Restaurant Visitor Forecasting Project

Singh, Gurjeet

February 11, 2018

Contents

1. Introduction

The purpose of this project is to predict the total number of visitors to a restaurant for future dates. This will help restaurants to know how many customers to expect each day to effectively purchase ingredients and schedule staff members.

The data consists of eight (8) relation files that come from two separate sites: "Hot Pepper Gourmet (hpg): similar to Yelp where users can search restaurants and also make a reservation online" and "AirREGI / Restaurant Board (air): similar to Square, a reservation control and cash register system". The training data covers the dates from 2016 until April 2017. The test set covers the last week of April and May of 2017. The test set intentionally spans a holiday week in Japan called the "Golden Week." There are days in the test set where the restaurants were closed and had no visitors. These are ignored in scoring.

The eight (8) relation files are described below:

- air_reserve.csv
  - This file contains reservations made in the air system.
- hpg_reserve.csv
  - This file contains reservations made in the hpg system.
- air_store_info.csv
  - This file contains information about select air restaurants. Column names and contents are self-explanatory.
- hpg_store_info.csv
  - This file contains information about select hpg restaurants. Column names and contents are self-explanatory.
- store_id_relation.csv
  - This file allows you to join select restaurants that have both the air and hpg system.
- air_visit_data.csv
  - This file contains historical visit data for the air restaurants.
- sample_submission.csv
  - This file shows a submission in the correct format, including the days for which you must forecast.
- date_info.csv
  - This file gives basic information about the calendar dates in the dataset along with the Japanese holidays.

For our purposes, we have used air_visit_data.csv and data_info.csv files. The air_visit_data.csv contains information on 829 stores (restaurants) and the number of visitors per day. The data_info.csv contains the Japanese holidays' information from January 2016 to May 2017 (for both the training and test set). For certain models, we have further split the training set into train and test using 70/30 split approach. These will be discussed later.

2. Data Exploration and Preparation

The first step towards any modeling project is the Exploratory Data Analysis (EDA). This helps us to understand and analyze the dataset to summarize the main characteristics of variables. For our purposes, we will look into the basic statistics to understand the data and examine the distributions of air_visit_data.csv and data_info.csv files.

2.1 Data Overview

Table 1 shows the summary statistic of the air visit dataset. The air visit data contains the total number of visitors per day for each store. We notice that the data range from January 1, 2016, and April 22, 2017. There are a total of 829 different stores in the dataset. The statistics show that there are outliers present in the data. For example, the minimum number of visitors is 1 and the maximum number of visitors is 877. These will need to be fixed prior to our modeling building.

Table 1: Summary Statistic: Air Visit Data

unique_store_id	air_store_id	visit_date	visitors
829	Length:252108	Min. :2016-01-01	Min. : 1.00
	Class :character	1st Qu.:2016-07-23	1st Qu.: 9.00
	Mode :character	Median :2016-10-23	Median : 17.00
		Mean :2016-10-12	Mean : 20.97
		3rd Qu.:2017-01-24	3rd Qu.: 29.00
		Max. :2017-04-22	Max. :877.00

Table 2 shows the structure of the data with first three (3) values. The store id starts with "air_" and is in the character data type, visit date is the date datatype and is in the year, month, and day format. Lastly, visitors field is in the integer data type and contains the number of visitors.

Table 2: Data Structure: Air visit data with first three values

variable	class	first_values
air_store_id	character	air_ba937bf13d40fb24, air_ba937bf13d40fb24, air_ba937bf13d40fb24
visit_date	date	2016-01-13, 2016-01-14, 2016-01-15
visitors	integer	25, 32, 29

Table 3 shows the summary statistic of the holiday data. We have holiday flags from January 1, 2016, to May 31, 2017. For holiday flag field is marked as "1" if it's a holiday.

Table 3: Summary Statistic: Holiday Data

calendar_date	day_of_week	holiday_flg
Min. :2016-01-01	Length:517	Min. :0.0000
1st Qu.:2016-05-09	Class :character	1st Qu.:0.0000
Median :2016-09-15	Mode :character	Median :0.0000
Mean :2016-09-15		Mean :0.0677
3rd Qu.:2017-01-22		3rd Qu.:0.0000
Max. :2017-05-31		Max. :1.0000

Table 4 shows the summary statistic of the first ten stores. The table shows the number of observations, missing values, minimum, maximum, mean, and median values for each store.There are 478 days of data for each store starting from January 1, 2016 to April 22, 2017. The confidence interval is 95%. Looking at the minimum and first quartile values, we see that there a big gap between these two numbers. Hence, indicating that there are outliers present in the data.

Table 4: Summary Statistic for first ten Store

	nobs	NAs	Minimum	Maximum	1. Quartile	3. Quartile	Mean	Median
air_00a91d42b08b08d9	478	246	1	99	18.00	34	26.081897	26
air_0164b9927d20bcc3	478	329	1	27	4.00	12	9.248322	8
air_0241aa3964b7f861	478	82	1	48	6.00	13	9.896465	9
air_0328696196e46f18	478	362	1	41	4.00	10	7.939655	6
air_034a3d5b40d5b1b1	478	227	1	116	6.00	20	14.828685	12
air_036d4f1ee7285390	478	197	4	188	14.00	29	22.455516	21
air_0382c794b73b51ad	478	180	1	74	18.00	30	23.687919	23
air_03963426c9312048	478	49	2	119	20.00	58	39.340326	36
air_04341b588bde96cd	478	6	1	119	24.75	47	35.870763	34
air_049f6d5b402a31b2	478	220	1	29	6.00	16	11.531008	11

2.2 Visualizations

Here we look at the distribution of air visit and date info data file.

2.2.1 Air Visit Data

Figure belo shows the missing values of the first ten (10) stores. On the left of the graph, we see the number of missing records per store. On the right, we see the missing values in the combination of other stores. Red color shows the missing values.

