Optimization and Machine Learning Project

Erik Oscar Riis Enander and Ludvig Lous

June 29, 2023

95601 - Optimization and Machine Learning

Alma Mater Studiorum - Università di Bologna

Contents

1 Task 1

1.1 (a)

The vgsales dataset was loaded as a dataframe-object in python using the read_csv() function from the pandas library. Using the shape-attribute of the dataframe, the dataset was found to consist of 11 columns (from here on referred to as features) and 16598 rows (from here on referred to as samples). Further, the features were found to consist of 7 numerical features and 4 categorical features.

1.2 (b)

By using the function info() on the dataset, the only features which contained missing values were the categorical features *Year* and *Publisher*, which contained 271 and 58 missing values respectively. As the samples containing missing values constituted at most around 2 percent of the dataset, it was decided to remove these samples from the dataset. An alternative approach would have been to use imputation by for example setting each missing year value to the mean value of the year, and every missing publisher value to the mode of the publisher values. As the percentage of samples with values was appreciably small however, it was decided to remove these samples entirely as it wouldn't be expected to change the characteristics of the dataset in any substantial way. The dataset was also checked for duplicate samples using the duplicated().sum() function, and there were found to be zero duplicates in the dataset.

The numerical features of the datset was plotted using boxplots to visualize the presence of outliers in the dataset. The plot is shown in Figure 1. As seen in the plot, all the sales-related features contained a large amount of outliers, while the *Year* feature had some outliers in the earliest recorded data. The outliers in the sales are probably caused due to some big, popular games which sell a lot more copies than the average

game, and the outliers in the release year are probably caused due to a lower amount of games being released in the 1980s. The outliers in the data were first removed using the Inter Quartile Range (IQR) method. After filtering the dataset using the IQR method, the number of samples decreased by around 28 percent. This was a fairly large amount of the dataset, and to decrease the risk of losing any potentially important characteristics of the dataset, the Percentile-based method was used to remove the outliers instead, keeping the data inside the 1st and 99th percentile. This removed the outermost outliers, while only decreasing the number of samples in the dataset by about 5 percent. The percentile-based method was thus seen as a good compromise between removing outliers and maintaining the characteristics of the dataset.

