THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

DATS 6450_15: Time Series Analysis and Modeling

Instructor: Dr. Reza Jafari

Final Project

Student: Jake Lieberfarb

JCL May 5th, 2021

Table of Contents:

Abstract:

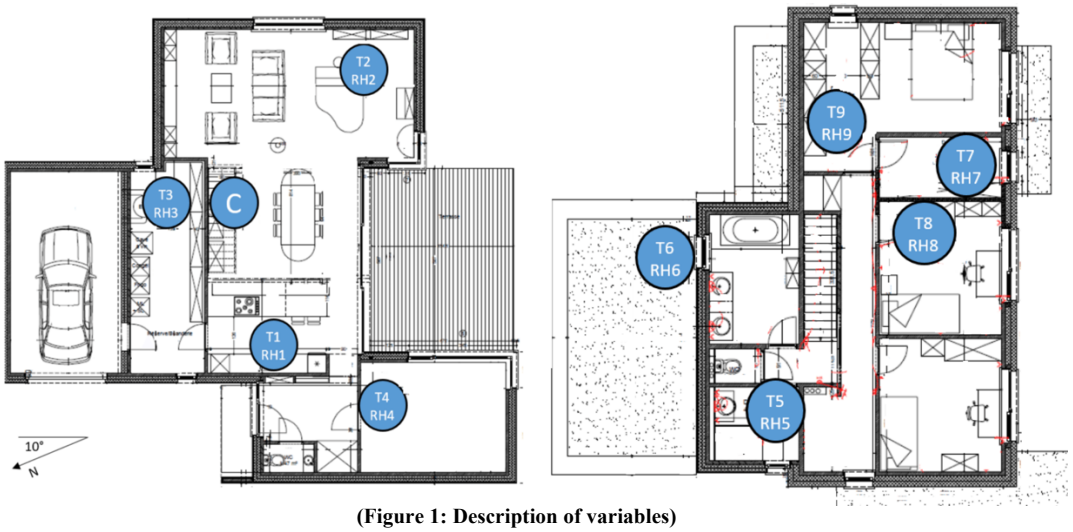
An analysis was conducted on identifying a time series model that could predict appliance electricity usage from readings on a home's temperature and humidity values as measured by the ZigBee wireless sensor network. The data contained twenty-eight different variables and had 19735 rows of data for intervals of ten minutes for 4.5 months. Different time series methods were utilized in this assessment such as Holt-Winter model, multiple linear regression, ARMA, SARIMA, and base model building (average, drift, naïve, and simple exponential smoothing). Of all the methods applied, the model that had the most accurate predictive power was the multiple linear regression model.

Introduction:

There were twenty-six independent variables and two dependent variables (Appliances and lights). All temperatures were recorded in Celsius.

Variables	**Description**	**Variables**	**Description**
Appliances	energy use in watts per hour (Wh)	T6	temperature outside the building (north side)
lights	energy use of light fixtures in the house (Wh)	RH_6	% humidity outside the building (north side)
T1	temperature in kitchen area,	T7	temperature in ironing room
RH_1	% humidity in kitchen area	RH_7	% humidity in ironing room
T2	temperature in living room area	T8	temperature in teenager room 2, in Celsius
RH_2	% humidity in living room area	RH_8	% humidity in teenager room 2
T3	temperature in laundry room area	T9	temperature in parents room, in Celsius
RH_3	% humidity in laundry room area	RH_9	% humidity in parents room
T4	temperature in office room	To*	temperature outside in Celsius
RH_4	% humidity in office room	Press_mm_hg*	pressure in mm Hg
T5	temperature in bathroom	RH_out*	% humidity outside
RH_5	% humidity in bathroom	Wind speed*	m/s
Visibility*	km	Tdewpoint*	Â°C
rv_1	random variable 1	rv_2	random variable 2

* reading taken from Chievres weather station

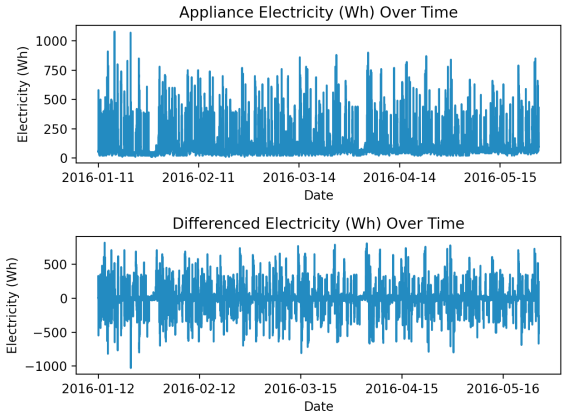


(Figure 1: Description of variables)

Once an understanding for the dependent and independent variables was obtained, the Appliance variable was selected as the sole dependent variable **(figure 1)**. The dimensions of the two-story home were presented for reference. The variable "lights" was treated as an independent variable. There was now a total of 27 independent variables predicting Appliances. First thehe original data was plotted. This was followed by checking the PACF/ACF, constructing a correlation matrix, splitting the data into 80% train and 20% test, assessing the stationarity and seasonality of the data and developing a time series decomposition.

