

# MIS 545 Project Stage 3 Report

Group 1: Jolly Miners

(Phillip Chan, Felicia Chang, Jo Jo Lin, Katie Wang, Haozhe Xu)

December 12, 2019

## Problem Description

Movie production is a complex process divided into development, pre-production, production, principal photography, wrap, post-production, and distribution. Whether a movie is successful or not depends on various factors, such as directors, cast and crew, production company, and genres of the movie. Knowing which movies are likely to succeed and which are likely to fail before the release could benefit the production houses greatly as it will enable them to focus their advertising campaigns which itself cost millions of dollars, accordingly [1].

In the era of Web 3.0, as more and more consumers share their ratings and comments with others after watching movies through online platforms like Rotten Tomatoes, movie rating becomes an important indicator to reflect its box office and profit margins. Therefore, the prediction of movie ratings is of great importance to the industry.




















A previous study [2] has shown that the popularity of the leading actress and the combination of past successful genre and a sequel movie are crucial to the success of a movie. Based on such known key factors, the accuracies of box office prediction are 51% with linear regression, 42.2% with logistic regression, and 30% with the SVM approach [1]. However, the accuracies of movie ratings in published papers are fairly low: 14.11% with kernel regression [3] and 9.83% with neural network analysis [4].

In stage 1, our team's goal was to apply algorithms learned from class to predict whether a movie will be successful before released to the public and to improve the accuracy of predictions. In this final stage, our team will add more attributes and apply a new algorithm to improve the accuracy of predictions from our previous stages.

## Dataset Description

Our original dataset came from Kaggle, which is named "imdb-dataset". This dataset has around 10.9 thousands rows and contains 21 variables (please see the below images). However, we only have around 5,000 rows in our data points since due to the 5,000 missing value for 'budget' column. We tried to research online to see if we can find those

missing values in Stage 3; however, only some of them were found and updated. Therefore, in Stage 3, we still only have around 5,000 data points.

Columns	Columns
 id	 tagline
 imdb_id	 keywords
 popularity	 overview
 budget	 runtime
 revenue	 genres
 original_title	 production_companies
 cast	 release_date
 homepage	 vote_count
 director	 vote_average
	 release_year
	 budget_adj
	 revenue_adj

There are a total of 21 attributes, but we only choose the attributes that are relevant. We are tried to predict if the movie will be successful after it releases; therefore, only the attributes that are collected before the release date will be useful. Those attributes like 'revenue', 'keywords', 'vote\_count', etc. are the after result data, which are irrelevant. So, there are only 7 attributes were used: 'budget', 'cast\_rating', 'director', 'runtime', 'production\_companies', 'release\_date', and 'writer\_rating' ('cast\_rating' and 'writer\_rating' are from the imdb-open source).

The 'budget' column shows the cost of making each movie. The 'cast\_rating' column includes the overall rating for the cast from those movies. The 'director' column shows the names of directors for each movie. The 'runtime' column lists the duration of each movie. The 'production\_companies' column indicates what companies were involved for making each movie. The 'release\_date' column provides the date that the movie is released. And the 'writer\_rating' column shows the overall rating of writers for each movie. The following image shows the descriptive statistic and data type for those attributes:

```
> summary(main)
```

director	runtime	is_r_weekend	is.top.prodcom	budget	is.successful	writer.rating	cast_rating
Min. :1.800	Min. : 0.0	Min. :0.0000	Min. :0.0000	Min. : 1	Min. :0.000	Min. :1.600	Min. :2.100
1st Qu.:6.274	1st Qu.: 93.0	1st Qu.:0.0000	1st Qu.:0.0000	1st Qu.: 8000000	1st Qu.:0.000	1st Qu.:5.969	1st Qu.:6.089
Median :6.767	Median :103.0	Median :1.0000	Median :0.0000	Median : 20000000	Median :1.000	Median :6.600	Median :6.412
Mean :6.656	Mean :107.2	Mean :0.5068	Mean :0.4589	Mean : 33890452	Mean :0.548	Mean :6.511	Mean :6.295
3rd Qu.:7.200	3rd Qu.:117.0	3rd Qu.:1.0000	3rd Qu.:1.0000	3rd Qu.: 44750000	3rd Qu.:1.000	3rd Qu.:7.198	3rd Qu.:6.684
Max. :8.782	Max. :248.0	Max. :1.0000	Max. :1.0000	Max. :425000000	Max. :1.000	Max. :8.956	Max. :8.600

```
> str(main)
'data.frame':  4347 obs. of  8 variables:
 $ director      : num  3.8 7.52 6.56 5.34 6.45 ...
 $ runtime       : int   82 125 91 79 128 109 197 103 77 110 ...
 $ is_r_weekend  : int    1 0 1 0 1 1 0 0 1 0 ...
 $ is.top.prodcom: int    0 0 1 0 0 0 0 0 1 0 ...
 $ budget        : int  62000 3000000 3000000 270000 2000000 806948 ...
 $ is.successful : Factor w/ 2 levels "0","1": 1 2 2 1 2 2 2 2 2 2 ...
 $ writer.rating : num   4.52 6.95 6 6.25 7.88 ...
 $ cast_rating   : num   4.4 7.01 6.64 6.76 7.01 ...
```

