NUS Business School

DBA4761A Sem in Analytics: Tidyverse Principles And Tidymodels

Group Project Report

Name	**Matriculation Number**
Chua Jia Zhi	
Francesca Ong Wei Si	
Goh Tian Wei	
Ong Swee Kiat	
Patrick Tan Jing Long	
Toh Han Rong, Benedict	

Table of Contents

1.0 General

In this section, we will explain the 6 folders present in our project folder.

data*:** The folder is further split into ***clean and ***raw*** data. Data in the ***raw*** folder includes the CSV files for the competition from Kaggle and the 'MacroeconomicFeatures.csv' used in *3a. preprocessing.R* and *3b. preprocessing_rf* under ***workflow***. Data in the ***clean*** folder includes the 4 different sets of training data created and 5 different sets of predictions.

figs: The folder includes the charts created in *1. exploration.R **(workflow)*** which are explained in Section 3.0 and the feature importance graphs generated for the LightGBM and Random Forest Models. The naive model does not contain a feature importance because there was no model training involved since it simply uses the last known value for each time series as the prediction. This folder also includes the chart for our workflow.

model: This folder contains our 5 models saved in RDS format. Model_1 to Model_10 are the models for **LightGBM - 10 Dedicated Models**.

report: This folder contains our project report.

***submission*:** This folder includes the final CSV output files.

***workflow*:** This folder contains the R working files that were used to create the model and generate the predictions. The workflow folder is split into:

Name	**Remarks**
0. libs	Libraries used
1. exploration	Exploratory data analysis
2. setup	Setting up of directories, parameters, and required functions Feature engineering steps
3a. preprocessing *3b. preprocessing_rf*	Preparing datasets for model training
4a. model_lgb_altogether *4b. model_rf* *4c. model_lgb_each_store* *4d. model_naive* *4e. model_lgb_simpler*	Model training Making predictions Evaluating RMSE Prediction outputs into submission CSV files

Table 1: Files in Workflow Folder

2.0 Usage

In this section, we go through the process flow our code as well as provide a gauge on how long the model training and output generation will take.

The flowchart depicting the workflow (Fig 1) is appended below:

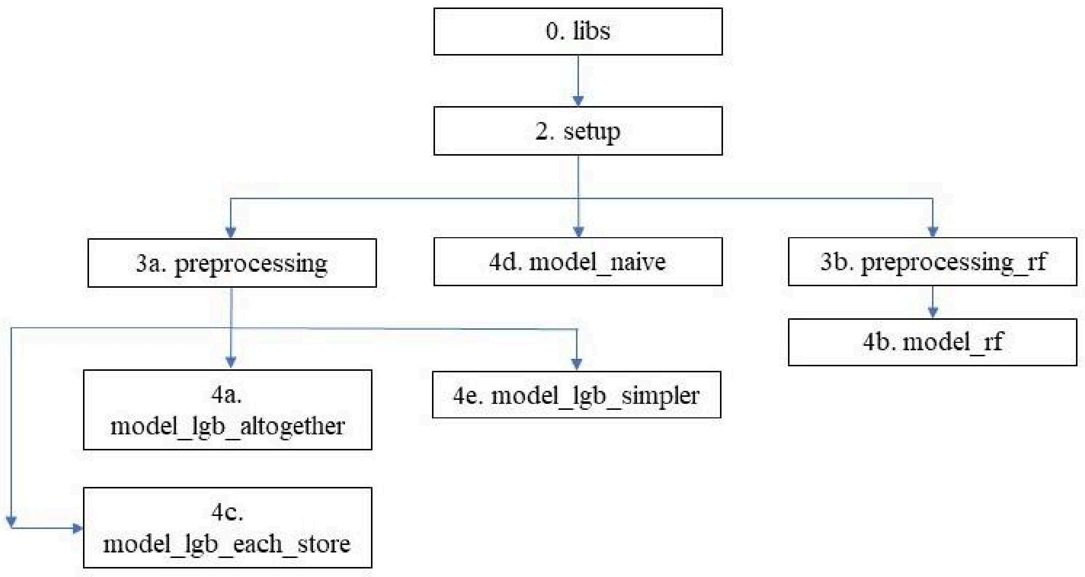


Fig 1: Workflow Process

0. libs and 2. setup must be run before we proceed with the other steps. For each model, follow a one-way path. For example, running *0. libs* → *2. setup* → *3a. preprocessing* → *4e. model_lgb_simpler* is one of the pathways.

However, we have saved the training data generated from *3a. preprocessing* and *3b. preprocessing_rf* in RDS format so simply running *0. libs*, *2. setup* and then running any of the model R scripts will work as intended. For example, running *0. libs* → *2. setup* → *4e. model_lgb_simpler* is a viable pathway since we already have the training data ready in *final_proj/data/clean*.

The process is computationally expensive with the exception of the naive model. The time taken for model training (Table 2) and output generation (Table 3) is as estimated:

Model Training Duration	
Naive	<1 second
LightGBM - Singular Model	3 hours
Random Forest	30 min
LightGBM - 10 Dedicated Models	1.5 hours
LightGBM - Simpler Features	1.5 hours

Table 2: Time Taken to Train Models

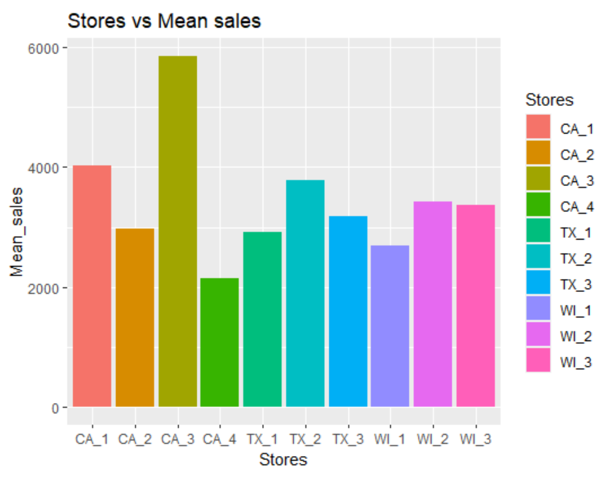
Output Generation Time	
Naive	<1 second
LightGBM - Singular Model	2 hours
Random Forest	3 hours
LightGBM - 10 Dedicated Models	2.5 hours
LightGBM - Simpler Features	2.5 hours

