Predicting Video Game Sales

Capstone Project "Machine Learning Engineer"

Oliver Tacke
March 14th, 2017

I. Definition

Project Overview

One of the first video games that I have ever played in my life was Munchkin. It was released in 1981 when the video game industry was still in its infancy. Today, it is a multi-billion dollar business. In 2014 in the U.S. alone, 155 million people played video games (cmp. Entertainment Software Association, 2015, p. 2). In total, they spent 15.4 billion US dollars (cmp. Entertainment Software Association, 2015, p. 12).

In an industry, success is not only measured in good critics, but the amount of money earned or units sold. In order to make decisions about future productions, publishers may want to predict the sales figures which they can expect after a new game has been released. Those decisions could possibly be based on historical data and a suitable regression model. For instance, we could hypothesize that good scores in reviews correlate positively with high sales figures. Also, we could assume, that those reviews could help us to project future sales. In fact, the general plausibility of this approach has been investigated and proven for the movie picture industry a decade ago: "Online movie reviews are available in large numbers within hours of a new movie's theatrical release. Their use, thus, allows the generation of reliable forecasts much sooner than before." (Dellarocas, Zhang & Awad, 2007, p. 39). For the video game industry, Beaujon (2012) developed a third-degree polynomial formula for predicting sales from historical data – basically manually using a spreadsheet.

In this report, I present a machine learning approach to this problem that uses more features and more data. It is based on a dataset about video game sales that uses data scraped from VGChartz (Video Game Sales with Ratings). It contains information about games, including name, platform, year of release, genre, and sales figures for several regions. This dataset has been extended with several features from Metacritic, adding e.g. quality ratings from metacritic's staff and from users, the amount of reviews, and also adding age/content ratings from the entertainment software rating board. In total, there are more that 6.000 complete cases.

Problem Statement

The earlier a video game company knows how many copies it can expect to sell of a game, the earlier it can estimate the revenue and the earlier it knows whether the game will be a financial success or not. For example, a decision about starting the production of a sequel or the production of a port to a different platform might depend on this information. In consequence, early knowledge about the sales figures can speed up the decision process.

The solution to the problem will be a regression model that can be fed with the variables that have been mentioned in the previous section, and it will output a prediction of sales figures for a video game.

Based on the problem definition above, the dataset will be analyzed and prepared. First, the data quality will be checked on a syntactical level. For example, columns that are expected to contain numbers should not contains strings. After that step, it is crucial to understand the data before experimenting with algorithms. Data will be checked for plausibility, because some feature values might be off the chart for whatever reason, etc. Also, by visualizing the data in different ways, we will get some more insight into the problem.

Subsequently, suitable algorithms will be selected. There are different approaches that might be appropriate for building a regression model and should be considered. Although quite a lot algorithms might suit the problem, the right choice depends on different criteria, such as efficiency and scalability. Since we are dealing with more than 50 but less than 100.000 samples for predicting a quantity, the Scikit-Learn cheat-sheet suggests to consider Lasso, Elastic Net, Ridge Regression, Support Vector machines or even Ensemble Regressors. Out of curiosity, we might also tinker with neural networks.

The next step will be to run and evaluate the algorithms that we have chosen. By comparing each algorithm's performance using cross validation, we can identify the algorithms that are best at “understanding” the data and at attacking the prediction of video game sales.

As a final step we are going to improve and finalize the results with focused experiments and fine tuning. Each algorithm has certain parameters that can be tweaked in order to get better results. Approaches such as a grid search will be helpful to support this task systematically. Possibly, even ensemble methods will be tried out for improving the results.

Metrics

The goal of this project is to predict the sales volume of video games, which simply is an integer. We can apply common statistics to compare and visualize the deviation of the predictions from the real results. The explained variance score or the R^2 score could be used to quantify the performance of our model.

It indicates how much of the variance within a dataset can be explained by a model.

Using this score, we can check our model for being prone to high bias or high variance. One method is to split our data into a training set that's used for training the model and a test set for testing its performance. This way we can identify and reduce underfitting or overfitting. Also, plotting the learning curves (training error and the cross validation error in relation to the training set size) can offer insight related to high bias, high variance, and to appropriate options for improvement such as collecting more samples or more features.

Instead of splitting the dataset, we can also use new data from the data sources (VGChartz, Metacritics) that are not available yet. This would basically be a real world test.

II. Analysis

Data Exploration

The raw dataset (Video_Games_Sales_as_at_22_Dec_2016.csv) offers 16 features and 16579 data points:

Table 1: Dataset Features

TITLE	DESCRIPTION	DATA TYPE
Name	Name of the game	String
Platform	Hardware Platform	String
Year_of_Release	Year of release	Numeric
Genre	Game genre	String
Publisher	Publisher	String
NA_Sales	Game sales in North America (in millions of units)	Numeric
EU_Sales	Game sales in the European Union (in millions of units)	Numeric
JP_Sales	Game sales in Japan (in millions of units)	Numeric
Other_Sales	Game sales in the rest of the world (in millions of units)	Numeric
Global_Sales	Total sales in the world (in millions of units)	Numeric
Critic_Score	Aggregate score compiled by Metacritic staff	Numeric
Critic_Count	The number of critics who generated the Critic_Score	Numeric
User_Score	Score by Metacritic's subscribers	Numeric
User_Count	Number of subscribers who gave the User_Score	Numeric
Developer	Party responsible for creating the game	String
Rating	The ESRB ratings	String

