Major Research Project
Walmart Sales Forecasting

Results

Moeen Bagheri

August 2020

Chapter 1

Results

The following four models were constructed:

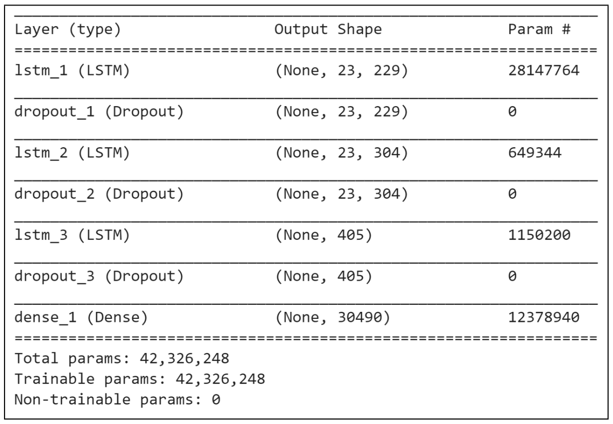
1. Long Short-Term Memory (LSTM)
2. Multi-Layer Perceptron (MLP)
3. LightGBM (LGBM)
4. LSTM-LGBM Hybrid

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 1, 2, 3, 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.

1.1 Long Short-Term Memory

Figure 1.1 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 3a 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 3b. 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
```

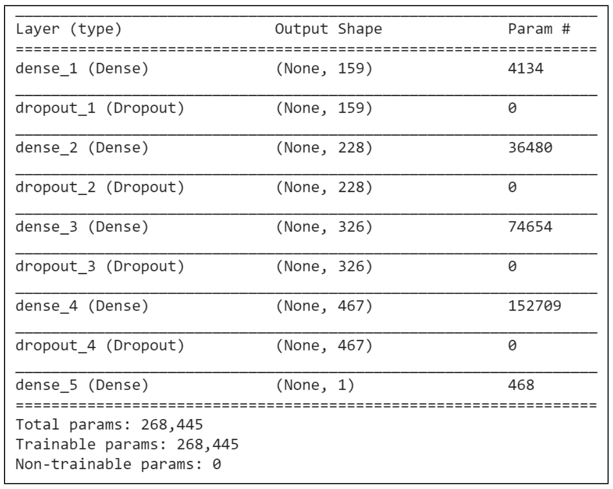
Figure 1.1: A summary of the structure of the LSTM model.

Figure 7 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.

The number of hidden layers, number of hidden units, learning rate, learning rate decay, dropout rate, batch size, and the number of historical observations for the LSTM model were optimized using Bayesian optimization. Moreover, Bayesian optimization was used to decide whether a batch normalization layer should be included in the architecture of the model after each LSTM layer. 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 5 shows the range of values that each hyperparameter was optimized for, as well as the optimized values.

1.2 Multilayer Perceptron

A summary of the MLP model's structure is shown in Figure 1.2. The RMSE of the MLP model on the training, validation, and test sets were 2.73888, 2.24467, and 2.29376, respectively. Figure 8 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 1.2: A summary of the structure of the MLP model.

The number of hidden layers, number of hidden units, learning rate, learning rate decay, dropout rate, and batch size of the MLP model were optimized using Bayesian optimization. The optimization process was also used to decide whether a batch normalization layer should be included in the structure of the model. The range of values that each hyperparameter was optimized on, as well as their optimized values are shown in Table 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.

1.3 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 9, 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.

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. Table 7 shows a the range of values that each hyperparameter was optimized for 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.

1.4 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 3b and 6b. 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 2

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 the Figures 10, 11, 12, and 13, 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 2.1, 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

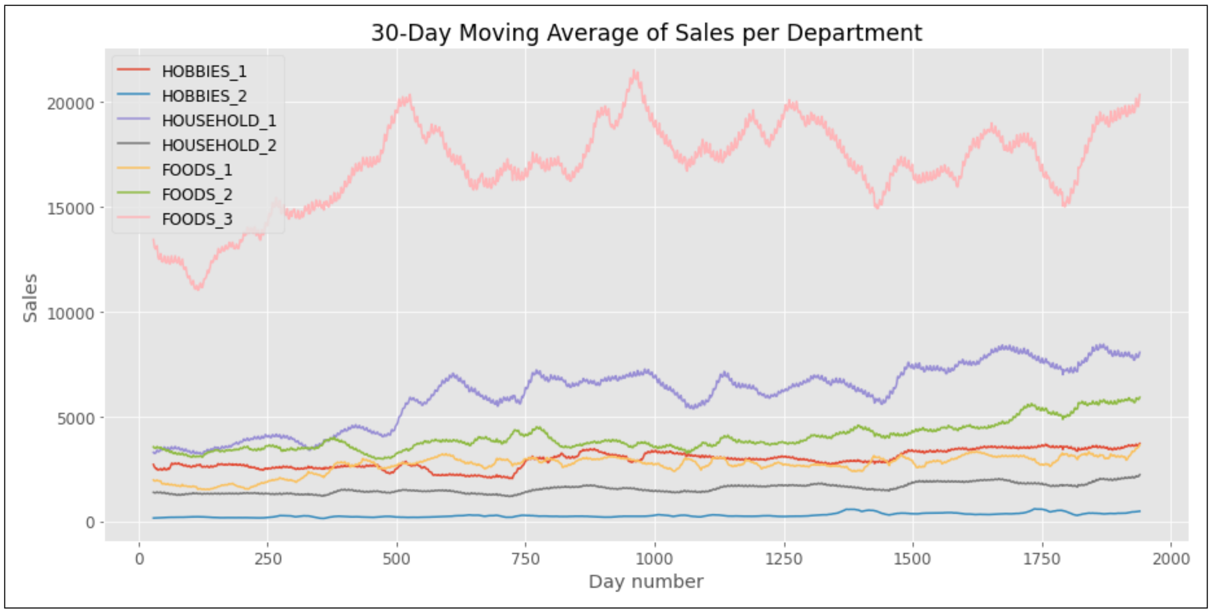


Figure 2.1: 30-day moving average of sales grouped by department.

