Time Series Forecasting Using ML models

XGBoost

For classification and regression issues, XGBoost is an effective use of gradient boosting. It performs well, if not best, on a variety of predictive modelling tasks and is both quick and effective. The time series dataset must first be converted into a supervised learning problem in order to apply XGBoost for time series forecasting. Additionally, it necessitates the use of a particular evaluation method known as walk-forward validation because k-fold cross validation would yield results that are optimistically skewed.

Loading the dataset:

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import xgboost as xgb
from sklearn.metrics import mean_squared_error
color_pal = sns.color_palette()
plt.style.use('fivethirtyeight')
```

In [2]:
```
df = pd.read_csv("C://Users//stebi//OneDrive//Desktop//visitors.csv")
df = df.set_index('Date')
df.index = pd.to_datetime(df.index)
```

In [3]:
```
df
```

Out[3]:

Date	Visitors
2011-01-31	175944
2011-02-28	141230
2011-03-31	184193
2011-04-30	177894
2011-05-31	199465
...	...
2019-09-30	784280
2019-10-31	665055
2019-11-30	543176
2019-12-31	534732
2020-01-31	503451

109 rows × 1 columns

converting the Dataset to log values:

In [4]:
```
df['Visitors'] = np.log2(df['Visitors'])
# Show the dataframe
df
```

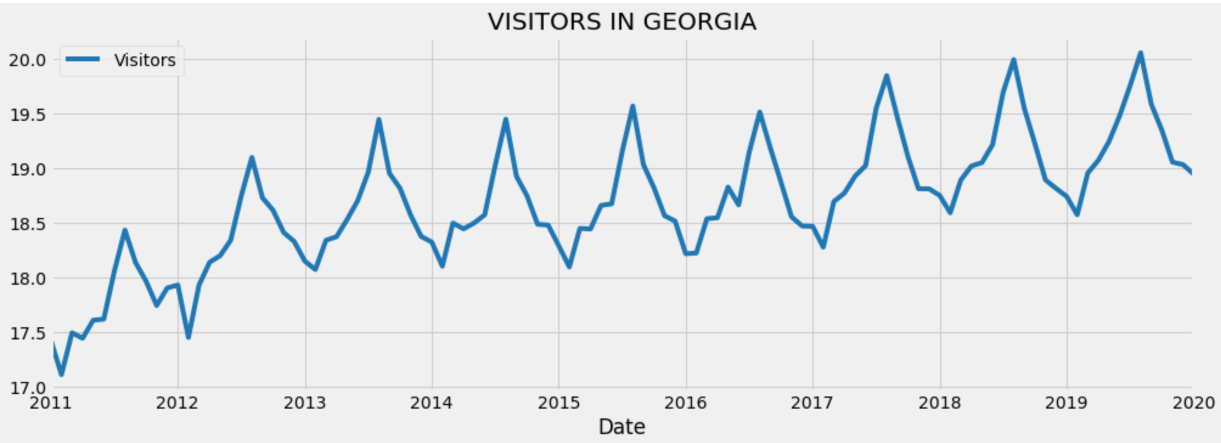
Out[4]:

Date	Visitors
2011-01-31	17.424757
2011-02-28	17.107687
2011-03-31	17.490859
2011-04-30	17.440658
2011-05-31	17.605776
...	...
2019-09-30	19.581009
2019-10-31	19.343114
2019-11-30	19.051060
2019-12-31	19.028456
2020-01-31	18.941492

109 rows × 1 columns

Time Series Plot:

In [5]:
```
df.plot(figsize = (15, 5),
        color = color_pal[0],
        title = 'VISITORS IN GEORGIA')
plt.show()
```

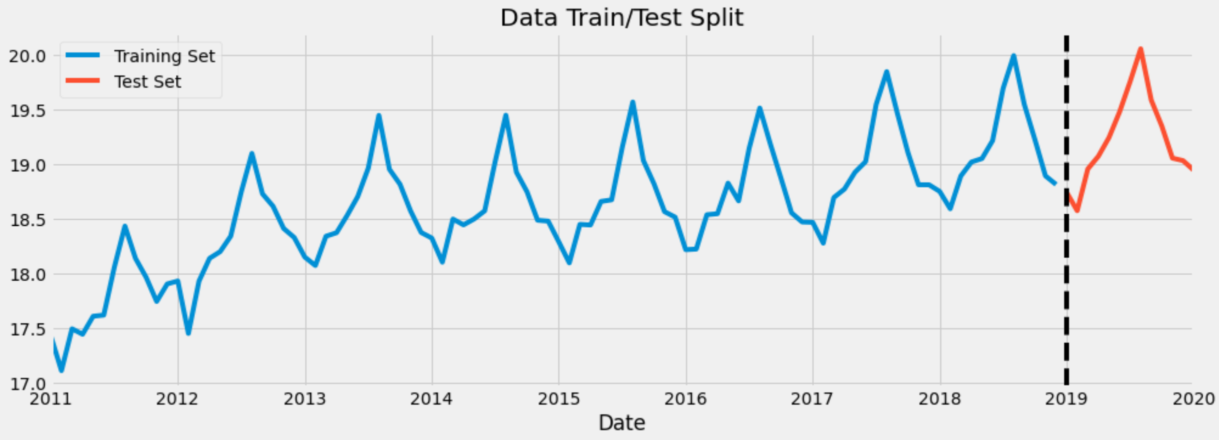


Plot showing the seasonal component is present and the model is multiplicative in nature

Splitting the Data into Training and Testing:

In [6]:
```
train = df.loc[df.index < '2019-01-01']
test = df.loc[df.index >= '2019-01-01']
```

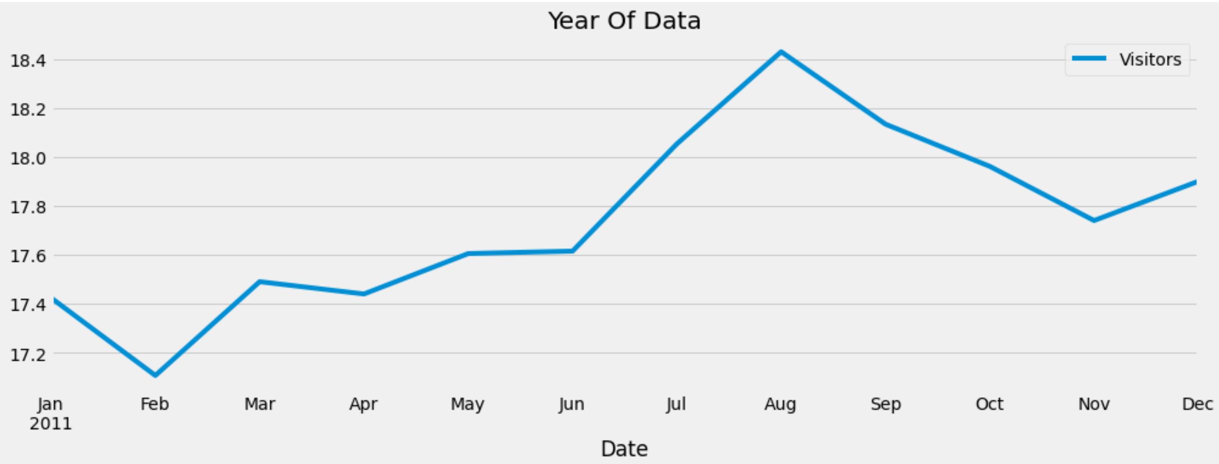
```
fig, ax = plt.subplots(figsize=(15, 5))
train.plot(ax=ax, label='Training Set', title='Data Train/Test Split')
test.plot(ax=ax, label='Test Set')
ax.axvline('2019-01-01', color='black', ls='--')
ax.legend(['Training Set', 'Test Set'])
plt.show()
```



Analysing the monthly wise data:

In [7]:

```
df.loc[(df.index >= '2011-01-01') & (df.index < '2012-01-01')]. \
plot(figsize=(15, 5), title='Year Of Data')
plt.show()
```



Feature Creation:

In [8]:

