Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Report on Mini Project

Time Series Analysis (DJ19DSC5012)
AY: 2021-22

Analysis and Forecasting on Household Electricity Consumption

Jainam Rajput (60009200022)
Jay Jain (60009200031)
Aryan Mehta (60009200013)

Guided By
Prof. Pradnya Saval

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

- We have used Daily Electricity City Consumption Data.
- The data is collected with smart meters and shared by the energy company.
- This is time-series data by nature from 2016 to 2018.
- We have performed Time Series Analysis using various methods on this dataset.
- After analysis of the dataset, we have forecasted the electricity usage for future time periods.

CHAPTER 2. DATA DESCRIPTION

The dataset is basically electricity usage by households.

It has different timestamps where we get to check what was the electricity consumption for a time period from start time to end time and what is the cost of consumption.

The data initially contains eight attributes.

- TYPE - This is an information column. The value is 'Electric usage' for all the observations.
- DATE - Date of electric consumption. There is no timestamp in this field.
- START TIME - Start time of the consumption.
- END TIME - End time of the consumption
- USAGE - Consumption in kWh
- UNITS - This column denotes measurement unit. It is kWh for all the observations.
- COST - Cost of consumption in $.
- NOTES - Mostly an empty column
- After data preprocessing and data cleaning we format the dataset into 3 columns:
- Date, Usage (Electricity Consumption), and Cost(Cost of Consumption)

Link to the Dataset:

https://drive.google.com/drive/folders/1Wk0srhJc6cC6u6cUcUR_lLEDJNu_vzq1?usp=share_link

CHAPTER 3. OBJECTIVE

- With the increasing population and our strong dependency on non-renewable sources of energy directly points towards utilizing electrical energy in a disciplined manner.

- Hence forecasting these values based on past observations can help find a solution for the efficient management of electricity. Starting from the very bottom of the hierarchy we are trying to analyze the consumption of electricity in San Jose, California. We aim to forecast the total units of usage for future days.

CHAPTER 4. DATA CLEANING

Firstly, on checking for null values, we found a great number of null values in the NOTES column. Hence, we dropped the Notes column.

The dataset also consisted of a few irrelevant columns namely TYPE and UNITS, hence they were dropped too.

The statistical description of the Data

	USAGE	COST
count	70368.000000	70368.000000
mean	0.121941	0.024684
std	0.210507	0.042646
min	0.000000	0.000000
25%	0.030000	0.010000
50%	0.050000	0.010000
75%	0.120000	0.020000
max	2.360000	0.650000

The time interval in the dataset was 15 mins. So we had 96 values of usage and cost for a single day. The values of usage and cost were minimal and would produce indistinguishable results. Hence to normalize and enlarge the data we resample it to daily data thus finally having a total of 733 columns from 22-10-2016 to 24-10-2018.
To do this we take the aggregate of the data and sum it on day-to-day intervals.

Link for the colab notebook:
https://colab.research.google.com/drive/1VdvJGfStT_McpCveSNWsI_zws7cUoUzQ?usp=sharing - Exp3

CHAPTER 5. DATA DECOMPOSITION

The various components of a Time Series:

- Trend
- Seasonal Variation
- Cyclic Variation
- Irregular or Residual variation

But we can apply various models to our TS dataset only if we get rid of these components. Thus, we detrend and deseasonalize our dataset.

For detrending we can use one of the following methods:

- Using Pandas Differencing
- Using Signal Detrending
- Using HP-filter

We check if our data has a seasonal variation by using autocorrelation using "autocorrelation_plot" from "pandas.plotting" and Statsmodels' seasonal decompose method neatly breaks down time series data into components.

Link for the colab notebook:
https://colab.research.google.com/drive/16WwRaiTZlBs9V8hLGNa3L2F8UDF8zhHW?usp=sharing - Exp1
https://colab.research.google.com/drive/1eQ0KQet26p3bIqP3PqnxJeEroKQiyJ1e?usp=sharing - Exp2

CHAPTER 6. SMOOTHING METHODS

To smoothen time-series data, various smoothing methods are available namely, Exponential smoothing and Holt Winter models.