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 2.2, 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

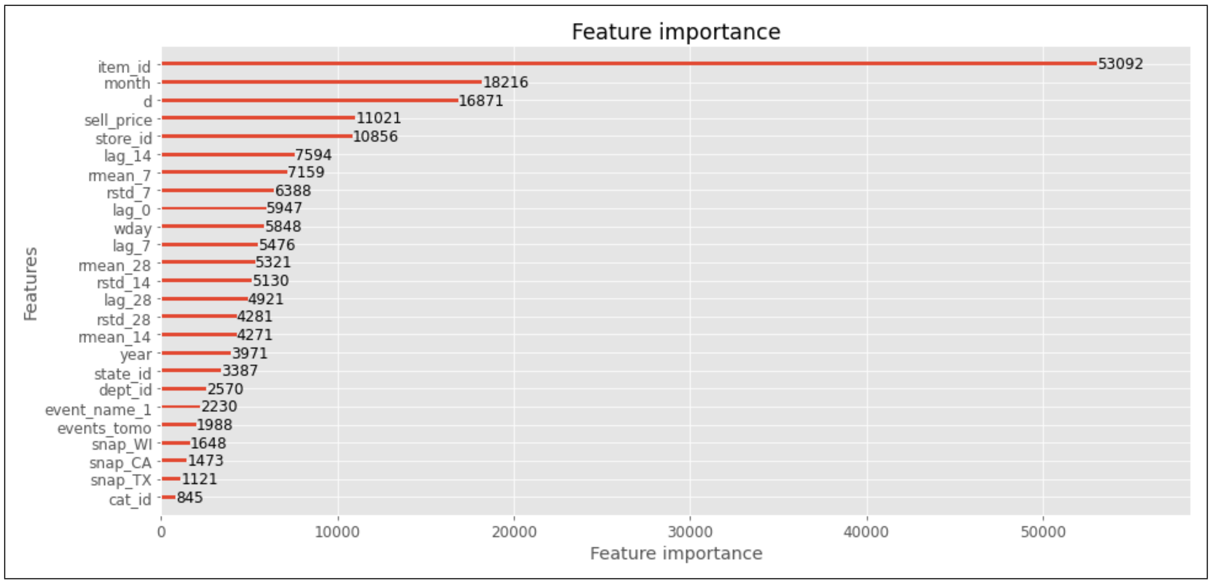


Figure 2.2: The importance of each feature obtained from the LGBM model.

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 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.

Appendices

Tables

Evaluation Metrics

Table 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 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 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 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 5: LSTM hyperparameters ranges for the optimization phase 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

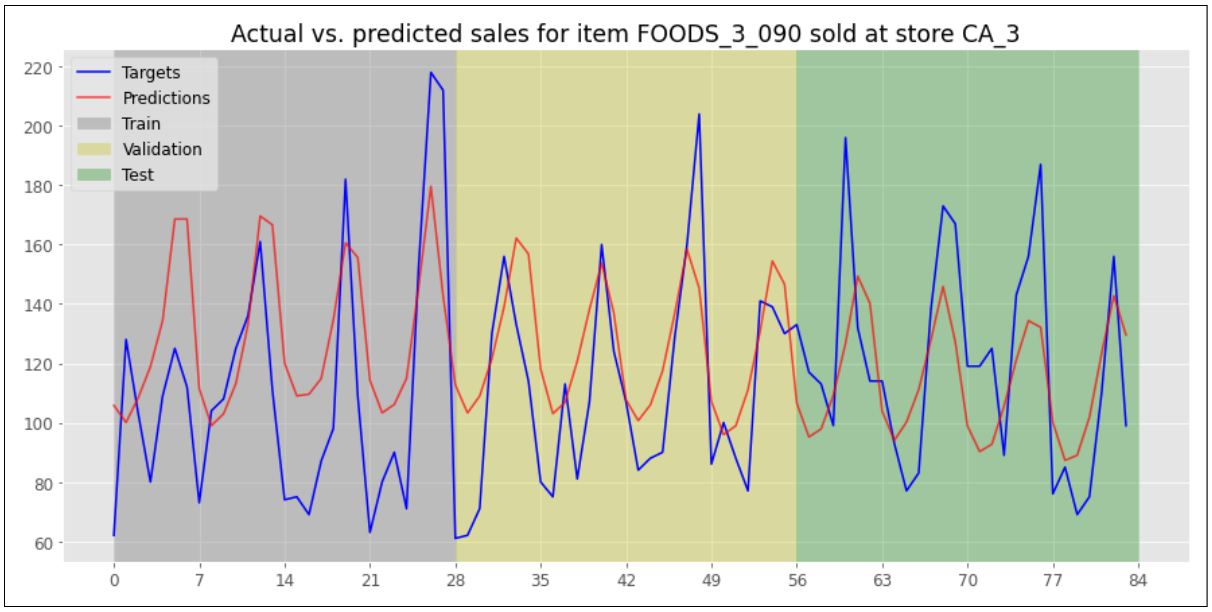
Table 6: MLP hyperparameters ranges for the optimization phase and their optimal values.

