D-214 DATA ANALYTICS GRADUATE CAPSTONE

TASK 2: DATA ANALYTICS REPORT AND EXECUTIVE SUMMARY

PAOLA WILLIAMS

COLLEGE OF IT, WESTERN GOVERNOR'S UNIVERSITY

JUNE 29th, 2023

Table of Contents

A. RESEARCH QUESTION

To what extent can call volume be predicted over time? Call volume can be predicted with 95% accuracy using RMSE as the measure of accuracy of our model. This hypothesis follows the assumption that the residuals are normally distributed, thus the observed values fall between +/- 2 x RMSE from the predicted values (Frost, Root Mean Square Error (RMSE))..

The Customer Support department is interested in having a forecast of calls received, so they can make certain they have the right number of agents to answer calls and help customers in an efficient way. The department would benefit from forecasting call volume to build a headcount and capacity plan and ensure the department reaches its performance goals as well as following their budget regarding salaries.

B. DATA COLLECTION

The dataset that will be used contains the calls received for the last 2 years.

It consists of 9 variables (Call Id, Date, Agent, Department, Answered Y/N, Resolved, Speed of Answer, AvgTalkDuration, and Satisfaction rating). It contains around 41,000 records where every record in the dataset represents a call received at the Call Center.

The dataset is posted on the website Kaggle.com which is a free open-to-public web repository. For the analysis, we downloaded the dataset as a .csv file and then loaded it into to a Jupyter Notebook using VScode.

An advantage of using a .csv file from Kaggle is that we can use it with no restrictions since it is public and free. We can use the .csv file for analyzing and predicting as we have done before during the program.

A disadvantage of this methodology is that at first, we do not know for sure how the data will look and if it will be a good fit for our analysis and model. After the exploratory analysis including the ACF and PACF analysis, we found that the ARMA model can be used for our dataset and our data will fit the model in an accurate way.

There were no challenges in the process of collecting the data.

C. DATA EXTRACTION AND PREPARATION

The following steps were taken to prepare the data for analysis:

1. Import libraries needed for data preparation.

Figure 1

Importing Python libraries

```
import pandas as pd
import numpy as np
from statsmodels.tsa.stattools import adfuller
import matplotlib.pyplot as plt
import datetime
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.graphics.tsaplots import plot_pacf
from scipy import signal
from pmdarima import auto_arima
import warnings
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn.model_selection import train_test_split
import datetime
```

✓ 0.3s Python

2. Loading the dataset into the Jupyter Notebook, checking for the variable's data types and look at the first 5 rows.

Figure 2

Importing the data file into a Pandas Dataframe

```python
df= pd.read_csv(r'C:\Users\paowm\Downloads\Call-Center-Dataset.csv')
```

✓ 0.1s Python

```python
df.info()
```

✓ 0.0s Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44895 entries, 0 to 44894
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Call Id              40517 non-null  object 
 1   Date                 40517 non-null  object 
 2   Agent                40517 non-null  object 
 3   Department           40517 non-null  object 
 4   Answered (Y/N)       40517 non-null  object 
 5   Resolved             40517 non-null  object 
 6   Speed of Answer      32759 non-null  float64
 7   AvgTalkDuration      32759 non-null  object 
 8   Satisfaction rating  32759 non-null  float64
dtypes: float64(2), object(7)
memory usage: 3.1+ MB
```

```python
df.head()
```

✓ 0.1s Python

	Call Id	Date	Agent	Department	Answered (Y/N)	Resolved	Speed of Answer	AvgTalkDuration	Sati
0	ID0001	1/1/2015 9:12	Diane	Washing Machine	Y	Y	109.0	0:02:23	
1	ID0002	1/1/2015 9:12	Becky	Air Conditioner	Y	N	70.0	0:04:02	
2	ID0003	1/1/2015 9:47	Stewart	Washing Machine	Y	Y	10.0	0:02:11	
3	ID0004	1/1/2015 9:47	Greg	Washing Machine	Y	Y	53.0	0:00:37	
4	ID0005	1/1/2015 10:00	Becky	Toaster	Y	Y	95.0	0:01:00	

3. Since the datatype of Date is object, we would need to change it to datetime64 to be able to create visualizations with the Date variable as the x-axis.

Figure 3

Changing data type of Date

```
df['Date'] = pd.to_datetime(df['Date'])
#df['Date2'] = df['Date'].dt.strftime('%m/%d/%Y')
df=df[['Call Id','Date']]
df = df.astype({'Date':'datetime64'})
print(df)
```

✓ 0.1s

