Introduction to project:

The theory of forecasting underpins that current and past knowledge can be leveraged to estimate future time points. Furthermore, identifying patterns in the past data and utilising those patterns in the process of prediction can be fruitful. Predicting future values at exact accuracy is not expected rather options such as an expected value, a prediction interval, a percentile, and an entire prediction distribution which can be collectively said to be called a ‘forecast.’ A forecasting method is defined as a certain sequence of steps taken which leads to production of future time points. Most if not all forecasting methods have **stochastic models** the produce similar point forecasts. A stochastic model aids by providing a data generation process which in addition to point forecasts could create prediction intervals and entire prediction distributions. Stochastic models make certain assumptions with respect to process and probability distribution and even if the forecasting method has stochastic model, the model is not considered unique. Forecasts combined from different methods is an effective forecasting method and the combination of these stochastic models is also considered a model (Petropoulos et al., 2022).

Understanding of variables and their role in the forecasting process is considered very vital. Univariate forecasting i.e. single time-series forecasting, forecasts are created by utilizing historical observations of the same time series whereas in multi-variate forecasting, different variables are used to produce forecasts such as in time series regression. Most importantly, data comes in a wide range of forms and these forms determine the appropriate forecasting methods to be considered. In certain cases, there is no historical data present and this case judgmental methods are applied. In certain cases, the form of data observed needs the development of an entirely new forecasting method. There can be differences in the frequency of our data where data is collected every minute, hourly, weekly, monthly, and yearly. In case of electricity industry, demand forecasts are required at hourly intervals as well as looking a decade forward. The data can consist of a single time series as well as millions of them and comes in a wide range of forms. In Economic analysis there are variables which affect one another and in Business, time series can have a hierarchical structure such as stock keeping unit or common ingredients. Furthermore, in certain cases most of the values might be zero called as intermittent time series (Petropoulos et al., 2022).

Finally, it is vital to evaluate the effectivity of our model and measure its accuracy. The main aspects used to evaluate the accuracy is to check the difference between the actual value and the point forecast of the value. Furthermore, loss functions are proposed which calculate the average of these differences and prediction intervals and percentiles are used to evaluate the point forecast and are considered as part of the forecast process. An additional but important tool is to evaluate the forecasts by metrics such as total costs or service levels (Petropoulos et al., 2022).

The retail industry is considered one of the key drivers of global economy with millions of enterprises and retailers offering goods and services to billions of customers. It captures a wide range of industry such as food, motor-vehicles, apparel, and electronics where transactions occur through distributions channels such as in-store and e-commerce.

In United States, Walmart, Amazon, and Costco are the top three retail companies in the world where the total retail market has reached a revenue of more than seven trillion U.S. dollars. China, being a dominant market in consumer goods and retail industry reached a revenue of more than two trillion U.S. dollars, followed by India with a revenue of 1.4 trillion U.S. dollars.

In Europe, The United Kingdom and Germany are the leaders in the retail market with the former generating sales close to 510 billion pounds and later worth 650 billion euros (Statistica, 2024).

Large retail organizations like Walmart, Costco, Amazon, and Target work on a business model where they enable selling of their own products as well as those of their competitors (Hassan et al., 2022).

In Retail, SKU (Stock keeping unit such as a shampoo of size X) is considered as the smallest unit important in operations such as daily stock replenishment and distribution. The number of these SKU’s can be in thousands depending on the retail chain. Nowadays, a typical supermarket or drugstore has tens of thousands of SKU items. Walmart, being the biggest retailer in the world, deals with more than one billion SKU and store combinations. Fashion chain such as Zara maintains SKU items in tens of thousands (Fildes, Ma, & Kolassa, 2022).

Considering the sheer number of unique items, it becomes immensely important for the retailer to predict the demand of their unique products. Successfully predicting the demand results in better inventory management, better distribution, thus minimizing loss and improving sales and customer satisfaction (Jain, Menon, & Chandra, 2015), (Gordon & Berry, 2004).

Furthermore, there are many external events that can affect demand. Some of them are competition, weather, seasonal trends. In addition to external factors, internal events such as promotions, sales events, and pricing play a significant role in the changing demand (Jain, Menon, & Chandra, 2015).

Supermarkets to prevent customer service issues and high inventory cost rely heavily on forecasts to aid in making strategic and tactical decisions as well as demand and supply planning (Fildes, Ma, & Kolassa, 2022).

The above factors make forecasting the demand of retail sales an important tool to add value to the business (Hassan et al., 2022). Furthermore, forecasting for the sake of forecasting has no value. Optimistic forecasts can lead to issues of overstocking leading to extra costs whereas pessimistic forecasts can lead to lost sales because of insufficient stocking. Thus, it is pivotal that our forecast has the highest accuracies.

Demand can be classified into four different types: intermittent, lumpy, smooth, and erratic (Syntetos, Boylan, & Croston, 2005), (Tian, Wang, & E, 2021). Intermittent demand occurs when there is a high proportion of zero values which is commonly observed in the retail industry.

