CSCI 567 Final Project: CSCI567_id22

Xinyue Cui Victor Yip Sun Hui Thomas Huang Pinyi Wang
University of Southern California
3650 McClintock Ave, Los Angeles, CA 90089
xinyuecu@usc.edu victoryi@usc.edu tyhuang@usc.edu pinyiw@usc.edu

Abstract

This report covers the detail of our solutions and previous attempts for CSCI 567, USC's graduate course in Machine Learning. For our final project, we were tasked with exploring a real life machine learning problem by participating in the Kaggle competition **Store Sales - Time Series Forecasting**. The goal of the competition is to use time-series forecasting to forecast store sales on data from Corporacion Favorita, a large Ecuadorian-based grocery retailer. Additionally, the final project report must follow the NeurIPS format. This report concludes with a detailed view of our final submission, as well as our thoughts and potential future improvement.

1 Data manipulation

As part of the dataset, we were provided with five data files.

train.csv - which comprised of features store_nbr, family, onpromotion, and sales over a period from 2013/01/01 to 2017/08/15. **test.csv** - which had the same features as the training data for the testing period, 15 days after the last date in the training data.

Additionally, we were also provided with three sets of data which may be useful for us. **stores.csv** - which stored Store metadata such as its location. **oil.csv** - which consisted of the daily oil price during the training and test data-frame, we were provided with additional information which stated that "Ecuador is an oil-dependent country and it's economical health is highly vulnerable to shocks in oil prices". **holidays_events.csv** - which consisted of holiday related metadata.

By the nature of Machine Learning models as well as the no free lunch theory, each model had drastic changes in performance depending on the data engineering, and there was no one-size-fits-all data manipulation solution which worked for all models. For each model, we will highlight a couple of methods which helped us reduce our Root Mean Squared Logarithmic Error (RMSLE), the scoring metric utilized by the Kaggle competition.

If a Machine Learning model benefited from a more specific data-split and data processing method, we will mention it in their respective sections.

1.1 Data exploration

Understanding the data and what should we be looking for was an integral part in developing our model. To construct a model to accurately predict the future sales of each store and each family, we adhered to the following two steps to perform analysis and identify trends not only within the sales data but also explain how other data such as holidays, oil prices, and customer habits can impact the performance of sales.

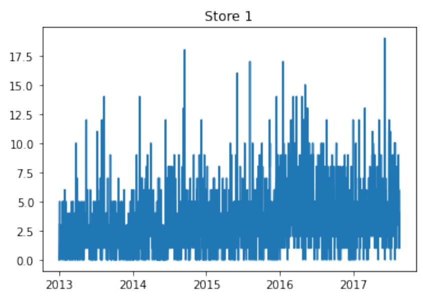

Figure 1: Store 1 Automobile sales vs time

First, we generated plots for the data of interest in relation to time, an example shown in figure 1. From these generated graphs, we observed patterns in the sales data that can be attributed to yearly, monthly, biweekly, weekly, weekday vs weekend trends, which will later be one of the core features that we focus on to optimize our final prediction.

Secondly, in an attempt to explain peaks in the sales data, we consulted non-sales data (holidays, oil prices, store, onpromotion, transactions) to search for correlation between non-sales and sales data. From our initial exploration, we were able to find that national holidays performed best at indicating the peaks in sales.

1.2 Train/Validation/Test split

As Kaggle limits our daily submission quota to five, we had to produce a validation set which could provide us an estimate of the performance of our model.

Finding the correct split proved to be a challenge, since the training set provided covered such an enormous period of time, the traditional 4-3-3, train-validation-test split would be ineffective. Therefore, we tried many different split variation, but ultimately, we settled on using the period of 2017/08/01 - 2017/08/15 as our validation set, as the data tested from this period of time is most similar to the test period 2017/08/16 - 2017/08/31.

Henceforth, "Validation score" would be the RMSLE produced by the model, predicted sales value between the validation period of 2017/08/01 - 2017/08/15 calculated against its true value provided by the dataset.

2 Tested Models

2.1 Linear regression

Our additional attempts of the Linear Regression model is a direct continuation of Problem 5 from Homework 4, maintaining the feature engineering completed for Part I of the problem. Where we computed the moving average of oil prices with a window size of 7 as data_oil["ma_oil"] and created an extra feature calendar["wd"] where we indicated whether each date is a workday.

Model	Train period	Validation score	Test score
Linear Regression	2017/04/01 - 2017/07/31	0.54696	0.45807
Linear Regression (order = 2)	2017/04/01 - 2017/07/31	0.58981	0.67697
Linear Regression w/workday	2017/04/01 - 2017/07/31	0.43526	0.45732
Linear Regression (seasonal=true)	2017/04/01 - 2017/07/31	0.47749	0.44617

Figure 2: Significant Linear Regression results (All models w/ 'ma_oil','wd' from HW4P5)

Post completion of Homework 4, we conducted small changes on the Linear Regression model in hope to produce a better understanding of the data, which generated varying degrees of success. Including workday features and setting seasonality to true both generated a better test score, while setting order to 2, simultaneously including both linear and quadratic terms, yielded a significantly worse test score.