Looking for high correlations within the dataset hardly reveals anything surprising. There are very high correlations between the global sales volume and the sales volume of North America (0.94) and Europe (0.90) respectively, because those two are the biggest markets contributing the most to the total sum. Still, there's a fairly high correlation between the other countries excluding Japan and the global sales volume (0.75). Also, there's a pretty high correlation between Europe and North America (0.77) and between Europe and other countries excluding Japan (0.72), suggesting that Europe could be a linking pin between the tastes of the other regions. That's it. The Japanese market seems to work differently.

Formal abnormalities

There are some obvious formal abnormalities. For instance, the years of release range from 1977 to 2020, so there must be at least one invalid entry or we're facing a temporal anomaly and should check the chronoton levels. We can also clearly see that Metacritic doesn't provide scores for all the games that have been listed at VGChartz. There are 8466 rows without a Critic Score and even 9013 rows without a User Score. In addition, not all games contain information about their publisher, developer or their rating.

A closer look at the data reveals that there are 269 games without a year of release, mostly for Japanese releases or for old Atari 2600 games. We can also note that there are quite some games that seem to have been released in Japan only. Some of those have an extra row for Japan sales instead of using just one for all the different regional sales.

Furthermore, we can come across a fun fact: In Germany, until 2002 it was forbidden to sell the game "River Raid" from 1982 with its clunky graphics and sound. It was said to resemble a paramilitary training and could cause muscle cramps, anger, aggressiveness, absent-mindedness, and headaches.(cmp. http://www.simulationsraum.de/blog/2011/03/31/river-raid-rage/) The game sold more than 1.6 million copies which is quite a lot even for modern games. It was a top seller.

Finally, there are some cells containing the string 'tbd' instead of a proper NaN/None, some cells contain double spaces within strings, and the Critic Score and User Score are scaled differently. While the former ranges from 0 to 100, the latter ranges from 0 to 10.

Statistical abnormalities

One of the aspects of statistical abnormalities is the subject of outlier identification. "Sample outliers can be identified asof two basic types. Here we are concerned with the first type, which may conveniently be termed representative outliers. These are sample elements with values that have been correctly recorded and that cannot be assumed to be unique. That is, there is no good reason to assume there are no more similar outliers in the nonsampled part of the target population. The remaining sample outliers, which by default are termed nonrepresentative, are sample elements whose data values are incorrect or unique in some sense." (Chambers, 1986, p. 1063). Some of those nonrepresentative elements have already been mentioned above, e.g. games from 2020.

When we investigate the feature of global sales, we can easily identify some games with very high sales volumes. For example, the top selling game is said to have been sold more than 82 million times and the tenth highest selling game still more than 28 million times – whereas the mean value of global sales is at merely 0.54 million copies and the median at 0.17. Fortunately, there's an explanation. The "seemingly" top selling game is "Wii Sports", which in fact went over the counter that often, but probably only because it came as a bundle with the Wii console in 2006. People bought it whether they wanted to or not and without paying respect to critics. The same is true for a lot of the top selling titles in the raw dataset.

Table 2: Top 5 selling games in raw dataset

		GLOBAL SALES
RANK	TITLE	
1	Wii Sports	82.53
2	Super Mario Bros. NES	40.24
3	Mario Kart Wii	35.52
4	Wii Sports Resort	32.77
5	Pokemon Red/Pokemon Blue	31.37

Those and other titles have probably been recorded correctly. Yet they could be considered outliers. Alternatively, we could suggest to add a feature that indicates that the game had been bundled with hardware and might account for the tremendous sales volume.

We can also detect very large values for the number of votes that come from customers. For example, the game "The Witcher 3" for the PC received over 10,000 votes while the mean value is at roughly 162 votes and the median at 24. It is eye-catching that the games with the highest user count also have a very high user score – at least most of the time. There are some exceptions, e.g. Diablo III for the PC. It received a critic score of 88 and was praised, yet the user score is just average. It seems that many people gave a very bad score, presumably because the game forced users to be online in order to be able to even play the game in single-player mode. The high count seems to either confirm love or hate for a game.

Table 3: Top 5 games concerning user count

RANK	TITLE	CRITIC SCORE	CRITIC COUNT	USER SCORE	USER COUNT
1	The Witcher 3: Wild Hunt (PC)	93	32	9.3	10665
2	The Witcher 3: Wild Hunt (PS4)	92	79	9.2	10179
3	Counter-Strike: Source (PC)	88	9	8.9	9851
4	Diablo III (PC)	88	86	4.0	9629
5	The Elder Scrolls V: Skyrim (PC)	94	32	8.1	9073

Given some of the statistical information above we can already assume that some of the features are skewed, e.g. sales volume or user count. The visualizations in the next chapter will show this fact more clearly, but just looking at the median, mean and standard deviation of the numerical values suggests that merely critic score, critic count and user score could be fairly normally distributed.