Table 3: Time Taken to Generate Predictions

3.0 Exploratory Data Analysis

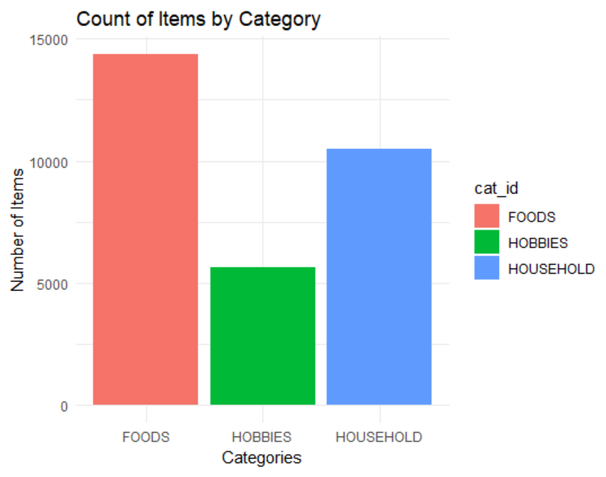
In this section, we aim to understand the structure of the data, such as the types and ranges of variables. We identified patterns, trends, and anomalies, which are key for effective forecasting. This phase would guide the process of feature engineering, and assist in the identification and transformation of relevant features for improved model performance.

3.1 Mean Sales by Store



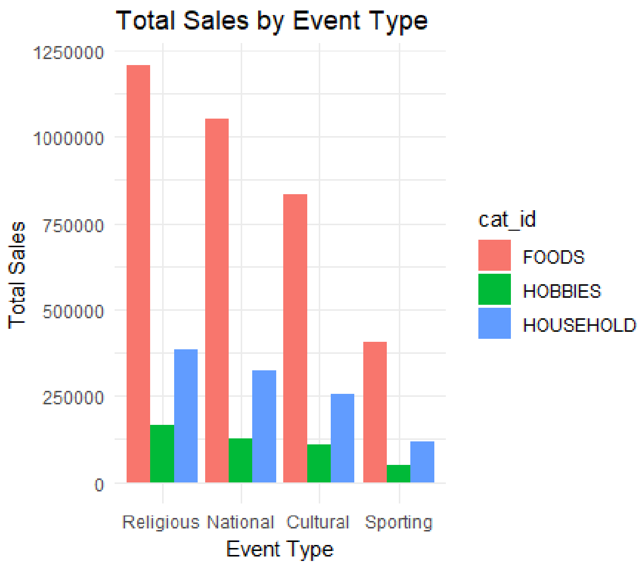
The bar graph above illustrates the mean amount of sales across all stores in 3 states: California (CA), Texas (TX), and Wisconsin (WI). Sales performance varies significantly across the stores. We have observed that the stores with the highest (CA_3) and lowest (CA_4) mean sales are situated in California. One possible assumption is that CA_3 is likely in a populated urban area while CA_4 is in a more rural location. We can infer that there is information varying from store to store that can be captured with feature engineering.

3.2 Count of Items by Category



The bar graph above illustrates the total items sold across all stores sorted into 3 categories: ‘FOODS’, ‘HOBBIES’, and ‘HOUSEHOLD’. Unlike discretionary items, demand for food items is inelastic and consistent, with the highest sales. Despite the regular purchases of household items to maintain our lifestyle, demand for household items is typically lower than food due to their longer use cycle. Lastly, ‘HOBBIES’ items are categorised as non-essential or discretionary, with the lowest sales among them. This offered insights into the difference in sales volume for the 3 item types.

3.3 Sales Volume by Event Type

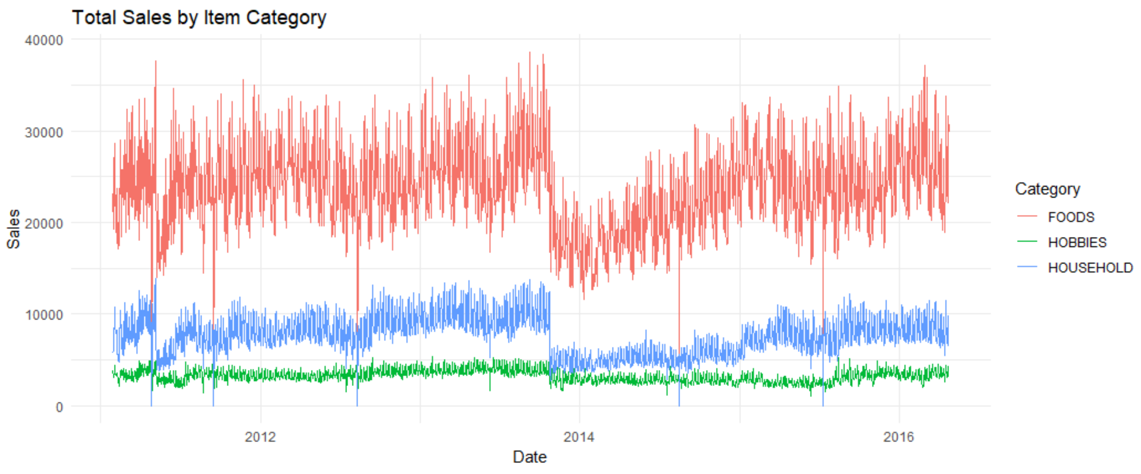


Across all three categories, the highest sales consistently coincide with religious events. This trend reflects the significance of religious events in the United States. Following closely, the second-highest sales are observed during national events. This aligns with a common retail phenomenon where consumers tend to increase spending during national celebrations. The third-ranking event type in terms of sales across categories is cultural events. In the Walmart retail landscape, this indicates that consumers consistently turn to the 'FOODS,' 'HOBBIES,' and 'HOUSEHOLD' sections for their cultural event-related needs. Finally, sporting events consistently represent the event type with the lowest sales across all three categories.