2.2 Facebook prophet

Prophet is an open source time-series model developed by Facebook, and is itself used in many applications for producing reliable forecasts for planning within the company. Prophet is often compared to another widely used time-series model named ARIMA (see Appendix A.1), but is preferred by time-series novice for its ease-of-use and reliable performance out of the box.

Our team began experimenting with Prophet soon after Linear Regression as it is an easy entrance point into time-series modelling, in order to familiarize ourselves with time-series forecasting.

Prophet is renowned for its good performance to obtain a reasonable forecast on data without much data processing, it is a model known to be robust to outliers, missing data, and dramatic changes in our time series. Which were characteristics exhibited by our dataset.

The prophet model accepts three parameters:
Trend: Gives us the choice of trend line over the data.
Seasonality: Seasonal effects such as fourier order, period, and number of seasons.
Holiday and events: Predictable trends within the dataset, such as the magnitude 7.8 earthquake which struck Ecuador on April 16, 2016.

Model	Train period	Validation score	Test score
Prophet	2013-/01/01 - 2017/07/31	0.57201	0.54291
Prophet	2017-/01/01 - 2017/07/31	0.520	0.56267
Prophet w/Holiday	2013-/01/01 - 2017/07/31	0.57633	0.54841

Figure 3: Significant Prophet results

Our first attempt at the Prophet model yielded a test score of 0.54291, which was worse than our best Linear Regression score of 0.44617. Subsequent attempts of using the model could also be seen from the figure above, where the test score did not differ by more than a hundreth of a decimal point. This set of results made it clear to us that Prophet would not produce the level of accuracy we desired.

2.3 LGBM regressor

One of the most effective model we tested was the LightGBM model, more commonly known as LGBM, this model is a gradient boosting framework that uses tree based learning algorithms. Initially developed by Microsoft, LGBM has a track record of good performance in other Kaggle Machine Learning competitions because of its robust handling of a a large variety of data types and fine-tunable hyper-parameters.

Credits:
The use of LGBM and its methods were inspired by kaggle notebook from Ferdinand Berr [1]
Where we experimented with the methods described in this notebook and subsequently improved its accuracy with our fine-tuning.

In order to produce the most accurate model for each product family, we generated 33 separate LGBM models for each family.

Data processing:
1. We applied $\log(x + 1)$ transformation and MinMaxScalar to all numeral features and the targeted output - "sales".
2. Added Holiday features - a binary value which indicates if the current date has a holiday or event for each store, based on location.
3. Added Promotion features - daily promotion, and 7 and 28 days moving averages.
4. Added Oil features - daily oil prices, and 7 and 28 days moving averages.
5. Date features - year, month, day, dayofyear, dayofweek, weekofyear, linear_increase.
6. Sales feature - past target or predicted "sales".

In addition to feature engineering, we also experimented with different combinations of past and future features with different ranges, using Past vs Future covariates [2].

Post processing:
1. Zero forecasting - manually set our sales predictions to 0's when the last 3 week's sales are all 0.
2. Prediction spike fixing - After a manual inspection of our sales prediction trends, we noticed that for the familiy "SCHOOL AND OFFICE SUPPLIES", the LGBM model occasionally predicted an abnormally high values on the test dates, we suspect that since we are only fitting one model per family, the model has picked up trend from other stores which included a sudden spike of sales around July and August - potentially due to the beginning of the school season.

After completing the steps above, we produced the following results. Note: the training period was kept consistent over all these models, which was 2013-/01/01 - 2017/07/31.

Different feature combinations attempted:
Ref.1: Future features: all except "transactions", Past features: all, Number of past "sales": 63, Zero forecasting: True, Prediction spike fixing: False.
Ref.2: Future features: all except "transactions", Past features: only transactions daily price, Number of past "sales": 63, Zero forecasting: True, prediction spike fixing: False.
Ref.3: Future features: all except "transactions:, Past features: only transactions daily price, Number of past "sales": 63, Zero forecasting: True, Prediction spike fixing: True.

Model	Validation score	Test score
LGBM (Ref.1)	0.33018	0.38616
LGBM (Ref.2)	0.33311	0.38477
LGBM (Ref.3)	N/A	0.38210

Figure 4: Significant LGBM results

We will further utilize the LGBM models and the post-processing techniques as a part of our final submission (see section 3).

2.4 Random Forest

One of the models we explored which turned out to be effective was random forest. Random forest is an ensemble machine learning method that constructs a collection of decision trees. It is widely used for predictive tasks with structured data, so we chose this model for our time series regression project. Since we noticed from data exploration that each product family can have vastly different sales trend, we decided to train 33 random forest models for each family respectively. The input of our models is the features of the date and the family, and the output is the predicted sales for each store on that date for that particular family.

