Data Collection:

Our training data was gathered from two main sources: The Movie Database and IMDB. The bulk of the data came from TMDB, compiling a dataset of 8050 movies with attributes like budget, release date, popularity score, TMDB rating, overview and tagline (for potential sentiment analysis), and revenue (class attribute). For our query to TMDB, we used the following criteria: the movie had to make money (some did have $0 revenue) and either had to have a popularity score of at least 5 for movies that came out in the last 30 years, or had to have a popularity score of at least 2 for movies that came out in the last ten years. We decided on these criteria to ensure relevant training data, as we gathered only the most notable older movies and a wider range of newer movies which we hypothesized would be more similar to a current movie release. Using the IMDB API, we were able to easily add to our dataset, gaining another rating metric. We also merged the IMDB rating average and count values using the common attribute of IMDB ID to add two more attributes to our TMDB query. These data-wrangling processes were quite successful, resulting in only a slightly smaller dataset of 8026 movies. After adding the IMDB ratings and running preliminary models, we decided to add a genre attribute to increase predictiveness. This required another query to TMDB, which produced a column of genre ids to be processed into useful data. From there, the genres were one-hot encoded into 18 further attributes, one for each common genre reported by TMDB. This will be discussed in greater depth later.

Data Analysis:

We explored many options for data analysis with a wide range of association learning, regression, and classification models. Such a variety of models required careful pre-processing to get our data in the proper formats.

Pre-Processing:

The first thing we addressed with pre-processing was getting attributes in their optimal forms for performing data analysis. This included changing the attribute ‘collection’ which gave the name of the movie collection that the movie belonged to or an empty cell for solo movies into a binary attribute with 1’s for movies that belonged to a collection and 0’s for movies that didn’t. We also used one-hot encoding to make the genre attribute more predictive. This made an individual column for each genre with 1 if the movie instance in that row is that genre, and 0 if not. These

genres were action, animated, drama, family, fantasy, history, comedy, war, crime, documentary, music, mystery, romance, science fiction, horror, thriller, Western, and adventure. Lastly, we split the release date column into three separate month, day, and year columns and also converted the budget and revenue columns from strings (Excel converted them into $ amounts) into integers. The rest of our pre-processing was specific to the models that we chose. Some models required normalization or scaling, such as the Support Vector Regression model. Similarly, our classification models required that the revenue column be discretized. We decided to use a 4 category split: the bottom 30% being 'low', the second 30% being 'average', the third 30% being 'high', and the top 10% being 'blockbuster'. Since we ran out of time to perform sentiment analysis, the title, overview, and tagline attributes were removed for all of our analyses.

Models:

In terms of regression, we ran four initial models that we hypothesized would be the most predictive: Multiple Linear Regression, KNN Regression, Support Vector Regression, and Regression Tree. Some of our outcome metrics were difficult to interpret with R^2 and MAE values contradicting each other, but in general, we found that linear regression and regression tree tended to be the best regression models. While these outcomes were encouraging, we were eager to find even better results. We decided to implement XGBoost, a gradient boosting regression tree algorithm. Seeing the improvement in R^2 and MAE, we decided this would be sufficient for our regression analysis.

The use of our feature selection code continued in the classification analysis. Our final list of classification models is as follows: KNN, Naive Bayes, Decision Tree, Random Forest, XGBoost, and Neural Network. Similarly to our regression models, this list originally did not include XGBoost. We initially set out to build an acceptable SVM model, but after much trial and error, we could not get accuracy scores up to our standards so we decided to scrap this model. Seeing as the XGBoost regression model was built on an algorithm similar to decision trees and random forest and had great outcomes, we figured this could be a viable replacement.

Feature Selection:

Each model was implemented in Python using a utility function which contains the operations of fitting the training data, predicting based on the test data, and returning a tuple of either MAE and R^2 or accuracy and confusion matrix. For some models, namely the KNN and

tree models, a layer of experimentation was added. We wanted to determine which neighborhood or maximum tree depth would create the best model based on the current set of predictor attributes. To implement this, 29 different models from 2 to 30 were run, and only the neighborhood sizes or max tree depths that returned the best accuracy metrics were returned. When feature selection began, each of these utility functions was run once on the new set of predictors, and once on the initial set of predictors. The original and best feature-selected evaluative tuples were output in separate dictionaries in the form {"Algorithm Name" : ["original" : original statistic, "feature selected" : F.S. statistic], … } for all the algorithms.

Results

Regression:

- Multiple Linear Regression

Statistic	Original	Feature Selected
MAE	42072432.05	41754665.19
R^2	0.75064	0.74995

- KNN Regression

Statistic	Original	Feature Selected
MAE	39316747.34	38003848.04
R^2	0.64595	0.67100

- Support Vector Regression

Statistic	Original	Feature Selected
MAE	60791809.00	60490055.92
R^2	.7136	.7134

- Regression Tree

Statistic	Original	Feature Selected
MAE	32167484.73	31457472.86
R^2	0.69967	0.73081

- XGBoost

Statistic	Original	Feature Selected
MAE	32862658.56	32283606.03
R^2	0.75950	0.77850

Classification (Test set: 2,651):

- **KNN**
    - Original Confusion Matrix: 59.343%

	Low	Average	High	Blockbuster
Low	617	158	13	0
Average	342	301	150	0
High	84	194	513	31
Blockbuster	7	3	95	141

