Time Series Analysis of Business Reviews in Yelp Academic Dataset

Group 4 AID: Vedant Naik, Dixit Patel, Akshay Raje
Group Leader: Vedant Naik

1. Abstract

Many businesses such as restaurants/shopping centers, have a varying rate of patron activity over a period of a year. Getting an estimate of this activity rate will help business owners take smarter decisions to prepare for the upcoming traffic. In light of this, we decided to use dates of reviews posted on Yelp as time-series data, and predict future activity.

2. Introduction of the background

After preliminary analysis of data, we learned that reviews posted by patrons who visit an establishment, can be used to evaluate how frequently do customers visit that establishment. Users tend to review any business soon after they visit the place. So, reviews will give a good estimate of an "activity" at that business.

In order to verify our results, we will use data prior to the year 2013, and forecast the review counts for the year 2013. We will then compare this against the original data of 2013.

3. Problem Definition

We will primarily use historical data of a business to predict its future activity. "Activity" can be considered as a count of all posted reviews for a month. So given review data of a business for 10 years, we will have (10*12) values of review counts.

4. Problem formalization

From the given Yelp Academic Dataset, the best estimate for activity at a business is the number of reviews posted for that business. We count all reviews posted for a business in a month and consider it as the "review count" for that month. So, each year will have 12 review counts.

In order to perform time series analysis on any business, we get review counts for all the years between 2004-2015. Since certain businesses started after the year 2004, we do not always have useful values in the first few years. We identify the "startyear" as the earliest year which has useful review counts for a business.

We organize the initial data as follows:

B: A business identified by an ID (for the sake of this report, we consider the establishment "Paris Las Vegas Hotels & Casino" identified by bID = 4bEjOyTaDG24SY5TxsaUNQ)

Sxxxx: Date counts over a period of one year (where, xxxx is a year).

[S will be a map (dictionary in python) of 12(months)]

e.g.: Review counts for a given business
(consider bID = 4bEjOyTaDG24SY5TxsaUNQ), is stored in a dictionary:

B = {2006: S2006, 2007: S2007, 2008: S2008, 2009: S2009,
2010: S2010, 2011: S2011, 2012: S2012, 2013: S2013, 2014: S2014]

Where each Sxxxx contains review counts for each month in that year:

e.g.: S2004 = {1: 1, 2: 1, 3: 1, … , 12: 1}
S2005 = {1: 1, 2: 1, 3: 1, … , 12: 2}
S2006 = {1: 3, 2: 1, 3: 1, … , 12: 2}
....
S2012 = {1: 62, 2: 45, 3: 65, … , 12: 63}
S2013 = {1: 68, 2: 75, 3: 71, … , 12: 86}
S2014 = {1: 120, 2: 78, 3: 114, … , 12: 79}

In order to use the proposed algorithms, we need to identify training data which is used to fit the model for a specific business. To maintain uniformity in the models for all businesses, we will use data between the "startyear" and 2013, as the training data. We will use this training data to fit our models for

that business, and forecast the review counts for the year 2013. Since we have real data for the year 2013, we can later verify our forecast results against it.

5. Data description and Preprocessing

5.1. Data Description: yelp_dataset_challenge_academic_dataset

The data consists of the following documents in json format:
yelp_academic_dataset_review.json
yelp_academic_dataset_business.json
yelp_academic_dataset_tip.json
yelp_academic_dataset_checkin.json
yelp_academic_dataset_user.json

Of these files, we use the following:

1. **yelp_academic_dataset_review.json:** Reviews posted by different users for a particular business since the business was present on Yelp. The date on which the review was posted is important to us.
2. **yelp_academic_dataset_business.json:** The details of each business including, but not limited to, its business_id, name, address, geo-location coordinates, city and categories.

business

```
{
    'type': 'business',
    'business_id': (encrypted business id),
    'name': (business name),
    'neighborhoods': [(hood names)],
    'full_address': (localized address),
    'city': (city),
    'state': (state),
    'latitude': latitude,
    'longitude': longitude,
    'stars': (star rating, rounded to half-stars),
    'review_count': review count,
    'categories': [(localized category names)]
    'open': True / False (corresponds to closed, not business hours),
    'hours': {
        (day_of_week): {
            'open': (HH:MM),
            'close': (HH:MM)
        },
        ...
    },
    'attributes': {
        (attribute_name): (attribute_value),
        ...
    },
}
```

review

```
{
    'type': 'review',
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'stars': (star rating, rounded to half-stars),
    'text': (review text),
    'date': (date, formatted like '2012-03-14'),
    'votes': {(vote type): (count)},
}
```

5.2. Pre-Processing

5.2.1 Data Reduction -

Total number of businesses is 61,184; and total number of reviews is 15, 69,264.

