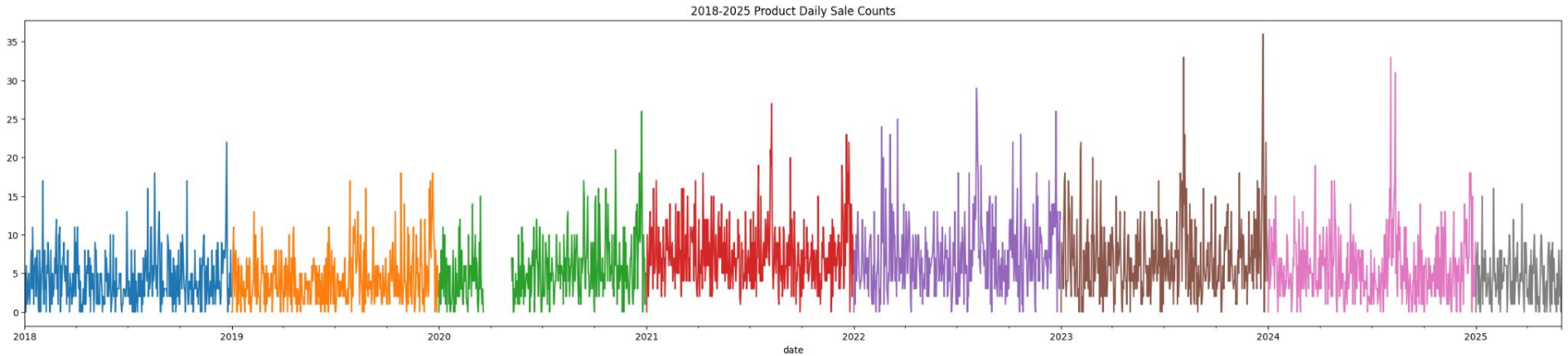
Product Sales Forecasting for Local Retail Store

https://github.com/isaacjeon/product_sales_forecasting

Problem Description

- This project will make use of sales data of a specific style of work pants for a small individually-owned clothing retail store to **analyze** and **forecast daily sales** of that product.
  - Analysis of correlations, trends, and seasonality
  - Forecast horizon of 28 days

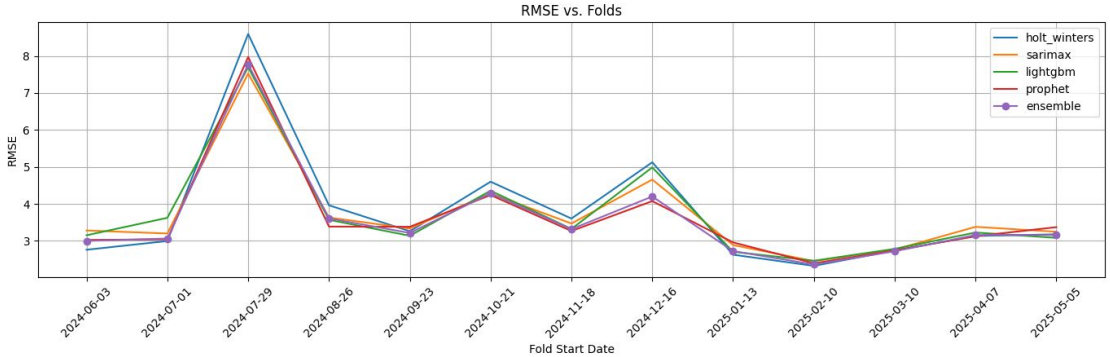


Overview

- Dataset includes time series data starting from 2018 containing sale counts as well as the base price of the product for each date.
  - Requires cleaning and processing
  - Certain models benefit or require extensive feature engineering (especially LightGBM)
- Dataset has nonlinear trend and both weekly and yearly seasonality (multi-seasonality)
  - Particularly strong seasonal effects during back-to-school (late July to early August) and Christmas shopping seasons
- Methods: **Holt-Winters**, **SARIMAX**, **LightGBM**, **Prophet**
  - Evaluation metrics: **RMSE**, **MAE**, **RMSLE**, **SMAPE**
  - Evaluated and tuned with **expanding window cross-validation**
  - Additional **ensemble** model using mean of predictions from each method

Overview

- Evaluated on 13 test folds of 28 consecutive days.
  - Holt-Winters is an ensemble average of two H-W models of 7 and 364 seasonal periods
  - Prophet was generally the most accurate.
  - LightGBM was the fastest and second-most accurate.
  - SARIMAX had comparable accuracy to LightGBM but is more computationally expensive.
  - Ensemble average of the four models had improved performance.
- Results are not great (median of 5, but predictions are off by about 2.9 on average)
  - Too much random noise at a daily level that doesn't model well
  - Possibly not enough data to forecast at a daily level
  - Can still be of practical use if aggregated by one- or two-week periods



Metric	Holt-Winters	SARIMAX	LightGBM	Prophet	Ensemble
RMSE	3.7633	3.7024	3.7016	3.6152	3.5843
MAE	2.9712	2.9481	2.9455	2.8855	2.8705
RMSLE	0.7043	0.7243	0.7198	0.6981	0.7020
SMAPE	65.9368	63.9775	63.6657	64.0489	63.4908