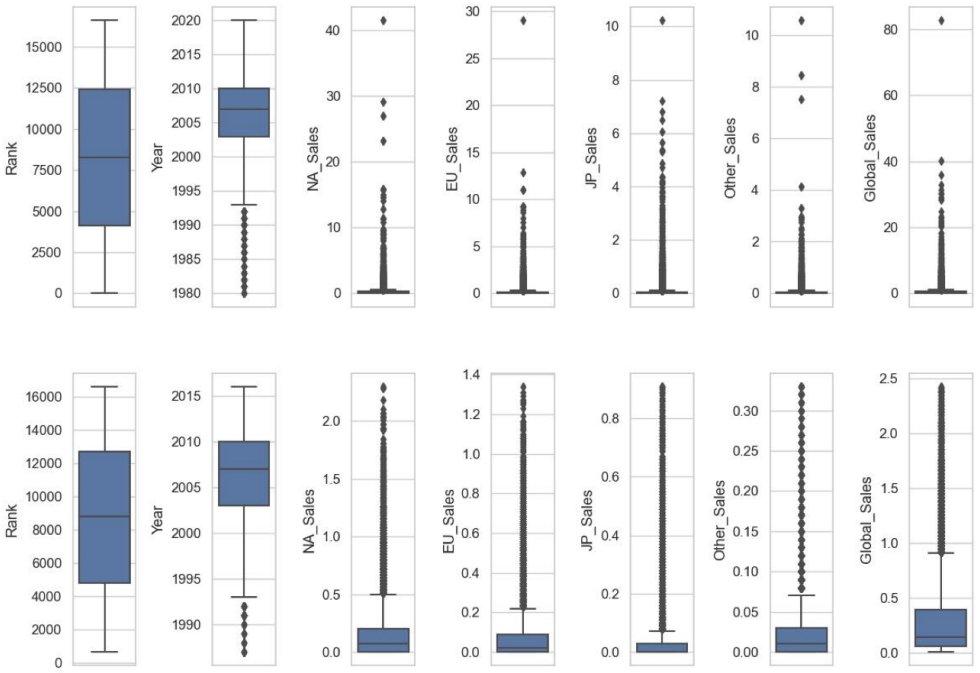

Figure 1: Box plots of the numerical features of the dataset. Upper: Before filtering, Lower: After filtering from 1st to 99th percentile

1.3 (c)

The plots in this section were chosen to be plotted on the dataset before the removal of the outliers. This was done in order to get a better understanding of the original data, and get a better overview of where some of the outliers to be removed were located in the dataset.

We would like to take a look at the sale of games in North America, and start by limiting the plot to only include the 10 games with the highest sales. This is created by sorting the dataset in order of *NA_Sales* and only include the first 10 values of the dataset by indexing. The barplot is plotted using the seaborn library.

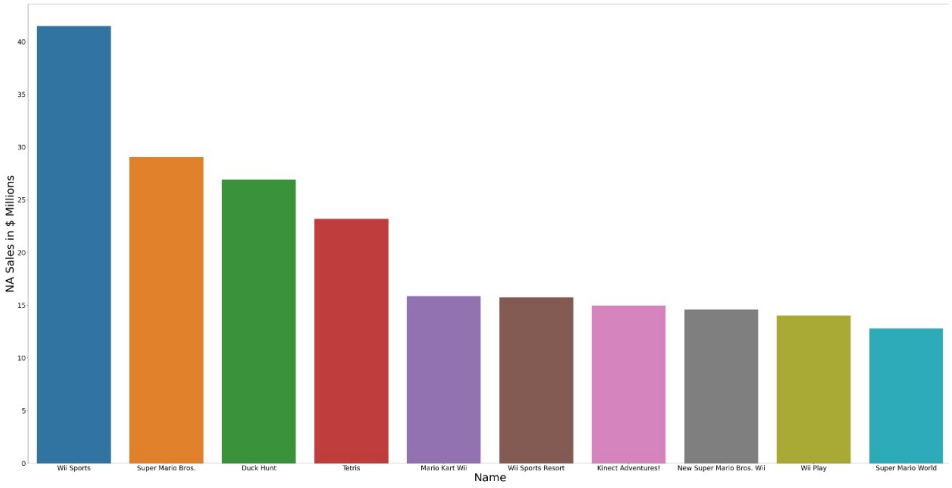

Figure 2: Top 10 games in North American Sales.

As we can see from the barplot, Wii Sports is clear at the top, with Super Mario Bros., Duck Hunt and Tetris following at the closest spots

If we instead are interested in visualizing all the games related to North American sales, we can use a scatterplot. We plot this together with the different genres of games, the results are shown in Figure 3.