```
      Call Id       Date
0      ID0001 2015-01-01
1      ID0002 2015-01-01
2      ID0003 2015-01-01
3      ID0004 2015-01-01
4      ID0005 2015-01-01
...       ...        ...
40512 ID40465 2016-12-21
40513 ID40466 2016-12-21
40514 ID40467 2016-12-21
40515 ID40468 2016-12-21
40516 ID40469 2016-12-21

[40517 rows x 2 columns]
```

4. We group the calls by Date with the function value_counts() and name the new column “call_volume” with the function rename_axis.

Figure 4

Call volume variable preparation

```
df = df['Date'].value_counts().rename_axis('call_date').reset_index(name='call_volume').sort_values('call_date').set_index(keys=['call_date'])
df
```

✓ 0.1s

	call_volume
call_date	
2015-01-01	58
2015-01-02	60
2015-01-03	58
2015-01-04	54
2015-01-05	56
...	...
2016-12-17	68
2016-12-18	71
2016-12-19	76
2016-12-20	81
2016-12-21	101

719 rows × 1 columns

5. We check for nulls with the isnull() function.

Figure 5

Checking for nulls

```
nulls = df.isnull().sum()
nulls
```

✓ 0.0s Python Python

```
call_volume    0
dtype: int64
```

6. To check if the data is stationary, we perform the ADFuller test. If the p-value is less than 0.05 then the data is stationary.

Figure 6

Performing ADFuller test for stationarity

```
df_test = adfuller(df['call_volume'], autolag='AIC')
df_output = pd.Series(df_test[0:4], index=['Test Statistic','p-value','# lags used','# of observations'])

for key,value in df_test[4].items():
    df_output['Critical// Value(%s)'%key] = value

print(df_output)
```

✓ 0.1s

```
Test Statistic          -6.281912e+00
p-value                  3.778373e-08
# lags used              1.600000e+01
# of observations        7.020000e+02
Critical// Value(1%)    -3.439700e+00
Critical// Value(5%)    -2.865666e+00
Critical// Value(10%)   -2.568967e+00
dtype: float64
```

7. Now that we know that the data is stationary, we can split it into the training and testing sets. We will split the dataset into 80% for the training set and 20% for testing.

Figure 7

Splitting dataset into training and testing sets

```
train, test = train_test_split(df, test_size=0.2, shuffle=False)
train

print('train shape', train.shape)
print('test shape', test.shape)
```

✓ 0.0s

```
train shape (575, 1)
test shape (144, 1)
```

Figure 8

Training and testing sets

train

✓ 0.0s

call_date	call_volume
2015-01-01	58
2015-01-02	60
2015-01-03	58
2015-01-04	54
2015-01-05	56
...	...
2016-07-24	25
2016-07-25	76
2016-07-26	69
2016-07-27	71
2016-07-28	59

575 rows × 1 columns

test

✓ 0.0s

call_date	call_volume
2016-07-29	38
2016-07-30	58
2016-07-31	54
2016-08-01	48
2016-08-02	68
...	...
2016-12-15	68
2016-12-16	71
2016-12-17	76
2016-12-18	81
2016-12-19	101

144 rows × 1 columns

8. Now that we know that the data is stationary, we can split it into the training and testing sets. We will split the dataset into 80% for the training set and 20% for testing.

Justification of the tools and techniques

We are running our analysis in a Jupyter Notebook. We chose Python as it contains all the necessary libraries for our data preparation from changing data types to grouping daily calls.

We performed the ADFuller test to check stationarity in the time series since it is the most widely used approach for this purpose (Jia, 2022).

Advantage of the tools and techniques

An advantage of choosing Python for our time series analysis is that we are using the statsmodels library for the most part of the analysis, i.e., the seasonal decompose analysis, ACF, PACF and the ARIMA model per se.

Disadvantage of the tools and techniques

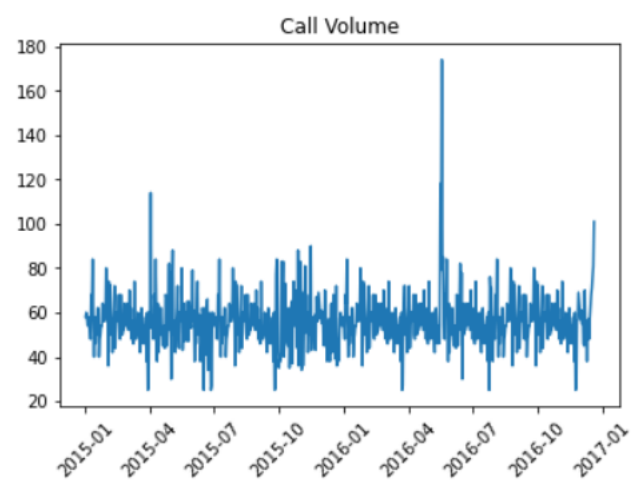
One disadvantage of using Python for forecasting is that running the actual arima model can take a couple of minutes to process. If we use a larger dataset in the future, then the processing time would increase.

D. ANALYSIS

We start our analysis by plotting the call volume to visualize the data.

Figure 9

Daily call volume



From the plot above, we can see some seasonality happening every 90 days approximately. We use this time ped as part of the parameters needed for the seasonal_decompose function.

We apply the seasonal_decompose function to split the time series into three components: trend, seasonality, and residuals (Lewinson, 2022). The period of the call_volume variable is set to 90.

Figure 10

Applying seasonal decompose function to call_volume

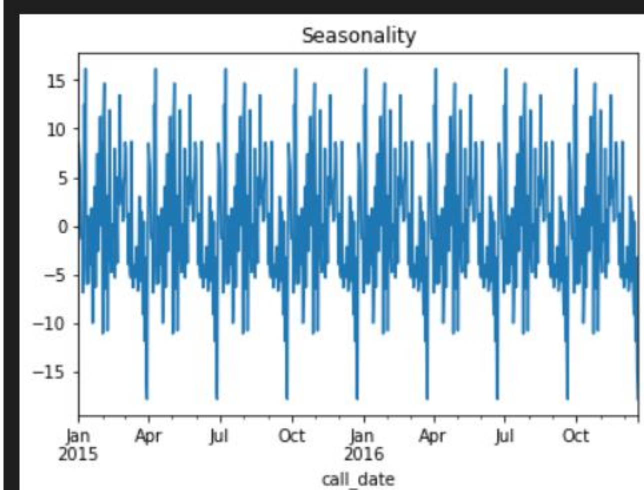
```
decomp=seasonal_decompose(df['call_volume'], period=90)
✓ 0.0s
```

Seasonality

Figure 11

Seasonality plot for the call_volume