5.2.2 Entity Reduction -

There are many business with too few reviews for our need. These won't be of much use during the analysis phase. So we prune the dataset to get rid of these businesses and only include those businesses with sufficient number of reviews. Using Kibana, we identified top 10% businesses in the city with most reviews (Las Vegas).

5.2.3. Attribute Reduction -

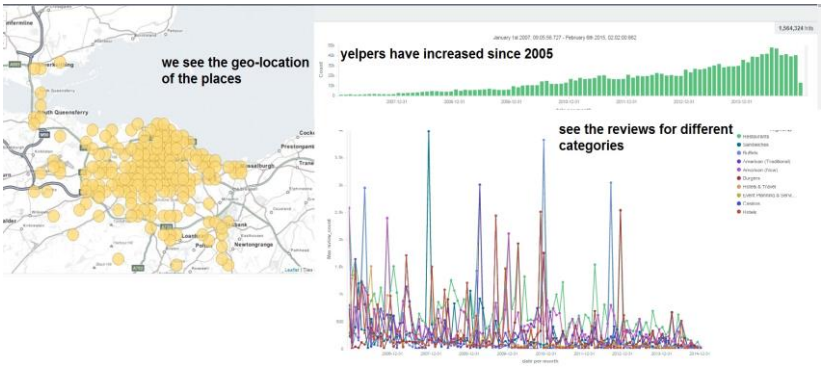
Since we will not require much data about reviews other than ID and date, we can safely ignore the remaining data such as review text, likes, comments etc. from the ‘reviews’ json file. From the ‘business’ json, we will only need the business_id, name, categories, and location.

6. Method Description

6.1 Data Preprocessing -

We designed ways to conveniently fetch, insert and interact with data much faster. We first uploaded all the raw data into Elasticsearch and used Kibana to visualize it. To further prune/reduce the data, we

wrote python scripts which read the yelp dataset files and produced custom comma separated value (csv) files, with time-series data, to give as input to R scripts which run the time-series forecasting algorithms.



Elasticsearch:

We chose Elasticsearch as a database for quick data fetching and visualizations using Kibana. For Elasticsearch, we designed mappings which allow fields to be treated as a particular data type. Once these mappings were designed, we inserted data into Elasticsearch after conducting some cleaning steps, such as replacing empty jsons with default values, combining latitude and longitude into 'geo-points'.

Kibana is a node.js tool used over Elasticsearch and facilitates basic visualizations. Here are a few examples:

Python Scripts:

We wrote python scripts to perform data reduction and conversion tasks. Also, we wrote scripts to automate the process and scale it for large number of businesses. Since we are using R packages for fitting time-series data and producing forecasts, we make calls to R scripts from our automated python scripts. The entire process is described in short here:

Once we got a general understanding of the data, and how it was structured, we wrote simple python scripts to extract relevant data from the raw dataset. We recorded the relevant review counts data in a time series data structure, which contains time series for all months between the year 2004 and 2015. We also record the maximum count value and "startyear" for each business. We use this information about the business to give as an input to the algorithms, and produce necessary prediction and evaluation plots.

In the automation step, we use the time series of businesses which we are interested in, and plot the following graphs:

- Original data (With ACF, PACF, Q-Q Plot, Histogram of the residual)
- Seasonal Decomposition
- ARIMA forecast (With ACF, PACF, Q-Q Plot, Histogram of the residual)
- Holt-Winters Additive forecast (With ACF, PACF, Q-Q Plot, Histogram of the residual)
- Holt-Winters Multiplicative forecast (With ACF, PACF, Q-Q Plot, Histogram of the residual)