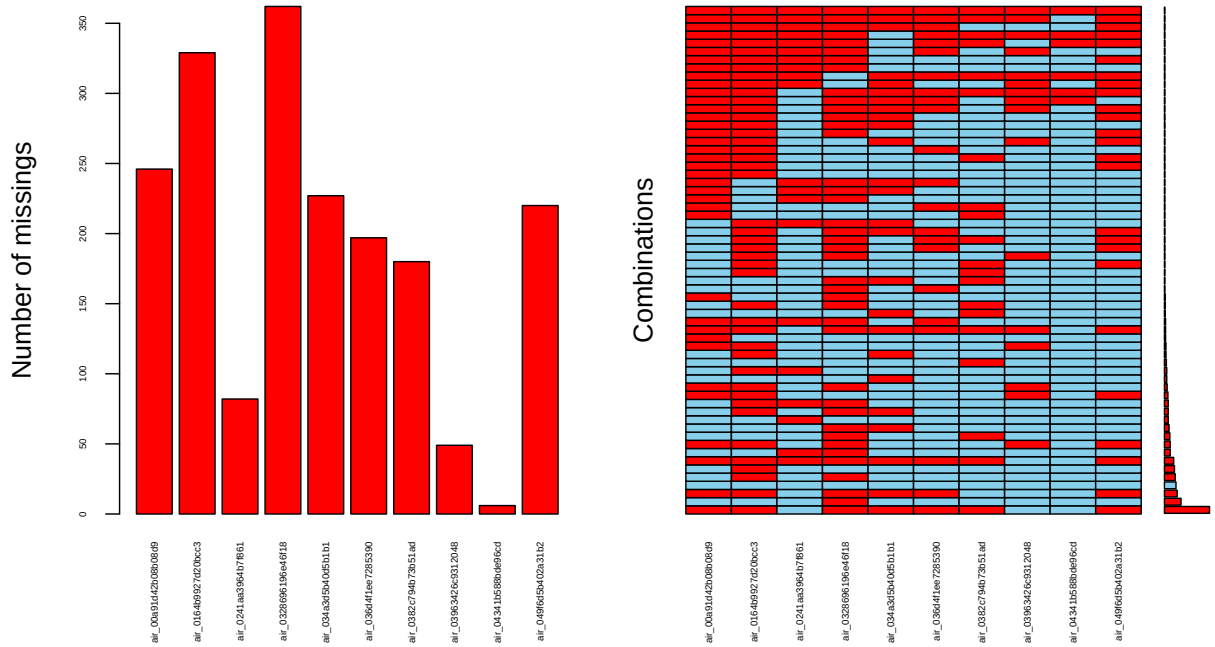


Figure below shows the missing values of the whole dataset. It is quite visible that a lot of the stores are missing values at the beginning of 2016. This means either the restaurant wasn't open at that or was closed due to some reasons. We will use this knowledge for our data preparation and model building.

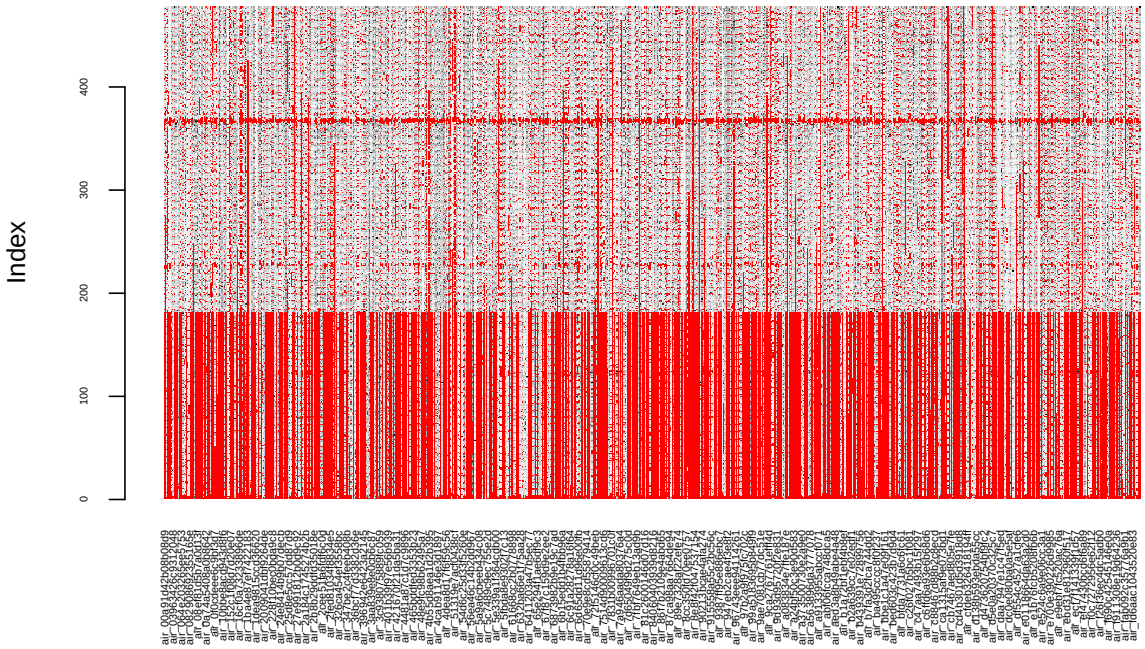
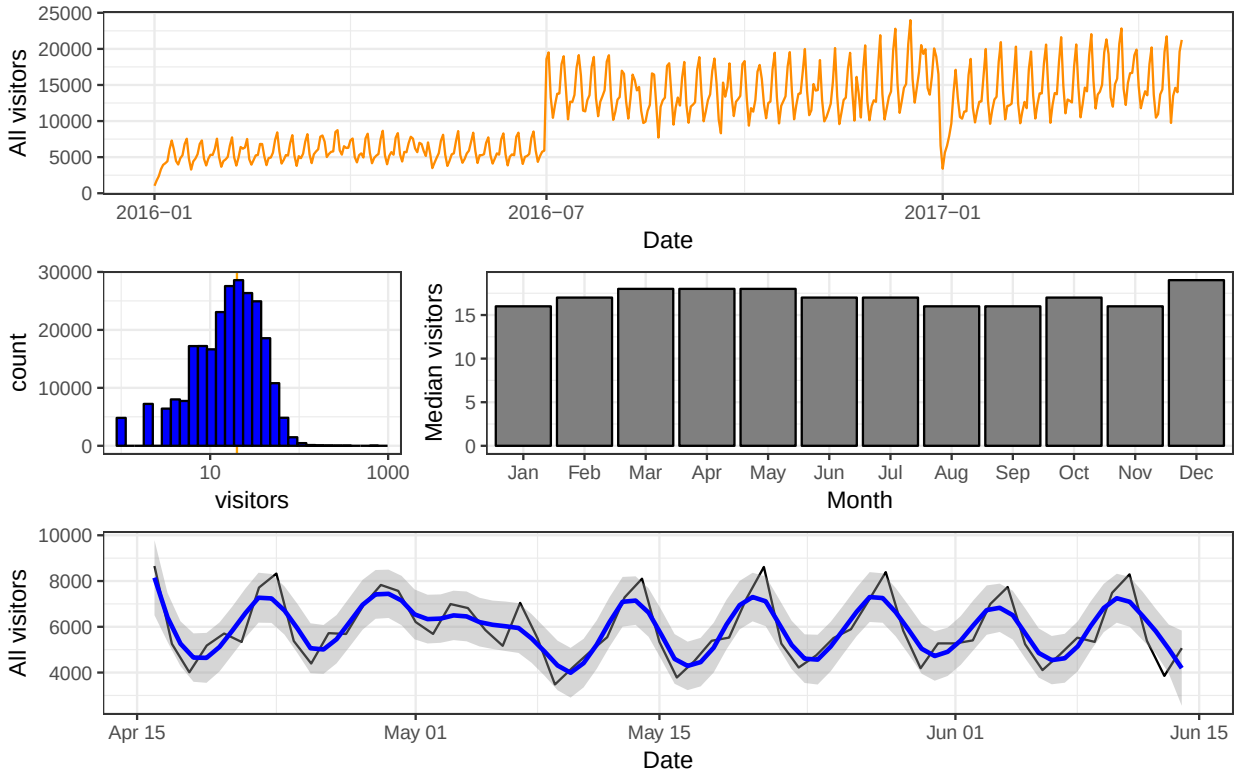


Figure below shows the plot of the total number of visitors per day from January 1, 2016 to April 22, 2017, the histogram of the total visitors, the median visitors per day of the month of the year, and review the visitors information in our 2016 training data for the same period of our forecasting in 2017.