Related Work

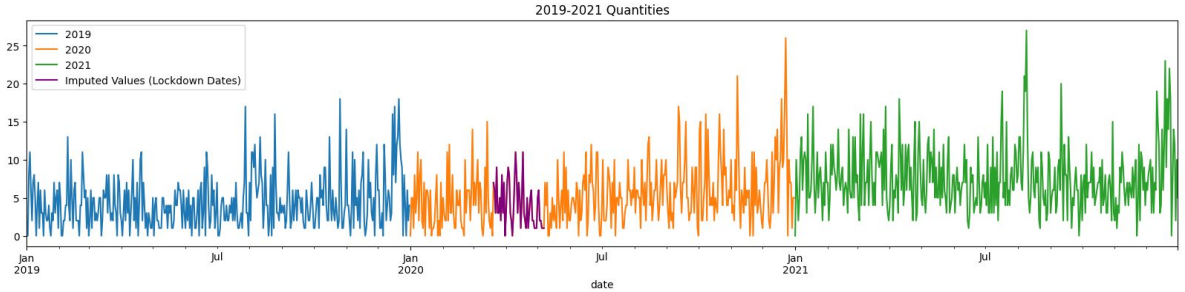
- Corporación Favorita Grocery Sales Forecasting and Walmart M5 Forecasting datasets
  - Kaggle competitions for retail sales forecasting
- Like my dataset, these are time series datasets based on retail sales.
- Unlike my dataset, the scale of these datasets are much larger and include data for multiple stores and product.
  - My problem is simplified since it involves one store and product, but the smaller amount of data provides an extra challenge.
  - Unlike contestants for these competitions, I have prior knowledge about my project's dataset and the store it comes from, and can leverage that information for feature engineering.
- Machine learning approaches that worked well for this dataset include tree-based methods like Random Forests, XGBoost, and LightGBM.
- Other methods used include exponential smoothing, ARIMA, and deep learning methods like LSTMs and CNNs.

Project Timeline

- Originally tried to forecasting customer foot traffic at a sub-hourly level, but scrapped the idea and changed the problem
  - Tried Random Forests, LightGBM, and Prophet but predictions were “smooth curves”
  - Intervals too granular with high variability, so hard to predict spikes and dips
- Switched to the current problem
  - Dataset was the same, just needed to extract the data for the new problem
  - Cleaned and processed data, including removal of outliers and imputing missing values
  - Exploratory data analysis
  - Feature engineering
  - Modeling and evaluation
  - Finish report and slides

Data Preprocessing and Cleaning

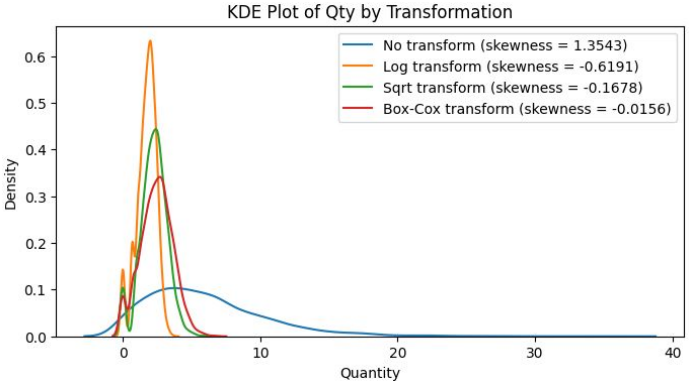
- Preprocessed original dataset of history of all transactions since 2018
  - Preprocessed dataset contained dates with associated sale count (target value) and base price of product
- Notable data cleaning steps
  - Remove outliers (large purchases made in a single transaction) above a threshold
  - Handle transactions based on “sale status” (regular sale, discounted, returned, voided)
  - Days with no sales of the product needed an entry with 0 target value
    - Days where the store is closed (i.e. Christmas, New Year’s Day) would also have 0
  - 7 consecutive week in 2020 with no data because of forced closure due to pandemic
    - Impute values based on previous year’s values compared to current year’s values



Target Value Distribution

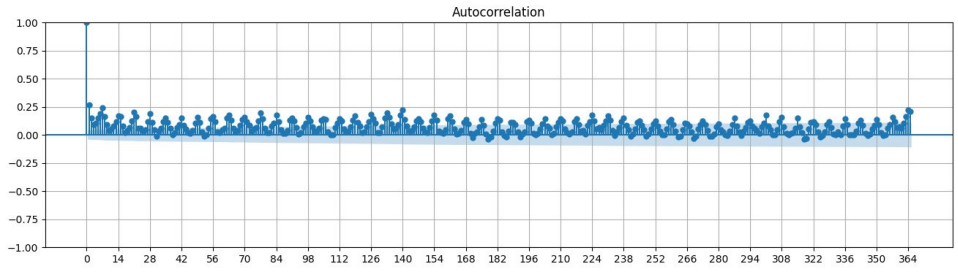
- Target distribution is **right-skewed**
  - Models may have trouble predicting outliers
- Certain transformations to the target value can improve performance for statistical models that assume normality
  - Particularly Holt-Winters and SARIMAX
  - Box-Cox results in a distribution closer to normal
- A time series is **stationary** if its statistical properties such as mean and variance do not depend on time.
  - SARIMAX assumes stationarity (this is not a requirement for the other three approaches)
  - The **Augmented Dickey-Fuller (ADF) test** can be used to check for stationarity. Applying the test to the data results in a p-value lower than 0.05, so we can reject the null hypothesis of non-stationarity and conclude that the time series is stationary.