```
def create_features(df):
    """
    Create time series features based on time series index.
    """
    df = df.copy()
    df['dayofweek'] = df.index.dayofweek
    df['quarter'] = df.index.quarter
    df['month'] = df.index.month
    df['year'] = df.index.year
    df['dayofyear'] = df.index.dayofyear
    df['dayofmonth'] = df.index.day
    return df

df = create_features(df)
```

In [9]:
```
df
```

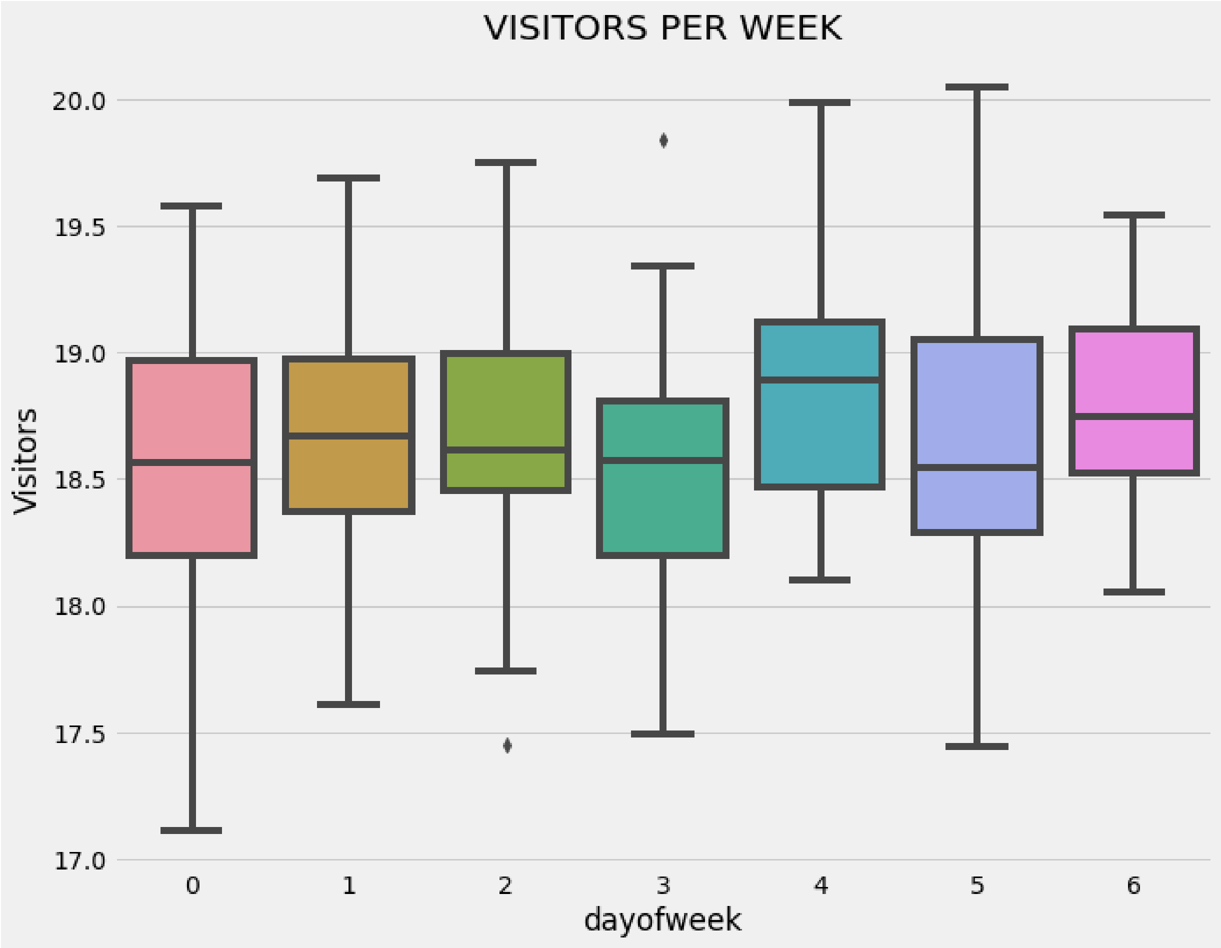
Out[9]:

Date	Visitors	dayofweek	quarter	month	year	dayofyear	dayofmonth
2011-01-31	17.424757	0	1	1	2011	31	31
2011-02-28	17.107687	0	1	2	2011	59	28
2011-03-31	17.490859	3	1	3	2011	90	31
2011-04-30	17.440658	5	2	4	2011	120	30
2011-05-31	17.605776	1	2	5	2011	151	31
...	...	...	...	...	...	...	...
2019-09-30	19.581009	0	3	9	2019	273	30
2019-10-31	19.343114	3	4	10	2019	304	31
2019-11-30	19.051060	5	4	11	2019	334	30
2019-12-31	19.028456	1	4	12	2019	365	31
2020-01-31	18.941492	4	1	1	2020	31	31

109 rows × 7 columns

Visualize our Feature / Target Relationship

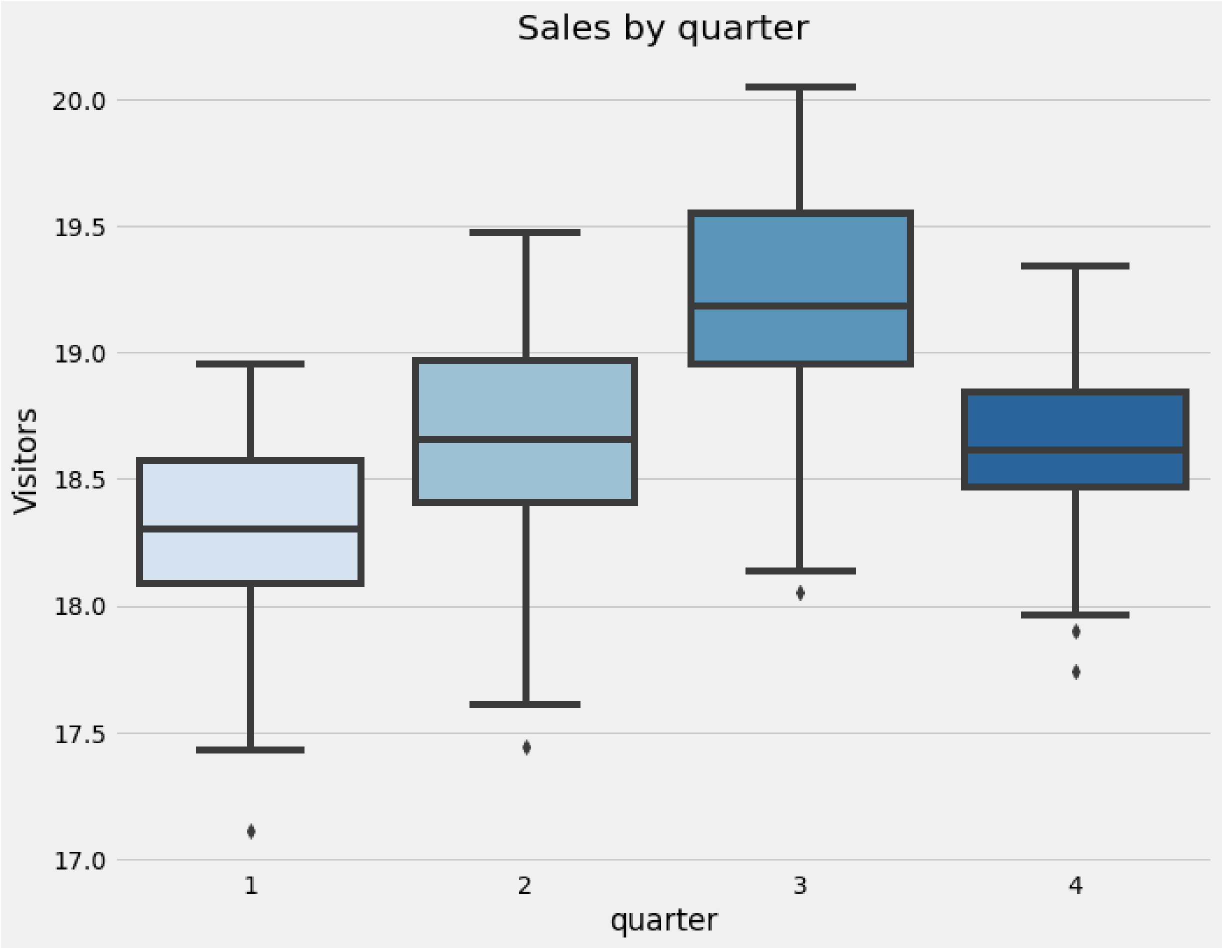
In [10]:
```
fig, ax = plt.subplots(figsize=(10, 8))
sns.boxplot(data=df, x= 'dayofweek', y = 'Visitors')
ax.set_title('VISITORS PER WEEK')
plt.()
```



There are no outliers in the data and the data is almost of equal proportion

Sales by Quarter:

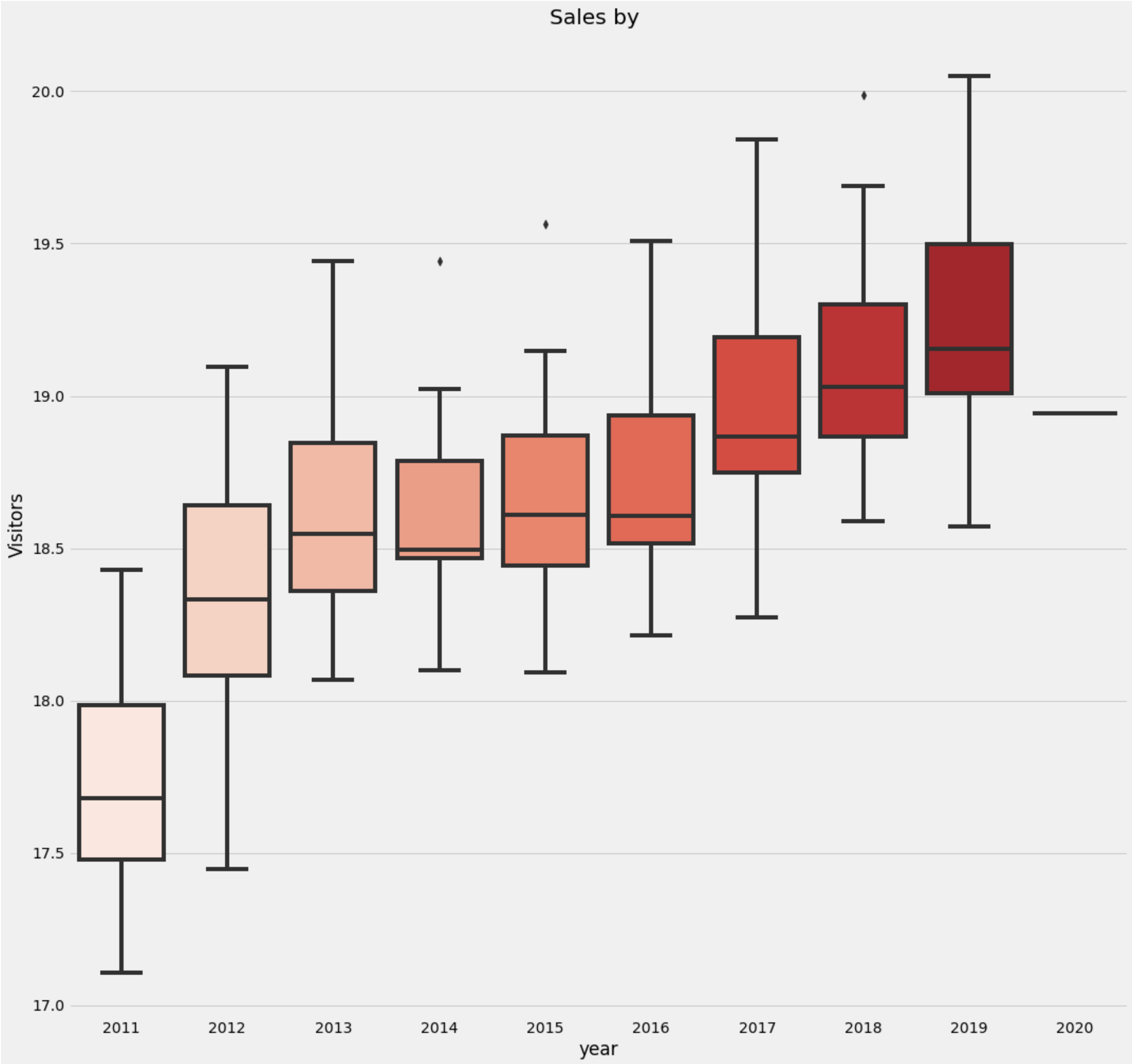
In [11]:
```
fig, ax = plt.subplots(figsize=(10, 8))
sns.boxplot(data=df, x= 'quarter', y = 'Visitors', palette = "Blues")
ax.set_title('Sales by quarter')
plt.show()
```



Yearly Wise Data:

In [12]:

```
fig, ax = plt.subplots(figsize=(16, 16))
sns.boxplot(data=df, x= 'year', y = 'Visitors', palette = "Reds")
ax.set_title('Sales by ')
plt.show()
```



CREATE THE XGBOOST MODEL

In [13]:

```
train = create_features(train)
test = create_features(test)

FEATURES = ['dayofyear', 'dayofweek', 'quarter', 'month', 'year']
TARGET = 'Visitors'

X_train = train[FEATURES]
y_train = train[TARGET]

X_test = test[FEATURES]
y_test = test[TARGET]
```

In [14]:

```
reg = xgb.XGBRegressor(base_score=0.5, booster='gbtree',
                       n_estimators=1000,
                       early_stopping_rounds=50,
                       objective='reg:linear',
                       max_depth=3,
                       learning_rate=0.01)
reg.fit(X_train, y_train,
        eval_set=[(X_train, y_train), (X_test, y_test)],
        verbose=100)
```

```
[14:50:08] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.6.0/sr
c/objective/regression_obj.cu:203: reg:linear is now deprecated in favor of reg:squa
rederror.
[0]	validation_0-rmse:17.93177	validation_1-rmse:18.53959
[100]	validation_0-rmse:6.65235	validation_1-rmse:7.25225
[200]	validation_0-rmse:2.50809	validation_1-rmse:3.04622
[300]	validation_0-rmse:0.98453	validation_1-rmse:1.40047
[400]	validation_0-rmse:0.41268	validation_1-rmse:0.70665
[500]	validation_0-rmse:0.19166	validation_1-rmse:0.39393
[600]	validation_0-rmse:0.10588	validation_1-rmse:0.26316
[700]	validation_0-rmse:0.07034	validation_1-rmse:0.20412
[800]	validation_0-rmse:0.05450	validation_1-rmse:0.17535
[900]	validation_0-rmse:0.04601	validation_1-rmse:0.15976
[999]	validation_0-rmse:0.04084	validation_1-rmse:0.15281
```

Out[14]:

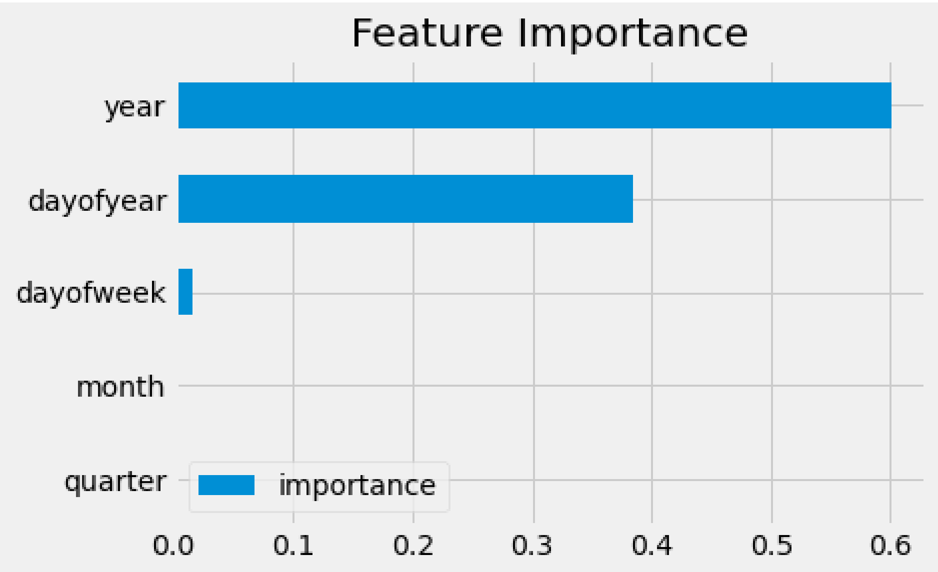
XGBRegressor

