COME VISIT AGAIN

BY Team 2D

DEEPAK NANDWANI
(MT2021037)
DIKSHA SINHA
(MT2021041)

UNDER SUPERVISION OF T.A. RACHIT YAGNIK AND T.A. SHOURABH PAYAL

INTRODUCTION To Data Provided

- Files Which we have been provided with are the following:

1. Date_info.csv
2. Chw_reserve.csv
3. Chw_store_info.csv
4. Yom_store_info.csv
5. Yom_reserve_info.csv
6. Train.csv
7. Test.csv
8. Sample_submission.csv

- Among all these files, Train.csv and Test.csv are self explanatory.
- These files contain store id and visit date in common.
- Visitors column is included in Train.csv which shows the number of visitors on a particular day in a restaurant whose id is represented by “chw_store_id”

Date_Info.csv (Date Features Analysis)

- We start our EDA from the date_info file provided as it would be quick and simple to gain intuition of time series we are provided with.

calendar_date	day_of_week	holiday_flg
2016-01-01	Friday	1
2016-01-02	Saturday	1
2016-01-03	Sunday	1
2016-01-04	Monday	0
2016-01-05	Tuesday	0

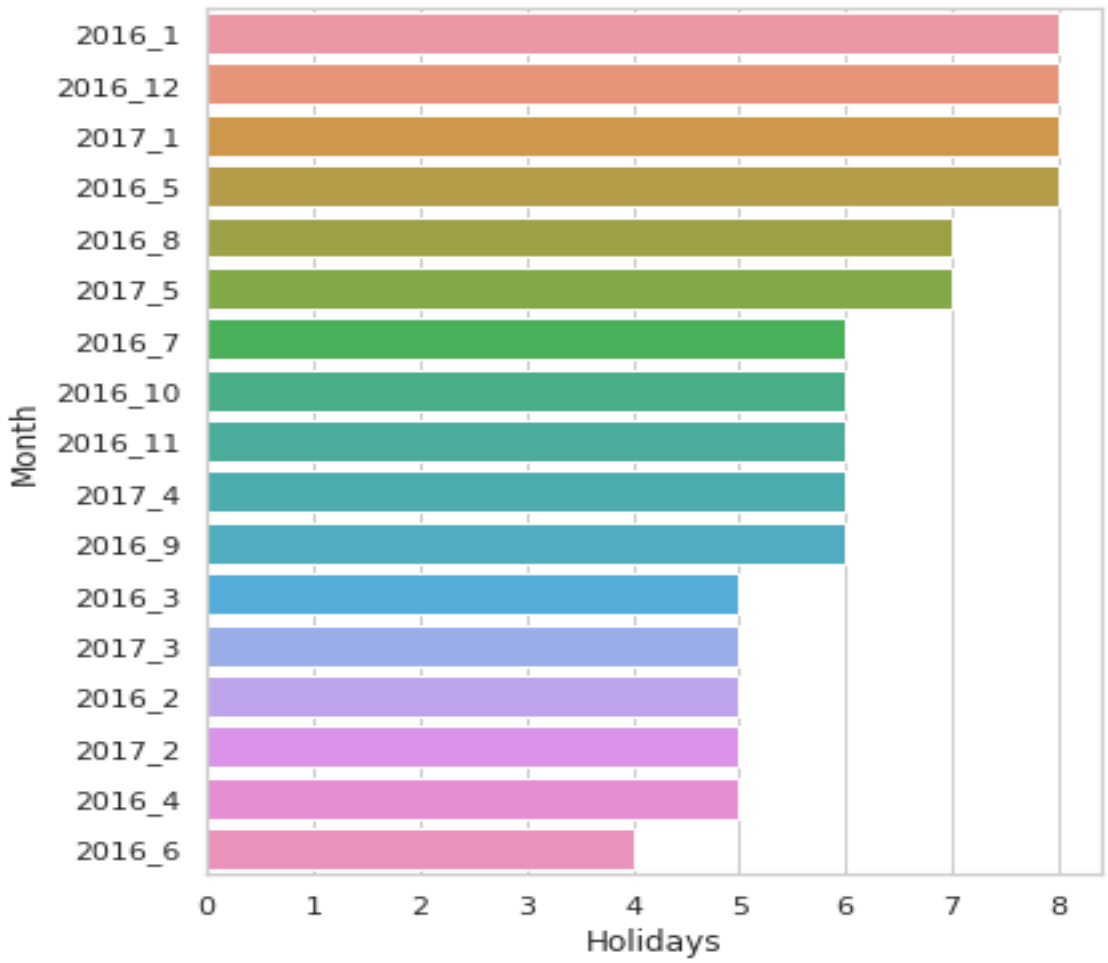
- As we see we have calender_date that’s the column using which we gonna merge the training data and date_info file.
- Day_of_week shows the day of the week and holiday_flg tells us weather there’s holiday on that particular day or not and also being the at the middle of our focus
- Below we do some pre-processing..

- First thing we do is some holiday analysis and of-course weekend plays an important role in predicting the visitors as we may have some surge on weekend and we gonna show it below as we move further down in document.

Weekend

```
date_info["week_end"] = 0
date_info.loc[date_info["day_of_week"]  == "Sunday", "holiday_flg"] = 1
date_info.loc[date_info["day_of_week"]  == "Friday", "week_end"] = 1
date_info.loc[date_info["day_of_week"]  == "Saturday","week_end"] = 1
date_info.columns  = ["date","week_day","holiday","week_end"]
```

- Number of holidays in a particular month, This has some correlation with visitors as if there are more holidays then so are visitors.



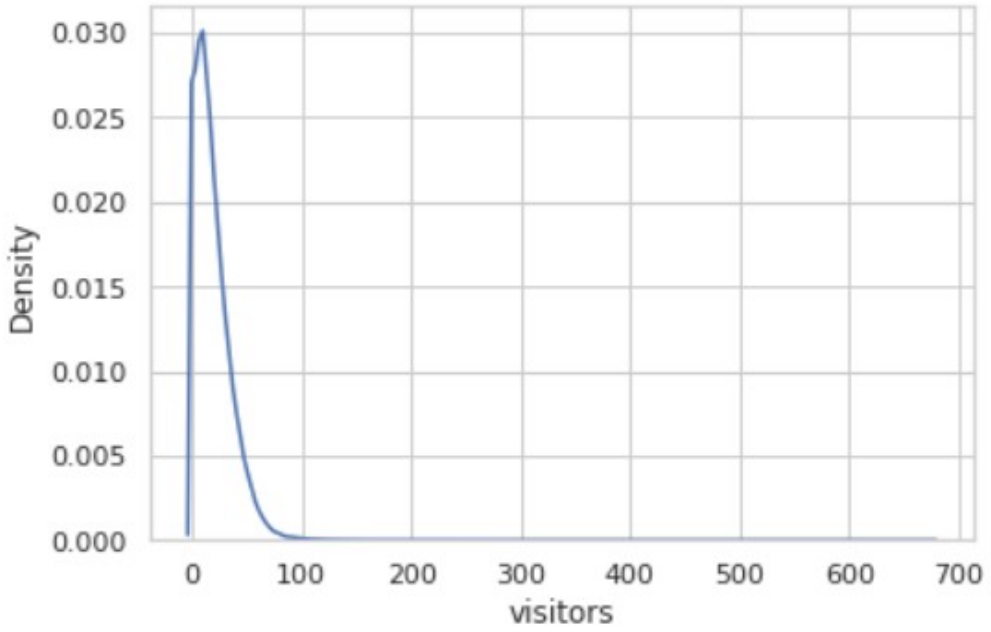
Apart from these we have also added : Week day, day, month and year by splitting date, so as we have

	date	week_day	holiday	week_end	week	month	year	day	days	month_holiday_agg	month_week_holiday_agg
0	1/1/2016	Friday	1	1	1	1	2016	1	31	8	3
1	1/2/2016	Saturday	1	1	1	1	2016	2	31	8	3
2	1/3/2016	Sunday	1	0	1	1	2016	3	31	8	3
3	1/4/2016	Monday	0	0	1	1	2016	4	31	8	3
4	1/5/2016	Tuesday	0	0	1	1	2016	5	31	8	3

