M5 Forecasting – Predicting Wallmart Sales

Cezary Gruszka

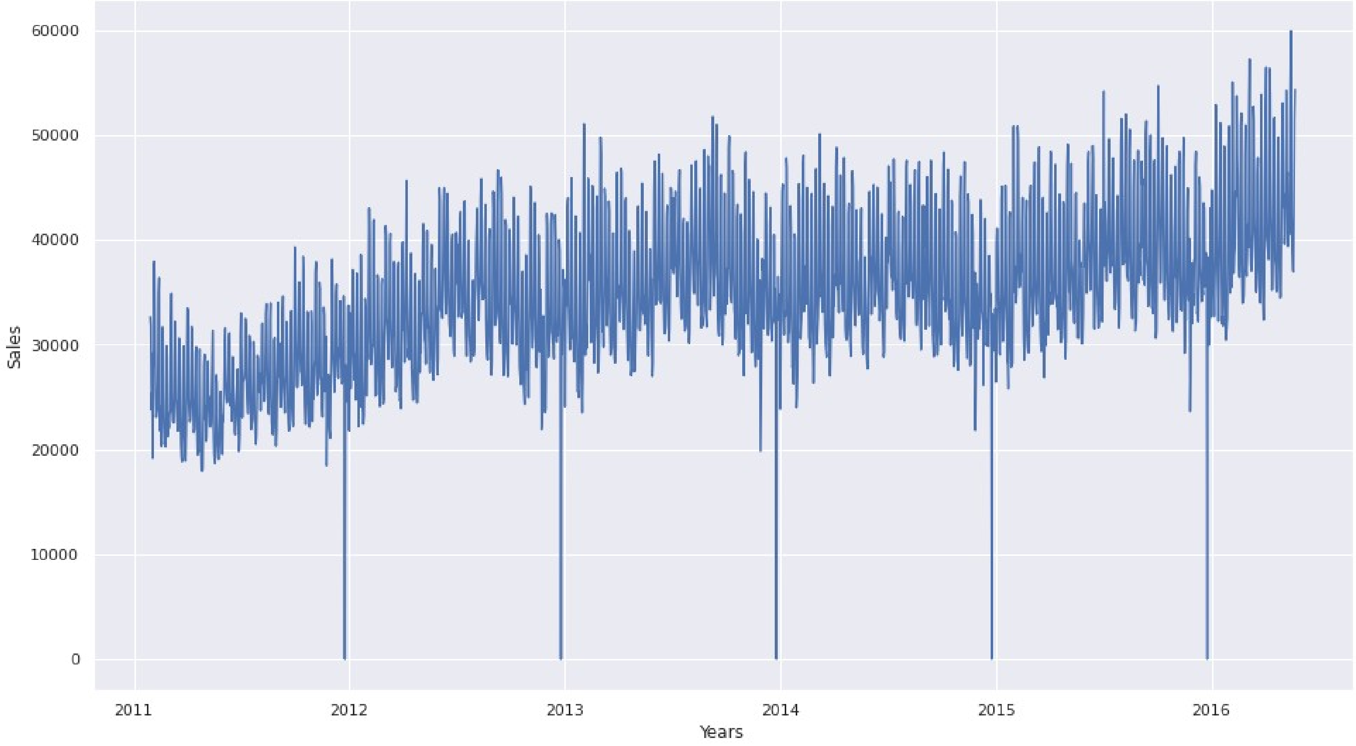
Problem and Target

The competition gives us the goal of predicting the amount of items sold in Wallmart stores. To make our prediction, we are provided with historical data of:

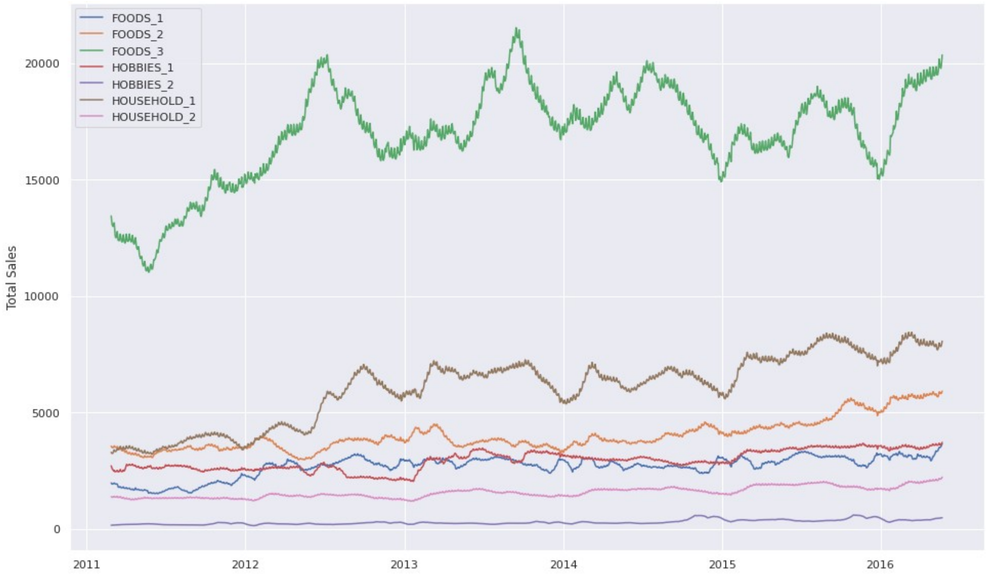
- Calendar data outlining any events, cultural, religious or othwerise
- Sales data of each product in each store on each day
- Product prices for each week

Dataset

- Sales fluctuate often
- Clear upwards trend
- A lot of noise
- Large amplitude



Breaking down the fluctuations



Sadly, even though the proportion of each product type is relatively stable with one noteable exception, our sales are still very unstable, even on a store basis. This means that the time component is crucial in our predictions.

Model Evaluation

- Given the competitive nature of the contest, we are only concerned with a single metric: the Weighted Root Mean Squared Scaled Error. This score determines the placings for the contest.

$$RMSSE = \sqrt{\frac{1}{h} \frac{\sum_{t=n+1}^{n+h} \left(Y_t - \widehat{Y}_t\right)^2}{\frac{1}{n-1} \sum_{t=2}^{n} \left(Y_t - Y_{t-1}\right)^2}}$$

$$WRMSSE = \sum_{i=1}^{42,840} w_i * RMSSE$$

Model 1 – Linear Regression

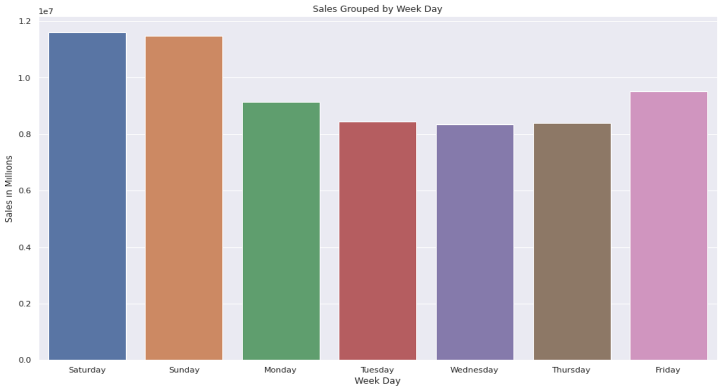
The idea behind this model is simple: we ignore the noise. Even though our data is very unstable, it showed a very clear and simple upwards trend.

```
rmse_train: 3.5637158109429197, rmse_test: 3.5780065258685196
```

Underwhelming. Could work for long term trends somewhat, but 28 days is far too small a timeframe for this kind of model.