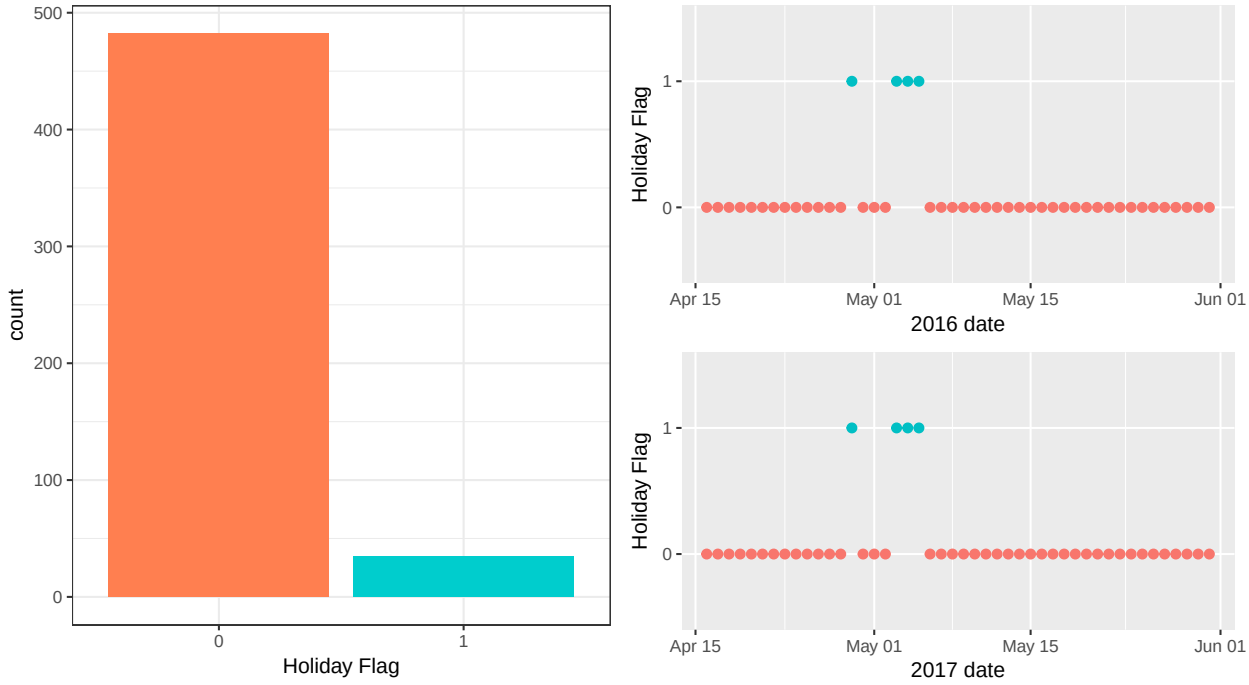
In the figure below, we notice that there is a seasonal periodic pattern to the data. It appears that it's a weekly pattern and is pretty consistent throughout the data. We do not see a strong increasing trend in the plot. However, the increase in the visitors starting July 2016 could be due to the opening of the new restaurants or some kind of addition of the dataset to the current database from different systems. We also find out that December is the busiest month of the year followed by March, April, and May which are also consistently busy.

Lastly, the bottom graph shows the effects of the "Golden Week" in 2016 between April 29 and May 5. We again see the weekly seasonal pattern. However, during the Golden week, we see the decline in the visitors right around the Golden Week holiday i.e. April 29. The decline continued until the end of the first week. Visitors traffic picked up around the second week.



2.2.2 Holiday Data

The figure below shows the plot of the holiday data. It appears that there are about 30 - 40 holiday days in the entire data set. The graphs on the right show how the holidays are distributed during the prediction time range in 2017 and the corresponding time in 2016. We notice that the holidays fall on the same days in 2017 as well.



2.3 Data Preparation

From our initial exploration, we know that we have missing values and outliers in the data. Before we proceed to build models, we will impute the missing values and also fix the outliers. For our purposes, we have taken multiple approaches to clean the dataset. Each of these approaches is discussed below:

- Approach 1:
  - Formatted the initial air visit data so all the stores appear in each individual columns.
  - Removed those 8 stores that are not part of the test set.
  - Added the date info (holiday) data to our initial dataset.
  - Replaced all the NAs, prior to the first value, with zero (0) for each store.
  - Replaced all NAs with zero (0) occurring every 7th day.
    - We assumed that if NA is consistent every 7th day from the first value, it means that the store is closed that day every week.
  - Replaced all NAs with zero if those days fall on a holiday.
  - Replaced all NAs with mean, median, linear, and exponential imputation.
  - Converted the dataset into a time series (Each imputed dataset converted into time series data).
- Approach 2:
  - Formatted the initial air visit data so all the stores appear in each individual columns.
  - Removed those 8 stores that are not part of the test set.
  - Added the date info (holiday) data to our initial dataset.
  - Replaced all NAs with zero (0) occurring every 7th day.
    - We assumed that if NA is consistent every 7th day from the first value, it means that the store is closed that day every week.
  - Replaced all NAs with zero if those days fall on a holiday.
  - Replaced all NAs with median imputation.
  - Fix the outliers in the data from the first value to the last record.
  - Converted the dataset into a time series.
- Approach 3:
  - Formatted the initial air visit data so all the stores appear in each individual columns.
  - Removed those 8 stores that are not part of the test set.
  - Added the date info (holiday) data to our initial dataset.
  - Replaced all NAs with zero (0) occurring every 7th day.
    - We assumed that if NA is consistent every 7th day from the first value, it means that the store is closed that day every week.
  - Replaced all NAs with zero if those days fall on a holiday.
  - Replaced all NAs with median imputation.
  - Fix the outliers in the data from the first value to the last record.
  - Converted the dataset into a time series.
  - Split the training data further into training and test dataset with 70/30 split approach.

3. Forecasting Methods

We turn our focus to the time-series forecasting methods. For our purposes, we have stayed with the basic forecasting methods i.e. Auto ARIMA and ETS methods. However, we took multiple approaches to clean the data and rerun these methods. For this project requirements, we are forecasting for 39 days in future (i.e. April 23rd - May 31st)

3.1 Model: Approach 1

For our first approach to building models, we cleaned the data as listed in Section 2.3 under "Approach 1". We then used auto.arima and ets (auto ets) methods to forecast for each kind of imputation i.e. mean, median, linear, and exponential imputation.

The result below is from the Kaggle submission. We see the Auto ARIMA not only performed well in the public score with the score: 0.581 but also in the private score with the score: 0.617. When comparing the private score, we notice that ETS models with mean and median imputations performed well with the score difference of 0.001 and 0.002 respectively.

