Capstone Project
Machine Learning Engineer Nanodegree
John Rangel
April 28th, 2017

Predicting Video Game Sales

Definition

Project Overview

Since the early 1970s, the video game industry has been consistently growing every year with the potential for huge sales increasing with it. Often times, it can be very hard to know how well a game will sell when released and which regions it will be most popular. Even highly rated video games can sell very little for what they offer while lower rated games can sell quite a bit. Regardless, it can be extremely useful if fairly accurate predictions can be made on how much sales a game will produce. Previous research has been done such as on the European market and showed that with the right formulas, sales predictions can be fairly accurate[1]. This research however did mostly statistical analysis rather than machine learning techniques.

This project uses a video games sales and ratings dataset available on Kaggle[2]. This dataset provides the models with ~17000 different titles with potentially useful features such as platform, title, metacritic ratings, genre, etc, which I can then compare to the sales of different regions. This dataset however, is actually based on a previous Kaggle dataset that was lacking some very useful critic and user score information from metacritic[3]. The core data from the original dataset was pulled from VGChartz.[4].

For this project, 5 different machine learning models are created and tuned using a subset of the dataset. Once that is done, the models can predict video game sale prices from different regions. The performance of all 5 models plus a benchmark model are then compared graphically to be visually clear which models are doing the best job at predictions.

Problem Statement

The problem is that the sales of a newly released game are unknown to most people but I believe that by selecting the most relevant and useful features and then using supervised learning on those features, the total sales of a game can be predicted. Thus. this is both a feature selection and a regression problem. Feature selection is needed because there are too many different features that could have influence on the

generated sales and the actual predictions are a regression problem because we want to understand the relationship between the relevant features and the sales of the game.

Evaluation Metrics

The primary evaluation metric that will be used to quantify the performance of the benchmark and solution model is going to be mean squared error or MSE. MSE is a measure of the mean of the squares of the errors. It is the preferred choice because the estimations aren't expected to be entirely spot on. The sales predictions of the models are intended to be as close as possible to the real values but it is very much okay for them to be off by seemingly large amounts. MSE works perfect for this since it will penalize outliers more than the smaller error values. This is because as the error values get larger, the squares of the errors increases dramatically. The similar alternative known as mean absolute error is less preferred since it does not square the error and treats all errors equally.

Analysis

Data Exploration

The video game sales with ratings database is pretty straightforward.
Potential features are as follows:

- *Name*- name of the video game. (String)
- *Platform*- device the game is played on. (String)
- *Year_of_Release*- year the game was released. (String)
- *Genre*- type of game(sports, action, shooter, etc). (String)
- *Publisher*- publisher of the game. (String)
- *Developer*- developer of the game. (String)
- *Rating*- ESRB rating of the game(E, T, M, etc). (String)
- *Critic_Score*- average review score from every Metacritic staff that reviewed the game with a range of 0-100. (Float)
- *Critic_Count*- the amount of critics that reviewed the game (Integer)
- *User_Score*- average review score from every Metacritic user that reviewed the game with a range of 0-10. (Float)
- *User_Count*- the amount of users that reviewed the game (Integer)

Potential targets are as follows:

- Global_Sales- total worldwide sales(in millions) (Float)
- NA_Sales- total North America sales(in millions) (Float)
- EU_Sales- total Europe sales(in millions) (Float)
- JP_Sales- total Japan sales(in millions) (Float)

- Other_Sales- total sales for the rest of the world(in millions) (Float)

For the purposes of this project, Name and Year_of_Release, Critic_Count, and User_Count will not be used as features. Pretty much every video game name is going to be different and won't be unique enough to benefit our models. The year of release is also a big problem for future data. The data was pulled in the middle of 2016 for example and will be missing half of its data. A model trained on this data also won't have any experience handling 2017 and beyond data. Additionally, the amount of users or critics that reviewed a game will also be omitted since the older a game is, the more reviews it will have. Other_Sales will also be ignored since it is far to vague compared to the other targets. The majority of the remaining inputs or features in this dataset will be very useful for solving the problem at hand and are similar to previous research in this domain.

