Mini Project On

Restaurant Visitor Forecasting

By

Shardul Rane[2018450041]

Under the guidance of

Internal Supervisor

Dr. Pooja Raundale

Department of Master Of Computer Application
Sardar Patel Institute of Technology
Autonomous Institute Affiliated to Mumbai University
2018-19

CERTIFICATE OF APPROVAL

This is to certify that the following students

Shardul Rane[2018450041]

Have satisfactorily carried out work on the project entitled

"Restaurant Visitor Forecasting"

Towards the fulfilment of project,
as laid down by Sardar Patel Institute of Technology
during year 2018-19.

Project Guide: Dr. Pooja Raundale

PROJECT APPROVAL CERTIFICATE

This is to certify that the following students

Shardul Rane[2018450041]

Have successfully completed the Project report on
“Restaurant Visitor Forecasting”,
which is found to be satisfactory and is approved

At

SARDAR PATEL INSTITUTE OF TECHNOLOGY,
ANDHERI (W), MUMBAI.

INTERNAL EXAMINER

EXTERNAL EXAMINER

HEAD OF DEPARTMENT

PRINCIPAL

Contents

List of Figures

1 Introduction

In service industry, an increasing number of business owners improve the quality of service and reduce cost using machine learning techniques. Based on historical data of business, predictive models of machine learning can estimate future moves of customers [1].Especially, forecasting the number of future visitors can help restaurant owners [2] make the best operations to maximise the revenue. With an accurate visitor forecasting model, restaurant owners can prepare suitable amount of ingredients that exactly satisfy future visitors. In addition, restaurant owners can also schedule a suitable number of staff that can serve these visitors.

Despite various visitor forecasting techniques for other purposes such as national tourism and hotel demand (e.g., [3], [4][5] [6] [7]) in the literature, little is known for restaurant owners to estimate the number of future visitors using big data. Previous work such as [6] focused only on visitors' revisit. Note that new customers may outnumber old customers in some restaurants of tourist destinations. Hence, it is necessary to develop anew method that can predict the total number of future visitors to a restaurant on a specic day.

2 Study of Algorithms

- Supervised learning algorithms
  - Linear Regression
    In ML, we have a set of input variables (x) that are used to determine the output variable (y). A relationship exists between the input variables and the output variable. The goal of ML is to quantify this relationship. In Linear Regression, the relationship between the input variables (x) and output variable (y) is expressed as an equation of the form y = a + bx. Thus, the goal of linear regression is to find out the values of coefficients a and b. Here, a is the intercept and b is the slope of the line.
  - Logistic Regression
    Linear regression predictions are continuous values (rainfall in cm),logistic regression predictions are discrete values (whether a student passed/-failed) after applying a transformation function.

    Logistic regression is best suited for binary classification (datasets where y = 0 or 1, where 1 denotes the default class. Example: In predicting whether an event will occur or not, the event that it occurs is classified as 1. In predicting whether a person will be sick or not, the sick instances are denoted as 1). It is named after the transformation function used in it, called the logistic function h(x)= 1/ (1 + ex), which is an S-shaped curve.
  - CART
    Classification and Regression Trees (CART) is an implementation of Decision Trees, among others such as ID3, C4.5.

    The non-terminal nodes are the root node and the internal node. The terminal nodes are the leaf nodes. Each non-terminal node represents a single input variable (x) and a splitting point on that variable; the leaf nodes represent the output variable (y). The model is used as follows to make predictions: walk the splits of the tree to arrive at a leaf node and output the value present at the leaf node.
  - Naïve Bayes
    To calculate the probability that an event will occur, given that another event has already occurred, we use Bayes' Theorem. To calculate the probability of an outcome given the value of some variable, that is, to calculate the probability of a hypothesis(h) being true, given our prior knowledge(d), we use Bayes' Theorem as follows:

$$P(h \mid d) = \frac{P(d|h)\,P(h))}{P(d)}$$

where:

- $P(h \mid d)$ = Posterior probability. The probability of hypothesis h being true, given the data d, where $P(h \mid d) = P(d1 \mid h)P(d2 \mid h)\ldots P(dn \mid h)P(d)$
- $P(d \mid h)$ = Likelihood. The probability of data d given that the hypothesis h was true.
- P(h) = Class prior probability. The probability of hypothesis h being true (irrespective of the data)
- P(d) = Predictor prior probability. Probability of the data (irrespective of the hypothesis)

This algorithm is called 'naive' because it assumes that all the variables are independent of each other, which is a naive assumption to make in real-world examples.

- KNN

  The k-nearest neighbours algorithm uses the entire dataset as the training set, rather than splitting the dataset into a trainingset and testset.

  When an outcome is required for a new data instance, the KNN algorithm goes through the entire dataset to find the k-nearest instances to the new instance, or the k number of instances most similar to the new record, and then outputs the mean of the outcomes (for a regression problem) or the mode (most frequent class) for a classification problem. The value of k is user-specified.

- Unsupervised learning algorithms
  - Apriori

    The Apriori algorithm is used in a transactional database to mine frequent itemsets and then generate association rules. It is popularly used in market basket analysis, where one checks for combinations of products that frequently co-occur in the database. In general, we write the association rule for 'if a person purchases item X, then he purchases item Y' as : X -¿ Y.
  - K-means

    K-means is an iterative algorithm that groups similar data into clusters.It calculates the centroids of k clusters and assigns a data point to that cluster having least distance between its centroid and the data point.
  - PCA

    Principal Component Analysis (PCA) is used to make data easy to explore and visualize by reducing the number of variables.