Exploratory Visualization

We start by looking at some of the aspects that have already been discussed above. For instance, we talked about the correlations between local sales and global sales. We can spot the very high positive correlations between North American sales and global sales and between EU sales and global sales easily. In the corresponding subplots, we see a thick straight line from the lower left corner to the upper right corner. It could be seen even better if we had got rid of the extreme sales numbers for games that have been sold with a console.

For this project, we could try to create regressors for each region, but since the regional sales volumes are highly correlated with the global sales (except for Japan), we are just trying to predict the latter.

For creating a regressor, we hope to have some features that account for a large portion of the final outcome, and our best hope are the scores awarded by critics and users. We could assume that there's a positive correlation: The higher the score, the higher the sales volume. We already know that this doesn't hold true entirely. There is a positive correlation between each score and the global sales, but it is far from being significant. We can check this by plotting the scores and the global sales. Obviously, the scores tell us a little bit since the sales volume seems to be higher if the score is high, but there are also many games with very good ratings but low sales.

If we have a closer look at the relationship between the score that critics awarded and the score that users awarded, we in fact notice that they are kind of correlated, but there's a lot of variation and the plot forms a longish cloud in the upper right corner (Please note that we already scaled the user score by a factor of

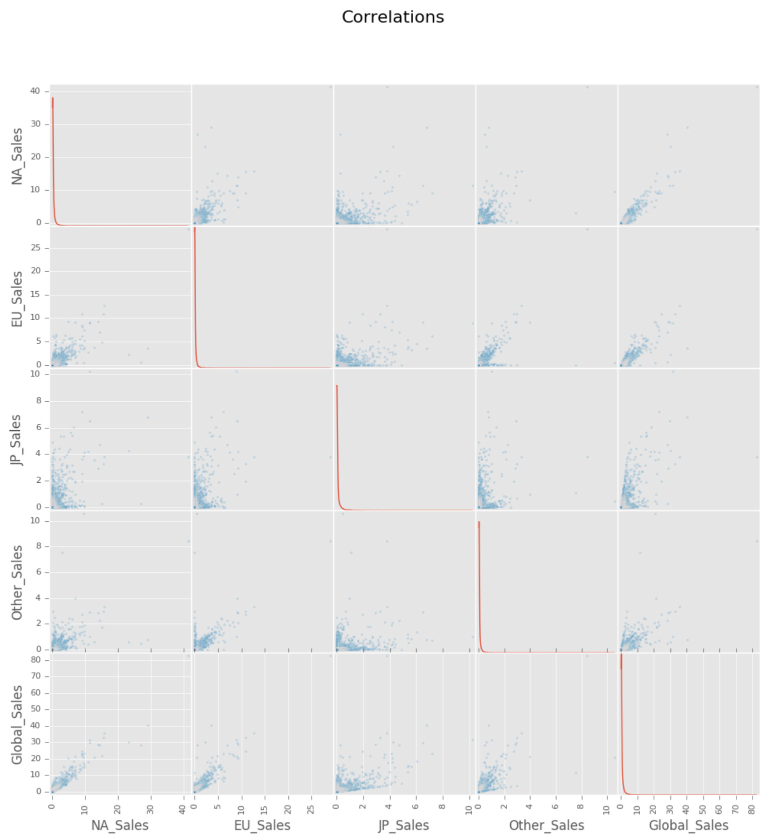

Figure 1: Correlations

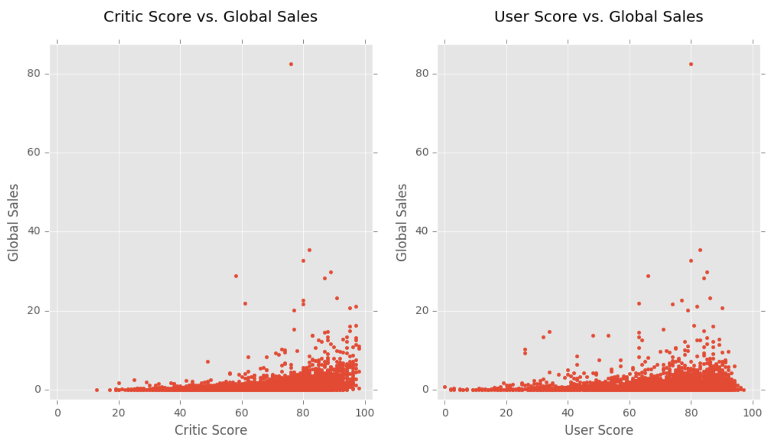

Figure 2: Score vs. Sales

10 in order to get nice square plots). Obviously, critics and users alike tend to award scores above average more often then scores below average. This allows us to “see” that both features are skewed and don’t portray a normal distribution.