Simple Exponential Smoothing

- This type of smoothing model is similar to the weighted moving average model. It takes into account past observations as well as past forecasts by assigning weights (α and 1-α respectively) to them.

$$F_t = \alpha * V_{t-1} + (1- \alpha) * F_{t-1}$$

- **For Simple Exponential Smoothing**, we found best results with weight (α) = 0.32

```
Fitting for smoothing level= 0.32
Evaluation metric results:-
MSE is : 8.00902129433436
MAE is : 2.108051810157824
RMSE is : 2.830021430013271
MAPE is : 0.9348250339832478
R2 is : -0.00011843117047383167
```

For Double Holt Winter Smoothing, we found best results with weight (α) = 0.6

smoothing_level	smoothing_slope	damping_slope	damped	MAPE	r2
0.6	0.5	0.1	False	0.488492	-1.041081

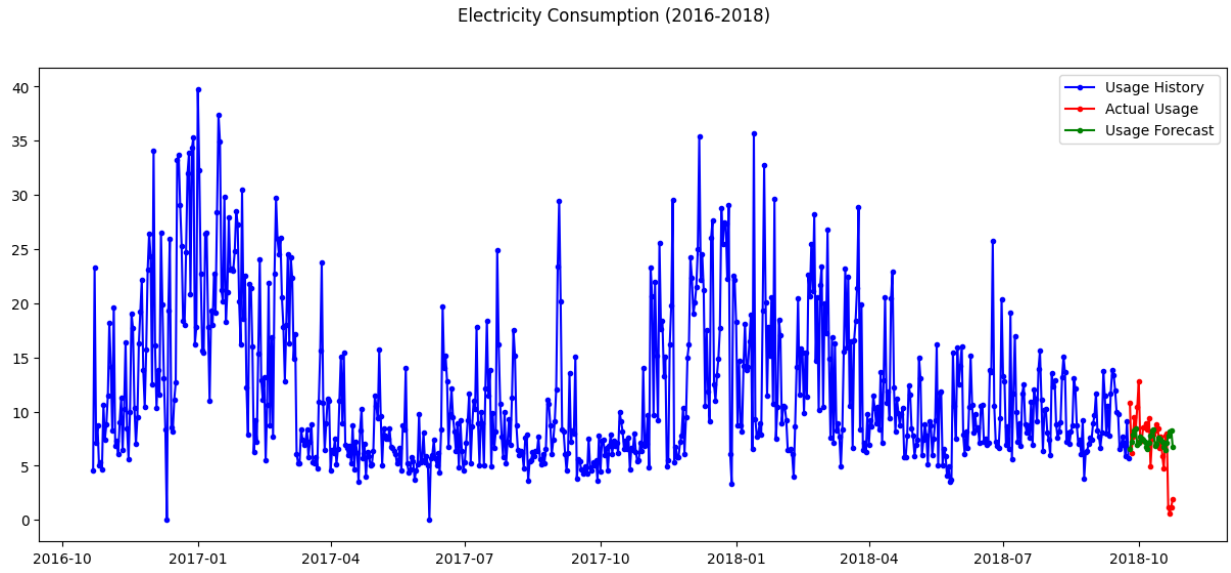
```
Evaluation metric results:-
MSE is : 7.8884953027777085
MAE is : 2.0684671499669975
RMSE is : 2.808646525068206
MAPE is : 0.9326737146521267
R2 is : 0.014932130085515749
```

For Triple Holt Winter Smoothing, we found the best match

ExponentialSmoothing Model Results

Dep. Variable:	usage	No. Observations:	703
Model:	ExponentialSmoothing	SSE	20914.417
Optimized:	True	AIC	2417.165
Trend:	Additive	BIC	2490.050
Seasonal:	Additive	AICC	2418.165
Seasonal Periods:	12	Date:	Mon, 05 Dec 2022
Box-Cox:	False	Time:	18:18:27
Box-Cox Coeff.:	None		

	coeff	code	optimized
smoothing_level	0.2876491	alpha	True
smoothing_trend	1.3333e-06	beta	True
smoothing_seasonal	6.7031e-05	gamma	True
initial_level	10.191934	l.0	True
initial_trend	-0.0047096	b.0	True
initial_seasons.0	-0.4804406	s.0	True
initial_seasons.1	-0.2820384	s.1	True
initial_seasons.2	0.3197619	s.2	True
initial_seasons.3	0.1745477	s.3	True
initial_seasons.4	0.0862642	s.4	True
initial_seasons.5	-0.1117923	s.5	True
initial_seasons.6	-0.5807770	s.6	True
initial_seasons.7	-0.8207643	s.7	True
initial_seasons.8	-0.1701665	s.8	True
initial_seasons.9	0.4891154	s.9	True
initial_seasons.10	0.9002016	s.10	True
initial_seasons.11	1.0374836	s.11	True