- Ensemble learning techniques:
  - ARIMA
    ARIMA stands for Auto Regressive Integrated Moving Average. There are seasonal and Non-seasonal ARIMA models that can be used for forecasting. This method has three variables to account for P = Periods to lag for eg: (if P= 3 then we will use the three previous periods of our time series in the autoregressive portion of the calculation) P helps adjust the line that is being fitted to forecast the series Purely autoregressive models resemble a linear regression where the predictive variables are P number of previous periods D = In an ARIMA model we transform a time series into stationary one(series without trend or seasonality) using differencing. D refers to the number of differencing transformations required by the time series to get stationary. Stationary time series is when the mean and variance are constant over time. It is easier to predict when the series is stationary. The first differencing value is the difference between the current time period and the previous time period. If these values fail to revolve around a constant mean and variance then we find the second differencing using the values of the first differencing. We repeat this until we get a stationary series The best way to determine whether or not the series is sufficiently differenced is to plot the differenced series and check to see if there is a constant mean and variance. Q = This variable denotes the lag of the error component, where error component is a part of the time series not explained by trend or seasonality

    Autocorrelation function plot (ACF): Autocorrelation refers to how correlated a time series is with its past values whereas the ACF is the plot used to see the correlation between the points, up to and including the lag unit. In ACF, the correlation coefficient is in the x-axis whereas the number of lags is shown in the y-axis. Normally in an ARIMA model, we make use of either the AR term or the MA term. We use both of these terms only on rare occasions. We use the ACF plot to decide which one of these terms we would use for our time series If there is a Positive autocorrelation at lag 1 then we use the AR model If there is a Negative autocorrelation at lag 1 then we use the MA model After plotting the ACF plot we move to Partial Autocorrelation Function plots (PACF). A partial autocorrelation is a summary of the relationship between an observation in a time series with observations at prior time steps with the relationships of intervening observations removed. If the PACF plot drops off at lag n, then use an AR(n) model and if the drop in PACF is more gradual then we use the MA term Autoregressive component: A purely AR model forecasts only using a combination of the past values sorta like linear regression where the number of AR terms used is directly proportional to the number of previous periods taken into consideration for the forecasting. Use AR terms in the model when the ACF plots show autocorrelation decaying towards zero PACF plot cuts off quickly towards zero ACF of a stationary series shows positive at lag-1 Moving Averages: Random jumps in the time series plot whose effect is felt in two or more consecutive periods. These jumps represent the error calculated in our ARIMA model and represent what the MA component would lag for. A purely MA model would smooth out these sudden jumps like the exponential smoothing method. Use MA terms in the model when the model is Negatively Autocorrelated at Lag — 1 ACF that drops sharply after a few lags PACF decreases more gradually Integrated component: This component comes into action when the time series is not stationary. The number of times we have to difference the series to make it stationary is the parameter(i-term) for the integrated component We can represent our model as ARIMA(ar-term, ma-term, i-term) Finding the correct model is an iterative process. Seasonal ARIMA (SARIMA) models: As the name suggests, this model is used when the time series exhibits seasonality. This model is similar to ARIMA models, we just have to add in a few parameters to account for the seasons We write SARIMA as ARIMA(p,d,q)(P, D, Q)m, p — the number of autoregressive d — degree of differencing q — the number of moving average terms m — refers to the number of periods in each season (P, D, Q)— represents the (p,d,q) for the seasonal part of the time series Seasonal differencing takes into account the seasons and differences the current value and it's value in the previous season eg: Difference for the month may would be value in May 2018—value in may 2017. In Purely seasonal AR model, ACF decays slowly while PACF cuts off to zero AR models are used when seasonal auto-correlation is positive In Purely seasonal MA model, ACF cuts off to zero and vice versa MA models are used when seasonal auto-correlation is negative

3 Literature Survey

There are many restaurants in the entire country which provides table booking via and online app in this many of the restaurants often get entire data.But denying the fact that there are many things which can be done with the efficient use of the data. Forecasting the number of future visitors is a meaningful taskin service industry. Researchers have proposed various tech-niques to predicting the volume of future visitors using big datain different areas.For instance, search engine data can be used to forecast thenumber of future tourists to a certain popular destination in China [3]. A strong correlation between the search queries and visitor numbers was found. Similarly, web traffic data in an area can be used to predict local hotel demand [4]. With the popularity of devices connected to the Internet, this prediction relies on the fact that more travellers generate more traffic data.However, obtaining web search data and web traffic data is not a trivial task. Especially, many tourist destinations do not have the network connection, e.g., high mountains and small islands. Hence, many techniques predict the number of future visitors using historical visitor numbers only, e.g., predicting visitors to a city using past tourist traffic [5]. Also, the characteristics of a location can be used for the prediction, e.g.,predicting customers' revisit to a restaurant using the attributes of the restaurant [6]. Moreover, the geographical influence is important factor in predicting future visitors going to Points-of-Interests (POIs) [7].