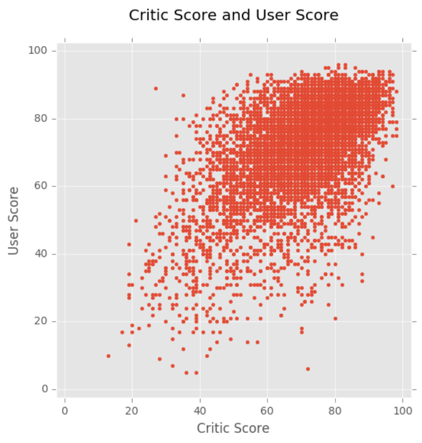

Figure 3: Score Comparison

Of course, we can also check the distributions directly by using a scatter matrix. It shows the numeric features that we could possibly transform. This wouldn’t make sense for the year of release, and the sales numbers are skewed to such a large degree that assuming a normal distribution might be a wrong premise. We can see that both, the distribution for the critic score and for the user score, are skewed negatively, the latter a little more than the former. On the other hand both counts are skewed positively considerably, the user count to such a large degree that transforming it to resemble a normal distribution might not be a wise decision.

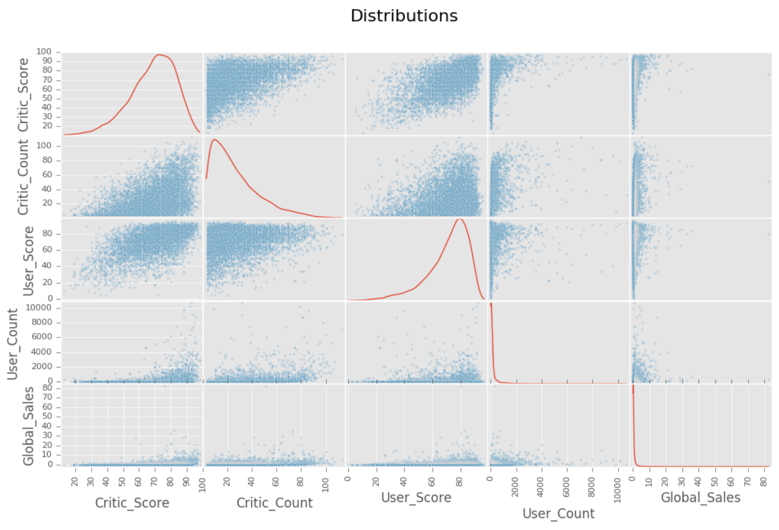

Figure 4: Distributions

Summing up the sales visualizations, we could suppose that especially the platform manufacturer and the mobility of the console might already hint to larger or smaller sales numbers because of their large fractions in the pie charts. Unfortunately, at this stage we have not yet cleaned the data as this is supposed to happen later in this template. We might recreate the visualizations later.

Algorithms and Techniques

As we stated above, the dataset may not be ready for use and data cleansing might be in place. There are lots of incomplete samples. Some information can be gathered or corrected manually. For example, we are going to add missing years of release. Some rows with missing scores will be removed. In the data preprocessing stage, we are also going to deal with outliers. Finally, before building a model, we should consider to scale the data and possibly also to transform some features in order to get more normal distributions.

After these steps, we can decide which algorithm we should use. According to the Scikit-Learn Algorithm Cheat-Sheet, for regressors with less than 100.000 samples we shoulg e.g. use Lasso and Elastic Net, but we cannot be sure that only few features should be important. Alternatively, we should use Ridge Regression and Support Vector Machines with different kernels or even ensemble methods.

Lasso, Elastic Net and Ridge Regression are linear regression algorithms that use regularization to prevent overfitting. They all offer the following parameters that we can use for tuning:

- `alpha` is a positive float that indicates the strength of the regularization. The larger the values are, the stronger the effect. Of course, very large values will lead to too simple models that cannot predict anything.
- `fit_intercept` is set to True by default and setting it to False assumes that the data is centered. Since we have very skewed data, using False should produce worse results.
- `normalize` is set to False by default and can be used to, well, normalize the dataset before regression.
- `max_iter` is the number of iterations. Larger values may produce better results.

Elastic Net Regression additionally gives us the parameter *l1_ratio* that we might tweak:

- `l1_ratio` is a float between 0 and 1 that determines the type of regularization. If close to 1, the penalty will be of type L2, and if close to 0, it will rather be like type L1.

Ridge Regression additionally provides us with the the parameter *solver* to set:

- `solver` allows us to choose from different computational routines, but can also be set to 'auto' in order to choose the solver automatically based on the type of data.

Furthermore, we can use a Support Vector Regressor which can be very effective in in high dimensional spaces, and we can also use the kernel trick to create non-linear decision boundaries in order to capture very complex patterns in the data that the linear models fail to grasp. Support vector machines don't perform well on large training datasets, but ours is still fine.

The following parameters might be worth tuning:

- `C` is the regularization parameter of support vector machines. The higher the value of C, the less the model tolerates errors (uses a hard margin) and the more prone to overfitting it becomes.
- `epsilon` determines the number of support vectors that will be used to define the margin. Increasing epsilon will decrease the number of support vectors and reduce the danger of overfitting, but contrarywise the solution may be less accurate.
- `kernel` defines the way how the computation is virtually transferred to a higher dimensional space. An appropriate kernel can help us to deal with non-linear problems, and 'rbf' seems to be the right choice.
- `gamma` defines the sphere of influence that a single training example has. If *gamma* is low, then the sphere of influence is large and more samples affect the definition of the margin. Vice versa, high values reduce the number of samples defining the margin, but may cause overfitting.