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

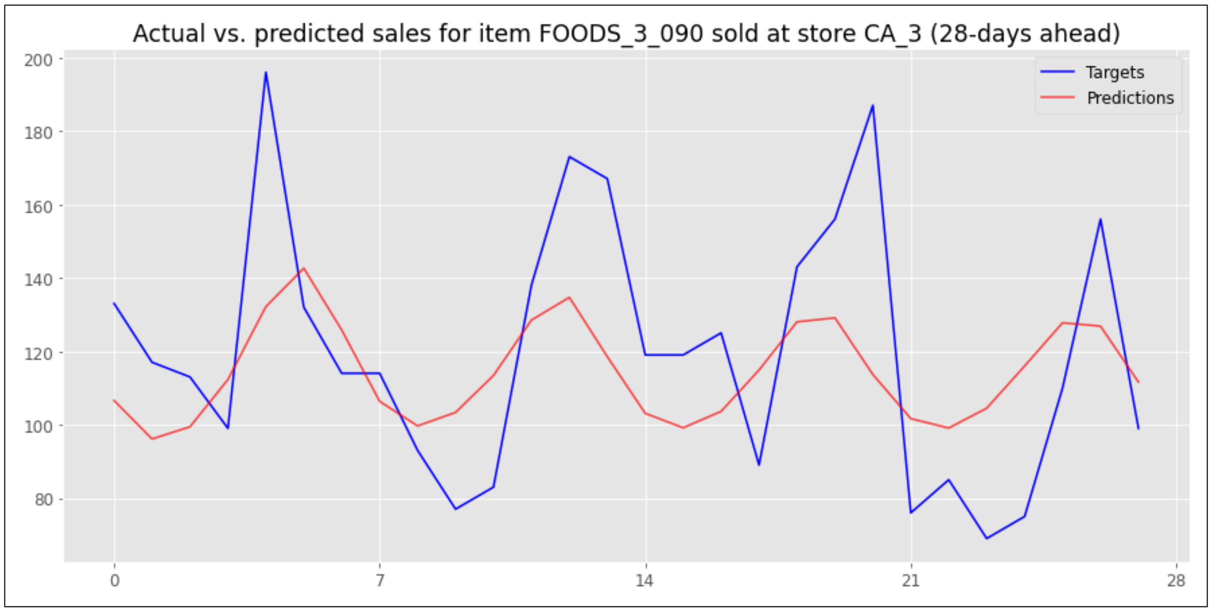
Table 7: LGBM hyperparameters ranges for the optimization phase and their optimal values.

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

Figures



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



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

Figure 3: Samples of LSTM predictions vs. the actual values.

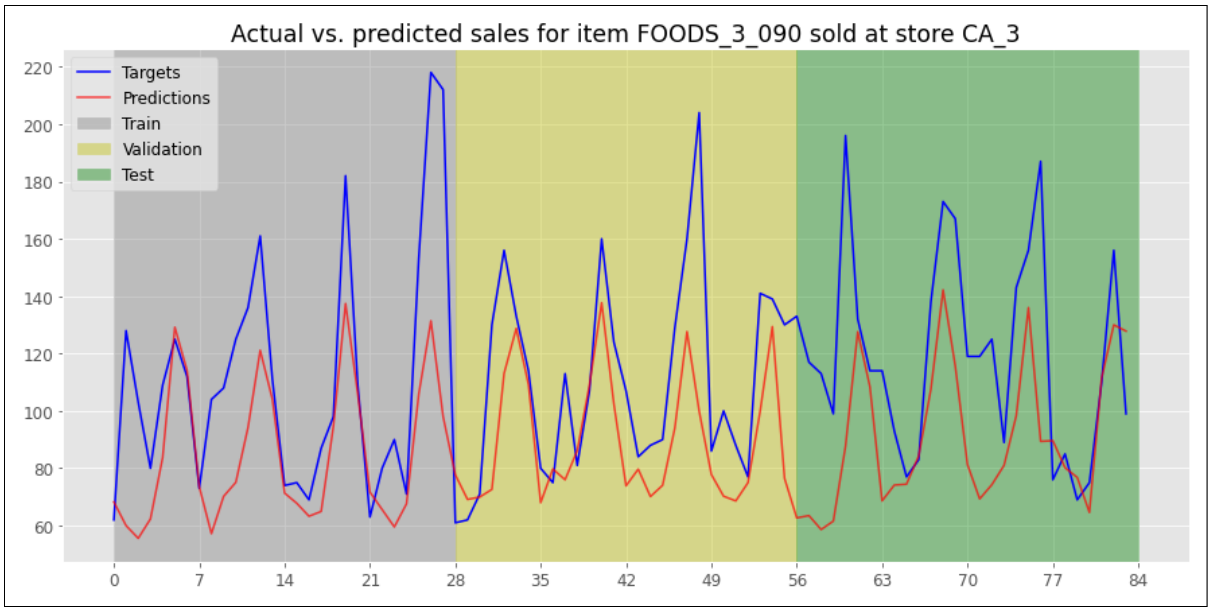


Figure 4: Samples of MLP predictions vs. the actual values.