Data mining and machine learning depend on classification which is the most essential and important task. Many experiments are performed on various datasets using multiple forecasting and feature selection techniques. A good amount of research on restaurant visitors forecasting is found in literature. Many of them show good forecasting accuracy.[2]

4 Objectives

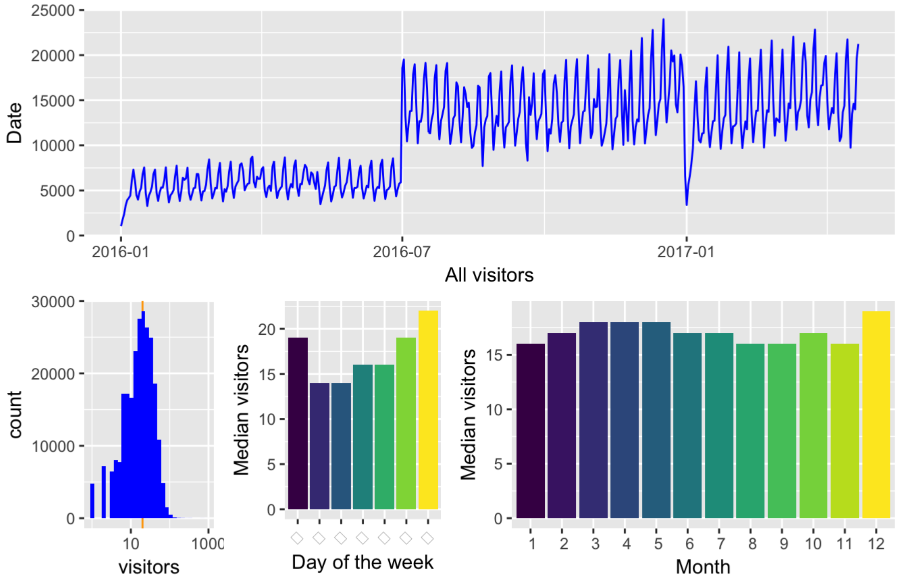
The objectives of this project are:

- Designing a novel approach to predict how many future visitors will go to a restaurant using Machine Learning and super-vised learning.This approach relies on low-computation algorithms so that restaurants can easily deploy them on common PCs.
- We show the effectiveness of our approach by an evaluation using large-scale real-world datasets from two restaurant booking websites(AIR and HPG).

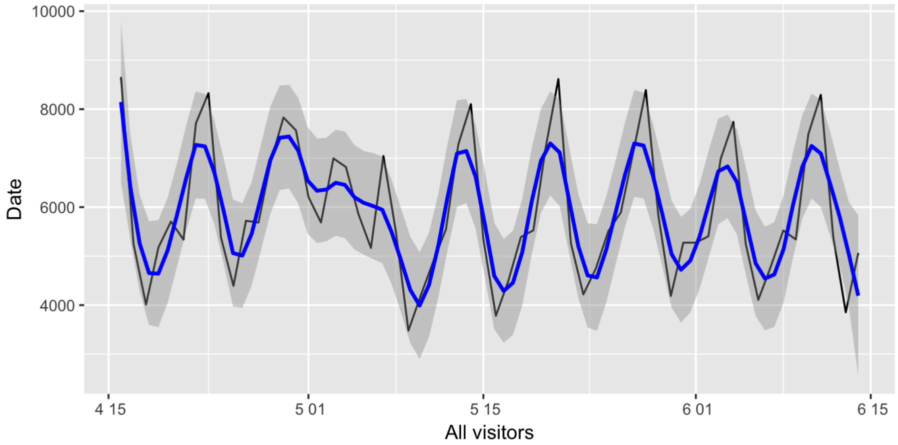
5 Methodology

5.1 Experiment and Results

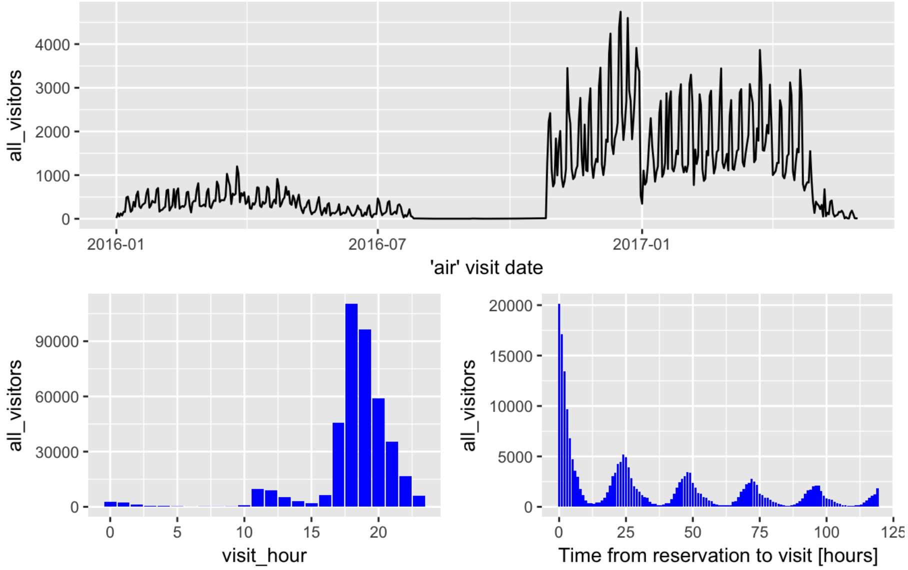
We have proposed a model using ARIMA (Auto Regressive Integrated Moving Average) which is implemented in a high configuration computer. The computer configuration was Intel Core i5 with 8GB RAM. We have used Scikit-learn which is an open-source software developed in Python for machine learning library. An Integrated development environment named as Spyder is used to run the program.



