WALMART MULTI-STEP SALES FORECASTING

by

Moeen Bagheri

A Major Research Paper
presented to Ryerson University
in partial fulfillment of
the requirements for the degree of
Master of Science (MSc)
in the Program of
Data Science and Analytics

Toronto, Ontario, Canada, 2020

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

Moeen Bagheri

WALMART MULTI-STEP SALES FORECASTING

Moeen Bagheri

Master of Science (MSc)

Data Science and Analytics

Ryerson University, 2020

Abstract

Acknowledgments

acknowledgements

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Sales forecasting is crucial for retailers in order to predict the future demand of products, which can in turn increase profit by ensuring the replenishment of the necessary supply to meet future demand, as well as minimizing product waste [1]. However, the volatility in demand makes sales forecasting a challenging problem [2]. This volatility is dependent on many external factors, such as holidays, events, price, and promotions. Hence, it is necessary to take into account the effects of these external factors when forecasting future sales. In some approaches, the manager guides the forecasting model based on his/her knowledge of the external factors by using fuzzy logic. However, recent studies have focused on creating models that are able to directly take into account the effects of these external factors.

In this paper, we will compare the performance of three models, which consist of Long Short-Term Memory (LSTM), Multi-Layer Perceptron (MLP), and LightGBM (LGBM) models. Furthermore, the LSTM model and the best performing model between the MLP and LGBM models will be used to construct a hybrid model to check if it is able to improve the performance compared to the two singular models.

Chapter 2

Literature Review

Numerous statistical and deep learning methods have been employed in the past for forecasting sales. Linear statistical models, such as multivariate linear regression, which make predictions based on the historical relationship between different influential factors and the demand, have the advantage of being efficient. However, these linear models perform well for linear problems, where the relationship between the dependent variable and one or more independent variables is linear with a constant rate of change, and hence fail to capture the nonlinear relationships and describe the complexity of the supply chain [3]. Similarly, basic univariate models, such as ARIMA, which consider the data as a time-series, are also unable to describe the complexity of the supply chain since they are only capable of capturing linear relationships in the data [3, 4]. On the other hand, nonlinear statistical models used for sales forecasting include Bayesian networks, support vector machines, and Markov chains. Unlike linear models, these models are able to learn complex nonlinear relationships from the data. In a study done in [5], a Random Forest model was used to forecast Major League Baseball game ticket sales. Their approach consisted of a dynamic month-ahead forecasting strategy, where the data is updated every month. Their results showed that their proposed RF model slightly outperforms their baseline model, which they chose to be an Ordinary Least Squares (OLS) regression model. However, even though nonlinear statistical models have a high capability of solving complex problems, selecting the right model for a certain problem is a difficult task, which requires expert knowledge of statistical models. Moreover, these methods are usually found to perform worse compared to deep learning methods [3].

Deep learning methods are able to automatically extract important features and have been found to obtain better results compared to statistical models [3]. The self-organizing and self-adjusting capabilities of Artificial Neural Networks (ANN) allows them to solve complex nonlinear problems [6]. In a study done by [7], a Multi-Layer Perceptron (MLP) was used to predict the monthly sale volumes of a Polish company, which imports fabric on a monthly basis, based on the previous three months. The MLP contained three input neurons, a single hidden layer with 15 neurons, as well as one output neuron, and was able to achieve a high accuracy on the data with a Root-Mean-Square Error (RMSE) of 3.34e-11. However, even though ANNs excel at solving complex problems, they lack the ability to interpolate and predict long-term sequences [6]. Alternatively, deep learning methods, such as LSTM, are able to preserve past information and capture the temporal relationships in the data [8]. However, even though deep learning methods can improve the accuracy of the predictions compared to statistical models, it is much more challenging to interpolate and draw conclusions from their results [3, 7].

In a study done in [9], the performances of ARIMA, MLP, and LSTM models were compared for forecasting and predicting cash flow. Interest Opportunity Cost (IOC), which is a measure based on financial concepts and allows finance-specific comparison of the models, was used in MLP and LSTM as the error function to be optimized. According to their results, LSTM was able to obtain the minimum error of 0.09, compared to MLP and ARIMA with an error of 0.10 and 0.23, respectively. The cash flow data exhibited a strong weekly pattern that assisted LSTM in its predictions. However, due to the small amount of data available (3 years), variances caused by holidays and other special events could not be explained.

On a separate note, many methods do not take into account external variables and factors, such as price changes and promotions and have been shown to only perform well in periods without the influence of any external factors [10, 11]. In practice, in order to incorporate the effects of promotions on the sales, many retailers use a base-times-lift

approach, where the sales are first forecasted based on a simple time-series and then adjusted based on the incoming promotions [12]. Recent studies have focused on optimizing these adjustments, which are made based on promotions and other external factors [12]. In an alternative approach, hybrid models have been used in order to take advantage of the strengths of different models together, which helps capture both the temporal information in the data, as well as the correlation between the demand and the external factors [3, 13]. The complex behaviour of a time-series cannot be explained by a single model if, for example, the time-series contains both linear and nonlinear correlations [14]. Hybrid models are usually constructed in a sequential manner, where the first component is fitted to the data first, and then the second component is fitted to the residuals of the first component [14]. The residuals of a model contain the information that could not be captured by that model [13]. However, hybrid models are not guaranteed to perform better than single models and model selection is still a crucial aspect of hybrid models [14].

The study done by [13] presents an example of a hybrid model used for forecasting. In this study, an LSTM model is combined with a Random Forest model to create a hybrid model for forecasting sales of a store with one online and 11 offline sale channels. In the hybrid model, LSTM is applied first to capture the linear and non-linear temporal information from the data. Next, the residuals from the LSTM are used as the dependant variable and the external variables are used as the independent variable in a Random Forest model in order to capture the non-temporal relationships in the data. Another challenge to address is the modeling of sales across multiple channels. One approach would be to model each channel separately, however, this approach will eliminate the aggregate demand information from the data. Therefore, this study forecasted the demand of a product based on its order origin (online vs. offline) instead in order to sustain the aggregate demand information. Their results showed that the hybrid model performed better than its two components, LSTM and RF, individually.

Chapter 3

Exploratory Data Analysis

The dataset [15] includes the unit sales of 3,049 products sold by Walmart in the USA in grouped time-series format. The products are classified in three categories (Hobbies, Foods, and Household) and seven departments. Additionally, these products are sold across ten stores, located in the three states of California (CA), Texas (TX), and Wisconsin (WI). Specifically, there are four stores located in California, three in Texas, and three in Wisconsin. The dataset contains the following three data files [16]:

- **calendar.csv**: contains information about the dates the products were sold, such as promotions, holidays, and other events.
- **sell_prices.csv**: contains the average weekly price of each item.
- **sales_train.csv**: contains the daily unit sales of each item.

Figure 3.1 shows the number of products per category. There are 1437, 1047, and 565 products in the Foods, Household, and Hobbies categories, respectively, for a total of 3049 products. Since these products are sold across ten different stores, there is a total of 30,490 items, out of which 22,243 had a price change at some point in time. In addition, from Figure B.1, which shows the price distribution of all

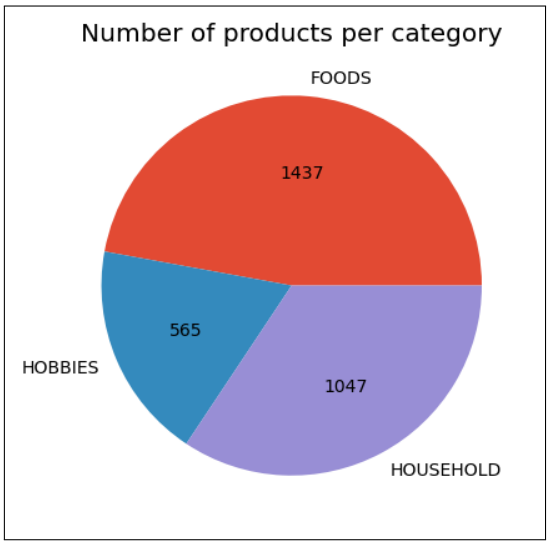


Figure 3.1: Number of products per category.

products, we can see that most items cost between three to four dollars. Furthermore, Figure B.2 shows the price distribution of each product category separately. It is apparent that the Hobbies products do not have the same price distribution as Foods and Household products and tend to have more products that are very cheap.

On another point, Figure B.3 shows a 4-week moving-average of the revenue earned, number of units sold, and the average price of all items. We see that the average price of products are increasing over time in a linear fashion. Moreover, as expected, revenue and unit sales have increased with almost an identical pattern. Additionally, examining the revenue and unit sales graphs, we can see some form of seasonality in the curves. Hence, we further explore for any seasonality patterns. From Figure 3.2, which shows the number of unit sales per day for the year 2011, we see a clear weekly pattern in the number of unit sales. Specifically, the number of units sold tends be the highest during the weekends and the lowest in the middle of the week. This can be confirmed by examining Figure 3.4, which shows the average number of units sold per day of week for all years. Additionally, Figure 3.2 shows national holidays and other special events, which are considered as external factors and may affect the number of unit sales. Most national holiday events,

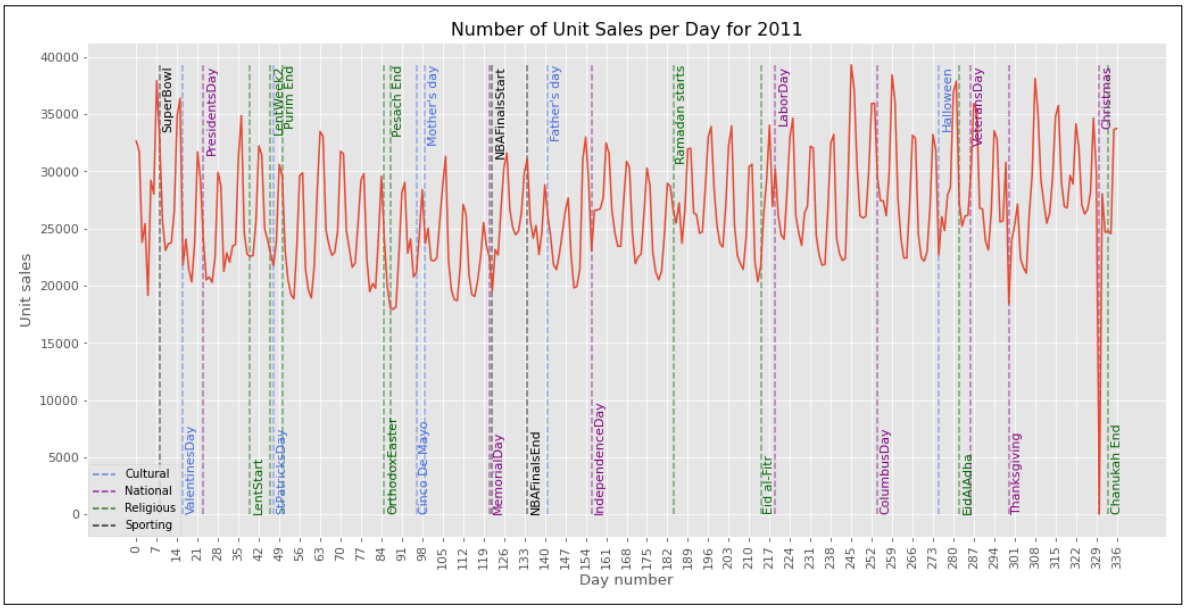


Figure 3.2: The number of unit sales per day in 2011.

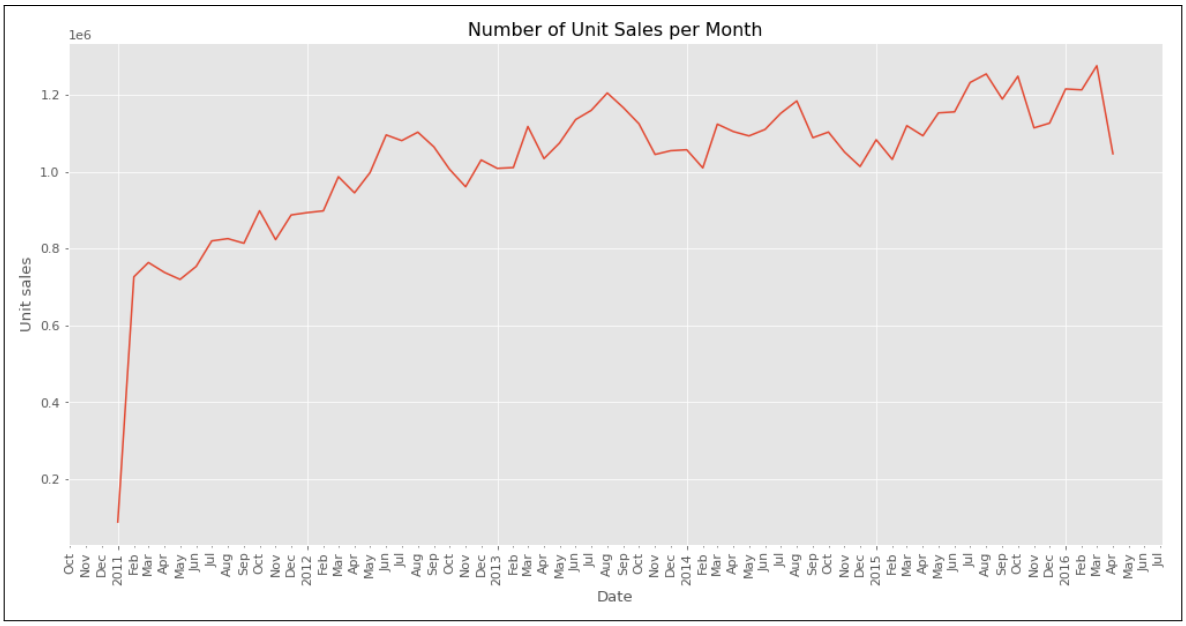


Figure 3.3: The number of unit sales per month.

such as Thanksgiving, have a negative effect on the number of unit sales, possibly because people tend to spend time with their families rather than shopping. Additionally, Figure 3.3 shows the number of unit sales per month, which shows that the number of sales tend to increase as we approach the middle of the year and falls off at the end of the year. Moreover, we can see a spike in sales during March. This is confirmed by examining Figure 3.5, which shows the average number of units sold in each month of the year.