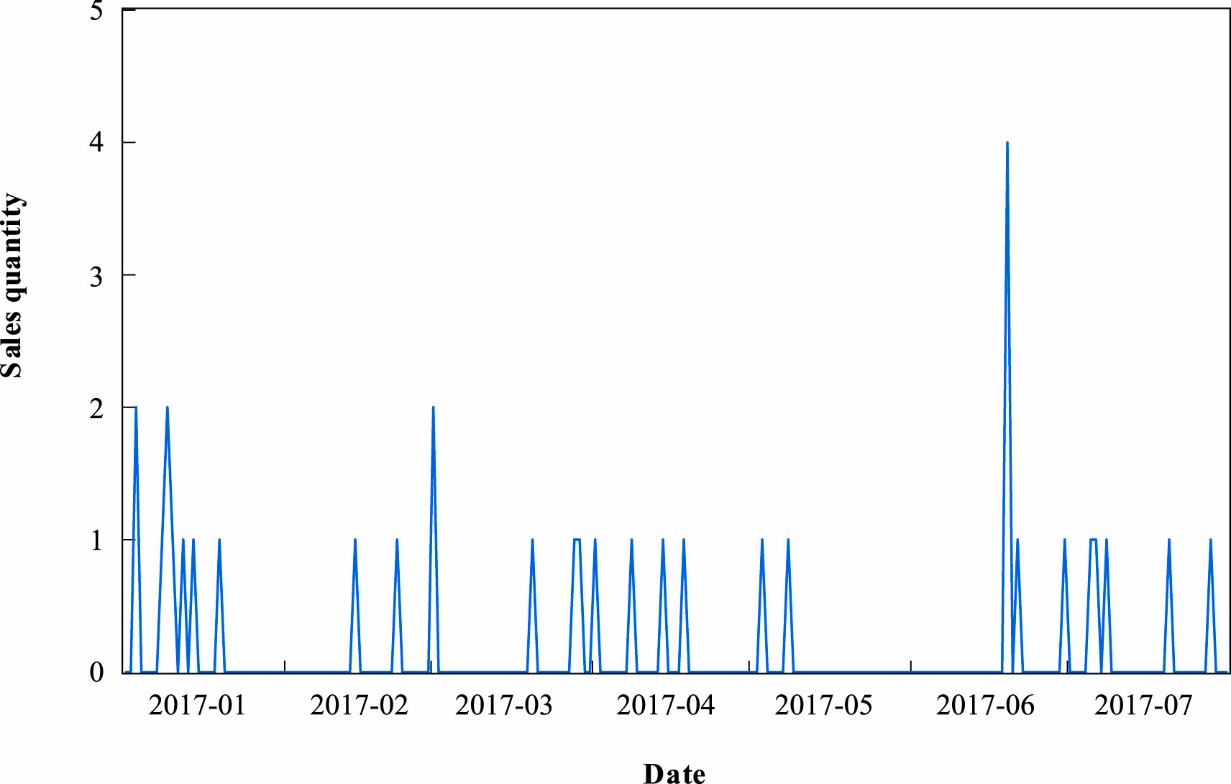


Figure 1: Illustrating the time series of an item (Fig. 1 from Tian, Wang, & E (2021)

The above figure is an example an item displaying intermittent demand from the Tmall.com retail platform. For 87% of the time, the item has no demand which can lead to inconsistencies in the stock levels and delivery center.

Forecasting of intermittent demand is considered as a challenging venture where two important difficulties need to acknowledged: 1) actual demand is irregular or random and 2) difficult to predict the occurrence or timing of demand (Nikolopoulos, 2021). Thus, it becomes vital to predict the quantity and time of demand occurrence.

Past decades of research have contributed to many models for demand prediction and forecasting that focus on fast-moving time-series but limited research has been done to forecast intermittent demand. Another reason for the lack of research is the difficulty as forecasting intermittent data is considered challenging compared to the traditional forecasting problem (Nikolopoulos, 2021).

Although contribution to intermittent demand forecasting has been limited, various researchers have developed solutions. The CR method is the first original work conducted on intermittent demand forecasting which has been practically successful (Croston, 1972). Syntetos and Boylan (2001) found bias in the CR method and proposed SBA method which deals with the bias and is considered an improvement to the CR method. Prestwich et al. (2014) proposed an unbiased model which is the combination of CR method and Bayesian inference. The problem of outdating items in intermittent demand forecasting has also attracted attention of many scholars (Babai et al., 2019), (Prestwich et al., 2014), (Teunter et al., 2011).

Furthermore, machine learning techniques such as neural networks have been widely used for intermittent demand forecasting. Lolli et al. (2017) utilized feedforward single-hidden layer neural network for different aggregation levels and Kourentzes (2013) proved neural networks to be effective for forecasting intermittent demand.

The M competitions have been organized since the last four decades with the objective of improving forecast accuracy by studying and evaluating state of the art forecasting methods ([Makridakis et al., 1982](https://www.sciencedirect.com/science/article/pii/S0169207021001874" \l "b27), [Makridakis et al., 1993](https://www.sciencedirect.com/science/article/pii/S0169207021001874" \l "b30), [Makridakis and Hibon, 2000](https://www.sciencedirect.com/science/article/pii/S0169207021001874" \l "b31), [Makridakis et al., 2020c](https://www.sciencedirect.com/science/article/pii/S0169207021001874" \l "b34).). The results from these competitions have had a profound impact on the practice of forecasting by providing valuable knowledge in improving forecast accuracy  ([Hyndman, 2020](https://www.sciencedirect.com/science/article/pii/S0169207021001874" \l "b15)). The first four competitions contributed in understanding the potential of combining, automatic forecasting, merits of simplicity and machine learning methods (Makridakis, Spiliotis, & Assimakopoulos, 2022).

The M5 competition was an extension of the previous ones which focused on retail sales forecasting by utilizing real world sales data that was hierarchically structured with intermittent and unpredictable behavior ([Syntetos and Boylan, 2005](https://www.sciencedirect.com/science/article/pii/S0169207021001874" \l "b54), [Syntetos et al., 2005](https://www.sciencedirect.com/science/article/pii/S0169207021001874" \l "b55)). Key findings obtained from this competition was the superior performance of the machine learning methods, value of combining, cross learning, and explanatory variables More interestingly, among all the previous competitions, M5 competition was the first where all the top performing methods were “pure” ML methods (Makridakis, Spiliotis, & Assimakopoulos, 2022).

LightGBM, a decision tree-based ML technique provided superior forecasting performance and was used by all the top 50 competitors. Additionally, deep learning techniques like DeepAR and N-BEATS have also shown potential.

Dataset

The data utilized in this research is hierarchical sales data provided by Walmart for the M5 competition. The level of hierarchy starts at the item-level and aggregating further to departments, product categories and stores in three geographical areas of the US: California, Texas, and Wisconsin. A diagram of a company structure

Description automatically generated

Figure 2: Illustrating the hierarchical nature of the Walmart data (Fig. 1 from Tian, Wang, & E (2021)

From the above figure, it can be observed that the sales data can be divided into two dimensions which is location and product type.