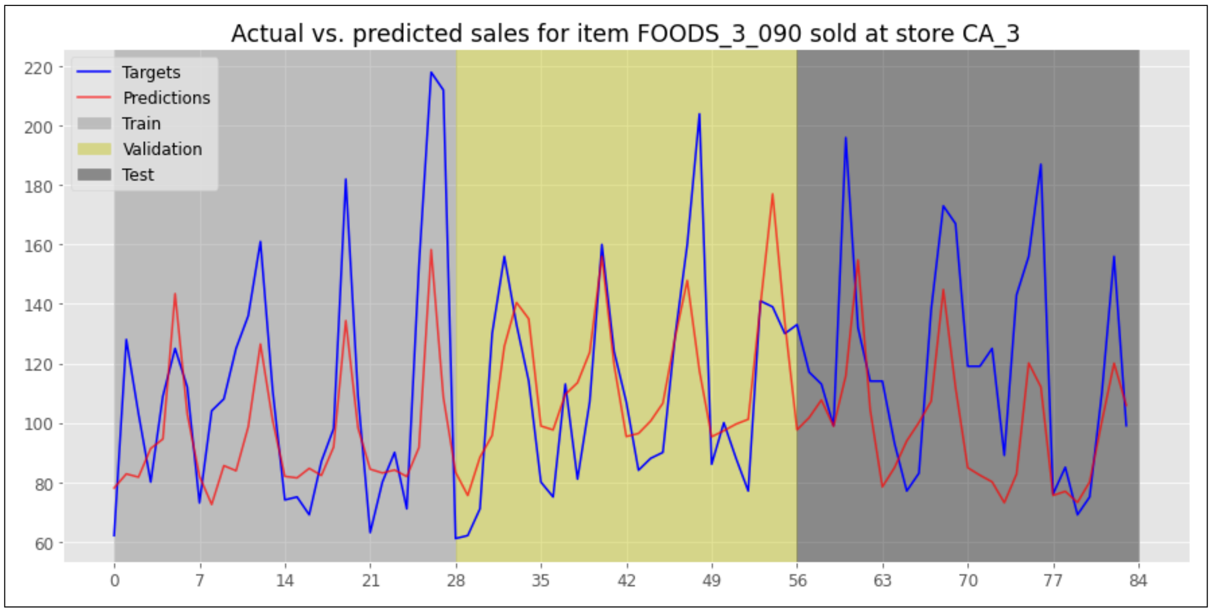
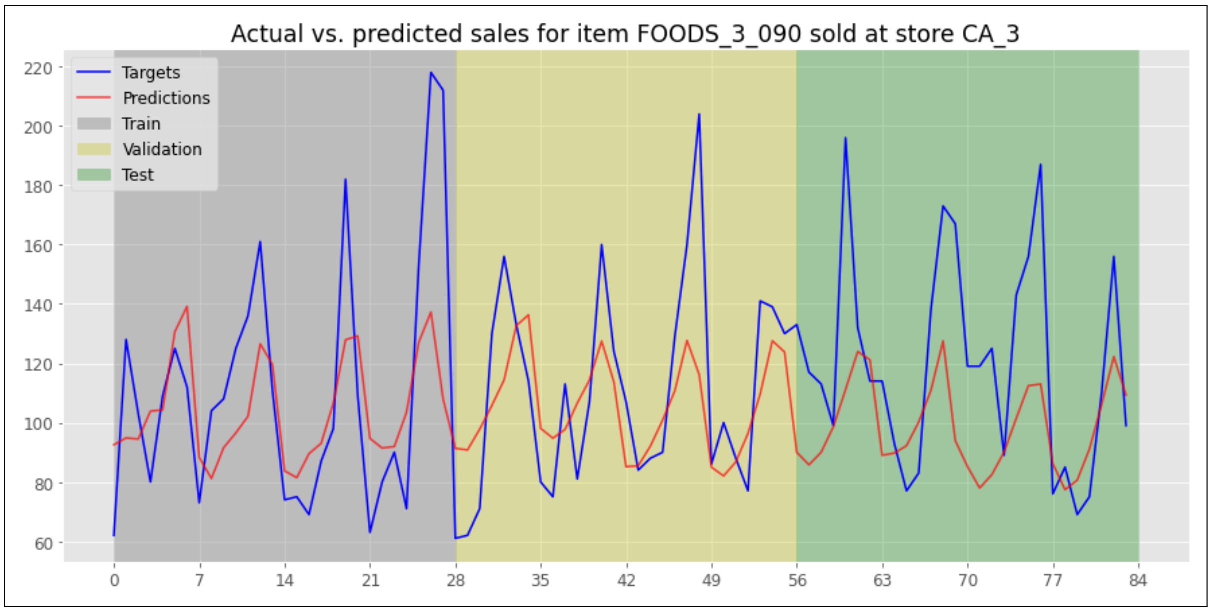
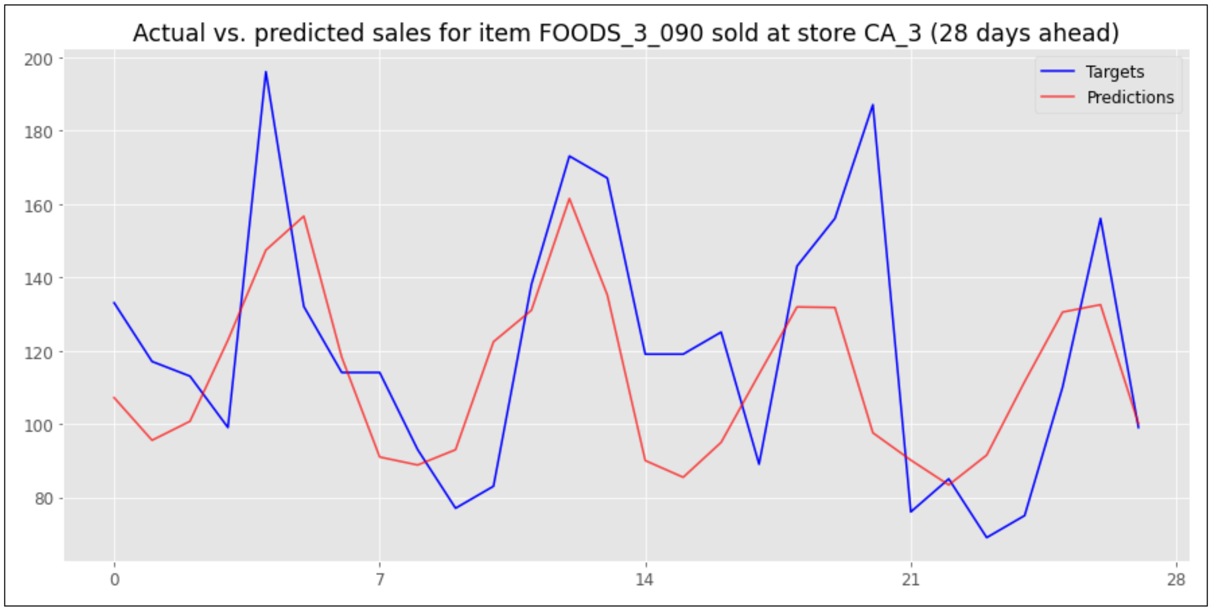


Figure 5: Samples of LGBM predictions vs. the actual values.



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