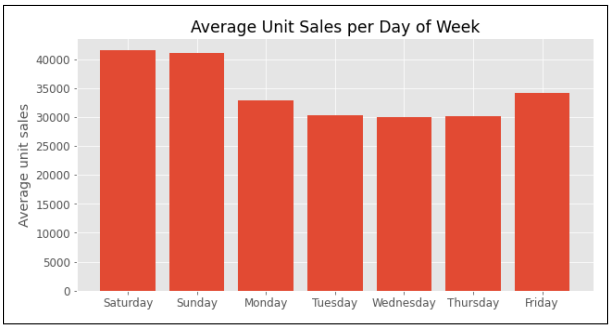


Figure 3.4: Average number of units sold per day of week.

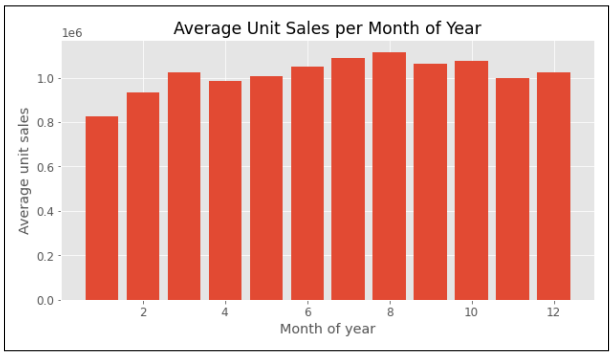


Figure 3.5: Average number of units sold per month.

In the rest of this chapter, we will group the items based on their categories, states, stores, and departments and compare their sales data.

3.1 Product Categories

Furthermore, we examine the number of units sold and revenue earned from each product category. By examining the pie charts in Figure 3.7, we see that the Foods category accounts for most of the units sold and revenue earned, with 68.6% of all products sold belonging to the Foods category, which corresponds to 58.0% of the total revenue earned. Moreover, figure 3.6 shows a time-series of the number of unit sales per category. It is apparent that the Foods category has a lot more sales that the other two categories. Moreover, the sales for the Foods category seem to have a lot of oscillations that is less present in the other two categories.

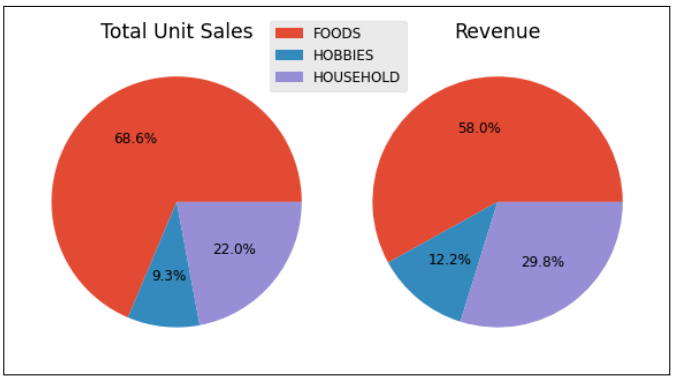


Figure 3.7: Proportion of units sold and revenue earned in each product category.

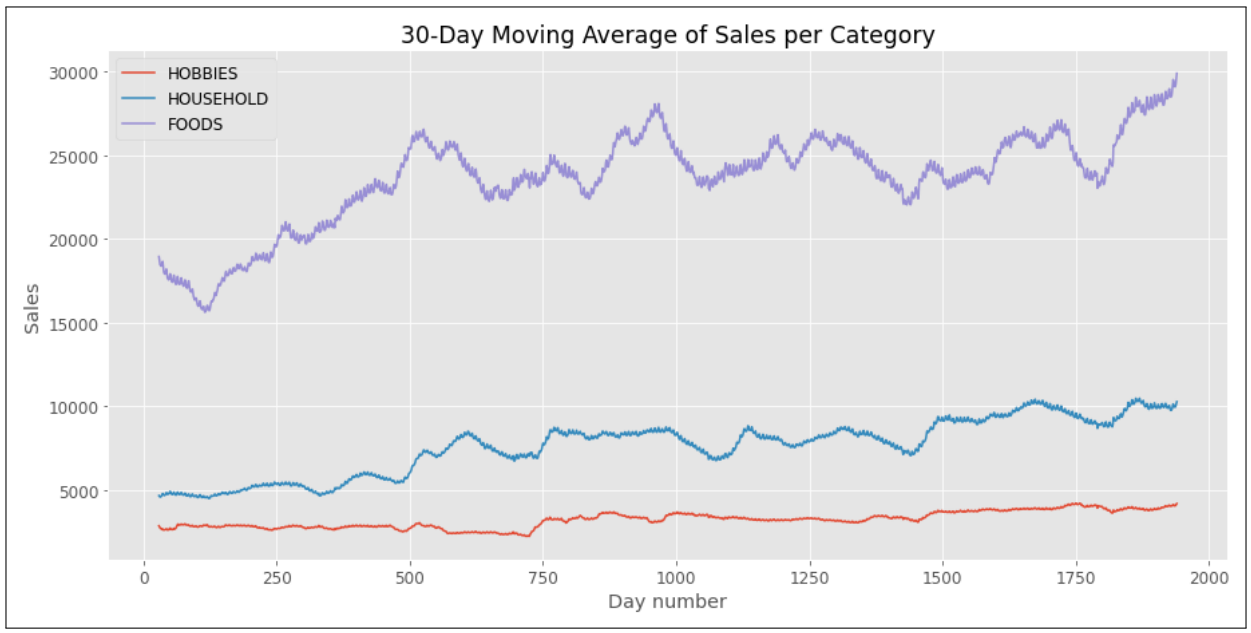


Figure 3.6: 30-day moving average of sales per category.

3.2 States

Another factor to examine is whether there is any difference in the number of units sold and revenue earned between various states. Figure 3.9 compares the total unit sales and revenue earned in each state. Since the number of stores is not constant between the states, the values for each state have been normalized by the number of stores in that state. By examining the pie charts we see that, on average, the three states have about the same number of unit sales and revenue earned per store, however, the store in state of California are, on average, performing slightly better compared to the stores in Texas and Wisconsin. Furthermore, Figure 3.8 shows that the state California has more oscillations in its sales compared to the other states.

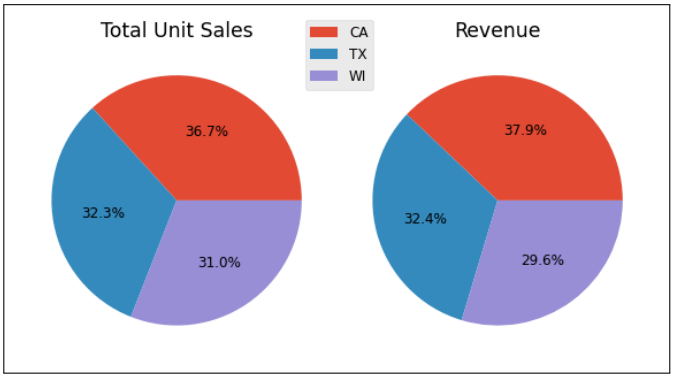


Figure 3.9: Proportion of units sold and revenue earned in each state.

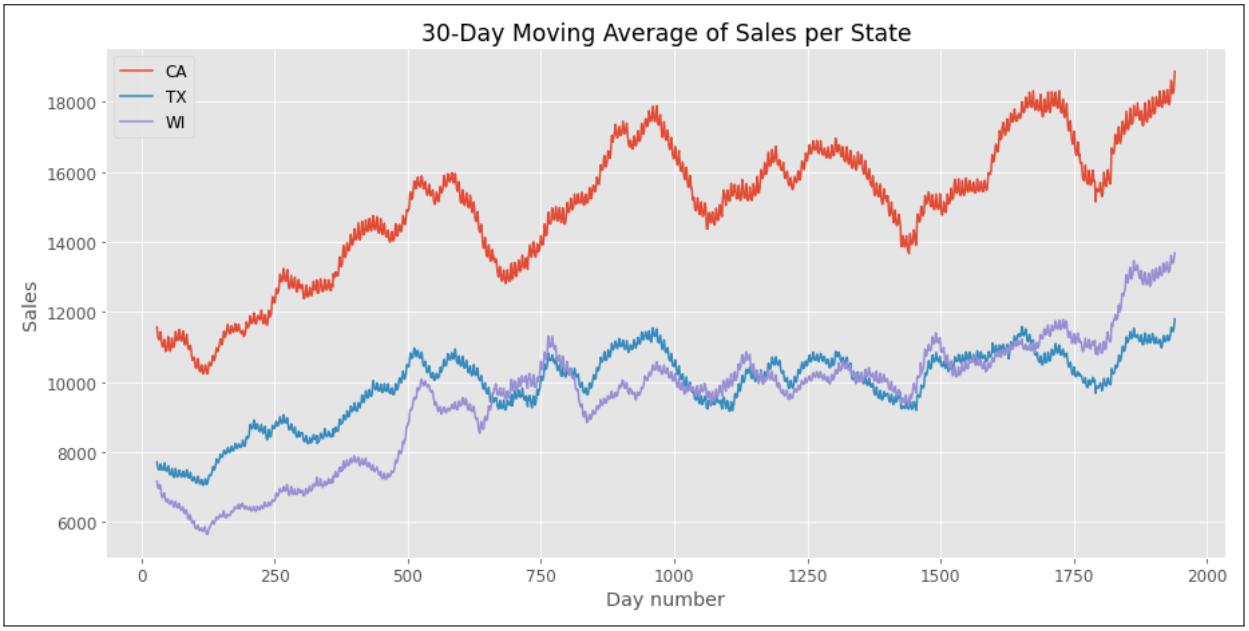


Figure 3.8: 30-day moving average of sales per category.

3.3 Stores

There are a total of nine stores in the dataset. The bar chart in figure 3.11 compares the number of unit sales and revenue earned between each stores. From this figure, we can see that store CA_3 has the highest number of unit sales and revenue earned, and store CA_4 has the lowest number of unit sales and revenue earned. We can further confirm this by examining the time-series of the number of sales per store, shown in Figure 3.10. Moreover, this figure shows that store CA_3 has a lot more volatility in its sales than the other stores.

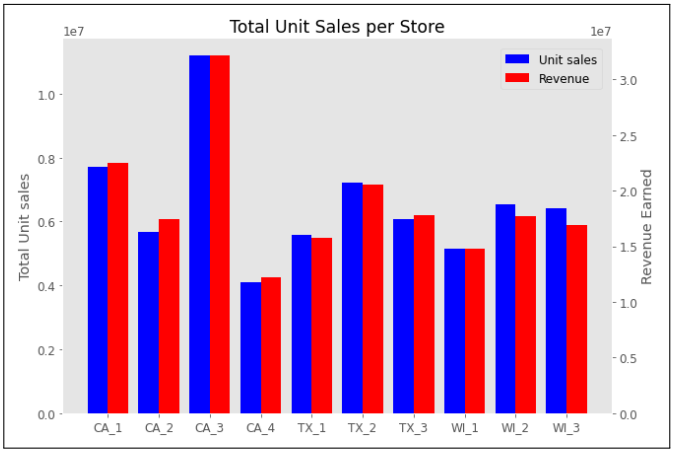


Figure 3.11: Proportion of units sold and revenue earned per store.

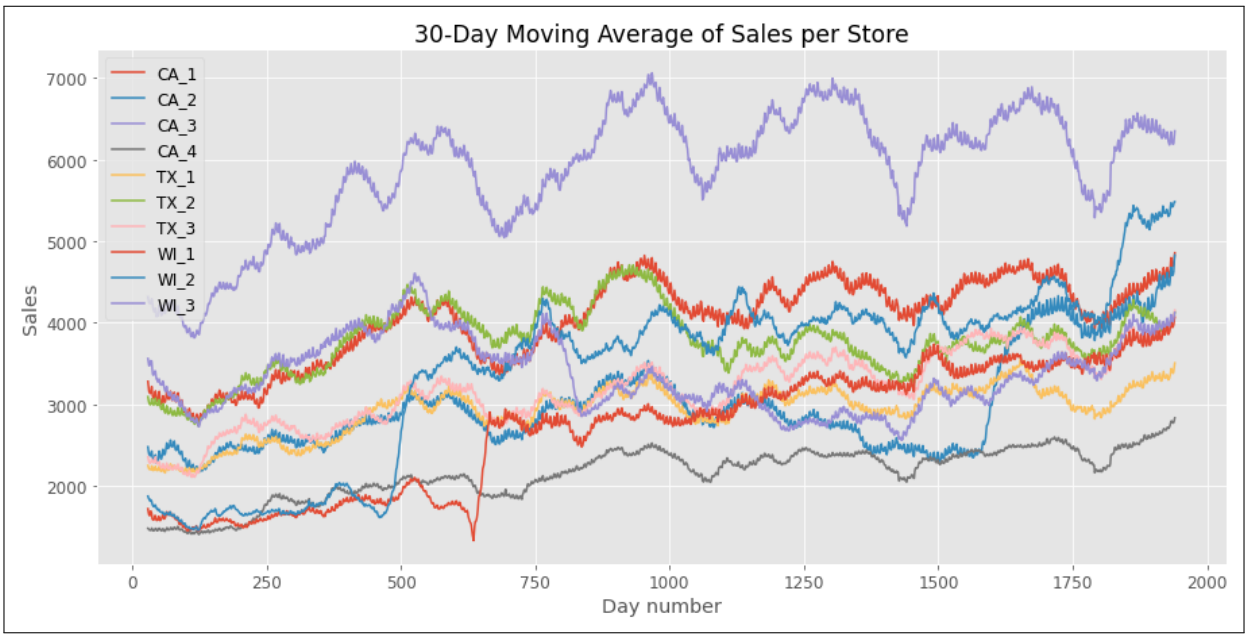


Figure 3.10: 30-day moving average of sales per store.