```
plt.title('Seasonality')
decomp.seasonal.plot()
✓ 0.4s
<AxesSubplot:title={'center':'Seasonality'}, xlabel='call_date'>
```

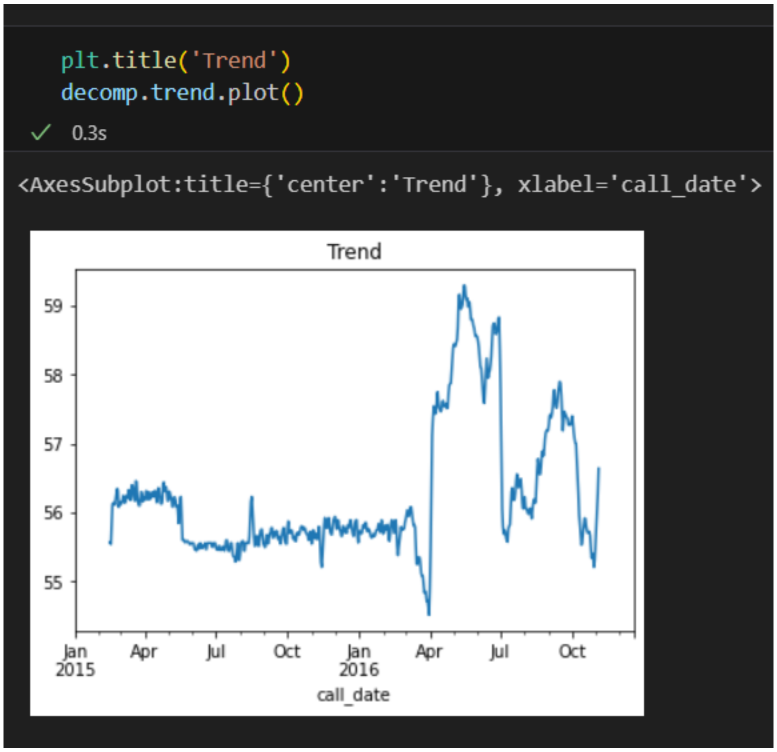


We can clearly see the seasonality in the call_volume every 90 days as we expected when visualizing the plot of the actual call volume in Figure 9.

Trend

Figure 12

Trend plot for the call_volume

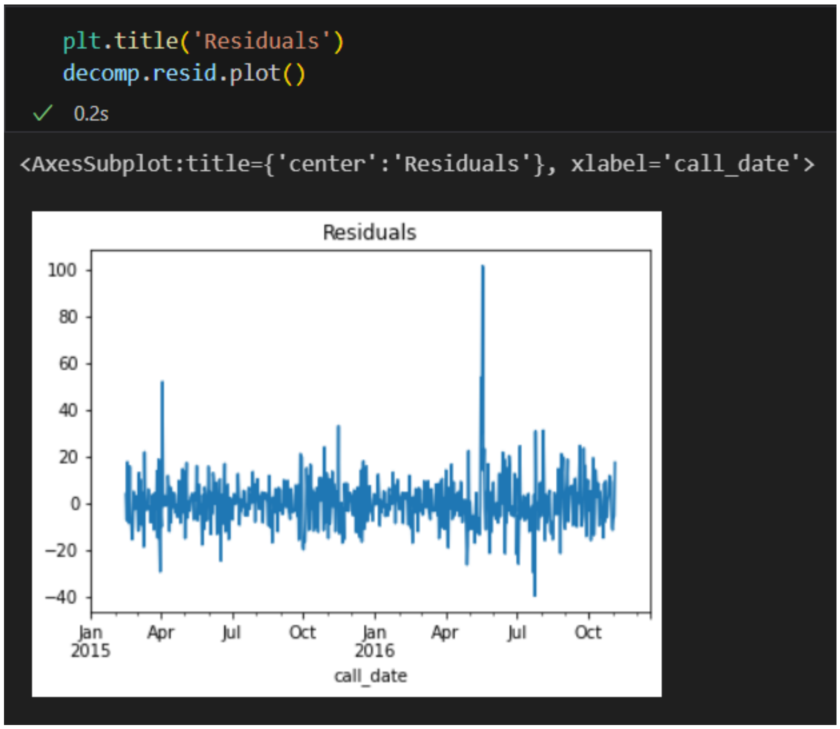


There is no evident trend in the call_volume. The data points do not show much variance in the first year and 3 months, but after that we see some visible variability with no apparent pattern since it goes up and down continuously.

Residuals

Figure 13

Residuals plot for the call_volume



The residuals analysis helps us recognize if our data is biased. If the residuals are mostly distributed around 0 it means that that they have zero or no correlation between them. In this case, the residuals are around 0 except for a few outliers. As a result, we can imply that our forecast model will be quite accurate (Hyndman).

Autocorrelation Function

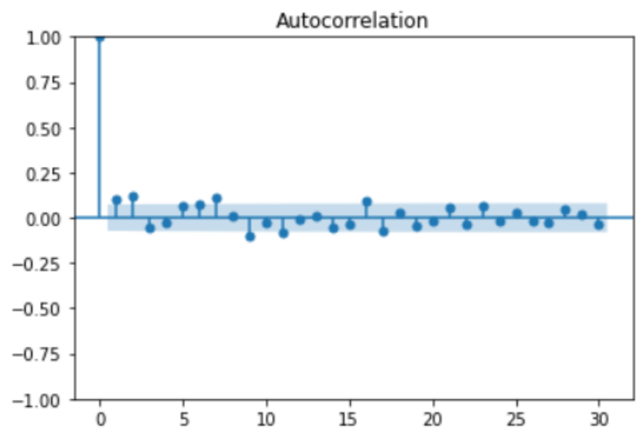
Autocorrelation is the correlation between two values in the time series (Frost). When a correlation exists, this may indicate that the current value is in part determined by previous values.

The autocorrelation function (ACF) helps us understand if there are any correlations in the time series and it is also used to build the prediction model. The ACF will determine the number of MA (q) terms in the model (Udit, 2022).

The ACF for our time series is shown below:

Figure 14