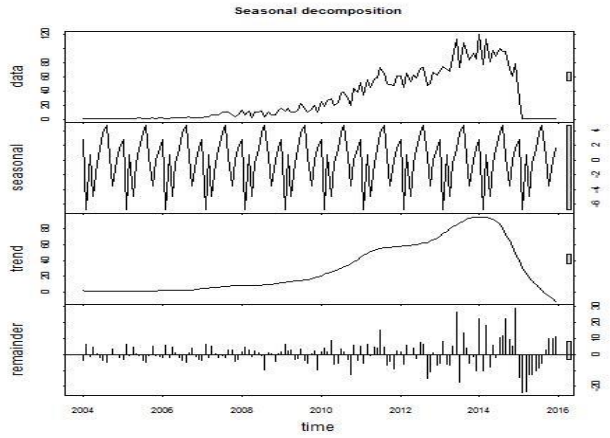
6.2 Algorithms

6.2.1 STL: Seasonal-Trend Decomposition Procedure Based on Loess -

STL is a filtering procedure of decomposing a time series into trend, seasonal, and remainder components. STL consists of a sequence of smoothing operations which employs the same smoothing - locally-weighted scatterplot smoothing (loess). Loess uses local regression to remove "jaggedness" from the data.

In Loess smoothing, a window of a specified width is placed over the data. The wider the window, smoother is the resulting loess curve. A regression line (or curve) is fitted to the observations that fall within the window, the points closest to the centre of the window being weighted to have the greatest effect on the calculation of the regression line. This process is repeated several times. We thereby obtain a point on the loess curve. This is the point on the regression line at the centre of the window. The loess curve is calculated by moving the window across the data. Each point on the resulting loess curve is the intersection of a regression line and a vertical line at the centre of such a window.

We have utilized *stl()* function of R which works well for seasonal-trend decomposition of time series with low or high seasonality and normally distributed noise. When calling *stl()* with *s.window="periodic"*, the seasonal component for a month is simply the mean of all values for that month. Otherwise, the seasonal component is calculated using Loess smoothing. Having calculated the seasonal component, the seasonally-adjusted data (the original data minus the seasonal component) is loess-smoothed to determine the trend. The remainder/noise is then the original data minus the seasonal and trend components.



The plot on the left hand side shows components in the order they are listed below:

1. Original time series: Time series of the monthly review counts for Business ID 4bEjOyTaDG24SY5TxsaUNQ.
2. Seasonal component identified by *stl()*.
3. Trend component identified by *stl()*.
4. Noise/remainder component identified by *stl()*.

6.2.2 ARIMA -

ARIMA models are used instead of ARMA models when the time series exhibits non-stationarity (i.e. parameters such as the mean and variance change over time), and are thus more general than ARMA.

An ARIMA model is represented as ARIMA(p,d,q), where:

p is the number of autoregressive terms,

d is the number of non-seasonal differences, and

q is the number of lagged forecast errors in the prediction equation.

Such that p, d, and q are integers greater than or equal to zero and refer to the order of the autoregressive, integrated and moving average part.

$$\underset{\uparrow}{(1-\phi_1 B-\cdots-\phi_p B^p)}\quad \underset{\uparrow}{(1-B)^d y_t}\quad = \quad c+\underset{\uparrow}{(1+\theta_1 B+\cdots+\theta_q B^q)}e_t$$
$$\text{AR}(p) \qquad d \text{ differences} \qquad \text{MA}(q)$$