Location: 3 States -> 10 Stores

Product: 3 Product categories-> 7 Product departments ->3049 Products

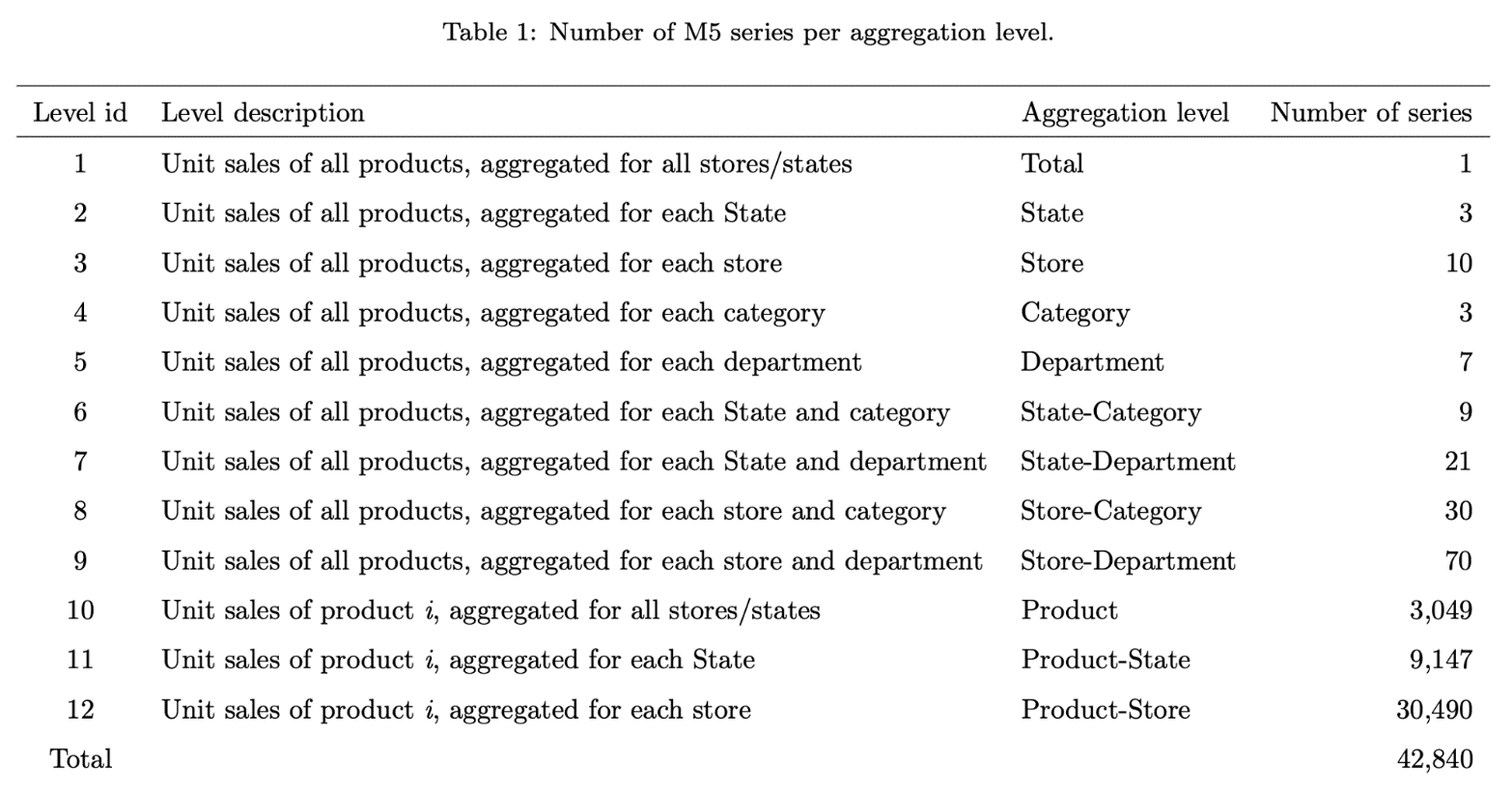


Figure 3: The above figure from (Hallam, J) breaks down the number of series as per the aggregation level.

Skforecast is a Python library that facilitates and simplifies the application of regression algorithms on time series. The library provides effective tools for training, validation, and prediction and covers a variety of cases which are commonly encountered with time-series. The library is built with using the scikit-learn API and has access to a wide array of functionalities such as feature engineering, model selection and hyperparameter tuning to name a few. Skforecast empowers the users to work on crucial aspects of the project while it manages the intricacies of time series analysis.

In this project, we utilized the skforecast library to perform time-series forecasting on Walmart sales data. From the figure above, it can be observed that the data has a hierarchy which is a challenging and computationally expensive task to manage. So first, we started by applying LightGBM regression model on HOBBIES\_1\_018\_CA\_1\_validation product time series which showcased intermittent behavior. Here we generated various iterations of the model and during each iteration we made some specific changes such as changing model parameters, selecting specific features and backtesting methods. Then we did the same for FOODS\_3\_555\_CA\_1\_validation but this time, series had a continuous behavior.

1) Croston Method - which is considered as the first method to address intermittent data (Nikolopoulos, 2021). 2) LightGBM model as it was the most effective technique applied in the M5 competition. The research also aims to apply Neural Networks and assess its performance.

This research questions addressed are as follows:

1. How accurate are the forecasts when comparing the Croston and LightGBM technique?
2. How can we maximise forecasting accuracy using feature engineering and parameter tuning?
3. M5 Competition RQ: “Can you estimate, as precisely as possible the point forecasts of the unit sales of various products sold in the USA by Walmart?” (M5 Forecasting-Accuracy, 2020)

Time series Decomposition:

Concepts explained:

Autocorrelation function:

As correlation is used to measure the linear relationship between two variables, Autocorrelation is used to measure the linear relationship between the original time series and its past values. For example, if yt is the original time series and yt-1 is the lagged by 1 version of time series. Then autocorrelation helps us understand if there is any relationship between our original time series and time series using previous day values (Hyndman and Athanasopoulos, 2021, ch.2.8).

Autocorrelation of a time series is calculated at various lags such as yt-1, yt-2,yt-3, yt-k. Autocorrelation helps in finding repeating patterns and trends in the time series data (Geeks for Geeks, 2024). For example, positive autocorrelation at specific lags for example at yt-7, yt-14 may indicate the presence of weekly seasonality (Geeks for Geeks, 2024). Autocorrelation helps determine the order of ARIMA and MA models by providing information as to which lags terms to utilize in the model (Geeks for Geeks, 2024).

Partial autocorrelation:

Partial autocorrelation measures the direct correlation between yt and yt-k after removing the correlation introduced by intermediate lags on yt and yt-k. The high partial correlation at lag k indicates that lag k adds additional information which is not accounted by lags before it. Thus, partial correlation can help us identify lags which can be used as features in our model (Machine Learning Mastery, 2020).

Time Series Stationarity:

Initial aim was to compare the performance of the Croston Method and Light GBM technique but through further research, the capabilities of the LightGBM and its success in the M5 competitions made it necessary to focus on the domain of Gradient Boosting Machines and LightGBM in particular Furthermore, the challenging nature of the forecasting problem and the hierarchical nature of the data, the main focus was to appropriately wrangle the data in a way which will start giving results and then improve iteratively by focusing on feature engineering and forecast accuracy.

The challenging nature of the forecasting problem

Light GBM model:

Wisdom of the crowd highlights that in many cases the answer to a complex question is more reliable when it is supported by a larger audience than by a single expert. Similarly, aggregating the predictions of a group of models/predictors can often lead to better predictions than accepting the prediction of a single model. The group of models is called an ensemble and the technique is called Ensemble Learning (Géron, 2019, Chapter 7).

Boosting is a type of Ensemble learning which combines several weak models or learners into a strong learner. The idea revolves around the notion that each model is trying to correct its previous models (Géron, 2019, Chapter 7), (Freund and Schapire, 1997).

