CSCI 567 Final Report: Store Sales - Time Series Forecasting

Shashank Rangarajan
sr87317@usc.edu

Indrani Panchangam
panchang@usc.edu

Soha Niroumandi
snirouma@usc.edu

Shriya Gumber
sgumber@usc.edu

1 Overview

Time series forecasting is the practice of using historical data to predict future values. It can be challenging, due to the complexity of time-series data and the difficulty of accurately modeling patterns and trends. In this project, we tackled the "Store Sales- Time Series Forecasting"(1) challenge on Kaggle, which involved forecasting sales at a grocery retailer in Ecuador. We trained multiple models, starting with linear regression as our baseline. After careful analysis, we improved upon the baseline and achieved an error score of 0.39525, a 13% improvement over our linear regression model.

2 Glossary

With respect to time-series forecasting:

- trend: refers to a long-term increase or decrease in the data.
- moving average: involves averaging the values of the data over a specific time period and using the average value as the predicted value for the next time point.
- seasonality: presence of regular and predictable fluctuations in data that are associated with particular seasons or periods of time.
- Weka: an open-source machine learning software, including different machine learning algorithms for data mining which is released by the University of Waikato.
- Prophet: an open-source times series forecasting library released by Facebook.
- ARFF: ARFF is the abbreviation of Attribute-Relation File Format, which is an ASCII text file including a list of instances that share some attributes. This data file type was initially introduced by a team from the University of Waikato to be used with the Weka software.

3 Problem Statement

The "Store Sales- Time Series Forecasting" problem involves data from 54 stores that sell products organized into 33 unique product families. The sales data for each product family at each store is provided for a period of about five years, from 2013-01-01 to 2017-08-15. The goal of the challenge is to develop ML/DL models that can accurately predict the unit sales for these stores over the next 15 days (2017-08-16 to 2017-05-31). Section 4 provides more information about the dataset, and section 5 describes the approaches used to create the predictors. The competition uses Root Mean Squared Logarithmic Error (RMSLE) as the evaluation metric to measure the performance of individual models.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

3.1 Evaluation Metric

To evaluate the performance of the model, the Root Mean Squared Logarithmic Error (RMSLE) is used. A smaller RMSLE value indicates that the model is performing better. The RMSLE is calculated using the following formula:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\log(1+\hat{y}_i)-\log(1+y_i)\right)^2}$$

where:

- n is the total number of instances/examples,
- $\hat{y}_i$ is the predicted value of the target for instance (i),
- y_i is the actual value of the target for instance (i), and,
- log is the natural logarithm

4 Dataset

4.1 Dataset Description

The following six data files are provided in this competition (1):

1. Training data - `train.csv`: This file includes information on products, stores, and sales, such as the stores where products were sold and the types of products sold at each store. It also includes the total sales of each product family and whether each product was on promotion.
2. Test data - `test.csv`: This file includes the same information as the training data file, but for the next 15 days after the last date in the training data.
3. Store data - `stores.csv`: This file includes detailed information on stores, such as the city, state, type, and cluster.
4. Oil price data - `oil.csv`: This file includes information on oil prices over time, which is important for this task because the economy and product sales in Ecuador are highly dependent on oil prices.
5. Holiday and event data - `holidays_events.csv`: This file includes information on working and non-working days, which can affect oil prices and product sales. Additionally, it also includes scope, type, description and if the holiday has been transferred to another date by the government is provided.

4.2 Observations

Following are the key observations made after diving deep into the data:

- The number of transactions is highly correlated with the total sales. (Spearman correlation coefficient: 0.8175)
- Store sales are high during December month of every year.
- Stores make more transactions during weekends.
- The sales go down with the increasing oil prices which can be because of inflation.
- The store numbers 20, 21, 22, 29, 36, 42, 52, and 53 have a low correlation with other stores.
- Some stores did not sell some product families at all. Thus, the forecast will be 0 for those particular products.

4.3 Dataset prepration

In this section, we describe the methods and steps used to generate different datasets for our machine learning models.

4.3.1 Dataset-1

Dataset-1 is the baseline dataset for this problem. It includes the sales, oil price, and a 7-day moving average of the oil price for the period from 2017-04-01 to 2017-08-15. The dataset also includes a feature that indicates whether the day is a weekday or weekend. The moving average of the oil price is used to smooth out any short-term fluctuations, and the backfill method is used to handle missing data in the oil price feature.