## Data Pre-processing

### Transformation on Director, Writer, and Cast Attributes

We transformed three attributes from string values to numerical values. They are director, writer, and cast. The below table shows demonstrate the transformation. In the Imdb open-source source (<https://www.imdb.com/interfaces/>), we accessed three new datasets to perform this transformation. First, we access to title.crew dataset that contains movie rating, directors, and writers which columns are corresponded to one another. Second, we get movie rating from title.ratings. Third, we get cast members from title.principals. All datasets share a imdbId attribute (tconst), which acts as the joining key.

To calculate the director rating per movie, movie, director, directed movies of such director, and rating of directed movies are joined. Director and directed movie are joined using “right” merge() to include all movies directed by the director. Directed movie and movie Rating are joined using “inner” merge() since they are of a 1-1 relationship. There might be more than one director listed per movie, but only one director is selected. Since the dataset lists director data by its significance, the first listed director can represent the main director of the movie. After getting all movie Rating, the average is calculated using aggregate() function with “mean”.

Example of transforming director attribute

movie	director	directed Movie	movie Rating
tt78901	nm45454	tt13436	8.4
		tt50156	7.9
		tt08192	8.6

		tt07728	7.1
<b>Average Director Rating</b>			<b>8.0</b>

The writer rating is transformed using the same technique from director. Only one writer is selected to represent the main writer for that movie.

Example of transforming writer attribute

movie	writer	written Movie	movie Rating
tt78901	nm12345	tt13436	7.3
		tt50156	7.5
		tt08192	8.3
		tt07728	6.9
Average Writer Rating			7.5

There are more than more cast members for one movie. For this project, all the cast members listed are selected together to represent one movie. For example, if a movie contains 10 cast, the historical movie rating for each of the cast members are extracted. The arithmetic mean is used to calculate the final average cast rating representing one movie.

Example of transforming cast attribute

movie	cast	acted movie	movie Rating
tt78901	nm00138	tt71122	7.9
		tt10869	8.6

	nm00687	tt89478	8.1
		tt64326	6.7
Average Cast Rating			7.82

### Transformation on Release Date Attribute

The “Release Date” is transformed from date format to binary value with 1 representing the movie is released on the weekend and 0 representing the movie is released on the weekday. The Excel “if” function is used to determine that Monday-Thursday is considered as “Weekday” and Friday-Sunday as “Weekend”. This same approach is maintained from Stage 1.

### Transformation on Production Company Attribute

The “Production Company” is transformed from string format to binary value with 1 representing the movie belongs to Top 20 production companies and 0 representing the movie does not belong to Top 20 production companies. According to the-numbers.com, it ranks the movie production companies based on the total worldwide box office revenue, which is relevant to how much they gain from playing their movies in the theaters. The Excel “if” function is used to find if the movie contains any of the top 20 production companies. This same approach is maintained from Stage 1.

