Peak Traffic Volume Forecasting

1. Introduction

Predicting traffic volume of shop webpages is important for providing a better online shopping experience, as webpages' traffic volume may surge to an extremely high level during big promotion events. This project focuses on Peak Traffic Volume ('PTV') forecasting based on the 'PTV' for a shop in Taiwan during 2017.

In this report, I explain the preprocessing I did with the dataset, how I got the features and how to select the features. Regression model was made, so that forecast about 'PTV' can be made on a certain day, given the date and promotion events on that day.

2. Data and Preprocessing (preprocess.py)

To understand the data, I plotted the data to get some insight, and corrected the wrong data.

1) PTV_TTV.csv

'PTV' is a time series. Time series is time dependent and may have trend or seasonality. Since most of the time series models work on the assumption that the time series is stationary, I first checked whether 'PTV' is a stationary time series. As is shown in Figure 1, it is clear from the rolling statistics, 'PTV' doesn't have a constant mean, thus is not a stationary time series.

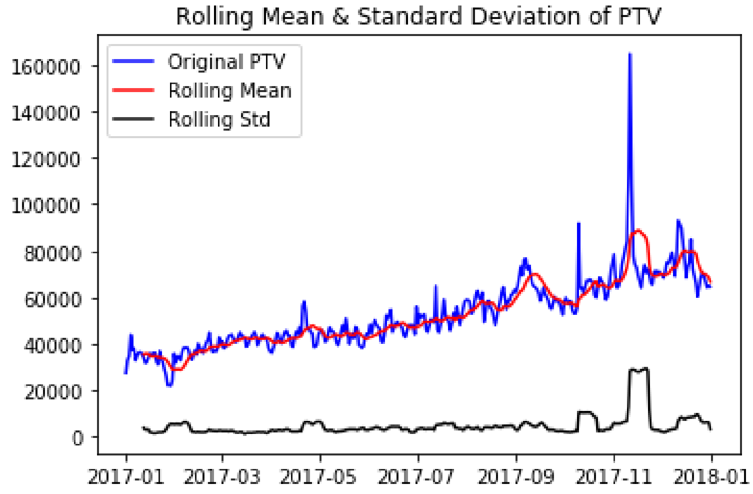

Figure 1 Rolling Mean & Standard Deviation of 'PTV'

Besides, I also ran Dickey-Fuller Test on 'PTV'. From Table 1, I see that given the null hypothesis being "'PTV' is a non-stationary process", the p-value and t-statics suggest that the null hypothesis is true.

Table 1 Dickey-Fuller Test of 'PTV'

Test Statistic	-2.075066
p-value	0.254658
#Lags Used	6.000000
Number of Observations Used	358.000000
Critical Value (1%)	-3.448749
Critical Value (5%)	-2.869647
Critical Value (10%)	-2.571089

Therefore, I concluded that 'PTV' is a non-stationary process, and it's clear from Figure 1 that there's a trend in 'PTV'.

2) promotions.csv

I found the following wrong data:

- Negative discount percent: I corrected them by resetting them to zero. (id: 499,3413,3739,4112,4258,4343,4379,4766,5012)

3) voucher.csv

I did the following steps:

- 'end_time'<'start_time': drop this item. (id: 10848,10850,36765,36865,37053,45472,54695,64284,80989,85323,100037,108577,114286,116747,125367,146297,147564)
- 'end_time' after 2018: I reset them to 2018-1-1 to make it easier for following coding
- Some items have the same 'start_time', 'end_time', 'discount', 'min_price', 'value', only 'usage limit' are different. I treaded them as the same voucher, and combine them with 'usage limit' added together.

3. Features (features.py)

I found the possible features and combine them in a single dataframe data named 'df'. Here I describe how to get the columns of 'df' from the given datasets.

When time is considered, I assume that people sleep during 00:00-07:00, thus promotion or voucher settings during this period are not important. Also, the promotion event of the previous day may have effect on the current day, so I form features to cover this part.

