1. **DATA UNDERSTANDING**

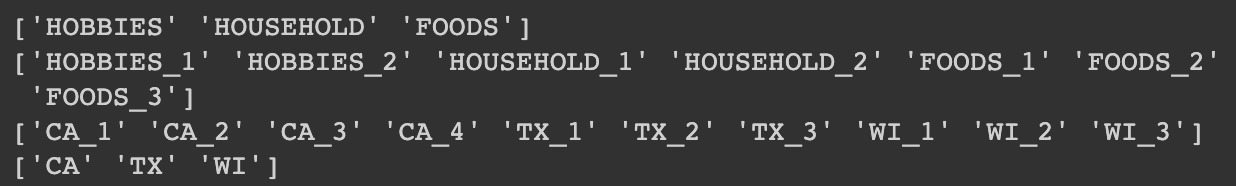
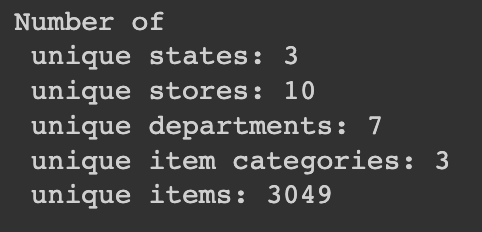
The data was obtained from the [Kaggle website](https://www.kaggle.com/competitions/m5-forecasting-accuracy/data). We used three datasets from the five that were made available:

* sales\_train\_validation.csv - containing the data about sales of individual items in selected shops of the Walmart supermarket chain
* sell\_prices.csv - with data about the sales prices of individual goods
* calendar.csv - with the dates on which the products are sold

Sales data

The original data frame *sales\_train\_validation* comprises 30490 rows and 1919 columns.

The first group of columns contains data about the item id, product category id, id of the department which it belongs to, and also the store and state in which the item was bought. One item is offered in multiple stores, therefore we can see that there is always one id for one particular item in one particular store. Hence it makes sense that there are 3049 unique items and 10 unique stores, which after multiplying goes up to 30490 rows.

We also see that the items belong to 3 possible categories (‘Hobbies’, ’Household’ or ‘Foods’) and come from 7 various departments, which are derived from the categories (see code results below). We can also see that the data comes from 3 states - California, Texas and Wisconsin, whereby there are 3 stores in each Texas and Wisconsin and 4 stores in California.

The other group of columns, which is much more abundant, contains day numbers from *d\_1* to *d\_1913*. These columns help us understand how many units of the given product were sold on a particular day.

After inspecting, we find out that all 1913 ‘day columns’ contain integer values, whereas the first descriptive columns mentioned above are all object values.

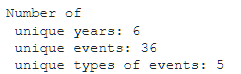
We also checked for the existence of null values in this dataframe, but we did not find any column- or row-wise. The data seems to be complete from this point of view. Just to make sure, we also checked for duplicated rows. Again, no such rows were found.

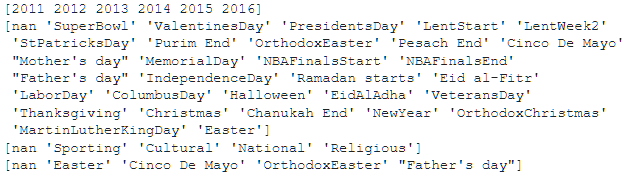
Calendar

The original data frame *calendar* comprises 1969 rows and 14 columns.

The columns contain data about:

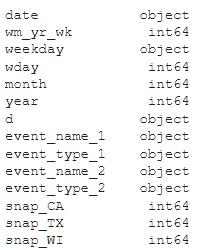
* Date (*date*),
* ID of the week that day belongs to (*wm\_yr\_wk*),
* Name and order of weekdays (*weekday, wday*),
* Month and year of observation (*month,year*),
* Ordinal number of days (*d*),
* Name and type of event, occurring that day - two times for two different events happening at the same time (*event\_name\_, event\_type\_1, event\_name\_2, event\_type\_2*),
* Variables indicating, if at that particular day SNAP purchases were allowed in Walmarts in the state of California, Texas or Wisconsin respectively(*snap\_CA, snap\_TX, snap\_WI).*