```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=50, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwis
e',
             importance_type=None, interaction_constraints='',
             learning_rate=0.01, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=3, max_leaves=0, min_child_weigh
t=1,
             missing=nan, monotone_constraints='()', n_estimators=1000,
             n_jobs=0, num_parallel_tree=1, objective='reg:linear',
```

Feature Importance

In [15]:

```
fi = pd.DataFrame(data=reg.feature_importances_,
             index=reg.feature_names_in_,
             columns=['importance'])
fi.sort_values('importance').plot(kind='barh', title='Feature Importance')
plt.show()
```

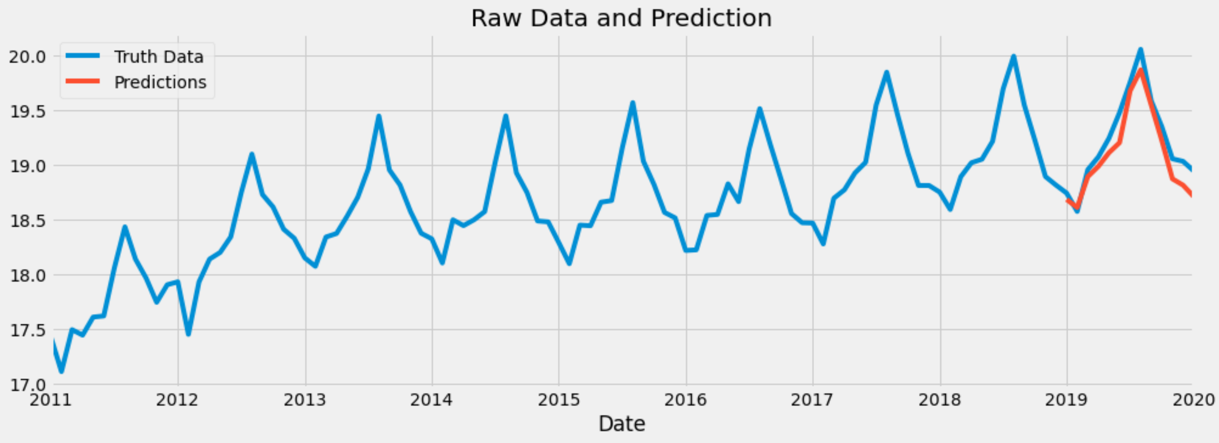


Forecast on Test

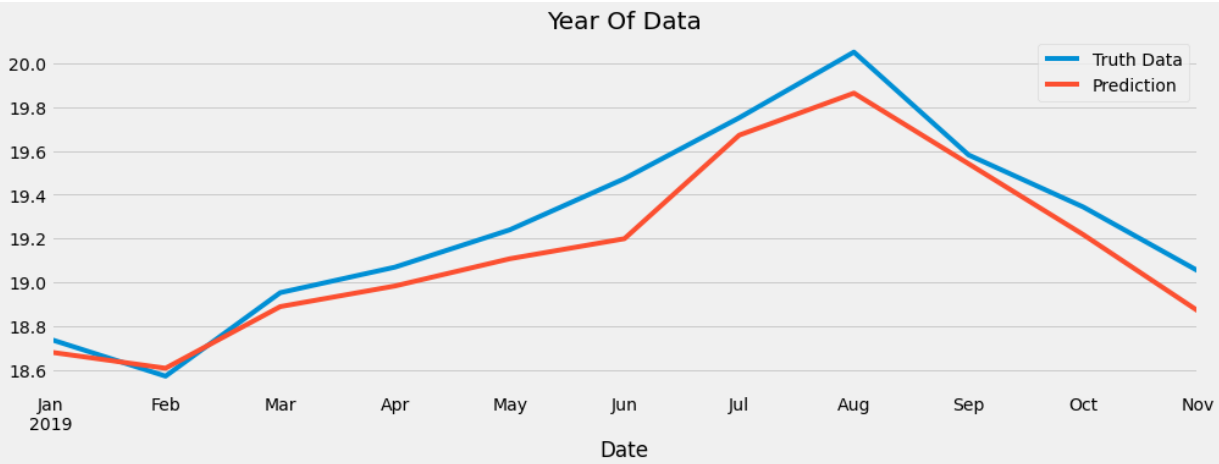
In [16]:

```
test['prediction'] = reg.predict(X_test)
df = df.merge(test[['prediction']], how='left', left_index=True, right_index=True)
ax = df[['Visitors']].plot(figsize=(15, 5))
```

```
df['prediction'].plot(ax=ax)
plt.legend(['Truth Data', 'Predictions'])
ax.set_title('Raw Data and Prediction')
plt.show()
```



In [17]:
```
ax = df.loc[(df.index >= '2019-01-01') & (df.index <= '2019-12-01')]['Visitors'] \
    .plot(figsize=(15, 5), title='Year Of Data')
df.loc[(df.index >= '2019-01-01') & (df.index <= '2019-12-01')]['prediction'] \
    .plot()
plt.legend(['Truth Data','Prediction'])
plt.show()
```



Score (RMSE)

In [18]:
```
score = np.sqrt(mean_squared_error(test['Visitors'], test['prediction']))
print(f'RMSE Score on Test set: {score:0.2f}')
```

```
RMSE Score on Test set: 0.15
```

Calculate Error

Look at the worst and best predicted days

In [19]:
```
test['error'] = np.abs(test[TARGET] - test['prediction'])
test['date'] = test.index.date
test.groupby(test['date'])['error'].mean().sort_values(ascending=False).head(10)
```

Out[19]:
```
date
2019-06-30    0.274338
```

```
2020-01-31    0.233608
2019-12-31    0.217738
2019-08-31    0.187662
2019-11-30    0.183095
2019-05-31    0.132168
2019-10-31    0.126436
2019-04-30    0.086094
2019-07-31    0.079347
2019-03-31    0.063762
Name: error, dtype: float64
```

The obtained Rmse score is 0.15 which is higher than the rmse score of SARIMA and WINTERS models and hence XGBOOST is not a good candidate for forecasting the output.

In [ ]:

TIME SERIES-CASE STUDY

GEORGIA VISITORS FORECAST

STEBIN GEORGE

21122061

INTRODUCTION:

Georgia country becomes one of the top travel destinations in recent years. Understanding the characteristics of the time series representing international visits to the country provides valuable insights for business. In this project, the number of visitors that would probably turn up in coming years is calculated using SARIMA, Winters and XGBOOST models. The outcome is used in budgeting and revenue planning for one of the local hotels. Some of the most effective forecasting methods in time series analysis include SARIMA and WINTERS Exponential Smoothing . The Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and Mean Absolute Square Error are used to measure the accuracy of the fitted models (MAE)

SARIMA MODEL:

Autoregressive Integrated Moving Average, or ARIMA, is one of the most widely used forecasting methods for univariate time series data forecasting. Although the method can handle data with a trend, it does not support time series with a seasonal component. An extension to ARIMA that supports the direct modeling of the seasonal component of the series is called SARIMA. The extension of ARIMA known as Seasonal Autoregressive Integrated Moving Average, or Seasonal ARIMA, specifically supports univariate time series data with a seasonal component. There are three trend components that need setting up. They match the ARIMA model in the following ways: p: Order of trend autoregression. d: Order of trend difference. q: Order of the trend moving average. There may be many different parameters and term combinations in seasonal ARIMA models. As a result, it is appropriate to test out a variety of models when fitting them to data and then

Winters Smoothing:

Numerous previous data are compressed using exponential smoothing via the Holt-Winters method in order to anticipate “typical” values for the present and the future. Exponential smoothing is the process of “smoothing” a time series using an exponentially weighted moving average (EWMA). The three order parameters of a Holt-Winters model are alpha, beta, and gamma. The level smoothing coefficient is specified by alpha. The coefficient for the trend smoothing is specified by beta. The coefficient for the seasonal smoothing is specified by gamma. The type of seasonality has a parameter as well: seasonality that is additive and has a fixed number of seasons. Seasonality that changes by a factor multiplicatively.

OBJECTIVE:

- A comparative study is performed to forecast the Visitors count in Georgia for the next five years using ARIMA and Holts exponential with the help of accuracy measures.

DATA SOURCE:

A time-series approach is used in the monthly wise Visitors data from 2011 to 2020 Jan, from the official website of Georgian tourism department. - https://knoema.com/atlas/Georgia/datasets

ANALYSIS:

Loading the dataset:

```
record_1 <- read.csv("C:\\Users\\stebi\\OneDrive\\Desktop\\visitors.csv")
record_1