In order to get a more robust result, we are going to apply cross validation (ShuffleSplit): We are going to randomly split our dataset into a training set and a test set several times and compare the performance on both sets. This way we will be able to detect overfitting and underfitting and decide what we can do to improve the results.

Benchmark

There are some people at Kaggle who seem to be experimenting with the dataset, too. For example, Jonathan Bouchet built a polynomial regression model in R and reports an R^2 score of 0.098404. Since Bouchet barely preprocessed the data and did not try other algorithms, we can be quite optimistic that we can come up with a better result.

III. Methodology

Data Preprocessing

The first phase of our data preprocessing was general cleaning. It was performed manually on the dataset itself, because those steps can be useful for anyone.

We had to correct some entries of the feature “user score” that contained the string “tbd” instead of a numeric value. Those were replaced by “NaN”, because without these changes the data type of the feature would not be detected correctly by Pandas. Afterwards, we deleted some double spaces and leading and trailing spaces in strings.

The next manual correction concerned the feature “year of release”. There were 269 dates missing, mostly for games that had only been released in Japan or for the Atari 2600 console. We were able to find relevant information for 245 of those entries. Some dates have not been added because the corresponding games seem to have been canceled and never been released – although there are numbers given for sales volume. This raised some doubts about the quality of the dataset.

Another quality problem arose from uneccessary rows. For some games, there were separate entries for the Japanese market only. Those were merged with the sample that contained the numbers for sales in other regions. In some cases, those sample already contained the sales volume for the Japanese market. In this case, we chose the higher value respectively.

The results of this cleaning process have been saved for further use. Afterwards, the feature “user score” was scaled by factor 10 in order to match the range of 0 to 100 of the feature “critic score”. Further preprocessing involved removing samples that are not considered fit for analysis: First of all, all rows that contained at least one NaN value have been removed. We wanted full information about our game titles.

Afterwards, we had to look at the feature “year of release”. There are only few rows before 2000 (even after adding some missing dates), because MetaCritic as supplier of scores was founded on July 16, 1999. Also, looking at the sum of global sales grouped by year, seamingly revealed a large gap from 2000 to 2001 – there probably wasn’t such a boost in video game sales, but MetaCritic had to get going first. We also had a gap from 2015 to 2016 in the other direction. This might be a result of declining video sales numbers that can be observed, but it might also reflect the Holiday Season business that was still not over. That’s why we considered rows before 2001 to be outliers to be removed, also those after 2015.

There were some more rows that were removed because they might not be good samples: There were several games that had only be released in Japan, so the Japanese sales volume would be the global sales volume. Taking into account the particularity of the Japanese market that has been detected above, we would distort our results when using global sales as our target variable. After this step, we had 6474 samples left, and this was the final step of data preprocessing that we can be sure about.

Beyond this sample removal, there are two features that we could generate from the existing data and that might be useful. On the one hand, we can group the platforms by their manufacturers, e.g. all the different types of Play Stations were built by Sony. If there’s fandom for a company, this might influence sales. On the other hand, we can differentiate stationary platforms from mobile devices. This might be an indicator for high or low sales numbers.

Furthermore, there are some features that we cannot make use of. Obviously, the name of a game hardly gives us any information. The publisher and the developer might be useful in general, because some of them might have a reputation for high or low quality games. Unfortunately, there were more than 200 different publishers and more than 1,000 different developers which is very high given the size of our dataset. We removed those features.

The rest of this section deals with aspects that are rather optional, and their usefulness might depend on the algorithm that’s used later.

If we take a step back and look at our plots showing score and sales, we can clearly see some but very few games that seem to be top sellers. Investigating some more, we can reveal that 75 of those 129 games that sold more than 5,000,000 copies had been sold as bundles together with a console. This is an important information for interpreting the sales numbers, because those games might not have been purchased based on opinions about the quality, but because someone wanted to buy a console anyway. It’s not clear whether those games should be treated as outliers and be removed, or whether they should be flagged using a new feature. The latter seems to be more appropriate, but a quick test with default parameters revealed that ridge regression gets better results, while a support vector regression returns worse results. Later on, we are going to check both options. Also, there may be some titles that slipped through our fingers and have been sold in bundles. Also, we have not yet checked those that sold less that 5 million copies. Since the manual cleaning and documentation process is very tedious, we hope that we found enough.

As mentioned above, we could now remove some statistical outliers based on their distance to the features' quartiles, but we might as well decide against because we could lose representative outliers (cmp. Chambers, 1986).

Basically the same is true for scaling or transforming the data. Some estimators might behave badly if the individual features do not more or less look like standard normally distributed data with zero mean and unit variance, and we might need to use the StandardScaler of Scikit-Learn to correct this. Some estimators may have trouble finding good solutions if the ranges of feature values are very different, so we might want to use a MinMaxScaler to scale all features to a range [0, 1]. (cmp. Scikit-Learn: Preprocessing data) Finally, we might want to use power transformation (BoxCox tranformation) in order to make the data more normal distribution-like. In some quick tests with default parameters, some algorithms yielded better results while some got worse. We are going to deal with this aspect in the next section.

