# Kaggle Data Science Competition: Predict Future Sales

Yuyin Li

Nov 2019

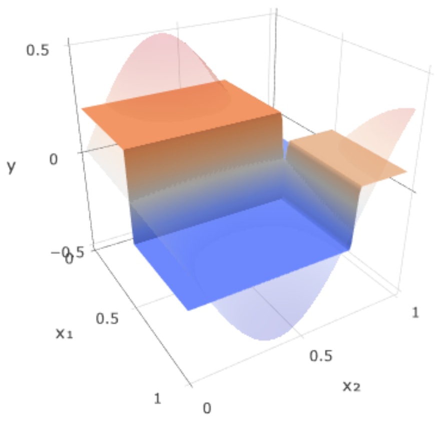
## 1 Introduction

In this project my task is to predict the future monthly sales of a shop based on its previous daily sales. The data sets are downloaded from the Kaggle Data Science Contest Website (`https://www.kaggle.com/c/competitive-data-science-predict-future-sales/data`). A personal goal will be getting a rank higher or equal to 1000 on the Kaggle public leaderboard. To make progress, I consulted data science websites, and the bulk of the code comes from the Coursera Data Science Course Website[Cou].

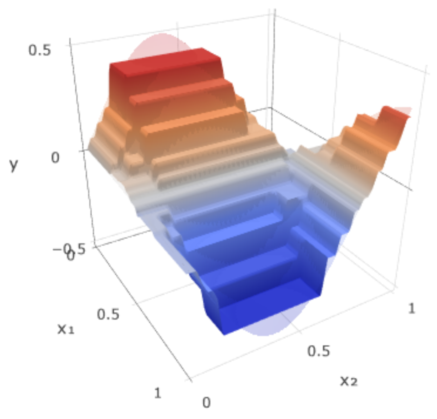
## 2 Approach

### 2.1 Decision Trees & Random Forests

Decision Trees are also known as CARTs, short for Classification and Regression Trees. Though the initial idea was to use the RNN Model, it is later seen that for this time series analysis, such complex implementation isn't necessarily needed. Decision Trees are basically binary trees, and are very popular in machine learning due to its relative simplicity. However, they tend to overfit the training data when grown deep, meaning that inputs out of the data sample will very likely be wrongly classified. The Random Forest Model is a further and higher level implementation of the Decision Tree Model. It grows not one, but multiple trees, and gathers them up into what is called an ensemble of trees, or in literal terms, a forest. By using multiple trees, the model can determine the result more accurately through the summation of the individual predicted results of the trees. The depth of the trees determine how well the trees fit into the desired function.[Rog]



(a) Tree with depth 2



(b) Tree with depth 6

As can be seen from the two figures above, the deeper the tree, the better it fits into the function. This will work very well for the training data, and can potentially get to a 100% accuracy rate while training; however, this might cause overfitting, and the model will very possibly not have a high accuracy while dealing with the testing data and real data.

## 2.2 AdaBoost

Adaptive Boosting is a common supplemental training algorithm used in machine learning. This algorithm boosts the accuracy of the random forest model, as it selects out "stumps", or 1 layer deep trees, from the forest, applies a certain voting power to each stump according to their credibility in the training data, and then stacks them back together so that the results are less biased. For this project, I used XGBoost, short for Extreme Gradient Boosting, a library used for gradient boosting. Its outstanding execution speed and model performance not only make training faster, but also increases the accuracy of the trained model. [Rog]

## 3 Experimental Setup

### 3.1 Data Pre-processing

The first step to machine learning will be pre-processing the data sets. There are 6 columns in the main data set for training, which are the date, date block number, shop id, item id, item price and item count per day, respectively. The date, written in a DD/MM/YY form, is clearly not suitable for the machine to learn from. Therefore, I converted it into Unix Time, a time format that counts the seconds starting from 1970 Jan 1st. Outliers are another thing to look out for. Since the data is collected from a real world shop, it is possible that during collection incorrect digits are inputted, causing the prices and sales of various items to be unrealistic. After box plotting the data, I removed 2 items with inconsistently high sales and 1 item with a extremely high price. The last step is filling out the blanks with the median of the respective data columns.

