Super Cue Boba Shop

Boba shops have grown exponentially in the bay area since the first boba shop opened in 1997. There are now countless boba shops through out the bay area, and the competition has become very intense for boba shop owners. In this project, I will use timeseries analysis to find out the optimal store hours for Super Cue Cafe's specific store location.

1. Data

Super Cue cafe has started to use Clover POS (point of sales) system since 2013. And this modern POS system stores numerous information, and contains free and pay to access apps for the owners.

As the ex-owner of Super Cue Cafes. I have kept hourly sales of my store locations for a little more than 2 years. Which, I manually gathered the data from the clover's analytics app, and put them into excel files to do analysis as a small business owner.

2. Method

Since the data consist of only time and sales, it is certainly a univariable data, and is defintely best suited as a time series model. However, most of the time series dataset have seasonalities, here are the models that works well with time series dataset that has multiple seasonalities:

1. **SARIMAX:** Seasonal AutoRegressive Integrated Moving Averages, Autoregression explores any dependent relationship between an observation and some number of lagged variables. Integrated to make time-series stationery by subtracting or differencing an observation from observation at the previous time step of the same time series. Moving Average explores the relationship between an observation and a residual error by application of moving average to lagged observations, with any given time window. X for exogenous variables. This model is well suited for data sets with multiple seasonalities, and with more than just time as independent variable.

2. **Auto_arima:** Automatically discover the optimal order for an ARIMA model, so it is similar to SARIMAX, but it does not depend on the PACF/Auto-Correlation (manual computation of differencing), but instead, it conducts differencing tests (i.e., Kwiatkowski–Phillips–Schmidt–Shin, Augmented Dickey-Fuller or Phillips–Perron) on its own to determine the order of differencing.

3. **TBATS:** T for trigonometric regressors to model multiple-seasonalities, B for Box-Cox transformations, A for ARMA errors, T for trend, S for seasonality; another model well suited for data sets with multiple seasonalities.

3. Data Cleaning

Data Wrangling

As I mentioned earlier, the data I personally collected consists of time (date and hour) and sales.

Problems that I had to solve for each sheet

- **Problem 1:** I begin by importing one sheet (1 month of data), and realized there are additional week each month, where that additional week is the month's hourly average by day of week and hour of day.
- **Problem 2:** Even though columns are Day of Weeks, there's an additional column that's the Average. ANd there are also rows that's the sum of the day's AM sales, PM sales and Total sales.
- **Problem 3:** The format isn't time series ready, since the column of the data set is Day of week, while index consists of date (e.g. 01-01-2000) and time (e.g. 11:00).
- **Problem 4:** Sometimes a sheet can have 4 or 5 weeks of data due to usually 4 weeks of data per sheet, while the accumulated dates may result in a month with 5 weeks of data once per 3 to 4 months.
- **Solution:** I defined a function that will clean the sheet with problems mentioned above by: remove empty rows, remove the unwanted columns/rows, break downeach weeks data to such way then combine all the weeks (except last week, since its' the average) data to one. Transpose it, so the data set has index as date and column as hour of day.