Examining the overall sales performance of categories within Walmart, 'FOODS' emerges as the top-performing category, likely due to the recurrent and essential nature of food-related purchases. 'HOUSEHOLD' follows closely, indicating the stability and continuous demand for household-related products. 'HOBBIES' secures its position as the third-ranking category, reflecting a sustained but varied interest in hobby-related items across diverse event contexts.

The key insight derived from this analysis is that including features that capture 1) the presence of events, and 2) type of event would be useful in making our forecasts.

3.4 Sales Over Time

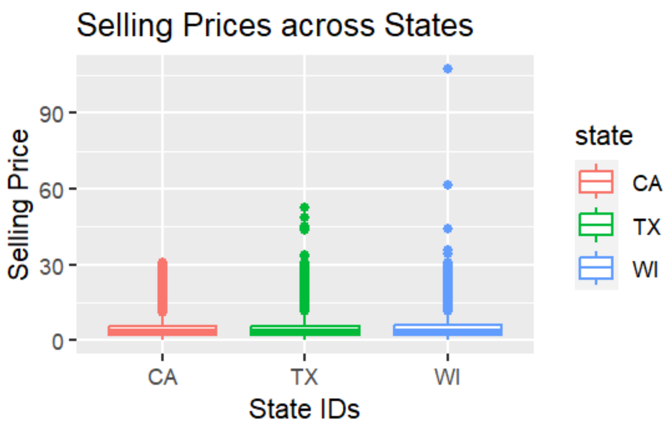


The graph reveals distinct patterns and volumes across the 'FOODS,' 'HOBBIES,' and 'HOUSEHOLD' categories. 'FOODS' category exhibits significant variability with pronounced peaks indicative of seasonal influence or promotional impacts, whereas 'HOBBIES' and 'HOUSEHOLD' show lower and more stable sales trends. In this graph, it is also observed that the total sales hit zero at the end of each year. This is likely due to Walmart being closed on Christmas Day and Thanksgiving Day (Kyle, 2023).

It is also worth noting that there was a sharp drop in sales across all categories in late 2013. Upon further research, we found that in the second half of 2013, the United States faced a significant crisis when the federal government shut down from 1 October to 17 October due to a funding gap triggered by a standoff over the Affordable Care Act (Kille, 2013). This shutdown had adverse effects on the economy, including a reduction in the fourth quarter real GDP growth by 0.2-0.6 percentage points, or $2-$6 billion in lost output. The Office of Management and Budget reported that consumer, business, and investor confidence was negatively affected. This period also saw a significant drop in the consumer confidence index, potentially leading to reduced consumer spending.

From this graph, we derived that introducing rolling statistics and lags would be imperative to smooth out the noise in the data, making the underlying trends more visible. A rolling mean with a window that matches the seasonal cycle would also help in adjusting the data for seasonality. In addition, noting the 2013 crisis in the US led us to postulate that macroeconomic factors steer consumer behaviour. As such, we extracted data on US unemployment rate, inflation rate, and GDP per capita to include in our feature engineering.

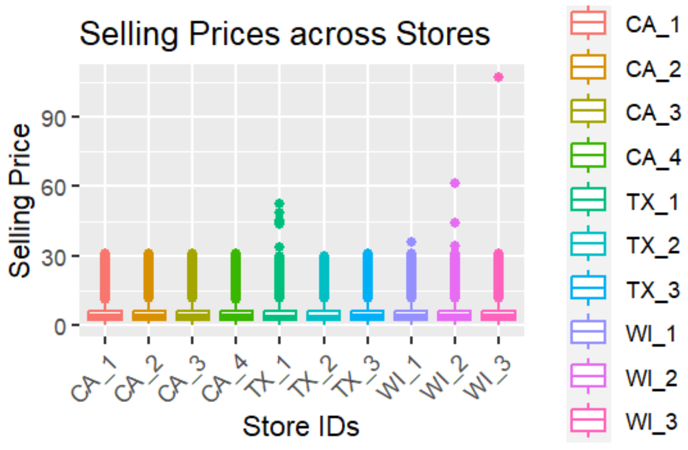
3.5 Selling Prices Across States



The boxplot above illustrates the distribution of selling prices across three states – California (CA), Texas (TX), and Wisconsin (WI).

A point of interest is the outliers, represented by dots beyond the whiskers. All three states have a significant number of outliers above the upper whisker, indicating that there are many exceptionally high selling prices, much higher than the typical range. However, the outliers in state WI show the greatest extent of deviation, followed by state TX, and then state CA. In state CA, all outliers are clustered together within the price range of roughly $10 to $30 with no outliers deviating from the cluster. In contrast, in both states TX and WI, majority of the outliers are clustered together within the price range of roughly $10 to $30 with some outliers deviating from its cluster. The most extreme outlier goes up to a selling price of $107.32, $52.62 and $30.98 in states WI, TX, and CA respectively.