4.3.2 Dataset-2

This dataset is used to train weka-based models(2). It is in the `arff` format, which is similar to `csv` but allows for the specification of the datatype for each column. To create the dataset, the train.csv file was transformed into 1782 `arff` files, one for each `(store, family)` pair. Each `arff` file contains only the sales data for the given date and store-family pair. Weka is used to interpret the `arff` files and generate time-series features such as:

```
sales, DayOfWeek, Weekend, date-remapped, Lag_sales-1 ... Lag_sales-7,
date-remapped*Lag_sales-1 ... date-remapped*Lag_sales-7
```

4.3.3 Dataset-3

This dataset is a plain data frame of train.csv which is directly fed into the prophet model. Several seasonality and trend features are extracted from this data by Prophet(3).

4.3.4 Dataset-4

After analyzing the trends and seasonality patterns in the given data, we decided to use them as features and created Dataset-4. This dataset includes two main types of features:

- **a.** Oil-related-features: This includes the moving averages for oil price over 6 different sliding windows (daily, weekly, monthly, yearly, quaterly, and semi-annually). Fig.1 shows the weekly and monthly averages. The other parameter is the Lag. We used lag to capture cyclic dependencies. We added lags with different shifts - [1, 2, 3, 4, 5, 6, 7, 10, 14, 21, 30, 60, 90] days
- **b.** Time-related features: This mainly includes features to capture trend and seasonality. We used CalendarFourier with annual, monthly, and weekly frequencies. Apart from this, we used features corresponding to DayOfWeek, isWeekend, MonthOfYear, Year, DaysInMonth, Quater, etc.

Moving averages can represent the trend in a series. Seasonality mostly plays a dominant role by capturing the repeating patterns within a time period. Fig.2 shows the seasonality in the sales and Fig.3 depicts the periodogram for the sales price. Lag further captures cyclic dependencies within features.

4.3.5 Dataset-5

We found that the performance of various models trained on Dataset-4 was limited, so we suspected that adding more features would improve the results. This motivated us to create Dataset-5, which is based on (4). In addition to the two types of features used in Dataset-4, Dataset-5 includes:

- **a.** Holiday-related features: These features were based on information about holidays in Ecuador that was provided to us. The holiday-related features include regional, national, and local holidays, as well as additional holidays like New Year and Easter. By incorporating these features, we hoped to gain insights into the relationship between holidays and product sales.
- **b.** Store-related features: In order to merge the holiday-related fields with the corresponding stores, we needed detailed information about the location of each store. Therefore, we used store-related features to obtain this information. By including these features, we were able to properly link the holiday data with the appropriate stores. This allowed us to analyze the impact of holidays on sales in different parts of the country.

c. Transaction-related features: We included transaction-related features in the dataset to gain insight into average sales across different product families. These features included lag and moving averages, and were based on the observation that transactions are highly correlated with sales in individual stores. By using these features, we hoped to improve the accuracy of our sales predictions.

5 Modeling

In this section, we detail the different forecasting methods we adopted and various models we created for our prediction task.

5.1 Baseline Linear Regression

Linear regression was the baseline model we explored for HW 4. One assumption made in linear regression is that the data points and errors are independent and normally distributed. However, each time-series record is dependent on previous records. To address this, we added a constant field that captures the time-step. This constant ranges from 0 to $length\_of\_dataset$, which inherently captures the dependency between different records. We also added trend and workday features to the model. This model was trained on Dataset-1. We suspected that the linear nature of the model might be limiting its performance, so we investigated non-linear models next.

5.2 SMOreg - SVM Regression with Weka

To quickly experiment with various models, we used auto ML-like frameworks like Weka (2) and Prophet (3) (more on this in the next section). We used Weka to train a nonlinear SMOReg model on Dataset-2. SMOReg stands for Sequential Minimal Optimization for Support Vector regression. Weka implements this as an SVM model for regression, using a PUK kernel (Pearson VII function-based Universal Kernel) to approximate a continuous variable defined in a real range. To train the SMOReg model, we passed individual training files in `arff` format into Weka's command line interface, which we automated using a python script. The SMOReg model automatically added various features (see section 4.3.2). However, when Weka failed to generate predictions for some `(store, family)` pairs, we decided to use the linear regression baseline for these predictions. Finally, we stacked the prediction files to create a csv file that was compatible with the Kaggle submission format. Using this software helped us to identify effective features that could be included in future models.

5.3 Prophet - Automatic Forecasting

To further explore potential models, we tried using Prophet - an open-source time series forecasting framework developed by Facebook. We used Dataset-3 for this. Essentially, Prophet automatically identifies non-linear trends in the data and adds the effects of daily, weekly, and seasonal patterns, as well as holiday effects, to make predictions. To improve the performance of the Prophet model, we also tried normalizing the sales prices with a log transformation and training a new model.