Visitors outliear analysis

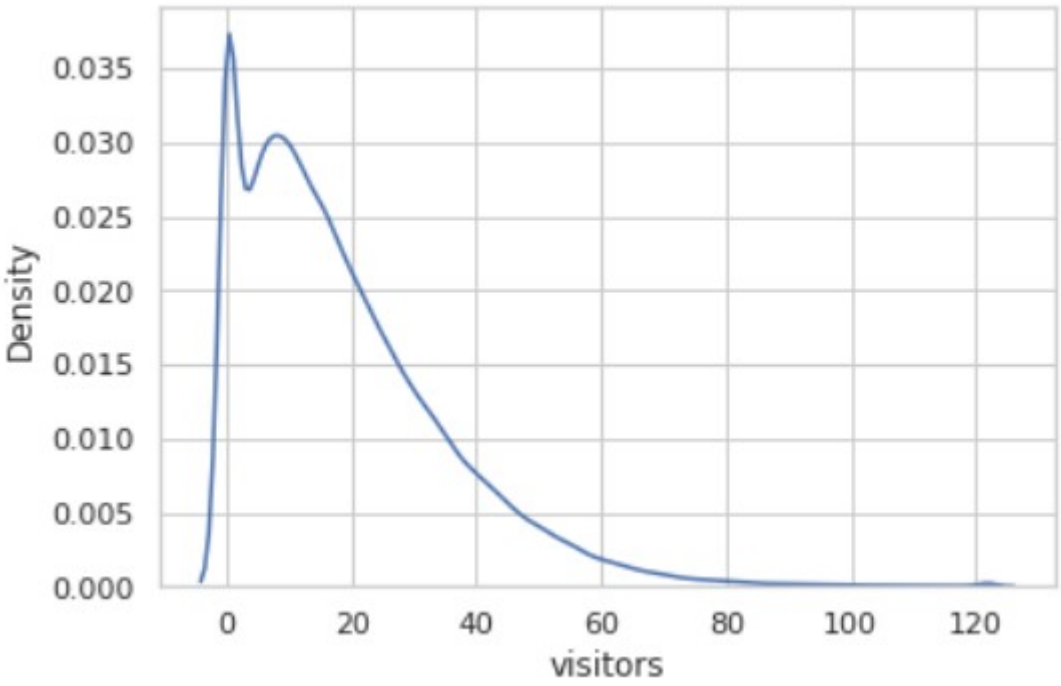
```
sns.kdeplot(data = all_data, x = "visitors")
```

```
<AxesSubplot:xlabel='visitors', ylabel='Density'>
```

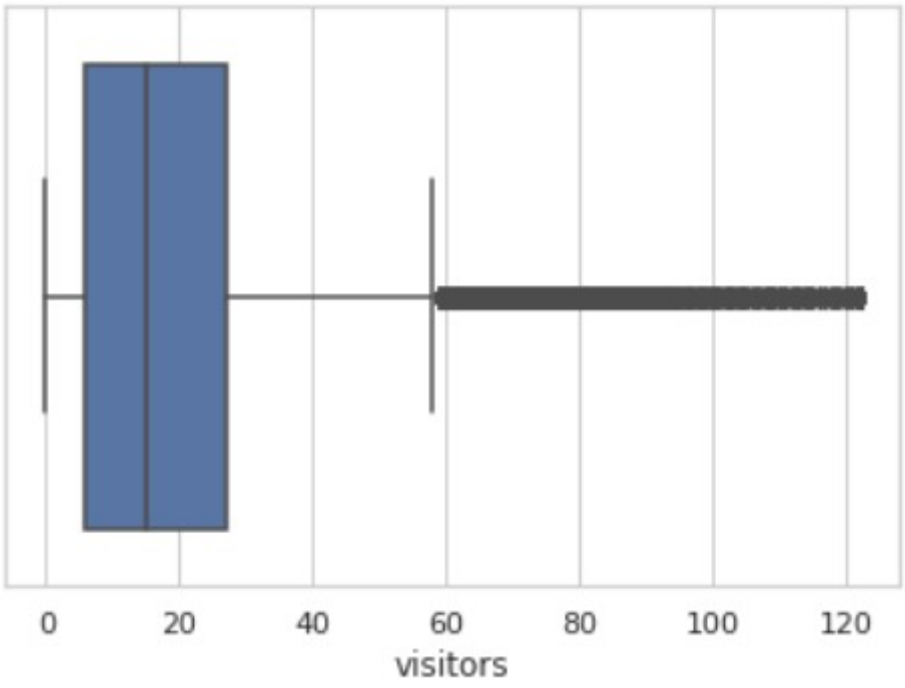


As we see there’s some skewness in data distribution of visitors so it’s not intuitive much so we gonna replace the values with more than 3*interquartile range + third quartile with this boundary value

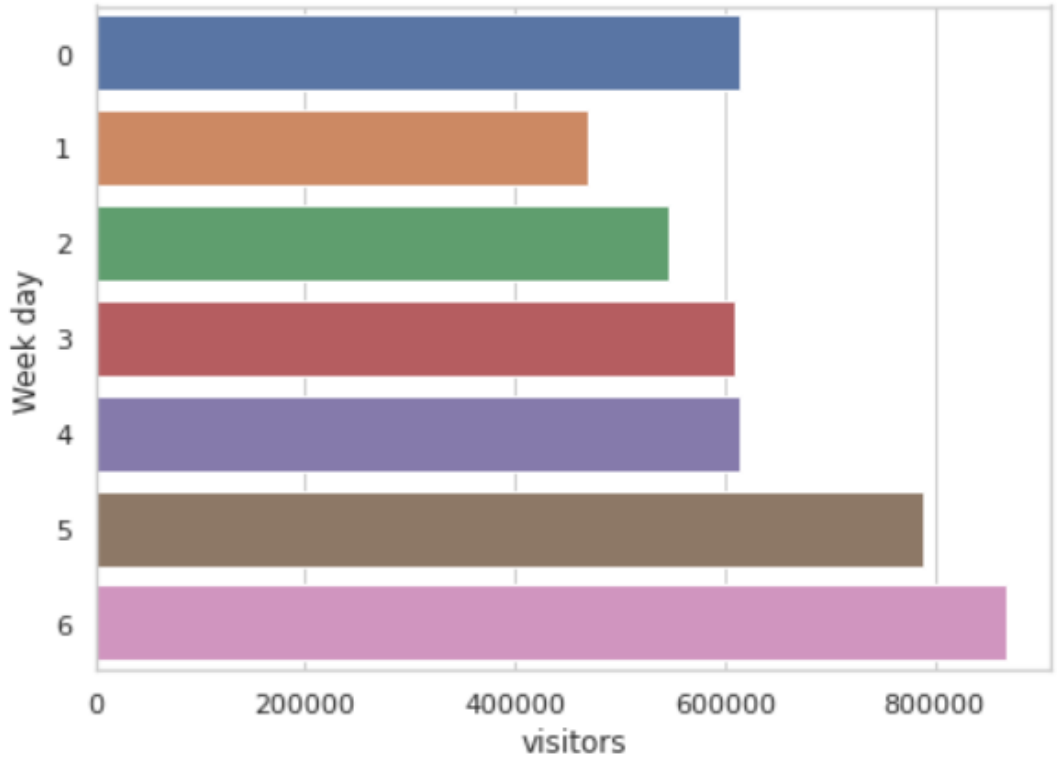
After doing that we have the following box plot and the data distribution that’s more intuitive to gain some inference.



Below is the box plot that depicts the same as the data after Q3 is densely populated and that’s good sign.