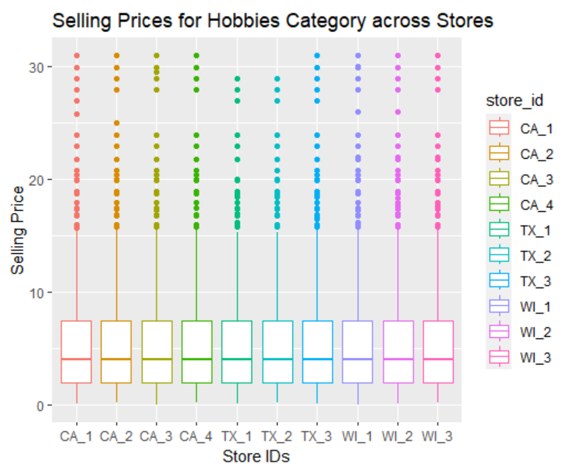
3.6 Selling Prices Across Stores

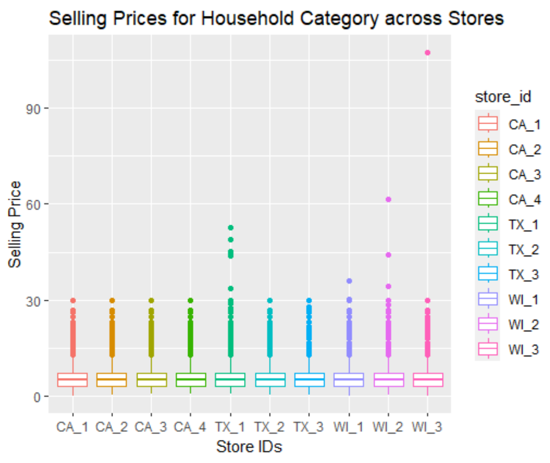


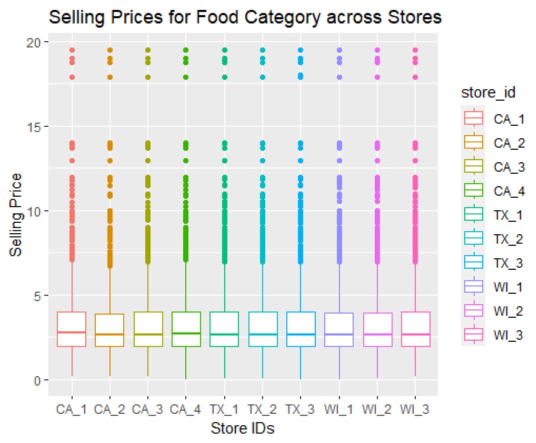
The boxplot above illustrates the distribution of selling prices across all stores in 3 states – California (CA), Texas (TX), and Wisconsin (WI). It delves further into detail, providing a more detailed breakdown of the boxplot shown in Section 3.5.

The boxplot allows us to pinpoint the specific store ids of the outliers that deviated from their cluster in states TX and WI. The outliers that deviated from the cluster in state TX were all from store TX_1. And the outliers that deviated from the cluster in state WI were from WI_2 and WI_3, with the most extreme outlier of $107.32 from WI_3.

3.7 Selling Prices for Different Categories across Stores







The boxplots above illustrate the distribution of selling prices for the three different categories – “HOBBIES”, “HOUSEHOLD” and “FOODS” – across all stores. It delves further into detail, providing a more detailed breakdown of the boxplot shown in Section 3.6.

The boxplots in this section reveal that the outliers that deviated from their cluster in TX_1, WI_2, and WI_3 are from the “HOUSEHOLD” category. This could stem from several factors. One possible factor is demand fluctuations. Local demand can cause price variations. Consumers in stores TX_1, WI_2, and WI_3 may have a greater preference or demand for certain products in the “HOUSEHOLD” category. As a result, retailers in stores TX_1, WI_2, and WI_3 capitalise on this by setting higher prices.

Despite the outliers in the “HOUSEHOLD” category, we can still observe that selling prices remain relatively constant in the three categories. This is advantageous for our prediction models because of several reasons.

Selling price can be considered a stable feature that provides a solid foundation for models, leading to more consistent predictions as they do not introduce much variability into the model’s training process (Master's in Data Science, 2020). They also enhance the interpretability of time series models, such as Prophet, where clear trends and seasonality are crucial. With selling price stability, the model can attribute changes in the time series to seasonal effects rather than price volatility (Brownlee, 2020).

Notwithstanding the above advantages, incorporating a low variance feature such as selling price into our feature engineering could potentially lead to underfitting. This is because such a feature may not contribute enough information for the model to effectively learn and generalize patterns (Das & Cakmak, n.d.). Hence, we decided to use selling price to create the ‘Percentage change in price features comparing current price with 1 day before & 1 year ago’ features that will be discussed further in Section 4.3.

4.0 Data Preparation

In this section, we explain *2. setup*, *3a. preprocessing,* and *3b. preprocessing_rf* in the workflow folder. This R script contains the steps we took to prepare our data for model training.

4.1 Parameters

Firstly, we set up the following parameters in *2. setup*:

Parameter	**Value**	**Remarks**
h	28	Number of days to forecast (Horizon)
max_lags	366	Maximum number of lag days we are considering Max lags for price change a year ago
tr_last	1913	d_1913 is the last day for training
fday	as.IDate ("2016-04-25”)	Date to forecast from. i.e d_1914.

Table 4: Parameters

4.2 Loading Data

Next, we defined a function in *2. setup* to load in the raw data:

Function Name	**Argument 1**	**Argument 2**
create_dt	is_train = TRUE	nrows = Inf

Table 5: Function for Data Preparation

This function takes the argument is_train which would be TRUE if preparing the training dataset, FALSE if preparing the test set.