5.1.1: AIR Visits

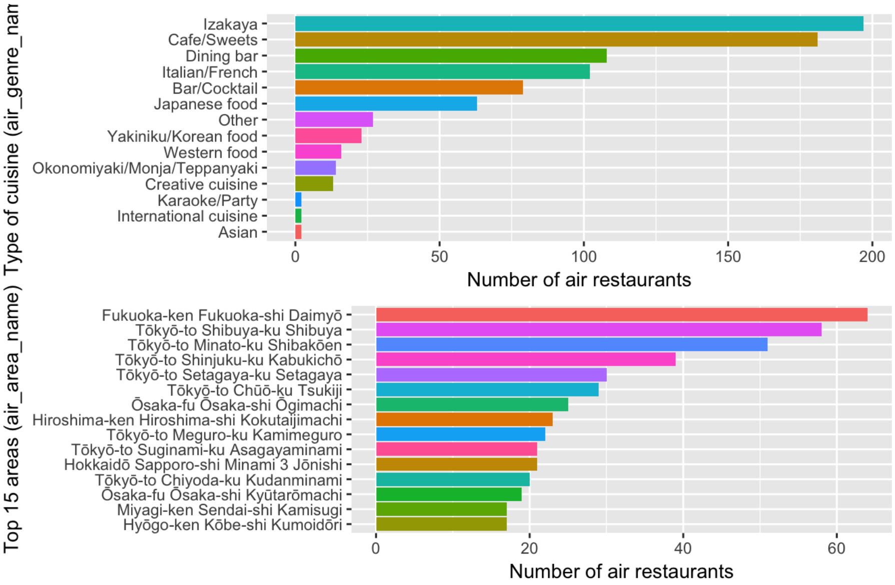


5.1.2: Forecasting for the last week of April plus May 2017

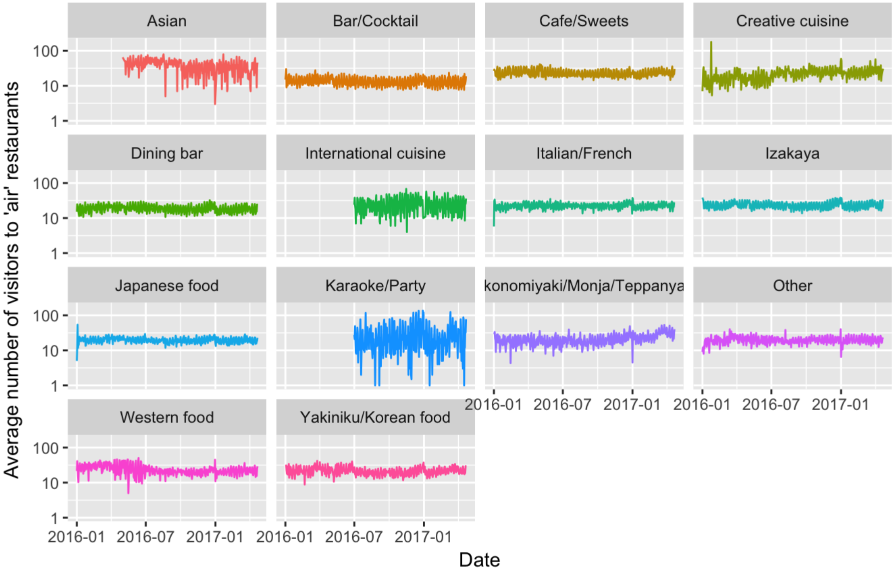


5.1.3: Reservations data VS The actual visitor number

.