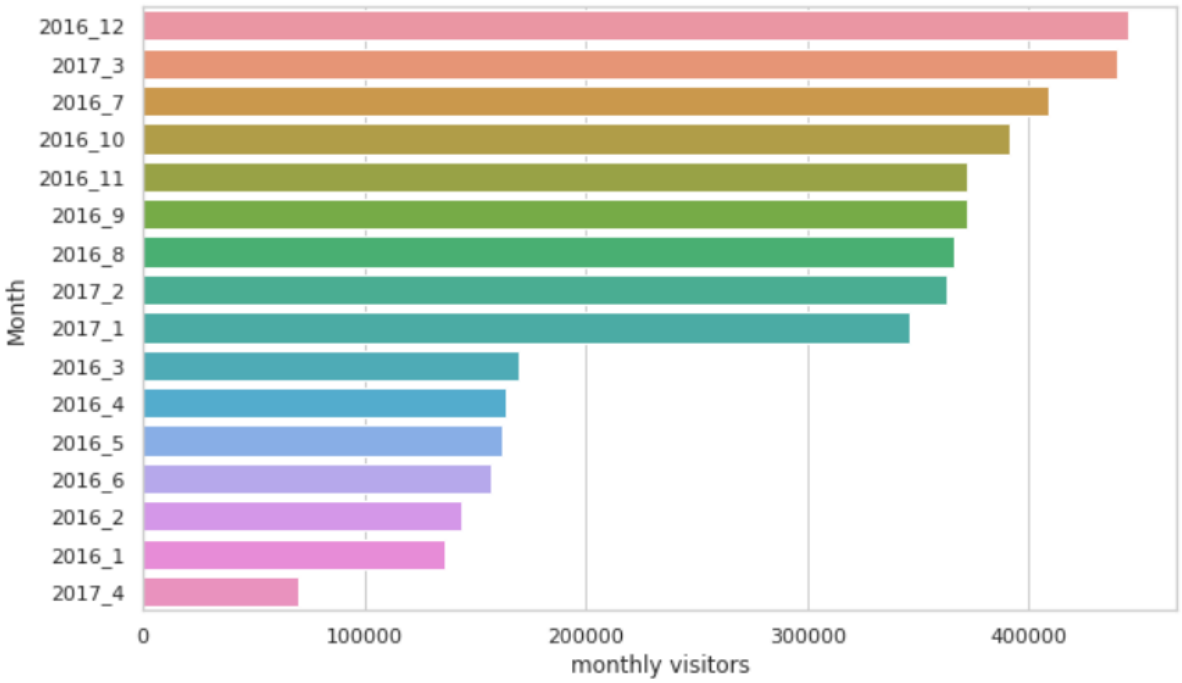
Weekly Visitors Analysis



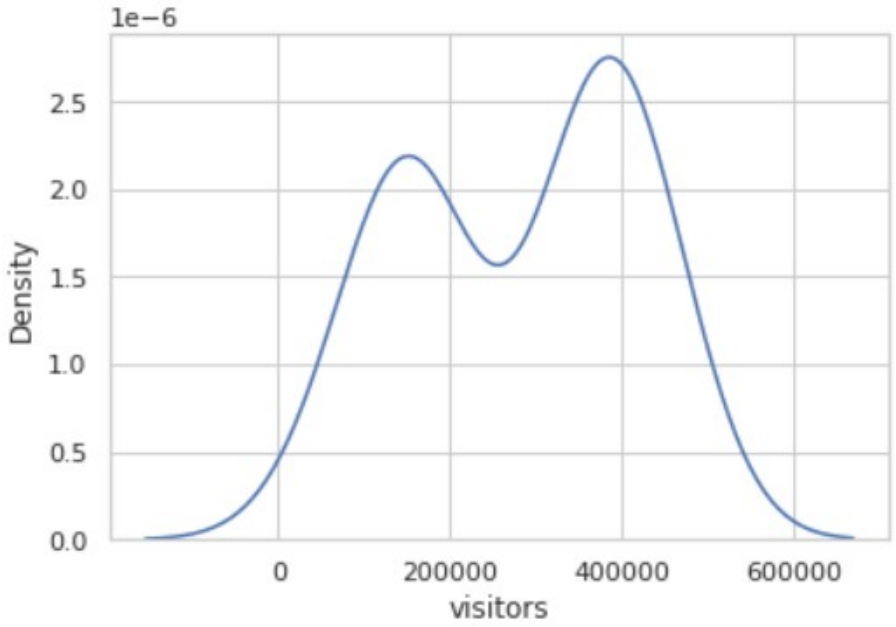
So our previous statement is indeed correct as we see the surge in number of visitors.

That’s the sole reason we have included the week end feature.

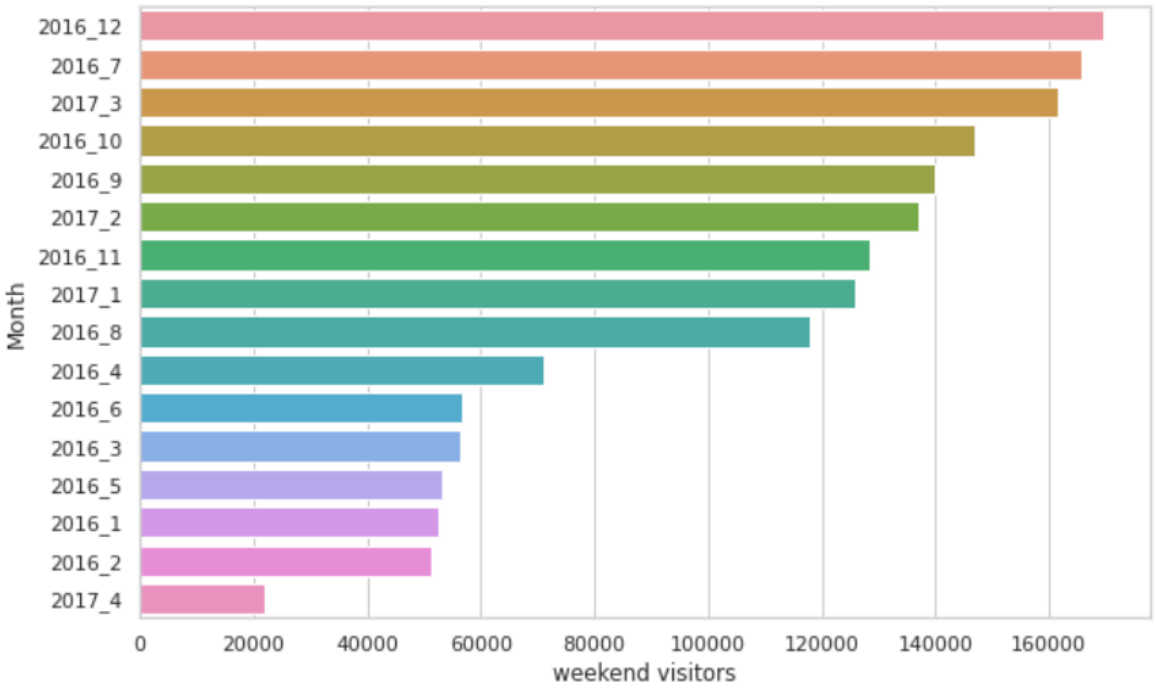
Monthly data analysis



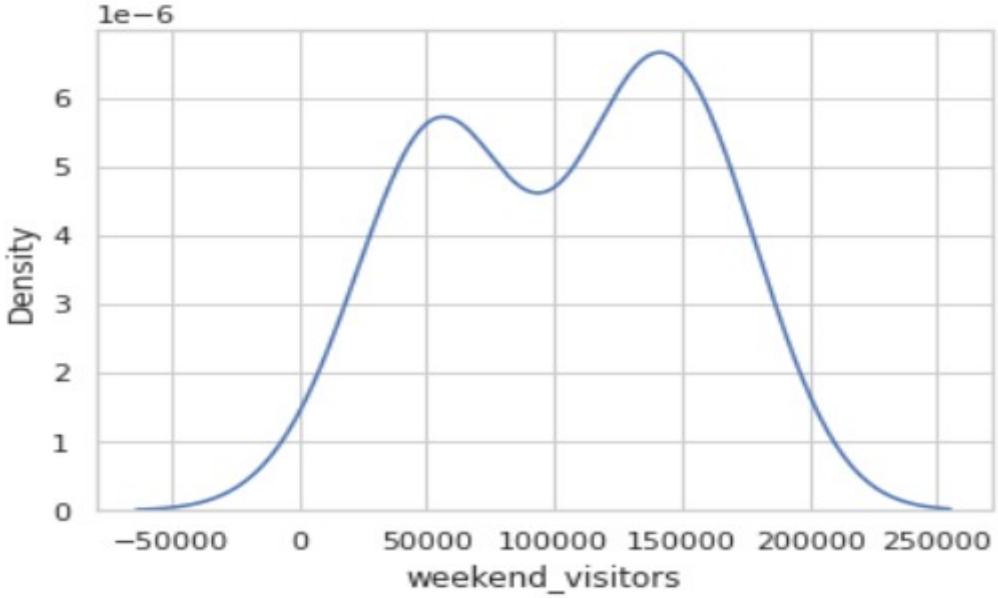
We see that December of 2016 have large number of visitors which must also be the case because the number of holidays in that and top few months are more compared to other months.