### Data Cleaning

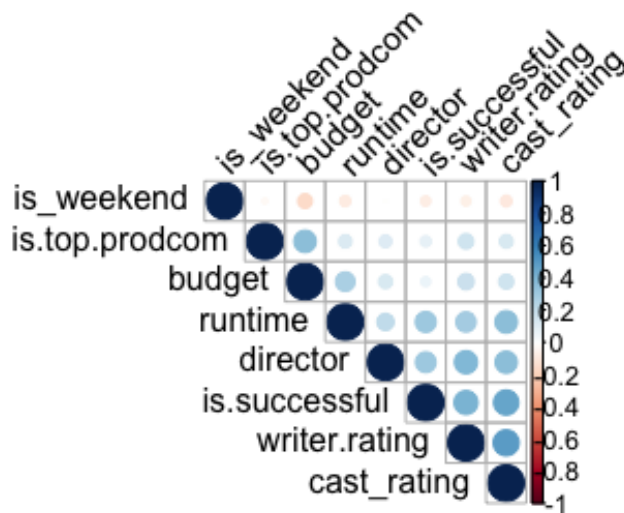
The “budget” attribute is numerical values and it stays the same format. However, the main dataset contains many empty budget values, which accounts for approximately 52%. These missing values are shown as either 0 or no value. We use both Excel and R functions to remove the missing values. This deletion reduces the total rows to 5,618 rows. For the next stage, we plan to use coding script to scrape a complete budget data value.

### Correlation Among Attributes

After the data transformation and data cleaning, we perform a correlation analysis in R to examine the relationships among attributes. As seen in the below chart, blue colors represent positive correlations; red colors represent negative correlations.

From looking at the correlations with “is.successful” (DV), we may see “cast” and “writer” (IVs) are the two highly correlated IVs to DV. “director” (IV) is the third highly correlated to DV.

It is also obvious to see director rating, writer rating, and cast rating have high positive correlations to one another. However, due to the small size of our attributes, we are not able to further remove these highly correlated attributes. Instead, our solution is to avoid using algorithms that prevent multicollinearity issue.



	is.successful
director	0.31118611
runtime	0.31881398
is_weekend	-0.07391443
is.top.prodcom	0.08577825
budget	0.06699804
writer.rating	0.41988958
cast_rating	0.46506001
is.successful	1.00000000

## Results of Individual Projects

### Phillip Chan

For the individual project, Phillip focused on random forest algorithm. He believes user votes and IMDB scores will be an important independent variable and will make an impact in our final project. If there aren't data with movie scores prior to the release date, he can use the current IMDB score. Movie scores is a great predictor to determine viewers to watch a movie with corresponding reviews. We can gather data from rotten tomatoes to receive movie ratings before its released. From there, we can average the number of user votes and movie score rating and create a new attribute for "pre-released score". This will add an additional independent variable and may affect our teams project. His accuracy was 76.3%. However, he used a different dataset with different attributes when running the random forest algorithm.

### Felicia Chang

Felicia collected two new attributes "writer rating" and "cast rating" due to our insufficient data attributes. The detail processes are documented above at "Transformation on Writer, Director, and Cast Rating" section. With these two new attributes, she was able to remove the two poorly performed attributes "release date" and "production company". With the newly processed dataset, she then performed a Decision Tree algorithm and a

SVM algorithm and found the results of 74.9% and 76.4% accordingly. The two new attributes she created, “writer rating” and “cast rating”, later prove to help increase our model with higher performance.

### **Jo Jo Lin**

For her individual project, Jo Jo focused on the Naïve Bayes algorithm, which is a classification technique that can assume the independent variables among the predictors by Bayes’ Theorem. New seven independent variables are used are ‘director’, ‘releaseDate’, ‘castRating’, ‘runtime’, ‘productionCompany’, ‘writerRating’ and ‘budget’, and the target variable is ‘vote\_average’. She successfully predicted 378 movies as unsuccessful, and 584 as successful (accuracy of 73.72%). However, she mistakenly 233 movies as successful and 110 movies as unsuccessful.

### **Katie Wang**

Katie used the collected two new attributes “writer rating” and “cast rating and performed random forest algorithm on the new dataset. The accuracy for the random forest is 76.9. Due to small data issue, she used 10-fold cross validation again on the dataset, and the accuracy was dropped to 74.3%. The main reason she guessed was overfitting, the model without 10-fold was picking up the noise and random fluctuations in the data and learned them as concepts.

### **Haozhe Xu**

Haozhe did another data transformation method in our DV method. He changed the DV method from the Binary type into three class character type. Because when he plot the original Rating value, he noticed that its distribution very similar with Normal distribution. Thus, he choose to change our DV variable. Then, he use the random forest and 10 fold cross validation to run our model. For the result, his average of accuracy is 88%, but it also increase our baseline which is 77%. Based on this situation, he discussed with Professor Zhang, he gave the feedback is that we need to use the Binary type in our stage 3, but it is a nice try.