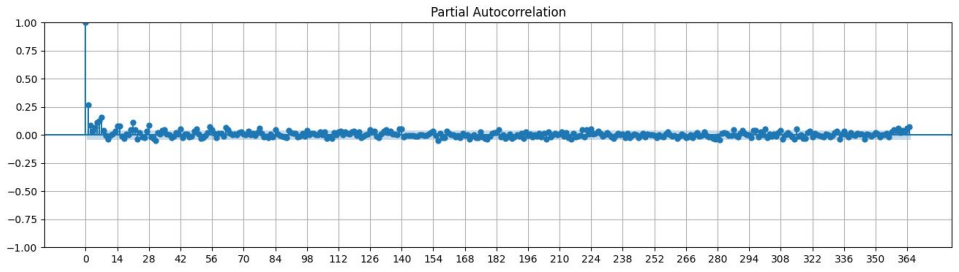
mean	5.909398
std	4.385056
min	0.000000
25%	3.000000
50%	5.000000
75%	8.000000
max	36.000000



Autocorrelation and Partial Autocorrelation

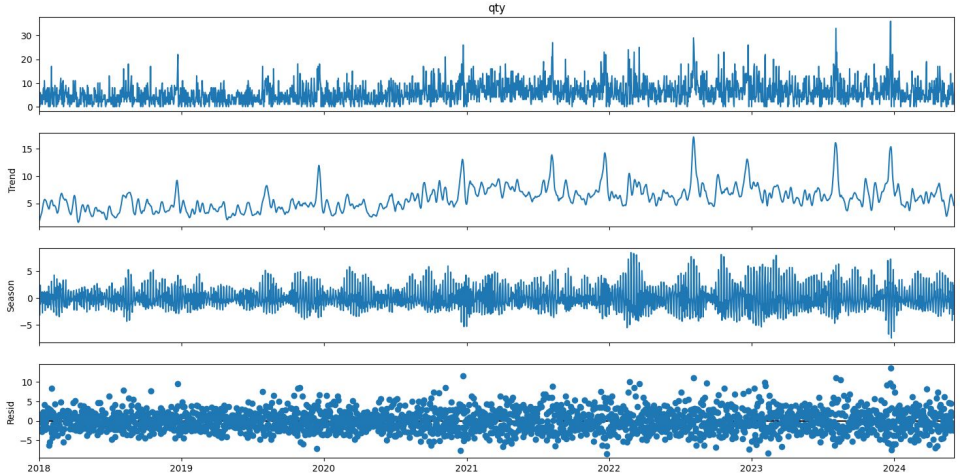
- **Autocorrelation** measures the correlation of a time series with a delayed version of itself; **partial autocorrelation** is similar but removes the influence of earlier lags, thus representing the direct effect of the lag.
- The **ACF and PACF plots** can be used to identify the AutoRegressive (AR) and Moving Average (MA) terms for the SARIMAX model (the p and q parameters, respectively), as well as seasonality.
  - ACF decays slowly, while PACF decreases relatively quicker. This suggests an autoregressive process so we might try fitting the model with p = 1 or 2.
  - There are peaks every 7 lags, indicating **weekly seasonality**.



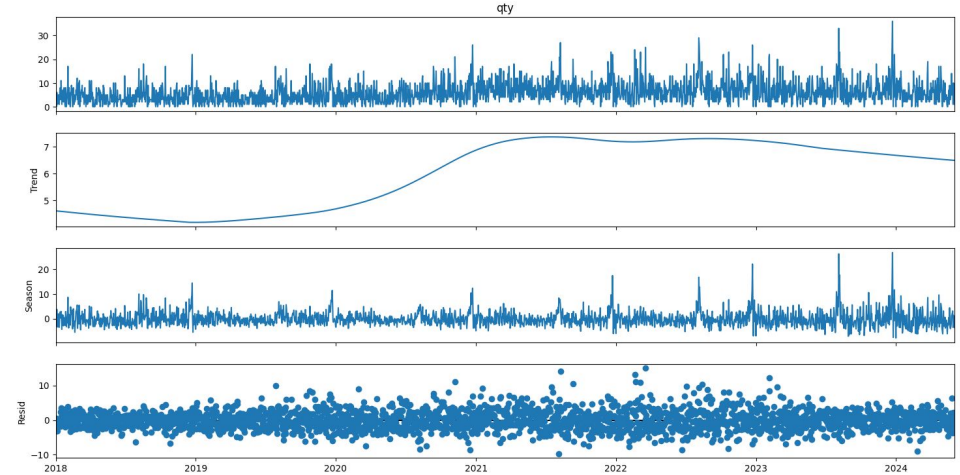


Seasonal and Trend Analysis with STL Decomposition

- **STL decomposition (Seasonal-Trend decomposition using LOESS)** decomposes a time series into its **trend**, **seasonal**, and **residual** (noise) components.
- The sum of these three components results in the original time series.



