Retail Data Analytics
Udacity Machine Learning Engineering Nanodegree
Capstone Final Report

Patrick Brus
Email: brus.patrick63@gmail.com

I. Definition

Retail Data Analytics (RDA) is used nowadays from shops in order to better predict the amount of articles, that might get sold and therefore to better estimate how much articles should be produced. This is very important, because the amount of sold articles can vary largely during the year. For example people are tend to buy more things before Christmas then during a normal, not holiday, week. This can be easily seen on the Amazon quarterly revenue on Statista [1]. The quarterly revenue of Amazon is always the largest for the fourth quarter, which indicates, that the people are consuming more during the fourth quarter than during the others. This is clear due to the fact that Christmas is within the fourth quarter and also the Black Friday, which leads to large worldwide consume, too. If a shop has too few products before Christmas, he would loose potential income. But if a shop has too much products, too much storage would be required and storage also costs money, so the company would again loose money. RDA can therefore be used in order to try to optimize the production of products, such that there is always an optimal amount available.

A. Project Overview

B. Problem Statement

The goal of this project is to predict the department wide weekly sales for a store. This should then help to optimize the manufacturing process and therefore to increase income while lowering costs. It should be possible to feed in past sales data from a department and to get the predicted weekly sales.

C. Metrics

The target data is numerical data. Therefore, root mean squared error (RMSE) can be used in order to get the best performing machine learning technique. Equation 1 shows the formula for computing the RMSE. In order to get a better understanding of the scale of RMSE, the normalized RMSE can be used. The normalized RMSE can be computed by dividing the RMSE by the standard deviation of the target variable. Additionally, the R^2 score can be used. The R^2 score represents the percentage of the variance a model can represent. The R^2 score computes the squared error of each point divided by the "dummy" estimator error, which is only predicting the mean of all target variables. Equation 2 shows the formula for computing R^2.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{T}(y_i - \hat{y_i})^2}{T}} \tag{1}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{T}(y_i - \hat{y_i})^2}{\sum_{i=1}^{T}(y_i - \bar{y_i})^2} \tag{2}$$

II. Analysis

A. Data Exploration

The RDA dataset from Kaggle [2] is used for this project. The dataset contains historical sales data from 45 stores located in different regions. Each store is further divided into departments. The data itself is stored in an excel sheet. The excel sheet contains three tabs. The first tab contains the data from the stores. The second contains the features and the third contains the sales data.

1) Stores: There is data of 45 stores in total. Every store has it's own type and size, which is also included in the excel sheet. The information contained in the excel sheet is anonymized. Table I contains the statistics of the size from the stores. In total there are three different types of stores (A, B, and C).

	Size of Store
mean	130287.6
std	63825.3
min	34875.0
max	219622.0

Table I
Statistics of Store Size

2) Features: The features are related to a store. Table II contains all available features and a short description of each one, while table III contains the statistics to some of the features. The data for MarkDown1 - MarkDown5 is very incomplete and has to be dropped or different methods for handling missing data have to be applied.

Feature	Description
Store	the store number
Date	the week
Temperature	average temperature in the region
Fuel Price	cost of fuel in the region
MarkDown 1-5	anonymized data related to promotional markdowns
CPI	the consumer price index
Unemployment	the unemployment rate
IsHoliday	whether the week is a special holiday week or not

Table II
FEATURES OF THE DATASET

	Temperature	Fuel Price	CPI	Unemployment
mean	59.4	3.4	172.5	7.8
std	18.7	0.4	39.7	1.9
min	-7.3	2.5	126.1	3.7
max	102	4.5	229	14.3

Table III
STATISTICS OF FEATURES

3) Sales: Each store also has historical sales data stored in the dataset. The sales data was collected from the fifth February 2010 until the first November 2012. Table IV contains all features related to the sales data and table V contains some statistics of the weekly sales.