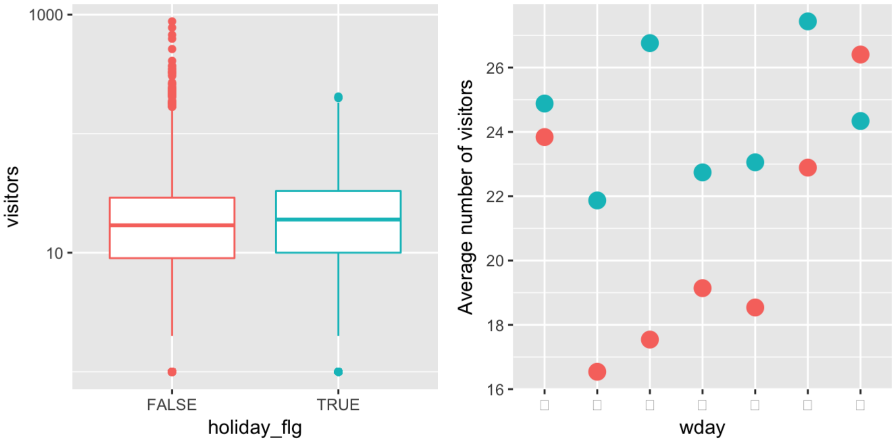


5.1.4: .Plot for number of different types of cuisines

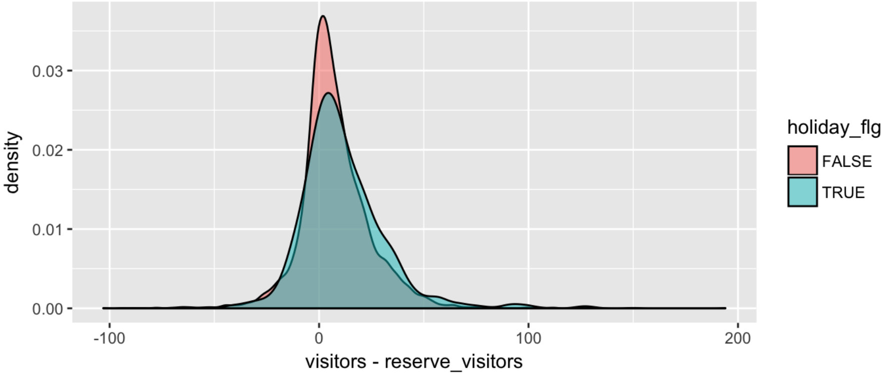


5.1.5: . Plot for number of different types of genre

.



5.1.6: . Influence of holidays VS visitor numbers



5.1.7: . Visitors VS Reserve Visitors

.