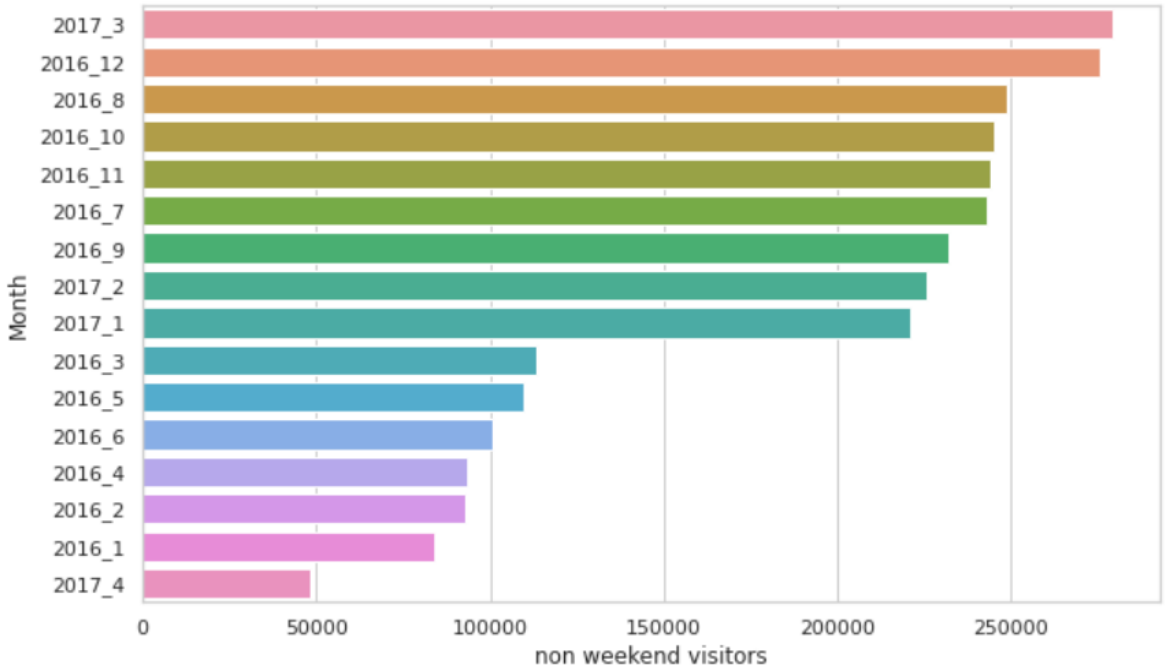
Monthly Week End data analysis



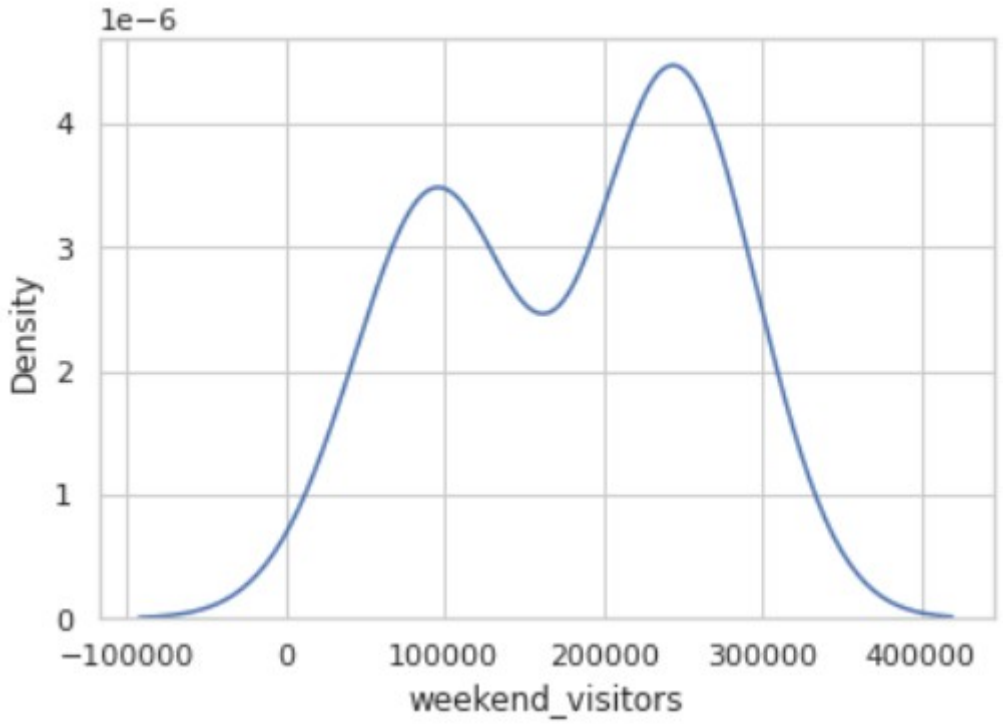
Here we shown the number of weekend visitors in each month, and we have December 2016 as the month having max weekend visitors followed by mid 2017 and late 2016 months.



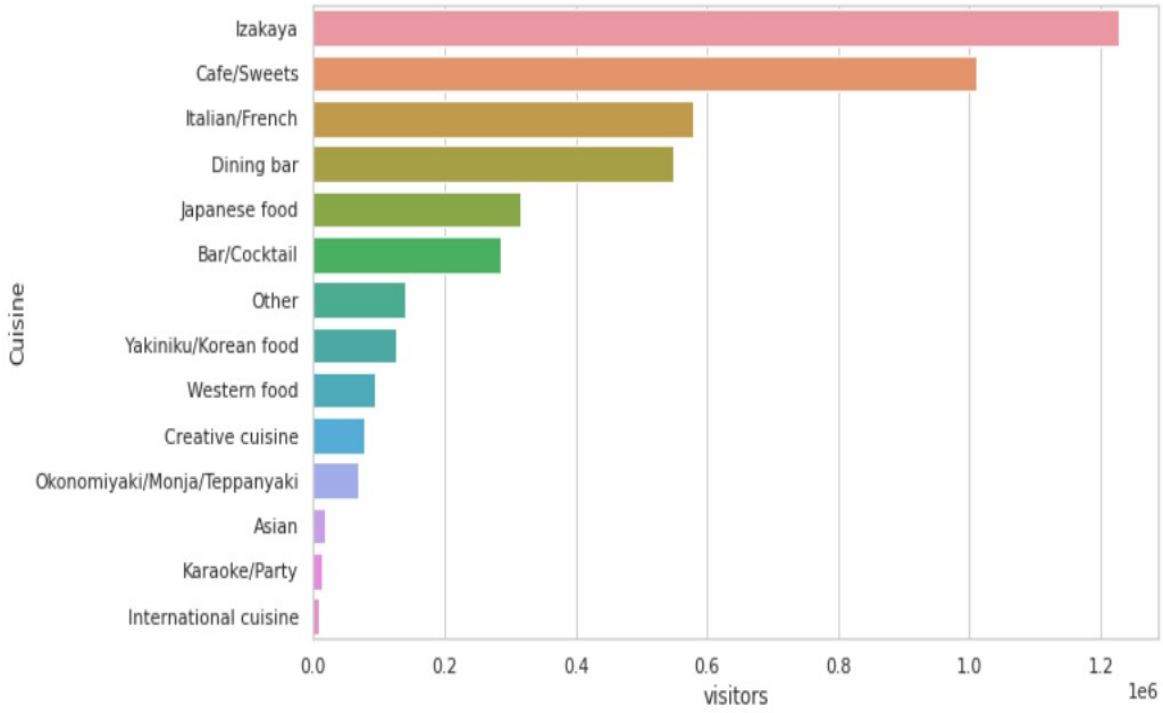
Monthly Non Week End data analysis



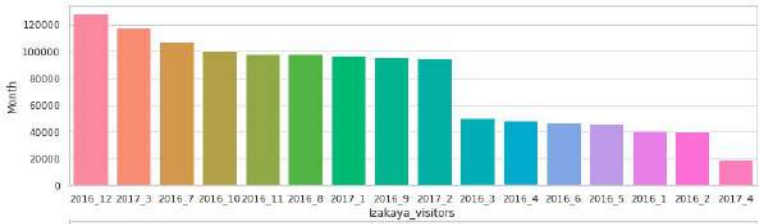
Here we see the non-week end visitors are max at early 2017 and low in early 2016 and mid 2017.