Feature	Description
Store	the store number
Dept	the department number
Date	the week
Weekly_Sales	sales for the department in the given store
IsHoliday	whether the week is a special holiday week or not

Table IV
FEATURES OF SALES DATA

	Weekly Sales
mean	15981.3
std	22711.2
min	-4989.0
max	693099.4

Table V
STATISTICS OF SALES

B. Exploratory Visualization

This chapter takes a more detailed view on the data and all features. Figure 1 shows a time series analysis of some features. The first row contains the plots for the temperature. The temperature is alternating year by year having it's maximum somewhere around July and it's minimum somewhere in January. The temperature itself doesn't seem to have any influence on the weekly sales. The second row of the figure contains the fuel price. The fuel price had a large increase from January 2011 to July 2011. Afterwards, the fuel price is oscillating up and down. The third row of the figure contains the Consumer Price Index (CPI). This was steadily increasing since begin of the time series. In the next row one can see the Unemployment rate, which is steadily decreasing since the beginning of the time series. The functions of CPI and Unemployment rate make totally sense, because when the people have more jobs, they have more money to buy things and therefore the CPI goes up, because the more demand, the larger the prices. The fifth row of the figure contains the Boolean data of whether it's a holiday week or not. The last row contains the weekly sales data. As one can see, the weekly sales data is not automatically larger if the week is a holiday week. The peaks of weekly sales are in November and December. The peak in November is possibly due to Black Friday, while the peak in December is possibly due to Christmas. In January the weekly sales are the lowest.
Figure 2 shows the stores analysis. The plot on the left shows how many stores per type are available. The stores of type "A" are the most common, while stores of type "C" are the least common. The plot in the middle shows a box plot of the weekly sales per store type. Store type "A" has the highest median of weekly sales, while type "C" has the lowest. This makes totally sense if one takes a look at the right plot, which shows a box plot of store sizes per store type. There one can see that store type "A" is the largest, while store type "B" the second largest and "C" the smallest. So the store size directly correlates with the amount of weekly sales.

C. Algorithms and Techniques

In order to solve the Problem stated in section I-B, machine learning shall be applied. Different state of the art machine learning techniques should be applied and the best performing should be used for the final application. Within the final application, an user should be able to enter a week and a store of interest and to get the predicted weekly sales as output. As Machine Learning algorithms, the following ones shall be applied and evaluated:

- Linear Regression
- Decision Tree Regressor
- Random Forest Regressor
- XGBoost Regressor

The final model should be able to follow the pattern of weekly sales. It should be able to detect the peaks around Black Friday and Christmas and the low values in January.

1) Linear Regression: Linear Regression is a machine learning technique, where it's tried to find a linear function combining all input features in order to get the target value. It's therefore assumed that the relationship between input features $x_1, x_2, ..., x_n$ and the target value y is linear. So a linear regression model takes the form of $y_i = w_1 * x_{1,i} + w_2 * x_{2,i} + ... + w_n * x_{n,i}$ where n is the number of input features and i is the sample number.

2) Decision Tree Regressor: A decision tree can represent linear and non-linear function and tries to build a tree of rules. Each node in the tree contains a condition and each branch the outcome. It can be thought of a simple "if condition then