ACF for call_volume



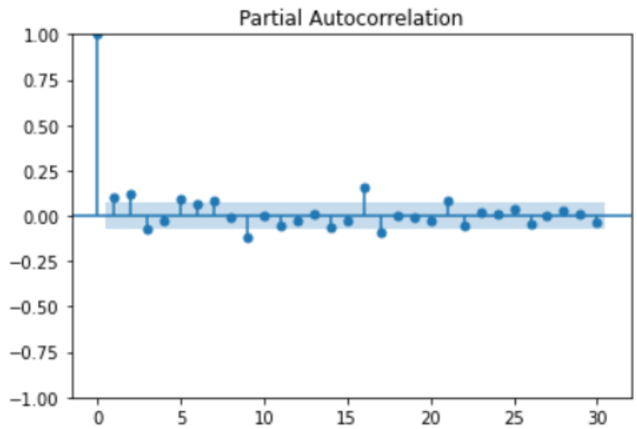
From the plot we can infer the following:
-There is only one lag statistically significant
-There are several autocorrelations that are not zero. Therefore, the time series is non-random.
-We can confirm that our time series is stationary since we see the lags quickly decreasing to 0
-The order of the model for MA, i.e., the value of q is 1.

Partial Autocorrelation Function

The partial autocorrelation function shows the correlation between a time series and its lagged values that the shorter lags between those values do not explain (Frost). It is used to identify the order of AR part of the model (p) (Udit, 2022).

Figure 15

PACF for call_volume



From the PACF plot above we can mention the following insights:

-There is only one strong correlation at lag 0.
-A sharp cut off may indicate the presence of some seasonality in the data. When we plotted Seasonality, we saw some seasonal pattern repeating every 90 days (Udit, 2022).
-Since there is only one statistically significant lag in the time series, the value of p that we will be using is 1.

Since both ACF and PACF plots decrease slowly to 0 in an exponential way, we can use an ARIMA model with the order (1,0,1). Considering that the time series is stationary, and we did not have to apply a differencing method, the value for d=0.

Spectral density

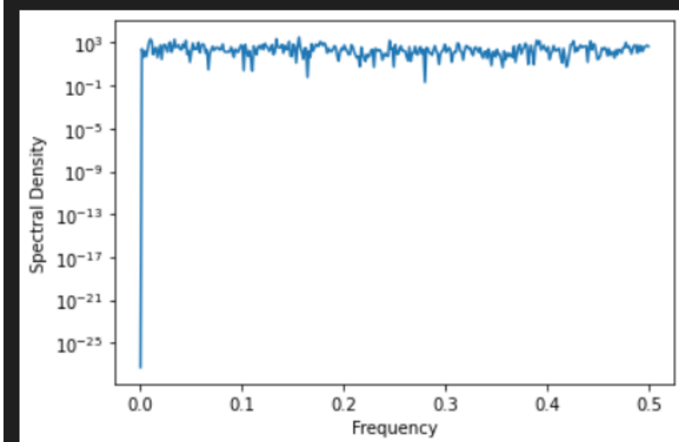
The spectral density plot or periodogram is used to estimate the frequencies of strict periodicities in a time series (Parzen, 1967).

Figure 16

Periodogram for call_volume