AdaBoost works on the basic notion that the new model can improve over the previous ones by focusing on the training points that the previous model could not fit (Freund and Schapire, 1997).

Gradient boosting is another popular boosting algorithm which operates by adding models in a consecutive manner. However, in contrast to AdaBoost which modifies an instance’s weights, gradient boosting fits the new model to the errors made by the previous predictor (Breiman, 1997), (Friedman, 2001).

I will try to explain this concept with an example:

Show statquest implementation

Gradient boosting decision tree (GBDT) (Friedman, 2001) is a popular machine learning algorithm known for its efficiency, accuracy, and interpretability. It has shown to achieve high quality performances in tasks such as multi-class classification (Li, 2012), click prediction (Richardson, Dominowska and Ragno, 2007) and rank learning (Burges, 2010).

However, the scale of data has expanded in terms of the number of features and number of data points simply called the big data. This scale of data has posed difficulties on the Gradient boosting decision tree (Ke et al., 2017). The current implementation of GBDT requires that for each feature, the algorithm will scan through all the data points and calculate the information gain of all the possible split points and this becomes alarming in terms of computational and time costs (Ke et al., 2017).

Thus, two new techniques are proposed by (Ke et al., 2017) called the Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) which makes up the LightGBM. Experimental results highlight the prowess of LightGBM in that it is shown to be an improvement over XGBoost and SGB in terms of computational and memory costs (Ke et al., 2017).

The motivation to apply LightGBM model to M5 Competition Data comes from the competition’s leaderboard. The winner of the M5 competition leveraged an equal weighted combination of different LightGBM models and various features such as calendar information, special days, promotions, prices and unit sales data were incorporated (Makridakis, Spiliotis and Assimakopoulos, 2022). The second placed method utilized the same equal weighted approach on LightGBM models with external adjustments from forecasts produced by N-BEATS(deep-learning NN for time series forecasting (Oreshkin et al., 2019). The fourth placed created forecasts for the product-stores series using non-recursive LightGBM models and trained for each store(i.e.10 models) whereas the fifth placed candidate applied recursive LightGBM models and was trained for each of the seven departments. Additionally, among the top 50 performing methods, a large number of candidates leveraged approaches similar to the first placed candidate by training recursive and non-recursive LightGBM models for each store, department, or store-department. The above results highlight the capabilities of the LightGBM model and its ability to process multiple relation time series and leverage exogenous/explanatory variables to improve forecast accuracy (Makridakis, Spiliotis and Assimakopoulos, 2022).

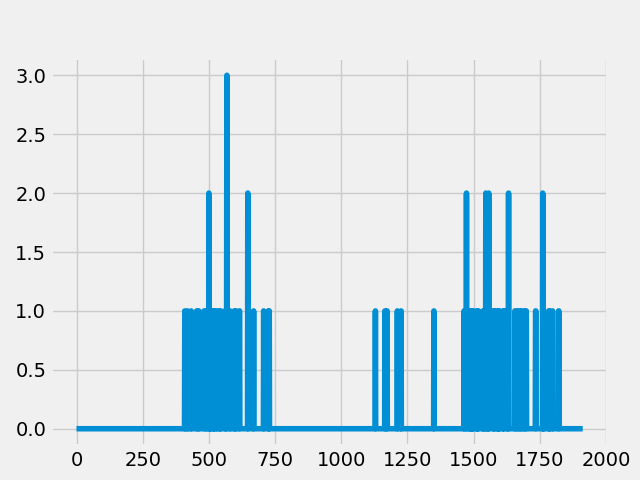
This project aims to leverage these capabilities of LightGBM models at a smaller scale by first applying these techniques to individual products and then aggregating them to higher levels such as department, categories, and stores. In addition we will testing various feature engineering and hyperparameter tuning approaches to improve the model performance and robustness.

**Chapter 1: Light Gradient boosting machine on HOBBIES\_1\_018\_CA\_1\_validation**

The M5 competition contains 3049 products which represent different departments, categories, store, and states. Due to the limitations of memory and the complexity of the problem. The initial objective was to forecast the sales of the single item/product/ SKU. The product chosen was HOBBIES\_1\_018\_CA\_1\_.

**Iteration 1:**

The plot of the figure is as follows: Explain the context here. Not clear



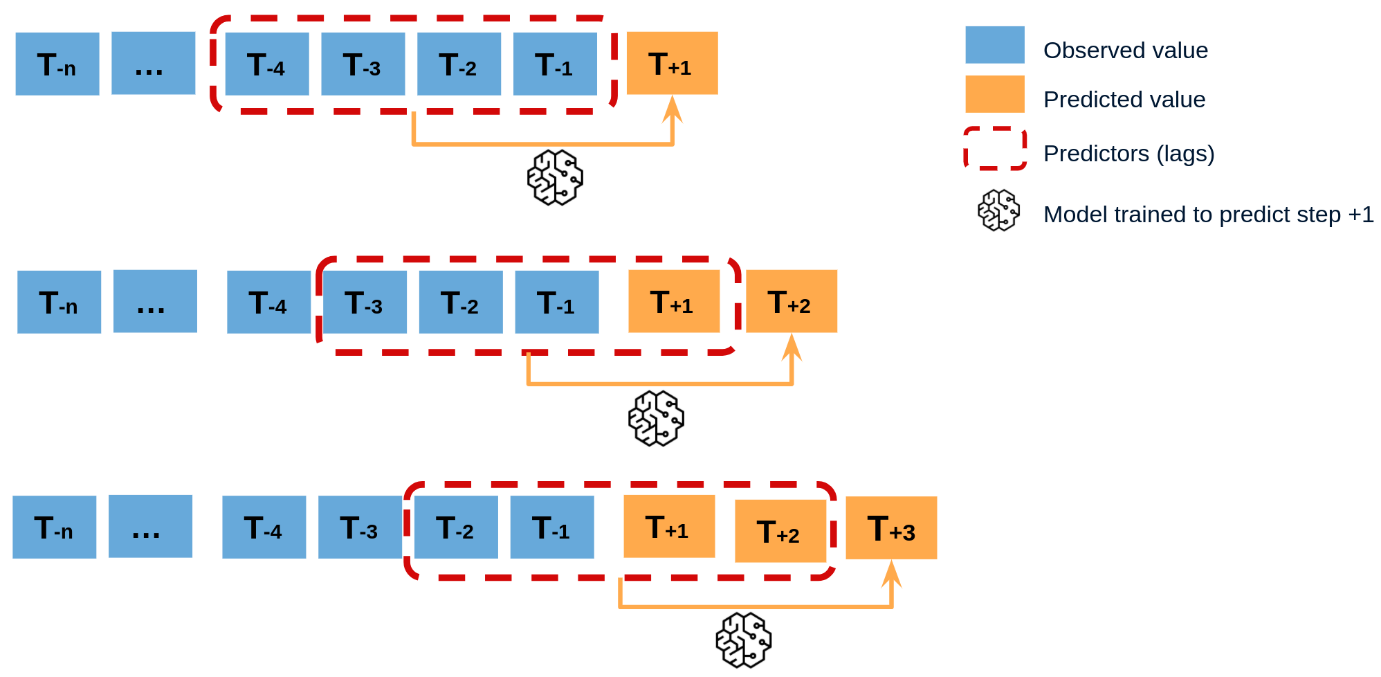
**Add x , y label and title and caption**

As observed from the above plot, where the x-axis is the number of days recorded and the y-axis is the sales on a given day. The plot illustrates that the time-series contains intermittent sales.