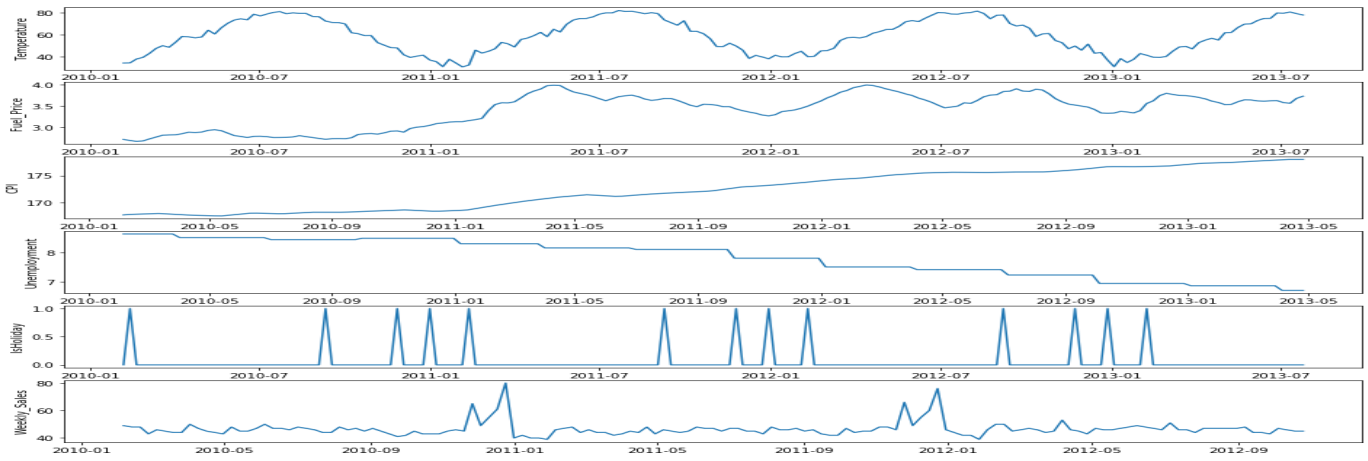

Figure 1. The historical data of features.

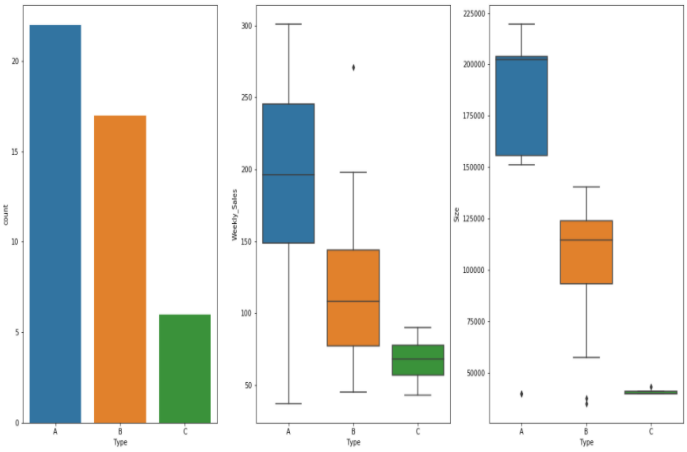

Figure 2. Storewise analysis.

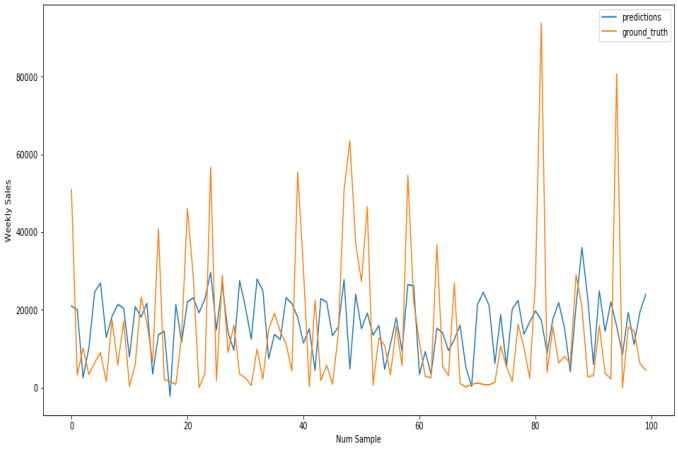

Figure 3. Predictions of Benchmark model.

branch1 else branch2" structure. The decision tree algorithm tries to build these rules in a way that the outcome performance can be optimized. When a prediction should be created, a "walk" through the nodes and branches of the decision tree is made and the final node contains the prediction of the tree. Decision Trees are very popular because of their easy interpretability. The structure of the tree can be easily visualized and the user can directly see the node conditions and the branches and can therefore easily follow the strategy, how the decision tree model creates the prediction.

3) Random Forest Regressor: A random forest creates more decision trees at training time in order to prevent overfitting. During inference time, a random forest then outputs the mean of all decision tree predictions in order to get the final prediction.