3.4 Departments

There are a total of seven departments in the dataset. There are three departments for the FOODS category, two departments for the HOBBIES category, and two departments for the HOUSEHOLD category. The number of unit sales and revenue earned in each department is compared in Figure **??** The bar chart shows that .... . Moreover, Figure 3.13 shows a 30-day moving average of the number of sales per department, which shows that the *FOODS_3* department has a higher number of unit sales, as well as a higher volatility compared to the other departments.

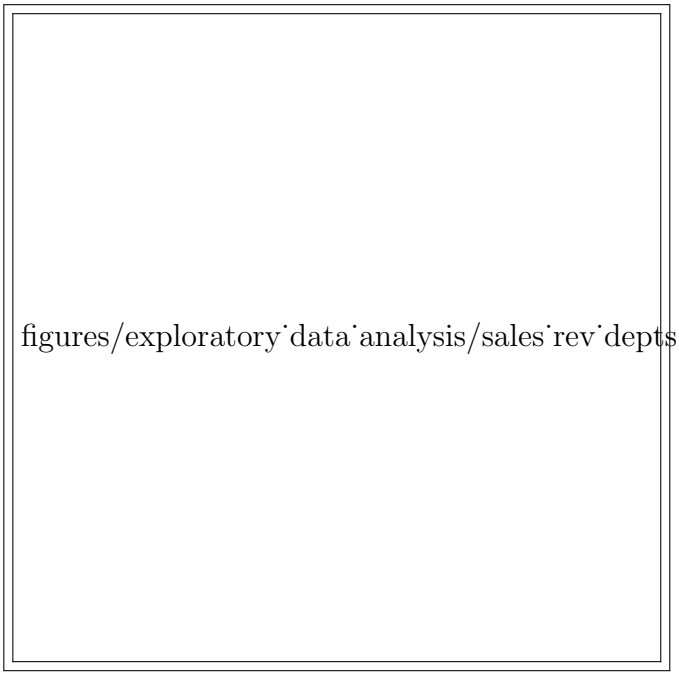


Figure 3.12: Proportion of units sold and revenue earned per store.

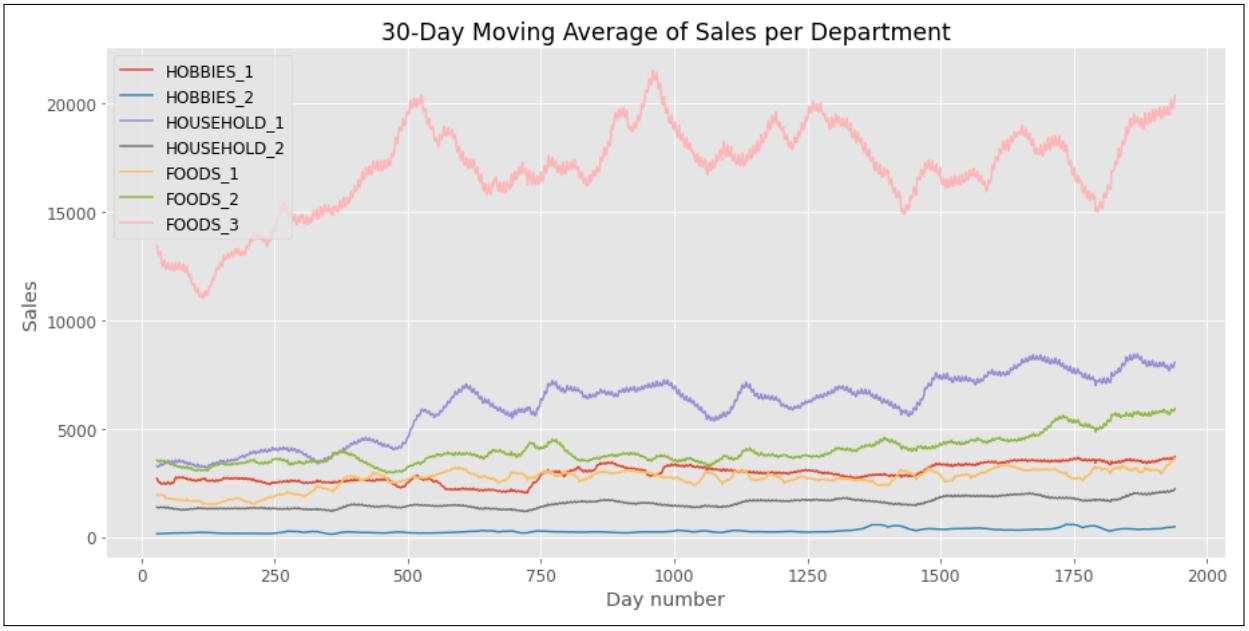


Figure 3.13: 30-day moving average of sales per store.

Chapter 4

Methodology

4.1 Aim of Study

The data contains a time-series of the number of sales for 3049 products across 10 different stores, for a total of 30490 items. We aim to create several models and compare their performance on predicting the number of sales for all 30490 items for the next 28 days. Specifically, we aim to compare the performance of deep learning and machine learning methods. For deep learning methods, we will compare the performance of Recurrent Neural Networks (RNN) with Artificial Neural Networks (ANN). Specifically, the recurrent neural network will be a Long Short-Term Memory (LSTM) model, and the artificial neural network will be a Multi-Layer Perceptron (MLP) model. Moreover, the machine learning method was chosen to be a LightGBM (LGBM) model. Additionally, we will create a hybrid model, consisting of the LSTM model and the best performing model between the MLP and LGBM models, and check if it is able to improve the performance compared to the sigular models.

4.2 Selection of the Response Variable

All models will be trained using the Root Mean Square Error (RMSE) as the loss function. Moreover, the RMSE on the validation set will be used to optimize the hyperparameters of each model. On the other hand, in order to evaluate and compare the performances of the models, we will obtain the Root Mean Square Scaled Error (RMSSE) of the models on

the test set. Compared to RMSE, RMSSE has the advantage of being scale independent, and so it can be used to compare the accuracy of forecasts across time-series with different scales. For example, an absolute error of 10 when approximating 100 should have a much higher forecast error than an absolute error of 10 when approximating 1,000,000. RMSSE is derived from the Mean Absolute Scaled Error (MASE), which is a common measure of accuracy for forecasting problems. In order to scale the error, the Mean Squared Error (MSE) of the model is compared to the MSE of a naive model, which predicts the sales at each time-step to be the same as the previous time-step. The original RMSSE uses the in-sample data (training set) for the naive model, since the out of sample data (test set) may not contain enough points to obtain a naive prediction when the forecasting horizon is too short: [16, 17]

$$RMSSE = \sqrt{\frac{MSE_{test,\,model}}{MSE_{train,\,naive}}}$$

However, since in our study we have a forecasting horizon of 28, we do not have to worry about this issue and hence for simplicity, we will use the out of sample data to obtain the MSE of the naive model:

$$RMSSE = \sqrt{\frac{MSE_{test,\,model}}{MSE_{test,\,naive}}}$$

Another advantage of RMSSE compared to RMSE is its interpretability. The value of RMSSE represents how well a model performs compared to a naive model. An $RMSSE > 1$ means that the model performs worse than the naive model and should be discarded, an $RMSSE = 1$ means the model performs just as well as the naive model, and an $RMSSE < 1$ means the model performs better than the naive model. The closer the value of RMSSE is to 0, the better the model performs.

In addition to RMSSE, the Mean Error (ME) of the models will be obtained in order to see whether the models have a tendency to over-forecast or under-forecast the target sale values.

4.3 Choice of Factors and Levels

The following four models will be evaluated and compared in this project.

1. LSTM,
2. Artificial Neural Network,
3. LightGBM, and
4. Hybrid.

The hyperparameters of all models will be optimized using Bayesian Optimization over a given range of hyperparameter values. In Bayesian optimization, the optimizer starts with a prior belief of the environment and with every combination of hyperparameter values that it examines, it updates its belief of the environment and looks for the hyperparameter values that will maximize the chance of improving our performance in the next round. For the LSTM and MLP models, Bayesian optimization was used to optimize the number of hidden layers, number of hidden units, learning rate, learning rate decay, dropout rate, and batch size. Moreover, Bayesian optimization was used to decide whether a batch normalization layer should be included in the architecture of the models. Additionally, the number of historical observations for the LSTM model was optimized as well. On the other hand, the LGBM model has many hyperparameters that can be optimized. In this project, Bayesian optimization was used to find the best hyperparameter values for the learning rate, the fraction of the features used, the value of lambda for L2 regularization, the minimum data in each leaf, and the number of leaves for the model.

4.4 Choice of Experimental Design

The time-series include a total of 1941 days of sales data, which will be split into training, validation, and test sets. The training set will include 1885 days of time-series data, which

will be used to train the hybrid model. The validation set will include a time-series data for the next 28 days, from day 1886 to day 1913, which will be used for hyperparameter optimization. Finally, the test set will include the last 28 days of time-series data, from day 1914 to day 1941, which will be used to evaluate and compare the models and assess their generalization on unseen data.

Since the dataset contains the sales data for 3049 products sold across ten different stores, we are dealing with a multi-channel problem. It is important to take into account the effects of aggregate demand on the sales, as well as the effects of one items on the sales of another item. Hence, rather than training separate models for each item, a single model will be created for all items. Furthermore, since we want to forecast the daily sales for the next 28 days, we are dealing with a multi-step forecasting problem. There are three main methods for dealing with multi-step forecasting problems. In the first method, a separate model is trained for each step. However, this is not very practical, not only because of the number of models that will need to be trained, but also because normally the sales on each day are somehow dependent on the sales of the previous days, which is not considered in this method. In the second method, a single model predicts all steps at once. However, this method also lacks the ability to obtain any information from the previous steps of the forecasting horizon to predict future steps. Alternatively, one can use a single model to forecast each step one at a time, where at each step, the previous forecasted values are used to forecast the sales of the current day. This method can be more consistent as it takes into account the effects of previous days when forecasting many days ahead. Hence, this is the method that will be used in this project to deal with multi-step forecasting.

In the rest of this section, we will talk about the two experiments that will be performed in this project.

Experiment 1

Three singular models (LSTM, MLP, and LGBM) will be constructed and their performances will be evaluated and compared for a 28-days point forecast. In order to help the MLP and LGBM models learn the seasonality from the time-series data and look more than one step in the past, we will introduce lag features in the dataset. To produce lag features, we will shift the sales data to the future by a fixed number of days. For example, the lag 7 of the sales at time-step t for an item will be its sales value at time-step $t - 7$. In addition to the lag features, we will introduce rolling means and rolling standard deviations of the sales with various sliding-window sizes in order to help the models learn from the time-series data.

Experiment 2

For the second experiment, a hybrid model will be constructed and its performance will be compared with the performances of its two components. There are two elements in the dataset that must be considered when constructing the model's architecture. The first element is the time-series data and the second element is all other data, such as promotions and holidays, that are considered to be external factors and might have an influence on the time-series data. Hence, the first component of the hybrid model will be the LSTM network that will learn from the time-series data. LSTM networks are able to handle both the linear and nonlinear demand variations, which eliminates the need of multiple methods for different demand variations [13]. The second component of the hybrid model will serve to learn from the external factors. Specifically, the second component will predict the difference in LSTM's predictions and actual demand based on the external factors. To elaborate, assume we want to predict the demand for a time period, $t \geq 1$, with the actual demand data, Y_t, known. If the LSTM model makes predictions, $\hat{y}_t$, for this time period,

then the difference in the LSTM's predictions and the actual demand is calculated as:

$$\Delta y_t = Y_t - \hat{y}_t$$

This difference, Δy_t will be used as the independent variable(s) in the second component of the hybrid model, along with the external factors as the dependant variables. The second component will then be trained to predict this difference, $\Delta \hat{y}_t$, based on the external factors. At last, the final forecast, $\hat{Y}_t$, will be obtained by aggregating the predictions of the two model components [13]:

$$\hat{Y}_t = \hat{y}_t + \Delta \hat{y}_t$$

The study done in [13] has already used a Random Forest (RF) network as the second component for forecasting demands of a multi-channel retail, and they were able to achieve very accurate predictions. In this study, we will use the best performing model between MLP and LGBM as the second component of the hybrid model. Similar to random forest, LGBM is a tree-based learning model, however, LGBM uses gradient boosting to enhance the performance of the model, whereas random forest uses bagging.

4.5 Technologies

All models will be created using Python. The LSTM and ANN models will be constructed using the package keras and LGBM will be implemented using the package lightgbm. The Bayesian optimization will be created using the bayesian-optimization package. Moreover, other packages, such as pandas and scikit-learn, will be used for data preprocessing and model evaluation.

Chapter 5

Results

The following four models were constructed:

1. Long Short-Term Memory (LSTM)
2. Multi-Layer Perceptron (MLP)
3. LightGBM (LGBM)
4. LSTM-LGBM Hybrid

Bayesian optimization was used to optimize the hyperparameters of the models within a given range of values. The optimization process consisted of a few rounds of exploration, where new points are randomly explored within the available space, and many rounds of exploitation, where Bayesian optimization is done to find the optimal hyperparameters. The performances of the optimized models were evaluated on the training, validation and test sets for a 1-day ahead point forecast and were compared to the performance of a naive model. Moreover, the performance of each optimized model was evaluated on the test set for a 28-days ahead point forecast and compared to the naive model. The ME, RMSE, and RMSSE of each model after hyperparameter optimization on the training, validation, and test sets are shown in tables A.1, A.2, A.3, A.4. Note that for the MLP and LGBM models, the 1-day ahead forecasts are the same as the 28-days ahead forecasts, but for the LSTM and hybrid models, these forecasts are different. This is due to the difference in the feature engineering done for the LSTM and hybrid models compared to the MLP and LGBM models, which will be discussed further in the discussion. From the results, it can

be seen that the LGBM model outperforms the other models with an RMSSE of 0.77283 on the test set for 28-days ahead forecasts.