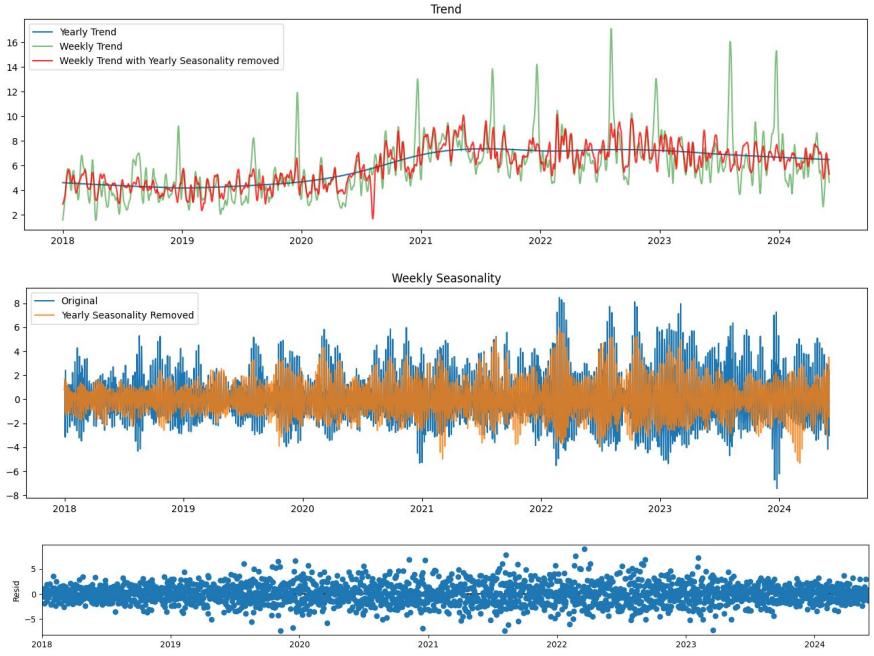
7-day period STL decomposition



365-day period STL decomposition

Seasonal and Trend Analysis with STL Decomposition

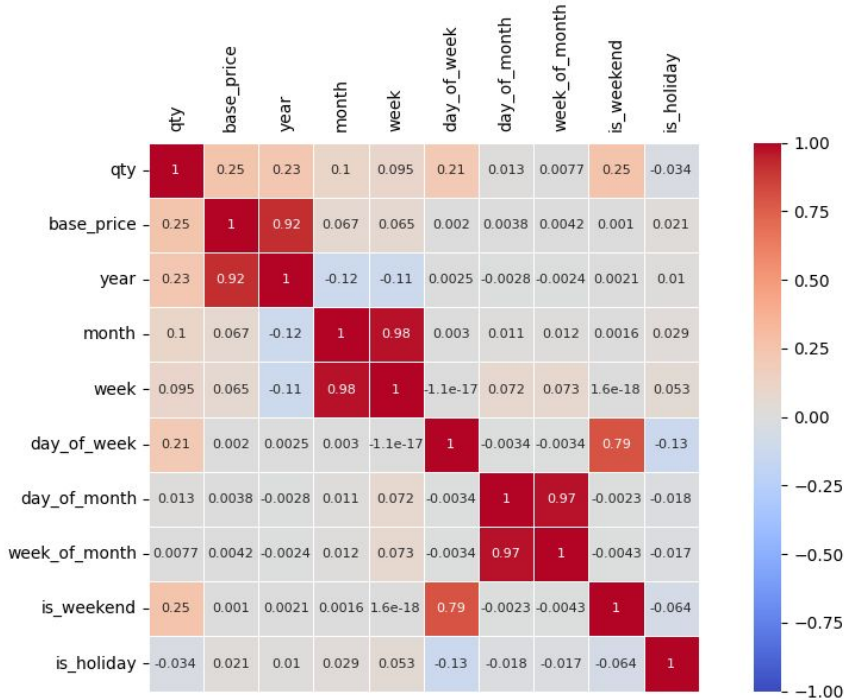
- Removing 365-day seasonality then performing STL decomposition with 7-day periods results in less noise in the trend and seasonal components, and less noise in residuals for the earlier and later years
- There is both **weekly and yearly seasonality**.
  - An issue for Holt-Winters and SARIMAX which model only one seasonal period
- One approach for modeling is to remove a seasonal component separately and add it back later

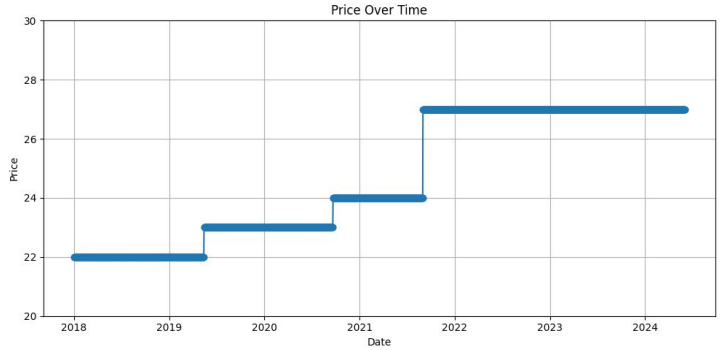


Plots of components from STL decomposition with 7-day periods on time series with 365-day seasonal component removed compared with the same decomposition on the original data.

Correlation matrix

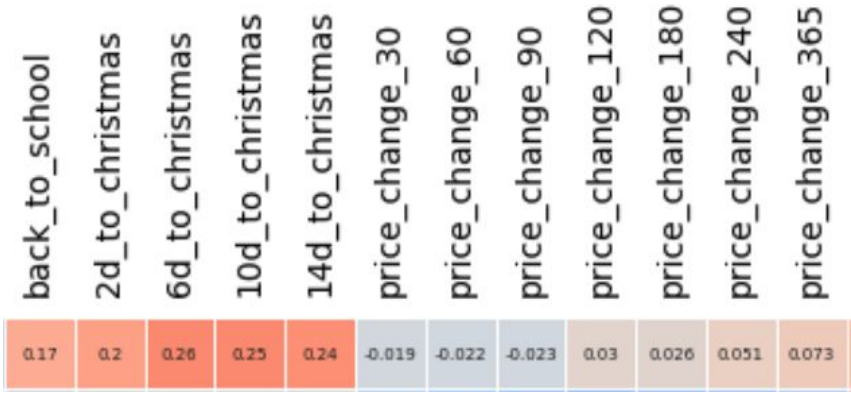
- **Correlation** with target
  - Positively correlated with base_price, year, day_of_month, is_weekend, month, week
  - Negatively correlated with is_holiday, probably because it includes closed days which are guaranteed to have 0 target values
- **Multicollinearity** among features
  - Not an issue for LightGBM, but can be an issue for SARIMAX and Prophet if adding regressors
  - base_price and year are very highly correlated. base_price plot happens to be similar to trend, so correlation with target may be coincidental