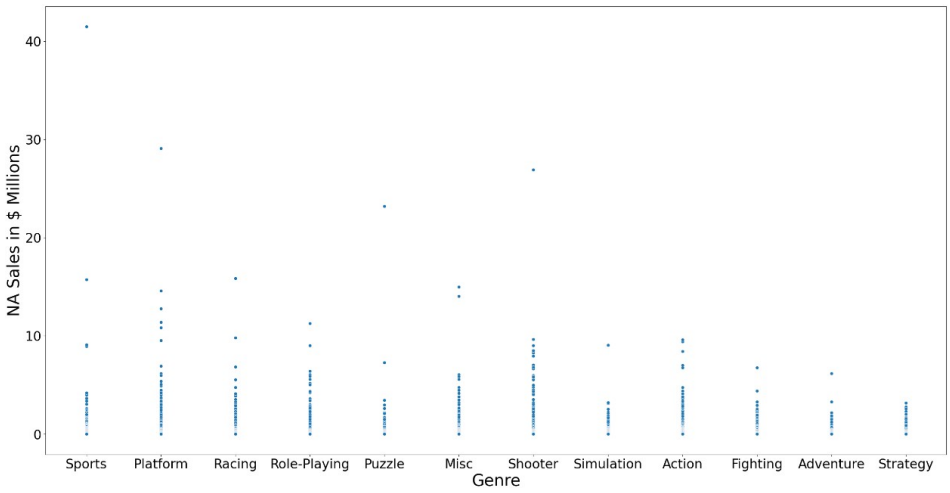

Figure 3: Top 10 games in North American Sales.

From this plot, we can see that sports is the genre with the biggest game in terms of sale, but Platform has the most games with over 10 million dollars in NA sales. Other points to notice from the plot is that strategy has no hugely popular games that separates from the rest, while the shooter category has the most even distribution up to the 10 million dollar mark.

A nice way to visualize the distribution of publishers is by using a wordcloud. We are interested in checking which publishers have published the most games. The wordcloud can also be visualized in python, but the visualization in R includes more details and look nicer, so we decided to use R for this plot. By importing the wordcloud function from the wordcloud library, we received the following plot:

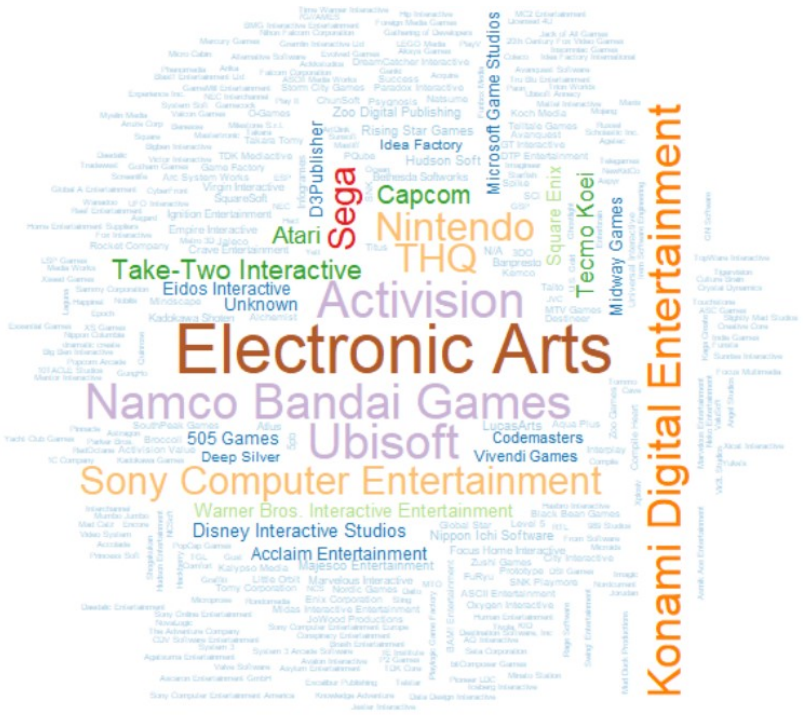

Figure 4: Wordcloud distribution of Publishers.

As we can see from the wordcloud, Electronic Arts is the largest publisher in number of games published, with Activision, Namco Bandai Games and Ubisoft in the spots after.