## **Algorithm Executions**

Because of the multicollinearity among our attributes, we understand that Neural Network cannot be performed due to its unique assumption. Therefore, we removed such algorithm. Instead, we went forward to performing Decision Tree, SVM, and Random Forest.

### **Cross-Validation**

Because we only have 4000 records, due to the limit of data, we decided to use the Cross-Validation. Cross-Validation is a resampling procedure used to evaluate machine learning models on a limited data sample. So we use 10 fold- cross validation for our stage 3.

```
 folds <- split(movies, cut(sample(1:nrow(movies)),10))
```

In Cross-Validation, it shuffle the dataset randomly firstly, and Split the dataset into 10 groups. For each unique group, it will take the group as a hold out or test data set, and it will take the remaining groups as a training data set and fit a model on the training set and evaluate it on the test set. At last, it will calculate the average of the accuracy.

We will use our data mining algorithms based on cross-validation.

```
test <- ldply(folds[i], data.frame)
train <- ldply(folds[-i], data.frame)
```

## Decision Tree

In decision tree, before training the model, we just apply the 10 fold- cross-validation firstly. Then we set all the IV variables as predictors variables and put them into the decision tree function (C5.0.default). In the IV variables, 'is\_r\_weekend' and 'is.top.prodcom' are binary variables and 'director','runtime','budget','writer.rating' and 'cast\_rating' are numeric value.

The variables to be used for this algorithm are the same as previous algorithms. They are: 'director', 'runtime', 'is\_r\_weekend', 'is.top.prodcom', 'budget', 'writer.rating' and 'cast\_rating', and the dependant variable is 'is.successful'.

## SVM

In the SVM algorithm, SVM is a discriminative classifier formally defined by a separating hyperplane. And it given labeled training data, the algorithm outputs an optimal hyperplane which categorizes new examples.

Before training the model, we apply the 10-fold cross-validation, and we put all the IV variable into SVM function. In the IV variables, 'is\_r\_weekend' and 'is.top.prodcom' are binary variables and 'director','runtime','budget','writer.rating' and 'cast\_rating' are numeric value.

The variables to be used for this algorithm are the same as previous algorithms. They are: 'director', 'runtime', 'is\_r\_weekend', 'is.top.prodcom', 'budget', 'writer.rating' and 'cast\_rating', and the dependant variable is 'is.successful'.



## Random Forest

The third algorithm we use is Random Forest. Random Forest is a powerful machine learning algorithm. This algorithm creates multiple decision trees and then combining the output generated by each of the decision trees.

Random Forest works on the same underlying principle as Decision Trees. However, it does not select all the data points and variables in each of the trees. It randomly samples data points and variables in each of the tree that it creates and then combines the output at the end. Compared with decision tree algorithm, random forest generally has higher accuracy. Based on this, we choose to run random forest on the newMain.csv dataset.

The variables to be used for this algorithm are the same as previous algorithms. They are: 'director', 'runtime', 'is\_r\_weekend', 'is.top.prodcom', 'budget', 'writer.rating' and 'cast\_rating', and the dependant variable is 'is.successful'.

First, we split the dataset:

```
> summary(TrainSet)
```

director	runtime	is_r_weekend	is.top.prodcom	budget	is.successful	writer.rating	cast_rating
Min. :1.800	Min. : 0	Min. :0.000	Min. :0.0000	Min. : 1	Min. :0.0000	Min. :1.600	Min. :2.100
1st Qu.:6.267	1st Qu.: 93	1st Qu.:0.000	1st Qu.:0.0000	1st Qu.: 7500000	1st Qu.:0.0000	1st Qu.:5.950	1st Qu.:6.095
Median :6.769	Median :103	Median :1.000	Median :0.0000	Median : 20000000	Median :1.0000	Median :6.600	Median :6.414
Mean :6.647	Mean :107	Mean :0.507	Mean :0.4561	Mean : 33620727	Mean :0.5459	Mean :6.503	Mean :6.302
3rd Qu.:7.191	3rd Qu.:117	3rd Qu.:1.000	3rd Qu.:1.0000	3rd Qu.: 44000000	3rd Qu.:1.0000	3rd Qu.:7.179	3rd Qu.:6.686
Max. :8.605	Max. :216	Max. :1.000	Max. :1.0000	Max. :380000000	Max. :1.0000	Max. :8.956	Max. :8.600

```
> summary(TestSet)
```