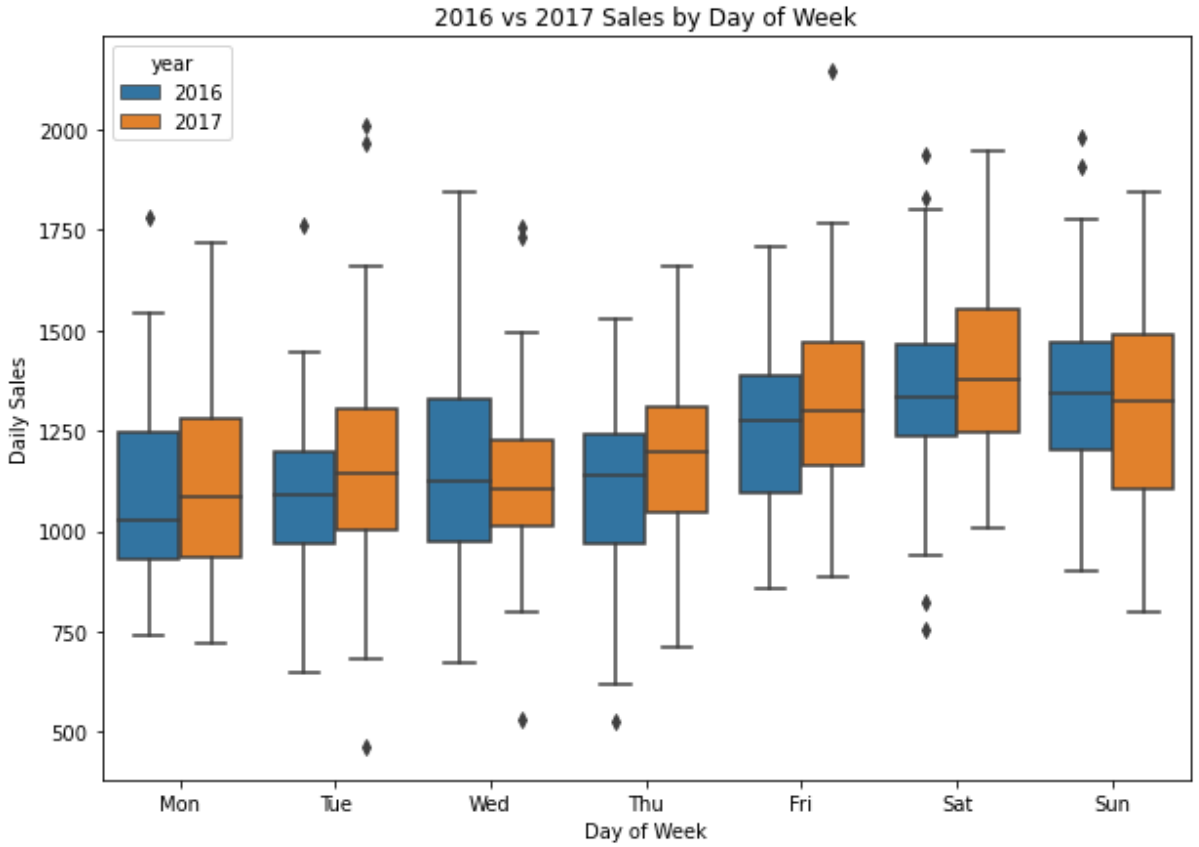
Problems after cleaning each sheet

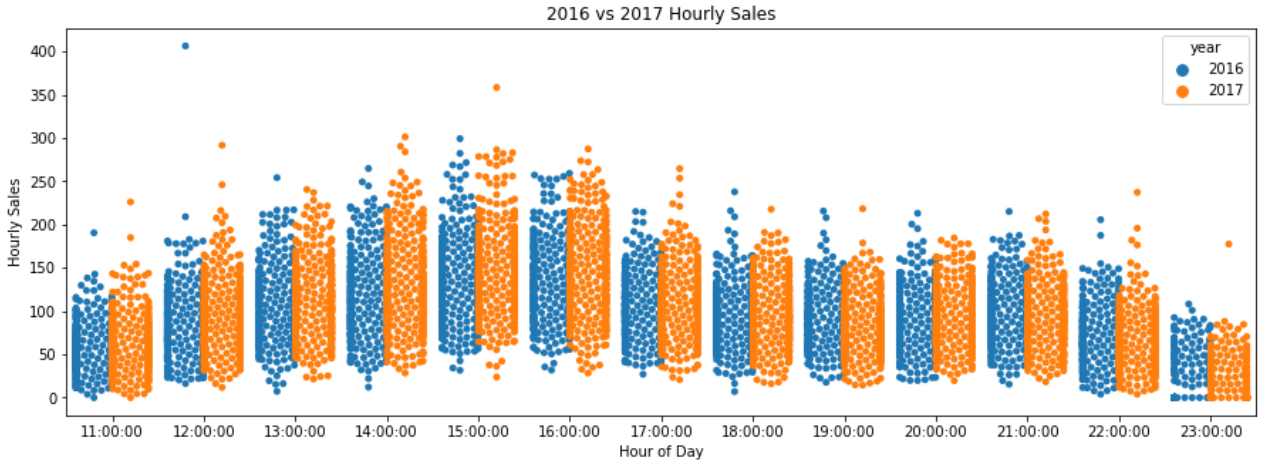
- **Problem 1:** There are missing values or outliers
- **Problem 2:** Need to make them into 1 big dataframe or a dataframe per year, such that the index is time (date and time!) and column is sales for that hour.
- **Solution:** I concat all the datasheets to one dataframe, then checked its standard deviation to find out the outliers and replace them and appropriate hours with that year's average. Then I used this data set where the index is date, and column is hours and Daily sum, to create a Dataframe that's more suitable for time series, where index = date and time while column is sales of that hour; I filled in 0 for hours that the store was closed.