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

Figure 6: Samples of the LSTM-LGBM hybrid model's predictions vs. the actual values.

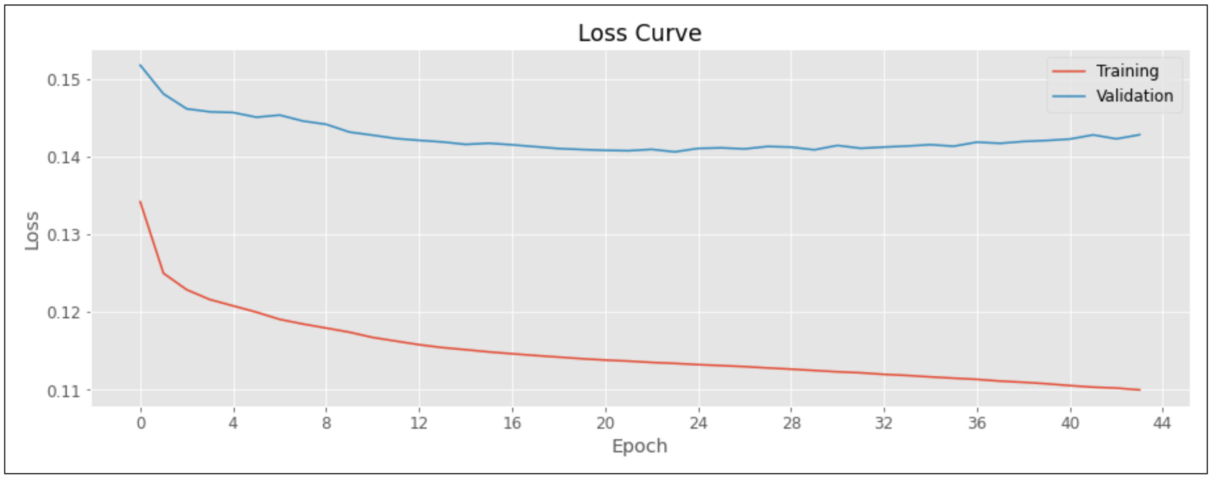


Figure 7: The learning curve of the LSTM model.