Implementation

As mentioned before, there are several operations that might be useful or not depending on the regression model used. That's why we implemented some kind of a grid search for meta parameters. Before applying each of our four regression algorithms (Lasso, Eleastic Net, Ridge Regression and Support Vector Regressor) with their parameters, we can choose to vary several meta parameters for operations that may take effect after we removed the "Japanese only" sales giving us a dataset containing 6474 samples:

- `params_remove_bundles` can be True or False. If True, the samples containing games that were part of a console bundle will be considered to be outliers and be removed from the dataset. If False, the feature "Bundle" will be introduced and set to "bundle" for games that were part of a console bundle and to "unknown" for the rest because we didn't check all games as mentioned already.
- `params_outlier_threshold` is -1 or a positive float. If -1, none of the remaining samples will be treated as outliers. Otherwise, the float represents the factor to multiply the interquartile range with for defining an outlier threshold (cmp. k in Tukey's range test, e.g. 1.5 as default for outliers). We will use [-1, 1.5, 1.7 and 2.0] for our initial grid search.
- `params_outlier_multiety` is a positive integer. It defines the number of features for one given sample that need to be considered as outliers for classifying the complete sample as an outlier that should be removed. So,

if 1, the sample will be removed if there's at least one feature that contains an outlying value. If 2, the sample needs to have at least 2 features with outlying values to be removed, etc. We will use [1, 2, 3] for our initial grid search.

- `params_standardize_set` can be True or False. If True, all features will be scaled using a StandardScaler, so they have zero mean and unit variance.
- `params_scale_set_0_1` can be True or False. If True, all features will be scaled to range [0, 1].
- `params_boxcox_set` can be True or False. If True, the features "Critic Score" and "User Score" will be transformed to be as normal distribution-like as possible – unless the data have been standardized which could lead to negative values preventing the BoxCox-transformation from working. We already discussed that only the skewness of "Critic Score" and "User Score" seems to be viable for this operation.

In order to improve our model and prevent overfitting, we also want to use some kind of cross validation. We use Scikit-Learn's "ShuffleSplit" because it's quite fast. It randomly splits our dataset into a training set and a test set. This allows us to train our model with the training set and then to compare the results with the predictions for the test set. This way we can detect overfitting. For ShuffleSplit, we cannot only define the fraction of samples that should be randomly assigned to the training set and the test set, but we can also define how often this procedure should be repeated. This way, we will be able to compute a mean R^2-score instead of using one for just one arbitrary training set and test set. Our implementation also allows us to choose two "grid search" parameters for ShuffleSplit:

- `params_n_splits` contains the number of iterations or "splits" used for ShuffleSplit. We will use just [20] for our initial grid search.
- `params_test_size` contains the fraction that should be used as a test set. We will use just [0.2] for our initial grid search.

Also, we have one more parameter:

- `params_random_state` contains an integer that can be used to set a pseudo-random number generator state used for random sampling. It will be used by ShuffleSplit and some of our regression models. We will just use [31415] for our initial grid search.

As mentioned in section "Algorithms and Techniques", we used Lasso, Eleastic Net, Ridge Regression and Support Vector Regressor. For each combination of the aforementioned meta paremeters, we ran a grid search for each algorithm that used some variations of the default values for alpha, C, etc. (see above). For each combination and algorithm we stored the grid search parameters, the R^2 value for the training sets and for the test sets and their absolute distance to each other – we don't only want our R^2 value to be high, but we also want a small gap indicating absence of overfitting.

The highest reported R^2 value for training sets was a magnificent 0.89, but it was accompanied by an R^2 value for the test sets of -0.02. This result clearly suffered from overfitting and cannot be used. We decided to go for a maximum threshold of 0.05 for the gap between the R^2 values and had a look at the best remaining results:

Table 4: best results for global large grid search

Regressor	R2 Train	R2 Gap	Std'ze / Scale / BoxCox	Remove Bundle	Threshold / Multiety
SVR C=1.3, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='auto', kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False	0.553	0.022	True / False / False	False	-1.0 / 1.0
Ridge alpha=1.7, copy_X=True, fit_intercept=True, max_iter=1000, normalize=False, random_state=31415, solver='auto', tol=0.001	0.522	0.006	True / False / False	False	1.5 / 2.0
Ridge alpha=1.7, copy_X=True, fit_intercept=False, max_iter=1000, normalize=True, random_state=31415, solver='auto', tol=0.001	0.511	0.041	False / True / True	False	1.7 / 2.0

The best three results were achieved by a Support Vector Regressor (R^2 = 0.553, gap = 0.022) and by two different Ridge Regressors (R^2 = 0.522, gap = 0.006 and R^2 = 0.511, gap = 0.041). All of them (and most of the top models) used the additional feature that flags game bundles. It seems that this variable is very useful for regression. We decided to use the SVR, not only because it had the higher R^2 score, but because it didn't rely on removing statistical outliers in contrast to the Ridge Regressors. As already stated, we considered all the samples to be carriers of relevant information. Furthermore, the next best solutions do not show a clear pattern concerning the meta parameters. This might be a sign for them rather optimizing the data to fit the model instead vice versa.