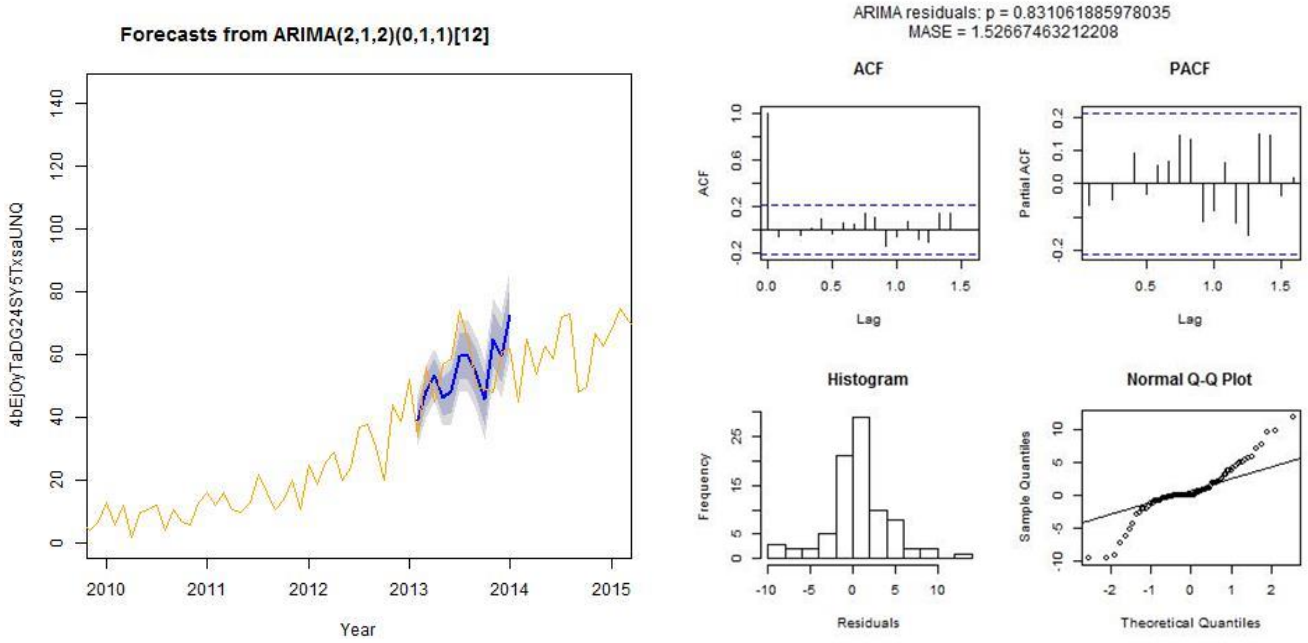
The representation of ARIMA model above is for non-seasonal component. When the model should take into consideration the seasonal component, the representation changes to the following -

$$\text{ARIMA}\ \underbrace{(p,d,q)}_{\uparrow}\ \underbrace{(P,D,Q)_m}_{\uparrow}$$
$$\left(\begin{array}{c}\text{Non-seasonal part}\\ \text{of the model}\end{array}\right)\ \left(\begin{array}{c}\text{Seasonal part}\\ \text{of the model}\end{array}\right)$$

Fitting ARIMA model:

1. If the data are non-stationary: take first differences of the data until the data are stationary.
2. Examine the ACF/PACF: Is an AR(p) or MA(q) model appropriate?
3. Try your chosen model(s), and use the AICc to search for a better model.
4. Check the residuals from your chosen model by plotting the ACF of the residuals, and doing a portmanteau test of the residuals. If they do not look like white noise, try a modified model.
5. Once the residuals look like white noise, calculate forecasts.

Steps 1-3 can be automated using the auto.arima() in R. forecast() function in R forecasts the values of differenced series in the future and add back the difference to give actual value of review count forecast.



ARIMA(2,1,2)(0,1,1) for business 4bEjOyTaDG24SY5TxsaUNQ (left) and residual plots (right)

To evaluate the model, we examine the ACF and PACF plots to make sure there are no spikes in the correlations, indicating dependency on lagged terms, and to make sure that the values lie within 95% confidence interval. When we are satisfied that the model is fit appropriately, we look at the residuals. If the histogram plot of the residuals is normally distributed, then we can say that the model is a good fit.