4. EDA

EDA

- By comparing the sales between 2016 and 2017 in day of week and hour of day, we can see that both of the years have similar results, suggesting daily and weekly seasonality.





5. Algorithms & Machine Learning

Due to having multiple seasonalities, it may be a good idea to breakdown the datasets into different time frequencies. So I have two modeling sections

Modeling (day)

I tried models that aren't suitable for multi-seasonality datasets as well (ARIMA, SARIMA, Holtz). But ultimately, the models that were able to capture multiple seasonalities were: TBATS, SARIMAX, Auto_arima, which were the method mentioned earlier that are recommended for time series with multiple seasonalities.

	model	MSE	MAE	AIC	ForeMSE	ForeMAE	p	d	q	P	D	Q	fit_time(s)	fit_time_per_row(s)
450	AA_DM2FT	66073.422118	190.475741	NaN	21330.794984	113.681134	NaN	NaN	NaN	NaN	NaN	NaN	25.02	0.0413
134	SARIMAX	65903.360646	191.998003	8076.731427	22572.938282	116.913271	0.0	0.0	0.0	1.0	0.0	1.0	NaN	NaN
449	AA_DM1FT	65985.656542	192.208360	NaN	22569.104497	116.964918	NaN	NaN	NaN	NaN	NaN	NaN	28.45	0.0470
133	SARIMAX	65986.388282	192.240132	8088.224671	22565.935834	116.963609	0.0	0.0	0.0	1.0	0.0	0.0	NaN	NaN
146	SARIMAX	66056.501214	192.457675	7965.062534	22624.064638	117.031592	0.0	0.0	2.0	0.0	0.0	1.0	NaN	NaN

'Top 5 MAE scores'

	model	MSE	MAE	AIC	ForeMSE	ForeMAE	p	d	q	P	D	Q	fit_time(s)	fit_time_per_row(s)
451	AA_DM3FT	67322.726891	192.798886	NaN	21133.293238	112.767890	NaN	NaN	NaN	NaN	NaN	NaN	32.80	0.0541
452	AA_DM4FT	69428.451047	194.963525	NaN	21098.806273	112.940502	NaN	NaN	NaN	NaN	NaN	NaN	19.43	0.0321
450	AA_DM2FT	66073.422118	190.475741	NaN	21330.794984	113.681134	NaN	NaN	NaN	NaN	NaN	NaN	25.02	0.0413
84	ARIMAX	65873.074606	193.495806	8029.195843	22576.327364	116.106628	5.0	0.0	3.0	NaN	NaN	NaN	NaN	NaN
393	SARIMAX	65873.074606	193.495806	8029.195843	22576.327364	116.106628	5.0	0.0	3.0	0.0	0.0	0.0	NaN	NaN

'Top 5 ForeMAE scores'

> *__NOTE:__ I choose ForeMAE as the accuracy metric over other metric scoring, because the dataset isn't smoothed, and is likely to have outliers (due to other variables not included in data, such as temperature and holiday) and RMSE score will be influenced more than MAE score from outliers. Then ForeMAE is calculated with entire year as testing data, and thus would capture the yearly seasonality much better than MAE which only accounted for one season (3 months) of testing data.

Modeling (hour)

From the Model (day) experiences, TBATS and auto_arima had the best results, so the modeling for hourly data focused on trying different exogenous variable settings for auto_arima. And using predictions from TBATS to multiply ratios calculated in different ways.

model	2017RMSE	2017MAE	fit_time_per_row
TBATSxR_2017	28.113221	15.361412	NaN
AA_TBATS2017	28.118901	15.366706	NaN
AA3X-hourly	28.208743	15.430962	1.2739
AA3R-hourly	28.459946	15.537691	0.5217
TBATS-hourly2017	28.538750	15.545161	0.0849
AA3N-hourly	28.721891	15.685231	0.5904

> *__NOTE:__ Due to having 24 times (24 hours a day) more data than daily sales, the fitting time became much longer, and concerning. And thus the fitting time became a big part for the scoring metric as well. Even though TBATSxR has N/A for its fitting time, it is definitely less time consuming than auto_arima models in both prepping (exog variables is needed for auto_arima, but not TBATS) and fitting the model.

WINNER: TBATSxR

This model uses the results from TBATS then multiply the reslt by a ratio calculated by another model or method, in this case, the ratio was calculated based on training data's average daily sales by month. And thus create a yearly seasonality effect based on month.