If we are interested in looking at the actual numbers behind the wordcloud, this could be visualized with a simple histogram. Since there exist around 750 publishers, this plot would look very chaotic, hence why we limit the plot to show the 30 publishers with the most games:

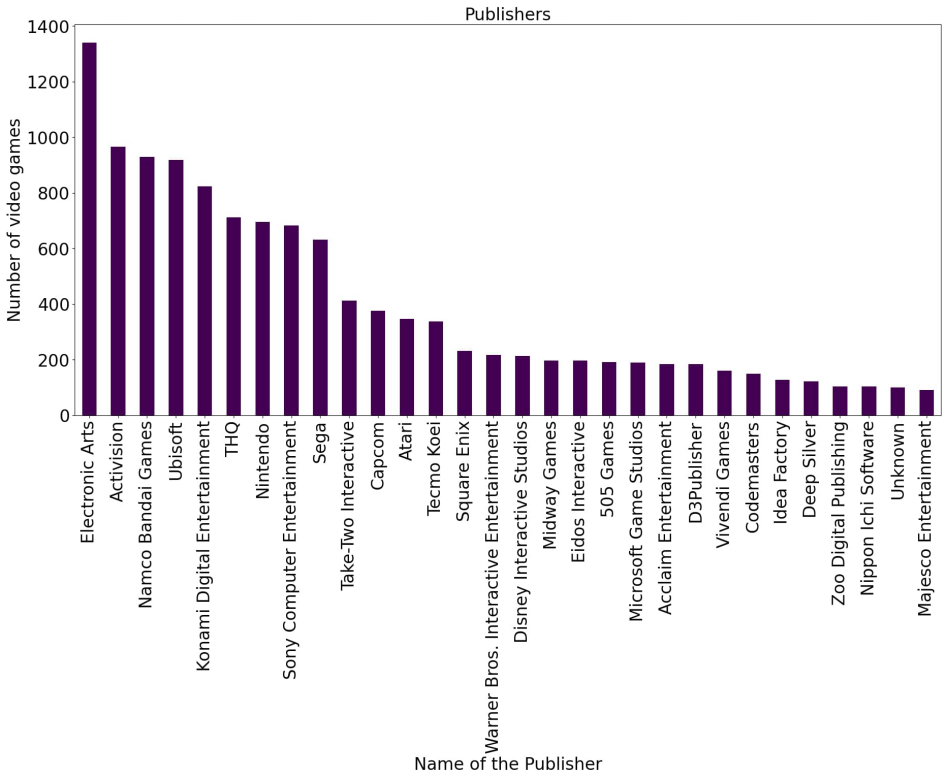

Figure 5: Histogram showing amount of games for each of the 30 biggest publishers.

It could also be interesting to see the correlation between *NA_Sales* and *Publishers*, but once again, the amount of different publishers make this difficult to visualize. Therefore, we select only the 20 largest publishers, and see what their sales are in North America. We plot the correlation with a horizontal barplot using the seaborn library, and get the following result:

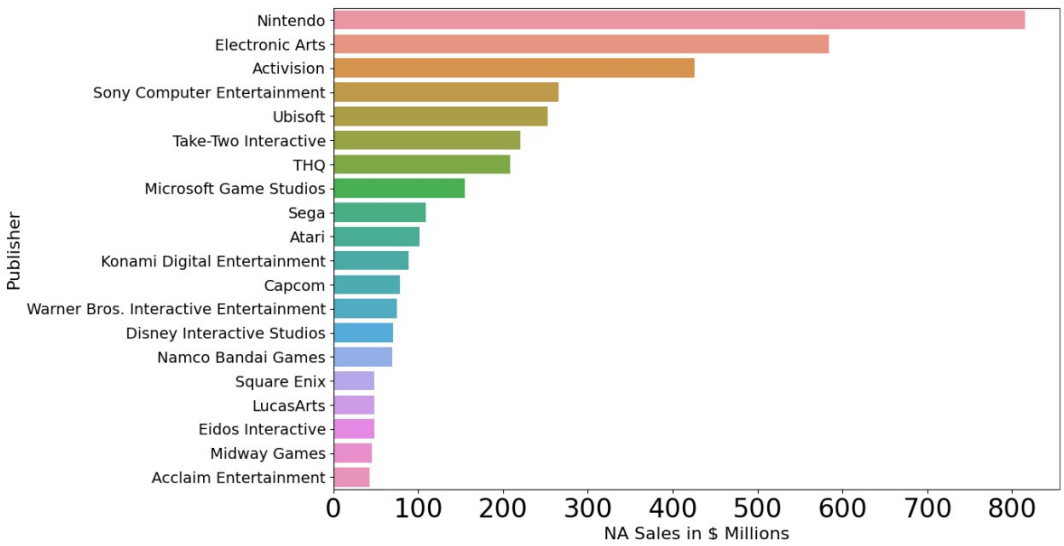

Figure 6: Horizontal barplot showing the 20 publishers with the biggest revenues from sale in North America.