In the rest of this chapter, the results of hyperparameter optimization and the performance of each model on the training, validation, and test sets will be discussed in detail.

5.1 Long Short-Term Memory

Figure 5.1 shows a summary the LSTM model's structure. The LSTM model was able to achieve an RMSE of 2.10657 on the training set, 2.11495 on the validation set, and 2.24916 on the test set, when predicting one day ahead. Moreover, the RMSE of the model on the test set for 28-days ahead forecasts was 2.27780. Figure B.7a shows the 1-day ahead predictions of the LSTM model versus the actual sale values for the last 28 days of the training set, as well as the entire validation and test sets, which shows that the model was able to learn the weekly pattern presented in the time-series data. The 28-days ahead predictions of the model on the test set are also shown in Figure B.7b. From these predictions, we can see that the model struggles to forecast the last days of the horizon and starts to lose its weekly pattern. This is most likely due to error propagation from the earlier predictions, which will be discussed further in the discussion.

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 23, 229)	28147764
dropout_1 (Dropout)	(None, 23, 229)	0
lstm_2 (LSTM)	(None, 23, 304)	649344
dropout_2 (Dropout)	(None, 23, 304)	0
lstm_3 (LSTM)	(None, 405)	1150200
dropout_3 (Dropout)	(None, 405)	0
dense_1 (Dense)	(None, 30490)	12378940

Total params: 42,326,248
Trainable params: 42,326,248
Non-trainable params: 0

Figure 5.1: A summary of the LSTM's structure.

Figure B.4 shows the learning curve of the model on the training and validation sets. It can be seen that the RMSE of the model plateaus around epoch 20, and the best validation RMSE occurs at epoch 23. Moreover, the RMSSE of the 28-days ahead forecasts on the test set was 0.80511, which means that the model was able to perform around 20.5% better than the naive model, and the ME of 28-days ahead forecasts on the test set was -0.10991,

which means that the model has a tendency to under-forecast the target sale values.

Using Bayesian optimization, the number of historical observations was optimized to be 23 days, and a batch size of 59 was chosen for the training process. Moreover, the optimized LSTM structure contained three hidden LSTM layers with 229, 304, 405 nodes, respectively. The dropout rate was optimized to be 0.225848, however, no batch normalization layers were included. Finally, the learning rate was optimized to be 0.000534 with a decay rate of 0.000445. Table A.5 shows the range of values that each hyperparameter was optimized over, as well as the optimized values.

5.2 Multilayer Perceptron

A summary of the MLP model's structure is shown in Figure 5.2. The RMSE of the MLP model on the training, validation, and test sets were 2.73888, 2.24467, and 2.29376, respectively. Figure B.5 shows the RMSE of the MLP model at each epoch on the training and validation sets. This learning curve shows that most of the learning was done on the first iteration, and that the model had a difficult time learning useful information that will generalize well on the validation set after the first iteration. The RMSE of the model exhibits a lot of oscillations, with the minimum validation error occurring at epoch 52. Moreover, the MLP model was able to achieve an RMSSE of 0.81075 for the 28-days ahead forecasts and was able to perform around 18.9% better than the naive model. Furthermore, the ME of the model on the test set was -0.08568, which means that the model has a tendency to under-forecast the target sale values.

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 159)	4134
dropout_1 (Dropout)	(None, 159)	0
dense_2 (Dense)	(None, 228)	36480
dropout_2 (Dropout)	(None, 228)	0
dense_3 (Dense)	(None, 326)	74654
dropout_3 (Dropout)	(None, 326)	0
dense_4 (Dense)	(None, 467)	152709
dropout_4 (Dropout)	(None, 467)	0
dense_5 (Dense)	(None, 1)	468

Total params: 268,445
Trainable params: 268,445
Non-trainable params: 0

Figure 5.2: A summary of the MLP's structure.

The range of values that each hyperparameter of the model was optimized over, as well as their optimized values are shown in Table A.6. The optimized MLP model contained four hidden fully-connected layers with 159, 228, 326, and 467 nodes, respectively. Moreover, the optimized model did not include any batch normalization layers but had a dropout layer with a rate of 0.006266 after each fully-connected layer. Additionally, for the training phase, the learning rate was optimized to be 0.004514 with a decay rate of 0.000478 and the training batch size was optimized to be 57798.

5.3 LightGBM

The LGBM model was able to achieve an RMSE of 2.34224, 2.11933, and 2.18648 on the training, validation, and test sets, respectively. The LGBM model was able to achieve the best performance on the test set compared to the other models. From the learning curve of the model, shown in Figure B.6, it can be seen that the model's error plateaus around epoch 30, however, the model still continues to learn at a very low rate until epoch 122, where it reaches its best validation error, and then starts to overfit on the training set. In addition, the ME and RMSSE of the 28-days ahead forecasts on the test set were -0.07056 and 0.77283. respectively. The RMSSE of the model shows that it was able to perform around 22.7% better than the naive model. Furthermore, the ME of the model shows that the model has a tendency to under-forecast the target sale values.

Table A.7 shows the range of values that each hyperparameter was optimized over and their optimized values. The final LGBM model had a learning rate of 0.097892, with 0.553332 fraction of features used to train each tree, and a lambda value of 0.219554 for L2 regularization. Moreover, the number of leaves for the model was chosen to be 749 with a minimum data of 1534 in each leaf.

5.4 LSTM-LGBM Hybrid

An LSTM model was chosen as the first component of the hybrid model in order to learn from the sales time-series data. Moreover, the best performing singular model between MLP and LGBM was used as the second component of the hybrid model. Additionally, the optimal hyperparameters found previously for each singular model were used as the hyperparameters for each component. The proposed hybrid model was able to obtain an RMSE of 2.10155 on the training set, 2.11706 on the validation set, and 2.26268 on the test set for the 1-day ahead point forecasts. These performances correspond to an RMSSE of 0.78133, 0.82453, and 0.86192 on the training, validation, and test sets, respectively. Moreover, the RMSE and RMSSE of the hybrid model for the 28-days ahead forecasts on the test set were 2.25698 and 0.79775, respectively, which shows that the model was able to perform around 20.3% better than the naive model. Even though the hybrid model performed slightly worse than the singular LSTM model on the 1-day ahead forecasts, the hybrid model was able to obtain a slightly better performance on the 28-days ahead predictions, which shows that the second component, LGBM, was able to help counteract the effects of error propagation, which can be confirmed by comparing Figures B.7b and B.10b. On the other hand, the hybrid model was not able to beat the performance of the singular LGBM model. Moreover, the ME of the model for the 28-days ahead forecasts on the test set was negative with a value of -0.11916, which shows that the model is slightly under-forecasting the targets sale values.

Chapter 6

Discussion

The hyperparameters of the models were optimized using Bayesian optimization. Due to the very large dataset and the time constraint, all models were run for 100 iterations with an early stopping rounds of 10 during the optimization phase. However, during the training phase, all models were run for 1000 iterations with an early stopping rounds of 20, in order to maximize the learning done by the models. On another note, for the LSTM and MLP models, optimizing the number of layers and the number of nodes per layer at the same time using Bayesian optimization is not a straight forward task. In order to optimize these hyperparameters, Bayesian optimization was given a specific model structure, in which the number of nodes decrease by a certain factor when going from the last hidden layer to the first. The Bayesian optimization was then used to optimize the number of layers, the number of nodes for the last hidden layer, as well as the decay rate of the number of nodes. As a result, Baysian optimization did not have full flexibility over the model's structure. For example, a model with three layers and 64, 128, 128 nodes per layer cannot be created using the model structure described, and hence, it is possible that another model exists with a better performance that does not follow the specified model structure. Moreover, different model structures have different learning patterns depending on their complexity. Simple models are able to learn quickly but plateau at a higher error value, whereas complex models learn slowly but can plateau at a lower error value. Since the models were only ran for 100 iterations and 10 early stopping rounds during the optimization phase, it is possible that a better but more complex model did not have enough time to plateau. Therefore, a better model can possibly be found by giving

more flexibility over the structure of the model, exploring a higher range of values for each hyperparamter, as well as increasing the number of iterations and early stopping rounds during the optimization phase.

Furthermore, looking at Figures B.11, B.12, B.13, and B.14, we can see that the models have a difficult time forecasting the sales for the FOODS_3 department. This shows that items in the FOODS_3 department do not follow the same patterns as the other items. This can be confirmed by examining Figure 3.13, which shows the difference in the patterns of sales between the FOODS_3 department and the other departments. Not only do the items in the FOODS_3 department have higher sales, but they also show a seasonality pattern that is less visible in the other departments. Hence, a possible way to improve forecasting accuracies is to train a separate model for the items belonging to the FOODS_3 department.

LSTM models have a high reputation in learning from time-series data. However, the LSTM model did not perform as well as expected in this project. This can be due to the fact that recurrent neural networks, such as LSTM, require extensive non-sparse data. However, in our case, the data contains many zeros, which can significantly hinder the performance of the LSTM model. Moreover, the LSTM model was very sensitive to the values of its hyperparameters, and hence, it requires extensive hyperparameter optimization. On the other hand, LightGBM models are easy to train and are able to achieve a relatively good performance without much hyperparameter optimization. Although the results of the LGBM model are note worthy and show the strength of boosting methods in improving the overall performance, it is possible that a well optimized LSTM model can outperform the LGBM model, however, this would require a considerable amount of hyperparameter optimization.

Similarly, the hybrid model was not able to meet expectations. The performance of the hybrid model seemed to be very closely tied to the performance of its first component, LSTM, which is possibly due to the fact that the first component receives the main sales data and hence provides the main predictions. Moreover, although the second component,

LGBM, was able to slightly improve the accuracy of the 28-days ahead forecasts, it was not able to learn much from the residual data that was provided and seemed to have a difficult time finding a pattern and predicting the error of the LSTM model. Therefore, it could be possible for the hybrid model to beat the performance of its two individual components, however, its first component will need to perform at least as good as its second component when used as a singular model, since the first component will be doing the main learning in the hybrid model. Furthermore, another possible way to improve the performance of the hybrid model is to perform hyperparameter optimization on its two components. In this project, the hyperparameters of the singular models were used in the hybrid model. Since the input data for the two components are similar to the input data of the singular models, this can be considered a case of transfer learning, and hence the same hyperparameters should be able to perform relatively well. However, with further hyperparameter optimization, we should still be able to find better hyperparameters for each component of the hybrid model.

Another factor to consider when evaluating the performances is the choice of features used. The main data for the LSTM model included the sales time-series for each item, which is what the model will be predicting. In order to help with the predictions for the singular LSTM model, seven other features were used from the dataset, and two additional features were introduced. The seven features used from the dataset include the current day number, weekday, month, year, SNAP days for California, SNAP days for Texas, and SNAP days for Wisconsin. Moreover, two additional features were created from the available events data, which represented whether an event exists on the day of the prediction, and whether an event exists the day after the prediction. The reason that the model was provided with information about the events of the next day is that people tend to go shopping before an event rather than the day of the event. For example, in case of Halloween, a lot of people tend to buy their candies before Halloween starts rather than on the same day. Moreover, I also tried providing the model with the price information of

each item, as well as the exact name and type of each event, however, the performance of the LSTM model was significantly hindered, and hence, these features were omitted. This is possible due to the large number of features added, since there is a new feature added for the price of each item, and hence, we are essentially doubling our feature set.