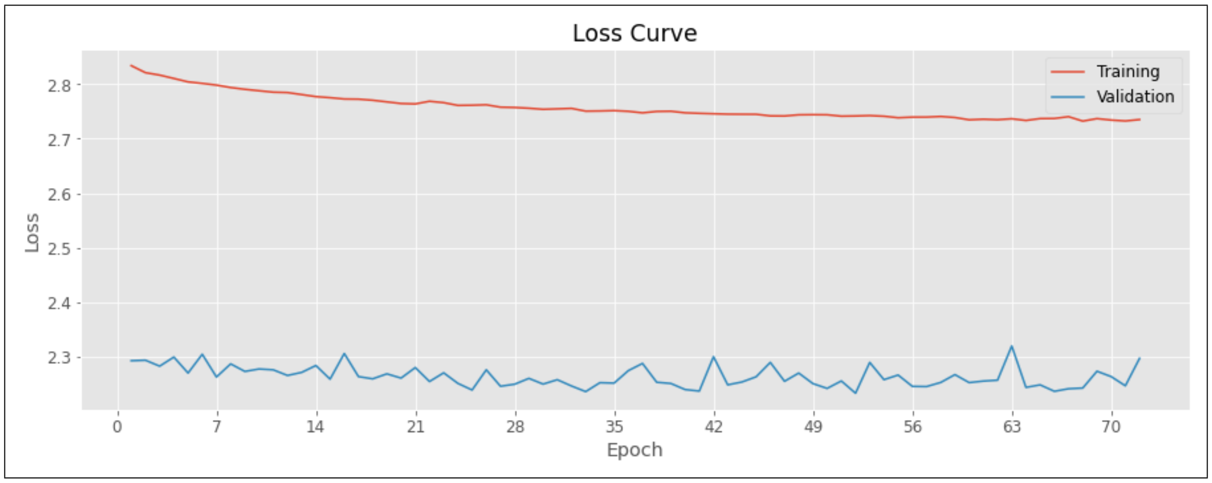


Figure 8: The learning curve of the MLP model.

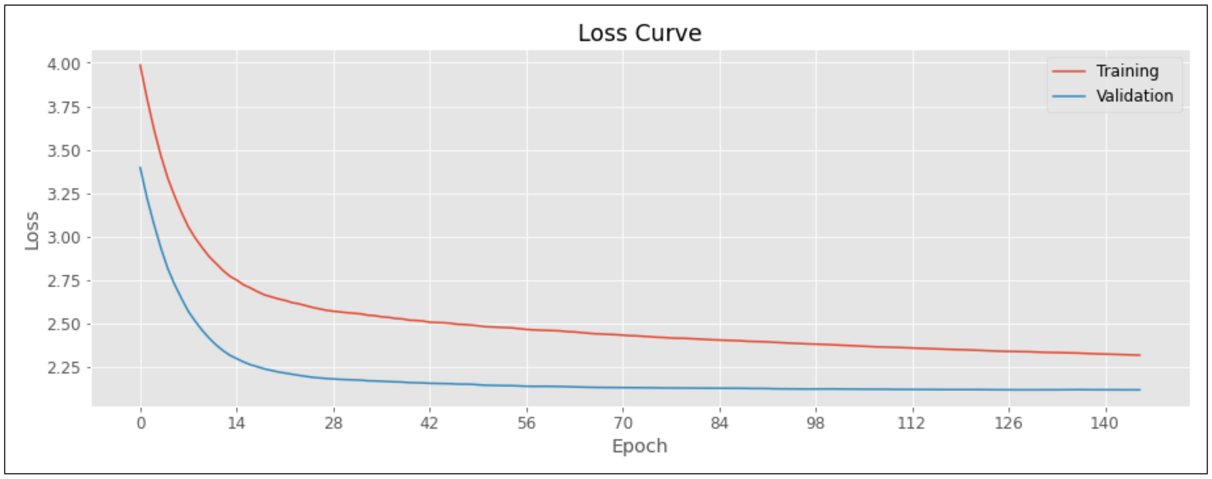


Figure 9: The learning curve of the LGBM model.

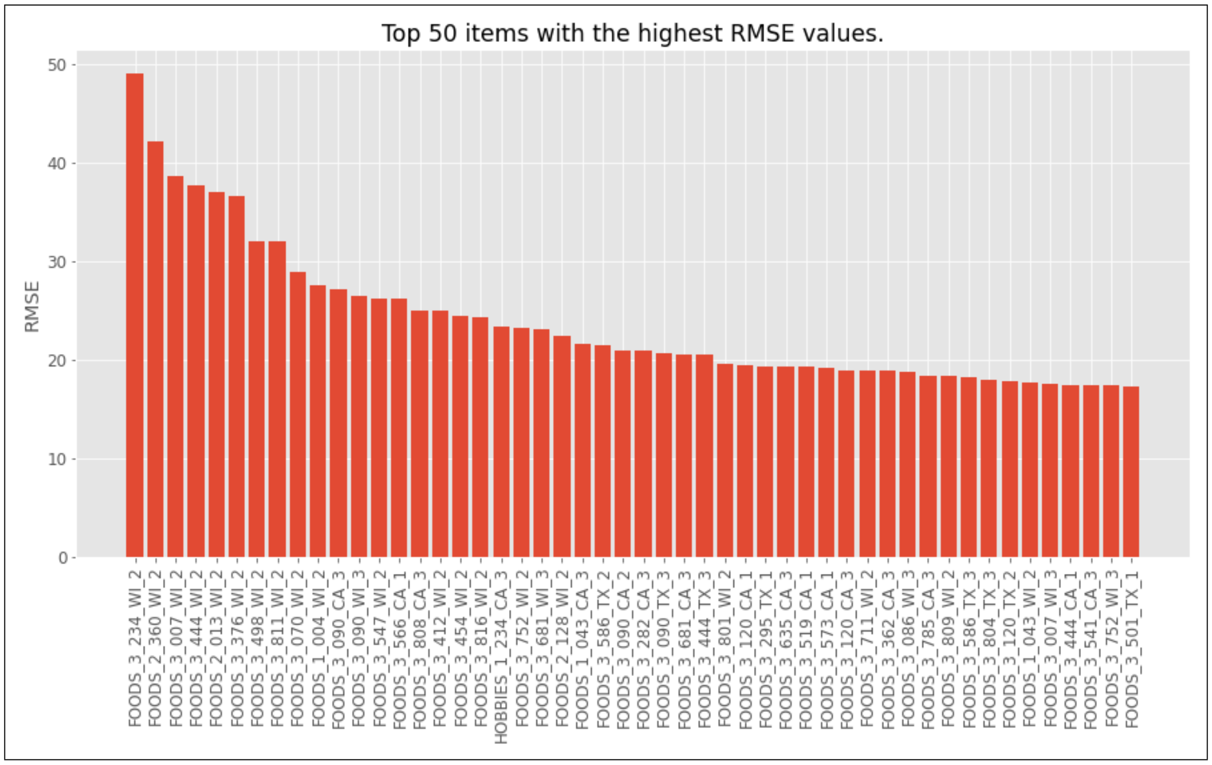


Figure 10: Top 50 items with the highest RMSE for the LSTM model.