- If is_train = TRUE, the function loads in calendar.csv, sales_train_validation.csv and sell_price.csv. The sales_train_validation.csv dataset is converted from wide format to long format before merging with calendar.csv and sell_price.csv.
- If is_train = FALSE, the function loads in both calendar.csv and sell_price.csv. The sales_train_validation.csv will then be loaded with data from d_1548 onwards to d_1941 to account for max_lags which takes on a value of 366 days. In addition, columns for d_1942 to d_1969 will be created. Next the sales_train_validation.csv dataset is converted from wide format to long format before merging with calendar.csv and sell_price.csv.

Moving on, we scraped data on US unemployment rate (FRED, n.d.), inflation rate (US Inflation Calculator, n.d.), and gdp per capita (FRED, n.d.) and loaded them into the environment (MacroeconomicFeatures.csv).

4.3 Feature Engineering

For our feature engineering, we defined another set of functions, each for their respective models except for the naive model:

Function Name	**Model**	**Argument 1**
create_fea	Simpler Features	dt
create_fea_singular	LGB Singular	
create_fea_store	LGB Dedicated Models	
create_fea_rf	Random Forest	

Table 6: Functions for Feature Engineering

This function takes the data table generated by create_dt as the argument. The function then processes the features we curated and outputs the data for training.

Based on our data exploration we were able to utilise our insights to conduct our feature engineering. Here are the consolidated features that we have created in all our create_fe functions:

- **Sales lag features** with 1, 7, 14, and 28 day windows
    - Lag features are essential for capturing temporal dynamics. They help our forecasting model to learn how past values influence future values (TWIL, 2020).
- **Rolling sales mean** using values from lag_28 with 7, 30, 90, and 180 day windows
    - This feature aims to capture information using a lagged value from the month before, incorporating a range of window sizes.
    - Rolling averages reduce the effect of outliers when making predictions. It also makes it easier to identify seasonal patterns in a time-series data set (Kumar, 2022).
- **Rolling sales max and variance** with a 28 day window
    - Rolling sales variance helps in quantifying the fluctuations in sales volume. This potentially offers insights into sales stability or volatility for each product and store.
    - Rolling sales max aids in identifying the highest demand levels in the 1-month window. This is crucial for forecasting peak sales periods, especially during special events or promotions.
- **Percentage change in price** features comparing current price with 1 day before & 1 year ago
    - Intuitively, changes in price would affect sales volume - a price hike would potentially lead to a drop in sales vice versa. Hence, we deem this feature as a useful price feature to include.
- Converted **categorical features to integers**

    - Integer encoding is generally less computationally expensive than one-hot encoding.
- Extracted and created columns for the **weekday, day of month, week of the year, month, quarter, and year** from the date
    - The intent of having this feature is our intuition that the subsidivisons of time has an impact on sales volume. For example, some months have more festivities which would likely drive sales volume upwards. Another example would be that when year-end bonuses are paid out (typically in the first quarter of the new year), sales might be driven higher (Indeed, 2023).
- Merged the macroeconomic features (**unemployment rate, inflation rate, and gdp per capita**) to our training data
    - As observed in our time series graph, macroeconomic factors do have an impact on consumer purchases. Hence, we decided to include these features into our dataset.
- Continuous **days with zero sales**
    - This feature counts the number of days a product had zero sales and resets the counter once a positive sale is encountered.
    - The intuition behind this is that it helps to distinguish between low demand and no demand, which is crucial for estimating the likelihood of various sales levels, including zero sales.
    - In addition, the dataset consists of many zeros. Recognising the pattern of continuous zero sales is crucial for effectively managing sparse data.

After running create_fea, we **remove NAs** which would affect our model training with na.omit(). Thereafter, we dedicate the most recent 56 days in the training set (d_1858 to d_1913) to be our cross-validation set and the rest of the data to be our training set for our **LightGBM - Singular Model** and **Random Forest**. We decided that our cross-validation set size should be twice of the forecast horizon (total 56 days) as a larger cross-validation set would allow for a more rigorous assessment of our model's performance during training. Opting for a validation set twice our forecast horizon is viable since we have a large dataset. For the **LightGBM - 10 Dedicated Models** and **LightGBM - Simpler Features** model, we decided that 28 days was sufficient for the validation set.

5.0 Model Approach

In this section, we go through our modelling process in *4a. model_lgb_altogether*, *4b. model_rf, 4c. model_lgb_each_store, 4d. model_naive,* and *4e. model_lgb_simpler*. We included a naive model because it provides a simple baseline against which more complex models can be compared. If a complex model does not perform significantly better than the naive model, it might not be worth the extra complexity. Naive models are also less prone to overfitting and can often be more robust to noise in the data, providing a reality check for more complex models. After the naive model, we decided to built 4 more complex models which would hopefully capture the variability in the data and provide better forecasts. In total we built and compared 5 models:

- Naive
- LightGBM - Singular Model,
- Random Forest,
- LightGBM - 10 Dedicated Models for Each Store, and
- LightGBM - Simpler Features

The models will then be tested against data from d_1914 to d_1941 for evaluation of the **Test RMSE**.

5.1 Naive Model Forecasting

For our Naive Model forecasting, we utilised the previous day's sales data as to predict the next day's sales. This approach assumes that the most recent previous day sales value is the best predictor for the current day sales. The model's performance was validated against sales from d_1885 to d_1913. We obtained a **CV RMSE of 3.32**.

To evaluate the effectiveness of this approach, we calculated the RMSE for this model against the test set. A **Test RMSE of 2.89** was achieved using this basic method without the incorporation of exponential smoothing, suggesting a moderate level of forecast accuracy.

In an attempt to refine our forecast, we integrated exponential smoothing. By setting the smoothing parameter alpha to a high value of 0.9, we placed significant emphasis on the latest sales data. Contrary to expectations, the **Test RMSE** increased to 3.25 when exponential smoothing was applied, indicating a decrease in predictive accuracy compared to the unsmoothed model.