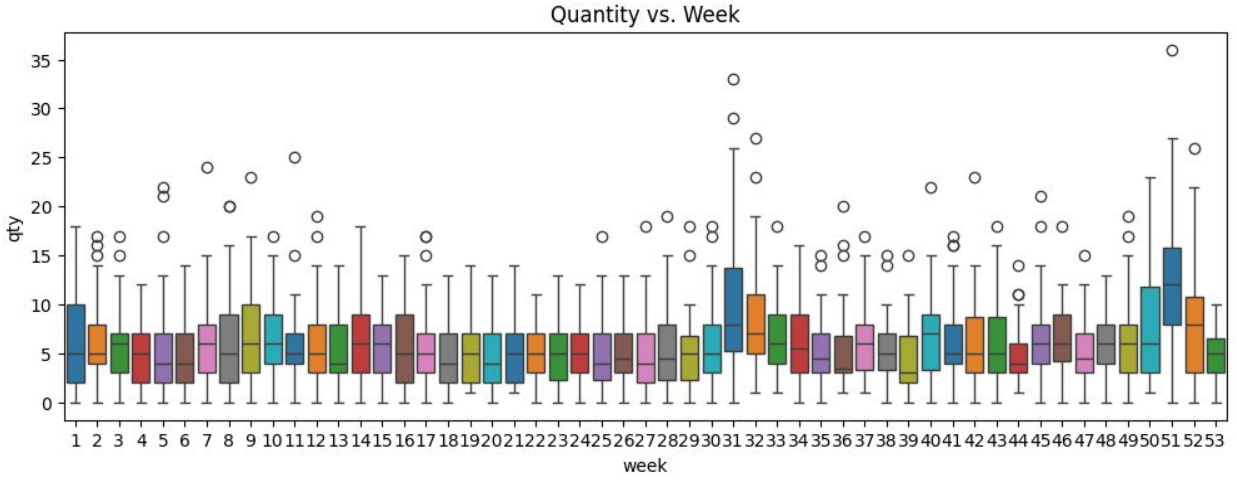




Feature Engineering

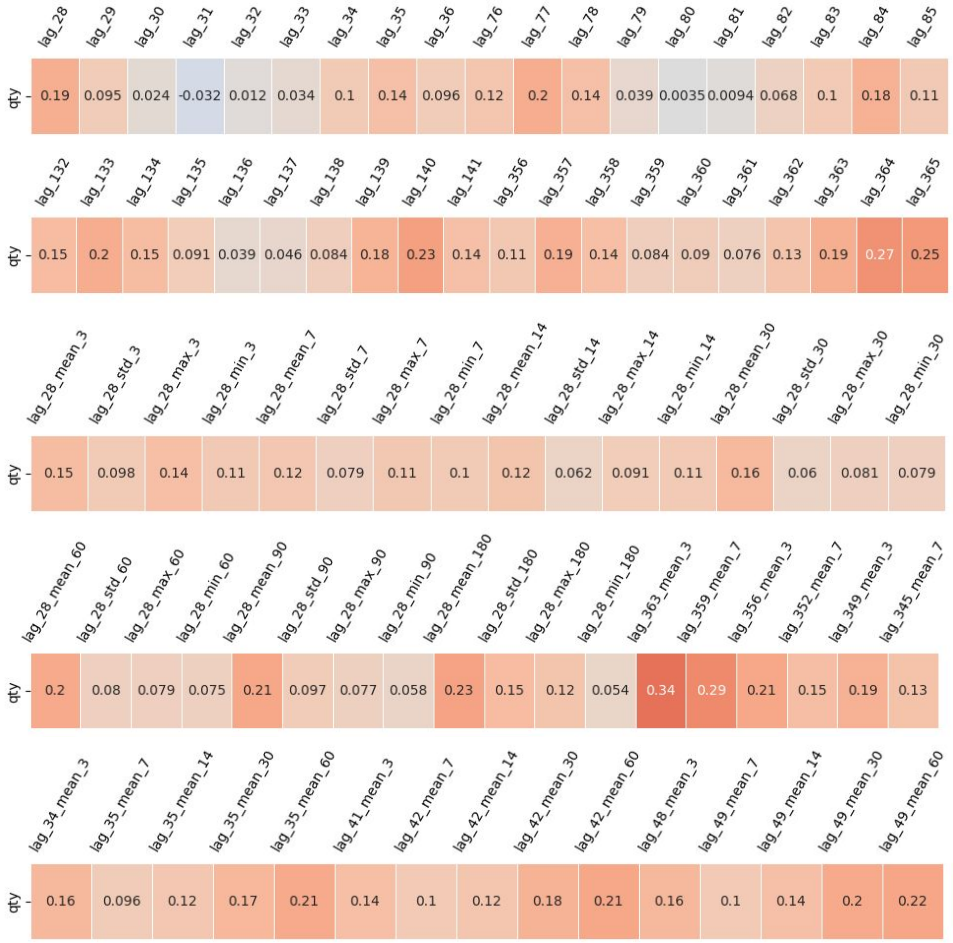
- Binary features for back-to-school and Christmas shopping season days
- Price change features price_change_i is the ratio of the the current day's price to the price from i days before.
    - Correlations with target are rather low, price changes might not be that important





Feature Engineering

- **Lag** features and **rolling window** features for mean, std, min, max
- It is important to not include lags shorter than the horizon (28 days for this project)
- I added a number of features of different combinations of lags and windows.
- As these features rely on past data, the earlier dates in the dataset will have null values for these features.
    - I used lags of up to one year, so I dropped the 2018 data from the dataset.
- Lag_363_mean_3 (lag of 363 days and rolling mean with window of 3, i.e. mean target value for 363-365 days prior) in particular has the highest correlation with the target



Experimental Setup

- Evaluation metrics
  - **Root Mean Squared Error (RMSE)**
  - **Root Mean Squared Logarithmic Error (RMSLE)**
  - **Mean Absolute Error (MAE)**
  - **Symmetric Mean Absolute Percentage Error (SMAPE)**

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}$$

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(log(y_i + 1) - log(\hat{y}_i + 1)\right)^2}$$