To model the series, the data was divided into three-parts train, validation, and test. The first 1322 data points were allocated for the train, the next 265 for validation and the next 326 for the test. It is important to note that techniques like random shuffle or traditional cross-validation does not work on time-series data as they violate the temporal order.

The python library used for forecasting is skforecast. To train a time series in skforecast we utilised two classes 1) ForecasterAutoreg and 2) Backtesting\_forecaster

The application of machine learning models to forecasting comes with a requirement of transforming the times series into a matrix form where features such as lags and external variables are utilised. Lags are values of the times series at some time points in the past. This transformation enables the machine learning models to capture patterns and relationships that exist between the past and future values. In this iteration we will be utilising recursive multi-step forecasting technique.



Add caption

The above image describes the concept behind recursive forecasting. In, the first step, we are training the model on the observed values and predicting one step into the future. In the next step, along with our observed values, we are incorporating the value predicted in the previous step to retrain the model and predict one step ahead and so on. This is basic concept behind Recursive forecasting which we will be applying using our class ForecasterAutoreg.

The second concept is the backtesting\_forecaster which is simply to say cross-validation for time series. It is utilised to evaluate the performance of the model on historical data and is considered an essential step in the development of a time-series forecasting model.

Inputs to Forecasterautoreg and backtesting\_forecaster are as follows:

forecaster = ForecasterAutoreg(

regressor=LGBMRegressor(

learning\_rate=0.1,

max\_depth=5,

n\_estimators=500,

num\_leaves=32, # Set num\_leaves to be greater than 2^max\_depth

random\_state=123,

force\_row\_wise=True,

# or force\_col\_wise=True if memory is a concern

),

lags=24

)

metric, predictions = backtesting\_forecaster(

forecaster=forecaster,

y=data['sales'],

initial\_train\_size=initial\_train\_size,

fixed\_train\_size=False,

steps=10,

refit=False,

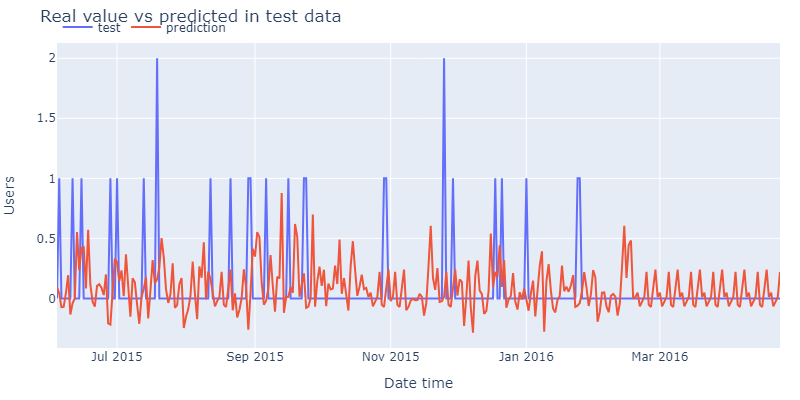
metric='mean\_squared\_error',

verbose=True # Change to True for detailed information

)

Upon running the above classes, the some of the inconsistencies which we observed were as follows: 1) **The backtest error: 0.12077583661944795** did not seem representative which rased doubt in the mean-squared-error metric and its appropriateness for the data where majority of the sales values were zero.

2) In the test-set the model is unable to capture any peaks or any meaning relationships in the data. This can be observed in the below figure:



**Add Caption**

Iteration 2:

In the second iteration, along with the sales data we joined calendar dataset which consisted of different features such as weekday and different event types. A lag of 28 days and rolling windows statistics such as mean and standard deviation for 7 and 28 days were calculated and incorporated as features. Furthermore, time features such as day and month were calculated and incorporated into the model. **WHYYYY?**

forecaster = ForecasterAutoreg(

regressor=LGBMRegressor(

learning\_rate=0.1,

max\_depth=5,

n\_estimators=300,

num\_leaves=40,

objective = "tweedie",# Set num\_leaves to be greater than 2^max\_depth

random\_state=123,

force\_row\_wise=True,

# or force\_col\_wise=True if memory is a concern

),

lags=24,

)

metric, predictions = backtesting\_forecaster(

forecaster=forecaster,

y=data['sales'],

initial\_train\_size=initial\_train\_size,

fixed\_train\_size=False,

steps=10,

refit=True,

metric='mean\_squared\_error',

verbose=True,

exog = exog # Change to True for detailed information

)

Some of the parameters which were incorporated in this iteration are objective = ‘tweedie’ in

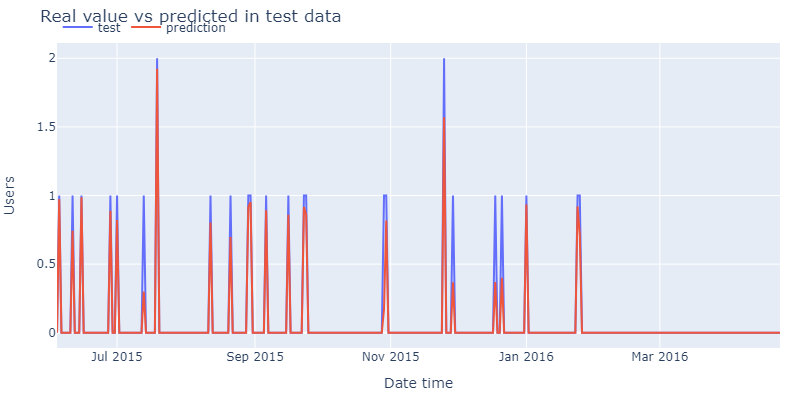
the first class and exog = exog in the second class. The first parameter was selected as

tweedie due its properties where it can have mixture of zeros and non-negative data points.