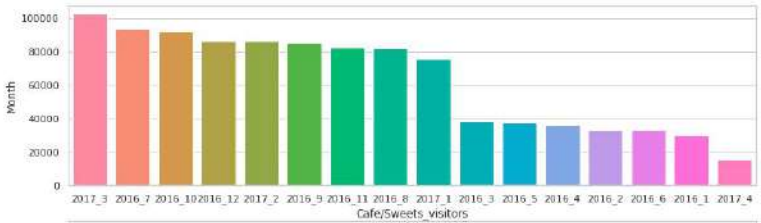


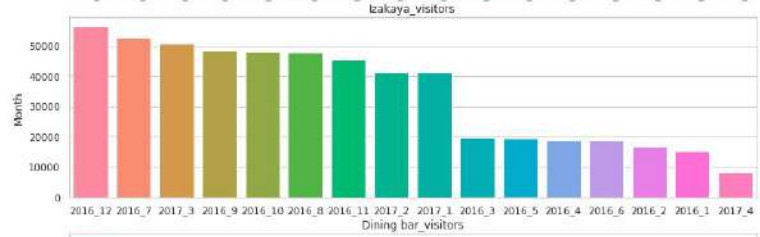
Cuisine analysis

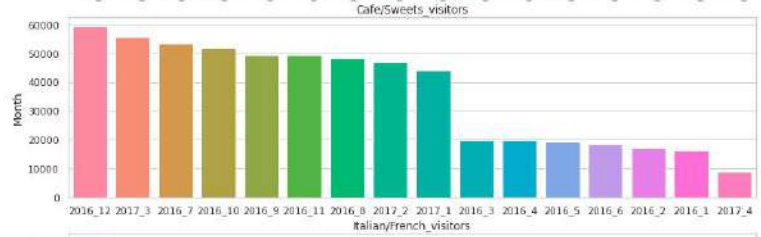


We have done monthly analysis for each cuisine and shown in below diagram.

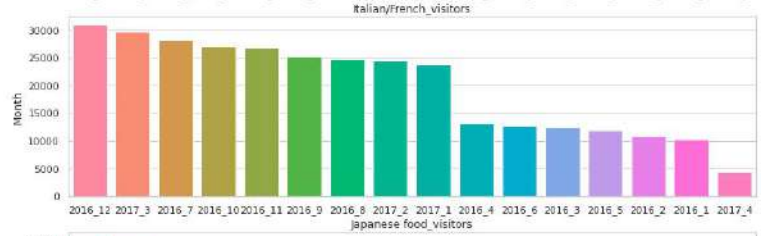

Month
2016_12 2017_3 2016_7 2016_10 2016_11 2016_8 2017_1 2016_9 2017_2 2016_3 2016_4 2016_6 2016_5 2016_1 2016_2 2017_4
Izakaya_visitors

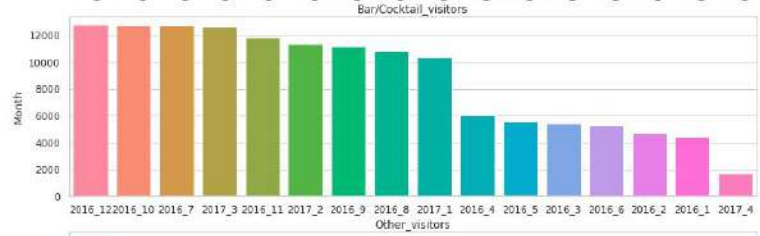

Month
2017_3 2016_7 2016_10 2016_12 2017_2 2016_9 2016_11 2016_8 2017_1 2016_3 2016_5 2016_4 2016_2 2016_6 2016_1 2017_4
Cafe/Sweets_visitors

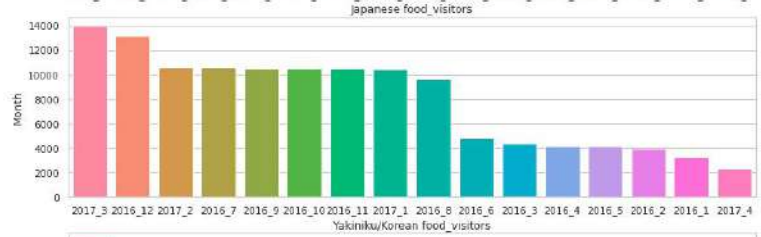

Month
2016_12 2016_7 2017_3 2016_9 2016_10 2016_8 2016_11 2017_2 2017_1 2016_3 2016_5 2016_4 2016_6 2016_2 2016_1 2017_4
Dining bar_visitors

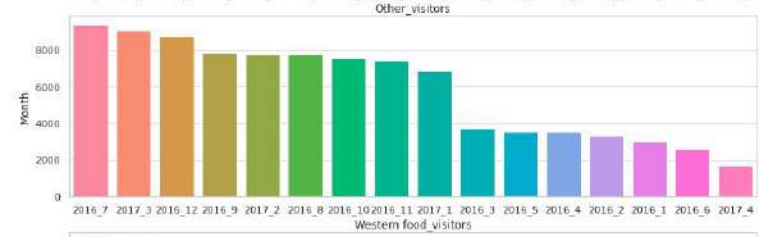

Month
2016_12 2017_3 2016_7 2016_10 2016_9 2016_11 2016_8 2017_2 2017_1 2016_3 2016_4 2016_5 2016_6 2016_2 2016_1 2017_4
Italian/French_visitors

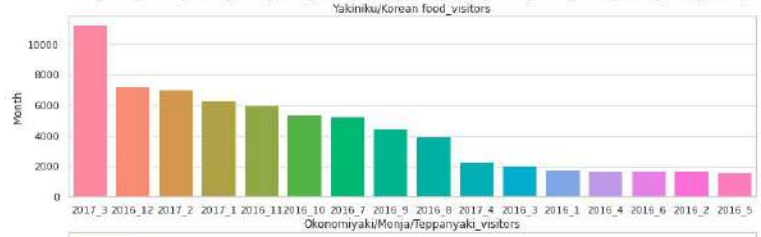

Month
2016_12 2016_7 2017_3 2016_10 2016_11 2016_9 2016_8 2017_2 2017_1 2016_3 2016_4 2016_5 2016_6 2016_2 2016_1 2017_4
Bar/Cocktail_visitors

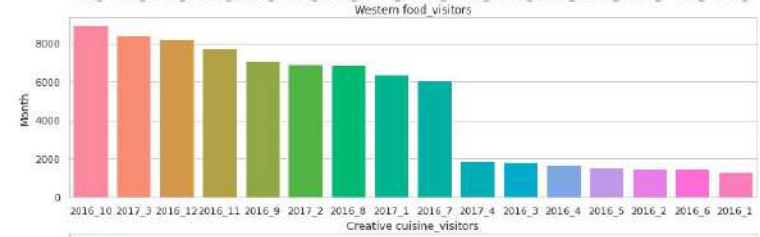

Month
2016_12 2017_3 2016_7 2016_10 2016_11 2016_9 2016_8 2017_2 2017_1 2016_4 2016_6 2016_3 2016_5 2016_2 2016_1 2017_4
Japanese food_visitors

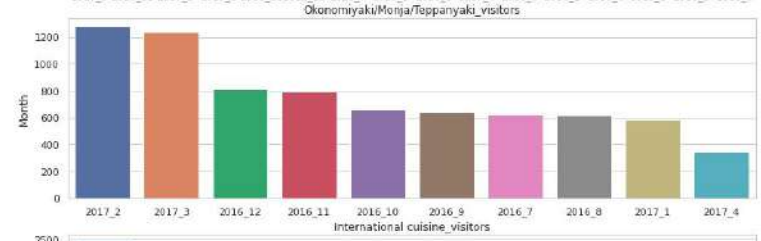

Month
2016_12 2016_10 2016_7 2017_3 2016_11 2017_2 2016_9 2016_8 2017_1 2016_4 2016_5 2016_3 2016_6 2016_2 2016_1 2017_4
Other_visitors

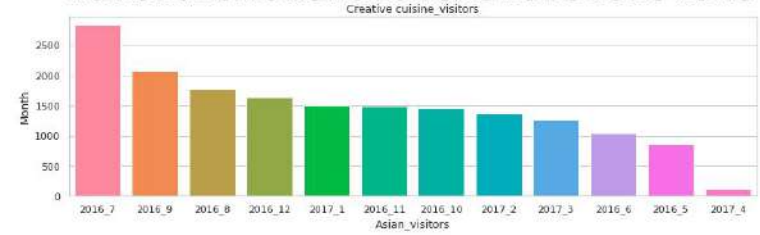

Month
2017_3 2016_12 2017_2 2016_7 2016_9 2016_10 2016_11 2017_1 2016_8 2016_6 2016_3 2016_4 2016_5 2016_2 2016_1 2017_4
Yakiniku/Korean food_visitors

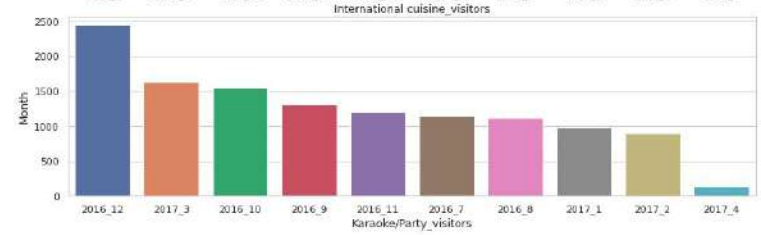

Month
2016_7 2017_3 2016_12 2016_9 2017_2 2016_8 2016_10 2016_11 2017_1 2016_3 2016_5 2016_4 2016_2 2016_1 2016_6 2017_4
Western food_visitors