```
f, Pxx_den = signal.periodogram(df['call_volume'])
plt.semilogy(f, Pxx_den)
plt.xlabel('Frequency')
plt.ylabel('Spectral Density')
plt.show()
```

✓ 1.4s



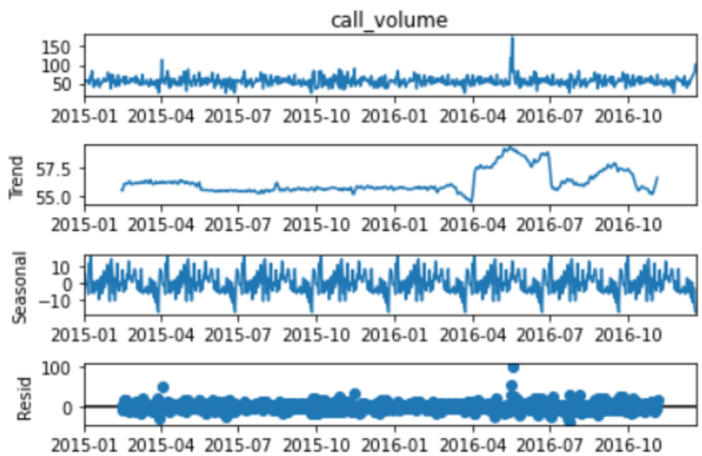
The spectral density plot shown above explains the presence of periodicity in the time series, hence the frequency is random.

Decomposed time series

The decomposed time series can be seen in the image below.

Figure 17

Decomposed time series plot



ARIMA model

The ARIMA model is a model that can be used to predict future values based on past values. It has 3 parameters, p, q, and d, which respectively determine the order of the autoregressive (AR), moving average (MA), and differencing steps (Zvornicanin, 2023)).

For our model, we use the values 1,0,1 for p, d, and q as mentioned before in the analysis. Considering that the time series presents seasonality, we will include the seasonality parameter in the model with 90 as our seasonality period.

Figure 17

ARIMA model code and output