6. Which Dataset to choose and how long for training data?

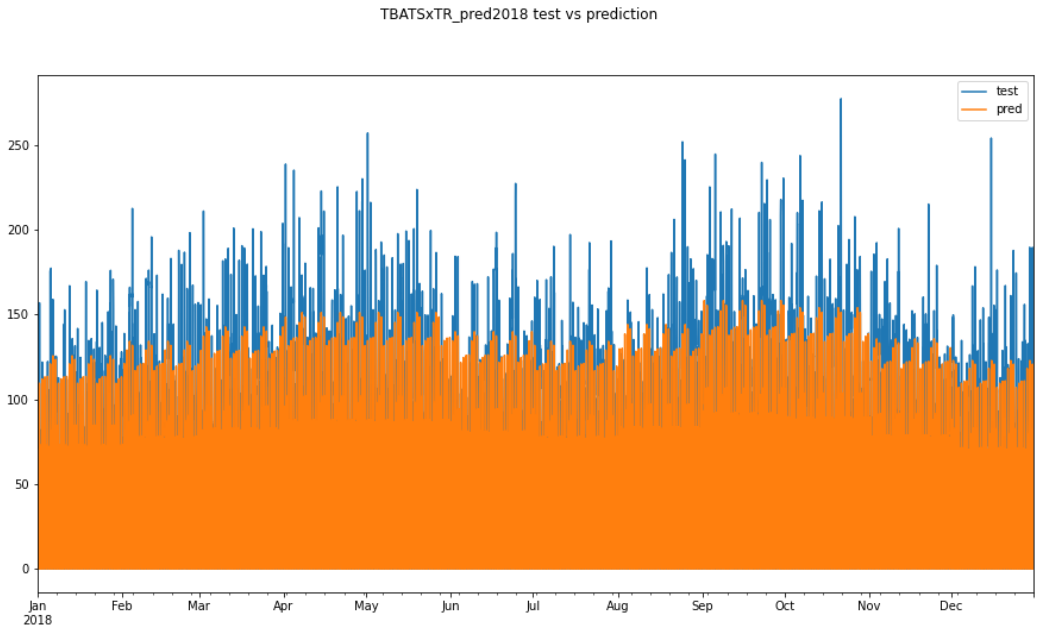
It is recommened to have 3 years or more of data for a proper time series analysis. Which trains the first 2 years and test on the 3rd year. However, we only have 2 years of data! Also, I have already tried several training/testing periods, because auto_arima with exog variables may consum a lot of memeory, the hourly models were mainly train=2016, and predict the 2017 to compare with the test. And from the result of training/testing various models, we have concluded earlier that TBATSxR is most suited for this project. And fortunately, TBATS doesn't take as much time and memory as auto_arima, so it was able to train the whole two years of data. But as for the missing 3rd year's data, I used the average of 2016 and 2017's data based on hour and day of week starting with Monday's 00:00. The result was much better than 2017's predictions, which had MAE of 15.xx.

	model	2018RMSE	2018MAE	fit_time_per_row
0	TBATSxTR_2018	20.712362	11.467182	N/A

7. Predictions

Finalized codes

In the finalized codes, it contains the codes on how I got the results for model(day) and model(hour), then the rest of it was used to make predictions for 2018. The following is the test vs prediciton for 2018.



8. Recommendation for Super Cue's store hours

- There were only two specific store hours where the sales is predicted to be having less than $40 in revenue, one of them is at 11:00AM, but I won't recommend to close during these hours, because the predicted value is very close to 40.

	pred
2018-01-01 11:00:00	39.429802
2018-01-08 11:00:00	39.431212
2018-01-15 11:00:00	39.431212
2018-01-22 11:00:00	39.431212
2018-01-29 11:00:00	39.431212
2018-12-03 11:00:00	38.545117
2018-12-10 11:00:00	38.545117
2018-12-17 11:00:00	38.545117
2018-12-24 11:00:00	38.545117

- The other predicted store hours where the revenue is less than 40 is at 11:00PM, which is on Friday and Saturday only, but as you can see from the counts (104), that's every accountable Friday & Saturday for 2018 (52 weeks per year times 2 days in a week), and with max predicted value of 13.05, I highly recommened the store to be closed at 11PM.

```
Recommend_closed.loc[(Recommend_closed.index.hour==23)].describe()
```

	pred
count	104.000000
mean	10.514056
std	1.389341
min	8.172995
25%	9.441219
50%	10.239730
75%	11.597632
max	13.054621