For feature engineering, we included numerical features as well as categorical features.
Numerical features:
1. time-related features: day, week, weekday, month, quarter, trend (from DeterministicProcess), fourier (using CalendarFourier)
2. 7-day moving average of oil price
3. 16-day lag sales of the family
4. onpromotion feature
Categorical features:
1. workday feature: 0 – workday, 1 – weekend, 2 – holiday
2. holidays/events type

For data cleanup, we performed log transformation and 0-1 scaling on all numerical features and output, and used one-hot encoding to transform all categorical features.
In addition to data processing, we finetuned the model hyperparameters using our validation dataset and selected n_estimators=100, max_depth=5 and left all remaining hyperparameters to default. Note that since we did not fix the random state, the test scores we get from each run could have small discrepancies, but never exceeding 0.001.

The table below demonstrates the progression of our random forest model by incrementally adding input features during our development process.

Model	Train period	Validation score	Test score
Random Forest	2017/06/01 - 2017/07/31	0.52328	0.4510
RF (workday feature)	2017/04/01 - 2017/07/31	0.51024	0.43707
RF (16-day lag sales)	2017/04/01 - 2017/07/31	0.47723	0.41719
RF (time-related features)	2017/04/01 - 2017/07/31	0.47091	0.39462

Figure 5: Significant RandomForest results

This random forest model will also be a component of our final submission (see section 3).

3 Our final submission

3.1 Final Result

Score: 0.37925
Class Rank: 5 (As of 12/10/2022)
Global Rank: 20 (As of 12/10/2022).

3.2 Weighted Average (RandomForest + LGBM regressor)

After further testing, we discovered that, RandomForest and LGBM regressor compete in providing the best performance depending on the specific Family of products. Therefore, the thought arose to combine the performance of the two models using weighted average, in hopes that through averaging individual predictions, we could arrive at an accurate final prediction.

We think that one of the reasons for the combined model's good performance is that during featurization, the LGBM model uses the entire training period starting from 2013, which captures the long-term sales trend, whereas the random forest model uses only 4 months of training data starting from April 2017, which focuses on a relatively local trend. Therefore, the combination of both models capture both long-term seasonality as well as local patterns.

With the limited amount of time we had left in the project, however, we were unable to fully implement a voting regressor to find the optimal balance between RandomForest and LGBM regressor. Instead, we experimented by taking the weighted average between the two models and were able to find significant improvements in comparison to the performance of each model individually (RandomForest 0.394, LGBM 0.382) . Figure 6 showcases some of the best test scores we get by choosing different weights for the LGBM and random forest models.

Weight of LGBM	Weight of Random Forest	Test score
0.5	0.5	0.38036
0.7	0.3	0.37925
0.85	0.15	0.37997

Figure 6: Significant Voting Regressor results

3.3 Running our code

Due to the non-deterministic nature of RandomForest, the final score from running our code would differ slightly to our final result. In practice, we submitted multiple test predictions from the model and used the prediction with the highest test score. For more details of running the scripts and installing the dependencies, refer to **README.md** in our submission.

General instructions:
1. Train and generate Random Forest predictions: python random_forest.py

2. Train and generate LGBM predictions: python lgbm_darts_submission.py
3. Generate weighted average predictions: python combine.py
4. The result predictions will be in submission_avg.csv.

4 Conclusion - Thoughts and future improvements

Competing in a global Machine Learning competition was a new journey for all members of our team and we are extremely satisfied with our final result that was wholly our own. We'd like to thank Dr Vatsal Sharan for his aid and guidance throughout CSCI 567.

In both implementing and testing our models, we saw the possibilities of additional exploration with varying levels of potential improvements.

4.1 Additional parametric tuning

Firstly, as mentioned in section 3.3, Random Forest utilize pseudo-randomness as the random splits depends on the seed to a random number function. We could perform more predictions using RandomForest in search for a potentially better performing Test score.

Secondly, we could perform a more fine-tuned weighted average for our voting regressor. Unfortunately however, the Kaggle daily submission limit prevented us from further attempts.

References

[1] Berr, Ferdinand. Darts Forecasting Deep Learning Global Models, Boosted Trees https://www.kaggle.com/code/ferdinandberr/darts-forecasting-deep-learning-global-models/notebook#4.4.-Boosted-Trees.

[2] Herzen, Julien. Time Series Forecasting Using Past and Future External Data with Darts https://unit8.com/resources/time-series-forecasting-using-past-and-future-external-data-with-darts.

A Appendix

A.1 ARIMA

AutoRegressive Integrated Moving Average, or more commonly known as ARIMA, is a model very well known for it's wide utilization in demand forecasting, specialized for time series analysis. Even though we had known about the existence of this model at the start of the competition, we decided to back off from attempting this modle because of our lack of familiarity with its nature as well as its cons of being less than easily explainable, which was important for our team as we worked in a short and demanding time period. However, we have no doubt it would perform very well for some of our store and families.