4) XGBoost Regressor: The XGBoost Regressor algorithm is an optimized version of the random forest. It uses gradient boosting technqiues in order to increase training speed and prediction performance.

D. Benchmark

The linear regression model is used as benchmark model. It is trained using the scikit-learn library. It has a RMSE of 21469.1 weekly sales, which is very large and tells us, that the weekly sales are not linearly dependent on the input features. Figure 3 shows 100 predictions and their true labels. As one can see, the benchmark model is not able to follow the structure of the weekly sales.

III. Methodology

A. Data Preprocessing

As first pre-processing step the column "IsHoliday" is converted to integer from boolean. Secondly, the store type shall be converted to a categorical column. Store type "A" shall be converted to 0, "B" to 1 and "C" to 2. As next step, the date column is converted to two separate features. The first one is the year, the second one is the week of the year. The week of the year shall be used, because the weekly sales shall be predicted. Afterwards, the high amount of missing values for the markdown columns shall be tackled. In order to remove

the missing values, the iterative imputer from sklearn library shall be used. The iterative imputer models each missing value as function of other features and uses that estimate for imputation. The features and the available values of the markdown columns are used to train a regression model. The missing values are then predicted using the trained regression model. The iterative imputer is only trained on training data in order to avoid data leakage into the hold-out test-and validation data.

B. Implementation

As first step I started to explore the available data in the notebook "1_Data_Exploration.ipynb". There I plotted the first lines of each data frame in order to get more insights on the available columns. Then I used the pandas $describe$ function in order to get the statistics of each column. Afterwards, I checked if there are missing values and if so, how many. This step showed, that there are a lot of missing values in the markdown columns. Table VI shows the exact percentages of missing values. As next step, I started to analyze the data by grouping all columns by date. This should help to get more insights of the features and the weekly sales in comparison to the time of the year. Figure 1 shows the plot of some features over time and chapter II-B shows the discussion regarding these plots. Then I analyzed some data storewise. Figure 2 shows a box-plot of the different store types, the weekly sales per store type and the size of each store type. Again, this figure is discussed in more detail in chapter II-B. After that, I started with the second notebook "2_Create_Train_and_Test_Data.ipynb". This notebook combines all data frames and all data into one final dataset. The pre-processing steps from chapter III-A are applied. Then the complete data set is split into train, test and validation sets. The test set size makes 20% of the total data set, while the validation set size makes 20% of the remaining data. In the end, the training set, validation set and test set are stored in csv files. Additionally, a csv file containing all feature names is created. This shall help the user to easily identify all feature names stored in the training, validation and test csv files. And last but not least I created the training notebook "3_Training_and_Deployment.ipynb". I started with loading the training, validation and test data to S3. Then I trained the benchmark model. The linear regression benchmark model is created and trained using the sklearn library. The code for training is located in the "train_linear_regression.py" script within the source folder, which is also used as entry point for the sagemaker sklearn estimator object in the notebook. Afterwards the benchmark model was deployed and evaluated. In the evaluation, I computed the normalized RMSE, the R^2 score and created a plot with the first 100 test predictions. Additionally, I plotted the first 100 predictions vs true values using the matplotlib library. After that, I trained the decision tree regressor and a random forest regressor. For both, I used the sklearn library and both have separate training scripts in the source folder as entry point. A decision tree and a random forest have a lot of different hyperparameters. In order to find the best matching, I used the skopt library for creating a Bayesian optimizer. The Bayesian optimizer takes as input a sklearn estimator object, a dictionary with hyperparameter names as key and their value ranges as value, an integer indicating how much iterations should be performed and the number of folds used for cross-validation. Figure 5 contains the hyperparameters and their ranges for the decision tree regressor. I used a cross-validation fold size of 5 and the Bayesian optimizer ran for 7 iterations. Figure 6 contains the hyperparameters and their ranges for the random forest regressor. The cross-validation fold size was again 5 and the Bayesian optimizer ran for 10 iterations. The random forest regressor additionally uses all four cpu cores in order to speed up the training. For both trained models, I computed the normalized RMSE and the R^2 score. And as last estimator, I trained the XGBoost regressor. The XGBoost regressor is trained using the already created container from Sagemaker. Again, the XGBoost estimator has a lot of hyperparameters. In order to find the best combinations, I used the "HyperparameterTuner" library from Sagemaker. Figure 4 contains all hyperparameters and their ranges. I trained 20 different XGBoost estimators and fetched and deployed the best performing. After deployment, I again computed the normalized RMSE and the R^2 score, plus the visualization of the first 100 test predictions.