Approach 1 Kaggle Submission Results:

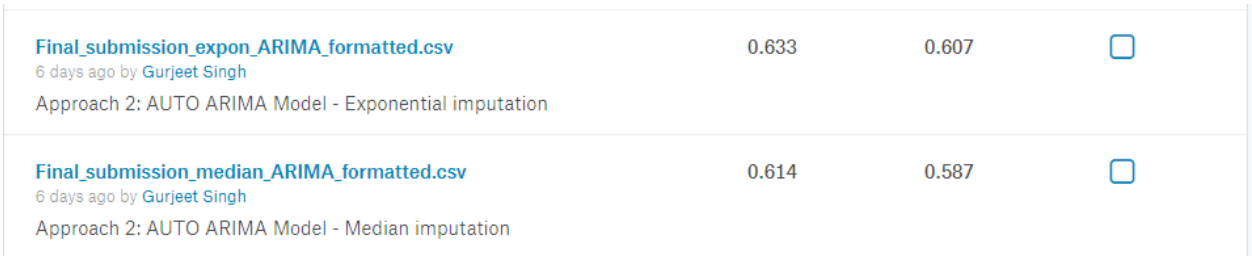
Final_submission_median_ARIMA_formatted.csv 7 days ago by Gurjeet Singh Approach 1: AUTO ARIMA Model - Median imputation	0.617	0.581	☐
Final_submission_expon_ETS_formatted.csv 7 days ago by Gurjeet Singh Approach 1: ETS Model - Exp imputation - ABS for non-neg. values	0.623	0.614	☐
Final_submission_expon_ETS_formatted.csv 7 days ago by Gurjeet Singh Approach 1: ETS Model - Exponential imputation	0.634	0.621	☐
Final_submission_median_EST_formatted.csv 7 days ago by Gurjeet Singh Approach 1: ETS Model - Median Imputation	0.618	0.605	☐
Final_submission_mean_EST_formatted.csv 7 days ago by Gurjeet Singh Approach 1: ETS Model - Mean Imputation	0.619	0.606	☐
Final_submission_linear_EST_formatted.csv 7 days ago by Gurjeet Singh Approach 1: EST Model - Linear Imputation	NULL	Error	☐
Final_submission_expon_EST_formatted.csv 7 days ago by Gurjeet Singh Approach 1: EST Model - Exponential imputation	NULL	Error	☐

3.2 Model: Approach 2

For our second approach to building models, we cleaned the data as listed in Section 2.3 under "Approach 2". This time we only used auto.arima methods to forecast the data with median and exponential imputation. Due to the poor performance of the model with exponential imputation, we dropped that model and only focused on the median imputed dataset.

The result below is from the Kaggle submission for the second approach. We see the Auto ARIMA with the median imputation performed better in both public and private score. This approach even performed better than our first approach in the private score with the score: 0.614.

Approach 2 Kaggle Submission Results:

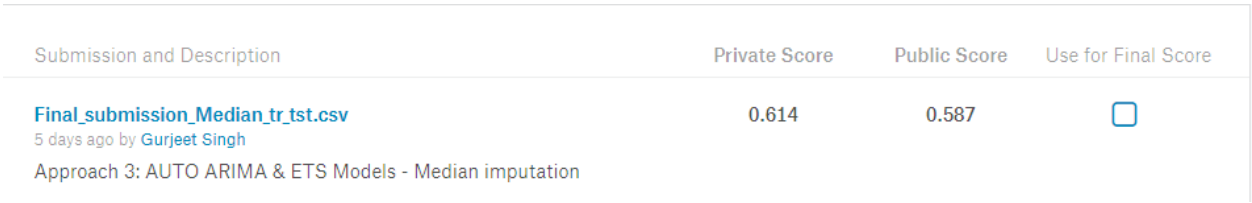


3.3 Model: Approach 3

For our third approach to building models, we cleaned the data as listed in Section 2.3 under "Approach 3". In this approach, we used the combination of auto.arima and ets methods together to forecast the data with median imputation. If the Mean Absolute Scaled Error (MASE) is lower for the model using the test set (split from the actual training set using 70/30 split approach), we would select that model. Otherwise, we would select the other model. We did this for all the stores.

The result below is from the Kaggle submission for the third approach. We notice that the public and private scores for approach three is similar to the one we got in approach 2.

Approach 3 Kaggle Submission Results:



4. Literature

For the purpose of this project, we have reviewed five kinds of literature available from the Northwestern library to understand how the models we selected were used in the similar situations. The five kinds of literature are listed below:

1. Building Forecasting Models for Restaurant Owners and Managers: A Case Study
   - The author uses Exponential Smoothing as one of the methods. Similar to us, the author uses ETS model as the base model. The author states that it possible for ETS to produce sluggish forecasts that do not react quickly to changes in data or forecasts that react too quickly to changes in the data based on the value of the smoothing constant. Hence, forecasts often leave with a large possible range of values.
2. Data Mining on Time Series
   - The author uses Box-Jenkins season ARIMA models. It was interesting to know about this model and how it is very useful in capturing the behavior of seasonal time series and generating accurate forecasts for such series. In our dataset, we have noticed the seasonal pattern. So, it would be ideal to build and use this model to compare with our auto.arima model.
3. Forecasting Restaurant Sales Using Multiple Regression and Box-Jenkins Analysis
   - The author uses Box-Jenkins models and several regression models to forecast weekly sales at a small campus restaurant for two years. Per the article, it seems like regression model beats the Box-Jenkins model.
4. A Hybrid Seasonal Autoregressive

- The author have used SARIMA, SARIMAX, Quantile Regression, and Cross-Validation to forecast daily sales of a perishable food. Not knowing about any of the methods, it was interesting to read this article and understand how the author solve this problem.

5. Time-Series Forecasting: Food-Service Industry
    - This case study examines the use of forecasting models for the food-service industry. There are three Time-Series forecasting methods are discussed: 1) Past Average Demand, 2) Single Exponential Smoothing and 3) Double Exponential Smoothing. It was quite interesting to read about the difference between single exponential smoothing and double exponential smoothing. The author states that single exponential smoothing is more appropriate if the firm requires a forecast of one month ahead. However, the author states that if the data is non-stationary and a multi-period forecast is required, double exponential smoothing is more appropriate.

5. Conclusion

In conclusion, we would like to state that we created multiple models using a dataset with various imputation. For our best model, we had a tie with our approach 2 and approach 3 models. However, we selected approach 2 ARIMA model with median imputation as our final model because it was less complicated than the model from approach 3.

One of the limitations of our model is that it totally relies on the system since we created our model using auto.arima function. The second limitation that we encountered was the limited knowledge of working with multiple time series data. It was really difficult to compare and select models prior to submitting to the final results to Kaggle. Hence, we relied on Kaggle to test our results for model selection. The other limitation we had was the slow processing of the models. Each run took a lot of runtimes which slowed us down from experimenting other approaches.

Due to the limitation of the time, we could not experiment a lot of different approaches. Hence, we plan to improve our models in future by taking the following steps:

1. Use a different methodology for imputing missing values. One of the options that we want to try is to use the Last Observation Carried Forward approach (na.locf function from the imputeTS package).
2. For each time series (for each restaurant) model, capture the MASE in a variable. Take the average of the MASE and then compare the value with other models. The model that has the best average MASE value, use that as the final model.
3. During my research, we came across "Prophet" modeling approach for forecasting. This is the open-source software developed by Facebook. We would like to try this modeling approach and compare our results.
4. During the literature review, we came across other methods (Box-Jenkins, SARIMA, SARIMAX, and Quantile Regression). We hope to use those to improve our models.
5. Lastly, the processing of our model was really bad. It took forever to produce results. Therefore, we plan to use "DoParallel" in future.

This whole project was a great learning experience. This gave me exposure on how to work with the multiple time series data, experience with Kaggle competition, interact with classmates to learn and understand their approaches. Lastly, this project required a lot of R coding which required a lot of research and helped us learn a few new R packages (lubridate, VIM, imputeTS) that we have never used before.