director	runtime	is_r_weekend	is.top.prodcom	budget	is.successful	writer.rating	cast_rating
Min. :2.100	Min. : 0.0	Min. :0.0000	Min. :0.0000	Min. : 1	Min. :0.0000	Min. :2.100	Min. :2.100
1st Qu.:6.284	1st Qu.: 94.0	1st Qu.:0.0000	1st Qu.:0.0000	1st Qu.: 8000000	1st Qu.:0.0000	1st Qu.:5.995	1st Qu.:6.070
Median :6.759	Median :105.0	Median :1.0000	Median :0.0000	Median : 22000000	Median :1.0000	Median :6.575	Median :6.405
Mean :6.693	Mean :107.9	Mean :0.5057	Mean :0.4701	Mean : 34968423	Mean :0.5563	Mean :6.540	Mean :6.267
3rd Qu.:7.264	3rd Qu.:119.0	3rd Qu.:1.0000	3rd Qu.:1.0000	3rd Qu.: 45000000	3rd Qu.:1.0000	3rd Qu.:7.249	3rd Qu.:6.667
Max. :8.782	Max. :248.0	Max. :1.0000	Max. :1.0000	Max. :425000000	Max. :1.0000	Max. :8.956	Max. :7.900

Then, we will create a Random Forest model with default parameters:

```
model1 <- randomForest(is.successful ~ ., data = TrainSet,  
mtry=2,ntree=500,importance = TRUE)
```

Model1:

```
> model1

Call:
randomForest(formula = is.successful ~ ., data = TrainSet, mtry = 2,      ntree = 500, importance = TRUE)
  Type of random forest: classification
    Number of trees: 500
No. of variables tried at each split: 2

      OOB estimate of  error rate: 25.05%
Confusion matrix:
      0      1 class.error
0 1077  502  0.3179227
1  369 1529  0.1944152
```

To tune the parameter of random forest, we use a for loop to change the value of mtry from 2 to 6, and see at which value of mtry I can get a higher accuracy.

```
# Using For loop to identify the right mtry for model
a=c()
i=5
for (i in 2:6) {
  model3 <- randomForest(is.successful ~ ., data = TrainSet, ntree = 500, mtry = i, importance = TRUE)
  predTest <- predict(model3, TestSet, type = "class")
  a[i-1] = mean(predTest == TestSet$is.successful)
}

a
```

When parameter mtry=2, we get highest accuracy of 76.2%.

```
> # Using For loop to identify the right mtry for model
> a=c()
> i=5
> for (i in 2:6) {
+   model3 <- randomForest(is.successful ~ ., data = TrainSet, ntree = 500, mtry = i, importance = TRUE)
+   predTest <- predict(model3, TestSet, type = "class")
+   a[i-1] = mean(predTest == TestSet$is.successful)
+ }
>
> a
[1] 0.7620690 0.7574713 0.7494253 0.7517241 0.7586207
```

I will use mtry =2 and ntree=500 to predict on our test dataset and test the accuracy.

## Algorithm Results and Interpretation

### Baseline

```
> table(movies$is.successful)
```

```
 0    1  
2313 2917
```

Based on dependent variable distribution, we calculate our baseline is 55.8%

### Stage 1 Result:

Algorithm	Accuracy	Opinion
Naive Bayes	68%	default
SVM	72.1%	Linear kernel, cost=0.1
Decision tree	70.9%	default
Neural networking	70%	4 layers hidden=c(3,2)

