# CSCI567 Team-54 Final Report

**Chutian Sun**
University of Southern California
chutians@usc.edu

**Hoseung Lee**
University of Southern California
hoseungl@usc.edu

**Kuo Liu**
University of Southern California
kuol@usc.edu

**Ruiqi Yu**
University of Southern California
ruiqiyu@usc.edu

## Abstract

In this project, we apply the machine learning techniques we have learnt to a "real life" situation. We are given a bunch of datasets, and our purpose is to predict the sales of given stores in the future.

## 1 Data preprocessing and cleaning

### 1.1 Datasets

#### 1.1.1 Train.csv

This data set contains training data, including information about store_nbr, family, and onpromotion as well as the target sales.

Store_nbr: categorical

family: categorical

onpromotion: categorical

sales: continuous

#### 1.1.2 Stores.csv

This dataset contains store information, where store_nbr identifies a store, about city, state, type and cluster of a store

store_nbr, city, state, type, cluster: categorical

#### 1.1.3 oil.csv

This dataset contains the oil price with the time since 2013-01-01 till 2017-08-31

Date: categorical

dcoilwtico: continuous

#### 1.1.4 Holiday_events

This dataset contains info about the holidays. Colunms are date, type, locale, locale_name, description and transferred.

Date, type, locale, locale_name, description, transferred: categorical

### 1.2 Data preprocessing and cleaning

#### 1.2.1 Null values and missing entries

Some datasets contains Null values and missing entries. For some dates between 2013-01-01 and 2017-08-31, we do not know the oil prices on that date. Therefore, we fill in those null values by using the date around that date as an approximation.

#### 1.2.2 Features we use in respective models

Moving average: We actually do not want to use the oil price as a feature in our model. As a result, we create a new feature "moving average", which is the average oil price across 7 days.

Workday: We create a feature, "workday", which indicates whether the given date is a workday.

Day of Week: a number representing monday to sunday

Trend: a natural number gong from 1, representing each date using integer.

Type_event, type_Holiday, type_Transfer: Only the holidays that are national are considered. There are a few types of holiday: They are bridge, workday, transfer and holiday.

oil price windows/lag feature:

month: 1, 2, 3, ..., 11, 12

day_of_month: 1...31

day_of_year: 1...365

week_of_month: indicate which week in a month

week_of_year: indicate which week in a year,

year: indicate which year

quarter: indicate which quarter in a year

is_month_start: indicate whether the date is at the beginning of a month

is_month_end: indicate whether the date is at the end of a month

is_quarter_start: indicate whether the date is at the start of a quarter

is_quarter_end: indicate whether the date is at the end of a quarter

is_year_start: indicate whether the date is at the start of a year

is_year_end: indicate whether the date is at the end of a year

season: 0: Winter - 1: Spring - 2: Summer - 3: Fall

wageday: indicate whether the date is when people get paid

school_season: indicate the month when school is active

0_sales: indicate the families which have 0 sales

product family average sales: The average sales of a family

bigholiday: We select the dates on which national holiday happens in the holiday_events.csv

#### 1.2.3 data cleaning

1. date 12/25 is dropped because training data does not have this date.

2. remove those rows which have dates before the stores opened

## 2 Learning algorithm and Model

### 2.1 Random Forest (RF)

#### 2.1.1 Explanation of RF

Random forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned.[1]

#### 2.1.2 Reasons of choosing RF

According to the coding the professor gave us before, we also considered using the random forest method to do this project. As we know, the random forest has very high accuracy, can effectively run on large data sets, be able to process high dimensional data without dimension reduction. Further, the introduction of randomness makes it difficult to over fit. Also, the training speed of RF is fast and good results can be obtained even for the default value problem.

#### 2.1.3 Data processing and parameters selection

During data processing, we integrated holidays' data into calendar. We used 'get_Dummies' (pandas package) to map the type of holidays_events file and the day of week in the original calendar. We got ten parts (features from National) including 'dow_1', 'dow_2', 'dow_3', 'dow_4', 'dow_5', 'dow_6', 'type_Additional', 'type_ Event', 'type_Holiday' and 'type_Transfer'. Through get_Dummies, we can also convert strings into floats and then provide accuracy for later building prediction models.

For random forest simulation, we selected 250 weak classifiers (n_estimators). The larger the value is, the better the accuracy is. However, when estimators are greater than a specific value, the improvement is very limited. Thus, we chose 250 as the number of weak classifiers. And we chose 2022 random states (random_state).

#### 2.1.4 Comparison with Fourier and without Fourier feature

We added Fourier feature before random forest simulation, and then fitted the data with RF model. We first analyzed the performance of the random forest without Fourier feature which the error rate is 0.44536. After adding the Fourier feature, the error rate of the random forest decreases to 0.44519. Therefore, we can see that by putting seasonality (Fourier feature) into RF model can slightly improve the prediction result.