From this plot, we can see an interesting correlation; Even though Nintendo is only number 7 in terms of games released, they have by far the most sales in North America in terms of revenue. Another noteworthy point is that THQ have produced more amount of games than Nintendo, but only have about a quarter of the sales as that of Nintendo has in North America.

2 Task 2

2.1 (a)

As the *Rank* feature of the dataset is a ranking of which games have sold the most copies globally, one would thus expect the *Rank* feature to be a relevant feature for the prediction of the European sales, as its contains some information about how much a game has sold compared to others. Although the ranking is based on global sales and not European ones explicitly, one would expect the European sales to be at least somewhat correlated to the global sales.

By plotting the mean of the *Sales*-features against the *Year* feature, we can get an idea of whether these features are relevant for predicting the European sales. The plots are shown in Figure 7. As seen from the plot, one can see that there seems to be a correlation between the different *Sales*-features, as they have fairly similar shapes in the plot. From the plot, it does not seem like the the European sales of a game is related to the year it was released. It was therefore decided to drop the *Year* feature from the machine learning model.

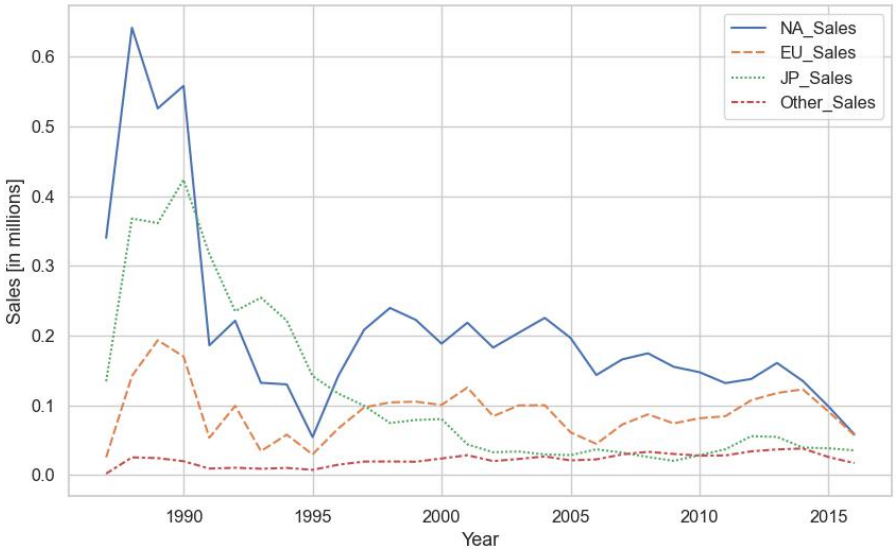

Figure 7: Mean sales of games plotted against the release year.

The categorical features *Name, Platform, Genre* and *Publisher* can also potentially influence the sales prediction. Some game consoles are more popular than others, and games on these consoles would probably sell better on average due to a larger player base. Some genres could be more popular than others and sell more games on average, and some publishers will probably have higher expectations for their upcoming games and more marketing money, which could also lead to higher sales per game. The game title is assumed to be fairly independent from the sales. Some video game franchises will however be more popular than others, and any video game titles containing words such as "Mario" or "Pokemon" would probably sell fairly well. Generally speaking, the

game title is not expected to be correlated to the sales.