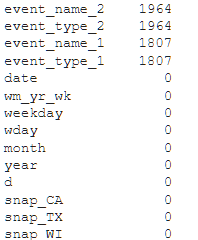
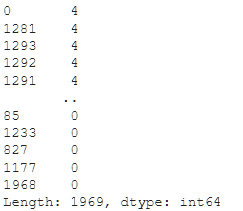
As we can see, those observations have been taken during six unique years (2011-2016) with 35 unique events, classified into 4 groups. Variable *event\_name\_2* does not contain any unique variables, those are not present in *event\_name\_1.*



Types of columns are divided into two groups - object and integer. Both groups contain 7 variables each.



Null values are present in the table due to the absence of specified events (via variable *event\_name\_1/2* and *event\_type\_1/2*). As we can see, *event\_name/type\_2* is present only 5 times, 1964 of rows missing any value. For *event\_name/type\_1* - 162 rows with values and 1807 without. Column-wise only those 4 variables are missing values in some of the rows.



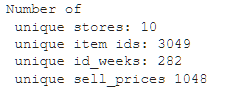
No duplicate rows were found, we can say that the dataset is complete with understanding of null values within event-related variables.

Sell prices

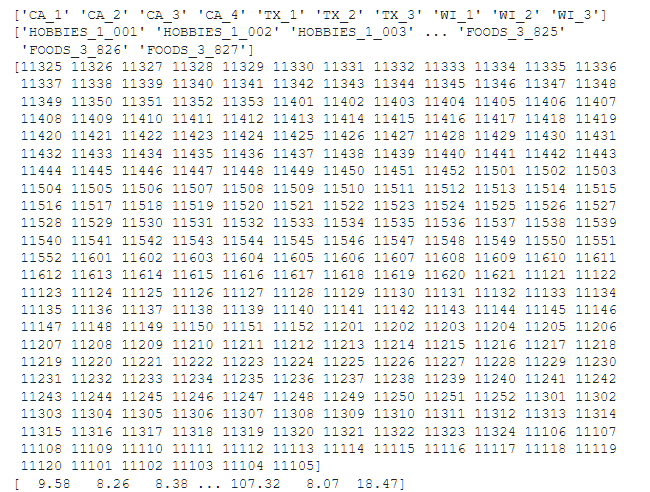
The original data frame *sell\_prices* comprises 6841121 rows and 4 columns.

Columns contain data about:

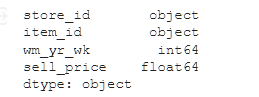
* Store\_id (*store\_id*)
* Item\_id (*item\_id*)
* ID of the week (*wm\_yr\_wk*)
* Sell price (*sell\_price*)



We can see that the number of unique stores and item ids is identical to dataset *sales\_train\_validation,* also we have 282 unique weeks and 1048 unique prices - some of the items are sold for the same price.



Data types of variables are object for *store\_id* and *item\_id*, *wm\_yr\_wk* is integer and *sell\_price* is float.



Null values are not present in data frame column-wise and row-wise. Duplicate rows were not found either. Data set seems complete.

1. **DATA PREPROCESSING**

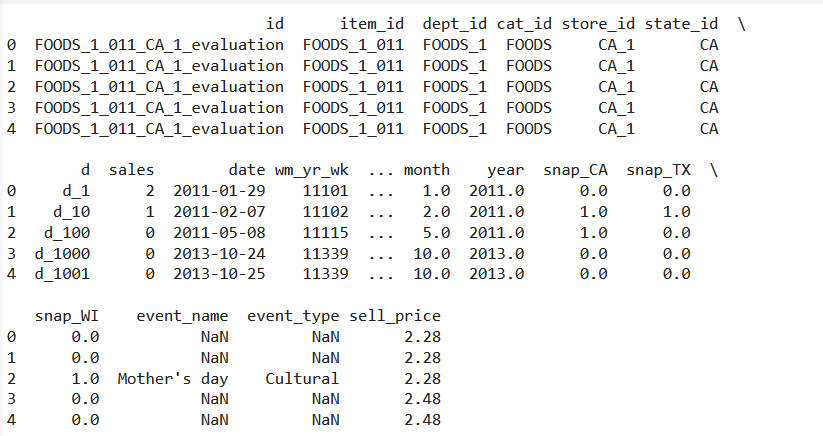
Following the first part of data understanding we proceeded to data preprocessing. We have uploaded the item\_id\_sample dataset and printed what’s inside. This dataset was basically a list of items which we should be considering in the whole rest of the process. This allowed us to get rid of a decent portion of the initial sales\_validation data by removing the undesired items.