5.2 LightGBM - Singular Model

This model trains a single gradient boosted tree model using the entire dataset. We used data up to d_1858 to train the model and d_1859 to d_1913 for validation. For our loss function, we used the Tweedie distribution (Stephanie, 2020) which is appropriate since our sales data contains many zeros. We considered "poisson" loss function as well since it is computationally more efficient and models well for count data. However, the Tweedie loss function seems to model the data with better performance. The metric parameter is set to "rmse" as our goal is to minimise the root mean squared error. L1 Regularisation with a tuning grid was used in the model as well. However it is worth noting that the effects were minimal where the best lambda = 0.8 only resulted in a reduction in test RMSE by 0.02. We opted for L1 regularisation as L1 regularisation tends to shrink some parameter coefficients to zero, eliminating unimportant features and keeping only the relevant features. This increases interpretability. Training the LightGBM model with tuned hyperparameters, we obtained a **CV RMSE of 1.80** and the following feature importance plot:

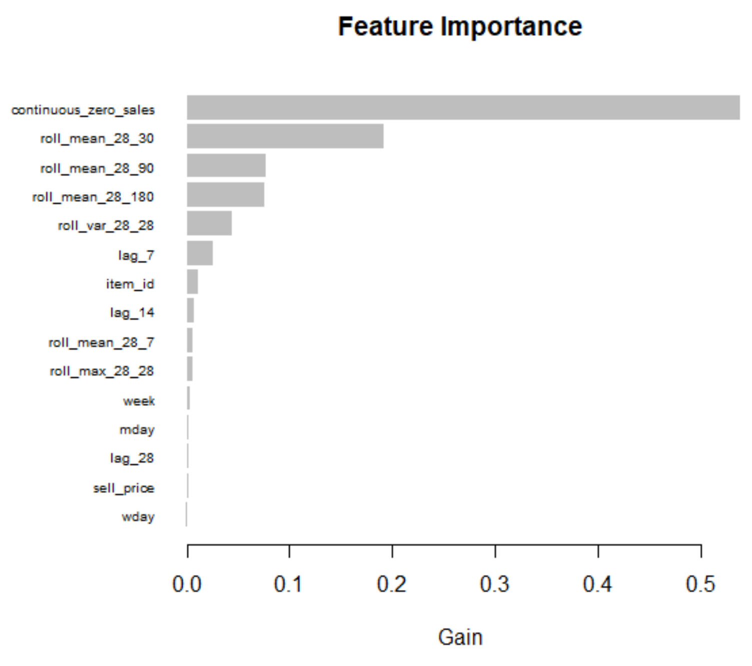


Fig 2: LGB Singular Model Feature Importance

5.3 LightGBM Random Forest

We used LightGBM's Random Forest algorithm as our third model. When the boosting parameter is set to "rf", the algorithm behaves as Random Forest and not as boosted trees in the normal LightGBM model (Bahmani, 2023) in Section 5.2. LightGBM Random Forest was used instead of randomForest as randomForest's (package) limitation in the number of categories allowed required one-hot encoding for particular features. This would have significantly impacted the size of the dataset and time needed to train the model.

Similar to the LightGBM model in the previous section, we used data up to d_1858 to train this LightGBM Random Forest model and d_1859 to d_1913 for validation. Randomness for the dataset was added through the bagging and feature subsampling hyperparameters. It is worth noting that the test RMSE increased by 0.08 when feature_fraction was decreased from 0.7 to 0.5. Given the large number of rows the data had, a value of 600 was used for num_trees. Training the LightGBM Random Forest model with tuned hyperparameters, we obtained a **CV RMSE of 3.34** and the following feature importance plot:

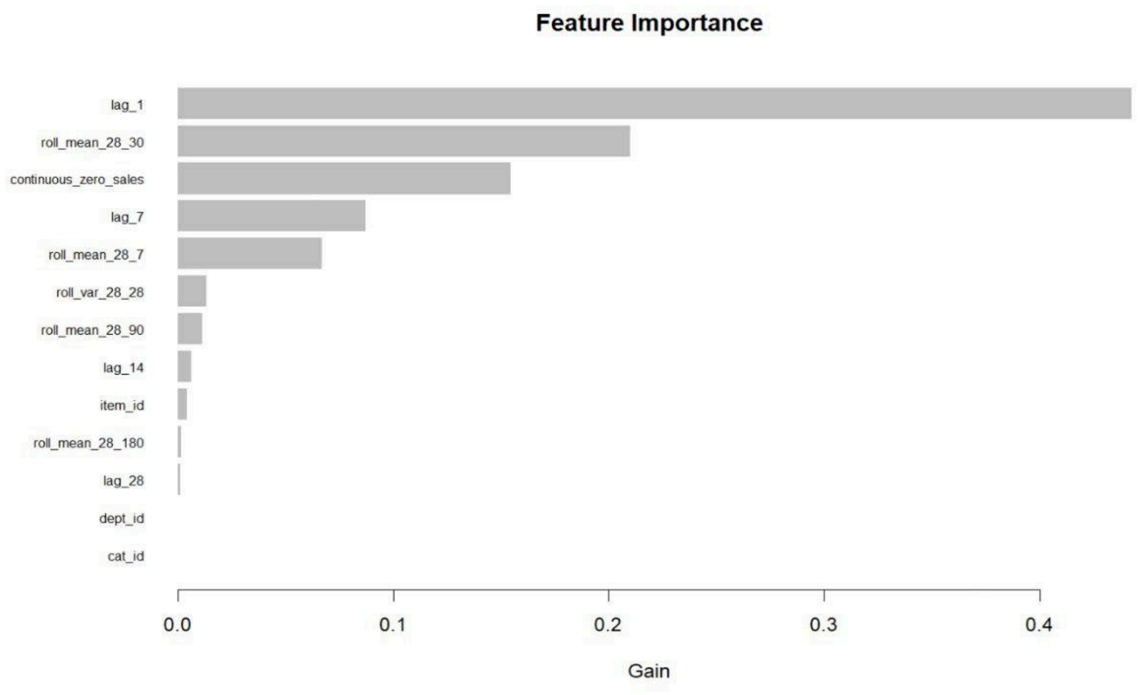


Fig 3: Random Forest Feature Importance

5.4 LightGBM - 10 Dedicated Models for Each Store

This model uses gradient boosted decision trees (gbdt) which is similar to what we did in section 5.2. The main difference is that we train unique models for each of the 10 stores (split by store_id). The intuition behind this approach is that there are unique traits within each of the stores, as seen in exploratory data analysis. For the dedicated store models, we processed another training dataset called *training_store.rds*. We also used data from 2013 onwards to reduce computation time as it would be excessive given the number of models being built. Our hypothesis is that by drilling down the hierarchical data and building a unique model for each store, we would be able to make better forecasts. Using the same set of hyperparameters as in the singular model, we were able to obtain an average **CV RMSE of 1.81** and the following importance plot which contains an averaged importance for each feature across the 10 models:

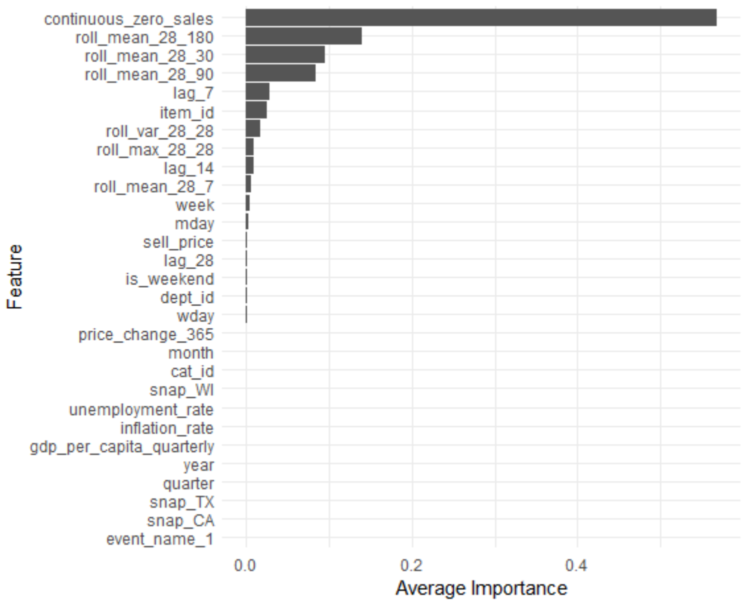


Fig 4: LGB 10 Dedicated Models Feature Importance

5.5 LightGBM - Simpler Features

Our final model is a LightGBM model which omits the following features from hitherto trained models: continuous_zero_sales, unemployment_rate, inflation_rate, gdp_per_capita_quarterly. The intuition is that there exists a possibility that we have introduced some noise into our data with some of the features that we have created. Hence, we decided to build a model with features that are less complex. In this model, we also opted for a “poisson” loss function since the Poisson distribution is suitable for modeling count data, particularly when the counts are small as in our sales data. In addition, it is also less computationally intensive. Training the model, we were able to get a **CV RMSE of 2.06**. The feature importance graph is appended here:

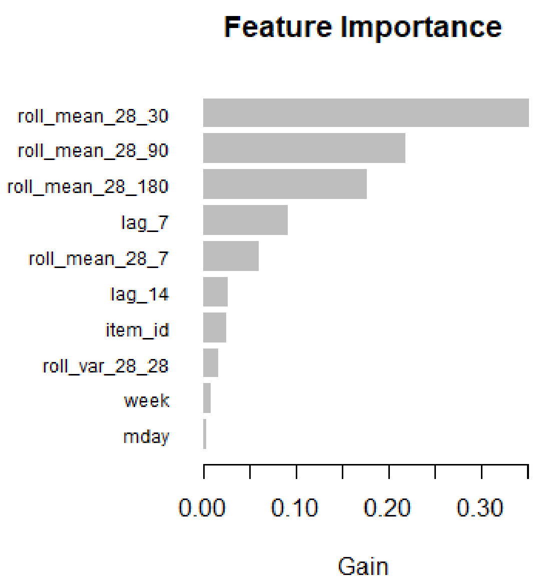


Fig 5: LGB Simpler Features Importance

6.0 Evaluating Model Performance

To evaluate the performance of our models, we made predictions for 28 days from d_1914 to d_1941. These 28 days are the test set with which we measured our predictions against. Generating the outputs and calculating, we obtained the following values:

Model	**CV RMSE**	**Test RMSE**	**Absolute Deviation**
Naive W/O Smoothing	3.32	2.89	0.57
LightGBM - Singular Model	1.80	2.34	0.54
LightGBM Random Forest	3.34	2.44	0.90
LightGBM - 10 Dedicated Models	1.81*	2.31	0.50
LightGBM - Simpler Features	2.06	2.14	0.08

**Average of the 10 models*

Table 7: Comparing Model Performance

From Table 7 above, we see that LightGBM - Simpler Features is the best performing model. The Naive model may be a useful baseline, but it was outperformed by the more sophisticated LightGBM and Random Forest approaches. Indeed, the added complexity proved beneficial to our forecasting performance by capturing nuances better. However, we observed that the LightGBM - Simpler Features model was the sweet spot between simplicity and complexity, which is a good demonstration of the bias-variance tradeoff. It is also well-fitted, evidently, from the small deviation between its CV RMSE and Test RMSE compared to the other models.