2.2 (b)

The numerical features which were deemed as relevant features were scaled by using the MinMaxScaler from the scikit-learn library (sklearn). The MinMaxScaler normalizes the numerical features such that all their values lies between 0 and 1. This was done to prevent features with large values from dominating the learning proccess, and potentially increase the performance of the subsequent machine learning algorithms which were to be used.

The dataset contained four categorical features, *Name*, *Platform*, *Genre* and *Publisher*. None of these features can be considered to be of an ordinal kind, as there are no clear ordering of their values. Most machine learning algorithms can only take in numerical features, and although there are no clear ordering of the variables such that we could use an ordinal encoding, it is possible to use One-Hot Encoding. This method replaces the categorical feature columns with boolean columns for each value in the category which are true when the sample is of the given category value. In order to perform One-Hot Encoding, however, the features need to be of a low cardinality, preferably lower than 15. The cardinality of *Name* and *Publisher* are 10791 and 557 respectively, so these features are not suited for One-Hot Encoding. The features *Genre* and *Platform* are of cardinality 12 and 31, however, and seem more suited. In order to lower the cardinality of the *Platform* feature, the feature values were grouped together according to their mean sales per game. The values were grouped into 5 new subcategories: *High Selling*, *Mid-High Selling*, *Mid Selling*, *Low-Mid Selling* and *Low Selling*.

3 Task 3

3.1 (a)

The dataset was split into training and test subsets using the function train_test_split from sklearn with a partition of 80 % for the training set and 20 % for the test set. The cross validation method with five folds was used to train the model, as this allows for both a larger training and test set in comparison to something like a 70/15/15 split of a training, validation and test set. The cross validation method also provides a more reliable measure of the performance of the machine learning algortihms, as it averages the performance over five combinations of training and validation data, making it less prone to randomness in the partition of the dataset.

3.2 (b)

The three machine learning algorithms Linear Regression, Decision Tree and Random Forest were all evaluated for the dataset, in order to see which model had the best performance on the datset.

3.3 (c)

The performance of the Linear Regression, Decision Tree, and Random Forest algorithms was compared on the dataset. Without any parameter tuning, the linear regression performed the best, having a Mean Absolute Error (MAE) of 0.0030. The MAE using the Decision Tree algorithm was 0.0148, and for the Random Forest algorithm the MAE was 0.0105. The Mean Squared Errors (MSE) and R-Squared Scores (R2) are given in Table 1. It is worth noting that the linear regression model has a MAE about three times as low as the random forest model, while the random forest model

performs better than the decision tree model.

Table 1: MAE, MSE and R2-score of the linear regression, decision tree and random forest model

Algorithm	Linear Regression	Decision Tree	Random Forest
Mean Absolute Error	0.0030	0.0148	0.0105
Mean Squared Error [10^{-5}]	2.662	212.90	91.88
R-Squared Score	0.999	0.910	0.962

When checking the performance of the algorithms when only using the relevant numerical features, we get that the MAEs for the Linear Regression, Decision Tree and Random Forest are 0.0030, 0.0131 and 0.0097 respectively. The MSEs and R2s are given in Table 2. It is worth noting that the performance of all the algorithms improves when removing the categorical variables.

Table 2: MAE, MSE and R2-score of the linear regression, decision tree and random forest model

Algorithm	Linear Regression	Decision Tree	Random Forest
Mean Absolute Error	0.0030	0.0131	0.0097
Mean Squared Error [10^{-5}]	2.656	164.55	81.27
R-Squared Score	0.999	0.931	0.966

3.4 (d)

The linear regression algorithm does not have any parameters, and so there is no need to adjust any hyperparameters here. For the decision tree and random forest however, parameters such as maximum number of nodes and numbers of estimators can be tuned to find the optimal parameters for the algorithm.