For the MLP and LGBM models, all available features were used. In addition, the lag of sale values with intervals 28, 35, 42, and 56 days, and the rolling mean and standard deviation of the sale values with window sizes 7, 14, and 28 days and a lag of 28 days were introduced. Figure 6.1, which was obtained from the singular LGBM model, shows a list of these features and their significance in attaining the predictions. Note that all lag and rolling mean/std features were shifted by 28 days, since we have a forecasting horizon of 28 days. Shifting these features by the value of the forecasting horizon prevents the error of the model to propagate through later predictions. For example, if we introduce a lag feature with an interval of 7 and try to predict 28 days ahead, we will not have the actual sale values after the seventh forecast and we will have to use the predicted values instead. This means that if there is an error in the predictions, the error will propagate through later forecasts as well. However, I also tried providing the MLP and LGBM models with lag and

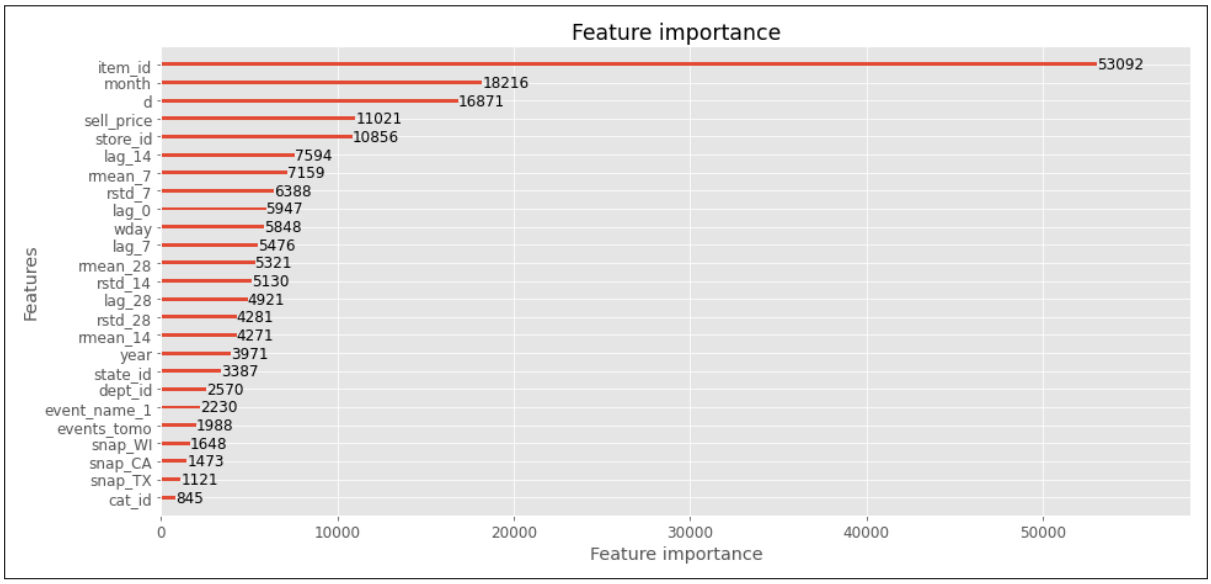


Figure 6.1: The importance of each feature obtained from the LGBM model.

rolling mean/std features that are within the forecasting horizon. Even though this boosted the performances on the 1-day ahead forecasts, it significantly hindered the performances on the 28-days ahead forecasts. Although, it is important to note the importance of recent sales in the foretelling the future sales. Having knowledge of recent sale values can greatly help with the predictions, however, this comes with the trade-off of error propagation when forecasting longer horizons. Therefore, further optimization can be possibly done by providing the models with lag and rolling mean/std features that are within the forecasting horizon, and balancing the aforementioned trade-off between recency of these features and error propagation. Moreover, the reason that the 1-day and 28-days ahead forecasts of the MLP and LGBM models are the same is the fact that all lag and rolling mean/std features are shifted by the value of the forecasting horizon, and so actual sale values are available for the entire forecasting horizon. However, this is not the case for the LSTM model, since the LSTM model will have to depend on its own predictions when it has a large forecasting horizon, and hence the LSTM and hybrid models obtain different 1-day and 28-days ahead predictions.

Chapter 7

Conclusions and Future Work

The performance of the LGBM model highlights the potential of boosting methods in improving the overall performance. Compared to the MLP model, which had the same preprocessing procedure as the LGBM model, not only was the LGBM model easier to construct, but it was also able to train much faster, which shows the ability of LGBM models in working with large datasets. Moreover, from the results of the LSTM and MLP models, we can clearly see the ability of LSTM models in working with time-series data compared to regular deep learning models.

Another goal of this paper was to evaluate the capabilities of hybrid models. Unlike the study done in [13], we did not find hybrid models to be useful for improving the performance. In our case, the performance of the hybrid model was closely tied to the performance of its first component. Future studies should further explore the capabilities of hybrid models and how much their performance depends on the performance of the first component. Moreover, future work on this dataset can focus on further hyperparameter optimization and feature engineering to improve the performance of the LSTM model.

All models in this paper were trained on the entire dataset. However, the results show that the models were unable to confidently forecast the sales for the $\mathrm{FOODS\_3}$ department. Hence, future studies studies can try to enhance the performances by training a separate model for the $\mathrm{FOODS\_3}$ department, or even one model per department. It is however important to balance the number of models in order to not lose too much inter-dependency between the different item groups.

Appendices

Appendix A

Tables

Evaluation Metrics

Table A.1: The evaluation metrics of the LSTM model.

Dataset	ME	RMSE	RMSSE
Train (1-day)	0.02139	2.10657	0.78320
Valid (1-day)	-0.00857	2.11495	0.82372
Test (1-day)	-0.10333	2.24916	0.85677
Test (28-days)	-0.10991	2.27780	0.80511

Table A.2: The evaluation metrics of the MLP model.

Dataset	ME	RMSE	RMSSE
Train (1-day)	-0.05645	2.73888	0.94049
Valid (1-day)	-0.03643	2.24467	0.87421
Test (1-day)	-0.08568	2.29376	0.87376
Test (28-days)	-0.08568	2.29376	0.81075

Table A.3: The evaluation metrics of the LGBM model.

Dataset	ME	RMSE	RMSSE
Train (1-day)	-0.00001	2.34224	0.80429
Valid (1-day)	-0.00679	2.11933	0.82540
Test (1-day)	-0.07056	2.18648	0.83289
Test (28-days)	-0.07056	2.18648	0.77283

Table A.4: The evaluation metrics of the hybrid LSTM-LGBM model.

Dataset	ME	RMSE	RMSSE
Train (1-day)	0.01417	2.10155	0.78133
Valid (1-day)	-0.04265	2.11706	0.82453
Test (1-day)	-0.14021	2.26268	0.86192
Test (28-days)	-0.11916	2.25698	0.79775

Hyperparameters

Table A.5: LSTM hyperparameters ranges and their optimal values.

Hyperparameter	Range	Optimal value	Description
n_layers	Integer(3, 5)	3	number of layers
last_units	Integer(128, 512)	405	number of nodes for the last hidden layer
units_decay	Float(0.2, 1)	0.752032	decay rate of the number of nodes
learning_rate	Float(0.0001, 0.01)	0.000534	learning rate of the model
lr_decay	Float(0, 0.001)	0.000445	decay rate of the learning rate
dropout_rate	Float(0, 0.3)	0.225848	dropout rate for the dropout layers
norm	Integer(0, 1)	0	whether to add batch normalization layers
batch_size	Float(32, 128)	59	batch size for the training phase
steps	Integer(14, 56)	23	number of time steps for LSTM data

Table A.6: MLP hyperparameters ranges and their optimal values.

Hyperparameter	Range	Optimal value	Description
n_layers	Integer(3, 5)	4	number of layers
last_units	Integer(64, 512)	467	number of nodes for the last hidden layer
units_decay	Float(0.2, 1)	0.699088	decay rate of the number of nodes
learning_rate	Float(0.0001, 0.01)	0.004514	learning rate of the model
lr_decay	Float(0, 0.001)	0.000478	decay rate of the learning rate
dropout_rate	Float(0, 0.3)	0.006266	dropout rate for the dropout layers
norm	Integer(0, 1)	0	whether to add batch normalization layers
batch_size	Float(32000, 100000)	57798	batch size for the training phase

Table A.7: LGBM hyperparameters ranges and their optimal values.

Hyperparameter	Range	Optimal value	Description
learning_rate	Float(0.001, 0.5)	0.097892	learning rate of the model
feature_fraction	Float(0.2, 1)	0.553332	fraction of features used to train each tree
lambda_l2	Float(0, 0.3)	0.219554	value of lambda for L2 regularization
num_leaves	Integer(50, 5000)	749	maximum number of leaves in one tree
min_data_in_leaf	Integer(10, 5000)	1534	Minimum number of data in each leaf

Appendix B

Figures

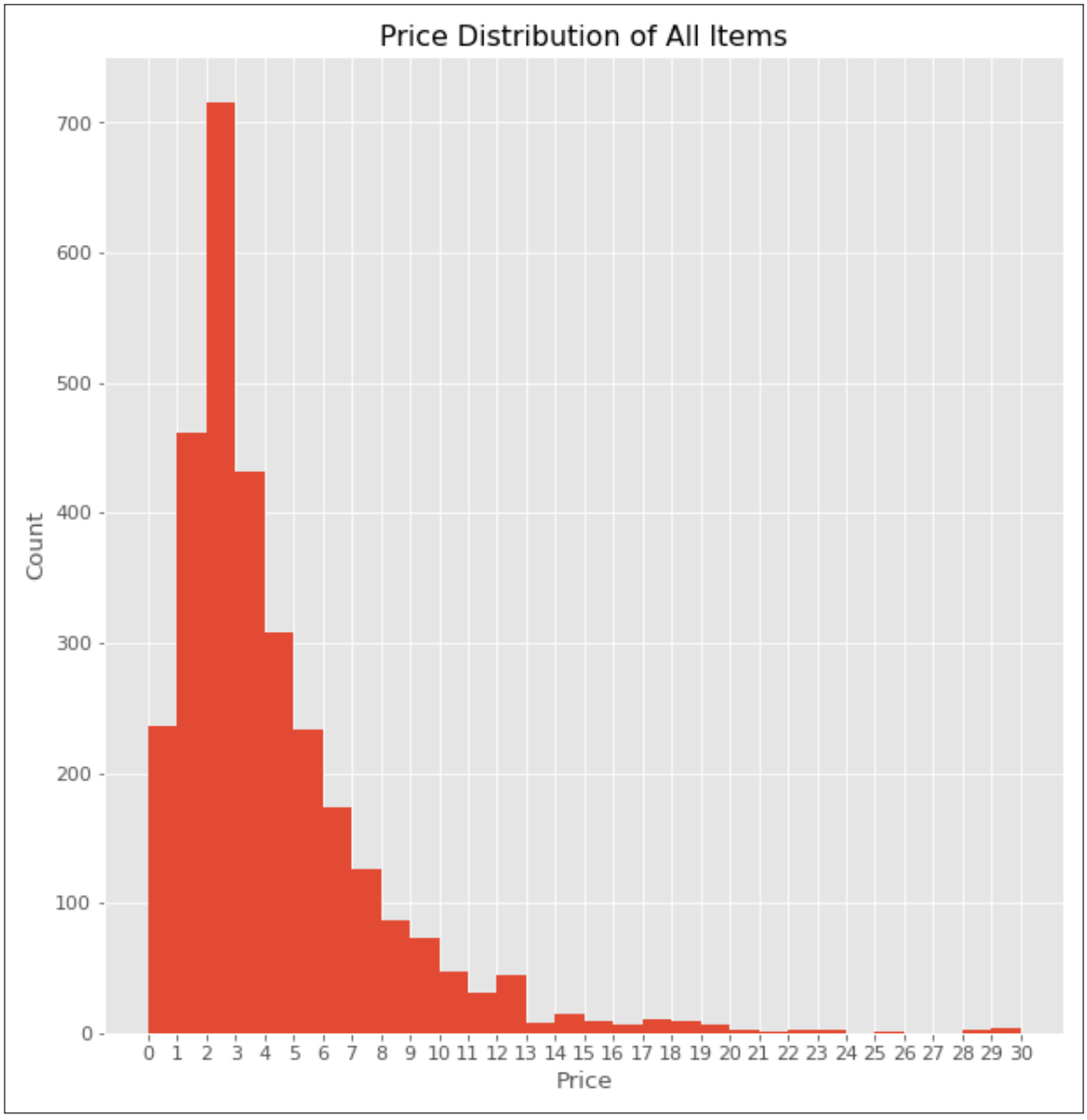


Figure B.1: The distribution of the prices of all products.

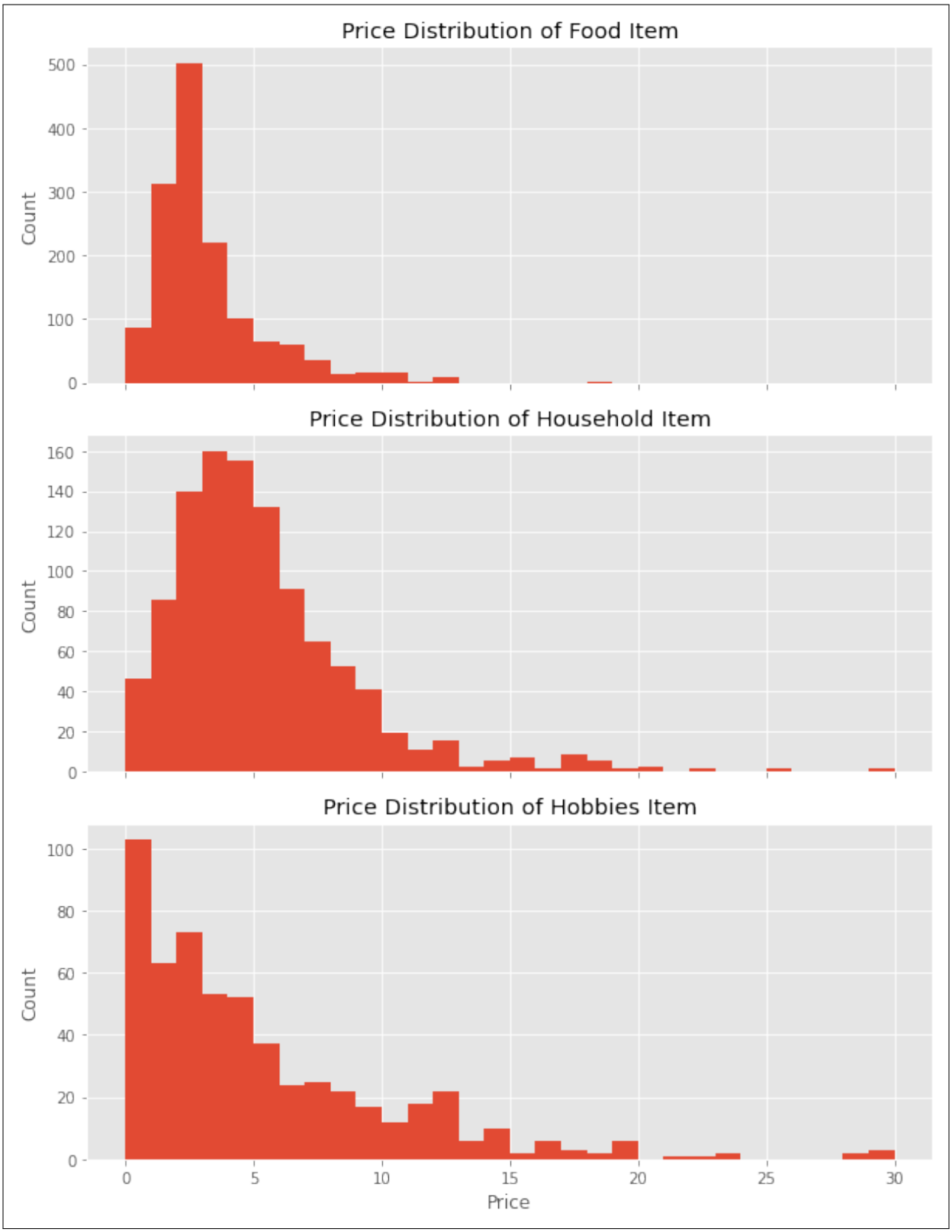


Figure B.2: The price distribution of products in each product category.