Link for the colab notebook is:
https://colab.research.google.com/drive/1XEjYVsqBQEIQr8q1CfMD2O97z4MRO6cT?usp=sharing - Exp4

CHAPTER 7. TESTING STATIONARY

- Stationary time series data do not depend on time.
- We can check stationary by either manually checking the mean and variance of time series or by using the Augmented Dicky Fuller Test function.
- The ADF test is fundamentally a statistical significance test. There is hypothesis testing involved with a null and alternate hypothesis and as a result, a test statistic is computed and p-values get reported. From the statistic test and the p-values, we can make an inference as to whether a given time series is stationary or not.
- Unit root is a characteristic of a time series that makes it non-stationary. The presence of a unit root means the time series is non-stationary.

On applying ADF Test,
The data was found to be non-stationary because of evident seasonality in the data. This is evident as we fail to reject the null hypothesis.

```
Results of Dickey-Fuller Test for column: usage
Test_Stats                -2.423278
P_Value                    0.135274
No_Of_Observation_Used    19.000000
Critical_Value           713.000000
Critical Value (1%)       -3.439555
Critical Value (5%)       -2.865602
Critical Value (10%)      -2.568933
dtype: float64
Conclusion:====>
Fail to reject the null hypothesis
Data is non-stationary
```

On differencing the data for order 1,

New data was found to be stationary since it rejects the null hypothesis

```
Results of Dickey-Fuller Test for column: usage
Test_Stats                -1.037090e+01
P_Value                    2.265619e-18
No_Of_Observation_Used     1.800000e+01
Critical_Value             7.130000e+02
Critical Value (1%)       -3.439555e+00
Critical Value (5%)       -2.865602e+00
Critical Value (10%)      -2.568933e+00
dtype: float64
Conclusion:====>
Reject the null hypothesis
Data is stationary
```

Link for the Colab Notebook:
https://colab.research.google.com/drive/1FmkwM0JD_XQLeQdcaPHuvSGITchFGfdD?usp=sharing

CHAPTER 8. JUSTIFICATION WHY IT IS A TIME SERIES PROBLEM.

- A time series is a set of numerical values of some variable obtained at a regular period over time(Yt).
- The series is usually tabulated or graphed in a manner that readily conveys the behavior of the variable under study.
- A time series can also be defined as a series of data points indexed in time order(chronologically arranged data).
- In the given dataset, Energy usage by households is indexed by date, thus making it a time series problem.

CHAPTER 9. IMPLEMENTATION AND INTERPRETATION FOR FORECAST

After Analyzing the time series data, the next step is to fit a suitable model for forecasting the time series data.
We have applied AR, MA, ARIMA, VAR, Convolutional Network, and GARCH models over our dataset to check which model provides the best result for our time series data values.

AR model:

- AR Stands for AutoRegressive Model.
- An Auto-regressive Model is when a value from the time series is regressed from the previous value of the time series
- The equation of an AR model is, $X_t = C + \phi_1 X_{t-1} + \epsilon_t$

MA model:

- MA stands for Moving Average Model.
- A moving average term in time series is a model of past errors multiplied by a coefficient.

- The equation of a MA model is, $r_t = c + \theta_1 \epsilon_{t-1} + \epsilon_t$

Both the AR model and the MA model is applied to stationary time series.
For non-stationary time series data, we need to first transform the data using differencing.
For non-stationary time series data, we should apply the ARIMA Model

ARIMA model

- ARIMA stands for Auto-Regressive Integrated Moving Average Model.
- The ARIMA model is a simple ARMA model where the objective is to predict the future time series by examining the differences between values in the series instead of actual values.

SARIMA model:

- SARIMA stands for Seasonal Auto-Regressive Integrated Moving Average Model.
- This model is used when there is seasonality in our time series data as AR, MA, ARMA, and ARIMA model does not give the best results for seasonal time series data.