There is also one significant problem with the dataset. The critic and user scores information are incomplete for most of the dataset. The actual amount of complete data the models will have available is 6825. This also brings a concern about the unique values of some of the String features. To understand this a little bit better, the unique values for the remaining features are as follows:

- Platform: 17
- Genre: 12
- Publisher: 262
- Developer: 1289
- Rating: 7

Based on the information above, developer will be omitted from the set of features as well. There are just too many possible values for a now relatively sparse dataset. It presents even more of a problem when looking at the amount of relevant rows based on if a video game was actually sold in a region(sales does not equal 0.0):

- Global Sales Rows: 6825
- NA Sales Rows: 6282
- EU Sales Rows: 5900
- JP Sales Rows: 2012

Exploratory Visualization

Below are some interesting charts and graphs to understand the dataset better.

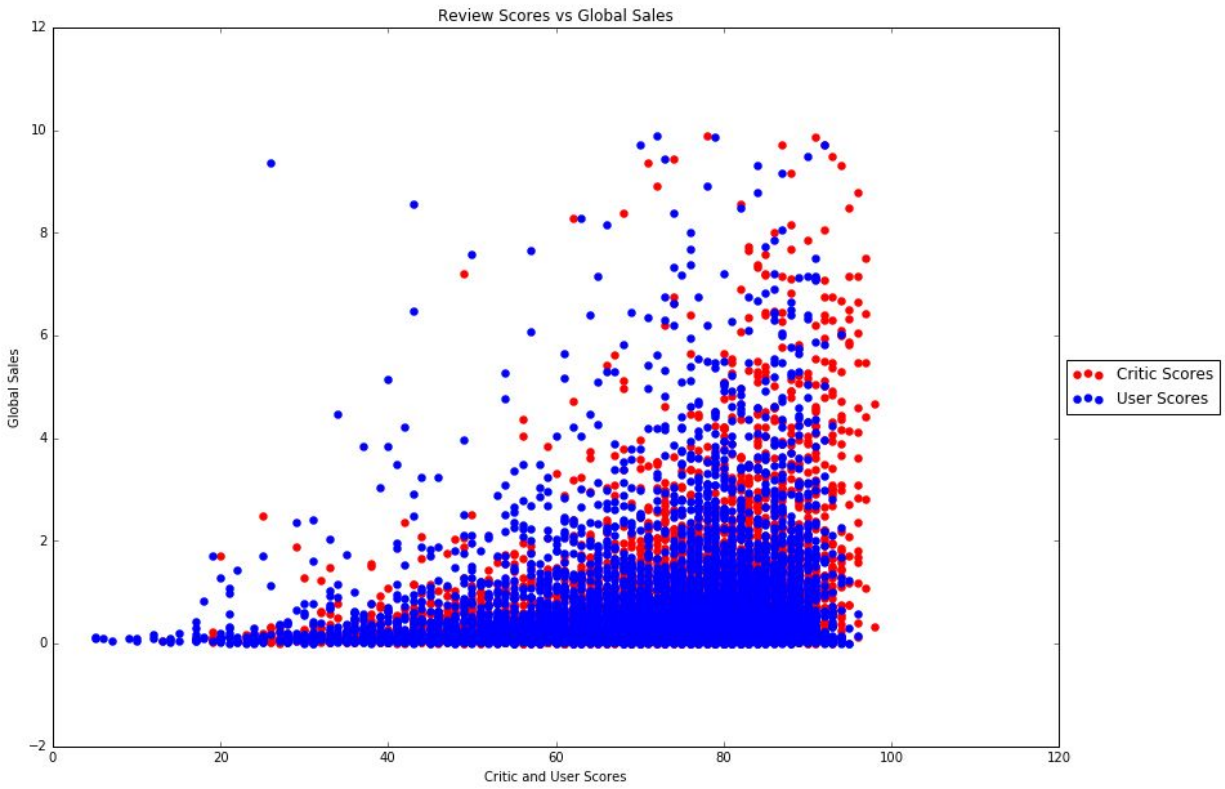

Figure 1 *A graphical representation of critic/user scores vs global sales. User_Scores was scaled to match Critic_Scores to have a better comparison of the two features. Based on the graph, there seems to be a positive correlation of review scores to Global Sales. However, the variance for each score is very large and will likely have a lot of error if only using user/critic scores. The two review scores though are similar enough that averaging the two into a singular score can be a very effective way of reducing dimensionality and make the scores more consistent.*

Platform Average Sales Changes:

	Wii	X360	PS4
Global_Sales	1.375741	0.994522	1.016862
NA_Sales	0.715887	0.621713	0.352134
EU_Sales	0.426159	0.273112	0.467699
JP_Sales	0.109332	0.011760	0.038870

Genre Average Sales Changes:

	Shooter	Action	Strategy
Global_Sales	0.945000	0.738135	0.260712
NA_Sales	0.519398	0.362718	0.124532
EU_Sales	0.302222	0.237902	0.094419
JP_Sales	0.021493	0.046730	0.016255

Publisher Average Sales Changes:

	Nintendo	Activision	Sega
Global_Sales	2.919210	1.088902	0.522606
NA_Sales	1.276151	0.623618	0.246444
EU_Sales	0.831993	0.338496	0.172711
JP_Sales	0.597251	0.010793	0.045317

Developer Average Sales Changes:

	Nintendo	Konami	Rockstar North
Global_Sales	7.792647	0.582526	8.533571
NA_Sales	3.392206	0.180105	3.945000
EU_Sales	2.454118	0.217474	2.726429
JP_Sales	1.306176	0.091368	0.205714

Rating Average Sales Changes:

	E	T	M
Global_Sales	0.941162	0.579436	0.994962
NA_Sales	0.473593	0.292036	0.505659
EU_Sales	0.283756	0.166899	0.325939
JP_Sales	0.091580	0.060858	0.043461

Figure 2 *The charts above are the result of an experiment to see how much of an effect each String feature would have on the average sales. For example, the platform chart was calculated by filtering for each parameter("Wii", "X360", and "PS4") and averaging the sales that made it through the filter. For the most part, all of our remaining features seem to have a relatively strong impact on the average sales so it will be best to keep them in our total list of features with the exception of developer for reasons stated in the previous section.*

Algorithms and Techniques

Since there are so many different machine learning methods that could be used to solve this problem, five models were chosen to be tested against the benchmark and each other. In the end, we want to conclude the best machine learning method of all those chosen so all steps in the figure below will be exactly the same except for which model we use.

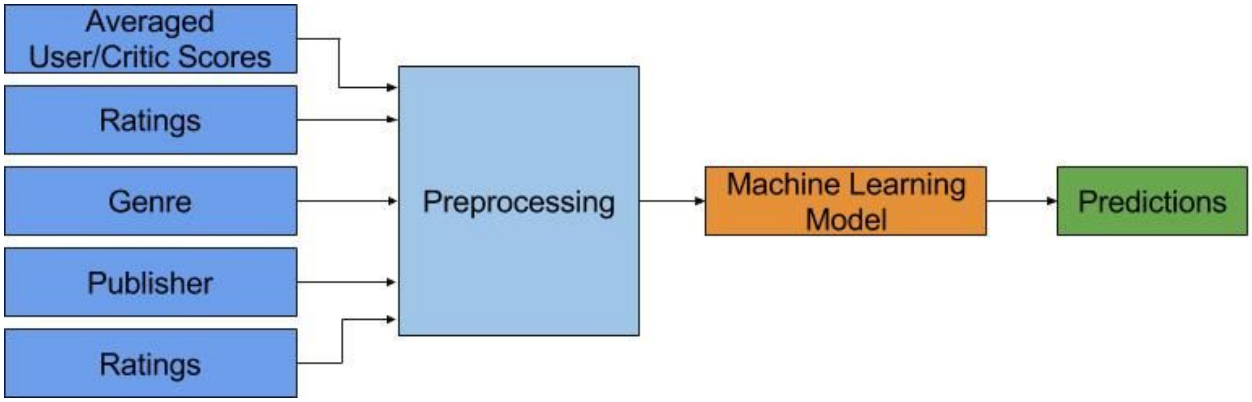

Figure 3 *Graphical representation of the process of taking features, preprocessing data, training it on the machine learning model, and then making predictions.*

Machine Learning Models[5]:

- Support Vector Regression or SVR
    - This method is very effective at handling high dimensions of data and is very versatile thanks to the different kernel functions that can be used. Due to the nonlinear nature of the data, the model will use a Radial Bias Function kernel.
    - Parameters that will need tuning
        - C- Penalty parameter.
        - Gamma- Kernel coefficient
- Decision Tree Regressor
    - One of the most simplest and yet still effective methods of machine learning. Might prove to be effective at handling the large spread of sales data. Decision Trees are prone to overfitting but that problem will be solved later.
    - Parameters that will need tuning
        - *max_depth*- the maximum depth of the tree
- K Nearest Neighbors Regression
    - Looks at k training points around new data point to predict the target value. This non generalizing method might be very useful if it turns out that the data can't be generalized very well at all.
    - Parameters that will need tuning
        - *n_neighbors*- number of neighbors or k
        - *weights*- uniform or distance (how nearby points will be weighted)
- Gradient Boosting Regression Trees
    - A very effective method of boosting that has strong predictive power, robustness to outliers, and is natural at handling mixed data types. These

characteristics are the reason I chose it and why I think it would perform the best.
    - Parameters that will need tuning
        - *n_estimators*- number of boosting stages to perform
        - *learning_rate*- learning rate that affects the contribution of each tree
        - *max_depth*- maximum depth of the individual regression estimators
- Ridge Regression
    - A linear model similar to linear regression but imposes a penalty on the size of coefficients. I don't expect this to perform that much better than the benchmark linear regression model but figured it would serve as a good comparison.
    - Parameters that will need tuning
        - *alpha*- regularization strength that reduces the variance of the estimates.
        - *tol*- tolerance that specifies the precision of the solution

Benchmark Model

The benchmark model used for this project is a simple linear regression model that takes a single feature composed of the average of user and critic scores and with a target being sale prices for each region. Since each target will have different amount of data available, a different model was created and trained for each. The MSE results for each region is shown below:

Global Sales	2.0920575008331856
NA Sales	0.64977312017815836
EU Sales	0.3250598202291653
JP Sales	0.24721697296277534

Figure 4 *Benchmark Model Results*

Methodology

Data Preprocessing

Features and targets for the data were preprocessed with the following steps:

1. Copying the dataset into four copies representing each target region and also filtering out any values in which the target region is "0.0" meaning the game wasn't sold there.

2. For each of the four copies, the following steps are done:
    a. User and Critic scores are averaged into one feature.
    b. All the unique values for the String features were assigned a key and then the features themselves were replaced in accordance to the new key system. For example, a feature with values [‘test1’, ‘test2’, ‘test1’, ‘test3’] would turn into [0, 1, 0, 2]. This is important since our models need numerical data as the input features.
    c. All the new features created by the above steps are then put into an array that the machine learning models will be able to use.

Implementation

My first attempt at implementing the model was fairly simple.

1. I split the data for each region into training and testing data using a constant randomized state. I didn’t want the splitting of the data to be completely random since I still need to compare the different models and having different train/test data would skew results.
2. After that, the model for each region was set up with manually entered parameters and fitted to training data.
3. The models were then given the testing data to make sales predictions for each region. The sales predictions were then compared to the actual sales and the mean squared error was calculated for each model.

While this works and provided decent results, it’s very likely that a more optimal model was not used. In order to fix this, I needed to use a technique called grid search in which I create a list of values to be tested for each parameter. In order to streamline the process, I created functions that accept training data as the input and then return an optimized model. This model is created by training and testing with every combination of the parameter lists and then scored using mean squared error. The model with the lowest MSE would then be returned as the optimal model. This was done for each of the 5 models for each of the four regions/targets.The results were then saved to be used for later analysis.

Refinement

As mentioned in the implementation station above, the final result had major differences compared to the initial solution. The initial solution was for the most part hard coded and didn’t test different values to get the model with the lowest error possible. Originally, there were only two machine learning algorithms as well. Only Support Vector Regression and Decision Tree Regression were going to be tested until more research was done on other methods.

Even after the final solution of 5 models with grid search parameter tuning, the best parameters were not being tuned and not enough values were being tested. This was expanded for the final solution and optimal models are now being reported. The actual process of refining the solutions was greatly helped by having individual functions for handling the tuning and creation of the different machine learning models. Although each of the functions are similar, they are entirely customizable to be appropriate to their respective models.