1) PTV_TTV.csv

- 'data' and 'PTV': 'date' and 'PTV' in 'PTV_TTV.csv'
- 'day': how many day is the date away from 2017-1-1.
- 'weekday': the day of the week of the date
- 'monthday': the day of the month of the date

2) promotions.csv

Current day:

- ‘promotion_price’: average promotion price weighted by promotion hours on a certain day
- ‘promotion_rebate’: average promotion rebate weighted by promotion hours on a certain day
- ‘promotion_discount’: average promotion discount weighted by promotion hours on a certain day
- ‘promotion_item’: the total number of promotion items on a certain day
- ‘promotion_hour’: the total number of promotion hours of every promoted items on a certain day

Previous day:

- ‘promotion_price_l’: average promotion price weighted by promotion hours on the previous day
- ‘promotion_rebate_l’: average promotion rebate weighted by promotion hours on the previous day ‘promotion_discount_l’: average promotion discount weighted by promotion hours on the previous day
- ‘promotion_item_l’: the total number of promotion items on the previous day ‘promotion_hour_l’: the total number of promotion hours of every promoted items hours on the previous day

3) voucher.csv

Current day:

- ‘voucher_value’: average voucher value weighted by promotion hours on a certain day
- ‘voucher_min’: average voucher min_price weighted by promotion hours on a certain day
- ‘voucher_discount’: average voucher discount weighted by promotion hours on a certain day
- ‘voucher_total’: average voucher usage limit weighted by promotion hours on a certain day
- ‘voucher_number’: the total number of vouchers on a certain day
- ‘voucher_hour’: the total number of voucher hours on a certain day
- ‘voucher_value_n’: average voucher value weighted by usage limit on a certain day
- ‘voucher_min_n’: average voucher min_price weighted by usage limit on a certain day
- ‘voucher_discount_n’: average voucher discount weighted by usage limit on a certain day

Previous day:

- ‘voucher_value_l’: average voucher value weighted by promotion hours on the previous day
- ‘voucher_min_l’: average voucher min_price weighted by promotion hours on the previous day

- ‘voucher_discount_l’: average voucher discount weighted by promotion hours on the previous day
- ‘voucher_total_l’: average voucher usage limit weighted by promotion hours on the previous day
- ‘voucher_number_l’: the total number of vouchers on the previous day
- ‘voucher_hour_l’: the total number of voucher hours on the previous day
- ‘voucher_value_n_l’: average voucher value weighted by usage limit on the previous day
- ‘voucher_min_n_l’: average voucher min_price weighted by usage limit on the previous day
- ‘voucher_discount_n_l’: average voucher discount weighted by usage limit on the previous day

4) Other factors: recorded in ‘event’ feature
   - Holidays: All public holidays excluding Chinese New Year, Dragon Boat Festival, and Mid-Autumn Festival
   - Sale season: 11.11 and 12.12
   - delivery stop: Chinese new year

4. Feature Selection and Regression Model (feature_selection.py & regression.py)

This part will introduce the regressors I used, and how I did the feature selection.

1) Estimating and eliminating trend

   It is obvious from Figure 1 that ‘TPV’ has a linear trend, so I used linear regression to estimate the trend with ‘day’, and eliminate the trend from ‘PTV to get a ‘Detrended PTV’. I plotted the rolling statistics of the ‘Detrended PTV’ as shown in Figure 2.

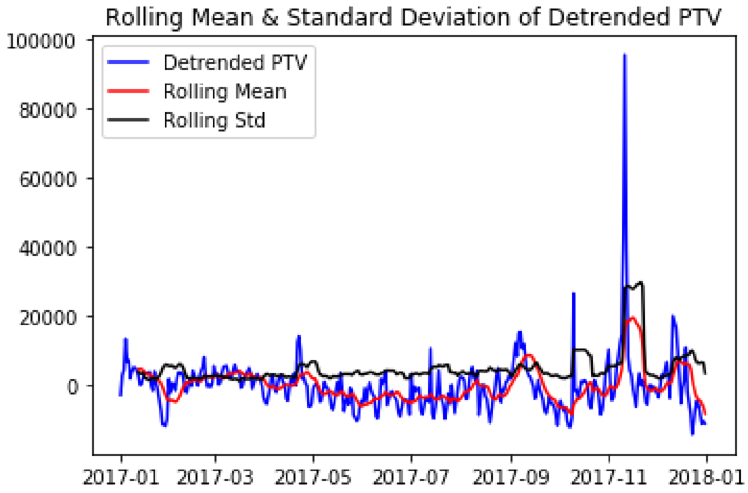

Figure 2 Rolling Mean & Standard Deviation of ‘Detrended PTV’

I also did Dicky-Fuller Test on ‘Detrended PTV’. From Table 2, I see that given the null hypothesis being “‘PTV’ is a non-stationary process”, the p-value and t-statics suggest that the null hypothesis should be rejected. Therefore, the ‘Detrended PTV’ is a stationary process.