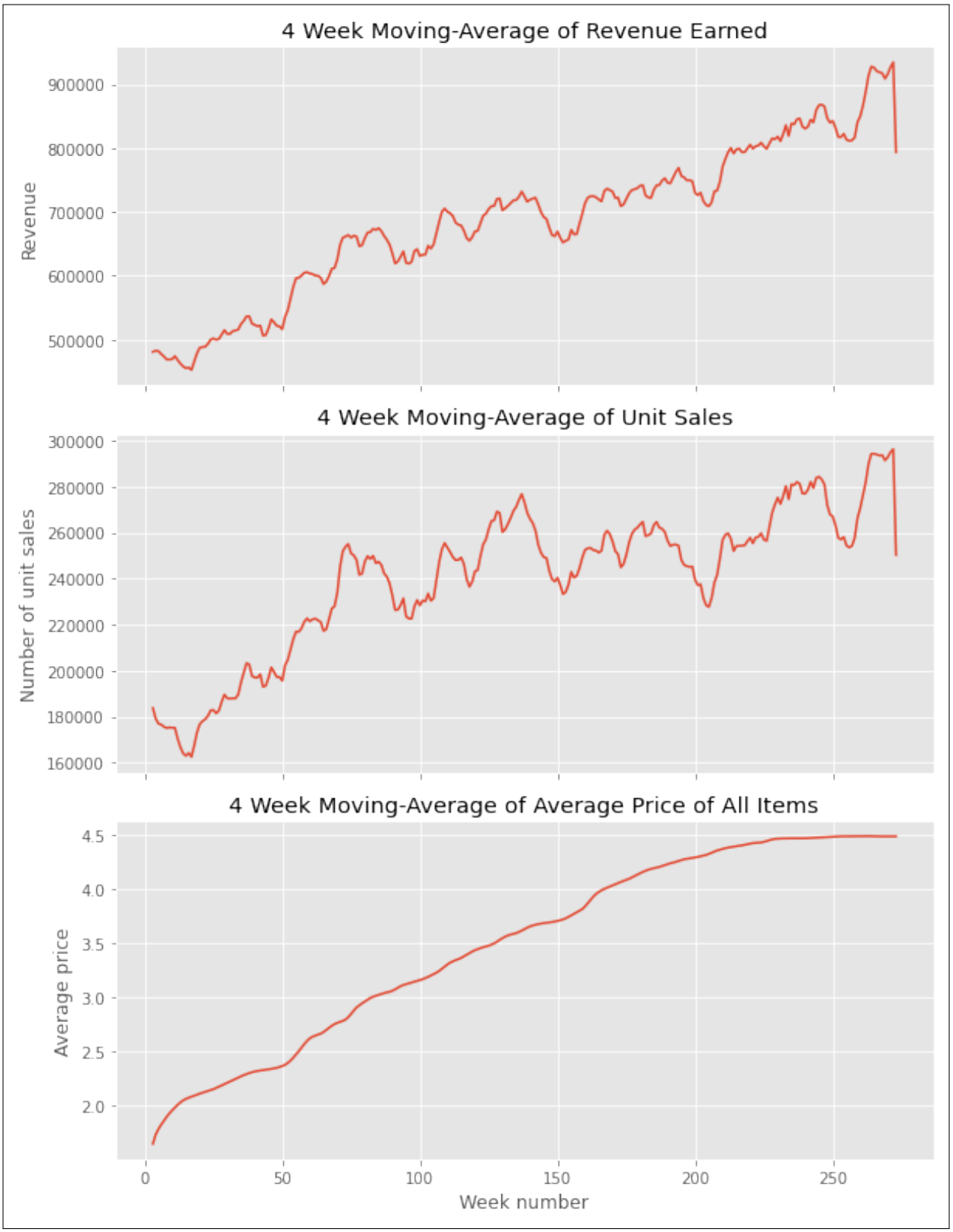


Figure B.3: 4 week moving-average of revenue, unit sales, and average price of items.

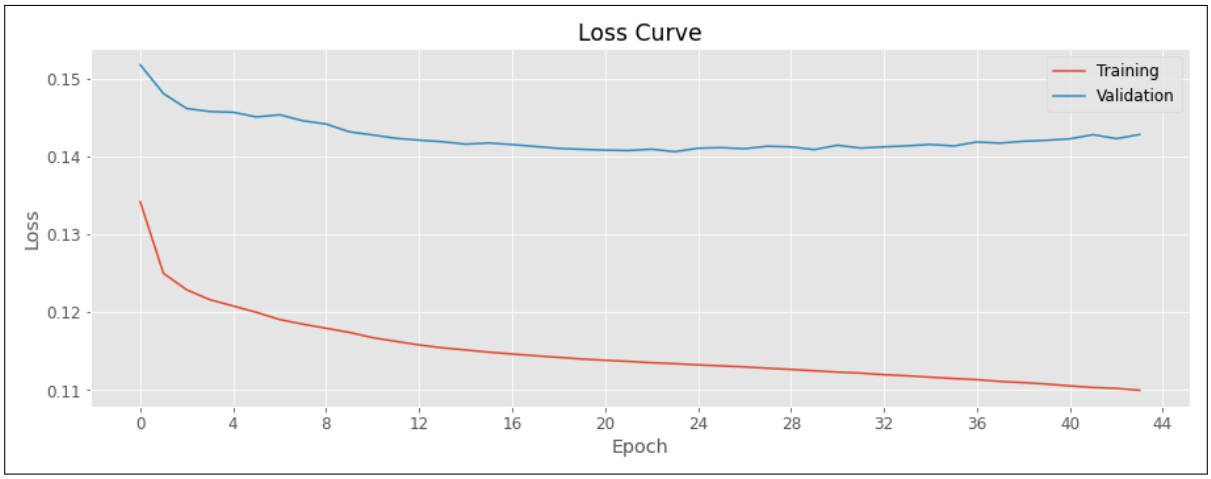


Figure B.4: The learning curve of the LSTM model.

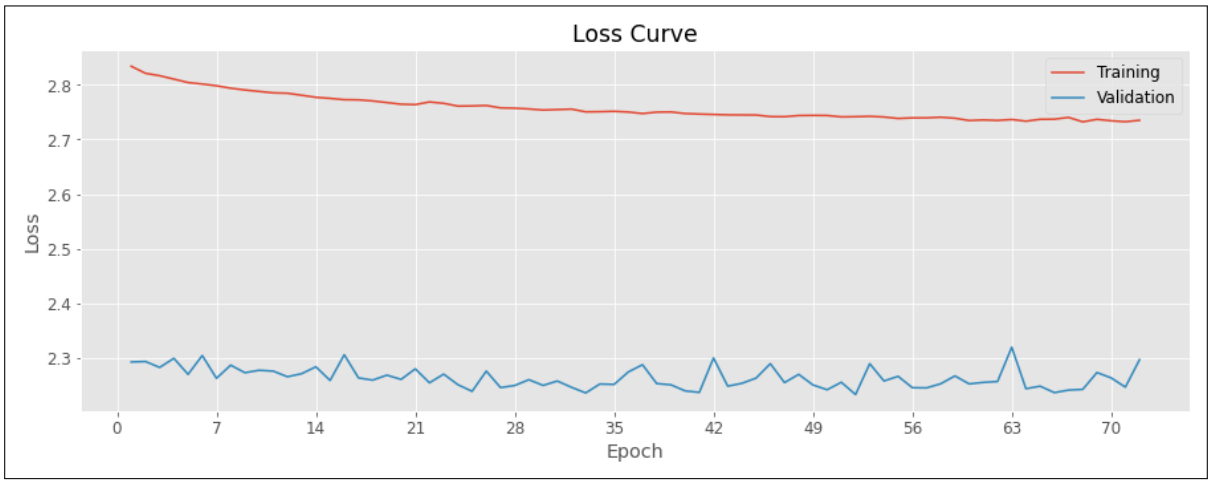


Figure B.5: The learning curve of the MLP model.

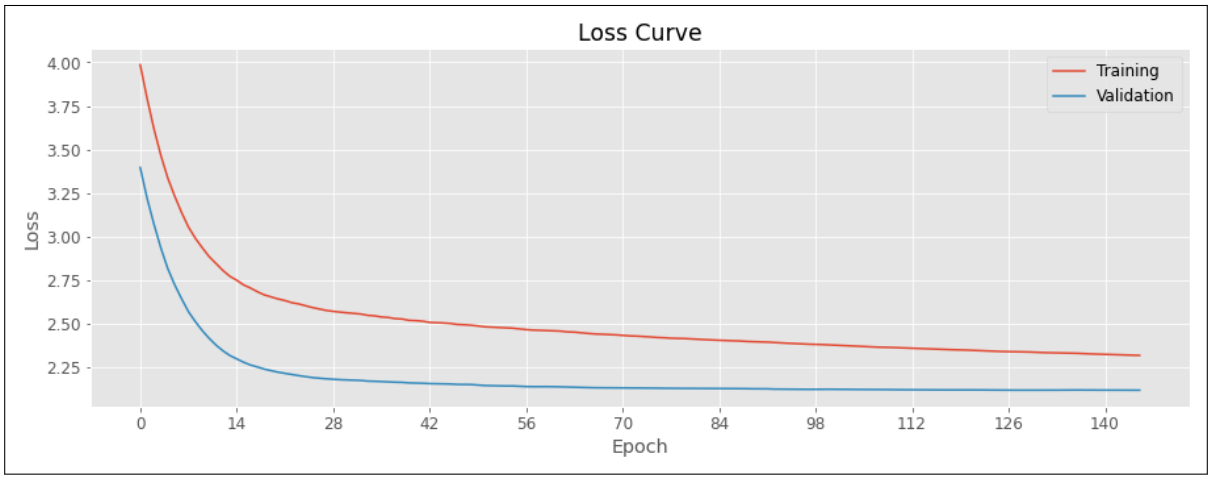
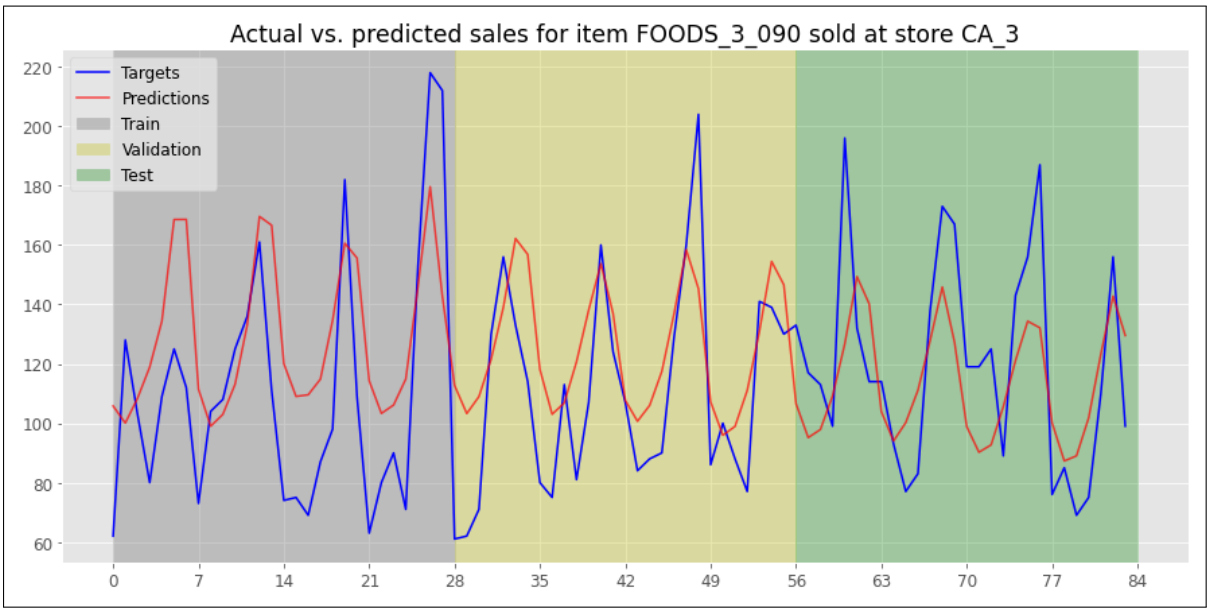
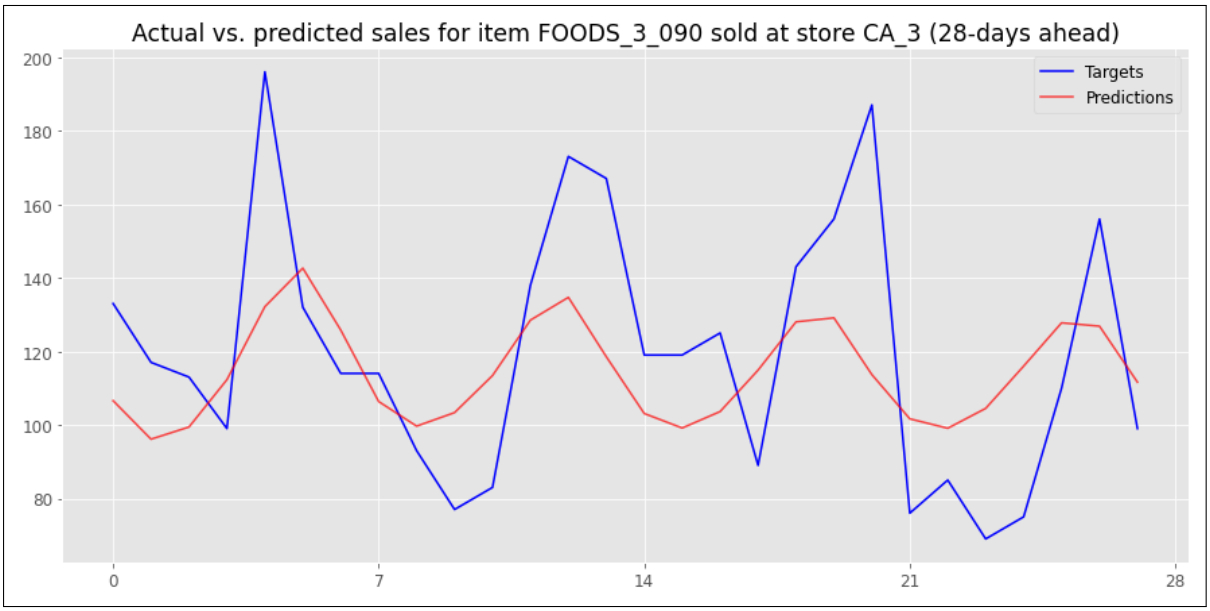


Figure B.6: The learning curve of the LGBM model.