Refinement

Our best initial solution was obtained for a preprocessed dataset that has additionally been standardized with "StandardScaler". The parameters for the Support Vector Regressor that had been set during the grid search were C, epsilon, gamma and kernel, and the R^2 score of 0.55 with a gap of 0.022 was achieved with C=1.3, epsilon=0.1, gamma='auto' and kernel='rbf'.

We decided to try to improve the R^2 score and the gap while setting a maximum gap at 0.010. This should be a very decent indicator for hardly showing any traces of overfitting. By iteratively tweaking C and epsilon manually, we finally got an R^2 score of 0.591 with a gap of 0.010 when using C = 1.74, epsilon = 0.12 (and still gammma='auto' and kernel='rbf').

Table 5: Intermediary results

C	EPSILON	GAMMA	KERNEL	R^2 (TRAIN)	R^2 GAP
1.30	0.10	auto	rbf	0.553	0.022
1.70	0.10	auto	rbf	0.587	0.008
1.80	0.10	auto	rbf	0.594	0.015
1.73	0.11	auto	rbf	0.590	0.010
1.74	**0.12**	**auto**	**rbf**	**0.591**	**0.010**
1.75	0.13	auto	rbf	0.592	0.011
1.74	0.14	auto	rbf	0.591	0.010
1.74	0.15	auto	rbf	0.592	0.011
1.74	0.10	auto	rbf	0.590	0.010

IV. Results

Model Evaluation and Validation

The final model is a Support Vector Regressor using the following parameters next to default values:

- `C`: 1.74
- `epsilon`: 0.12
- `gamma`: 'auto' (1/number of features by default)
- `kernel`: 'rbf'

The model was chosen because it returned one of the highest R^2 values while still not overfitting to the training data. It was also chosen because it doesn't require to remove outliers for purely statistical reasons. Thus, it may preserve relevant information that would be lost otherwise.

First of all, we tested the model with some varying meta parameters:

- `params_n_splits` in [20, 25, 30, 100]
- `params_test_size` in [0.2, 0.25, 0.3, 0.5]
- `params_random_state` in [31415, 42, 123]

We can see that even varying those can have a large influence. The R^2 score ranges from 0.51 to 0.61. This seems to be a rather significant difference in my opinion. Luckily, the gap ranges from 0.001 to 0.06 only, and this result is still pretty good. On the other hand, we have reason to believe that while the results are not completely arbitrary, they are not stable either. For example, when sorting the data by the R^2 score, they are virtually also sorted by random_state. All top scores result from random_state = 42, then nearly all the next best scores result from random_state = 31415, and finally 123 kicks in. In consequence, the results seem to be quite dependant on the random sampling within the cross validation. We might also be able to find values for "random_state" that will give us even worse results.

Unfortunately, we still cannot trust our model. Even our best R^2 value is still too low. We cannot predict much more than half of the variance within the global sales. Also, we'd have to investigate the robustness concerning the random sampling and possible trade some portion of our R^2 score in order to reduce variance and overfitting.

This might in fact be a good idea as we can conclude from checking the performance for completely new data. Those must be scaled (standardized) using Scikit-Learn's "StandardScaler" that was obtained from the training data, but then we're good to go. For our sensitivity analysis, we can use the samples from 2016 that had been removed because we didn't have data for the complete year and because the dataset was probably lacking sales information from the Holiday Season. This restriction might influence the reliability of our sensitivity analysis a little bit, but it should still give us some insight about the reliability of our model.

Our previous scepticism was justified. When using our regressor to predict the sales of the games that had been released in 2016, we computed an R^2 score of 0.38. It's not terrible, but it's definitely not close to 0.59.

Justification

As we learned from the sensitivity analysis, we should consider to reduce the precision of our model in order to account for some signs of overfitting. We should not assert that we have pushed the R^2 score beyond 0.38, but we still got a significantly higher R^2 score than 0.09 that was achieved by Jonathan Bouchet using a polynomial regression model. However, we cannot claim to have solved the problem. As we stated in the previous section, an R^2 score of 0.59 – or more realistically of roughly 0.4 – still leaves much space for variance that cannot be explained by our model.

V. Conclusion

Free-Form Visualization

I consider learning curves to be an interesting and useful tool for machine learning problems in general and for this project in particular. It was quite helpful to have a look at them to assess how the algorithms are performing. We could have a look at the curves of the final solution.