Firstly, we checked whether there are some id’s in the sales\_validation which are not in the items dataset and found out there are a lot of them. So with the merge function we **filtered out** all of the rows with ids which are not in the items dataset.

Then we checked the calendar dataset and realized there is a column “d” in a **long-form** while our so far-merged data frame contains the same column but in **wide-form**. We’ve melted columns d\_1 to d\_1941 to long-form and merged the calendar with our dataframe.

Then we dealt with columns event\_name\_1, event\_name\_2, event\_type\_1, event\_type\_2 which are, for some reason, splitted into two parts. Because of the data saving we’ve decided to merge these columns together so instead of 4 columns we have only 2: **event\_name** and **event\_type**.

Then based on store id, item id and wm\_yr\_wk variables we’ve merged into our dataframe last dataset sell prices. The view of the dataset so far is:



After further examination we have decided to create two more columns: **weekday\_binary** and **event\_binary**. Weekday binary is a binary column derived from column “weekday” where 0 = weekday, 1 = weekend. Event binary is derived from event\_name where 0 = no event this day, 1 = some event on this day. We’ve created this because it allows us to work efficiently with the data during machine learning transformations and also due to parsimony for our analysis is enough to have binary variables marking potentially significant events (weekend, holiday) than have two variables marking a lot of different day names and events which could confuse the model and thus worsen the results.

Then we’ve changed the column “d” the datatype from string to integer by removing the “d\_” in each row. It allows us to work with the number of days in numeric form more efficiently for the algorithm.

We’ve also created a column “**profit**” (sales\*sell\_price) because it is a basic economic metric which we assume could be significant for our analysis.

1. **Modeling**

While preparing data for modeling, we determined that utilizing aggregate data for departments, stores, and dates would streamline our process. Subsequently, we incorporated aggregate variables related to profit, such as profit\_month\_avg, profit\_month\_min, profit\_month\_max, profit\_week\_avg, profit\_week\_min, profit\_week\_max, profit\_d-1, profit\_d-2, and profit\_d-3.

For the categorical variables, we generated dummy variables through one-hot encoding. Additionally, we partitioned the data into training + validation and testing sets.

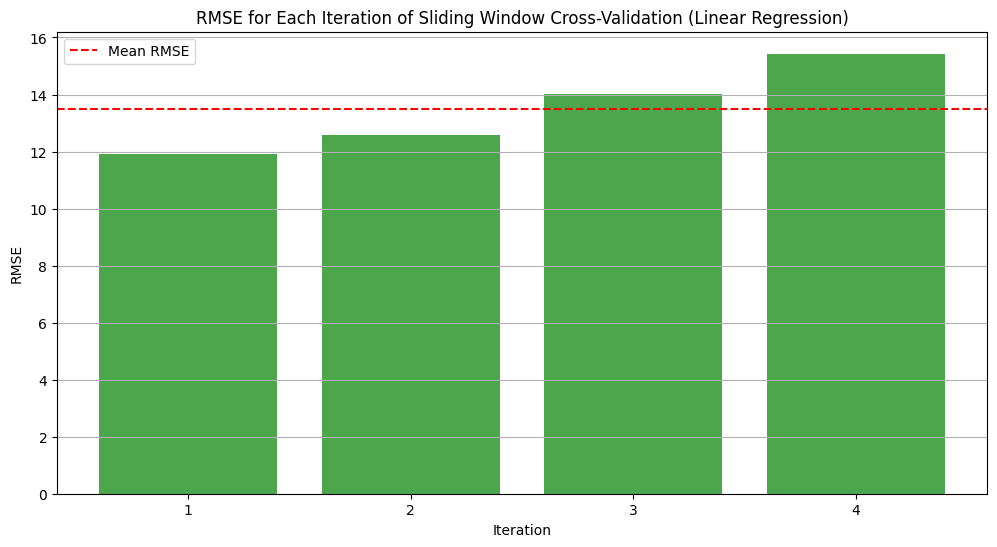
In all our modeling endeavors, we implemented sliding window cross-validation. We partitioned the training + validation data into five chronological frames.