---

6. References

Kaggle Kernels and Discussion boards: (https://www.kaggle.com/c/recruit-restaurant-visitor-forecasting)
Missing Values Visualization: (http://rstudio-pubs-static.s3.amazonaws.com/4625_fa990d611f024ea69e7e2b10dd228fe7.html)

7. Appendix I: R Code

```
knitr::opts_chunk$set(echo = TRUE, warning = F, message = F)

##-------------------
# Load Libraries
##-------------------
options(knitr.kable.NA = '')
library(tidyverse)
library(dplyr)
library(lubridate)
library(fpp)
options(scipen = 999)
##-------------------
# Import Data
##-------------------

#Import air_visit data
visit.data.df <-
  read_csv("Restaurant_Visitor_Forecasting_air_visit_data.csv")
#Loading holiday data
date.info.df <-
  read_csv("Restaurant_Visitor_Forecasting_date_info.csv")
##-------------------
# Data Exploration
##-------------------
sumStat <- summary(visit.data.df)
unIds <- c(nrow(distinct(visit.data.df[,1])),
           NA, NA, NA, NA, NA)
knitr::kable(cbind("unique_store_id"=unIds,sumStat),
             caption = "Summary Statistic: Air Visit Data")

variable <- names(visit.data.df)
class <- sapply(visit.data.df, typeof)
  class[2] <- "date"
first_values = sapply(visit.data.df, function(x) paste0(head(x,3),  collapse = ", "))

data.frame(variable, class, first_values, row.names = NULL) %>%
  knitr::kable(caption = "Data Structure: Air visit data with first three values")
knitr::kable(summary(date.info.df),
             caption = "Summary Statistic: Holiday Data")
#format the data from rows to columns
visit.df <-   visit.data.df %>%
              spread(air_store_id, visitors) %>%
              separate(visit_date, c("Year","Month","Day"))

knitr::kable(t(fBasics::basicStats(visit.df[,4:13])[c(1,2,3,4,5,6, 7,8),]),
             caption = "Summary Statistic for first ten Store")
##-------------------
# Data Visualizations
##-------------------
#, fig.height=5, fig.width=9, fig.cap="Air Visit - Missing values of First ten stores"}
VIM::aggr(visit.df[,4:13],
```

```
     prop = F,
     numbers = T,
     cex.axis = 0.45)
VIM::matrixplot(visit.df[,4:829], interactive = F, cex.axis = 0.4)
plot1 <- visit.data.df %>%
          group_by(visit_date) %>%
          summarise(all_visitors = sum(visitors)) %>%
          ggplot(aes(visit_date,all_visitors)) +
          geom_line(col = "dark orange") +
          labs(y = "All visitors", x = "Date") +
          theme_bw()


plot2 <- visit.data.df %>%
              ggplot(aes(visitors)) +
              geom_vline(xintercept = 20, color = "orange") +
              geom_histogram(fill = "blue", bins = 30, colour = "black") +
              scale_x_log10()+
          theme_bw()

plot3 <- visit.data.df %>%
            mutate(month = month(visit_date, label = T)) %>%
            group_by(month) %>%
            summarise(visits = median(visitors)) %>%
            ggplot(aes(month, visits)) +
            geom_col(show.legend = F, fill = "grey50", colour = "black") +
            theme_bw() +
            labs(x = "Month", y = "Median visitors")

plot4 <- visit.data.df %>%
          filter(visit_date > ymd("2016-04-15") & visit_date < ymd("2016-06-15")) %>%
          group_by(visit_date) %>%
          summarise(all_visitors = sum(visitors)) %>%
          ggplot(aes(visit_date,all_visitors)) +
          geom_line() +
          geom_smooth(method = "loess", color = "blue", span = 1/7) +
          labs(y = "All visitors", x = "Date") +
          theme_bw()

gridExtra::grid.arrange(plot1, plot2,plot3, plot4, layout_matrix = rbind(c(1, 1, 1),
                        c(2, 3,3),
                        c(4,4,4)))

holidays <- date.info.df

plot1 <- holidays %>%
  ggplot(aes(as.factor(holiday_flg), fill = holiday_flg)) +
  geom_bar(show.legend = F, fill = c("coral","cyan3")) +
  xlab("Holiday Flag") +
  theme_bw()

plot2 <- holidays %>%
  filter(calendar_date > ymd("2016-04-15") & calendar_date < ymd("2016-06-01")) %>%
```

```
  ggplot(aes(calendar_date, as.factor(holiday_flg), color = as.factor(holiday_flg))) +
  geom_point(size = 2) +
  theme(legend.position = "none") +
  labs(x = "2016 date", y = "Holiday Flag")

plot3 <- holidays %>%
  filter(calendar_date > ymd("2017-04-15") & calendar_date < ymd("2017-06-01")) %>%
  ggplot(aes(calendar_date,  as.factor(holiday_flg), color = as.factor(holiday_flg))) +
  geom_point(size = 2) +
  theme(legend.position = "none") +
  labs(x = "2017 date", y = "Holiday Flag")

gridExtra::grid.arrange(plot1, plot2,plot3, layout_matrix = rbind(c(1, 2),
                        c(1, 3)))

##-------------------
#Modeling Approach 1
##-------------------
  visit.df.rm <-  visit.df[,!(
                              names(visit.df) %in% c(
                              "air_0ead98dd07e7a82a",
                              "air_229d7e508d9f1b5e",
                              "air_2703dcb33192b181",
                              "air_b2d8bc9c88b85f96",
                              "air_cb083b4789a8d3a2",
                              "air_cf22e368c1a71d53",
                              "air_d0a7bd3339c3d12a",
                              "air_d63cfa6d6ab78446"
                              )
                              )]

  #combining holiday data
  visit2.df <- as.tibble(cbind.data.frame(
                          date.info.df[1:478,1:3],
                          visit.df.rm[1:478,]))

  #Creating a working dataset
  visit.working.df <- visit2.df

  #Multiple dataframe with different imputation
  visit.mean.mean.df      <- visit.working.df[,1:6]
  visit.mean.median.df    <- visit.working.df[,1:6]
  visit.in.linear.df      <- visit.working.df[,1:6]
  visit.ma.expon.df       <- visit.working.df[,1:6]

  #Imputation to remove missing values
  for(i in 7:ncol(visit.working.df)){
    #step1 - get all the missing values for each column
    rowStart <- which(!is.na(visit.working.df[,i]))
    #step2 - replace all the NA to 0 until first value encountered
    visit.working.df[1:min(rowStart)-1,i] <- 0
    #step3 - Get all the rows with a value.
    actNa <- which(is.na(visit.working.df[,i]))
```

```
    if(length(actNa) != 0){
        #step4: Replace all the values for NA if store has NA occuring in 7th day
               # this could mean it closes 1 day a week.
        for(k in 1:length(actNa)){
          if(k+1 <= length(actNa)) {
            if(abs(actNa[k] - actNa[k+1]) == 7){
              visit.working.df[actNa[k],i] <- 0
            }
          }
        }
        #Step5 - Get all the left over missing values
        holCheckNA <- which(is.na(visit.working.df[,i]))
        #step6: If the missing value is due to a holiday, add it as 0
        for(l in 1:length(holCheckNA)){
              #l<-2
          if(visit.working.df[holCheckNA[l],3] == 1){
            visit.working.df[holCheckNA[l],i] <- 0
          }

        }
    }
    #Step6 - Chec for final NA for imputation
    visit.mean.mean.df[,i] <- round(imputeTS::na.mean(
                                    as.numeric(
                                      unlist(visit.working.df[,i])),
                                        option = "mean"),0)
    visit.mean.median.df[,i] <- round(imputeTS::na.mean(
                                    as.numeric(
                                      unlist(visit.working.df[, i])),