##          Date Visitors
## 1  31-01-2011   175944
## 2  28-02-2011   141230
## 3  31-03-2011   184193
## 4  30-04-2011   177894
## 5  31-05-2011   199465
## 6  30-06-2011   200852
## 7  31-07-2011   272101
## 8  31-08-2011   353191
## 9  30-09-2011   287727
## 10 31-10-2011   255330
## 11 30-11-2011   219011
## 12 31-12-2011   244759
## 13 31-01-2012   249356
## 14 29-02-2012   178749
## 15 31-03-2012   249111
## 16 30-04-2012   287868
## 17 31-05-2012   300318
## 18 30-06-2012   331347
## 19 31-07-2012   439289
## 20 31-08-2012   559802
## 21 30-09-2012   433255
## 22 31-10-2012   400427
## 23 30-11-2012   347861
## 24 31-12-2012   328491
## 25 31-01-2013   290114
## 26 28-02-2013   275015
## 27 31-03-2013   331060
## 28 30-04-2013   338558
## 29 31-05-2013   377694
```

```
## 30  30-06-2013   425624
## 31  31-07-2013   510039
## 32  31-08-2013   712493
## 33  30-09-2013   506072
## 34  31-10-2013   460203
## 35  30-11-2013   388519
## 36  31-12-2013   339057
## 37  31-01-2014   327326
## 38  28-02-2014   280861
## 39  31-03-2014   369236
## 40  30-04-2014   355844
## 41  31-05-2014   369341
## 42  30-06-2014   388906
## 43  31-07-2014   532717
## 44  31-08-2014   713435
## 45  30-09-2014   496842
## 46  31-10-2014   438920
## 47  30-11-2014   366455
## 48  31-12-2014   364448
## 49  31-01-2015   319735
## 50  28-02-2015   279589
## 51  31-03-2015   357000
## 52  30-04-2015   355549
## 53  31-05-2015   412806
## 54  30-06-2015   417043
## 55  31-07-2015   581264
## 56  31-08-2015   775545
## 57  30-09-2015   535073
## 58  31-10-2015   461760
## 59  30-11-2015   386894
## 60  31-12-2015   373741
## 61  31-01-2016   304060
## 62  29-02-2016   305257
## 63  31-03-2016   379540
## 64  30-04-2016   381874
## 65  31-05-2016   463705
## 66  30-06-2016   414228
## 67  31-07-2016   578494
## 68  31-08-2016   746414
## 69  30-09-2016   593962
## 70  31-10-2016   478741
## 71  30-11-2016   384125
## 72  31-12-2016   362416
## 73  31-01-2017   361707
## 74  28-02-2017   316852
## 75  31-03-2017   423173
## 76  30-04-2017   446436
## 77  31-05-2017   496801
## 78  30-06-2017   531224
## 79  31-07-2017   763593
```

```
## 80  31-08-2017   940129
## 81  30-09-2017   722189
## 82  31-10-2017   563122
## 83  30-11-2017   458869
## 84  31-12-2017   458735
## 85  31-01-2018   439918
## 86  28-02-2018   394105
## 87  31-03-2018   484989
## 88  30-04-2018   529892
## 89  31-05-2018   541752
## 90  30-06-2018   606792
## 91  31-07-2018   845588
## 92  31-08-2018  1040544
## 93  30-09-2018   763529
## 94  31-10-2018   611152
## 95  30-11-2018   485319
## 96  31-12-2018   459770
## 97  31-01-2019   437218
## 98  28-02-2019   389218
## 99  31-03-2019   507064
## 100 30-04-2019   549761
## 101 31-05-2019   618709
## 102 30-06-2019   727634
## 103 31-07-2019   882331
## 104 31-08-2019  1086596
## 105 30-09-2019   784280
## 106 31-10-2019   665055
## 107 30-11-2019   543176
## 108 31-12-2019   534732
## 109 31-01-2020   503451
```

The dataset has 109 rows with the target column as the Visitors.

Converting to a Time-Series Data:

```
library(tseries)

## Warning: package 'tseries' was built under R version 4.1.3

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

visit=ts(record_1$'Visitors',start=c(2011,1),end=c(2020,1),frequency =12)
visit

##          Jan     Feb     Mar     Apr     May     Jun     Jul     Aug
Sep
## 2011  175944  141230  184193  177894  199465  200852  272101  353191
287727
## 2012  249356  178749  249111  287868  300318  331347  439289  559802
433255
```

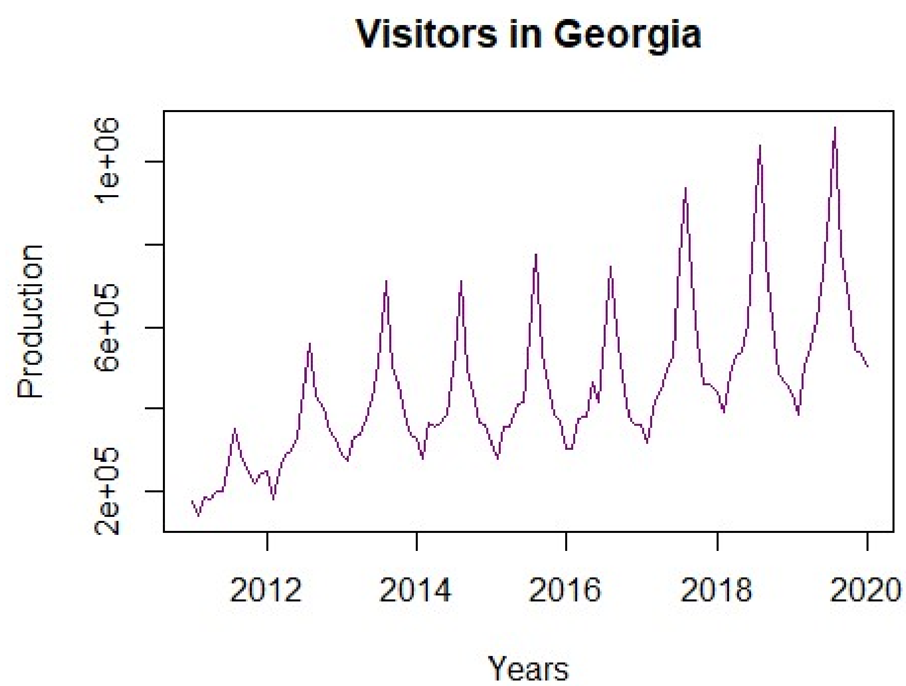
```
## 2013  290114  275015  331060  338558  377694  425624  510039  712493
506072
## 2014  327326  280861  369236  355844  369341  388906  532717  713435
496842
## 2015  319735  279589  357000  355549  412806  417043  581264  775545
535073
## 2016  304060  305257  379540  381874  463705  414228  578494  746414
593962
## 2017  361707  316852  423173  446436  496801  531224  763593  940129
722189
## 2018  439918  394105  484989  529892  541752  606792  845588 1040544
763529
## 2019  437218  389218  507064  549761  618709  727634  882331 1086596
784280
## 2020  503451
##          Oct     Nov     Dec
## 2011  255330  219011  244759
## 2012  400427  347861  328491
## 2013  460203  388519  339057
## 2014  438920  366455  364448
## 2015  461760  386894  373741
## 2016  478741  384125  362416
## 2017  563122  458869  458735
## 2018  611152  485319  459770
## 2019  665055  543176  534732
## 2020
```

Defining the Class and Length:

```
class(visit)

## [1] "ts"

length(visit)

## [1] 109
```

Plotting the time series Graph:

```
ts.plot(visit,main='Visitors in
Georgia',xlab='Years',ylab='Production',col='darkmagenta')
```



From the graph it can be noted that the seasonal components are evident and the model is of multiplicative nature.

Checking whether data is nonseasonal or not:

```
library(seastests)

## Warning: package 'seastests' was built under R version 4.1.3

isSeasonal(visit)

## [1] TRUE
```

After the data is imported, the information is converted into time series and plotted. We observe an Multiplicative model with a trend and Seasonal component in the plot. Here ADF test is performed to check the stationarity.

ADF_TEST:

Null Hypothesis: The series is unit root non-stationary Alternate Hypothesis: The series is unit root stationary

Since p value obtained 0.01 which is less than 5% significance level, we could conclude that the data is Stationary.