As our baseline model, we selected simple linear regression, featuring only one feature, profit\_d-1, which contains information about the previous day's profit. This model yielded an RMSE of 18.24.

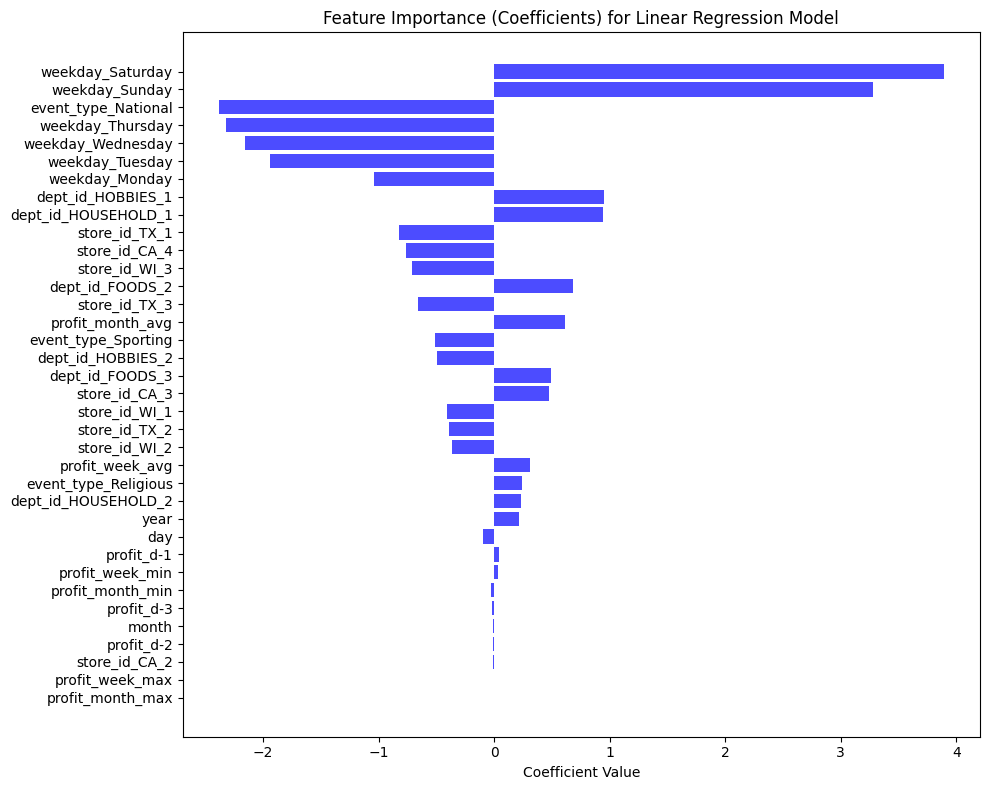
We also explored other models, including Linear Regression (RMSE - 13.48),

Random Forest (RMSE - 13.49), and XGBoost (RMSE - 13.35). Notably, the XGBoost model outperformed the others, although it demanded considerably more computational time compared to Linear Regression, which exhibited a competitive RMSE.