                                        option = "median"), 0)
    visit.in.linear.df[,i] <- round(imputeTS::na.interpolation(
                                    as.numeric(
                                      unlist(visit.working.df[,i])),
                                        option = "linear"),0)
    visit.ma.expon.df[,i] <- round(imputeTS::na.ma(
                                    as.numeric(
                                      unlist(visit.working.df[,i])),
                                        weighting = "exponential"),0)
}
colnames(visit.mean.mean.df) <- colnames(visit.working.df)
colnames(visit.mean.median.df) <- colnames(visit.working.df)
colnames(visit.in.linear.df) <- colnames(visit.working.df)
colnames(visit.ma.expon.df) <- colnames(visit.working.df)

#define the time series
visit.mean.ts =as.ts(visit.mean.mean.df[, -c(1,2,3,4,5,6)],
                     start=c(2016,1,1), frequency=365)
visit.median.ts =as.ts(visit.mean.median.df[, -c(1,2,3,4,5,6)],
                     start=c(2016,1,1), frequency=365)
visit.linear.ts =as.ts(visit.in.linear.df[, -c(1,2,3,4,5,6)],
                     start=c(2016,1,1), frequency=365)
visit.expon.ts =as.ts(visit.ma.expon.df[, -c(1,2,3,4,5,6)],
                     start=c(2016,1,1), frequency=365)
```

```
####-----------------
## ETS Model
####-----------------

for(i in 1:ncol(visit.mean.ts)){
  fcast.mean.est <- round(forecast(ets(visit.mean.ts[,i]),39)$mean,0)
  fcast.median.est <- round(forecast(ets(visit.median.ts[,i]),39)$mean,0)
  fcast.linear.est <- round(forecast(ets(visit.linear.ts[,i]),39)$mean,0)
  fcast.expon.est <- round(forecast(ets(visit.expon.ts[,i]),39)$mean,0)
  if(i == 1){
    visit.mean.fcast <- fcast.mean.est
    visit.median.fcast <- fcast.median.est
    visit.linear.fcast <- fcast.linear.est
    visit.expon.fcast <- fcast.expon.est
  }
  if(i > 1){
    visit.mean.fcast <- cbind(visit.mean.fcast,fcast.mean.est)
    visit.median.fcast <- cbind(visit.median.fcast,fcast.median.est)
    visit.linear.fcast <- cbind(visit.linear.fcast,fcast.linear.est)
    visit.expon.fcast <- cbind(visit.expon.fcast,fcast.expon.est)
  }
}
colnames(visit.mean.fcast) <- colnames(visit.mean.ts)
colnames(visit.median.fcast) <- colnames(visit.median.ts)
colnames(visit.linear.fcast) <- colnames(visit.linear.ts)
colnames(visit.expon.fcast) <- colnames(visit.expon.ts)

#Reformat and Export Final submission file
#Mean Imputation - EST Model
data.result.mean <- rownames_to_column(as.tibble(visit.mean.fcast),var = "RowID")
final_submission.mean <- as.tibble(data.result.mean %>%
                                mutate(Date = seq(from = as.Date("2017-04-23"),
                                                  to = as.Date("2017-05-31"),
                                                  by= 'day')) %>%
                                gather("StoreID","visitors",
                                       2:ncol(data.result.mean)) %>%
                                unite("id",StoreID,Date, sep = "_")
                              )[,2:3]
write_csv(final_submission.mean,
          "Approach_1_Final_submission_mean_ETS_formatted.csv")

#Median Imputation - EST Model
data.result.median <- rownames_to_column(as.tibble(visit.median.fcast),var = "RowID")
final_submission.median <- as.tibble(data.result.median %>%
                                  mutate(Date = seq(from = as.Date("2017-04-23"),
                                                    to = as.Date("2017-05-31"),
                                                    by= 'day')) %>%
                                  gather("StoreID","visitors",
                                         2:ncol(data.result.median)) %>%
                                  unite("id",StoreID,Date, sep = "_")
                               )[,2:3]
write_csv(final_submission.median,
          "Approach_1_Final_submission_median_ETS_formatted.csv")
```

```r
#Linear Imputation - EST Model
data.result.linear <- rownames_to_column(as.tibble(visit.linear.fcast),var = "RowID")
final_submission.linear <- as.tibble(data.result.linear %>%
                                       mutate(Date = seq(from = as.Date("2017-04-23"),
                                                         to = as.Date("2017-05-31"),
                                                         by= 'day')) %>%
                                       gather("StoreID","visitors",
                                              2:ncol(data.result.linear)) %>%
                                       unite("id",StoreID,Date, sep = "_")
                                     )[,2:3]
final_submission.linear$visitors[final_submission.linear$visitors <0] <-
  abs(final_submission.linear$visitors[final_submission.linear$visitors <0])

write_csv(final_submission.linear,
          "Approach_1_Final_submission_linear_ETS_formatted.csv")

#Exponential Imputation - EST Model
data.result.expon <- rownames_to_column(as.tibble(visit.expon.fcast),var = "RowID")
final_submission.expon <- as.tibble(data.result.expon %>%
                                      mutate(Date = seq(from = as.Date("2017-04-23"),
                                                        to = as.Date("2017-05-31"),
                                                        by= 'day')) %>%
                                      gather("StoreID","visitors",
                                             2:ncol(data.result.expon)) %>%
                                      unite("id",StoreID,Date, sep = "_")
                                    )[,2:3]
final_submission.expon$visitors[final_submission.expon$visitors <0] <-
  abs(final_submission.expon$visitors[final_submission.expon$visitors <0])

write_csv(final_submission.expon,
          "Approach_1_Final_submission_expon_ETS_formatted.csv")


####-----------------
## Auto Arima Model
####-----------------

for(i in 1:ncol(visit.mean.ts)){
  fcast.mean.aa <- round(forecast(auto.arima(visit.mean.ts[,i]),39)$mean,0)
  fcast.median.aa <- round(forecast(auto.arima(visit.median.ts[,i]),39)$mean,0)
  fcast.linear.aa <- round(forecast(auto.arima(visit.linear.ts[,i]),39)$mean,0)
  fcast.expon.aa <- round(forecast(auto.arima(visit.expon.ts[,i]),39)$mean,0)
  if(i == 1){
    visit.mean.fcast.aa <- fcast.mean.aa
    visit.median.fcast.aa <- fcast.median.aa
    visit.linear.fcast.aa <- fcast.linear.aa
    visit.expon.fcast.aa <- fcast.expon.aa
  }
  if(i > 1){
    visit.mean.fcast.aa <- cbind(visit.mean.fcast.aa,fcast.mean.aa)
    visit.median.fcast.aa <- cbind(visit.median.fcast.aa,fcast.median.aa)
    visit.linear.fcast.aa <- cbind(visit.linear.fcast.aa,fcast.linear.aa)
    visit.expon.fcast.aa <- cbind(visit.expon.fcast.aa,fcast.expon.aa)
```

```
  }
}


colnames(visit.mean.fcast.aa) <- colnames(visit.mean.ts)
colnames(visit.median.fcast.aa) <- colnames(visit.median.ts)
colnames(visit.linear.fcast.aa) <- colnames(visit.linear.ts)
colnames(visit.expon.fcast.aa) <- colnames(visit.expon.ts)


#Mean Imputation - Auto ARIMA Model
data.result.mean.aa <-
        rownames_to_column(as.tibble(visit.mean.fcast.aa),var = "RowID")
final_submission.mean.aa <- as.tibble(data.result.mean.aa %>%
                                    mutate(Date = seq(from = as.Date("2017-04-23"),
                                                      to = as.Date("2017-05-31"),