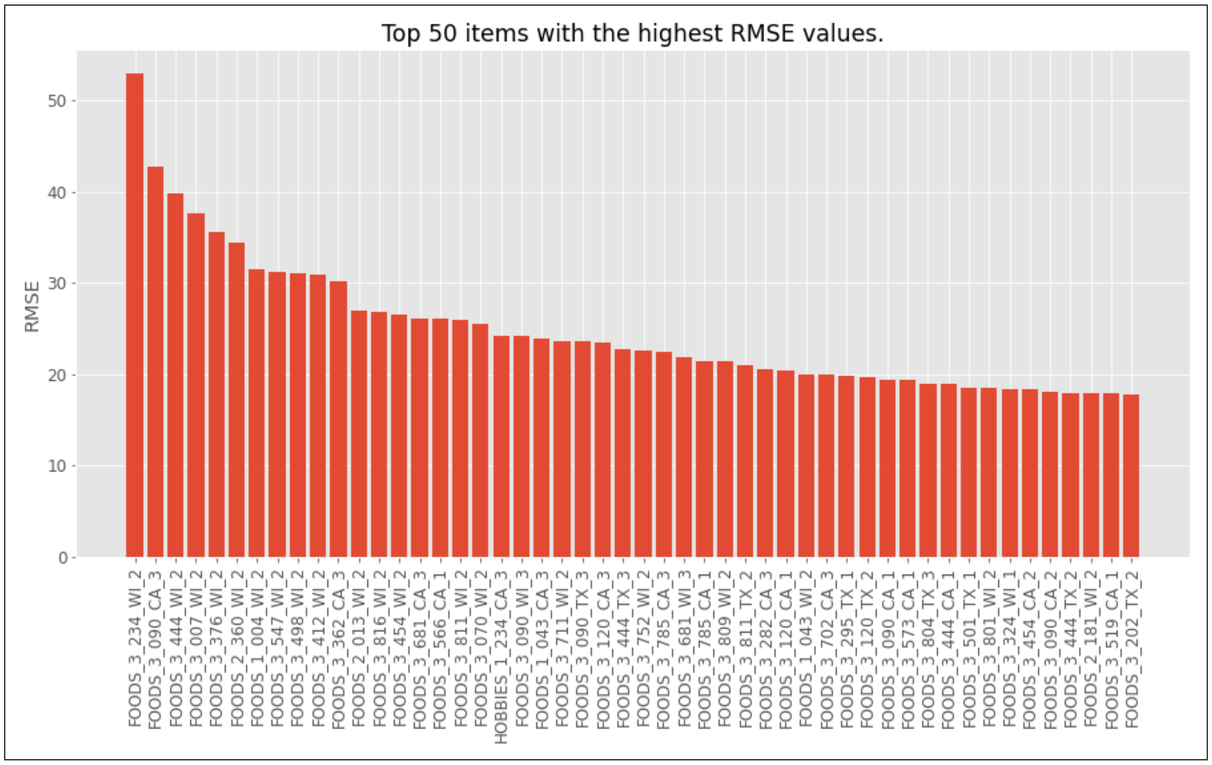


Figure 11: Top 50 items with the highest RMSE for the MLP model.