### Stage 3 Result:

#### Decision Tree

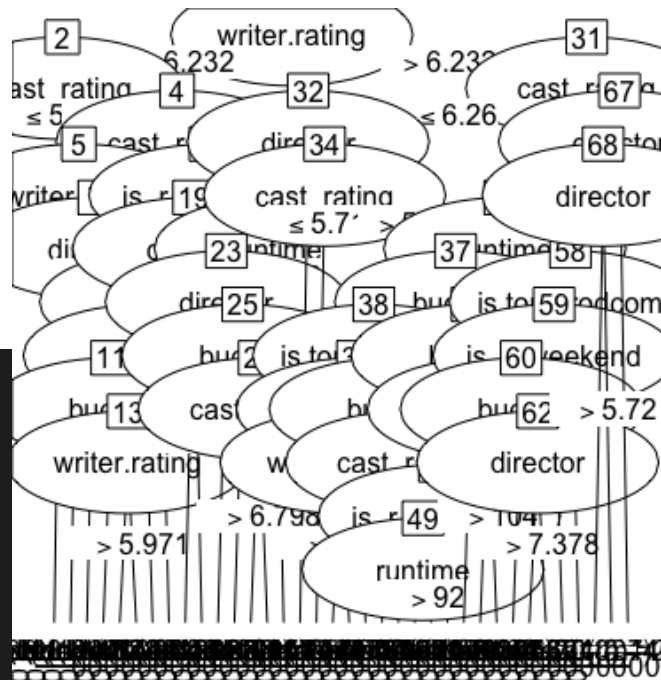
We use the `C5.0.default()` function to train our dataset

```
form <- "is.successful ~ director + runtime + is_r_weekend + is.top.prodcom +budget+writer.rating+cast_rating"  
tmp.model <- C5.0(as.formula(form), train)
```

After training our model, we predict the test value and compute the test set accuracy.

Attribute usage:

```
100.00% writer.rating
100.00% cast_rating
79.49% director
19.68% budget
18.47% runtime
8.06% is.top.prodcom
7.93% is_r_weekend
```



Based on this plot, we can see that writer.rating and cast\_rating and director attribute are the most impactful variable.

```
tmp.predict
  0  1
0 135 63
1  44 189
```

The confusion matrix shows that accuracy heavily on correct prediction on successful. Then we calculate the average accuracy is 74.4%.

## SVM

We use the svm() function to train our dataset

```
svm.model <- svm(is.successful~., data = train, method="C-classification", kernel="linear")
```

After training our model, we predict the test value and compute the test set accuracy and we calculate the average of accuracy is 74.8%.

## Random Forest

We will use random forest model (mtry =2 and ntree=500) to test on our test dataset. The accuracy is around 76.9%

```
> mean(predTest == TestSet$is.successful)
[1] 0.7689655
```

And the confusion matrix is :

```
> table(predTest, TestSet$is.successful)

predTest    0    1
      0 276  91
      1 110 393
```

Next, we will use 10-fold cross validation on our model.

```
numFolds <- trainControl(method = "cv", number = 10)
```

We need to pick the possible values for our cp parameter, using the expand.grid function.

```
cpGrid <- expand.grid(.cp = seq(0.01, 0.5, 0.01))
```

```
train(is.successful ~ ., data = TrainSet, method = "rpart", trControl = numFolds, tuneGrid = cpGrid)
```

The output of this 'train' function shows that accuracy was used to select the optimal model using the largest value and the final value used for the model was cp = 0.01.

Next, we will create a new CART model using the picked parameter:

```
model1CV <- rpart(is.successful ~ ., data = TrainSet, method = "class", cp = 0.01)
```

```
predTestCV <- predict(model1CV, newdata = TestSet, type = "class")
```

```
table(TestSet$is.successful, predTestCV)
```

```
> table(TestSet$is.successful, predTestCV)
      predTestCV
      0      1
0 293    93
1 131   353
```

The accuracy when using 10-fold cross validation is 74.3%, this accuracy is slightly lower than not using 10-fold, which is 76.9%. The reason that 10-fold cross validation accuracy is higher than not using 10-fold may due to overfitting. In this case, the noise or random fluctuations in the training data is picked up and learned as concepts by the model.

## Project Evaluation [Haozhe, Felicia, everyone]

### Conclusion

After performing Random Forest, SVM, and Decision Tree with 10-fold cross validation technique. We get the following results. As shown in the table, the accuracies are all at 74%. The results increased about 2% from the result of Stage 1 (72%). However, the results are more accurate than Stage 1 results because we performed a 10-fold cross validation. Using this technique, we are able to get an average and more accurate results from our small dataset.