The second parameter exog is utilised to incorporate additional variables such as our day,

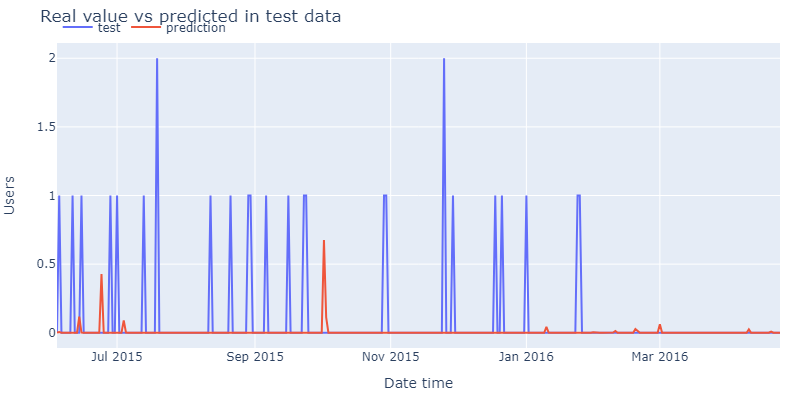
year and different events. Upon running this experiment, the backtest error is :

0.009078603489111351 and the graph is as follows:



**LABEL????**

Upon observing, it looks like the model can capture the peaks well. But this is a result of a major error in the backtesting\_forecaster class where in the parameter exog which enables to incorporate external variables, we are giving target i.e. (y) as feature in exog. Upon removing the target as a feature, the observed graph is as follows:



**LABEL????**

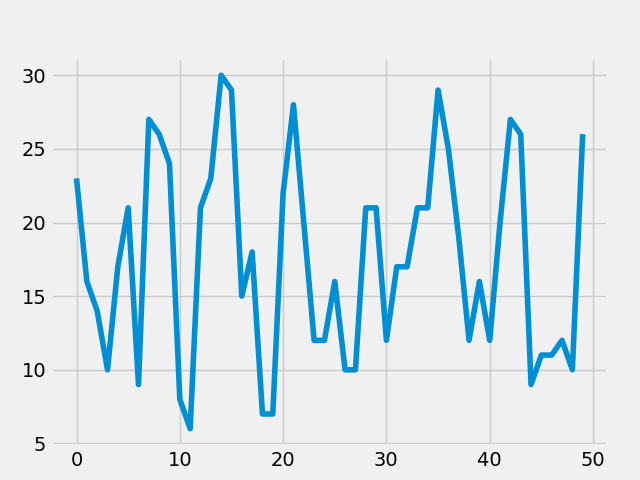
As observed from the above figure, the only improvement from first iteration is that the tweedie distribution has enabled our predictions to stay at zero instead of fluctuating in the negative y-axis. Apart from this, its seems that model is unable to capture any relationships in the data which raises doubts and emphasizes the need to re-examine the experiment.

Chapter 2: Light Gradient boosting machine on FOODS\_3\_555\_CA\_1\_validation

**How is it different from chapter 1. What is happen8ing here?**

**Iteration 1:**

The above time-series data had many zero values where trend and seasonality patterns could not be observed. Thus, we turn our attention to another SKU which has continuous values and try to model it. The graph of the time series looks as follows:



**X LABEL , Y LABEL TITLE , CAPTION**

In the first iteration the time-series was modelled by incorporating features such as 'year', 'snap\_CA', 'snap\_TX', 'snap\_WI', 'y\_window\_7\_mean', 'y\_window\_7\_std', 'y\_window\_28\_mean', 'y\_window\_28\_std', 'month', 'day' and lag values till 30(be more specific what you mean).

The parameters for the LGBM regressor are:

learning\_rate=0.1,

max\_depth=6,

n\_estimators=500,

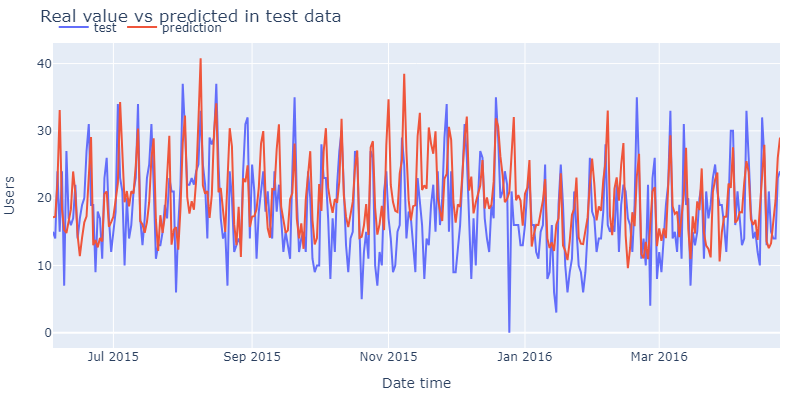
num\_leaves=30,

objective = "tweedie

random\_state=123,

force\_row\_wise=True

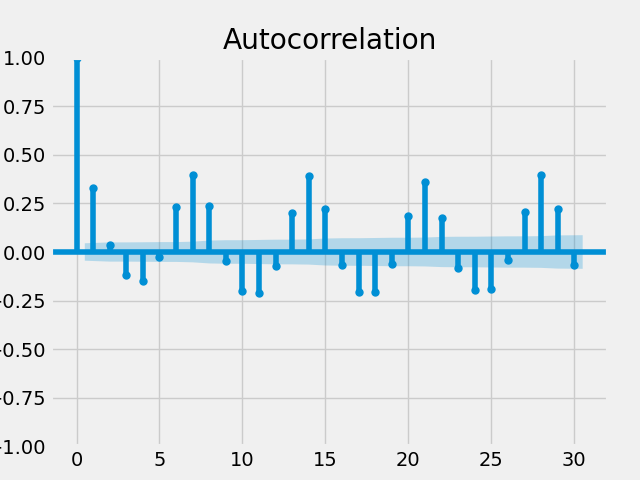
This resulted in the backtesting error of 39.0764816046939



From the above figure, for the first iteration, the model can capture peaks to some extent but not troughs.

Thus, further modelling and feature engineering is required to improve performance.

Let us look at the Autocorrelation function of this time series. The autocorrelation function enables us to visualise the correlation of a time series with a lagged version of itself. (explain better) (use auto correlation in LR) (the less questions—better writing)



From the above figure, it appears that lags 1, 6, 7, 8, 10, 11 are particularly significant. Incorporating the following lags for our next iteration.

Iteration 2:

**WHYYYY?**

Features incorporated are 'year', 'month', 'day' and lags 1, 6, 7, 8, 10, 11 and model parameters are kept the same as iteration 1.