### 2.2 Neural Network (NN)

#### 2.2.1 The Basics of the NN Model

One of the methods we used for sales modeling was a neural network. We included the following variables in our model: the pay day, the month, the day of the week, significant holidays, weekends, and the average weekly oil price. The pay day is the 15th and last day of each month, which is significant because this is when public sector employees in Ecuador receive their paychecks. The theory is that people will purchase more products on pay days, making it a significant variable in predicting store sales. The month and day of the week variables capture the sales pattern throughout the calendar year. For example, in America, sales are high in December due to the holiday season. These monthly and weekly patterns likely also exist in Ecuador, although their effects may be different. Holidays are a more specific version of time-based variables, where purchases are expected to be higher. These variables are also significant in predicting sales. The oil is significant because Ecuador's main export is oil, with 58% of its exports being oil. The mean weekly oil price should have a significant impact on Ecuadorian's purchasing power.

For our neural network we used a regression neural network. It is identical to a standard neural network except it is used more for continuous values such as sales price. Just like a standard neural network, a regression NN has multiple layers of connected neurons and these neurons have weights that are used to make predictions on the data. These weights are adjusted during the model training

process. The main difference of the Regression NN compared to a standard one is the activation function used. The output activation function of a standard NN is usually a sigmoid or softmax function which displays the probabilities of the different classifications. For a regression neural network, we often use a linear activation function which allows us to make predictions in the form of continuous values.

#### 2.2.2 Results of NN and analysis

The final neural network for this model had 2 hidden layers, both with 64 neurons. The learning rate of the model was 0.07. The other parameters were set to the default parameters. The accuracy obtained using this model was 0.51 according to kaggle's metrics. This was worse than the simple linear regression model accuracy (given in professor's code) which had an accuracy of 0.45. This was problematic especially given that the original simple linear regression model only had weekend and oil as variables for the regression. This suggests several potential problems with the RNN, namely: possible overfitting, over complex model, need more hyperparameter tuning, etc. Dealing with this issues was a problem for two main reasons:

1. We didn't have a validation set to measure the accuracy of the model accurately, we only relied on the squared mean error of the data to check for accuracy.
2. The neural net model took a relatively long amount of time to run, 20-30 minutes. This was the 2nd most time intensive model after lightGBM model which took hours to run.

Number 1 is a problem because the least squares error is not a reliable measure to tune hyperparameters due to overfitting concerns.

Number 2 was a problem because each iteration of the hyperparameter tuning would take a long time, requiring us to wait up to half an hour to see the efficacy of any given hyperparameter setting we try out.

With this we concluded, at least by itself, that regression neural networks wouldn't be sufficient (at least by itself, with our current hyperparameters, and variables used) in increasing our sales prediction accuracy.

### 2.3 LightGBM

#### 2.3.1 Introduction

Gradient Boosting Decision Tree (GBDT) is a popular machine learning algorithm and has quite a few effective implementations. Although many engineering optimizations have been adopted in these implementations, the efficiency and scalability are still unsatisfactory when the feature dimension is high and the data size is large. A major reason is that for each feature, they need to scan all the data instances to estimate the information gain of all possible split points, which is very time-consuming. LightGBM (lgbm) is a GBDT open-source tool enabling highly efficient training over large-scale datasets with low memory cost.[2]

#### 2.3.2 Dataset construction

For each product family, we created a training set from 2013-01-01 to 2017-07-31, a validation set from 2017-08-01 to 2017-08-15, and a testing set from 2017-08-16 to 2017-08-31. The number of features is 38, including store ID, promotion, holiday/event/workday, oil statistical features, and some time-related features.

#### 2.3.3 Learning process

First, we used all of the features to train an lgbm model and output the features' importance rank. Based on the order of rank, we iterated over all features, computed the RMLSE of the model in the validation set, and confirmed the best combination of features using the early stopping condition. Since the hyperparameters, such as *num_leaves*, *bagging_fraction*, *feature_fraction*, and *num_round*, have a big impact on the performance of model. So we used hyperopt (parameter tuning package) to find the best hyperparameter values. Finally, using the model with the best feature combination and best hyperparameter values to predict the sales of the testing set. The public score is 0.5164

### 2.4 NN +RF

#### 2.4.1 Reasons for choosing NN+RF

As mentioned in the previous section, the Regression Neural Net wasn't very accurate for the purpose of predicting sales as it underperformed simple linear regression models. As such, one of the methods we tried was to combine the Neural Net method with the Random Forests method, which had the highest single model accuracy. The plan was to look at the least mean square errors and choose the model that had a better accuracy to train specific product categories. It should be noted that the parameters for the RF and the RNN model are identical to the hyperparameters use in their solo models. This is visualized in the chart below:

| family | neural net | trees | | |
|---|---|---|---|---|
| AUTOMOTIVE | 0.340868 | 0.267844 | 0.267844 | Forests |
| BABY CARE | 0.144324 | 0.089108 | 0.089108 | Forests |
| BEAUTY | 0.290802 | 0.232869 | 0.232869 | Forests |
| BEVERAGES | 0.021343 | 0.480921 | 0.021343 | NN |
| BOOKS | 0.038854 | 0.059981 | 0.038854 | NN |
| BREAD/BAKERY | 0.018357 | 0.356189 | 0.018357 | NN |
| CELEBRATION | 0.322203 | 0.257322 | 0.257322 | Forests |
| CLEANING | 0.062261 | 0.429009 | 0.062261 | NN |
| DAIRY | 0.022622 | 0.378649 | 0.022622 | NN |