                                                      by= 'day')) %>%
                                    gather("StoreID","visitors",
                                           2:ncol(data.result.mean.aa)) %>%
                                    unite("id",StoreID,Date, sep = "_")
                                  )[,2:3]
write_csv(final_submission.mean.aa,
          "Approach_1_Final_submission_mean_ARIMA_formatted.csv")

#Median Imputation - Auto ARIMA Model
data.result.median.aa <-
     rownames_to_column(as.tibble(visit.median.fcast.aa),var = "RowID")
final_submission.median.aa <- as.tibble(data.result.median.aa %>%
                                      mutate(Date = seq(from = as.Date("2017-04-23"),
                                                        to = as.Date("2017-05-31"),
                                                        by= 'day')) %>%
                                      gather("StoreID","visitors",
                                             2:ncol(data.result.median.aa)) %>%
                                      unite("id",StoreID,Date, sep = "_")
                                    )[,2:3]
write_csv(final_submission.median.aa,
          "Approach_1_Final_submission_median_ARIMA_formatted.csv")

#Linear Imputation - Auto ARIMA Model
data.result.linear.aa <-
      rownames_to_column(as.tibble(visit.linear.fcast.aa),var = "RowID")
final_submission.linear.aa <- as.tibble(data.result.linear.aa %>%
                                      mutate(Date = seq(from = as.Date("2017-04-23"),
                                                        to = as.Date("2017-05-31"),
                                                        by= 'day')) %>%
                                      gather("StoreID","visitors",
                                             2:ncol(data.result.linear.aa)) %>%
                                      unite("id",StoreID,Date, sep = "_")
                                    )[,2:3]
final_submission.linear.aa$visitors[final_submission.linear.aa$visitors <0] <-
  abs(final_submission.linear.aa$visitors[final_submission.linear.aa$visitors <0])

write_csv(final_submission.linear.aa,
```

```
          "Approach_1_Final_submission_linear_ARIMA_formatted.csv")

#Exponential Imputation - Auto ARIMA Model
data.result.expon.aa <-
      rownames_to_column(as.tibble(visit.expon.fcast.aa),var = "RowID")
final_submission.expon.aa <- as.tibble(data.result.expon.aa %>%
                                    mutate(Date = seq(from = as.Date("2017-04-23"),
                                                      to = as.Date("2017-05-31"),
                                                      by= 'day')) %>%
                                    gather("StoreID","visitors",
                                           2:ncol(data.result.expon.aa)) %>%
                                    unite("id",StoreID,Date, sep = "_")
                                    )[,2:3]
final_submission.expon.aa$visitors[final_submission.expon.aa$visitors <0] <-
  abs(final_submission.expon.aa$visitors[final_submission.expon.aa$visitors <0])

write_csv(final_submission.expon.aa,
          "Approach_1_Final_submission_expon_ARIMA_formatted.csv")

##-------------------
#Modeling Approach 2
##-------------------
#Import air_visit data
visit.data.df <-
  read_csv("Restaurant_Visitor_Forecasting_air_visit_data.csv")
#Loading holiday data
date.info.df <-
  read_csv("Restaurant_Visitor_Forecasting_date_info.csv")

#format the data from rows to columns
visit.df <-   visit.data.df %>%
              spread(air_store_id, visitors) %>%
              separate(visit_date, c("Year","Month","Day"))

  visit.df.rm <-  visit.df[,!(
                             names(visit.df) %in% c(
                             "air_0ead98dd07e7a82a",
                             "air_229d7e508d9f1b5e",
                             "air_2703dcb33192b181",
                             "air_b2d8bc9c88b85f96",
                             "air_cb083b4789a8d3a2",
                             "air_cf22e368c1a71d53",
                             "air_d0a7bd3339c3d12a",
                             "air_d63cfa6d6ab78446"
                             )
                             )]
  #combining holiday data
  visit2.df <- as.tibble(cbind.data.frame(
                         date.info.df[1:478,1:3],
                         visit.df.rm[1:478,]))

  #Creating a working dataset
  visit.working.df <- visit2.df
```

```
 #Imputation to remove missing values
for(i in 7:ncol(visit.working.df)){
   #step1 - get all the missing values for each column
   rowStart <- which(!is.na(visit.working.df[,i]))

   #step3 - Get all the rows with a value.
   actNa <- which(is.na(visit.working.df[,i]))
   actNa <- actNa[actNa > min(rowStart)]

   if(length(actNa) != 0){
         #step4: Replace all the values for NA if store has NA occuring in 7th day
                # this could mean it closes 1 day a week.
         for(k in 1:length(actNa)){
           if(k+1 <= length(actNa)) {
             if(abs(actNa[k] - actNa[k+1]) == 7){
               visit.working.df[actNa[k],i] <- 0
             }
           }
         }

         #Step5 - Get all the left over missing values
         holCheckNA <- which(is.na(visit.working.df[,i]))
         holCheckNA <- holCheckNA[holCheckNA > min(rowStart)]

         #step6: If the missing value is due to a holiday, add it as 0
         for(l in 1:length(holCheckNA)){
               #l<-2
           if(visit.working.df[holCheckNA[l],3] == 1){
             visit.working.df[holCheckNA[l],i] <- 0
           }

         }
   }
   #Step6 - Chec for final NA for imputation
   st <- min(rowStart)
   en <- nrow(visit.working.df)

    fcast.median.est <-  round(
                           forecast(ets(
                              tsclean(
                                imputeTS::na.mean(
                                  as.numeric(
                                    unlist(visit.working.df[st:en,i])),
                                  option = "median"),
                                replace.missing = F)),39)$mean,0)

    fcast.median.aa <- round(
                         forecast(
                           auto.arima(
                             tsclean(
                               imputeTS::na.mean(
                                 as.numeric(
                                   unlist(visit.working.df[st:en,i])),
```

```
                                  option = "median"),
                                replace.missing = F)),39)$mean,0)
    if(i == 7){
      #Median
       visit.median.fcast <- fcast.median.est
       visit.median.fcast.aa <- fcast.median.aa
    }
    if(i > 7){
      #Median
      visit.median.fcast <- cbind.data.frame(visit.median.fcast,fcast.median.est)
      visit.median.fcast.aa <- cbind.data.frame(visit.median.fcast.aa,fcast.median.aa)
    }
}
colnames(visit.median.fcast) <- colnames(visit.working.df[,7:827])
colnames(visit.median.fcast.aa) <- colnames(visit.working.df[,7:827])

####-----------------
## ETS Model
####-----------------

#Median Imputation - EST Model
data.result.median <-
    rownames_to_column(as.tibble(visit.median.fcast),var = "RowID")
final_submission.median <- as.tibble(data.result.median %>%