```
model = ARIMA(train, order=(1,0,1), seasonal_order=(1,0,1,90))
results = model.fit()
results.summary()
```

✓ 35.1s

```
c:\Users\paowm\AppData\Local\Programs\Python\Python310\lib\site-pack
  self._init_dates(dates, freq)
c:\Users\paowm\AppData\Local\Programs\Python\Python310\lib\site-pack
  self._init_dates(dates, freq)
c:\Users\paowm\AppData\Local\Programs\Python\Python310\lib\site-pack
  self._init_dates(dates, freq)
```

SARIMAX Results

Dep. Variable:	call_volume	No. Observations:	575
Model:	ARIMA(1, 0, 1)x(1, 0, 1, 90)	Log Likelihood	-2254.072
Date:	Tue, 27 Jun 2023	AIC	4520.143
Time:	22:01:18	BIC	4546.269
Sample:	01-01-2015	HQIC	4530.333
	- 07-28-2016		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	56.3086	0.969	58.102	0.000	54.409	58.208
ar.L1	0.6443	0.302	2.132	0.033	0.052	1.237
ma.L1	-0.5865	0.322	-1.824	0.068	-1.217	0.044
ar.S.L90	0.5314	0.132	4.026	0.000	0.273	0.790
ma.S.L90	-0.2485	0.160	-1.552	0.121	-0.562	0.065
sigma2	146.1713	4.660	31.369	0.000	137.038	155.304

Ljung-Box (L1) (Q):	0.20	Jarque-Bera (JB):	6328.60
Prob(Q):	0.65	Prob(JB):	0.00
Heteroskedasticity (H):	1.37	Skew:	1.99
Prob(H) (two-sided):	0.03	Kurtosis:	18.76

We want to find the best model by applying the auto_arima function. We then compare the AIC scores with this model. AIC is a measure to compare accuracy between different regression models (Zach, 2021). The model with the lowest AIC is considered the one with the best fit. This would be the model we would use to predict future values.

Figure 18

Applying Auto ARIMA function

```
model2 = auto_arima(train, trace = True, error_action ='ignore', suppress_warnings = True)
model2.fit(train)
model2.summary()
```

✓ 6.5s

```
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(0,0,0)[0] intercept   : AIC=4558.381, Time=2.34 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=4568.186, Time=0.06 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=4565.883, Time=0.10 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=4566.758, Time=0.23 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=6296.816, Time=0.04 sec
 ARIMA(1,0,2)(0,0,0)[0] intercept   : AIC=4557.632, Time=0.91 sec
 ARIMA(0,0,2)(0,0,0)[0] intercept   : AIC=4556.501, Time=0.32 sec
 ARIMA(0,0,3)(0,0,0)[0] intercept   : AIC=4557.030, Time=0.31 sec
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=4564.759, Time=0.66 sec
 ARIMA(1,0,3)(0,0,0)[0] intercept   : AIC=4558.673, Time=0.25 sec
 ARIMA(0,0,2)(0,0,0)[0]             : AIC=5486.964, Time=0.26 sec

Best model:  ARIMA(0,0,2)(0,0,0)[0] intercept
Total fit time: 5.861 seconds
```

SARIMAX Results

Dep. Variable:	y	No. Observations:	575
Model:	SARIMAX(0, 0, 2)	Log Likelihood	-2274.250
Date:	Tue, 27 Jun 2023	AIC	4556.501
Time:	22:01:24	BIC	4573.918
Sample:	01-01-2015	HQIC	4563.294
	- 07-28-2016		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	56.2389	0.727	77.390	0.000	54.815	57.663
ma.L1	0.0931	0.037	2.533	0.011	0.021	0.165
ma.L2	0.1573	0.029	5.418	0.000	0.100	0.214
sigma2	159.5668	6.321	25.245	0.000	147.178	171.955

Ljung-Box (L1) (Q):	0.04	Jarque-Bera (JB):	2236.74
Prob(Q):	0.85	Prob(JB):	0.00
Heteroskedasticity (H):	1.27	Skew:	1.40
Prob(H) (two-sided):	0.10	Kurtosis:	12.25

The best model resulted from the auto_arima has an AIC greater than the model we used at first, so we will use the first model.

Forecasting

We create a forecast variable by using the function get_prediction to the results generated from the ARIMA model. The start of the forecast would be the same as the test set, thus we can compare the forecasted values with the testing set.

Consecutively, we convert the variable forecast_mean to a dataframe to plot the forecast values along with the testing set values for a visual comparison.

Figure 19

Forecasting call_volume values