```
adf.test(visit)

## Warning in adf.test(visit): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  visit
## Dickey-Fuller = -7.1881, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

Since the P value is less than .05 we reject the null hypothesis and states that the dataset is Stationary.

```
visit=log(visit)
visit
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
Aug
## 2011 12.07792 11.85815 12.12374 12.08894 12.20339 12.21032 12.51393
12.77476
## 2012 12.42664 12.09374 12.42565 12.57026 12.61260 12.71092 12.99291
13.23534
## 2013 12.57803 12.52458 12.71005 12.73245 12.84184 12.96131 13.14224
13.47653
## 2014 12.69871 12.54562 12.81919 12.78225 12.81948 12.87109 13.18575
13.47785
## 2015 12.67525 12.54108 12.78549 12.78142 12.93073 12.94094 13.27296
13.56132
## 2016 12.62498 12.62891 12.84672 12.85285 13.04700 12.93417 13.26818
13.52304
## 2017 12.79859 12.66619 12.95554 13.00905 13.11594 13.18294 13.54579
13.75377
## 2018 12.99434 12.88437 13.09188 13.18043 13.20256 13.31594 13.64779
13.85525
## 2019 12.98819 12.87189 13.13639 13.21724 13.33539 13.49755 13.69032
13.89856
## 2020 13.12924
##           Sep      Oct      Nov      Dec
## 2011 12.56977 12.45031 12.29688 12.40803
## 2012 12.97908 12.90029 12.75956 12.70226
## 2013 13.13443 13.03942 12.87010 12.73392
## 2014 13.11603 12.99207 12.81163 12.80614
## 2015 13.19016 13.04280 12.86591 12.83132
## 2016 13.29457 13.07892 12.85872 12.80055
## 2017 13.49004 13.24125 13.03652 13.03623
## 2018 13.54571 13.32310 13.09256 13.03848
## 2019 13.57252 13.40763 13.20519 13.18952
## 2020
```

The log value is taken to convert the multiplicative natured data into a additive model.

```
visit1=visit[1:104]
visit2=ts(visit1,start=c(2011,1),end=c(2019,8),frequency =12)
visit2
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
Aug
## 2011 12.07792 11.85815 12.12374 12.08894 12.20339 12.21032 12.51393
12.77476
## 2012 12.42664 12.09374 12.42565 12.57026 12.61260 12.71092 12.99291
13.23534
## 2013 12.57803 12.52458 12.71005 12.73245 12.84184 12.96131 13.14224
13.47653
## 2014 12.69871 12.54562 12.81919 12.78225 12.81948 12.87109 13.18575
13.47785
## 2015 12.67525 12.54108 12.78549 12.78142 12.93073 12.94094 13.27296
13.56132
## 2016 12.62498 12.62891 12.84672 12.85285 13.04700 12.93417 13.26818
13.52304
## 2017 12.79859 12.66619 12.95554 13.00905 13.11594 13.18294 13.54579
13.75377
## 2018 12.99434 12.88437 13.09188 13.18043 13.20256 13.31594 13.64779
13.85525
## 2019 12.98819 12.87189 13.13639 13.21724 13.33539 13.49755 13.69032
13.89856
##           Sep      Oct      Nov      Dec
## 2011 12.56977 12.45031 12.29688 12.40803
## 2012 12.97908 12.90029 12.75956 12.70226
## 2013 13.13443 13.03942 12.87010 12.73392
## 2014 13.11603 12.99207 12.81163 12.80614
## 2015 13.19016 13.04280 12.86591 12.83132
## 2016 13.29457 13.07892 12.85872 12.80055
## 2017 13.49004 13.24125 13.03652 13.03623
## 2018 13.54571 13.32310 13.09256 13.03848
## 2019
```

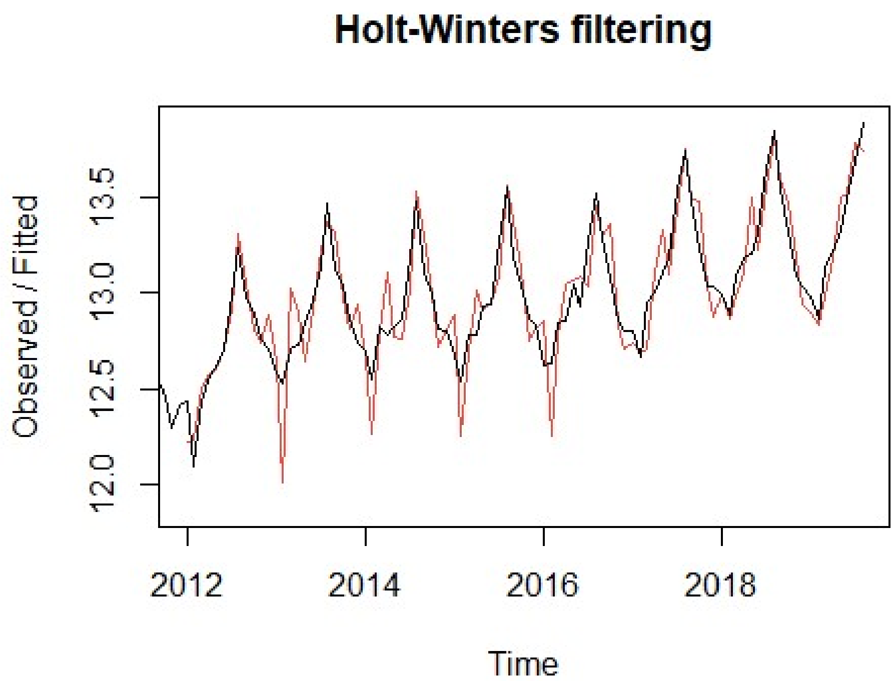
Here the dataset is created again so that we can perform an In-Sample forecast such that the last 5 known values with in the dataset is used to prediction so that the accuracy measures can be obtained by calculating with the known values.

WINTER'S EXPONENTIAL SMOOTHING

```
library(astsa)
```

```
## Warning: package 'astsa' was built under R version 4.1.3
```

```
seasonal_visit=HoltWinters(visit2,beta= TRUE,gamma = TRUE)
plot(seasonal_visit)
```



The graph shows the actual and the fitted line using the winters model.

```
library(forecast)

## Warning: package 'forecast' was built under R version 4.1.3

##
## Attaching package: 'forecast'

## The following object is masked from 'package:astsa':
##
##     gas

forecast_data=forecast(seasonal_visit,h=5)
forecast_data

##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Sep 2019       13.57432 13.37801 13.77062 13.27410 13.87454
## Oct 2019       13.48947 13.10641 13.87254 12.90362 14.07532
## Nov 2019       13.47767 12.85301 14.10233 12.52233 14.43300
## Dec 2019       13.56202 12.65482 14.46923 12.17457 14.94947
## Jan 2020       13.57292 12.34846 14.79738 11.70027 15.44557
```

The forecast of the last 5 known values in the dataset is forecasted here. This value along with the actual value is used for evaluating the performance of the model.

Evaluation Metrics:

```
library(Metrics)
```

```
## Warning: package 'Metrics' was built under R version 4.1.3

##
## Attaching package: 'Metrics'

## The following object is masked from 'package:forecast':
##
##     accuracy

actual_data=c(13.57252,13.40763,13.20519,13.18952,13.12924)
predict_data=c(13.57432,13.48947,13.47767,13.56202,13.57292)
result_1=rmse(actual_data,predict_data)
result_1

## [1] 0.2886363
```

The rmse value of the model is 0.2886363.

```
result_2=mape(actual_data,predict_data)
result_2

## [1] 0.01778126
```

The mape value of the model is 00.001778145.

```
result_3=mase(actual_data,predict_data)
result_3

## [1] 2.115683
```

The mase value of the model is 2.115683

RESIDUAL ANALYSIS : HOLTS EXPONENTIAL SMOOTHING:

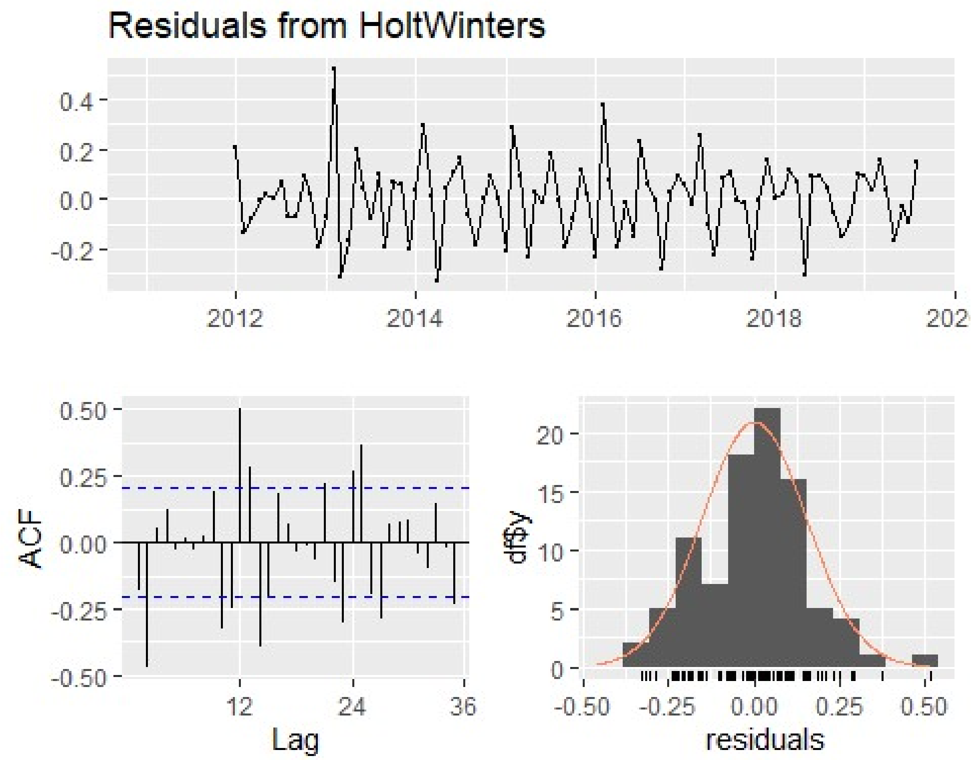
ASSUMPTIONS:

- Residuals are uncorrelated
- Residuals are normally distributed

The "residuals" in a time series model are what is left over after fitting a model. For many (but not all) time series models, the residuals are equal to the difference between the observations and the corresponding fitted values.

```
res <- resid(forecast_data)
#Checing whether the residuals are uncorrelated
Box.test(res, type = "Ljung-Box")

##
##  Box-Ljung test
##
## data:  res
## X-squared = 3.2643, df = 1, p-value = 0.0708
```

```
forecast::checkresiduals(forecast_data)

## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```




```
# normality test
shapiro.test(res)

##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.97488, p-value = 0.07246
```

Here, we have done the Winters exponential smoothing in the sample forecast and the values for 2019 and 2020 are predicted and compared with the original values and obtained the rmse,mape, and mase. Here we can see that mase and mape is significantly less and rmse is not very high. Hence it is a good model, and when we check the residuals, we obtain that the residual assumptions are not followed. Since both the p-values are less than .05 they are correlated and are not normally distributed.Hence the proposed model could be increased since much of the informationa re in the error/residual as well.

SARIMA:

Setting the value of seasonal = TRUE in the time Ariama model will make the sarima model. Auto function fetches the best value for p,q,d in the pool of data and returns the best model with minimum error and better forecasting rate.

```
library(forecast)
#Model is fitted
arima_fit=auto.arima(visit2,seasonal = TRUE)
arima_fit

## Series: visit2
## ARIMA(0,1,1)(0,1,1)[12]
##
## Coefficients:
##           ma1     sma1
##       -0.4321  -0.5231
## s.e.   0.0904   0.1322
##
## sigma^2 = 0.004584:  log likelihood = 115.1
## AIC=-224.2   AICc=-223.92   BIC=-216.67
```

ARIMA(0,1,1)(0,1,1)[12] is the best Sarima model for the given dataset.The Akaike information criterion (AIC) is a mathematical method for evaluating how well a model fits the data it was generated from. In statistics, AIC is used to compare different possible models and determine which one is the best fit for the data.

Forecasting usig the SARIMA:

```
forecast_data<-forecast(arima_fit,h=5)
forecast_data

##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Sep 2019       13.62989 13.54312 13.71666 13.49719 13.76259
## Oct 2019       13.41525 13.31547 13.51504 13.26265 13.56786
## Nov 2019       13.20078 13.08949 13.31207 13.03058 13.37098
## Dec 2019       13.16233 13.04062 13.28404 12.97619 13.34847
## Jan 2020       13.10692 12.97561 13.23823 12.90610 13.30774
```

Evaluation Metrics:

```
actual_data=c(13.57252,13.40763,13.20519,13.18952,13.12924)
predict_data=c(13.62989,13.41525,13.20078,13.16233,13.10692)
result_1=rmse(actual_data,predict_data)
result_1

## [1] 0.03035229
```

The rmse value is 0.03035229.

```
result_2=mape(actual_data,predict_data)
result_2
```

```
## [1] 0.001778145
```

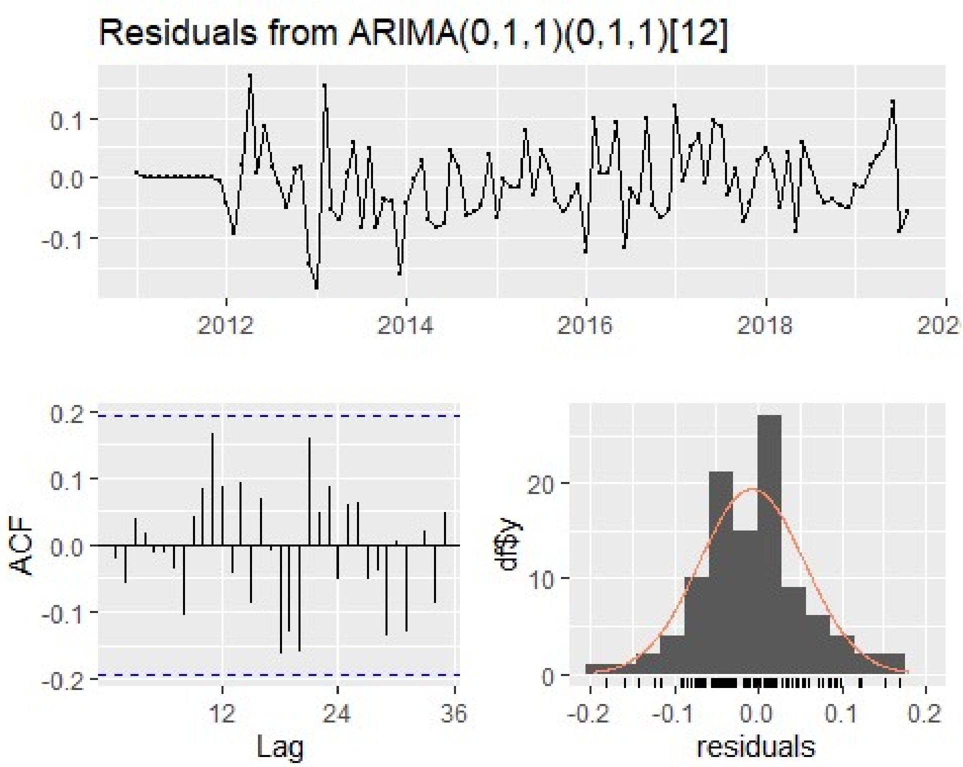
The mape value is 0.001778145

```
result_3=mase(actual_data,predict_data)
result_3

## [1] 0.2146003
```

The mase value is 0.2146003

RESIDUAL ANALYSIS -SARIMA:

```
res <- resid(forecast_data)
#Checing whether the residuals are uncorrelated
Box.test(res, type = "Ljung-Box")