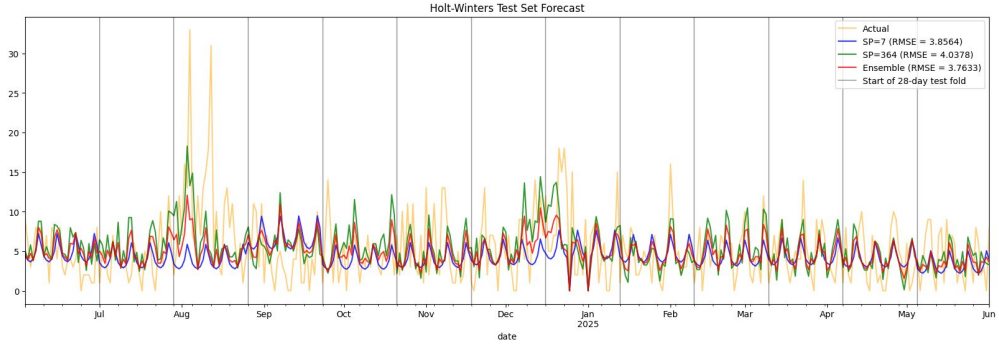
$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right|$$

$$SMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|y_i - \hat{y}_i\right|}{\left(\left|y_i\right| + \left|\hat{y}_i\right|\right)^2 / 2}$$

- **Expanding window cross-validation**
  - Because data is time series, I split the data chronologically rather than randomly since the training set should contain data for dates before those in the test set.
  - 13 folds of 28 consecutive days each (364 days total for testing), and iteratively add each previously-evaluated fold to the training set when evaluating next fold
  - Test set: the last 364 days in the dataset from June 3, 2024 to June 1, 2025 (split into 13 folds for evaluation)
  - Validation set (for tuning): the 364 days from June 5, 2023 to June 2, 2024

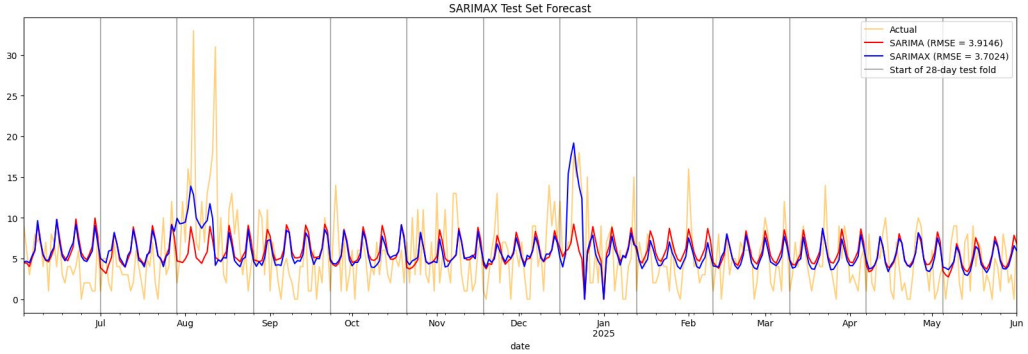
Holt-Winters (Triple Exponential Smoothing)

- Applies **exponential smoothing** three times and breaks the time series down into three components: **level**, **trend**, and **seasonality**.
- Pros: works well on data with clear trends and seasonality, is relatively fast
- Cons: can be too simple, assumes linear trends and seasonality, does not support multiple seasonalities or external regressors
- **Ensemble** model using the average of two H-W models with 7 and 364 seasonal periods provides more balanced predictions and improved performance



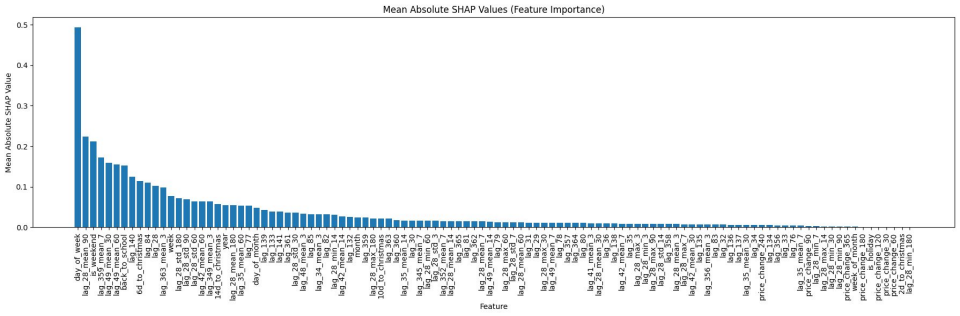
SARIMAX (Seasonal ARIMA with Exogenous Regressors

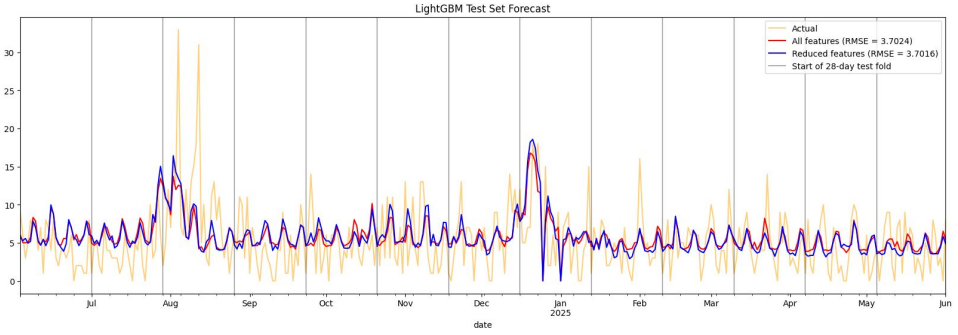
- **ARIMA (Autoregressive Integrated Moving Average)** models the time series as a linear combination of its past values, past forecast errors, and differenced values. **SARIMAX** adds support for seasonality and exogenous regressors.
- Assumes linearity and one seasonal period
- Much more computationally expensive compare to the other methods
- I included three regressors: back_to_school, 6d_to_christmas, and lag_363_mean_3