Table 2 Dickey-Fuller Test of ‘Detrended PTV’

Test Statistic	-7.133892
p-value	3.460943e-10
#Lags Used	2.000000
Number of Observations Used	360.0000
Critical Value (1%)	-3.448544
Critical Value (5%)	-2.869557
Critical Value (10%)	-2.571041

2) 10-fold cross validation.
   I used 10-fold cross validation, with MSE criteria to select regression model, select parameters for the models and do feature selection.
3) Select regression model.
   I found that ensemble regressors generally perform well, and among them Extra-Tree Regressor performs the best. So I used Extra-Tree Regressor as a base regressor, and then fit the remaining errors with Polynomial Regressor.
4) Find best parameters for regressors to avoid overfitting and underfitting.
   - Extra-Tree Regressor
     Find ‘max_leaf_nodes’. I chose ‘max_leaf_nodes’=7, since is shown in Figure 3.
   - Polynomial Regressor
     In a similar way, I chose ‘degree=2’.
5) Feature Selection
   Find the best number of features with the highest ‘feature importance’.
   - Extra-Tree Regressor
     According to the result in Figure 4, I chose the 12 best features as the input to Extra-Tree Regressor. The selected features (with importance from the highest to the lowest) are: ['event_l', 'event', 'promotion_item', 'weekday' , 'voucher_min_n', 'monthday', 'promotion_hour', 'voucher_number', 'voucher_value_n' , 'voucher_discount_l', 'voucher_number_l', 'voucher_total_l']
   - Polynomial Regressor
     In a similar way, I chose the features to be ['event', 'promotion_item']

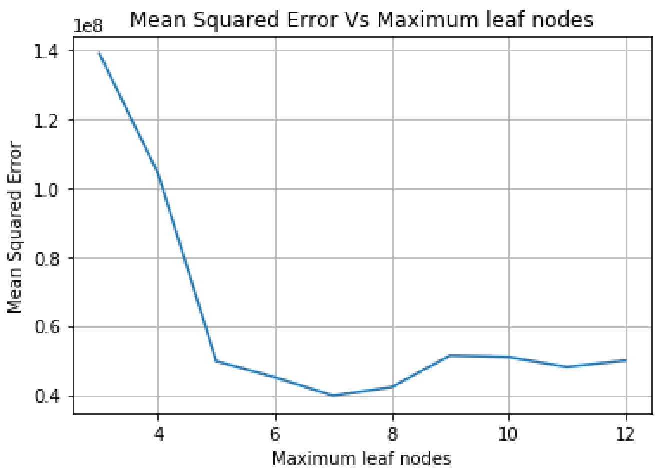

Figure 3 Mean squared error with 10-fold cross validation Vs changing maximum leaf nodes of Extra-Tree Regressor

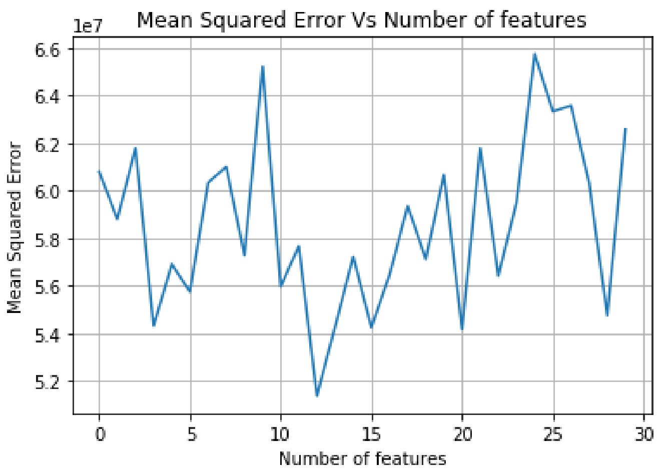

Figure 4 Mean squared error with 10-fold cross validation with changing numbers of features. Features are selected with the highest feature importance

5. Results

1) Prediction Model

   Prediction Model is the sum of 3 regressors.