##
##  Box-Ljung test
##
## data:  res
## X-squared = 0.045685, df = 1, p-value = 0.8307

forecast::checkresiduals(forecast_data)
```

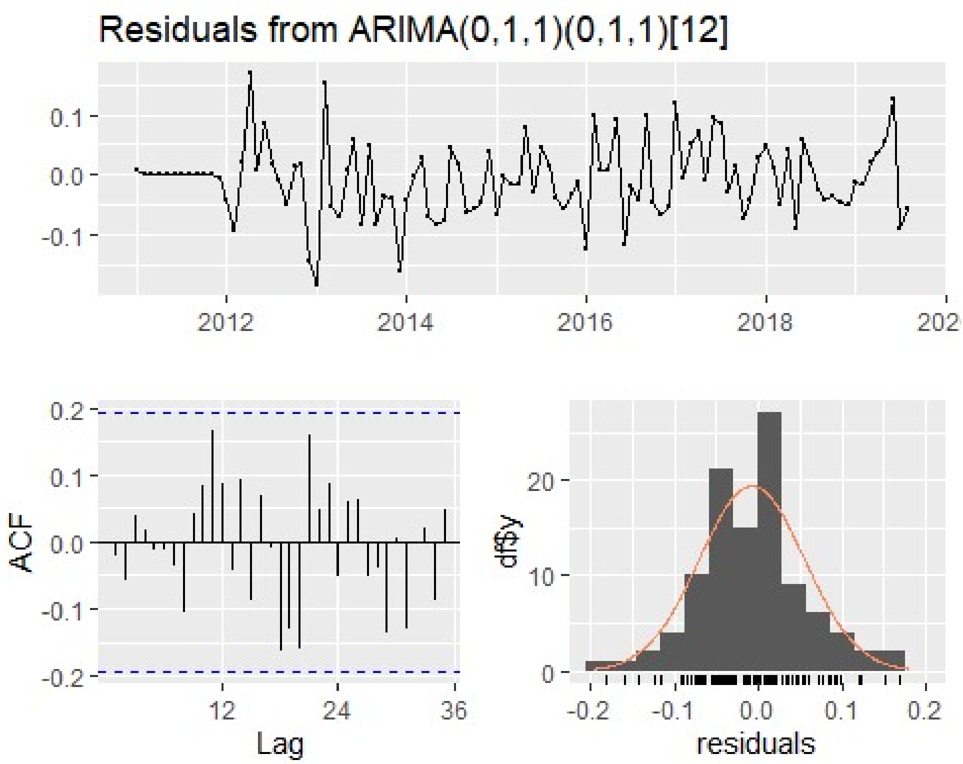



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1)(0,1,1)[12]
```

```
## Q* = 22.401, df = 19, p-value = 0.2648
##
## Model df: 2.   Total lags used: 21
```

RESIDUAL ANALYSIS: ARIMA

```
res <- resid(arima_fit)
#Checing whether the residuals are uncorrelated
Box.test(res, type = "Ljung-Box")

##
##  Box-Ljung test
##
## data:  res
## X-squared = 0.045685, df = 1, p-value = 0.8307

forecast::checkresiduals(arima_fit)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1)(0,1,1)[12]
## Q* = 22.401, df = 19, p-value = 0.2648
##
## Model df: 2.   Total lags used: 21
```

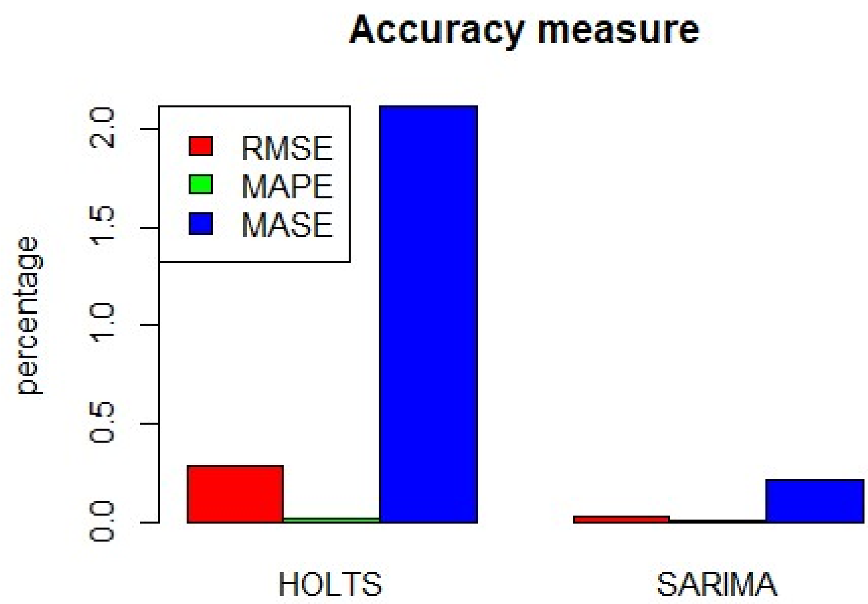
normality test

```
shapiro.test(res)

##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.98726, p-value = 0.4268
```

Since all lags in acf and pacf doesn’t lies inside the threshold line, the correlations in the all lags are not negligible. Therefore, the residuals are correlated. In residuals we observe that they are correlated and not normally distributed. Hence SARIMA also doesn’t follows the residual assumptions.

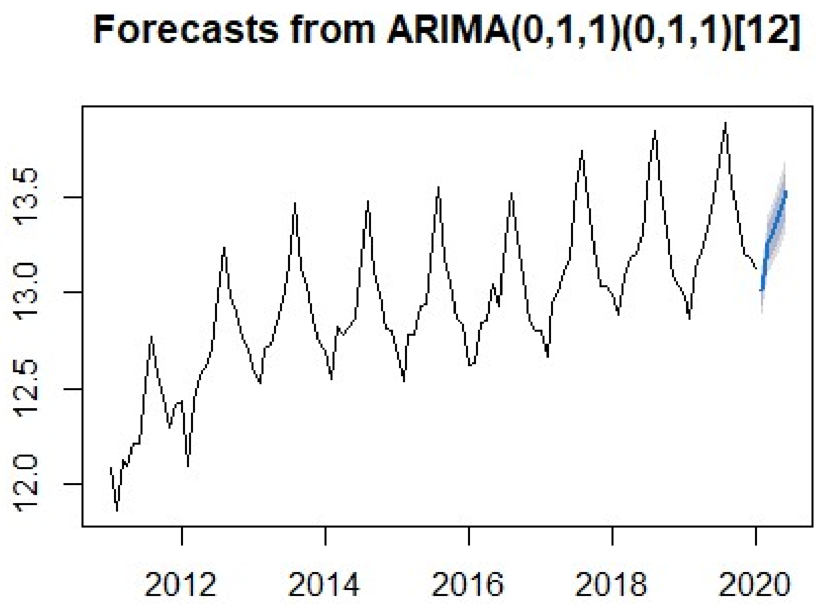
COMPARING THESE TWO MODELS.

```
# Create a data frame
col1 <- c(0.2886363,0.01778126,2.115683)
col2 <- c( 0.03035229,0.001778145,0.2146003)

data <- data.frame(col1,col2)
names(data) <- c("HOLTS","SARIMA")

# barplot with colors. Make sure that the plot and legends have same colors
for items.
barplot(height=as.matrix(data), main="Accuracy measure", ylab="percentage",
beside=TRUE,col=rainbow(3))

#Add legends
legend("topleft", c("RMSE","MAPE","MASE"),fill=rainbow(3))
```



OUT-SAMPLING FORECASTING USING THE BEST MODEL.

```
library(astsa)
#fitted the model
best_model=auto.arima(visit,seasonal = TRUE)
library(forecast)
# prediction for next 5 years
forecast_data=forecast(best_model,h=5)
forecast_data

##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Feb 2020       13.01424 12.92947 13.09900 12.88459 13.14388
## Mar 2020       13.26339 13.16552 13.36127 13.11371 13.41308
## Apr 2020       13.32962 13.22020 13.43904 13.16228 13.49697
## May 2020       13.42687 13.30701 13.54673 13.24356 13.61018
## Jun 2020       13.53247 13.40301 13.66193 13.33448 13.73046

plot(forecast_data)
```



Conclusion:

In the fields of business and industry, economics, medical, and finance, forecasting is a more significant problem that is used for planning and decision-making. We calculated Georgia's Visitors Rate for this study. The projected values are the ones that are closest to the actual numbers, while some of the forecasting values are even closer.From the comparision between the Sarima and Winters model Sarima performed better in terms of rmse,mase and mape values.

The available observed value for SARIMA is nearly closest to the anticipated value according to the model's verification using the observed data that is currently available and the forecasts from 2011 to 2020. In comparison to ARIMA, the SARIMA approach has the advantage of providing superior accuracy in RMSE, MAPE, and MASE. The production anticipated by this model indicated an increase in visits over the following years. Therefore, to draw more visitors to the nation, more and better tourist-friendly destinations and packages should be added. Better facilities should also be created in advance for the visitors. SARIMA is hence the strategy utilised to predict future values.