Month
2017_3 2016_12 2017_2 2017_1 2016_11 2016_10 2016_7 2016_9 2016_8 2017_4 2016_3 2016_1 2016_4 2016_6 2016_2 2016_5
Okonomiyaki/Monja/Teppanyaki_visitors

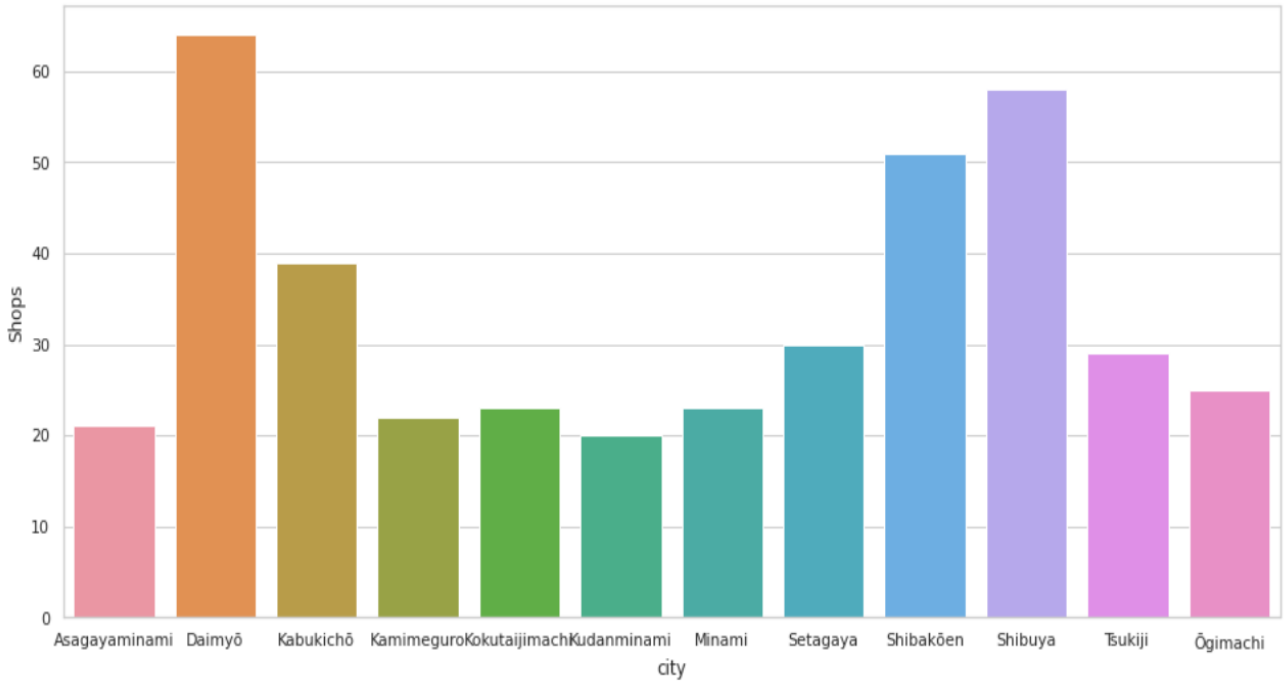

Month
2016_10 2017_3 2016_12 2016_11 2016_9 2017_2 2016_8 2017_1 2016_7 2017_4 2016_3 2016_4 2016_5 2016_2 2016_6 2016_1
Creative cuisine_visitors


Month
2017_2 2017_3 2016_12 2016_11 2016_10 2016_9 2016_7 2016_8 2017_1 2017_4
International cuisine_visitors

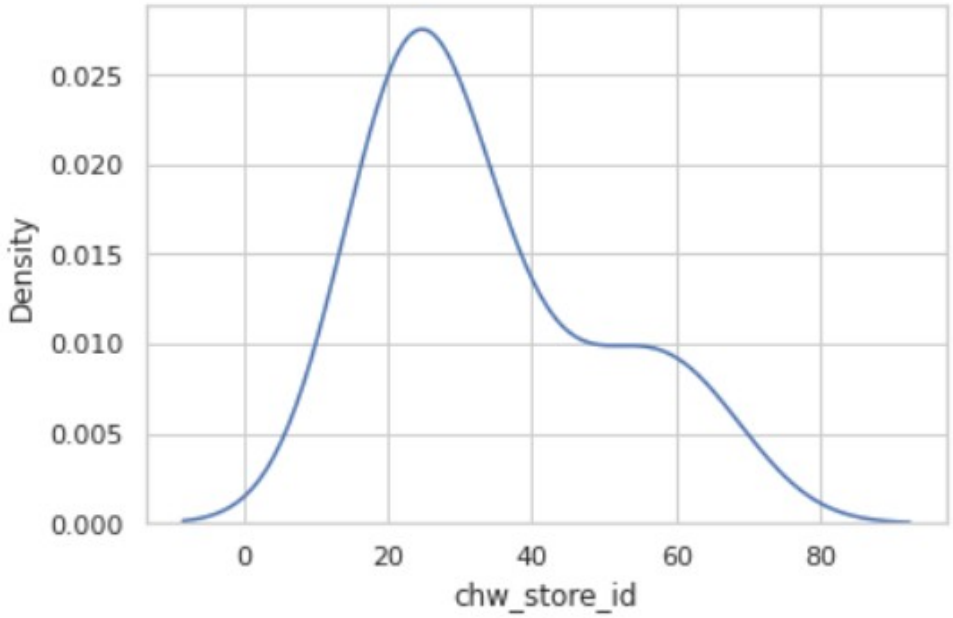

Month
2016_7 2016_9 2016_8 2016_12 2017_1 2016_11 2016_10 2017_2 2017_3 2016_6 2016_5 2017_4
Asian_visitors


Month
2016_12 2017_3 2016_10 2016_9 2016_11 2016_7 2016_8 2017_1 2017_2 2017_4
Karaoke/Party_visitors

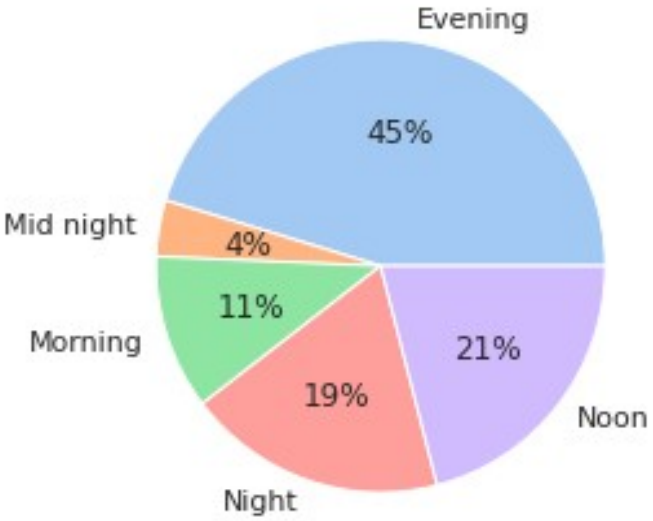
Locality shops analysis

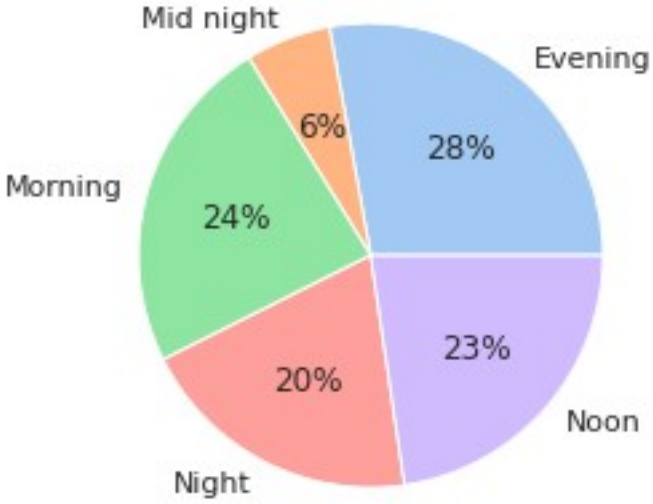


We have also shown the distribution plot of number of shops!