Results

Model Evaluation and Validation

For each region, the optimal parameters that were chosen to be tuned from the Algorithms and Techniques section can be seen in the chart below:

	SVR	Decision Trees	K Nearest Neighbors	Gradient Boosting Trees	Ridge
Global	C:10 gamma: 0.25	max_depth: 2	N_neighbors: 10 Weights: 'distance'	N_estimators: 100 Learning_rate: 0.1 Max_depth: 3	tol: 0.001 alpha: 1.0
NA	C:10 gamma: 0.5	max_depth: 2	N_neighbors: 7 Weights: 'uniform'	N_estimators: 100 Learning_rate: 0.1 Max_depth: 3	tol: 0.001 alpha: 1.0
EU	C:10 gamma: 0.5	max_depth: 4	N_neighbors: 10 Weights: 'distance'	N_estimators: 100 Learning_rate: 0.1 Max_depth: 3	tol: 0.001 alpha: 1.0
JP	C:1 gamma: 0.1	max_depth: 2	N_neighbors: 10 Weights: 'uniform'	N_estimators: 100 Learning_rate: 0.1 Max_depth: 3	tol: 0.001 alpha: 1.0

Figure 5 *The final weights for each of the models for each region*

Based on the different final parameters for models such as SVR, Decision Trees, and K Nearest Neighbors, I would conclude that parameter tuning was successful. The fact that different parameters were needed for the optimal model for different regions shows that manually setting a constant parameter would have led to unoptimized model. The tuning of these parameters essentially lead to an increase in robustness of all the models.

Justification

Using the final results of each machine learning model, I calculated the percentage improvements against the benchmark model performance. The chart below shows all the MSE improvements and deficiencies compared to the benchmark MSE results.

	SVR	Decision Trees	K Nearest Neighbors	Gradient Boosting Trees	Ridge
Global	16.38%	1.76%	9.56%	19.10%	4.80%
NA	14.01%	3.56%	13.37%	23.65%	3.91%
EU	-30.13%	-25.22%	-19.78%	-2.65%	3.31%
JP	18.01%	6.27%	20.55%	31.50%	3.43%

Figure 6 *The percentage improvements of the machine learning models vs the benchmark.*

Looking at the chart, there are some definite improvements in certain aspects of the model and also some drawbacks. All the models seemed to improve a little bit for predicting Global, NA, and JP sales compared to the benchmark but across the board, EU Sales predictions is either far worse than the benchmark or barely meeting it. If I had to pick one model to go with for predicting video game sales though, i would definitely be using Gradient Boosting Regression Trees. It outperformed all the other models for Global, NA, and JP sales and is only slightly worse than the benchmark for EU. Gradient Boosting doesn't even have to be used for EU predictions though. If we wanted to specifically target the EU, ridge regression would be the optimal choice.

Is this problem solved? I think some serious progress was made on more accurate predicting sales but there is still a lot of room for improvement before the problem can be considered solved.