6.2.3 Holt - winters -

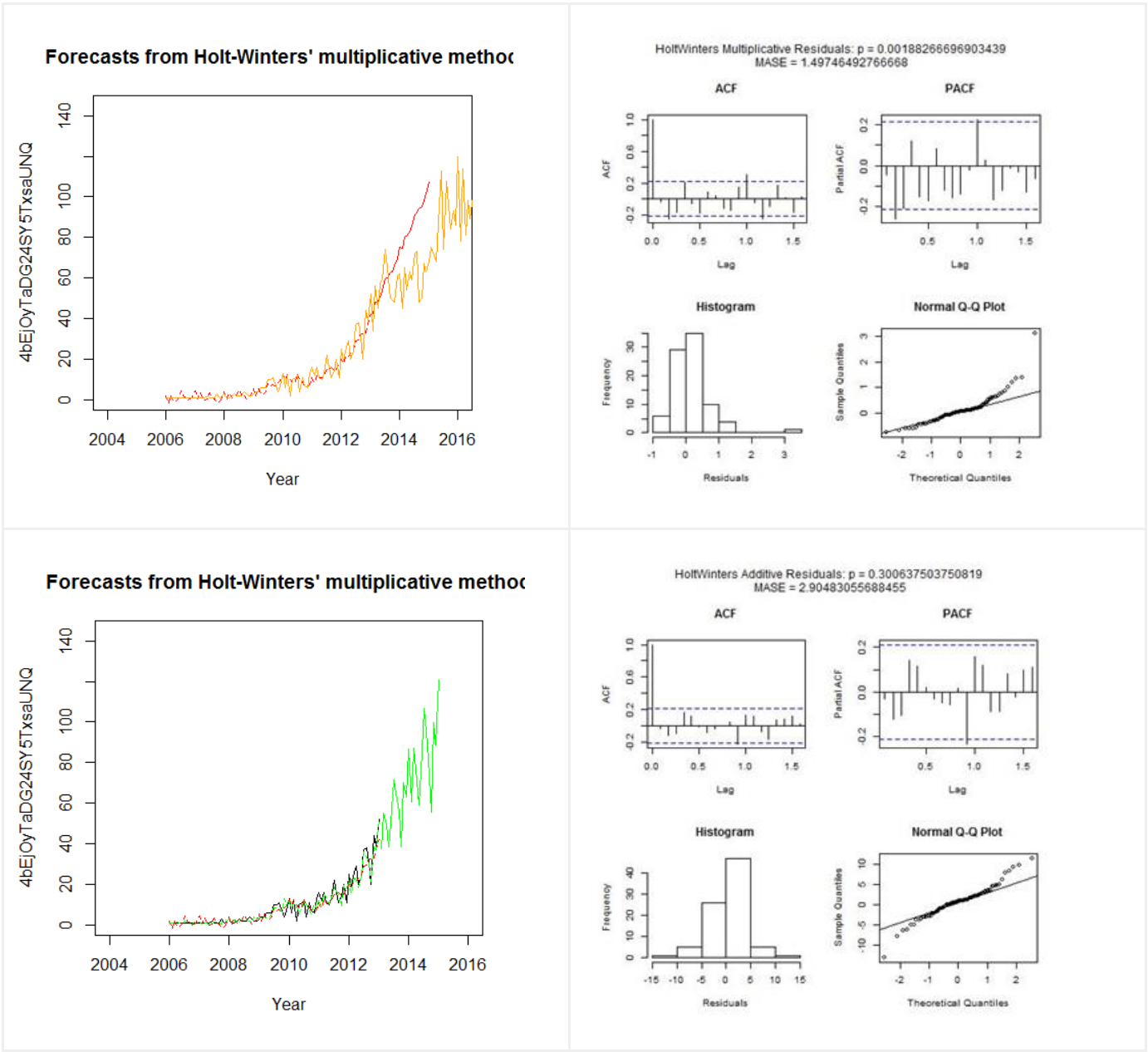
Holt winters deals with the exponential moving average which are smoothed by applying declining weights to the past.

Holt winters is a special case of exponential moving average which takes into account the forecast equation and three smoothing equations — one for the level ℓ_t, one for trend b_t and seasonal component s_t. It is a triple smoothing mechanism in which three components are smoothed to produce predictions viz. the level, trend and seasonality. These are controlled by the parameters alpha, beta and gamma. The more alpha you have the more closely you track the recent data. The time-series package in R automatically calculates the best alpha, beta and gamma by calculating mean squared prediction error.

There are two variations to this method, additive and multiplicative. Time series which exhibit constant additive seasonal variation are dealt with by Holt- winters additive and ones with multiplicative seasonal variations by Holt-winters multiplicative.

Holt Winters Additive	Holt Winters Multiplicative:
$\hat{y}_{t+h\|t} = \ell_t + hb_t + s_{t-m+h_m^+}$ $\ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1})$ $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$ $s_t = \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m},$	$\hat{y}_{t+h\|t} = (\ell_t + hb_t)s_{t-m+h_m^+}.$ $\ell_t = \alpha\frac{y_t}{s_{t-m}} + (1-\alpha)(\ell_{t-1} + b_{t-1})$ $b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1}$ $s_t = \gamma\frac{y_t}{(\ell_{t-1} + b_{t-1})} + (1-\gamma)s_{t-m}$
where level is indicated by ℓ_t, trend b_t and seasonal component s_t	

Here we see both the variations of Holt Winters on the business: 4bEjOyTaDG24SY5TxsaUNQ
By looking at the ACF, PACF, Histogram of residuals and Q-Q plot we conclude that the multiplicative Holt Winters is better than additive Holt Winters. And Holt winters multiplicative is the better performing algorithm when compared to additive in-general as well. Holt winters can also be used on damped trend and damped method prediction is considered to be the single most accurate and successful forecast methods.