This resulted in the backtesting error of 46.95461867886095

Iteration 3:

**WHY???**

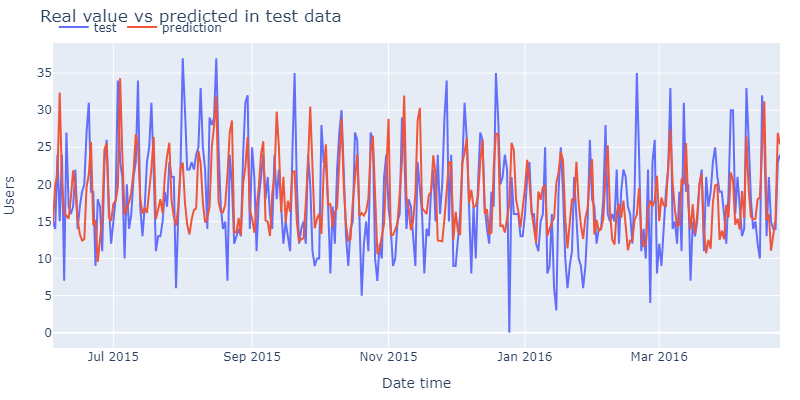
Features incorporated are 'year', 'snap\_CA', 'snap\_TX', 'snap\_WI', 'y\_window\_7\_mean', 'y\_window\_7\_std','y\_window\_14\_mean', 'y\_window\_14\_std', 'y\_window\_28\_mean',

'y\_window\_28\_std', 'month', 'day' and lags 1, 6 , 7, 8, 10, 11 and model parameters are kept the same as iteration 2.

This resulted in the backtesting error of 39.12254845450437

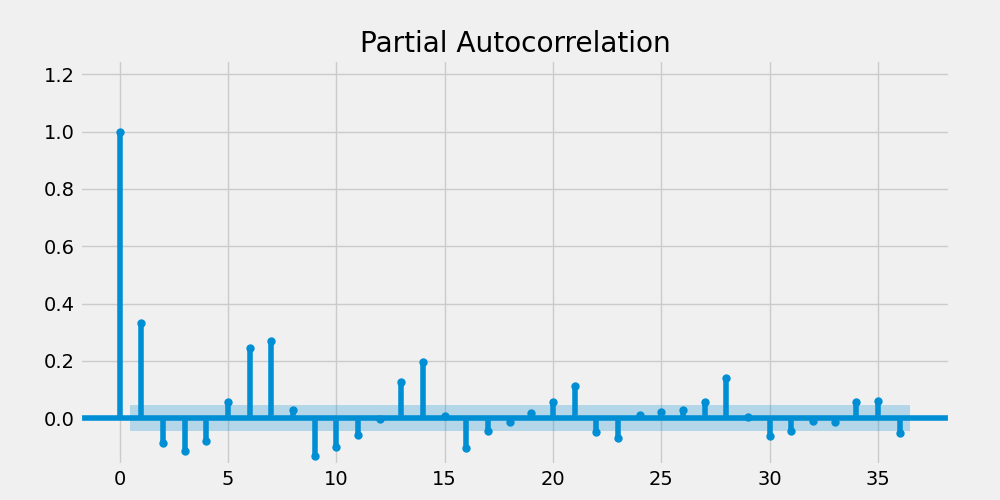
**Iteration 3: Check this plzz**

Interestingly, after incorporating lags features until 60 and num\_leaves = 64, the backtesting error was reduced to 35.76475385173678



Iteration 4:

In this iteration, we will applying autocorrelation and partial autocorrelation function to our time series. Partial autocorrelation measures the direct correlation of a time series with a lagged version of itself by eliminating the effects of the intermediate lags. This helps us to identify lags which have a direct strong effect on the original time series and can be a strong indication of utilising that lags as our features. Thus, the results from partial autocorrelation will help us decide which lags to incorporate in our model**. (why did I use auto, par auto?) (why did it help?) What can you show?**

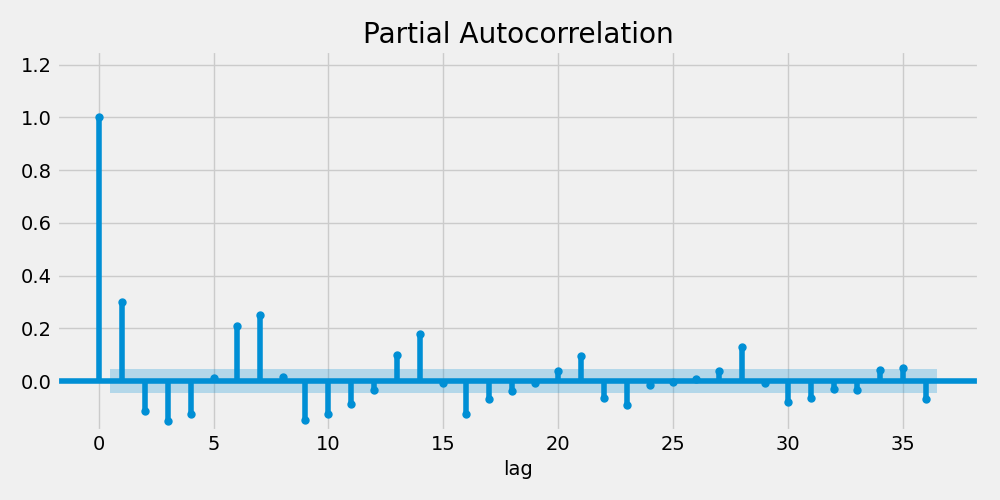


The above figure gives significant PACF values at lag 1, 6 and 7. The PACF value at lag7 and its multiples at 14, 21, 28 indicates that there is some weekly seasonality in the data. Furthermore, lag1 and lag6 also look significant and we incorporate them in this iteration.

Furthermore, we calculated PACF without de-trending and de-seasonalising our data as the main assumptions to calculate PACF requires the time series to be stationary i.e. mean and variance of the time series should be constant.

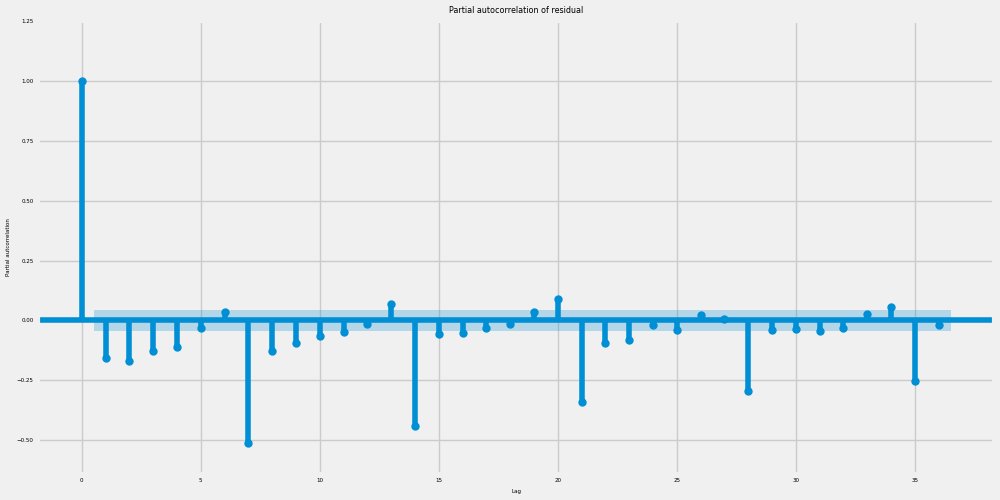
First, we calculate PACF by de-trending our data. This can be achieved with a technique called as LOWESS.