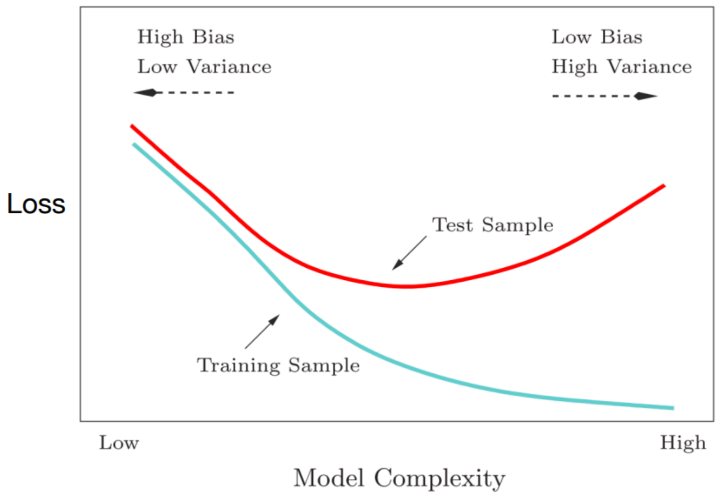


Fig 6. Bias-Variance Tradeoff

7.0 Submission Generation

7.1 Test Predictions

The predictions from each model are saved in *final_proj/data/clean* as .RDS files. The predictions are generated by the individual model .R files. We decided to not utilise any magic multipliers as it would overfit the data to the test set. It would also be unwise to do so in a real-world scenario.

7.2 Output Submission

The final output for submission in .csv format will be generated through the individual model .R files (e.g *4b. model_rf* will generate its own final output) and saved in *final_proj/submission*.

8.0 Conclusion

Overall, the M5 competition project was a novel and enriching task for the team. The scale and structure of the data was unlike other projects involving predictive analytics. It was a daunting challenge as we all had no experience with forecasting let alone a project of this scale but the project provided us with an opportunity to explore a valuable skill. Researching on the competition was also overwhelming due to the vast amount of information available. However, we were fortunate to chance upon the Kaggle notebooks by Konstantin Yakolev (Yakolev, 2020) and Jeon (Jeon, 2020) which contributed to our insights and research. To summarise the key learning points, we learnt that forecasting requires a deep focus on feature engineering. In addition, an appropriate loss function is also vital. For instance, if the target variable contains continuous values instead of integer values (like sales in this project), perhaps a regression loss like Mean Absolute Error would be a viable choice instead of Tweedie or Poisson. Finally, when dealing with massive datasets, it is also highly beneficial to do sanity checks with subsets of the main dataset frequently since the entire process is time-consuming. Ultimately, it was a fruitful learning experience.

9.0 References

Bahmani, M. (2023). *Understanding LightGBM Parameters (and How to Tune Them)*. Neptune.ai. Retrieved November 20, 2023, from https://neptune.ai/blog/lightgbm-parameters-guide

Brownlee, J. (2020, September 16). *Feature Selection for Time Series Forecasting with Python - MachineLearningMastery.com*. Machine Learning Mastery. Retrieved November 20, 2023, from https://machinelearningmastery.com/feature-selection-time-series-forecasting-python/

Das, S., & Cakmak, U. M. (2018). *Excluding features with low variance - Hands-On Automated Machine Learning [Book]*. O'Reilly. Retrieved November 20, 2023, from https://www.oreilly.com/library/view/hands-on-automated-machine/9781788629898/00dc736a-3a50-41d5-995b-b83004242975.xhtml

FRED. (n.d.). *Real gross domestic product per capita (A939RX0Q048SBEA) | FRED | St. Louis Fed*. FRED Economic Data. Retrieved November 22, 2023, from https://fred.stlouisfed.org/series/A939RX0Q048SBEA

FRED. (n.d.). *Unemployment Rate (UNRATE) | FRED | St. Louis Fed*. FRED Economic Data. Retrieved November 22, 2023, from https://fred.stlouisfed.org/series/UNRATE

Glen, S. (2020, July 9). *Tweedie Distribution: Definition and Examples*. Statistics How To. Retrieved November 18, 2023, from https://www.statisticshowto.com/tweedie-distribution/

Gontariu, D. (2020, June 5). *Dig deep enough. Create meaningful Features for ML… | by Ali TWIL*. Towards Data Science. Retrieved November 18, 2023, from https://towardsdatascience.com/dig-deep-enough-features-engineering-techniques-for-time-series-analysis-in-python-500c96aebade

Indeed. (2023, February 16). *When Are Bonuses Paid Out?* Indeed. Retrieved November 18, 2023, from https://www.indeed.com/career-advice/pay-salary/when-are-bonuses-paid-out

Jeon. (2020, July 1). *3rd place solution - NN approach*. Kaggle. Retrieved November 21, 2023, from https://www.kaggle.com/competitions/m5-forecasting-accuracy/discussion/164374

Kille, L. W. (2013, November 7). *Economic effects of the 2013 U.S. federal government shutdown*. The Journalist's Resource. Retrieved November 18, 2023, from https://journalistsresource.org/economics/economic-effects-2013-us-federal-shutdown/

Kumar, A. (2022, December 4). *Moving Average Method for Time-series forecasting*. Analytics Yogi. Retrieved November 18, 2023, from https://vitalflux.com/moving-average-method-for-time-series-forecasting/

Master's in Data Science. (2020). *What Is the Difference Between Bias and Variance?* Master's in Data Science. Retrieved November 20, 2023, from https://www.mastersindatascience.org/learning/difference-between-bias-and-variance/

US Inflation Calculator. (n.d.). *Current US Inflation Rates: 2000-2023*. Inflation Calculator. Retrieved November 22, 2023, from https://www.usinflationcalculator.com/inflation/current-inflation-rates/

Yakolev, K. (2020). *M5 - Three shades of Dark: Darker magic*. Kaggle. https://www.kaggle.com/code/kyakovlev/m5-three-shades-of-dark-darker-magic