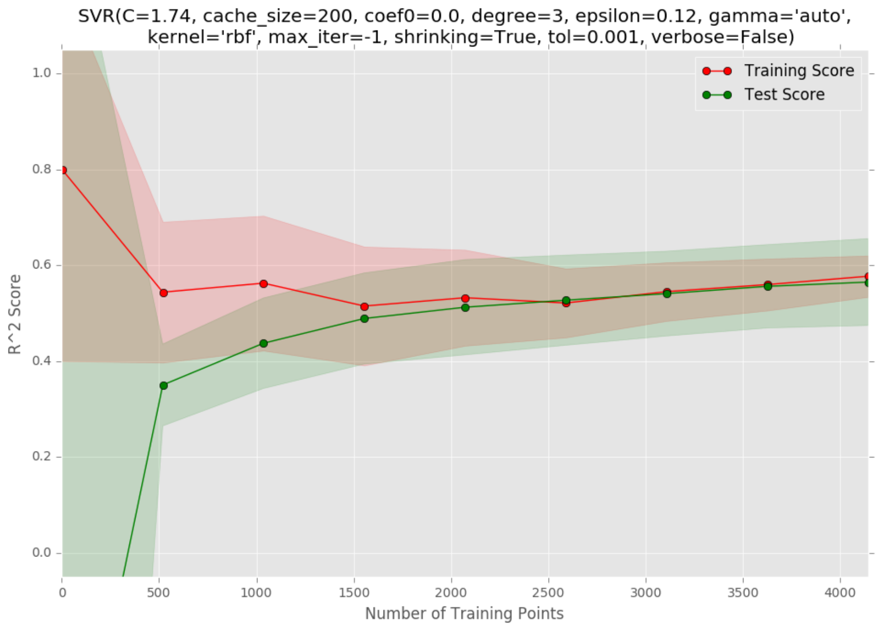

Figure 5: Final Learning Curve

We can see that the training score and the test score converge pretty quickly. At roughly 1500 samples, the gap between the two is already very small. This is a good sign. Unfortunately, we also observe two not so nice facts:

- We already know that 0.591 is our final R^2 score which is far from being perfect. The progression of the curves also tells us that more samples will hardly help us. We need more independent features.

- We can see a red corridor and a green corridor for our curves. They represent the minimum and maximum score that was achieved by the algorithm for all the random samples that were created by ShuffleSplit. Consequently, while the mean scores for training and testing are close, there's still a lot of variance. Depending on the split or random sample, the gap might be as large as about 0.15 which would indicate overfitting. We might want to trade some portion of our R^2 score for less variance.

Reflection

To be fully honest, I was a quite surprised by the results. Although our model is not fit for productive use, I didn't expect to get such a high R^2 score for predicting global sales by basically looking at critic score and user score. Cleaning the data and generating the "bundle" feature was a tedious job, but it seems it has been worth the effort. Also, this process seemed to be pretty straight forward.

Much more difficult for me was choosing the algorithms and techniques. The linear regressors that were suggested by Scikit-Learn for this type of problem had not been covered in the course, and I only found pretty mathematical explanations that were hard to tackle. That's why I rather used a brute force approach by setting up a grid search for what I called meta parameters. Computing these results took my computer more than one day (and several restarts because of thing I missed). It is obvious to me that this may have worked in my case, but it is absolutely not feasible for datasets with more samples or features. I will have to learn more about the algorithms in order to know when they might be beneficial or not.

I think that trying to predict global sales for video games could be possible if some things get improved. I am going to elaborate on this aspect in the upcoming section.

Improvement

There are several ways how our solution could be improved. We could either approach our data or our methodology.

We already discussed that collecting more samples will hardly help to improve our model. We would have to improve, gather or create more independent features:

- We could check more games for being part of a console bundle and flag them correctly.
- We could define a binary feature that indicates whether a game is part of a franchise or not. For example, there might be games that receive a low rating by critics or users, but people buy them anyway because the contain a beloved character such as Super Mario or because people simply

collect all the games from the Final Fantasy series. There might be some debate about the definition of a franchise, but it might be worth a try.

- It could be helpful to add the spendings on advertising as they might influence sales. Unfortunately, there's probably no good way to acquire this information.
- We could scrape scores from other services such as IGN. However, this would require to clean and merge the datasets. Also, we'd probably lose some samples if not all the sources contain information about all games. Finally, we can surmise that the scores might correlate and wouldn't give us relevant information.

Besides improving our data quality, we could try other algorithms and techniques. For example, we could apply a principal component analysis to make our dataset smaller. This would possibly allow us to compute results quicker and might also give us some more insight into the structure of the data. And, of course, Scikit-Learn offers a bunch of other algorithms for regression. We could e.g. experiment with neural networks that right now create a furor in many areas. If only our lives weren't finite ;-)

VI. Sources

- Beaujon, Walter S. (2012). *Predicting Video Game Sales in the European Market.* Retrieved from https://www.few.vu.nl/nl/Images/werkstuk-beaujon_tcm243-264134.pdf (January 12, 2016).
- Chambers, Raymond L. (1986). Outlier Robust Finite Population Estimation. *Journal of the American Statistical Association*, 81(396), 1063-1069.
- Dellarocas, Chrysanthos, Zhang, Xiaoquan (Michael) & Awad, Neveen F. (2007). Exploring the value of online product reviews in forecasting sales: The case of motion pictures. *Journal of Interactive Marketing, 21(4)*, 23-45.
- Entertainment Software Association (2015). *Essential Facts About The Computer And Video Game Industry. 2015 Sales, Demographic And Usage Data.* Retrieved from http://www.theesa.com/wp-content/uploads/2015/04/ESA-Essential-Facts-2015.pdf (January 12, 2016).