LightGBM (Light Gradient-Boosting Machine)

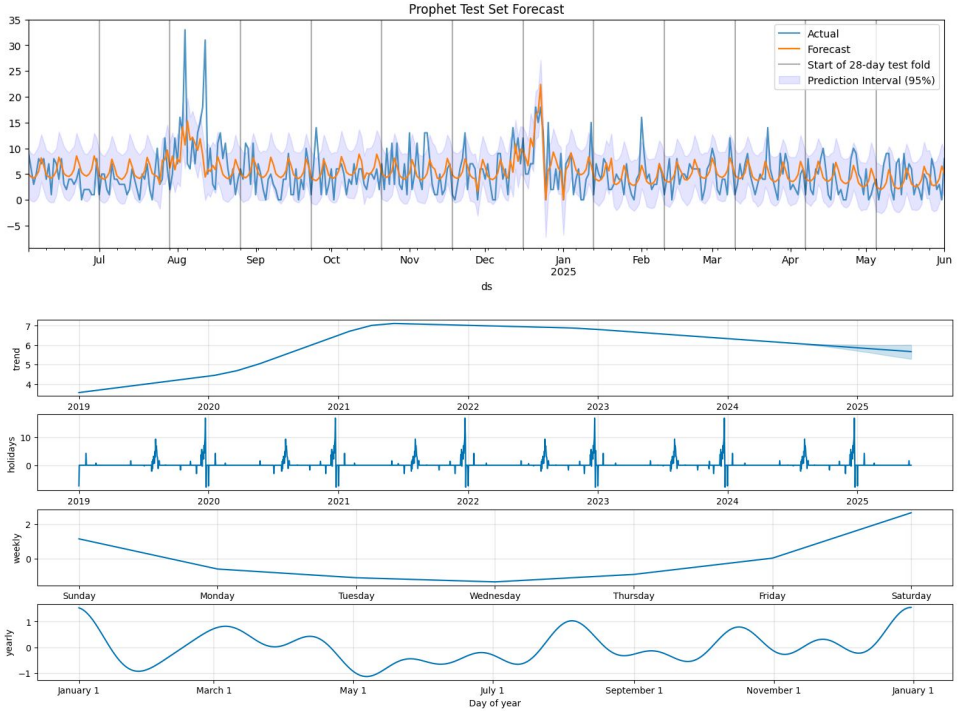
- **Tree-based** ML algorithm with **gradient-boosting** framework.
- Can handle nonlinearities and multiple seasonalities (via feature engineering, and is also relatively fast
- Not a time series model in itself, so it **requires extensive feature engineering**
- Cannot extrapolate trends, and accuracy of forecasts can degrade quickly over long horizons
- Can remove unimportant features to potentially reduce overfitting
  - Remove features with low importance values (such as **SHAP values**)
  - In this case, slightly improved performance and faster fit time





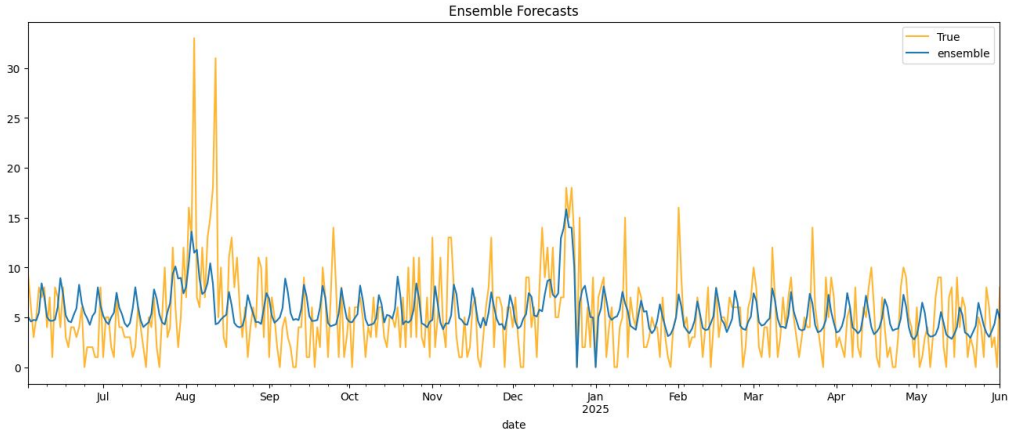
Prophet (by Meta/Facebook)

- Based on an additive model that fits non-linear trends with multiple seasonalities as well as holiday effects.
  - More accurately, models semi-nonlinear trends through piecewise linear or logistic functions.
  - Can also take regressors
  - Runs efficiently especially for small to medium-sized datasets
  - Has a relatively simple API
- Does not model autocorrelation
  - While it models trend, seasonality, and specified holiday effects, it does not assume dependencies with past values
- Performed better than other models, possibly due to identifying certain holiday and seasonal effects better