(a) 1-day ahead predictions on the training, validation, and test sets.



(b) 28-days ahead predictions on the test set

Figure B.7: Samples of LSTM predictions vs. the actual values.

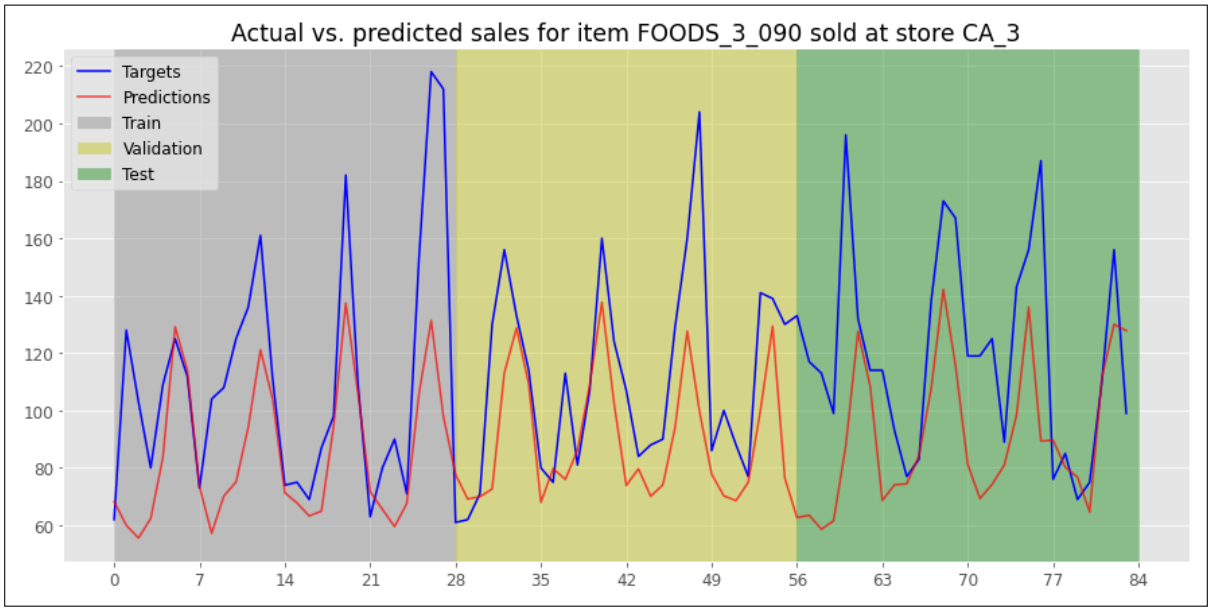


Figure B.8: Samples of MLP predictions vs. the actual values.

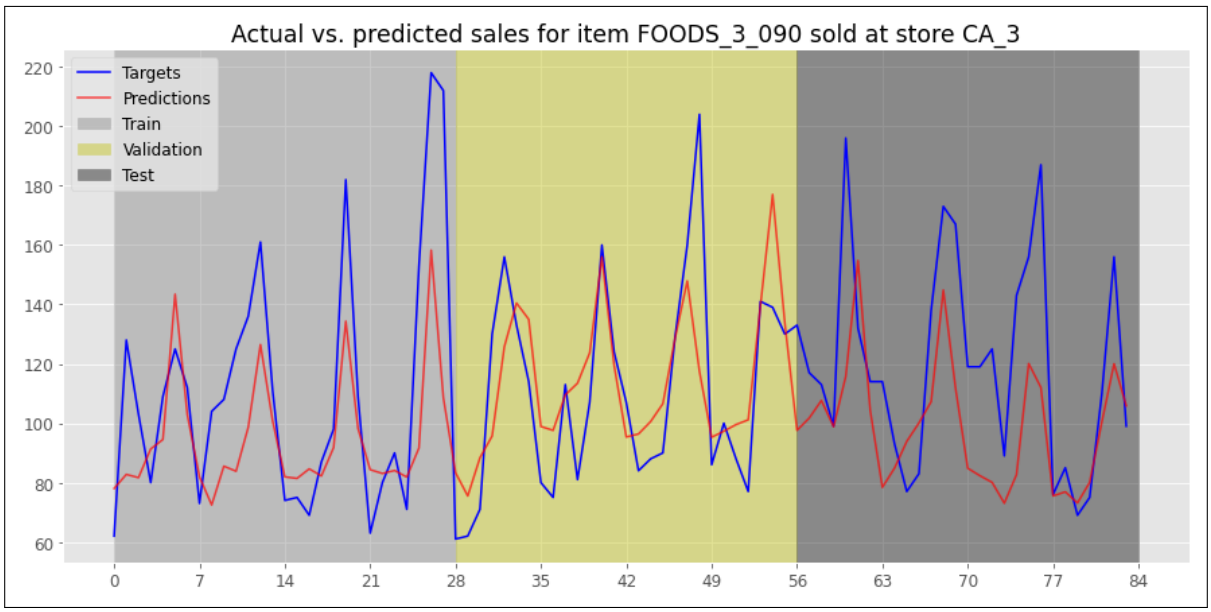
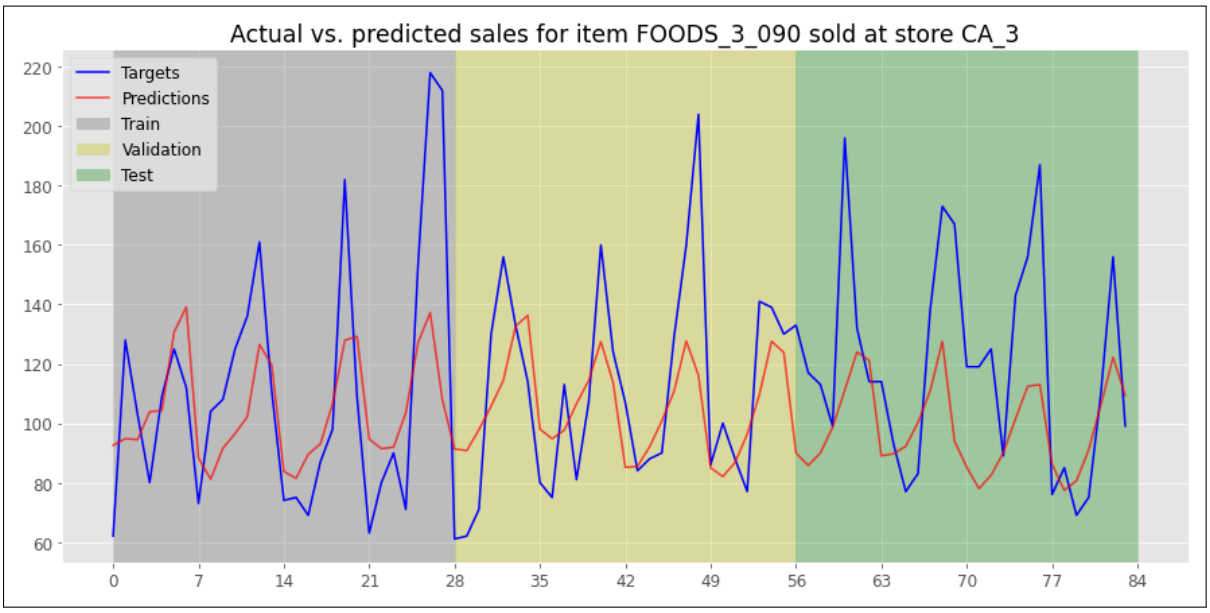
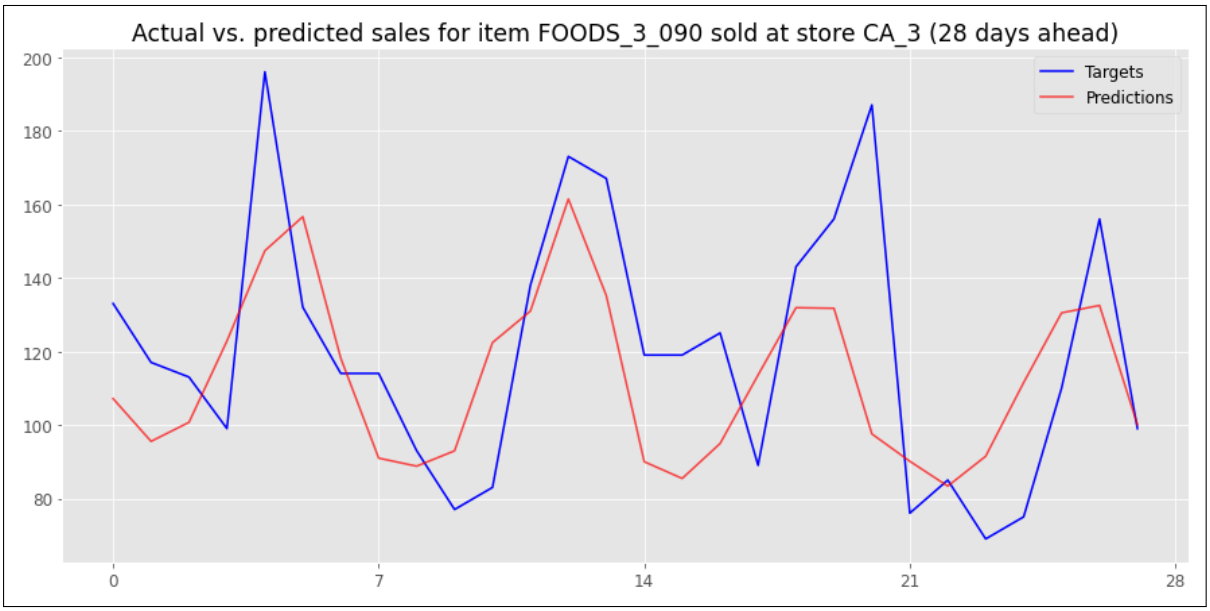


Figure B.9: Samples of LGBM predictions vs. the actual values.



(a) 1-day ahead predictions on the training, validation, and test sets.



(b) 28-days ahead predictions on the test set

Figure B.10: Samples of the LSTM-LGBM hybrid model's predictions vs. the actual values.

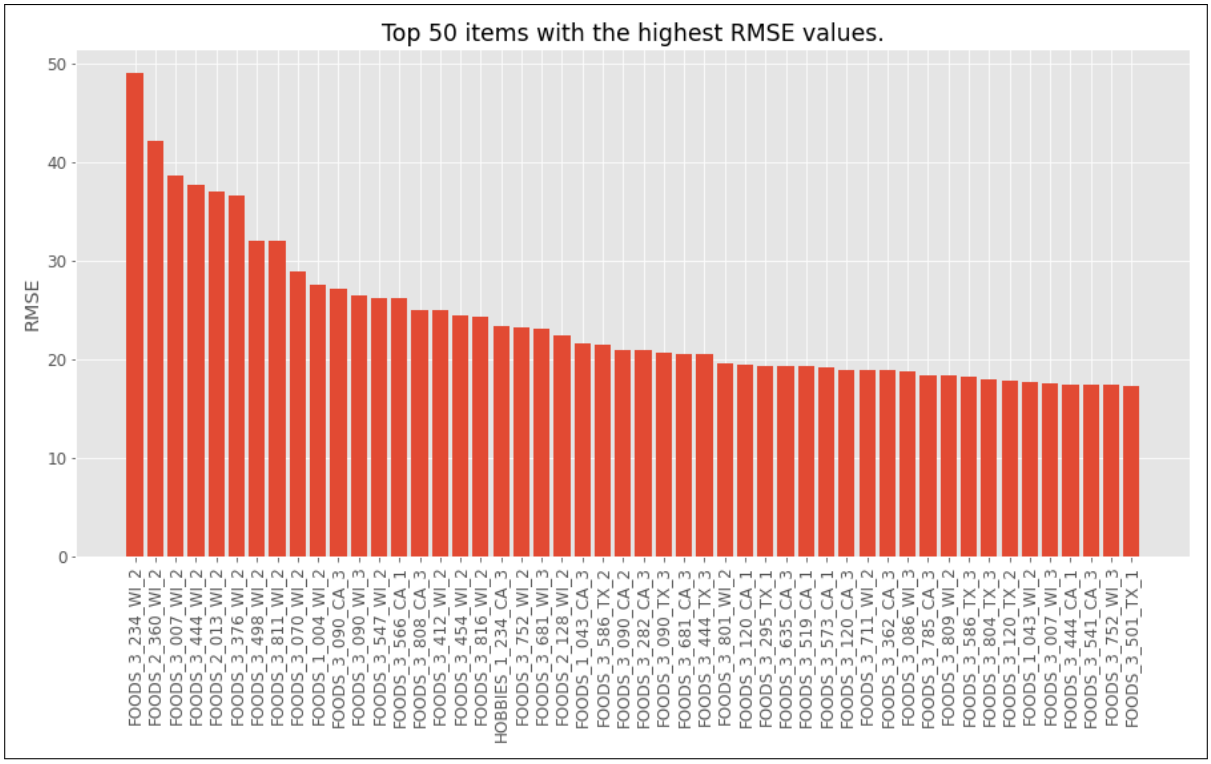


Figure B.11: Top 50 items with the highest RMSE for the LSTM model.