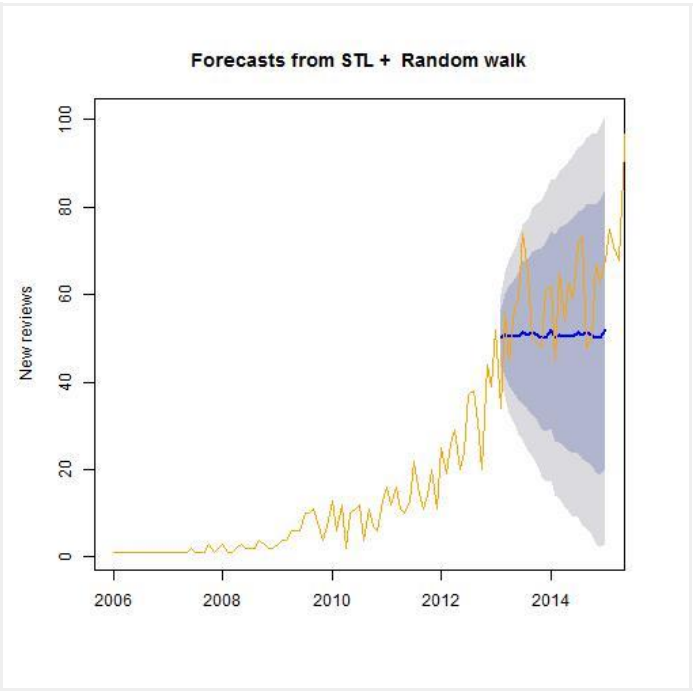
Holt-Winters' multiplicative method for business 4bEjOyTaDG24SY5TxsaUNQ (left) and residual plots (right)

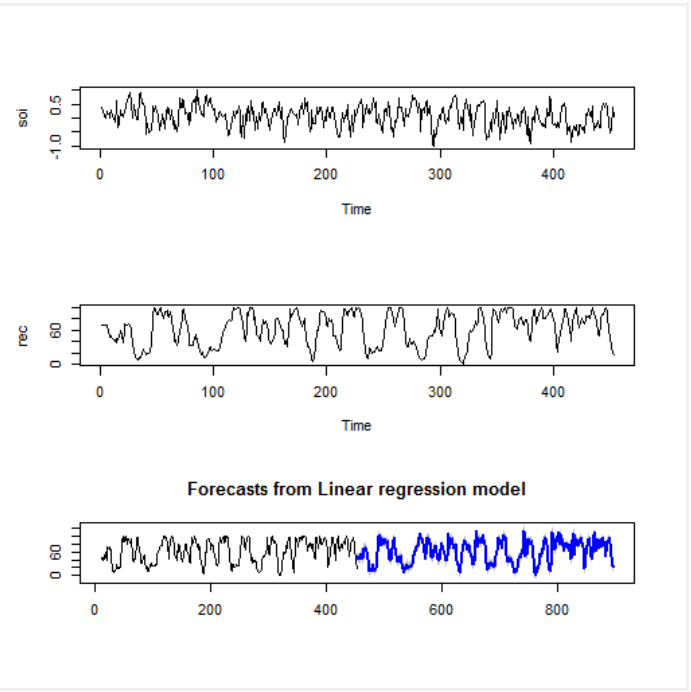
7. Experiment design and evaluation

7.1 Experiment Design:

7.1.1 STL + random Walk -

When using decomposition for studying time-series, historical changes can also be used for forecasting. We won't go into the details of the algorithm since it didn't predict good results for us, but for demonstration purposes we include this over here.





7.1.2 Prediction using Lagged Correlation:

Predictions can be done with the help of lagged correlation of other time-series as well. Lagged correlation is obtained by taking a ccf() function over the two series and building a linear model out of the terms showing the maximum correlation and taking the lag of that order w.r.t time-series. This linear model is then used to predict using the forecast package to make use of the lagged correlation of other time-series. We were able to run it on a toy-data set but not on our time-series.

7.1.3 Experiments with features -

Granularity of time-series:

When we looked at our data we first got an impression that the data could be used on per-day basis without any aggregation. But that wasn't the case. We then varied the granularity of time-series to weekly data and still saw many weeks with no-activity. Finally month data, as the granularity of series, gave the best results and hence we continued with aggregating data over monthly time-series.