After applying LOWESS, the de-trended data is obtained which is illustrated below:



As the original data did not have a strong trend to it, the PACF of the de-trended series is like that of the original series.

Now, PACF is calculated by de-trending (i.e. removing the trend component) and de-seasonalising (i.e. removing the seasonal component from the data). What remains is the residual for which PACF will be calculated and plotted.



There appears to be some significant lags at multiples of 7 which suggests that some part of the seasonal component is still in the residuals and was not perfectly extracted by STL. Practically speaking from looking at this plot there would not be an additional lag beyond 1 or 2 that we would want to add for feature engineering purposes.

So after, calculating PACFs, we will be incorporating lags1, 6 and 7 into our model. **Backtest error: 42.58595000607901**

Iteration 5:

In this iteration, we are trying to replicate the competition’s objective of forecasting for the next 28 days ahead. The first 1885 data points were utilised for training our model and the remaining 28 data points were used for testing. The significant change in this iteration is the backtesting forecaster parameters. Here backtesting with refit and increasing training size was implemented which was obtained by change the following parameters:

refit=True*and*fixed\_train\_size=False.

Number of observations used for initial training: 1885

Number of observations used for backtesting: 28

Number of folds: 3

Number of steps per fold: 10

Number of steps to exclude from the end of each train set before test (gap): 0

Last fold only includes 8 observations.

Fold: 0

Training: 2011-01-29 00:00:00 -- 2016-03-27 00:00:00 (n=1885)

Validation: 2016-03-28 00:00:00 -- 2016-04-06 00:00:00 (n=10)

Fold: 1

Training: 2011-01-29 00:00:00 -- 2016-04-06 00:00:00 (n=1895)

Validation: 2016-04-07 00:00:00 -- 2016-04-16 00:00:00 (n=10)

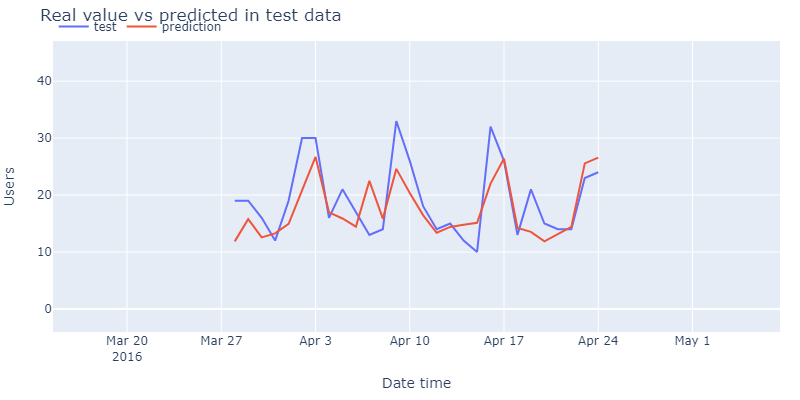
Fold: 2

Training: 2011-01-29 00:00:00 -- 2016-04-16 00:00:00 (n=1905)

Validation: 2016-04-17 00:00:00 -- 2016-04-24 00:00:00 (n=8)

In Fold 0, the model used 1885 data points to train and predicted the next 10 steps in the horizon. In Fold 1, the model incorporated next 10 data points into training and predicted the next 10 steps in the horizon. As you can see the size of training set is increasing by 10 at every fold. The model was trained 3 times, where for folds 1 and 2 model incorporated additional 10 data points and was re-trained.

The backtesting error was 22.652761244964772



Visualize one item across 10 stores, seasonality, similarity,

Make calculation to identify what items are similar at a pnt in time.

What is the spread.. diff between the highest and the lowest.

Finding most similar items among them. Percentage difference between them. Visualise and then rolling averages.

References:

Hyndman, R.J., & Athanasopoulos, G. (2021) *Forecasting: principles and practice*, 3rd edition, OTexts: Melbourne, Australia. OTexts.com/fpp3. Accessed on: 13/07/2024

Geeks for Geeks (2024) *AutoCorrelation*. Available at: https://www.geeksforgeeks.org/autocorrelation/ (Accessed: 13/07/2024).

Machine Learning Mastery (2020) *A Gentle Introduction to Autocorrelation and Partial Autocorrelation* Available at: https://machinelearningmastery.com/gentle-introduction-autocorrelation-partial-autocorrelation/ (Accessed: 13/07/2024).

Petropoulos, F., Apiletti, D., Assimakopoulos, V., Babai, M.Z., Barrow, D.K., Taieb, S.B., Bergmeir, C., Bessa, R.J., Bijak, J., Boylan, J.E. and Browell, J., 2022. Forecasting: theory and practice. *International Journal of Forecasting*, *38*(3), pp.705-871.

Geron, A. (2019) *Hands-on Machine Learning with Scikit-Learn, Keras and Tensorflow*. 2nd Edition. USA: Oreily Media

Freund, Y. and Schapire, R.E., 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, *55*(1), pp.119-139.

Breiman, L., 1997. *Arcing the edge* (pp. 1-14). Technical Report 486, Statistics Department, University of California at Berkeley.

Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp.1189-1232.

Li, P., 2012. Robust logitboost and adaptive base class (abc) logitboost. arXiv preprint arXiv:1203.3491.

Richardson, M., Dominowska, E. and Ragno, R., 2007, May. Predicting clicks: estimating the click-through rate for new ads. In Proceedings of the 16th international conference on World Wide Web (pp. 521-530).

Burges, C.J., 2010. From ranknet to lambdarank to lambdamart: An overview. Learning, 11(23-581), p.81.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. and Liu, T.Y., 2017. Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems, 30.

Oreshkin, B.N., Carpov, D., Chapados, N. and Bengio, Y., 2019. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*.

Makridakis, S., Spiliotis, E. and Assimakopoulos, V., 2022. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, *38*(4), pp.1346-1364.