Conclusion

Free-Form Visualization

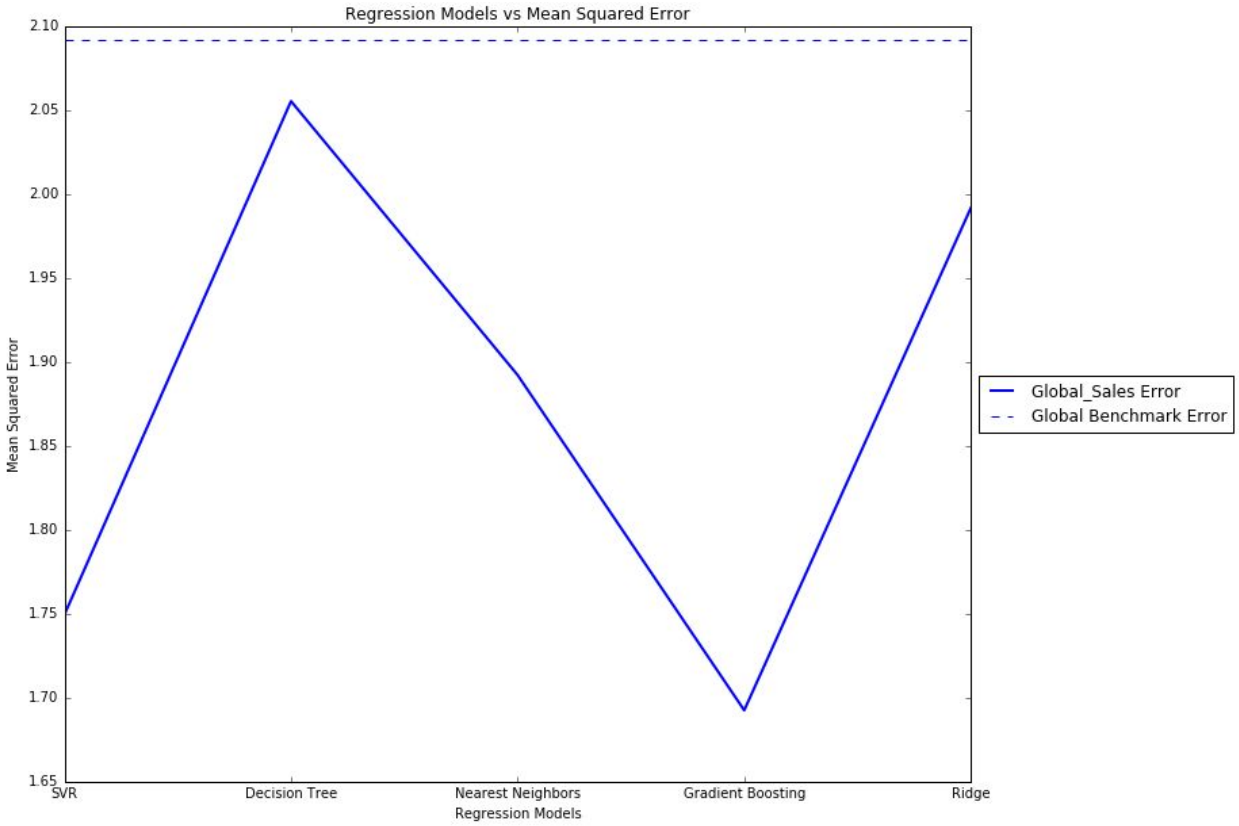

Figure 7 *A graph of Regression Models vs MSE of Global Sales. According to the graph, SVR and Gradient Boosting far outperform the other models at predicting Global Sales. However, all the models are able to outperform the benchmark score. However, the error score for all models including the benchmark are terrible compared to the other scores from Figure 8. All of the models are insufficient at predicting Global Sales.*