Using the ADF test we have found that our data is non-stationary and on Data Decomposition we have observed trends and seasonality in our data.
So we apply ARIMA and SARIMA models over our time series data
We find these results

```
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=4350.202, Time=0.92 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=4562.925, Time=0.09 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=4482.643, Time=0.23 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=4390.022, Time=0.36 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=4560.925, Time=0.04 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=4352.116, Time=0.45 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=4352.112, Time=0.56 sec
 ARIMA(0,1,2)(0,0,0)[0] intercept   : AIC=4355.497, Time=0.26 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=4442.736, Time=0.28 sec
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=4353.682, Time=0.79 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=4348.208, Time=0.13 sec
 ARIMA(0,1,1)(0,0,0)[0]             : AIC=4388.025, Time=0.10 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=4480.644, Time=0.07 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=4350.122, Time=0.21 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=4350.119, Time=0.24 sec
 ARIMA(0,1,2)(0,0,0)[0]             : AIC=4353.499, Time=0.24 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=4440.737, Time=0.10 sec
 ARIMA(2,1,2)(0,0,0)[0]             : AIC=4351.687, Time=0.33 sec

Best model:  ARIMA(1,1,1)(0,0,0)[0]
Total fit time: 5.465 seconds
```

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	703
Model:	SARIMAX(1, 1, 1)	**Log Likelihood**	-2171.104
Date:	Tue, 15 Nov 2022	**AIC**	4348.208
Time:	07:46:58	**BIC**	4361.870
Sample:	0	**HQIC**	4353.488
	- 703		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.3371	0.032	10.383	0.000	0.273	0.401
ma.L1	-0.9048	0.016	-57.909	0.000	-0.935	-0.874
sigma2	28.3899	1.138	24.940	0.000	26.159	30.621

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	136.22
Prob(Q):	0.97	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.58	**Skew:**	0.66
Prob(H) (two-sided):	0.00	**Kurtosis:**	4.71

Colab Link for the SARIMA model:
https://colab.research.google.com/drive/1z2R9ZCuMKUNDVlNYAvyqhEXu8a4L-09g?usp=sharing

VAR Model:

- VAR stands for Vector Auto-Regression Model.
- VAR is a multivariate forecasting algorithm that is used when two or more variables are dependent on or influence each other.
- For applying the VAR model basic requirement is the presence of two variables in the data

- Our data has two Variables ‘Usage’ and ‘Cost’ and from the **cointegration test** we find out that the **two variables are correlated with each other**

```
Column Name   >  Test Stat > C(95%)    =>   Signif
 ----------------------------------------
usage  >  487.15    > 12.3212   =>   True
cost   >  228.17    > 4.1296    =>   True
```

Summarization of the model is:

```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Sat, 03, Dec, 2022
Time:                     18:04:08
--------------------------------------------------------------------
No. of Equations:         2.00000    BIC:                    0.437332
Nobs:                     696.000    HQIC:                   0.333188
Log likelihood:          -2042.26    FPE:                     1.30675
AIC:                     0.267534    Det(Omega_mle):          1.25927
--------------------------------------------------------------------
Results for equation usage
============================================================================
               coefficient       std. error           t-stat            prob
----------------------------------------------------------------------------
const            0.002646         0.202459            0.013           0.990
L1.usage        -0.231001         0.196747           -1.174           0.240
L1.cost         -1.536596         0.964354           -1.593           0.111
L2.usage        -0.273863         0.205992           -1.329           0.184
L2.cost         -0.894431         1.009690           -0.886           0.376
L3.usage        -0.356968         0.206687           -1.727           0.084
L3.cost         -0.031030         1.018820           -0.030           0.976
L4.usage        -0.572925         0.206361           -2.776           0.005
L4.cost          1.121088         1.017441            1.102           0.271
L5.usage         0.004069         0.206070            0.020           0.984
L5.cost         -1.052950         1.009856           -1.043           0.297
L6.usage         0.190723         0.196534            0.970           0.332
L6.cost         -1.562432         0.965107           -1.619           0.105
============================================================================

Results for equation cost
============================================================================
               coefficient       std. error           t-stat            prob
----------------------------------------------------------------------------
const            0.001626         0.041255            0.039           0.969
L1.usage         0.023123         0.040091            0.577           0.564
L1.cost         -0.645409         0.196505           -3.284           0.001
L2.usage        -0.008208         0.041975           -0.196           0.845
L2.cost         -0.412342         0.205743           -2.004           0.045
L3.usage        -0.008362         0.042116           -0.199           0.843
L3.cost         -0.301878         0.207604           -1.454           0.146
L4.usage        -0.075887         0.042050           -1.805           0.071
L4.cost          0.041649         0.207323            0.201           0.841
L5.usage         0.031154         0.041991            0.742           0.458
L5.cost         -0.358727         0.205777           -1.743           0.081
L6.usage         0.065535         0.040047            1.636           0.102
L6.cost         -0.447663         0.196659           -2.276           0.023
============================================================================

Correlation matrix of residuals
            usage      cost
usage    1.000000  0.981187
cost     0.981187  1.000000
```