                                       mutate(Date = seq(from = as.Date("2017-04-23"),
                                                         to = as.Date("2017-05-31"),
                                                         by= 'day')) %>%
                                       gather("StoreID","visitors",
                                              2:ncol(data.result.median)) %>%
                                       unite("id",StoreID,Date, sep = "_")
                                )[,2:3]
write_csv(final_submission.median,
          "Approach_2_Final_submission_median_ETS_formatted.csv")

####-----------------
## Auto Arima Model
####-----------------
#Median Imputation -  Auto ARIMA Model
data.result.median.aa <-
  rownames_to_column(as.tibble(visit.median.fcast.aa),var = "RowID")
final_submission.median.aa <- as.tibble(data.result.median.aa %>%
                                          mutate(Date = seq(from = as.Date("2017-04-23"),
                                                            to = as.Date("2017-05-31"),
                                                            by= 'day')) %>%
                                          gather("StoreID","visitors",
                                                 2:ncol(data.result.median.aa)) %>%
                                          unite("id",StoreID,Date, sep = "_")
)[,2:3]
write_csv(final_submission.median.aa,
          "Approach_2_Final_submission_median_ARIMA_formatted.csv")

##------------------
#Modeling Approach 3
```

```
##-------------------
#Import air_visit data
visit.data.df <-
  read_csv("Restaurant_Visitor_Forecasting_air_visit_data.csv")
#Loading holiday data
date.info.df <-
  read_csv("Restaurant_Visitor_Forecasting_date_info.csv")

#format the data from rows to columns
visit.df <-   visit.data.df %>%
              spread(air_store_id, visitors) %>%
              separate(visit_date, c("Year","Month","Day"))

  visit.df.rm <-  visit.df[,!(
                              names(visit.df) %in% c(
                              "air_0ead98dd07e7a82a",
                              "air_229d7e508d9f1b5e",
                              "air_2703dcb33192b181",
                              "air_b2d8bc9c88b85f96",
                              "air_cb083b4789a8d3a2",
                              "air_cf22e368c1a71d53",
                              "air_d0a7bd3339c3d12a",
                              "air_d63cfa6d6ab78446"
                              )
                              )]
  #combining holiday data
  visit2.df <- as.tibble(cbind.data.frame(
                          date.info.df[1:478,1:3],
                          visit.df.rm[1:478,]))

  #Creating a working dataset
  visit.working.df <- visit2.df

  #Imputation to remove missing values
 for(i in 7:ncol(visit.working.df)){
    #step1 - get all the missing values for each column
    rowStart <- which(!is.na(visit.working.df[,i]))

    #step3 - Get all the rows with a value.
    actNa <- which(is.na(visit.working.df[,i]))
    actNa <- actNa[actNa > min(rowStart)]

    if(length(actNa) != 0){
          #step4: Replace all the values for NA if store has NA occuring in 7th day
                 # this could mean it closes 1 day a week.
          for(k in 1:length(actNa)){
            if(k+1 <= length(actNa)) {
              if(abs(actNa[k] - actNa[k+1]) == 7){
                visit.working.df[actNa[k],i] <- 0
              }
            }
          }
```

```
        #Step5 - Get all the left over missing values
        holCheckNA <- which(is.na(visit.working.df[,i]))
        holCheckNA <- holCheckNA[holCheckNA > min(rowStart)]

        #step6: If the missing value is due to a holiday, add it as 0
        for(l in 1:length(holCheckNA)){
              #l<-2
          if(visit.working.df[holCheckNA[l],3] == 1){
            visit.working.df[holCheckNA[l],i] <- 0
          }

        }
}
#Step6 - Chec for final NA for imputation
st <- min(rowStart)
en <- nrow(visit.working.df)

## Find the best model
visit_per_rest <- round(tsclean(imputeTS::na.mean(as.numeric(
                      unlist(visit.working.df[st:en,i])),
                      option = "median"),replace.missing = F),0)

visit.ts <- as.ts(visit_per_rest, start(2016,1,1), frequency=365)
train_split <- round(length(visit.ts) * 0.7)

train.ts <- window(visit.ts, end = train_split)
test.ts <- window(visit.ts, start = train_split+1)

#ETS Model
fit.ETS <-  ets(train.ts)
acc.ets <- accuracy(fit.ETS, data=test.ts)

#Auto Arima
fit.arima <- auto.arima(train.ts)
acc.arima <- accuracy(fit.arima, data=test.ts)

if(i == 7){
  if((is.nan(acc.ets[6])== TRUE) & (is.nan(acc.arima[6]) == TRUE)){
    visit.median.fcast <- round(forecast(auto.arima(visit.ts), 39)$mean,0)
  }else if ((is.nan(acc.ets[6])== TRUE) & (is.nan(acc.arima[6]) == FALSE)){
    visit.median.fcast <- round(forecast(ets(visit.ts), 39)$mean,0)
  }else if ((is.nan(acc.ets[6])== FALSE) & (is.nan(acc.arima[6]) == TRUE)){
    visit.median.fcast <- round(forecast(auto.arima(visit.ts), 39)$mean,0)
  }else if(acc.ets[6] < acc.arima[6]){
    visit.median.fcast <- round(forecast(ets(visit.ts), 39)$mean,0)
  }else{
    visit.median.fcast <- round(forecast(auto.arima(visit.ts), 39)$mean,0)
  }
}
if(i > 7){
  #Median
  if((is.nan(acc.ets[6])== TRUE) & (is.nan(acc.arima[6]) == TRUE)){
    visit.median.fcast <-
```

```
                cbind.data.frame(visit.median.fcast,
                                 round(forecast(auto.arima(visit.ts), 39)$mean,0))
      }else if ((is.nan(acc.ets[6])== TRUE) & (is.nan(acc.arima[6]) == FALSE)){
        visit.median.fcast <-
                cbind.data.frame(visit.median.fcast,
                                 round(forecast(ets(visit.ts), 39)$mean,0))
      }else if ((is.nan(acc.ets[6])== FALSE) & (is.nan(acc.arima[6]) == TRUE)){
        visit.median.fcast <-
                cbind.data.frame(visit.median.fcast,
                                 round(forecast(auto.arima(visit.ts), 39)$mean,0))
      }else if(acc.ets[6] < acc.arima[6]){
        visit.median.fcast <-
                cbind.data.frame(visit.median.fcast,
                                 round(forecast(ets(visit.ts), 39)$mean,0))
      }else{
        visit.median.fcast <-
                cbind.data.frame(visit.median.fcast,
                                 round(forecast(auto.arima(visit.ts), 39)$mean,0))
      }
    }
}
colnames(visit.median.fcast) <- colnames(visit.working.df[,7:827])

#Median Imputation - FINAL Export
data.result.median <-
        rownames_to_column(as.tibble(visit.median.fcast),var = "RowID")
final_submission.median <- as.tibble(data.result.median %>%
                                       mutate(Date = seq(from = as.Date("2017-04-23"),
                                                         to = as.Date("2017-05-31"),
                                                         by= 'day')) %>%
                                       gather("StoreID","visitors",
                                              2:ncol(data.result.median)) %>%
                                       unite("id",StoreID,Date, sep = "_")
                                     )[,2:3]
write_csv(final_submission.median,
          "Approach_3_Final_submission_Median_AutoARIMA_ETS.csv")
```