    - Feature Selected Confusion Matrix: 59.834%

	Low	Average	High	Blockbuster
Low	621	151	16	0
Average	338	310	145	0

High	85	186	521	30
Blockbuster	7	3	103	133

- **Naive Bayes**
    - Original Confusion Matrix: 55.153%

	Low	Average	High	Blockbuster
Low	730	55	3	0
Average	510	229	49	5
High	160	263	349	50
Blockbuster	6	5	82	153

    - Feature Selected Confusion Matrix: 55.153%

	Low	Average	High	Blockbuster
Low	730	55	3	0
Average	510	229	49	5
High	160	263	349	50
Blockbuster	6	5	82	153

- **Decision Tree**
    - Original Confusion Matrix: 60.929%

	Low	Average	High	Blockbuster
Low	589	183	16	0
Average	287	362	143	1
High	48	225	500	49

Blockbuster	5	4	74	163

- Feature Selected Confusion Matrix: 62.967%

	Low	Average	High	Blockbuster
Low	571	195	22	0
Average	253	373	167	0
High	41	179	557	45
Blockbuster	5	4	70	167

- **Random Forest**
  - Original Confusion Matrix: 66.402%

	Low	Average	High	Blockbuster
Low	616	158	14	0
Average	228	425	137	3
High	28	198	551	45
Blockbuster	3	4	72	167

  - Feature Selected Confusion Matrix: 66.818%

	Low	Average	High	Blockbuster
Low	618	156	14	0
Average	221	428	143	1
High	28	191	561	42
Blockbuster	3	4	76	163

- **XGBoost**
  - Original Confusion Matrix: 66.969%

	Low	Average	High	Blockbuster
Low	611	157	20	0
Average	223	418	151	1
High	29	180	569	44
Blockbuster	4	1	65	176

- Feature Selected Confusion Matrix: 67.157%

	Low	Average	High	Blockbuster
Low	604	163	21	0
Average	220	427	145	1
High	29	177	575	41
Blockbuster	4	1	68	173

- **Neural Net**
  - Original: 46.131%

	Low	Average	High	Blockbuster
Low	460	10	317	1
Average	232	11	549	1
High	68	2	750	2
Blockbuster	7	0	238	1

- Feature Selected: 46.206%

	Low	Average	High	Blockbuster

Low	473	0	315	0
Average	243	0	550	0
High	71	0	751	0
Blockbuster	8	0	238	0

Discussion:

Using our feature selection code and our Tableau visualizations, we gained valuable insight as to which attributes are the most predictive in determining success of a movie. Some of our most predictive attributes were rather intuitive and did not come as a surprise such as the budget and whether or not the movie belonged to a collection. However, as our feature selection code determined, neither of these were our most predictive attribute (although budget was #2). Surprisingly, the number of rating votes from each of our data sources were 2 of the top 3 predictors with tmdb_vote_count being #1 and imdb_vote_count being #3. While we had initially reasoned that these website ratings would be important attributes, we were expecting that the actual scores of the ratings would be the valuable piece. However, in hindsight, it makes sense that the vote count would be predictive, because it indicates how many people both saw the movie and felt compelled enough to evaluate the movie online. In this sense, the vote count acts as a very relevant proxy for popularity and even revenue, indirectly. Beyond vote counts, budget, and collections, the next most predictive attribute was genre. Since we used one-hot encoding to give each genre its own attribute, we were able to compare the value of each genre. What we found was that the best-performing genres were those that fit superhero and other blockbuster action movies. We found that the 3 most predictive genres were adventure, action, and fantasy. Also somewhat surprisingly, we found that comedy and animated genres were not very predictive of a movie's revenue. Our best explanation for this is that comedy and animated movies have high variability, with the highest performers topping the charts such as the Frozen movies, and the lowest performers barely scratching the surface of relevance.

After performing a range of regression methods learned in class, we had decent results, but seeing as plain linear regression was arguably our best model, we were not satisfied. We found a package called xgboost, a python implementation for "extreme gradient boost", an

algorithm similar to SciKit Learn's 'DecisionTreeRegressor' except using gradient boosting decision trees. Fortunately, this model showed significant improvement, reaching our highest R^2 value and nearly our lowest MAE. One more minor result that came as a surprise to us was that linear regression tended to be a better model than SVR with a linear kernel. While the R^2 values are comparable, the SVR model has over ⅔ higher MAE than the linear regression. This result suggests overfitting in the SVR model, although it is interesting to note that this overfitting did not lead to having a higher R^2 than linear regression.

One of the most interesting patterns that we noticed with our classification models was that while accuracy varied from model to model, the overall trends of the confusion matrices were very similar. In other words, the classification models tended to struggle with and succeed on the same predictions. Percentage wise, the models tended to have the hardest time differentiating between 'low' and 'average' movies and 'high' and 'blockbuster' movies. This did not come as a surprise, as there inevitably are plenty of movies near the borderlines of our revenue categories, and the hardest part of a challenge like this is predicting the close calls. However, considering the variability and limited information in the data we collected, we were mostly pleased with the results, especially when testing on the set of movies currently playing.

Future Improvements:

The first potential improvement to this project that comes to mind is the use of sentiment analysis. While we collected overviews and taglines for each movie, we ran out of time to find a way to add this data to our predictions. Additionally, there are several sources (including our original sources) that could provide more data. Some attributes that come to mind are top-billed cast members, official trailer views on YouTube, social media presence, etc. As far as changes to our actual models, our neural network could use some serious tuning and we believe we could find a regression model to improve on our linear regression, regression tree, and xgboost regression models.