Colab Link for the VAR Model:

https://colab.research.google.com/drive/1Tl0z8FpMIrfJGkR-R1gfl8wKLWRKi-0i?usp=sharing

Convolutional Neural Network:

- 1-D Convolution can be applied to time series data.
- Imagine a time series of length *n* and width *k*. The length is the number of timesteps, and the width is the number of variables in a multivariate time series.
- The convolution kernels always have the same width as the time series, while their length can be varied. This way, the kernel moves in one direction from the beginning of a time series toward its end, performing convolution.

On Applying Convolutional Network to our data we get the model summary as

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 2, 64)             192

 max_pooling1d (MaxPooling1D  (None, 1, 64)            0
 )

 flatten (Flatten)           (None, 64)                0

 dense (Dense)               (None, 100)               6500

 dense_1 (Dense)             (None, 1)                 101

=================================================================
Total params: 6,793
Trainable params: 6,793
Non-trainable params: 0
_________________________________________________________________
```

Colab Link for Convolutional Model:
https://colab.research.google.com/drive/17VNGlKDPDVhMxX0bzAW2Ys2-Q3oyf2Wl?usp=sharing

GARCH Model:

- ➔ GARCH stands for Generalized AutoRegressive Conditional Heteroskedasticity.
- ➔ The GARCH process is an approach to estimating the volatility of data.
- ➔ GARCH aims to minimize errors and forecasting by accounting for errors in prior frequency and enhancing the accuracy of ongoing predictions.

On applying the GARCH model to our data, the model summarizes as

Constant Mean - GARCH Model Results

Dep. Variable:	usage	**R-squared:**	0.000
Mean Model:	Constant Mean	**Adj. R-squared:**	0.000
Vol Model:	GARCH	**Log-Likelihood:**	-2306.29
Distribution:	Normal	**AIC:**	4632.57
Method:	Maximum Likelihood	**BIC:**	4678.54
		No. Observations:	733
Date:	Mon, Dec 05 2022	**Df Residuals:**	732
Time:	06:50:29	**Df Model:**	1

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
mu	8.8064	0.590	14.919	2.491e-50	[7.649, 9.963]

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	1.9842	11.211	0.177	0.860	[-19.990, 23.958]
alpha[1]	0.4270	2.059	0.207	0.836	[-3.609, 4.463]
beta[1]	0.2162	3.407	6.346e-02	0.949	[-6.461, 6.893]
beta[2]	0.0000	3.194	0.000	1.000	[-6.259, 6.259]
beta[3]	0.0000	1.930	0.000	1.000	[-3.782, 3.782]
beta[4]	0.0651	2.809	2.320e-02	0.981	[-5.440, 5.570]
beta[5]	0.1463	0.891	0.164	0.870	[-1.601, 1.894]
beta[6]	0.1280	2.673	4.790e-02	0.962	[-5.110, 5.366]
beta[7]	0.0000	4.257	0.000	1.000	[-8.343, 8.343]

Colab Link for the GARCH Model:
https://colab.research.google.com/drive/1COALtlh27G329KywbyEHDW8Rc8f50Tvx?usp=sharing

CHAPTER 10. REASONS FOR SELECTING THE TIME SERIES MODEL

SARIMA Model is selected for the forecasting of future values as it shows the best results out of all the models we trained and validated the dataset in.

The order of the SARIMA Model is (1,1,1)(0,0,0)[0]

SARIMA Model justifies the results because

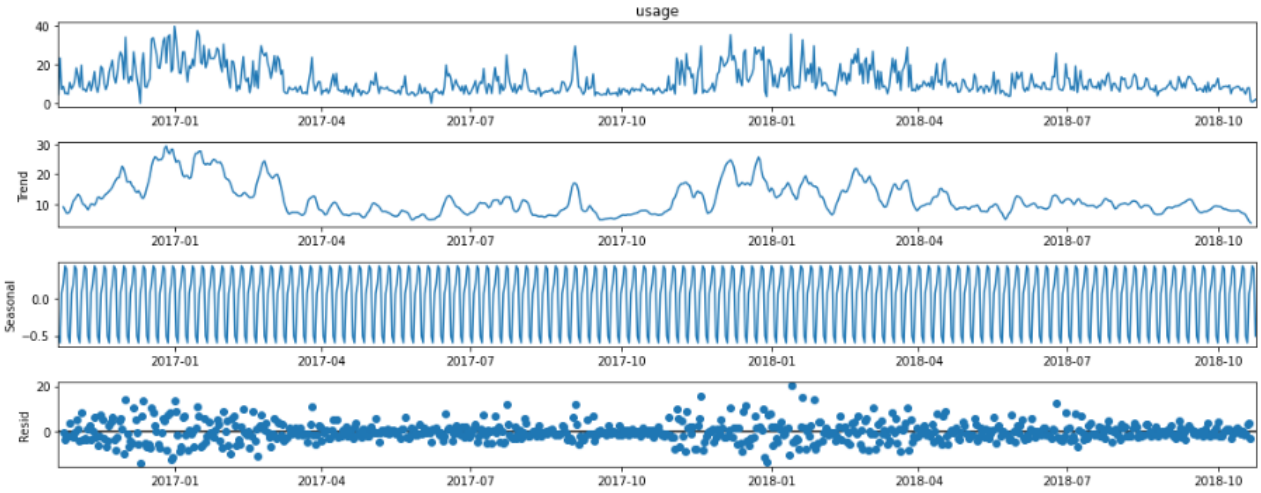
- The data has trends and seasonality that we can check from the plots and also from seasonal decomposition.
- On applying Augmented Dickey-Fuller to check for stationarity and the presence/absence of unit roots we find that data is not stationary.
- SARIMA Model can be applied to data having trends and seasonality as the S in the SARIMA model stands for Seasonality and I stands for Integration.

Integrated means instead of predicting the time series, the model predicts the differencing of time series from one to previous time stamps.

- Since Seasonality is present in the data other models do not give accurate results as SARIMA does.

These Plots show us the presence of Seasonality and Trends in the dataset