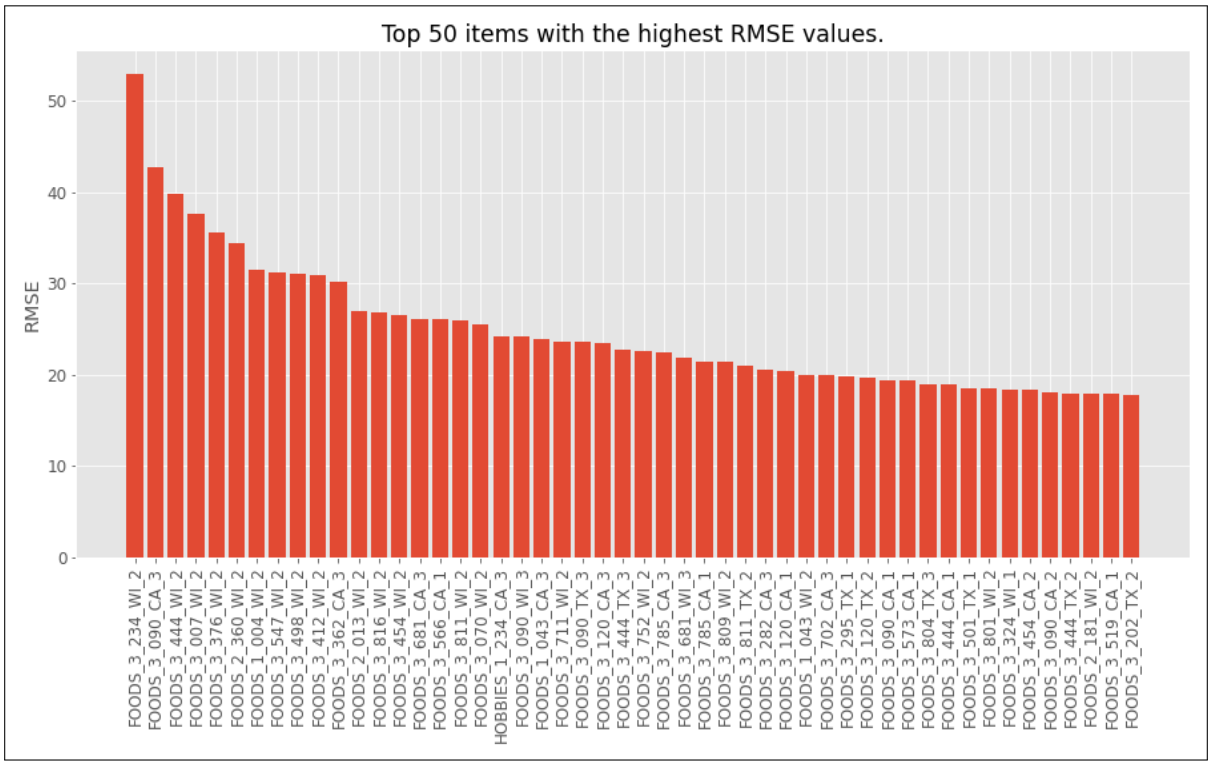


Figure B.12: Top 50 items with the highest RMSE for the MLP model.

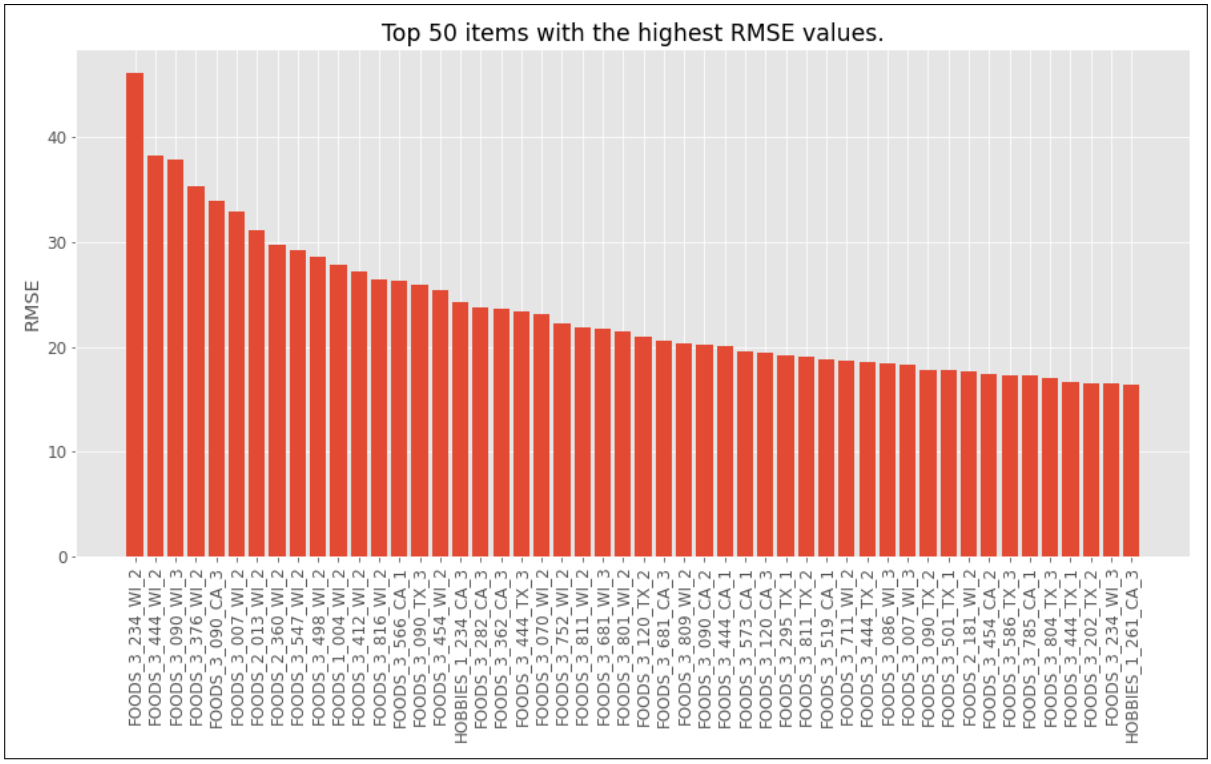


Figure B.13: Top 50 items with the highest RMSE for the LGBM model.

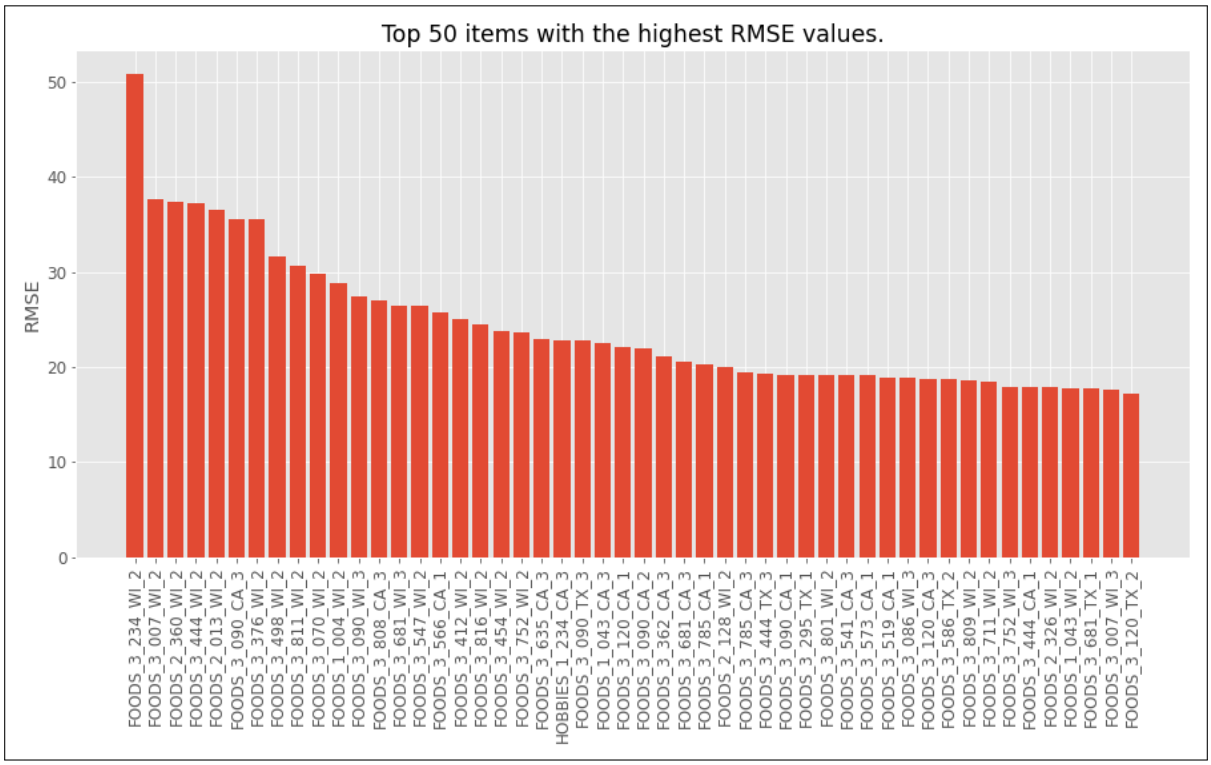


Figure B.14: Top 50 items with the highest RMSE for the LSTM-LGBM hybrid model.

Bibliography

[1] Q. Yu, K. Wang, J. O. Strandhagen, and Y. Wang, "Application of long short-term memory neural network to sales forecasting in retail—a case study," pp. 11–17, 2017.

[2] Z.-L. Sun, T.-M. Choi, K.-F. Au, and Y. Yu, "Sales forecasting using extreme learning machine with applications in fashion retailing," *Decision Support Systems*, vol. 46, no. 1, pp. 411 – 419, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923608001371

[3] T. Weng, W. Liu, and J. Xiao, "Supply chain sales forecasting based on lightgbm and lstm combination model," *Industrial Management Data Systems*, vol. 120, no. 2, pp. 265–279, 2019;2020.

[4] Z.-Y. Chen and R. J. Kuo, "Evolutionary algorithm-based radial basis function neural network training for industrial personal computer sales forecasting," *Computational Intelligence*, vol. 33, no. 1, pp. 56–76, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/coin.12073

[5] S. Q. Mueller, "Pre- and within-season attendance forecasting in major league baseball: a random forest approach," *Applied Economics*, vol. 0, no. 0, pp. 1–18, 2020. [Online]. Available: https://doi.org/10.1080/00036846.2020.1736502

[6] B. Shao, M. Li, Y. Zhao, and G. Bian, "Nickel price forecast based on the lstm neural network optimized by the improved pso algorithm," *Mathematical Problems in Engineering*, vol. 2019, pp. 1–15, 2019.

[7] M. Scherer, "Multi-layer neural networks for sales forecasting," *Journal of Applied Mathematics and Computational Mechanics*, vol. 17, no. 1, pp. 61–68, 2018.

[8] S. Helmini, N. Jihan, M. Jayasinghe, and S. Perera, "Sales forecasting using multivariate long short term memory network models," *PeerJ PrePrints*, 2019.

[9] H. Weytjens, E. Lohmann, and M. Kleinsteuber, "Cash flow prediction: Mlp and lstm compared to arima and prophet," *Electronic Commerce Research*, 2019.

[10] C. P. d. Veiga, C. R. P. d. Veiga, W. Puchalski, L. d. S. Coelho, and U. Tortato, "Demand forecasting based on natural computing approaches applied to the foodstuff retail segment," *Journal of Retailing and Consumer Services*, vol. 31, pp. 174–181, 2016.

[11] Ö. G. Ali, S. Sayın, T. Van Woensel, and J. Fransoo, "Sku demand forecasting in the presence of promotions," *Expert Systems with Applications*, vol. 36, no. 10, pp. 12 340–12 348, 2009.

[12] T. Huang, R. Fildes, and D. Soopramanien, "The value of competitive information in forecasting fmcg retail product sales and the variable selection problem," *European Journal of Operational Research*, vol. 237, no. 2, pp. 738–748, 2014.

[13] S. Punia, K. Nikolopoulos, S. P. Singh, J. K. Madaan, and K. Litsiou, "Deep learning with long short-term memory networks and random forests for demand forecasting in multi-channel retail," *International Journal of Production Research*, pp. 1–16, 2020.

[14] T. Taskaya-Temizel and M. C. Casey, "A comparative study of autoregressive neural network hybrids," *Neural Networks*, vol. 18, no. 5, pp. 781–789, 2005.

[15] Kaggle, "M5 forecasting - accuracy," University of Nicosia, 2020. [Online]. Available: https://www.kaggle.com/c/m5-forecasting-accuracy/

[16] "The m5 competition," M Open Forecasting Center (MOFC), 2020. [Online]. Available: https://mofc.unic.ac.cy/m5-competition/

[17] M. Kuhn and D. Vaughan, "Mean absolute scaled error - mase," yardstick. [Online]. Available: https://yardstick.tidymodels.org/reference/mase.html