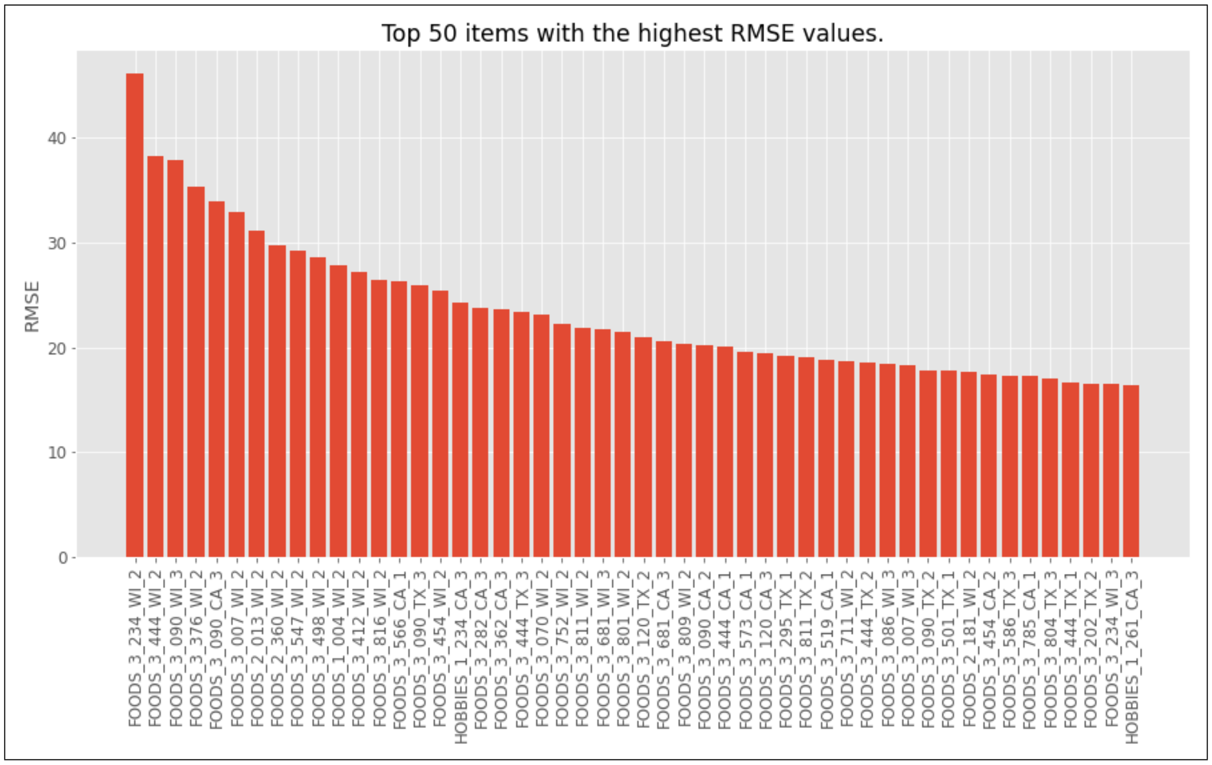


Figure 12: Top 50 items with the highest RMSE for the LGBM model.

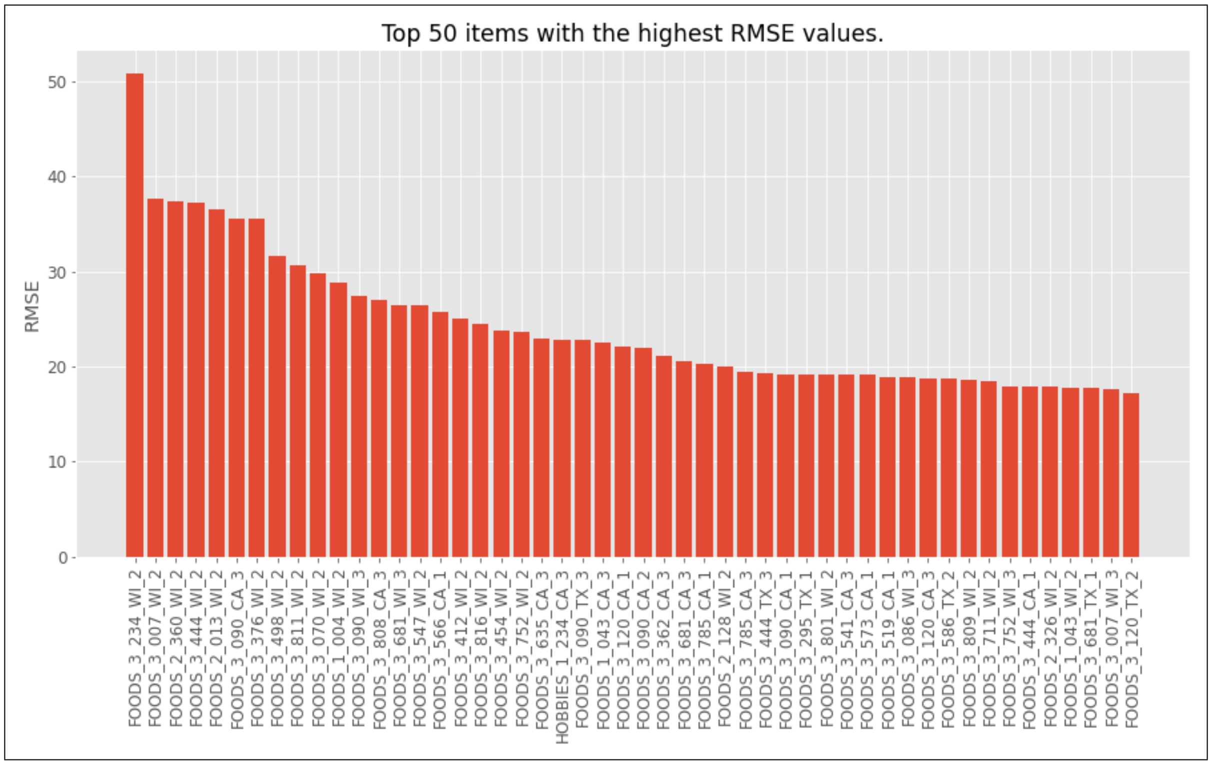


Figure 13: Top 50 items with the highest RMSE for the LSTM-LGBM hybrid model.