Model 2 – Random Forest

Our hope is this model manages to identifiy events that simple sharp changes in our data. Like the impact of weekends.




```
rmse_train: 3.1610962650355, rmse_test: 3.248270722756284
```

… yeah no. Our data is too densely packed, and events are not impactful enough. This calls for a more sophisticated model.

Model 3 -LightGBM

- LGBM is a fantastic model for time series forecasting. It does however require proper tuning for great results. As such we will introduce two new special features: lags and rolling means.

```
train.columns
```

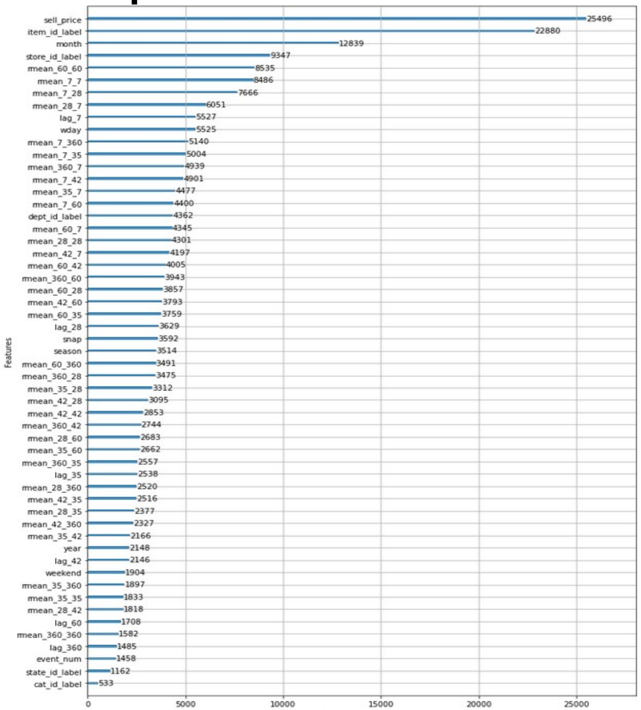
```
Index(['index', 'id', 'item_id', 'dept_id', 'cat_id', 'store_id', 'state_id',
       'wm_yr_wk', 'wday', 'month', 'year', 'items_sold', 'sell_price',
       'event_num', 'snap', 'weekend', 'season', 'item_id_label',
       'dept_id_label', 'cat_id_label', 'store_id_label', 'state_id_label',
       'lag_7', 'lag_28', 'lag_35', 'lag_42', 'lag_60', 'lag_360', 'rmean_7_7',
       'rmean_28_7', 'rmean_35_7', 'rmean_42_7', 'rmean_60_7', 'rmean_360_7',
       'rmean_7_28', 'rmean_28_28', 'rmean_35_28', 'rmean_42_28',
       'rmean_60_28', 'rmean_360_28', 'rmean_7_35', 'rmean_28_35',
       'rmean_35_35', 'rmean_42_35', 'rmean_60_35', 'rmean_360_35',
       'rmean_7_42', 'rmean_28_42', 'rmean_35_42', 'rmean_42_42',
       'rmean_60_42', 'rmean_360_42', 'rmean_7_60', 'rmean_28_60',
       'rmean_35_60', 'rmean_42_60', 'rmean_60_60', 'rmean_360_60',
       'rmean_7_360', 'rmean_28_360', 'rmean_35_360', 'rmean_42_360',
       'rmean_60_360', 'rmean_360_360'],
      dtype='object')
```

23385830 rows × 64 columns

Our train set is becoming gigantic.

Feature importance

Our model is very happy with the new features and incorporates a lot of them into our final predictions.



Final prediction

The model, unsurprisingly, vastly outperformed the other two:

- It incorporates specific time shifts very well (say week or month)
- Lags and rolling features work great for time series forecasting
- Cross validation was far more effective

```
rmse_train: 1.9160976387384971, rmse_test: 2.061217187706627
```

These values suggest that the model's prediction should be competitive on the competition leaderboard