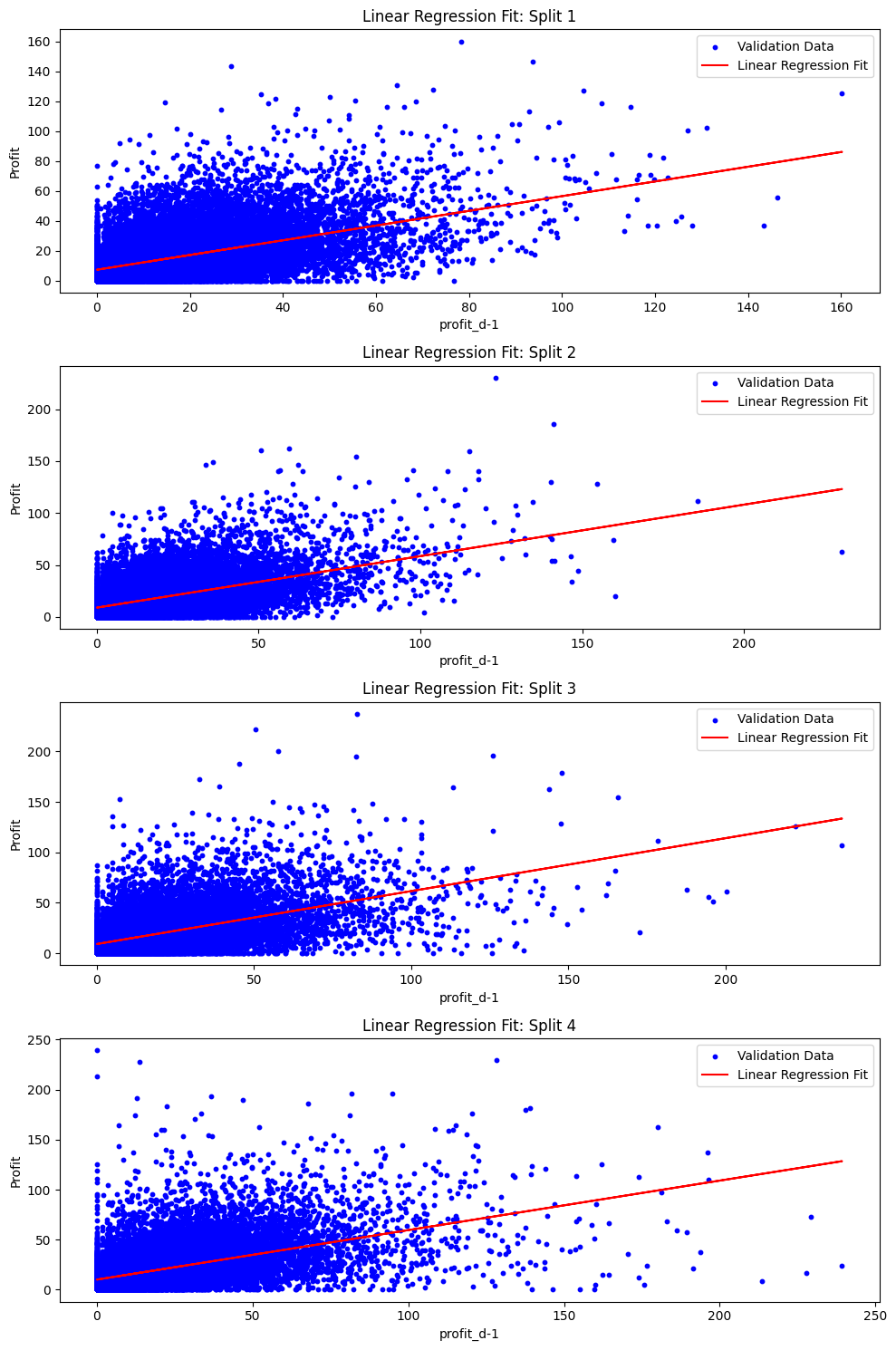
Consequently, we determined that Linear Regression was the most suitable choice as the best model, primarily due to its ease of interpretation. Now, we will look at graphs for linear regression more closely.



In the bar graph, we can observe the RMSE (Root Mean Squared Error) values for each iteration of a sliding window cross-validation using a Linear Regression model, with a red dashed line indicating the mean RMSE across all iterations. It is very important to know the importance of each feature, which is shown in the next graph.



The importance of coefficients for our linear regression model is shown above, sorted by coefficient value descending. The final graph displays the linear regression fits for the first four iterations of a sliding window cross-validation. Each subplot illustrates the relationship between the "profit\_d-1" feature and the "Profit" target variable, showing how the linear regression model captures the patterns in the data for each iteration.



In the final phase, we applied the best Linear Regression model to make predictions on the test data, resulting in a final RMSE of 17.70.