6 Exploratory Data Analysis

6.1 Code

```
# Check R Version
# general visualisation
library('ggplot2') # visualisation
library('scales') # visualisation
library('grid') # visualisation
library('gridExtra') # visualisation
library('RColorBrewer') # visualisation
library('corrplot') # visualisation

# general data manipulation
library('dplyr') # data manipulation
library('readr') # input/output
library('data.table') # data manipulation
library('tibble') # data wrangling
library('tidyr') # data wrangling
library('stringr') # string manipulation
library('forcats') # factor manipulation

# specific visualisation
library('ggrepel') # visualisation
library('ggridges') # visualisation
library('ggExtra') # visualisation
library('ggforce') # visualisation
library('viridis') # visualisation

# specific data manipulation
library('lazyeval') # data wrangling
library('broom') # data wrangling
library('purrr') # string manipulation

# Date plus forecast
library('lubridate') # date and time
library('timeDate') # date and time
library('tseries') # time series analysis
library('forecast') # time series analysis
library('prophet') # time series analysis
library('timetk') # time series analysis

# Maps / geospatial
library('geosphere') # geospatial locations
library('leaflet') # maps
library('leaflet.extras') # maps
library('maps') # maps


multiplot <- function(..., plotlist=NULL, file, cols=1,
    layout=NULL) {

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine
      layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of
        cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)
        ),
                     ncol = cols, nrow = ceiling(numPlots
                         /cols))
  }

  if (numPlots==1) {
    print(plots[[1]])

  } else {
    # Set up the page
    grid.newpage()
    pushViewport(viewport(layout = grid.layout(nrow(
        layout), ncol(layout))))

    # Make each plot, in the correct location
    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that
          contain this subplot
      matchidx <- as.data.frame(which(layout == i, arr.
          ind = TRUE))

      print(plots[[i]], vp = viewport(layout.pos.row =
          matchidx$row,
                                      layout.pos.col =
                                          matchidx$col))
    }
  }
}
rpath=”C:/Users/DELL/Desktop/restaurant-visitors-dataset/
    ”

air_visits <- as.tibble(fread(str_c(rpath,'air_visit_data
    .csv')))
air_reserve <- as.tibble(fread(str_c(rpath,'air_reserve.
    csv')))
hpg_reserve <- as.tibble(fread(str_c(rpath,'hpg_reserve.
    csv')))
air_store <- as.tibble(fread(str_c(rpath,'air_store_info.
    csv')))
hpg_store <- as.tibble(fread(str_c(rpath,'hpg_store_info.
    csv')))
holidays <- as.tibble(fread(str_c(rpath,'date_info.csv'))
    )
store_ids <- as.tibble(fread(str_c(rpath,'store_id_
    relation.csv')))
test <- as.tibble(fread(str_c(rpath,'sample_submission.
    csv')))

sum(is.na(air_visits))



air_reserve <- air_reserve %>%
  mutate(visit_datetime = ymd_hms(visit_datetime),
         reserve_datetime = ymd_hms(reserve_datetime))

hpg_reserve <- hpg_reserve %>%
  mutate(visit_datetime = ymd_hms(visit_datetime),
         reserve_datetime = ymd_hms(reserve_datetime))

air_store <- air_store %>%
  mutate(air_genre_name = as.factor(air_genre_name),
         air_area_name = as.factor(air_area_name))

hpg_store <- hpg_store %>%
  mutate(hpg_genre_name = as.factor(hpg_genre_name),
         hpg_area_name = as.factor(hpg_area_name))

holidays <- holidays %>%
  mutate(holiday_flg = as.logical(holiday_flg),date = ymd
      (calendar_date))

#Graph 1 ke liye ggplot and geom_line ka use karenge
```

```
p1 <- air_visits %>%
  group_by(visit_date) %>%
  summarise(all_visitors = sum(visitors)) %>%
  ggplot(aes(visit_date,all_visitors)) +
  geom_line(col = "blue",group=1) +
  labs(y = "All␣visitors", x = "Date")

p2 <- air_visits %>%
  ggplot(aes(visitors)) +
  geom_vline(xintercept = 20, color = "orange") +
  geom_histogram(fill = "blue", bins = 30) +
  scale_x_log10()

p3 <- air_visits %>%
  mutate(wday = wday(visit_date, label = TRUE)) %>%
  group_by(wday) %>%
  summarise(visits = median(visitors)) %>%
  ggplot(aes(wday, visits , fill = wday)) +
  geom_col() +
  theme(legend.position = "none", axis.text.x  = element_
     text(angle=45, hjust=1, vjust=0.9)) +
  labs(x = "Day␣of␣the␣week", y = "Median␣visitors")

p4 <- air_visits %>%
  mutate(month = month(visit_date, label = TRUE)) %>%
  group_by(month) %>%
  summarise(visits = median(visitors)) %>%
  ggplot(aes(month, visits , fill = month)) +
  geom_col() +
  theme(legend.position = "none") +
  labs(x = "Month", y = "Median␣visitors")

layout <- matrix(c(1,1,1,1,2,3,4,4),2,4,byrow=TRUE)
multiplot(p1, p2, p3, p4, layout=layout)
```

```
air_visits %>%
  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(x = "All␣visitors", y = "Date")

foo <- air_reserve %>%
  mutate(reserve_date = date(reserve_datetime),
         reserve_hour = hour(reserve_datetime),
         reserve_wday = wday(reserve_datetime, label =
             TRUE),
         visit_date = date(visit_datetime),
         visit_hour = hour(visit_datetime),
         visit_wday = wday(visit_datetime, label = TRUE),
         diff_hour = time_length(visit_datetime - reserve
             _datetime, unit = "hour"),
         diff_day = time_length(visit_datetime - reserve_
             datetime, unit = "day")
  )

p1 <- foo %>%
  group_by(visit_date) %>%
  summarise(all_visitors = sum(reserve_visitors)) %>%
  ggplot(aes(visit_date, all_visitors)) +
  geom_line() +
  labs(x = "'air'␣visit␣date")

p2 <- foo %>%
  group_by(visit_hour) %>%
  summarise(all_visitors = sum(reserve_visitors)) %>%
  ggplot(aes(visit_hour, all_visitors)) +
  geom_col(fill = "blue")

p3 <- foo %>%
  filter(diff_hour < 24*5) %>%
  group_by(diff_hour) %>%
  summarise(all_visitors = sum(reserve_visitors)) %>%
  ggplot(aes(diff_hour, all_visitors)) +
  geom_col(fill = "blue") +
  labs(x = "Time␣from␣reservation␣to␣visit␣[hours]")

layout <- matrix(c(1,1,2,3),2,2,byrow=TRUE)
multiplot(p1, p2, p3, layout=layout)


leaflet(air_store) %>%
  addTiles() %>%
  addProviderTiles("CartoDB.Positron") %>%
  addMarkers(~longitude, ~latitude,
             popup = ~air_store_id, label = ~air_genre_
                 name,
             clusterOptions = markerClusterOptions())

p1 <- air_store %>%
  group_by(air_genre_name) %>%
  count() %>%
  ggplot(aes(reorder(air_genre_name, n, FUN = min), n,
      fill = air_genre_name)) +
  geom_col() +
  coord_flip() +
  theme(legend.position = "none") +
  labs(x = "Type␣of␣cuisine␣(air_genre_name)", y = "
      Number␣of␣air␣restaurants")

p2 <- air_store %>%
  group_by(air_area_name) %>%
  count() %>%
  ungroup() %>%
  top_n(15,n) %>%
  ggplot(aes(reorder(air_area_name, n, FUN = min) ,n,
      fill = air_area_name)) +
  geom_col() +
  theme(legend.position = "none") +
  coord_flip() +
  labs(x = "Top␣15␣areas␣(air_area_name)", y = "Number␣of
      ␣air␣restaurants")

layout <- matrix(c(1,2),2,1,byrow=TRUE)
multiplot(p1, p2, layout=layout)



foo <- air_visits %>%
  left_join(air_store, by = "air_store_id")

foo %>%
  group_by(visit_date, air_genre_name) %>%
  summarise(mean_visitors = mean(visitors)) %>%
  ungroup() %>%
  ggplot(aes(visit_date, mean_visitors, color = air_genre
      _name),group=1) +
  geom_line(group=1) +
  labs(y = "Average␣number␣of␣visitors␣to␣'air'␣
      restaurants", x = "Date") +
  theme(legend.position = "none") +
  scale_y_log10() +
  facet_wrap(~ air_genre_name)



foo <- air_visits %>%
  mutate(calendar_date = as.character(visit_date)) %>%
  left_join(holidays, by = "calendar_date")

p1 <- foo %>%
  ggplot(aes(holiday_flg, visitors, color = holiday_flg))
      +
  geom_boxplot() +
  scale_y_log10() +
  theme(legend.position = "none")

p2 <- foo %>%
  mutate(wday = wday(date, label = TRUE)) %>%
  group_by(wday, holiday_flg) %>%
  summarise(mean_visitors = mean(visitors)) %>%
  ggplot(aes(wday, mean_visitors, color = holiday_flg)) +
  geom_point(size = 4) +
  theme(legend.position = "none") +
  labs(y = "Average␣number␣of␣visitors")

layout <- matrix(c(1,2),1,2,byrow=TRUE)
multiplot(p1, p2, layout=layout)


foo <- air_visits %>%
  mutate(calendar_date = as.character(visit_date)) %>%
  left_join(holidays, by = "calendar_date")

p1 <- foo %>%
  ggplot(aes(holiday_flg, visitors, color = holiday_flg))
      +
  geom_boxplot() +
  scale_y_log10() +
  theme(legend.position = "none")

p2 <- foo %>%
  mutate(wday = wday(date, label = TRUE)) %>%
  group_by(wday, holiday_flg) %>%
  summarise(mean_visitors = mean(visitors)) %>%
  ggplot(aes(wday, mean_visitors, color = holiday_flg)) +
  geom_point(size = 4) +
  theme(legend.position = "none") +
  labs(y = "Average␣number␣of␣visitors")

layout <- matrix(c(1,2),1,2,byrow=TRUE)
multiplot(p1, p2, layout=layout)

foo <- air_reserve %>%
  mutate(visit_date = date(visit_datetime)) %>%
  group_by(air_store_id,visit_date) %>%
  summarise(reserve_visitors_air = sum(reserve_visitors))

bar <- hpg_reserve %>%
  mutate(visit_date = date(visit_datetime)) %>%
  group_by(hpg_store_id,visit_date) %>%
  summarise(reserve_visitors_hpg = sum(reserve_visitors))
      %>%
  inner_join(store_ids, by = "hpg_store_id")
all_reserve <- air_visits %>%
  inner_join(foo, by = c("air_store_id", "visit_date"))
     %>%
  inner_join(bar, by = c("air_store_id", "visit_date"))
     %>%
  mutate(reserve_visitors = reserve_visitors_air +
      reserve_visitors_hpg)

all_reserve %>%
  mutate(date = visit_date) %>%
  left_join(holidays, by = "date") %>%
  ggplot(aes(visitors - reserve_visitors, fill = holiday_
      flg)) +
  geom_density(alpha = 0.5)
```