Ensemble Average Model

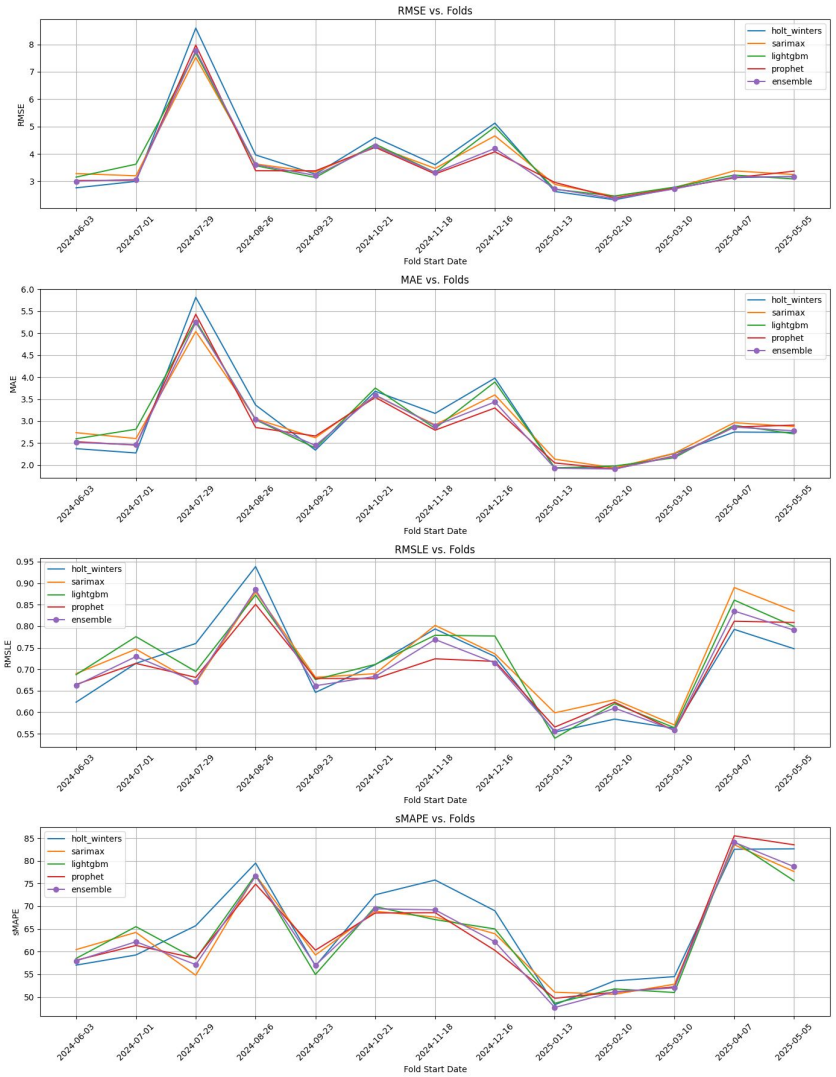
- An ensemble model was built by simply taking the average of the best models for each of the four methods
  - Alternative ensemble methods include weighted averaging and stacking (the predictions are fed into a “meta learner”)
- Generally resulted in better performance



	holt_winters	sarimax	lightgbm	prophet	ensemble
RMSE	3.7633	3.7024	3.7016	3.6152	3.5843
MAE	2.9712	2.9481	2.9455	2.8855	2.8705
RMSLE	0.7043	0.7243	0.7198	0.6981	0.7020
sMAPE	65.9368	63.9775	63.6657	64.0489	63.4908

Discussion

- **Prophet** was the best single model, while **LightGBM** was second best but also fastest.
- **Holt-Winters** was too simple
- **SARIMAX** was comparable to LightGBM due to regressors, but was too slow
- **Ensembling** improved performance
- Evaluating at a per-fold level, each model had the potential to do better (or worse) than the others.
- Model performance seemed to largely depend on how well seasonal patterns are captured.
- Values at a daily level are too noisy to get good forecasts, so working with 1- or 2-week intervals may provide more usable results



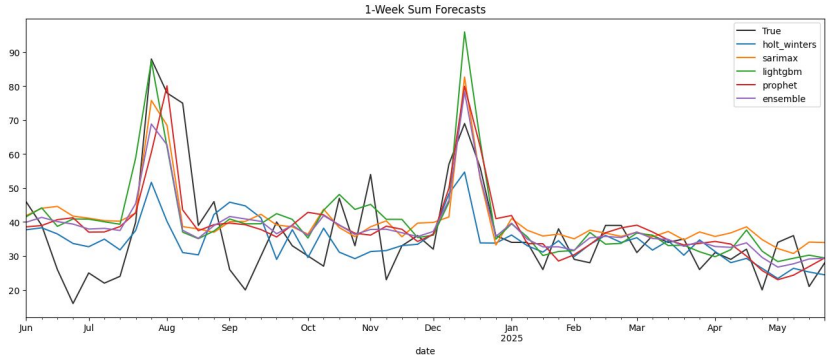
Discussion

- Limitations of dataset
  - Variability at daily level compounded by smaller dataset size makes daily forecasts difficult.
  - Working at lower granularities (e.g. 1- or 2-week intervals) may provide more usable results
  - Additional historical data may be necessary (exact dates for first day of school, dates of manufacturer shortages and relevant social media posts, etc.)
- Possible Improvements
  - Different ensembling methods, additional feature engineering, further hyperparameter tuning
  - Other Approaches: TBATs, LSTMs, CNNs, NeuralProphet

Statistics and metrics for 1-week sums

	count	mean	std	min	25%	50%	75%	max
true	26.0	73.307692	26.550735	42.0	60.0	69.5	76.0	166.0

	holt_winters	sarimax	lightgbm	prophet	ensemble
RMSE	22.9800	18.2882	19.0144	16.8006	17.2540
MAE	14.5731	13.8939	14.0706	12.7317	12.2769
RMSLE	0.2668	0.2615	0.2543	0.2375	0.2364
sMAPE	18.5924	19.1434	18.5651	17.4486	16.7887



Conclusion

- Takeaways
  - Even a dataset much smaller and limited in information than those of large corporations may still be viable for trend and seasonality analysis as well as forecasting
  - Extensive feature engineering backed by domain knowledge is especially crucial
- Future Works
  - Weekly forecasting is promising and can be sufficient for real-world applications
    - Can possibly still use daily counts as features for predicting weekly counts
  - Can extend the problem to different product levels and groups (different sizes/colors, between products, by categories, etc.) or for another store.
    - Including data for similar products could also be help forecast sales for the item in question