```
<ipython-input-9-972a3c490795>:2: FutureWarning: the 'freq'' keyword is deprecated, use 'period' instead
  result = seasonal_decompose(df['usage'], model='add',freq=6)
```



SARIMAX Results

Dep. Variable:	y	**No. Observations:**	703
Model:	SARIMAX(1, 1, 1)	**Log Likelihood**	-2171.104
Date:	Tue, 15 Nov 2022	**AIC**	4348.208
Time:	07:46:58	**BIC**	4361.870
Sample:	0	**HQIC**	4353.488
	- 703		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.3371	0.032	10.383	0.000	0.273	0.401
ma.L1	-0.9048	0.016	-57.909	0.000	-0.935	-0.874
sigma2	28.3899	1.138	24.940	0.000	26.159	30.621

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	136.22
Prob(Q):	0.97	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.58	**Skew:**	0.66
Prob(H) (two-sided):	0.00	**Kurtosis:**	4.71

CHAPTER 11. COMPARATIVE RESULT ANALYSIS

Results of each of the models on the dataset are

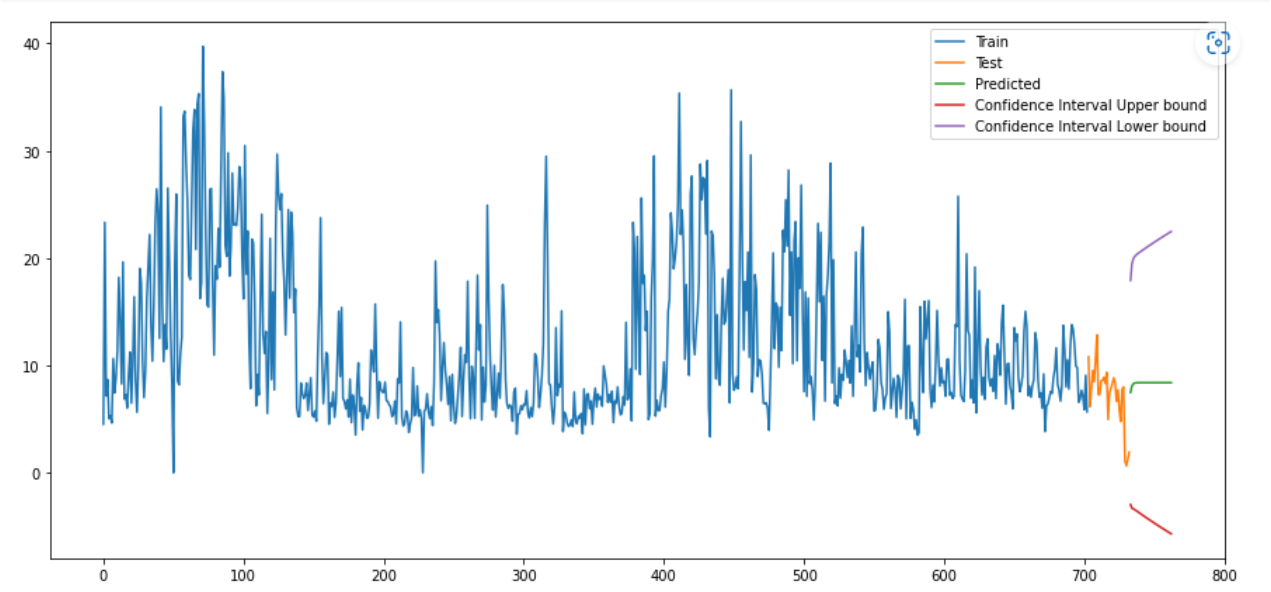
(We will compare our models using Mean Square Error-MSE, Mean Absolute Error-MAE, Mean Absolute Percentage Error-MAPE)

Model	**MSE**	**MAE**	**MAPE**
SARIMA	9.587	2.050	1.083
VAR	41.19	5.837	0.770
Convolution	19.42	3.5401	23229690.00
GARCH	19.96	4.195	3.188

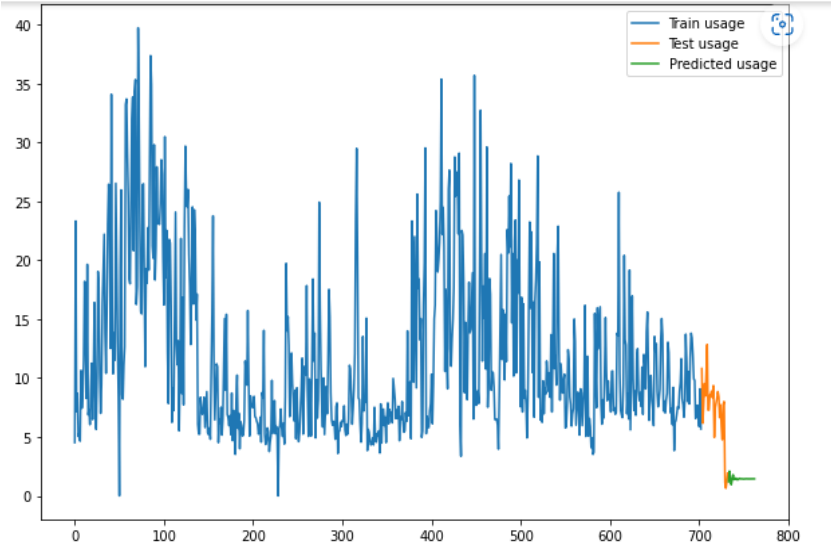
From All these results we understand that the SARIMA model gives most accurate forecasts.

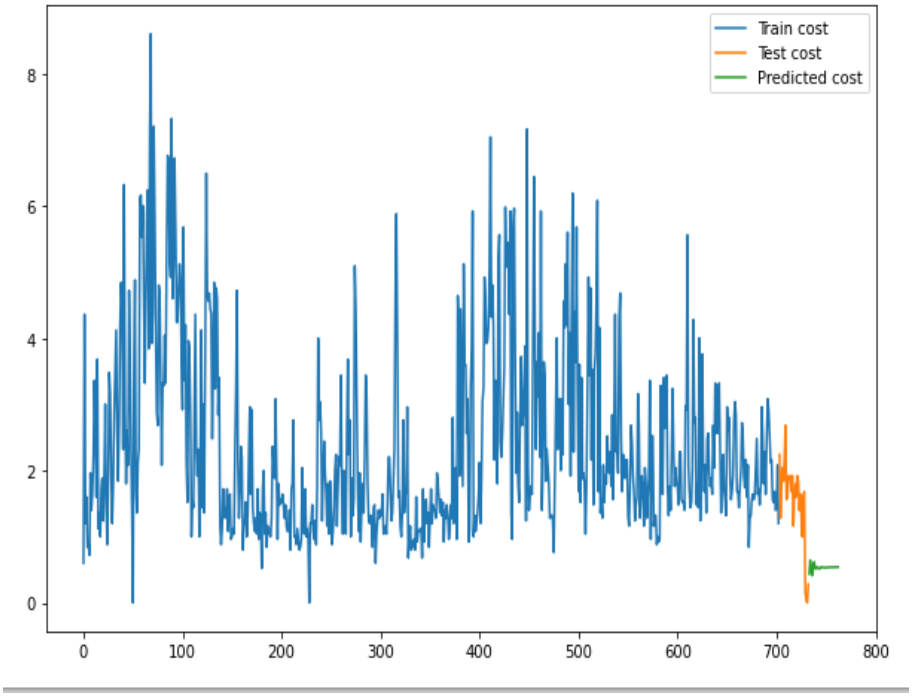
The plots of predictions versus the actual observations are:

For ARIMA Model:

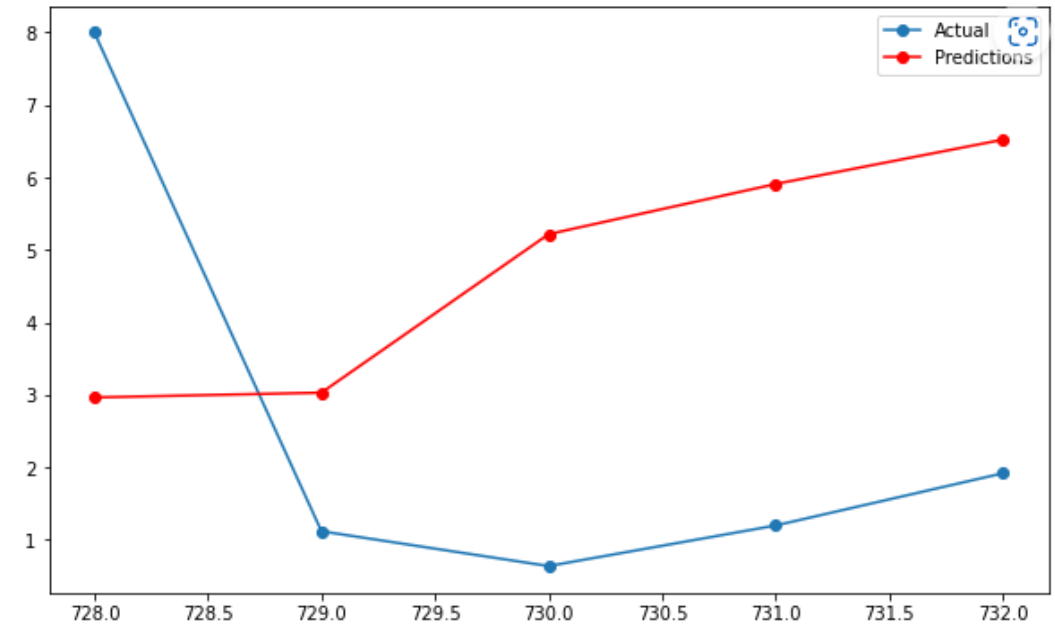


For VAR Model:





FOR GARCH Model:



CHAPTER 12. CONCLUSION

- In this project we use a SARIMA - Seasonal AutoRegression Integrated Moving Average Model for forecasting the values of Electricity Usage for future days in San Jose, California.
- The order of the SARIMA model is (1,1,1)(0,0,0)[0].

CHAPTER 13. FUTURE SCOPE

Applying Prophet model over the data

- Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects.
- It works best with time series that have strong seasonal effects and several seasons of historical data.
- The dataset has seasonality and non-linear trends so the prophet model may provide us with better forecasting results

CHAPTER 14. REFERENCES

- ➔ Decomposition Of Time Series
- ➔ Smoothing of Time Series
- ➔ Data Preprocessing for Time Series predictive modeling
- ➔ Smoothing Techniques for time series data
- ➔ Time Series: ARIMA Model
- ➔ Time Series Analysis
- ➔ Vector Auto Regression(VAR)
- ➔ Convolutional Neural Networks
- ➔ ARCH and GARCH