Types of features:

a. **Popularity Index:**

We defined another measure to try out our predictions using, which we labeled as popularity index, which takes into account the rating of a review and the number of useful/cool counts given by different users. We consider 70% weights of the rating star with useful counts + 20% weights by cool and 10% by default of the number of stars to a review. This number is then normalized by the review stars.

Popularity Index = (0.7 * review_stars * useful_count
+ 0.2 * review_stars * cool
+ 0.1 * review_count) / review_stars

This experiment didn't provide us with useful insight for business busyness.

b. **Tip Count:**

Tips are similar to review in terms of providing insight into the activity at an establishment. We can use tips like reviews, to predict a trend. This will be another way of verifying the result by comparing the predicted trend using both reviews and tips. We later noticed the tips are sparse for business and do not provide much insight in the data.

7.1.4 Observations

We tried time-series forecasting on other datasets like "Airline Passenger Data" and "Sunspot Monthly time-series" and found that when the data is sampled over multiple decades the forecasting models worked better.

Within yelp dataset as well we observe businesses which are established since 2004 show better forecasts than businesses established from 2010.

7.2 Evaluation Measures

7.2.1 Residual - The residual data of the simple linear regression model is the difference between the observed data of the dependent variable y and the fitted values ŷ.

$$\text{Residual} = y - \hat{y}$$

7.2.2 MASE (Mean Absolute Scaled Error) - MASE is a scale-free measurement of forecast accuracy, given by

$$\text{MASE} = \frac{1}{n}\sum_{t=1}^{n}\left(\frac{|e_t|}{\frac{1}{n-1}\sum_{i=2}^{n}|Y_i - Y_{i-1}|}\right) = \frac{\sum_{t=1}^{n}|e_t|}{\frac{n}{n-1}\sum_{i=2}^{n}|Y_i - Y_{i-1}|}$$

where the numerator e_t is the forecast error for a given period, defined as the actual value (Y_t) minus the forecast value (F_t) for that period: $e_t = Y_t - F_t$, and the denominator is the average forecast error of the one-step "naive forecast method", which uses the actual value from the prior period as the forecast: $F_t = Y_{t-1}$

7.2.3 ACF (Autocorrelation Function) - The ACF of the residuals for a model can be used to identify if there are any significant correlations for any lag.. Autocorrelation is the correlation of a variable with itself at differing time lags i.e. for the series x_t and lagged values of the series for lags of 1, 2, 3, etc. The lagged values can be written as x_{t-1}, x_{t-2}, x_{t-3}, and so on. The ACF gives correlations between x_t and x_{t-1}, x_t and x_{t-2},, etc.

ACF shows the lag on horizontal and autocorrelation on y-axis. We want no significant autocorrelation in residuals, indicated by the 95% confidence interval.

7.2.3 PACF (Partial Autocorrelation Function) - The PACF measures the linear correlation of a series { x_t } and a lagged version of itself { x_{t+k} } with the linear dependence of { x_{t-1}, x_{t-2},...., $x_{t-(k-1)}$ } removed. PACF of residuals should have non-significant values for all lags. Sometimes we see unusual residual autocorrelation at certain lags - which we can ignore. But significant values at important lags should not be ignored and usually mean model is not right.

7.2.4 Histogram of Residuals - The histogram of residuals should be normally distributed. The histograms tells us how well our sample can predict the normal distribution in the population. In some case, due to outliers, the histogram is skewed towards left or right.

7.2.5 P-P Plot - The P-P Plot, also called as Q-Q plot is used to check normal assumption for errors. The plot should be near straight line for normal errors. The plot is made of residual values vs fitted values.