   - **Linear Regressor**, the input is ‘day’.
   - **Extra-Tree Regressor**, parameter setting is ‘max_leaf_nodes’=7, input is ['event_l', 'event', 'promotion_item', 'weekday' , 'voucher_min_n', 'monthday', 'promotion_hour', 'voucher_number', 'voucher_value_n' , 'voucher_discount_l', 'voucher_number_l', 'voucher_total_l']
   - **Polynomial Regressor**, parameter setting is ‘degree=2’, input is ['event', 'promotion_item']

2) Result

With 10-fold cross validation, Figure 5 shows the predicted ‘PTV’ and the true ‘PTV’. The mean square error of the prediction is 3.98×10^7.

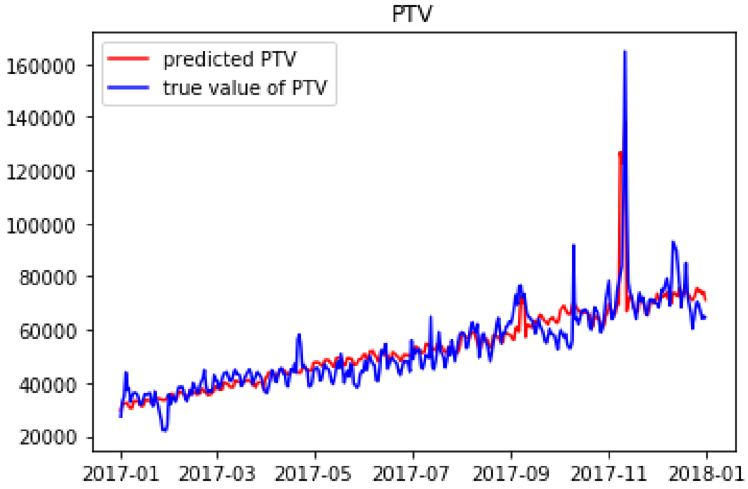

Figure 5 Predicted and True Value of ‘PTV’ with 10-fold Cross Validation

6. Interpretation

- There’s a steady increase in PTV with the time.
- From Figure 6, we can see that ‘event’ (public holidays, sale seasons, etc.) is the most important factor. By applying the model, we can see that even if this shop has no promotion event during public holidays, the PTV will still increase.
- The number of promoted items and the number of vouchers influence PTV much more than the discount or refund offered by promotion and vouchers.

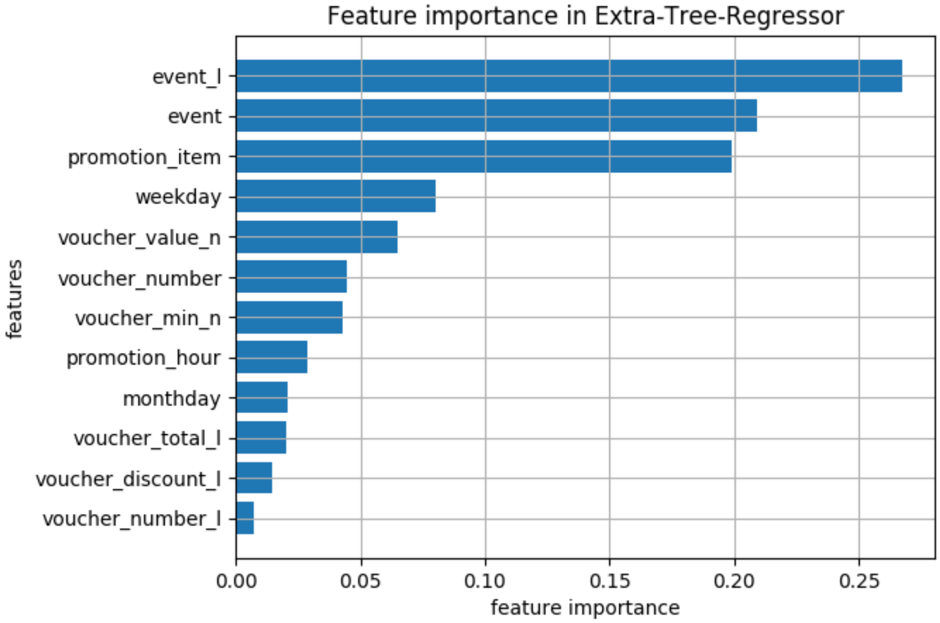

Figure 6 Feature Importance in Extra-Tree-Regressor