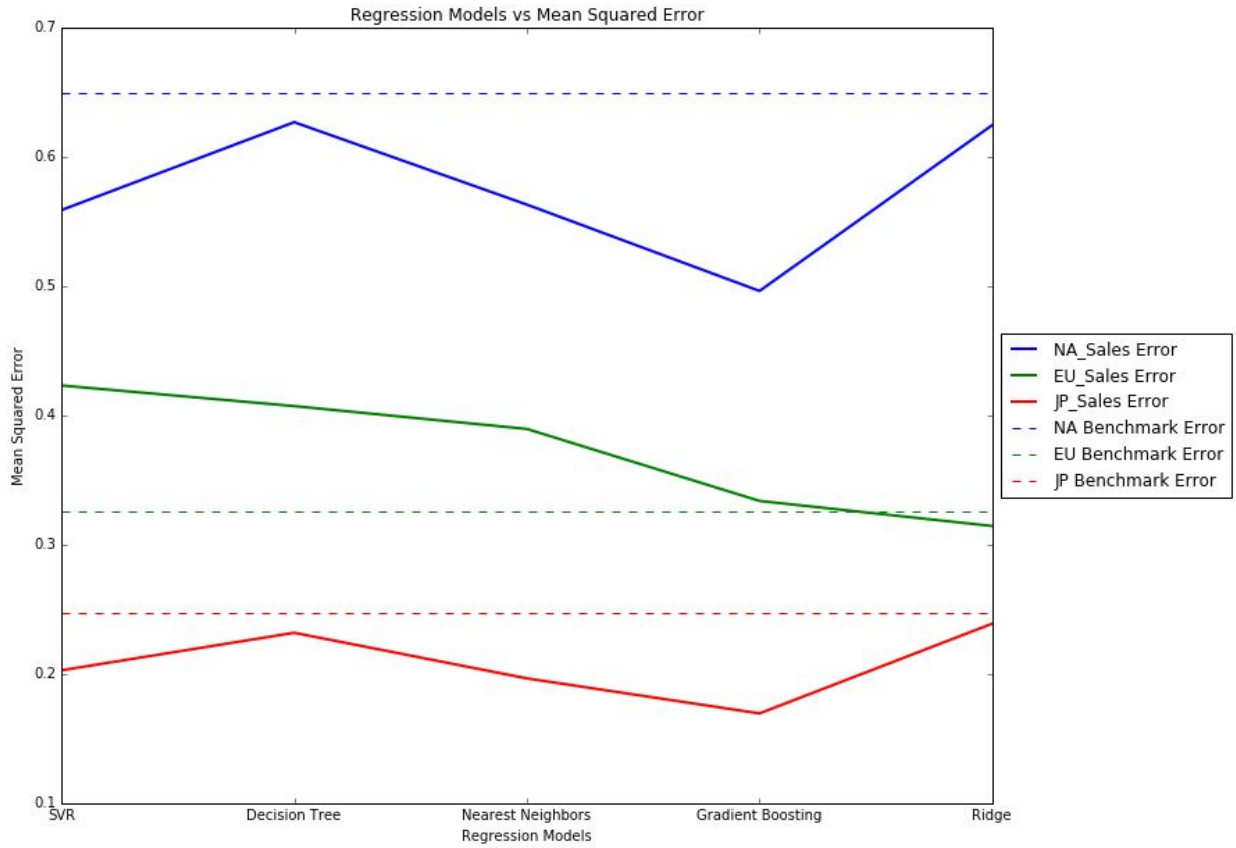

Figure 8 *Regression Models vs MSE of NA, EU, and JP Sales. NA Sales overall are better compared to Global Sales with SVR and Gradient Boosting performing the best. For EU Sales, we originally thought the models were terrible for predictions but that's only because the benchmark itself did a pretty good job itself. JP Sales predictions were absolutely phenomenal across the board. Gradient Boosting is again outperforming the others but the difference is minimal.*

Reflection

The steps needed to complete this project are as follows:

1. A problem was identified and relevant dataset was found
2. The data was analyzed to find the most useful features
3. The data was preprocessed to be used by different machine learning models.
4. A simple linear regression benchmark was created and results recorded.
5. For each target sales region, a set of machine learning models was trained while grid searching for optimal parameters.
6. Each machine model was then scored via MSE to its predictions.

The most difficult part of this project was definitely preprocessing all the data to be useful for all of the machine learning models. There were so many rows in the dataset that were incomplete or that had no impact on the target region. It was also tough

learning new regression techniques that I maybe knew the theory of but never applied such as Gradient Boosting or boosting in general. I also thought it was very interesting how well the benchmark model did in comparison to the machine learning models. It was able to keep up with and in the case of Europe, actually outperformed most of the models. I also didn't expect the spread of user/critic scores to video game sales to be as spread out as it was and yet there was still a very obvious correlation.

Improvement

There are definitely a lot of improvements that can be made to potentially increase the MSE of the models. First off, I think having so many rows missing user/critic scores really hurt the robustness of the model. Although the models were tuned the best they could, not having enough training and testing data probably hurt the performance quite a bit. I also think having more features that weren't provided in the dataset can have drastic impacts on the model. For example, the budget of the game, years of experience of the game director, and marketing budget would probably be extremely useful in sales predictions. All of the string features were helpful but user/critic scores seemed to really drive the majority of the models. Having more directly correlated features like the above examples could have drastic impacts in reducing the total error.

As far as improving the models themselves, I would definitely include neural networks as a future model to be trained and tested. I would also be interested in converting the problem into a classification project such as "Will this game sell a million units". I think that would open up a lot of classification models that could be very accurate.

References:

1. Beaujon, Walter Steven. "Predicting Video Game Sales in the European Market." (n.d.): n. pag. Web. <https://www.few.vu.nl/nl/Images/werkstuk-beaujon_tcm243-264134.pdf>.
2. Smith, Gregory. "Video Game Sales." Video Game Sales | Kaggle. Kaggle, n.d. Web. 20 Apr. 2017. <https://www.kaggle.com/gregorut/videogamesales>.
3. Kirubi, Rush. "Video Game Sales With Ratings." Video Game Sales with Ratings | Kaggle. Kaggle, n.d. Web. 20 Apr. 2017. <https://www.kaggle.com/rush4ratio/video-game-sales-with-ratings>.
4. "Video Game Charts, Game Sales, Top Sellers, Game Data." VGChartz. N.p., n.d. Web. 20 Apr. 2017. <http://www.vgchartz.com/>.
5. "1. Supervised learning¶." 1. Supervised learning — scikit-learn 0.18.1 documentation. Scikit learn, n.d. Web. 30 Apr. 2017. <http://scikit-learn.org/stable/supervised_learning.html#supervised-learning>.