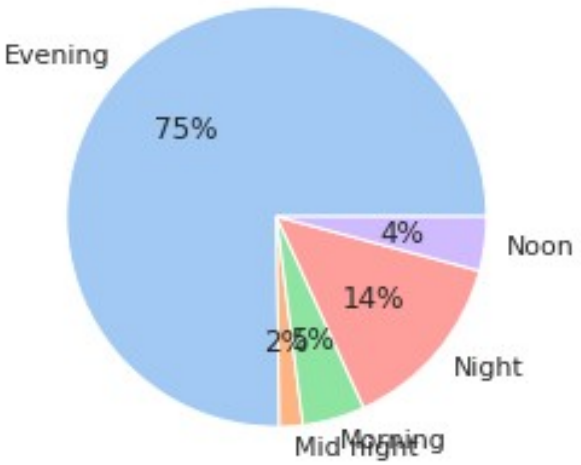
Time based reservation analysis

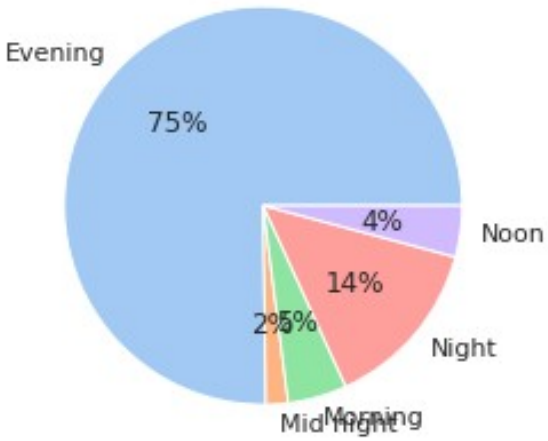




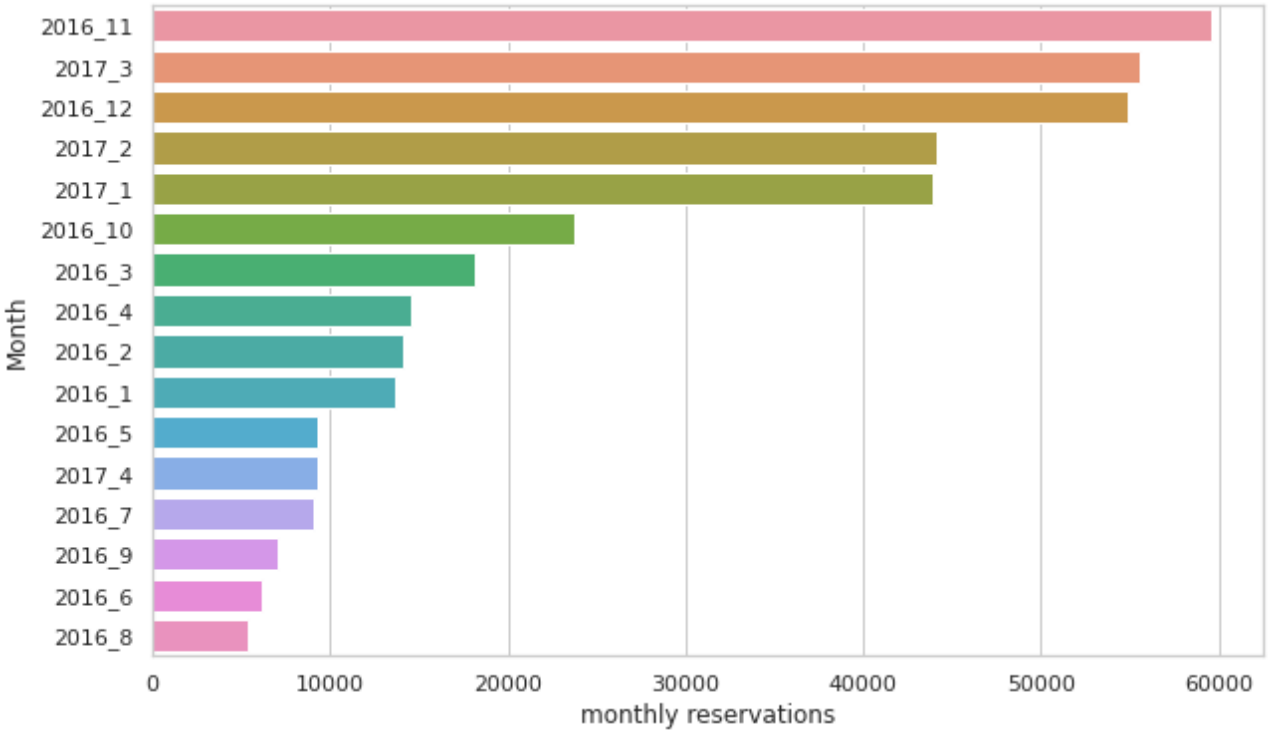
Evening is the time we see most of the reservations in chwiggy (left) and same goes for yomato (right) although reservations are distributed evenly in yomato.

Time based Reserve visitors analysis





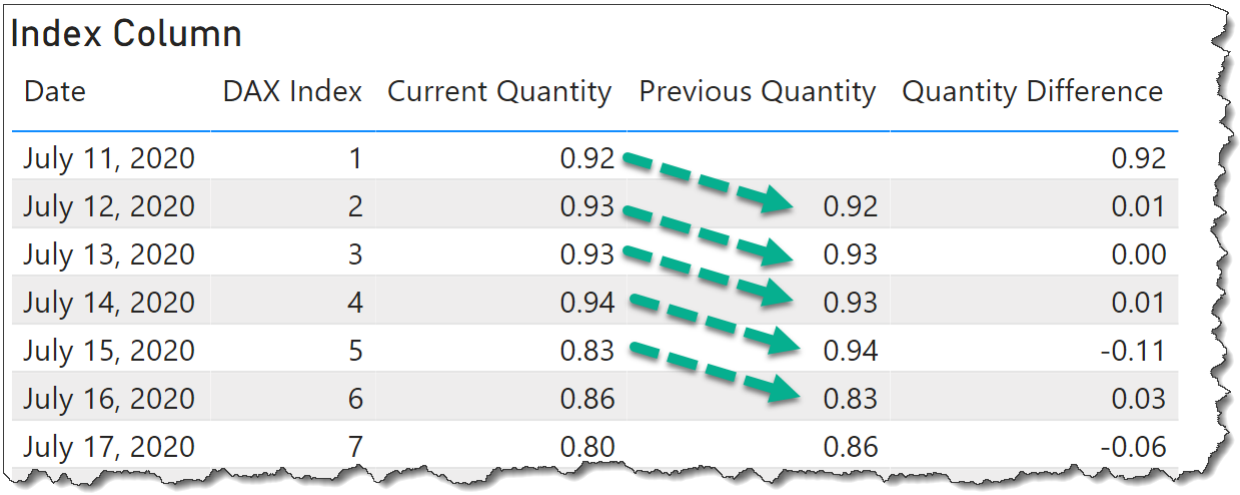
Monthly Reservation Analysis



As our intuition suggests as the reservations are more in the month of december 2016 so as the number of visitors.

Design of various features :

LAGS in Data Science



Index Column

Date	DAX Index	Current Quantity	Previous Quantity	Quantity Difference
July 11, 2020	1	0.92		0.92
July 12, 2020	2	0.93	0.92	0.01
July 13, 2020	3	0.93	0.93	0.00
July 14, 2020	4	0.94	0.93	0.01
July 15, 2020	5	0.83	0.94	-0.11
July 16, 2020	6	0.86	0.83	0.03
July 17, 2020	7	0.80	0.86	-0.06

We will be creating the lagging features of target variable that's visitors, also we can create lag features of multiple numeric featurs inclusing the number of visitors after reservations and the number of visitors itself.

Not ony the features that we are provided with but also the features that we have created using the group by aggregations can be used to create lags.

GROUP BY Aggregation:

Following features have been created using the group by function.

1) mean visitors of the day in locality.

2) mean visitors of the day in locality of specific genre.

3) mean visitors in city.

4) mean visitors in city for specific genre.

5) mean visitors for specific genre.

6) mean visits in week.

7) mean visits in month.

8) mean/median/sum of visits till now.

9) most profitable week for specific resturent.

10) most proitable week for specific genre.

11) most profitable week for specific locality.

12) most profitable week for specific locality with specific genre.

Hundereds of features will be created when we lag all these visitors along with the lag of visitors.

1) Lag of past seven days.

2) Lag of past month.

3) Lag of future 7 days.

4) lag of future 30 days.