5.4 Random Forest Regressor

Random forests are popular for time series forecasting because they are flexible, easy to use, and can handle various types of time series data. They are also less susceptible to overfitting, which is a common issue with time series data. After comparing the performance of several models, including SVM, XGboost, linear regression, and ridge regression, we observed that the Random Forest Regressor gave the best performance across various datasets and approaches. The following are examples of where Random Forest outperformed other models:

5.4.1 Per Family Predictor - averaging sales

In Dataset-4, we tried to average the sales across all the stores for a given family in order to train one model that predicts sales per family. Among all the different models we tried, we found that the Random Forest model performed the best. However, this approach resulted in identical predictions for a given `(family, date)` pair, so we decided to use the store number as a feature.

Table 1: Summary of results

S.No.	Model	Dataset	RMSLE
1	Linear Regression	Dataset-1	0.45672
2	Weka- SVM	Dataset-2	0.45421
3	Prophet	Dataset-3	0.44932
4	Random Forest	Dataset-4	0.42707
5	XGBoost	Dataset-4	1.02026
6	**Random Forest**	**Dataset-5**	**0.39525**

5.4.2 Per Family Predictor - stores as feature

Seeing identical predictions for given `(family, date)` pairs in the previous approach, we decided to include the store number as a categorical feature in the data. We trained several models on this remodeled Dataset-4, and the Random Forest Regressor was the best-performing model for this task.

5.4.3 Per (Family, Store) Predictor - Adding more features

After training several models on Dataset-4, we found that the features in the dataset were limiting further performance gains. Since the Random Forest Regressor outperformed all other models on Dataset-4, we decided to focus on improving its performance by training it on more features. We created Dataset-5 with features related to stores, holidays, and transactions (see section 4.3.5). We trained one model for each `(family, store)` pair and performed a hyperparameter search to find the best model. We also tried incorporating PCA as a pre-processing step, but this did not improve the model performance. Finally, we added additional custom features from (4), and we saw a significant increase in the performance of our Random Forest Regressor model.

5.5 Other Models

In addition to all the above models, we tried several popular algorithms on all our datasets. However, Random Forest Regressor outperformed all other models. Following are some of the models we explored:

1. Boosting based: We saw that models tend to overfit in time-series, and hence decided to use boosting and bagging techniques to prevent overfitting. As part of boosting algorithms we tried:
   - **a.** XG Boost Regressor - XGBoost is the model of choice for Kaggle competitions.
   - **b.** Gradient Boost Regressor
   - **c.** Cat Boost Regressor
   - **d.** Ada Boost Regressor
2. Multi-layer perceptron Regressor
3. Ridge Regressor

6 Results and Conclusion

Table 1 summarizes the performance of various models trained on different datasets. The Random Forest Regressor trained on Dataset-5 had the best performance, with an MSLE of 0.39525.

Our exploration and analysis of various approaches and datasets indicate that feature engineering is crucial for improving the performance of time series forecasting models. With the same input features (features in Dataset-1), we were unable to improve our results by using different algorithms. However, adding new relevant features such as trends and seasonality to Dataset-4 improved the results. We were able to further improve the results by adding additional features to Dataset-5, such as holidays and events.

Based on our results, effective feature engineering to capture relevant features is crucial for this problem.

7 Future work

One limitation of our work is that we did not include external data sources that could potentially provide additional information about the sales data. For example, we did not consider information about the local economy, consumer behavior, or market trends that could be relevant for forecasting sales. In future work, we can explore incorporating external data sources to see if this improves the performance of our models. We can also continue to explore preprocessing techniques and advanced DL algorithms e.g. LSTMs, and Transformers for time series data to further improve the accuracy of our predictions.

8 Instructions to run the code

To run the submitted code, run the best-model-final.ipynb on kaggle/ colab ensuring the dependencies are installed. The code may take 2 hours to run

References

[1] `https://www.kaggle.com/competitions/store-sales-time-series-forecasting/overview`

[2] Holmes G, Donkin A, Witten IH. Weka: A machine learning workbench. InProceedings of ANZIIS'94-Australian New Zealnd Intelligent Information Systems Conference 1994 Nov 29 (pp. 357-361). IEEE.

[3] Taylor SJ, Letham B. Forecasting at scale. The American Statistician. 2018 Jan 2;72(1):37-45

[4] @ARTEMCHISTYAKOV's - Store Sales - EDA + RF, `https://www.kaggle.com/code/artemchistyakov/store-sales-eda-rf`

Fig 1. Moving average plots of oil price

Fig 2. Seasonal plots of sales price

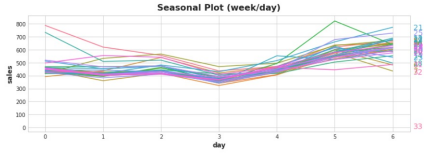

Fig 3. Periodogram for sales price