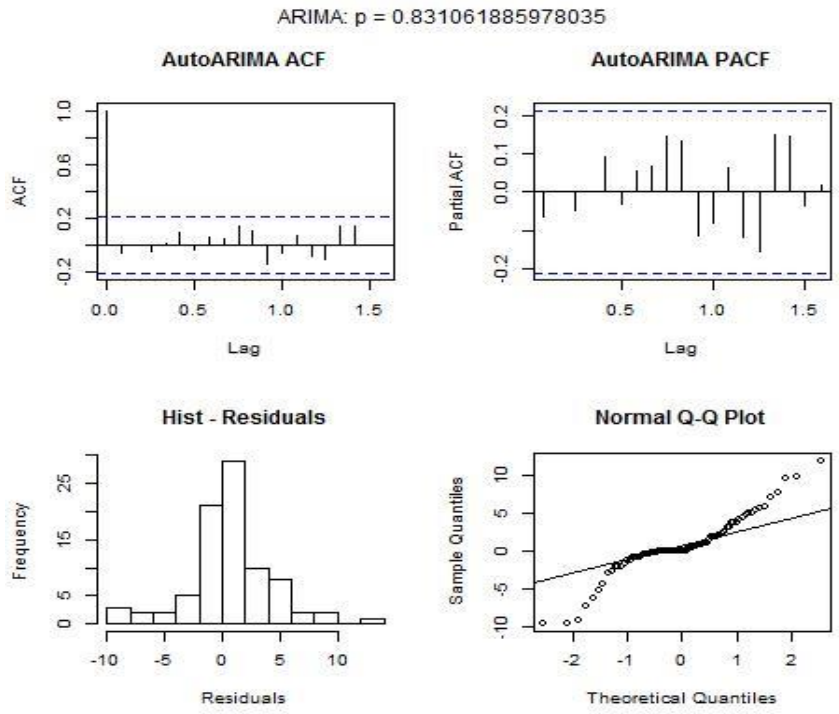
7.2.6 Akaike Information Criterion (AIC) - The Akaike information criterion (AIC) is a measure of the relative quality of statistical models for a given set of data. The model which is considered to be best according to AIC is the one that minimizes the Kullback-Leibler distance between the model and the truth. It's based on information theory, but a heuristic way to think about it is as a criterion that seeks a model that has a good fit to the truth but few parameters.

$$\text{AIC} = -2\,(\ln(\text{likelihood})) + 2K$$

Where likelihood is the probability of the data given a model and K is the number of free parameters in the model. AIC is a number that is helpful for comparing models as it includes measures of both how well the model fits the data and how complex the model is.

7.2.7 Ljung-Box Test - The Ljung-Box also helps us identify if the residuals are independent of, with the help of p-value, which we want for identifying if the model we select is correct. The Ljung-Box test is used to check if the residuals from a time series resembles white noise - ARIMA (0,0,0). A threshold of 0.05 is kept for p-value which indicates if the residuals are independent

Assume the plots for a particular businesses: 4bEjOyTaDG24SY5TxsaUNQ



Let's look at the plots on ARIMA model residuals.

ACF Plot: The ACF plot do not show any significant values for any lags above the 95% confidence interval.

PACF Plot: The PACF plots do not show any correlation or significant values above the interval.

Histogram of Residuals: The histogram of residuals for the ARIMA model residuals are normally distributed.

Q-Q Plot: The Q-Q plot shows the residual vs fitted values, showing linear trend.

P-value: The p-value obtained from Ljung-Box is greater than 0.05 indicating residuals are independent. All these measures indicate that the model is suitable for this data.

8 Conclusion

Lowest MASE values for predicted models of different businesses will give the best forecast for the respective business. Hence, businesses should try out different models and use AIC and MASE for each model as a guide.

Normally, the data used is dense and spans over multiple decades. We use review counts per month, which are less dense, to get an estimate of the activity. Over a period of time, more frequent reviews or check-ins with dates will help improve forecast accuracy.

Use of our project:

- For any business to get an estimate of upcoming traffic and plan for the future
- For business owners to assess the competition
- View trends of similar businesses

In general, we observed that for Yelp academic dataset Seasonal ARIMA models and Holt-Winters multiplicative models give a better prediction for our data.

9. References:

- Model Evaluation: https://www.otexts.org/fpp/2/5
- Yelp Dataset Challenge: http://www.yelp.com/dataset_challenge
- ARIMA: Avoid Herd in ARIMA
- Statsmodels ARMA example: statsmodel-arma
- ARIMA Simplified: http://www.johnnylogic.org/?p=688
- Time Series - Definition of ARIMA models: ARIMA Models
- AIC: https://en.wikipedia.org/wiki/Akaike_information_criterion
- STL: STL model STL example
- Mean Absolute Scaled Error (MASE): http://robjhyndman.com/papers/foresight.pdf
- Forecasting: principles and practice (Online book): https://www.otexts.org/fpp/