The decision tree was tuned for the maximum number of nodes only. The optimal number of nodes were found to be equal to the size of the validation sets at about 3500, as the MAE reached a constant value here at 0.0136, which is higher than for the regular decision tree. This suggests that something was wrong in the implementation

of the parameter tuning.

For the random forest model, the only parameter which was tuned was the number of estimators (number of trees used in the model). The optimal number of estimators was found to be 1350, having a MAE of 0.0097. This was fairly similar to the regular model without parameter tuning, although the one with parameter tuning was marginally better (not seen with our number of significant figures).

4 Task 4

4.1 (a)

The linear regression and the decision tree, as well as the random forest model with its best-found parameters were fitted to the training data and then used to predict the target variable using the test features dataset.

4.2 (b)

The predicted values of the models were compared to the target variables in test set. The linear regression model was found to have a MAE of 0.0031, while decision tree model had a MAE of 0.0139, and the random forest model had a MAE of 0.0096. This is also presented in Table 3.

Table 3: MAE the linear regression, decision tree and random forest model on the test set.

Algorithm	Linear Regression	Decision Tree	Random Forest
Mean Absolute Error	0.0031	0.0139	0.0096

4.3 (c)

The linear regression had a significantly better performance that the decision tree and random forest models with a MAE which was about three times as low as for the others. This may seem a bit counterintuitive, as it seems like a rather simple model, but if there are relatively linear relations between the features and the target variable, the linear regression can perform really well. Is is also no surprise that the random forest model performed better than the decision tree, as the random forest model is just multiple decision trees together, giving more expressibility.

The performances of all the models seemed to be fairly good, the standard deviation of the European sales values was about 0.5, which is two orders of magnitude larger than the linear model MAE, and between one to two orders of magnitude larger than the tree and forest MAEs.

5 Task 5

5.1 (a)

The vgsales dataset was filtered by removing the samples which contained missing values, and removing the samples which lied outside the 1st to 99th percentile of each feature variable. The features *Year* and *Name* were dropped, as they didn't seem to have any high influence on the European sales. The numerical features were all normalized such that they had values between 0 and 1 in order to ensure that features with large values didn't dominate the learning, and to improve performance. The values of the categorical variable *Platform* were grouped into subcategories in order to perform a One-Hot encoding along with the *Genre* feature.

The dataset was divided into a training set consisting of 80 percent of the samples and a

training set consisting of 20 percent of the samples. Cross Validation was used to train the algorithms, and the Mean Absolute Error, Mean Squared Error and R-Squared Score was obtained for the three algorithms. Although the One-Hot encoding were thought to improve performance, it turned out to actually slightly worsen it compared to just using the scaled numerical features. This could be due to the categorical features having an insufficient correlation to the target variable, causing mostly additional noise in the dataset rather than giving additional information about the target variable, which would reduce the performance. It could also possibly be due to the rather high amount of new coulmns being added, all being rather sparse as well. Adding 16 sparse columns to just 10 other columns could cause some overfitting.

The dataset was split 80-20 into a training and test set. Cross validation was used to assess the performances of linear regression, decision tree and random forest. The linear model performed the best, even after the parameter tuning of the two other models. The parameter tuning was probably implemented in the wrong manner however, as no kind of overfitting was seen when increasing the expressibilities of the models. This could be due to looking at the wrong kinds of hyperparameters, or looking at the wrong values of the hyperparameters.

The models were used to predict the European sales of the test set, and was afterwards compared to the actual European sales of the test set. The MAEs of the linear regression, tree and forest models were found to be 0.0031, 0.0139 and 0.0096 respectively. The models were able to predict the European sales to a reasonable degree.

5.2 (b)

For future improvements of the model, one could for starters use a more sophisticated method of hyperparameter tuning. In this project, only one parameter was adjusted for the decision tree and the random forest, and the implementation was probably a bit

wrong, as the models showed no tendency of overfitting with increased expressibility. Trying a range of different variables could find a more optimally tuned model. Different algorithms could also be tested for the project, such as SVMs or neural networks.