7 Implementation

7.1 Code

```
# Check Python Version
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 24 14:52:17 2019

@author: SHARDUL
"""

import numpy as np
from flask import Flask, request, jsonify, render_
    template,session, redirect
import pickle
import matplotlib.pyplot as plt
import pandas as pd
from datetime import datetime as dt
import seaborn as sns; sns.set()
import statsmodels.api as sm
import warnings
import itertools

app = Flask(__name__)
model = pickle.load(open('Model/trained.pickle', 'rb'))

@app.route('/')
def home():
    return render_template('index3.html')

@app.route('/eda', methods=['GET', 'POST'])
def eda():
    return render_template('eda.html')
@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    int_features = [x for x in request.form.values()]
    #int_features[0]
    #final_features = [np.array(int_features)]
    #prediction = model.predict(final_features)

    plt.style.use('fivethirtyeight')
    plt.rcParams['axes.labelsize'] = 14
    plt.rcParams['xtick.labelsize'] = 12
    plt.rcParams['ytick.labelsize'] = 12
    plt.rcParams['text.color'] = 'k'
    df=pd.read_csv('air_visit_data.csv',sep=”;”)

    ss=str(int_features[0])
    ss=str('air_ba937bf13d40fb24')
    own=df.loc[df['air_store_id'] == ss]
    df=own
    df['visit_date'] = pd.to_datetime(df['visit_date'])

#     dfinal.dtypes
    df['visit_date'].min(), df['visit_date'].max()

#     df.isnull().sum()

    df = df.groupby('visit_date')['visitors'].sum().reset
        _index()

    df = df.set_index('visit_date')
#     df.index
    y = df['visitors'].resample('MS').mean()
    y.fillna(0, inplace=True)
#     y['2016':]

    p = d = q = range(0, 2)
    pdq = list(itertools.product(p, d, q))
    seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(
        itertools.product(p, d, q))]
    pak_param=0
    seaonal=0
    maic=9999999

    for param in pdq:
        for param_seasonal in seasonal_pdq:
            try:
                mod = sm.tsa.statespace.SARIMAX(y,
                                                order=
                                                    param,
                                                seasonal_
                                                    order=
                                                    param_
                                                    seasonal
                                                    ,
                                                enforce_
                                                    stationarity
                                                    =False
                                                    ,
                                                  enforce_
                                                     invertibility
                                                    =False
                                                    )
                results = mod.fit()
                if results.aic<maic:
                    maic=results.aic
                    pak_param=param
                    seaonal=param_seasonal

    #             print('ARIMA{}x{}12 - AIC:{}'.format(
        param, param_seasonal, results.aic))
            except:
                continue

    mod = sm.tsa.statespace.SARIMAX(y,
                                     order=pak_param,
                                     seasonal_order=
                                        seaonal,
                                     enforce_stationarity=
                                        False,
                                     enforce_invertibility
                                        =False)
    results = mod.fit()
    pred = results.get_prediction(start=pd.to_datetime('
       2019-01-01'), dynamic=False)
    pred_ci = pred.conf_int()
    #own.boxplot(own.reserve_visitors)
    y_forecasted = pred.predicted_mean
    y_truth = y['2016-01-01':]
    mse = ((y_forecasted - y_truth) ** 2).mean()
#     print('The Mean Squared Error of our forecasts is
    {}'.format(round(mse, 2)))
#     print('The Root Mean Squared Error of our forecasts
    is {}'.format(round(np.sqrt(mse), 2)))
    pred_uc = results.get_forecast(steps=30)
    pred_ci = pred_uc.conf_int()
#     ax = y.plot(label='Observed Past Records', figsize
    =(10, 5))
#     pred_uc.predicted_mean.plot(ax=ax, label='Expected
    Number Of Visitors')
#     ax.fill_between(pred_ci.index,
#                     pred_ci.iloc[:, 0],
#                           pred_ci.iloc[:, 1], color='k', alpha
    =.25)
#       ax.set_xlabel('Date')
#       ax.set_ylabel('Number Of Visitors In A Month')

#       print(pred_uc)
#       print(pred_uc.conf_int())
#       print("Predicted Vaues")
#       print(pred_uc.predicted_mean)
#       output = round(prediction[0], 2)
#       type(pred_uc.predicted_mean)
    final=pred_uc.predicted_mean.to_frame()
    final.reset_index(inplace=True)

    final=final.rename(columns={int(0): "Visitors_Count"
        })
    final["Visitors_Count"]=final["Visitors_Count"].abs()
#       final.round("a")
    final["Visitors_Count"]=final["Visitors_Count"].round
        ()
    #for i in range(0,len(pred_uc.predicted_mean)):
    #       pred_uc.predicted_mean[i]=int(round(pred_uc.
        predicted_mean[i]))
    output=str(final['Visitors_Count'])
    return render_template('index3.html',tables=[final.to
        _html(classes='data')],titles=final.columns.values
        )
#       return render_template('index3.html', prediction_
    text=output)
if __name__ == "__main__":
    app.run(debug=True)
```