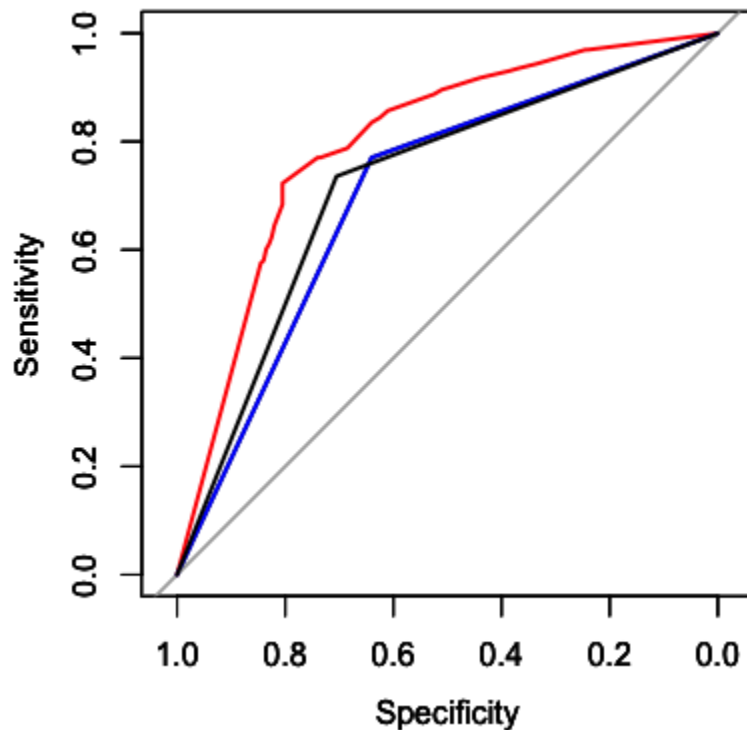
Algorithm	Accuracy	Opinion
Random Forest	74.3%	10-fold cross validation
SVM	74.8%	10-fold cross validation
Decision tree	74.4%	10-fold cross validation

Since the results from the three algorithms are very similar. Other performance measures need to be further calculated to see more details.

### Performance Measures

Algorithm	Accuracy	Precision	Recall	F-score
Random Forest	74.3%	0.7632091	0.8115124	0.7862102
SVM	74.8%	0.6900369	0.8461538	0.7601626
Decision tree	74.4%	0.7546595	0.8226206	0.7865175

## ROC curve



The red line is the decision tree, blue line is SVM and black line is random forest.

According to this table and model evaluation, we can conclude that with the 10-fold cross validation decision tree model, we can get the best accuracy, which can help us to predict movie ratings before being released.

## Final Comments

There are strengths and weaknesses of our project.

The strengths include having strong attributes. Cast and writer attributes contribute to highest usage as well as high accuracy of our models. Similar to cast and writer, director attribute also contribute high performance. This is because a complete historical movie records can be accessed and we are able to obtain the accurate numerical data that truly reflects the roles. Furthermore, 3 attributes use the same base/range as target variable: Writer ratings, director ratings, cast ratings are collected from IMDB movie ratings that are the same range our target variable is calculated from. This would be the major reason why they are the three highly performing attributes.

The weaknesses of our project are small data, small dimensions, and multicollinearity limitation. Because our dataset is small, our models do not have enough “materials” (training data) to properly learn. This prevents our models from improving its performance. Secondly, we keep a small size of attributes (approximately 7-8). Our goal is to feed 5-7 attributes to our models. This leaves us little room to screen through the low performing attributes. In this case, feature selection cannot be of help. Also, because of this limitation, we also cannot remove attributes that have high multicollinearity. This leaves us to only remove the Naive Bayes option and go with other algorithms.

## Reference

[1] Nithin VR, Pranav M, Sarath Babu PB, Lijiya A. Predicting Movie Success Based on IMDb Data, International Journal of Data Mining Techniques and Applications ISSN:2278-2419.

[2] Krushikanth R. Apala, Merin Jose, Supreme Motnam, C.-C. Chan, Kathy J. Liszka, Federico de Gregorio. Prediction of Movies Box Office Performance Using Social Media, Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining.

[3] Nick A., Kevin Y. Movie Rating Prediction. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.87.1964&rep=rep1&type=pdf>

[4] Achal A., Manas P. User rating prediction for movies. Retrieved from <https://pdfs.semanticscholar.org/e7ae/2379489570302b28f396084b368be64db4ca.pdf>