Following is the defination of function used to create lags.

```
def create_lags_with_gb(lag,index_cols,df, null_value = 0):
    global all_data
    df_temp = df.copy()
    df_temp["date"] = pd.to_datetime(df_temp['date']) + pd.to_timedelta(lag,unit='d')
    df_temp["date"] = pd.to_datetime(df_temp['date']).dt.strftime("%m/%d/%y")
    columns = []
    if lag != 0:
        for col in df_temp.columns.values:
            if col in index_cols:
                columns.append(col)
            else:
                columns.append(col + "_lag_" + str(lag))
        df_temp.columns = columns

    all_data = all_data.merge(df_temp,how="left",on=index_cols)
    del df_temp
    gc.collect()
    for col in [col for col in columns if not col in index_cols]:
        all_data[col].fillna(null_value, inplace = True)
        all_data[col] = all_data[col].astype(np.float32)
        gc.collect()
```

Following is the final list of features that were included in our project.

```
'visitors',
'date',
 'week_day',
 'holiday',
 'quarter',
 'week_end',
 'week',
 'month',
 'year',
 'day',
 'days',
 'month_holiday_agg',
 'month_week_holiday_agg',
 'month_num',
 'week_num',
 'median',
 'mean',
 'chw_genre_name',
 'latitude',
 'longitude',
 'province',
 'city',
 'locality',
 'reservations',
 'reserve_visitors',
 'visitors_mean_past_30_2',
 'visitors_median_past_30_2',
 'visitors_mean_past_14_2',
 'visitors_median_past_14_2',
 'visitors_mean_mid_14_2',
 'visitors_median_mid_14_2',
```

```
'visitors_mean_future_30',
'visitors_median_future_30',
'visitors_mean_fu_14',
'visitors_median_fu_14',
'visitors_mean_past_30',
'visitors_median_past_30',
'visitors_mean_past_14',
'visitors_median_past_14',
'visitors_mean_mid_14',
'visitors_median_mid_14',
'visitors_mean_fu_mid_14',
'visitors_median_fu_mid_14',
'visitors_lag_1',
'visitors_lag_2',
'visitors_lag_3',
'visitors_lag_4',
'visitors_lag_5',
'visitors_lag_6',
'visitors_lag_7',
'visitors_lag_8',
'visitors_lag_9',
'visitors_lag_10',
'visitors_lag_11',
'visitors_lag_12',
'visitors_lag_13',
'visitors_lag_14',
'visitors_lag_15',
'visitors_lag_16',
'visitors_lag_17',
'visitors_lag_18',
'visitors_lag_19',
'visitors_lag_20',
'visitors_lag_21',
'visitors_lag_22',
'visitors_lag_23',
```

```
'visitors_lag_24',
'visitors_lag_25',
'visitors_lag_26',
'visitors_lag_27',
'visitors_lag_28',
'visitors_lag_29',
'visitors_lag_-1',
'visitors_lag_-2',
'visitors_lag_-3',
'visitors_lag_-4',
'visitors_lag_-5',
'visitors_lag_-13',
'visitors_lag_-14',
'visitors_lag_-15',
'visitors_lag_-16',
'visitors_lag_-18',
'visitors_lag_-20',
'visitors_lag_-21',
'visitors_lag_-28',
'reservations_lag_1',
'reservations_lag_2',
'reservations_lag_3',
'reservations_lag_4',
'reservations_lag_5',
'reservations_lag_6',
'reservations_lag_7',
'reserve_visitors_lag_1',
'reserve_visitors_lag_2',
'reserve_visitors_lag_3',
'reserve_visitors_lag_4',
'reserve_visitors_lag_5',
'reserve_visitors_lag_6',
'reserve_visitors_lag_7',
'median_lag_2',
'median_lag_3',
```

```
'median_lag_4',
'median_lag_5',
'median_lag_7',
'median_lag_14',
'median_lag_21',
'median_lag_28',
'mean_lag_2',
'mean_lag_3',
'mean_lag_4',
'mean_lag_5',
'mean_lag_7',
'mean_lag_14',
'mean_lag_21',
'mean_lag_28',
'median_lag_-2',
'median_lag_-5',
'median_lag_-7',
'median_lag_-4',
'median_lag_-3',
'median_lag_-21',
'median_lag_-28',
'mean_lag_-2',
'mean_lag_-5',
'mean_lag_-7',
'mean_lag_-4',
'mean_lag_-3',
'mean_lag_-21',
'mean_lag_-28',
'local_visitors',
'local_visitors_lag_1',
'local_visitors_lag_2',
'local_visitors_lag_3',
'local_visitors_lag_4',
'local_visitors_lag_5',
'local_visitors_lag_6',
```

```
'local_visitors_lag_7',
'local_genere_visitors',
'local_genere_visitors_lag_1',
'local_genere_visitors_lag_2',
'local_genere_visitors_lag_3',
'local_genere_visitors_lag_4',
'local_genere_visitors_lag_5',
'local_genere_visitors_lag_6',
'local_genere_visitors_lag_7',
'genere_visitors',
'genere_visitors_lag_1',
'genere_visitors_lag_2',
'genere_visitors_lag_3',
'genere_visitors_lag_4',
'genere_visitors_lag_5',
'genere_visitors_lag_6',
'genere_visitors_lag_7',
'city_visitors',
'city_visitors_lag_1',
'city_visitors_lag_2',
'city_visitors_lag_3',
'city_visitors_lag_4',
'city_visitors_lag_5',
'city_visitors_lag_6',
'city_visitors_lag_7',
'city_genere_visitors',
'city_genere_visitors_lag_1',
'city_genere_visitors_lag_2',
'city_genere_visitors_lag_3',
'city_genere_visitors_lag_4',
'city_genere_visitors_lag_5',
'city_genere_visitors_lag_6',
'city_genere_visitors_lag_7',
'province_genere_visitors',
'province_genere_visitors_lag_1',
```

```
'province_genere_visitors_lag_2',
'province_genere_visitors_lag_3',
'province_genere_visitors_lag_4',
'province_genere_visitors_lag_5',
'province_genere_visitors_lag_6',
'province_genere_visitors_lag_7',
'previous_day_holiday',
'next_day_holiday',
'previous_day_holiday_2',
'next_day_holiday_2',
'max_shop_visit_month',
'max_shop_visit_week',
'max_local_visit_month',
'max_local_visit_week',
'max_genere_visit_month',
'max_genere_visit_week',
'max_city_visit_month',
'max_city_visit_week',
'max_city_genre_visit_month',
'max_city_genre_visit_week',
'max_local_genre_visit_month',
'max_local_genre_visit_week'
```

As you may have noticed some features have negative number after theier name which means we are going into future and creating the lag that incluces future data shifted by that number..

Models used with their scores and some breif.

Neural network:

We started with shallow neural networks and tested them with one hot feature encoding of our categorical features and then got a score which was way too less than our expectation.

We tried all feature scalings and also used multiple folds and tweaked various hyper-parameters but still score wasn’t as expected.

Xg – boost / Boosting

Well this was probably the best fit for our dataset.. using the features as listed above gave us way better results compared to neural network and we hadn’t used any feature scaling and one hot encoding initially but as time progressed we noticed that even though the decision tree doesn’t require this feature engineering, we can still make use of them and can get better results so here we introduced scaling + one hot encoding to get expected results.

Ensembling

Finally ended up with some ensembling and the way we did it is that we used our previous outputs and substituted it with our test set values and used that test file instead of new one to make our features...

This way we got rid with zeros in our test set features!

Proved to be the best strategy. As we used some light boosting method along with our ensembling to get our best score.

Model	SCORE
Neural Network	0.5290
Xgboost	0.50378
Light Gbm + Ensembling	**0.49091**

Contributions by individual

We have divided our work efforts in various sections..

1) Reporting

While one works with EDA reporting other manages the notebook.

2) Model Selection

While one keeps track of what progress a model reported compared to previous one, other keeps on finding new ideas to apply to the project.

3) Feature Selection

While one keeps track of what features were added and how the integration results can be used, other finds new features to be added to the model.

4) presenting

Both actively participated while presenting, while one explains EDA other explains Model training and selection.

Conclusions.

We will conclude with the description of type of model that we have used..

Ensembling with Light GBM proved to be the best model and although in ensembling we used the xgboost to get our predictions and feeding the same predicitons again to xgboost to get brushed up predictions..

We then used these predictions to replace the test set’s target values which originally contained just zeros..

Then we ran the feature generation cycle again to get the features and trained the light gbm model again on top of them.

References

1) How to win kagge competitions, HSE course on coursera..

2) Andrew NG.

3) MIT

4) Statistical Learning and Therory Book by MIT profs