### 3.2 Baseline Model

Before implementing the boosted regression tree model, I first tried using a simple model based on the Markov process assumption, which states that the probability distribution of future events will be solely dependent on present events, i.e.:

$$\mathbf{P}(X_{n+1} = x_n | \{X_i = x_i\}_{i=0}^{n}) = \mathbf{P}(X_{n+1} = x_{n+1} | X_n = x_n)$$

Based on this assumption, the machine produced what I refer to as a "baseline prediction". The expected prediction should be quite similar to the averaged value of the original sales curve, as shown below.

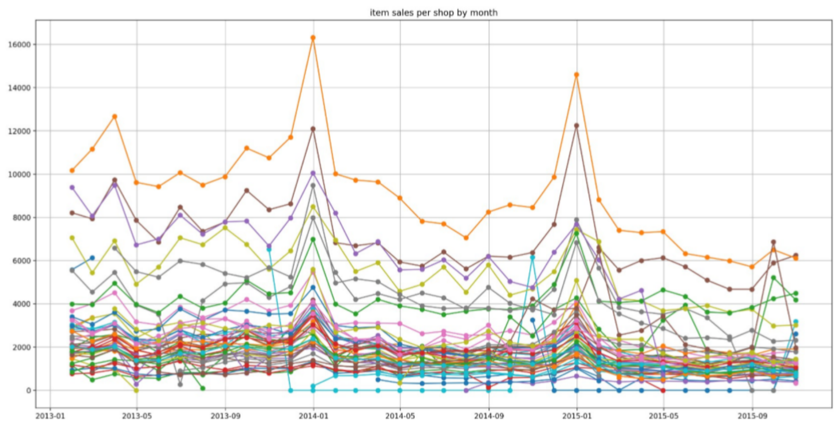


Figure 2: Item sales per month

Uploading it onto Kaggle resulted in an RMSE (Root mean squared error) of 1.16115, ranking 1000+ on the public leaderboard (Note: the leaderboard ranking changes as the competition is still ongoing and submissions from other Kagglers will have certain influences).

### 3.3 Feature Engineering

The next step to regression tree training is feature extraction. Certain types of data, known as features, are manually identified and listed as classification tags. The regression tree is then built and trained based on the features extracted from the datasets. In this type of time series analysis question, lag features are a must-have. These are features created through shifting the dataset columns with respect to time, creating a potential now and then relation. Using the shift function of the Pandas library, I was easily able to create lag features to use in regression tree growing. There are a total of 39 features extracted.

### 3.4 XGBoost Regression Model

After final preparations, the data is ready to be fed into the XGBoost Regression Tree. Using the Pickle library, I read in all the extracted features, and through the XgBRegressor fit function, I was able to fit the data into the model. I made many choices in selecting the features, which lead to different RMSE values, and thus different rankings. The results will be discussed in the following section.

## 4 Result & Analysis

As expected, the final boosted regression tree had a lower RMSE than the baseline model. However, after fiddling with the different features, I realized that feeding all 39 features into the model didn't work well as feeding in 35. The feature importance graph tells me which feature influences the prediction more, and after consideration, removing 4 of the lowest features gives me the best RMSE value of 0.91187.

## 5 Conclusion

In this project, I implemented a baseline model to define a mininum RMSE value of 1.16115. After consideration of which model to use, I chose the XGBoosted Regression Tree Model, which resulted in an RMSE of 0.91187, having a best rank of 1007 on the Kaggle Leaderboard.

## References

[Cou] Coursera. *How to Win a Data Science Competition: Learn from Top Kagglers.* URL: `https://www.coursera.org/learn/competitive-data-science`. (accessed: 11.25.2019).

[Rog] Alex Rogozhnikov. *Gradient Boosting explained[demonstration].* URL: `https://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html`. (accessed: 11.25.2019).