Figure 1: Model selection

The neural net results were visualized in red and the tree results were visualized in green. We trained and ran both models once to get these accuracy measurements. Afterwards, we retrained the two models only on the products they outperformed in and then ran them again to get the final results. Unfortunately, the result for this new model was barely better than linear regression by itself, with a Kaggle accuracy measure of about 0.47. We theorized that this was because the neural network had a tendency to overfit the data more than the random forest method.

#### 2.4.2 Results and Analysis

The regression model had a mean least squared error of 0.186 average over all the different product categories. The RF model had an average of 0.315. This is curious because the random forests model had almost double the mean error but it performed better than the regression neural network model as it had a 0.44 accuracy measure compared to 0.5 for the neural network. This suggests that if a given category has a similar mean least square error, the RF model is more likely to perform better on the test set because it is more robust against overfitting compared to the neural network model. So we ran this model again, except this time, a category was only given to the neural network model if it was >0.1 better than RF instead of simply having a lower least square value. After this, while there was an improvement, the Kaggle accuracy measure was 0.44 which is still not better than the simple linear regression model.

### 2.5 LR + RF

#### 2.5.1 Ridge Regression

Ridge regression is a type of regularized linear regression that aims to prevent overfitting by adding a penalty term to the cost function. This penalty term is called the "L2 regularization term" and it is the sum of the squared coefficients of the model. By adding this regularization term, the cost function is no longer just the mean squared error of the model but also includes the L2 regularization term. This has the effect of shrinking the coefficients of the model, which can help to prevent overfitting and improve the generalization of the model.

We used each day's data from 2017-04-01 to 2017-08-15 as a sample in the training set. The sample has 9 features including trend, 6 Fourier statistical features, moving average of oil price, and workday. The labels are sales per product family per store from 2017-04-01 to 2017-08-15. So the size of samples is (137, 9), size of labels is (137, 1782).

The training set's Root Mean Squared Logarithmic Error (RMSLE) is 0.27086.

#### 2.5.2 Random Forest

As for random forest, we used the same process as 2.1. The training set's Root Mean Squared Logarithmic Error (RMSLE) is 0.27086.

#### 2.5.3 Model Comparison and Selective Prediction

After the above two steps, we got the RMSLE of each product family in the training set. Based on the RMSLE, we selected the better predictor of each product family's sales between LR and RF.

For ridge regression, it performed better on the product family list as shown below.

*'BABY CARE', 'BEVERAGES', 'BOOKS', 'BREAD/BAKERY', 'CLEANING', 'DAIRY', 'DELI', 'EGGS', 'FROZEN FOODS', 'GROCERY I', 'HOME AND KITCHEN II', 'HOME APPLIANCES', 'HOME CARE', 'MEATS', 'PERSONAL CARE', 'PET SUPPLIES', 'PLAYERS AND ELECTRONICS', 'POULTRY', 'PREPARED FOODS', 'PRODUCE'*

For random forest, the corresponding list is shown below.

*'AUTOMOTIVE', 'BEAUTY', 'CELEBRATION', 'GROCERY II', 'HARDWARE', 'HOME AND KITCHEN I', 'LADIESWEAR', 'LAWN AND GARDEN', 'LINGERIE', 'LIQUOR, WINE, BEER', 'MAGAZINES', 'SCHOOL AND OFFICE SUPPLIES', 'SEAFOOD'*

Then, we retrained two models on the training set of their corresponding product families and predicted product family sales in the next 16 days. The public score of prediction is 0.42939

## 3 Result and Insight

The public score of each model is shown as Table 1. RF+LR has the highest score. During the competition, we have some insights as follows:

1. Data set construction will influence model's prediction accuracy and training efficiency.
2. A large number of features may not help models learn the pattern of data, but a few representative features resulting from the analysis do.
3. A validation set would have been very convenient to tune hyperparameters
4. Combining different models to get accuracy is good because they benefit from each other's strengths to make on a better model
5. Time series (Fourier) is very useful in time-based prediction tasks
6. Complicated models aren't always better (simple linear regression with just oil + weekday) had a better result than all of our models except RF
7. NN and lightGBM have a tendency to overfit the data

Table 1: Models' Public Score

| Model | Public Score |
|---|---|
| RF | 0.44519 |
| NN | 0.51353 |
| LightGBM | 0.5164 |
| RF+NN | 0.45201 |
| RF+LR* | 0.42939 |

## References


[1] Wiki. Random forest. Available: *https://en.wikipedia.org/wiki/Random_forest.*

[2] Microsoft. (2021, November 21). LightGBM. Microsoft Research. *https://www.microsoft.com/en-us/research/project/lightgbm*