Column Name	Percentage of Missing Values
MarkDown1	51%
MarkDown2	64%
MarkDown3	56%
MarkDown4	57%
MarkDown5	51%

Table VI
MISSING VALUES

C. Refinement

The benchmark model was the first try to get a working predictor. But, as one can see in Figure 3, the benchmark model is not able to follow the underlying structure in the data and to adequately predict the weekly sales. Therefore, a decision tree regressor was trained as second model. The results and analysis of the decsion tree regressor can be seen in section IV. The decision tree regressor already works quite well. In order to have more comparisons, a random forest regressor and a XGBoost model were also trained. The results can again bee seen in chapter IV. All models have more hyperparameters and so Bayesian optimization was used in order to effectively search the hyperparameter space for the best performing combinations.

IV. RESULTS

A. Model Evaluation and Validation

Figure 7 shows the first 100 predictions of the trained decision tree regressor compared to the ground truth labels. As one can see, the decision tree regressor is already able to follow

```
# Define exploration boundaries (default suggested values from Amazon SageMaker Documentation)
hyperparameter_ranges = {
    'alpha': ContinuousParameter(0, 1000, scaling_type="Auto"),
    'eta': ContinuousParameter(0.1, 0.5, scaling_type='Logarithmic'),
    'gamma':ContinuousParameter(0, 5, scaling_type='Auto'),
    'lambda': ContinuousParameter(0,100,scaling_type='Auto'),
    'max_delta_step': IntegerParameter(0,10,scaling_type='Auto'),
    'max_depth': IntegerParameter(0,10,scaling_type='Auto'),
    'min_child_weight': ContinuousParameter(0,10,scaling_type='Auto'),
    'subsample': ContinuousParameter(0.5,1,scaling_type='Logarithmic'),
    'num_round': IntegerParameter(50,1000,scaling_type='Auto')
}
```

Figure 4. Hyperparameter Ranges XGBoost.

```
{
    "max_depth": (5, 15),
    "splitter": ["best", "random"],
},
```

Figure 5. Hyperparameter Ranges Decision Tree.

```
{
    "max_depth": (5, 15),
    "n_estimators": (10, 50),
    "bootstrap": [True, False]
},
```

Figure 6. Hyperparameter Ranges Random Forest.

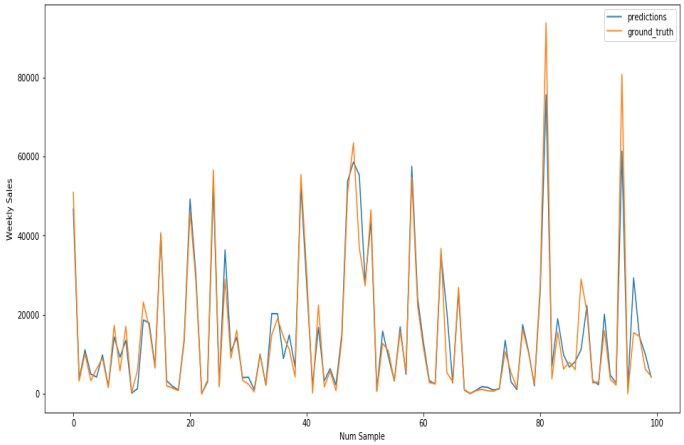

Figure 7. Predictions of Decision Tree Model.

the pattern of weekly sales. Table VII contains the RMSE and the R^2 score. Figure 8 shows the first 100 predictions of the trained random forest model. The random forest is again able to follow the structure of weekly sales, but the random forest regressor has a slightly larger normalized RMSE while having the same R^2 as the decision tree model. Figure 9 shows the first 100 predictions of the trained XGBoost model which had the best hyperparameter combinations according to the bayesian optimization. The XGBoost model also has the lowest normalized RMSE and the highest R^2 score and is therefore the best perfomring estimator of all trained models.