```
temp_forecast= pd.DataFrame(forecast_mean)
temp_forecast.rename(columns={'predicted_mean' : 'call_volume'}, inplace=True)
temp_forecast
```

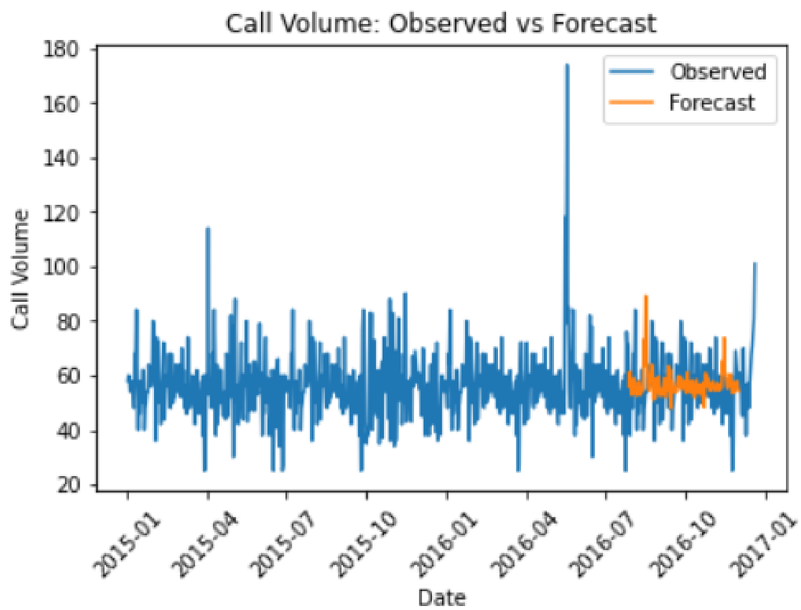
✓ 0.1s

	call_volume
2016-07-28	56.101549
2016-07-29	61.283448
2016-07-30	55.258011
2016-07-31	52.866109
2016-08-01	54.592033
...	...
2016-12-10	60.056791
2016-12-11	55.351484
2016-12-12	59.381692
2016-12-13	51.864124
2016-12-14	57.197969

140 rows × 1 columns

Figure 20

Call_volume observed vs forecasted values



RMSE

The root mean squared error measures the average difference between predicted values and observed ones (Frost, Root Mean Square Error (RMSE)). It is an absolute error measure which squares the deviations to avoid the cancelling of the positive deviations with the negative ones.

The closer the RMSE is to 0, the closer the forecasted values are to the actual values.

The code used to calculate the RMSE is shown below:

Figure 21

RMSE calculation for our model

```
rmse = mean_squared_error(temp_forecast, temp_forecast.iloc[-127:], squared=False)
print(f"The root mean squared error of this forecasting model is {round(rmse, 5)}")
✓ 0.1s
The root mean squared error of this forecasting model is 0.0
```

The value of RMSE in our model is 0 which indicated the difference between our predicted values and the actual values, i.e., the testing dataset, is 0 points. When comparing the scale of the variable call_volume and the magnitude of the RMSE, we can conclude our model is effective at predicting values.

To forecast call_volume for the next 90 days we use the following code:

Figure 22

RMSE calculation for our model

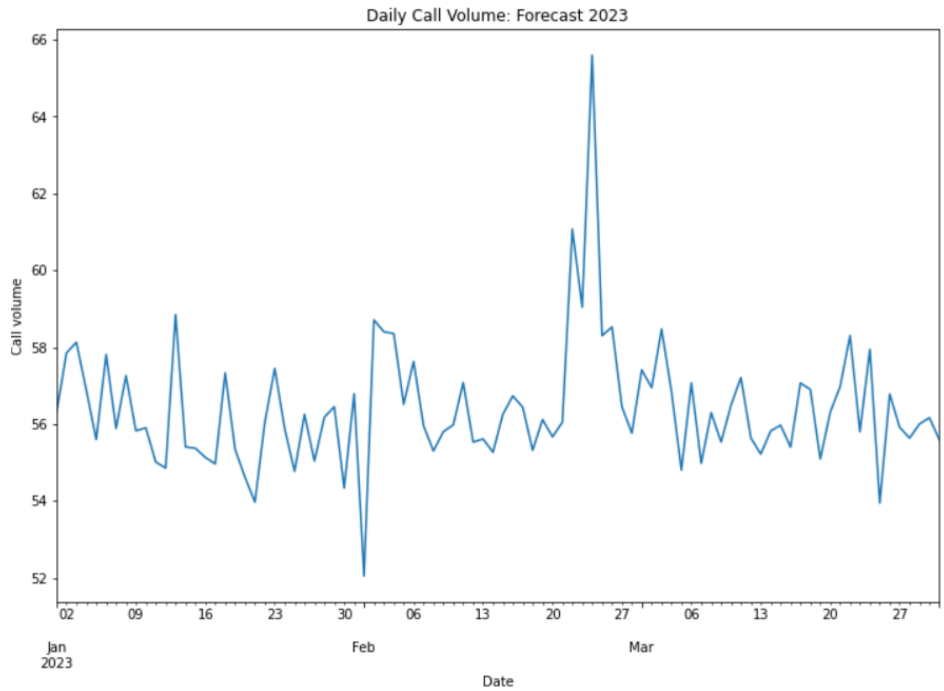
```
index_future = pd.date_range(start='2023-01-01', end='2023-03-31')
print(index_future)
pred = results.predict(start=len(df), end=len(df)+89, type='levels')
pred.index=index_future
print(pred)

