Data cleaning and preprocessing of 45,000 movie data set, report writing Patrick Dalrymple - Decision Tree Implementation, Data cleaning and preprocessing of 9,000 movie data set, report writing

REFERENCES

- [1] Related work - Lash and Zhao: <https://arxiv.org/pdf/1506.05382.pdf>
- [2] Related work - Shahrivar and Jernbacker <http://www.diva-portal.org/smash/get/diva2:1106715/FULLTEXT01.pdf>
- [3] Related work - Galvao and Henriques: <https://pdfs.semanticscholar.org/6d4f/1003fd164ffe30e2e45dd252715fecf9e61.pdf>

- [4] Related work - Ryan Anderson: <https://towardsdatascience.com/what-makes-a-successful-film-predicting-a-films-revenue-and-user-rating-with-machine-learning-e2d1b42365e7>
- [5] Data cleaning reference - <https://www.kaggle.com/danofer/movies-data-clean>
- [6] Keras References - <https://machinelearningmastery.com/regression-tutorial-keras-deep-learning-library-python/>
<https://machinelearningmastery.com/multi-class-classification-tutorial-keras-deep-learning-library/>