7.2 Strategy

We have utilized the Use Only Useful i.e. The data which is of actual use should only be consider while building the model .This technique is utilized to improve the accuracy of the model.As the analysis showed that the data from a particular date-range was missing we skipped the outlier while the creating the model.

7.3 Algorithms

7.3.1 Auto Regressive Integrated Moving Average

It is a series which needs to be differentiated in order to be made stationary is an “integrated” (I) series. Lags of the stationarized series are called “autoregressive” that refers to (AR) terms Lags of the forecast errors are called “moving average” which refers to (MA) terms. It is basically used for forecasting Arima is a Generalized random walk models which is fine-tuned to eliminate all residual autocorrelation. It is a Generalized exponential smoothing model that can incorporate long-term trends and seasonality. The Stationarized regression model uses lags of the dependent variables and/or lags of the forecast errors as regressors. Here the forecasting model of time series can be stationarized by using transformations like differencing, logging and deflating. By this we can say that a time series is “Stationary” if all the Statistical properties like mean, variance, autocorrelation etc. are constant in time the equation is as follows:

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \ldots + \beta_p Y_{t-p}$$
$$+ \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q}$$

8 Conclusion

We present an approach to estimate how many future visitors will go to a restaurant using Python,R and supervised learning. The included data involves restaurant information, historical visits and historical reservations. With features constructed from the data, our approach generates predictions by performing regression using ARIMA model. We evaluate our approach using large-scale real-world datasets from two restaurant booking websites. The evaluation results show the effectiveness of our approach, as well as useful insights for future work.

9 Future Study

Beyond features considered by our approach, many more factors can facilitate accurate prediction of future visitors to a restaurant. For instance, bad weather of the restaurant location may reduce visitors. Also, if a new restaurant is opened next to an existing restaurant, the number of future visitors going to this existing restaurant may drop. Moreover, social events can bring more visitors to restaurants of the related venues. Hence, in future work, it is necessary to include more information to the predictive model, such as weather, competitors and social events. For some kinds of restaurants, it is challenging to accurately count how many visitors have come on a day. For example, one customer may buy food for all its friends at fast food restaurants, such as McDonald's (especially when customers in a car buy food via "Drive-Thru"). Future work can also explore new customer counting methods for restaurants based on new technology (such as combining video camera and the Internet of Things [86], or new software/hardware sensing infrastructure [4])

10 References

[1] Amir Gandomi, and Murtaza Haider, ”Beyond the hype: Big data concepts, methods, and analytics”, International Journal of Information Management, Vol 35, no. 2, pp. 137-144, 2015. [2] Daryl Ansel, and Chris Dyer, ”A framework for restaurant information technology”, Cornell Hotel and Restaurant Administration Quarterly, Vol 40, no. 3, pp. 74-84, 1999. [3] Xin Yang, Bing Pan, James A. Evans, and Benfu Lv, ”Forecasting Chinese tourist volume with search engine data”, Tourism Management, Vol 46, pp. 386-397, 2015. [4] Yang Yang, Bing Pan, and Haiyan Song, ”Predicting hotel demand using destination marketing organizations web traffic data”, Journal of Travel Research, Vol 53, no. 4, pp. 433-447, 2014. [5] Vincent Cho, ”A comparison of three different approaches to tourist arrival forecasting”, Tourism management, Vol 24, no. 3, pp. 323-330, 2003. [6] Usep Suhud, and Arifin Wibowo, ”Predicting Customers Intention to Revisit A Vintage-Concept Restaurant”, Journal of Consumer Sciences, Vol 1, no. 2 (2016)