pred.plot(figsize=(12,8), xlabel='Date',ylabel='Call volume', title='Daily Call Volume: Forecast 2023')
```

Our forecasted values range between 53 and 65 days with a mean close to 56 days. We can visualize the forecasted values in the following image:

Figure 23

Call_volume forecasted values for 90 days



Justification of the ARIMA model

We selected an ARIMA model to make our predictions based on our findings when analyzing the ACF and PACF. Both plots show lags that are slowly decreasing to 0 in an exponential way (Christiansen, 2018). This behavior is indicative of an ARIMA model. Since there is seasonality present, we will use the corresponding seasonality parameters to ensure our model fits the data in the most accurate way.

Advantage of the ARIMA model

An advantage of the ARIMA model is that is good for short term forecasting hence it works for this time series analysis since we are forecasting 90 days (Hayes, Autoregressive Integrated Moving Average (ARIMA) Prediction Model, 2022).

Disadvantage of the ARIMA model

As a disadvantage, we can mention that the ARIMA model works to predict future values based on actual values. Considering that ARIMA models are a form of regression, it assumes that the current values are somewhat correlated to past values. Consequently, we would not be able to use this model when data points are not correlated with past values.

E. DATA SUMMARY AND IMPLICATIONS

We have been able to predict future daily call volume with the ARIMA model. Our model has an RSME of 0 thus our model is accurate at predicting correct values.

Our research question mentions that we can predict with 95% accuracy. The reason is that when using RSME, we are assuming that the residuals follow a normal distribution which means that about 95% of the observed values fall between +/- 2 x RSME from the predicted values. This implies that 95% of the actual values are in the range of +/- 2 x 0= 0 points of the predicted values (Frost, Root Mean Square Error (RMSE)).

Limitation of the analysis

A limitation of our analysis is that the Customer Support department not only answer calls, but also works on Customer Support cases, which is a separate task. This insight is helpful to know since agents could be working on cases and not taking any calls while doing so. In this case, to build a more accurate headcount and capacity plan, we would need an analysis of the support cases aside from the call volume.

Course of action

Now that we have forecasted the daily call volume for the next 90 days at a 95% confidence level, the company can make sure to add the call volume to the capacity and headcount analysis to decide whether to hire new agents or not to. They can compare performance levels in the past with performance levels they want to achieve and analyze what the needs are of the Customer Support department.

Approaches to further studies

For future studies, the Customer Department can breakdown the call volume data by Tiers. This division can help understand better how headcount should be assigned in terms of how many calls are received by Tier. Those Tiers than get the most calls should have more agents designated to answer calls and the other way around. Considering this new data collection process, we can further forecast call volume by Tier.

Another approach could be to forecast call volume and Customer Support cases, so these can be added to the headcount plan. In this case, we would have multivariable time series. For this kind of analysis, we could use a KATS model and the Vector AutoRegression (VAR) principle (Afzal, 2022).

F. SOURCES

Afzal, A. (2022, August 4). *Analytics Vidhya.* Retrieved from Multi-variate Time Series Forecasting using Kats Model: https://www.analyticsvidhya.com/blog/2022/08/multi-variate-timeseries-forecasting-using-kats-model/

Christiansen, L. E. (2018). Retrieved from https://www.youtube.com/watch?v=ZE_WGBe0_VU

Frost, J. (n.d.). *Root Mean Square Error (RMSE).* Retrieved from Statistics by Jim: https://statisticsbyjim.com/regression/root-mean-square-error-rmse/#:~:text=What%20is%20the%20Root%20Mean,line%20and%20the%20data%20points.

Hayes, A. (2022, December 18). *Autoregressive Integrated Moving Average (ARIMA) Prediction Model.* Retrieved from https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp

Lewinson, E. (2022, March 31). Retrieved from Towards Data Science: https://towardsdatascience.com/time-series-diy-seasonal-decomposition-f0b469afed44

Pandey, A. (2020). *Call center dataset.* Retrieved from https://www.kaggle.com/datasets/ashishpandey5210/call-center-dataset

Parzen, E. (1967). *The Role of Spectral Analysis in Time Series Analysis.* Retrieved from https://www.jstor.org/stable/1401395

Shetty, C. (2020, September 22). *Time Series Models.* Retrieved from Medium: https://towardsdatascience.com/time-series-models-d9266f8ac7b0

Udit. (2022, December 30). Retrieved from Towards Data Science: https://itsudit.medium.com/deciphering-acf-and-pacf-plots-a-guide-to-time-series-forecasting-3323948935fb

Zach. (2021). Retrieved from https://www.statology.org/what-is-a-good-aic-value/

Zvornicanin, E. (2023, May 15). *Choosing the best q and p from ACF and PACF plots in ARMA-type modeling.* Retrieved from https://www.baeldung.com/cs/acf-pacf-plots-arma-modeling