Model Name	Normalized RMSE	R^2
Linear Regression	0.95	0.09
Decision Tree Regressor	0.24	0.94
Random Forest Regressor	0.25	0.94
XGBoost Regressor	0.19	0.96

Table VII
OVERVIEW OF FINAL MODEL SCORES

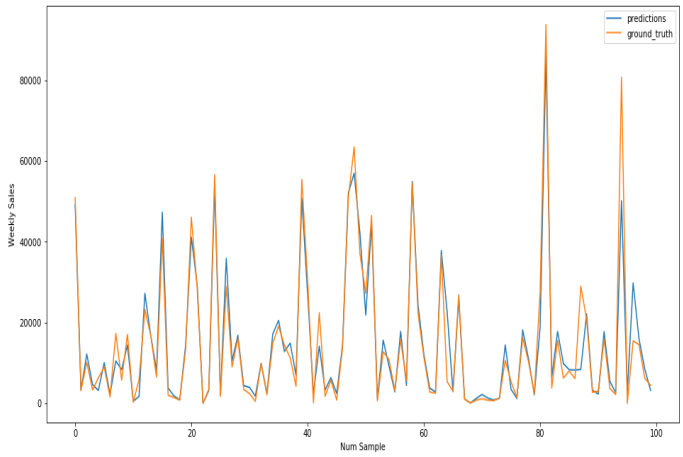

Figure 8. Predictions of Random Forest Model.

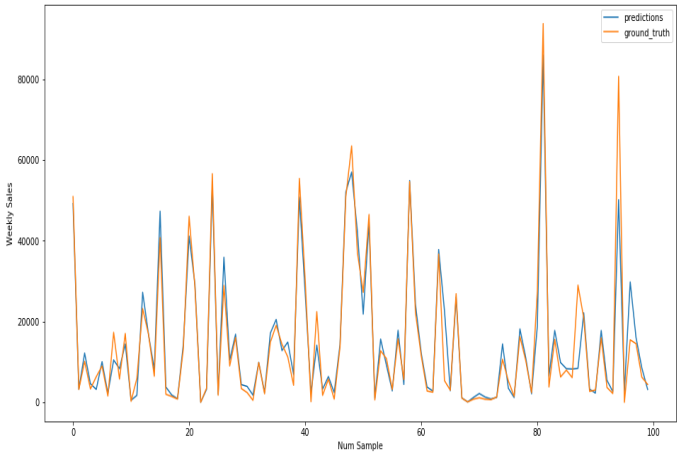

Figure 9. Predictions of XGBoost Model.

B. *Justification*

The XGBoost model has a more than ten times larger R^2 score than the benchmark model. The normalized RMSE is five times smaller than the RMSE of the benchmark model. The R^2 score is at 0.96 which means that the final model can represent 96% of the variance given in the test data set. The RMSE is still at 19% of the standard deviation from the weekly sales. But the range of weekly sales is also very large, with the minimum weekly sales at -4989 and the maximum weekly sales at 693099.4 (c.f. Table V). When taking a look at Figure 9, one can see that the final model has learned the underlying structure and can therefore be used as predictor, but one still has to take potential outliers in the prediction under consideration.

REFERENCES

[1] Statista, "Net revenue of amazon from 1st quarter 2007 to 3rd quarter 2020," https://www.statista.com/statistics/273963/quarterly-revenue-of-amazoncom/, accessed 21-12-2020.

[2] Kaggle, "Retail data analytics," https://www.kaggle.com/manjeetsingh/retaildataset, accessed 21-12-2020.