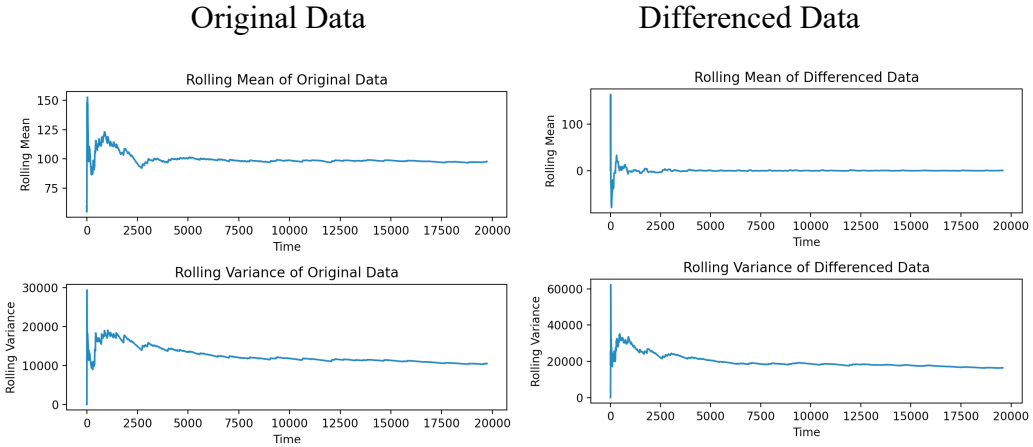
Next, 10 models were constructed. These models were: Holt-winter, multiple linear regression, ARMA(3,0), ARMA(3,1), ARMA(3,3), ARIMA(3,0,0)x(0,3,0,12), and base models (average, naïve, drift, and single exponential smoothing). The best model was identified by having the predicted lowest MSE reading on the test data. The Ljung-Box test was used to calculate the Q statistics of each model with a set lag of five. Within each step there were additional requirements and processes that were completed.

Descriptive Statistics:



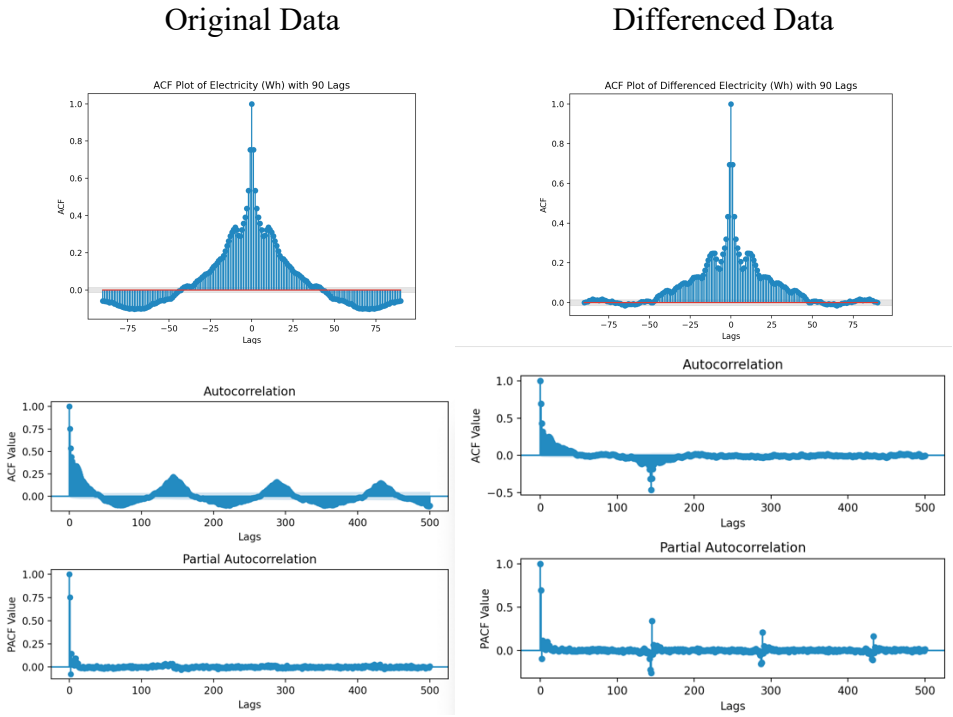
(Figure 2: Plot of Original and Differenced Data)

Although the plot for Appliances appeared to be stationary, upon closer inspection of the data, there were high levels of seasonality **(figure 2)**. Differencing of 144 was applied to the data as there were 144 ten-minute intervals in each day. The differenced data was also plotted.



(Figure 3: Rolling Mean and Variance Plot of Original and Differenced Data)

The rolling mean and variance for the original and differenced data was constructed to see, visually, if the data was stationary **(figure 3)**. Both the original and differenced mean and variance values leveled out over time. The differenced mean and variance appeared to level out at a quicker rate than the original data. This visualization supported the hypothesis that the data was stationary.



(Figure 4: PACF and ACF Plot of raw and differenced Data)

The ACF plot of the original data appeared to indicate seasonality as there were peaks in the data at every iteration of 144 **(figure 4)**. The PACF values of the original data appeared to

stationary after lag 12, the data became stationary. The ACF plot centered at zero for the original data showed clear seasonality with the lags going in a clear pattern of positive ACF values then negative.

The differenced data showed that the ACF plot became stationary after lag 46, then spiked at 144, and then became stationary right after. The PACF plot of the differenced data became stationary after lag 12 and spiked at every iteration of 144. The ACF plot centered at zero for the differenced data became stationary after lag 46.

Original Data

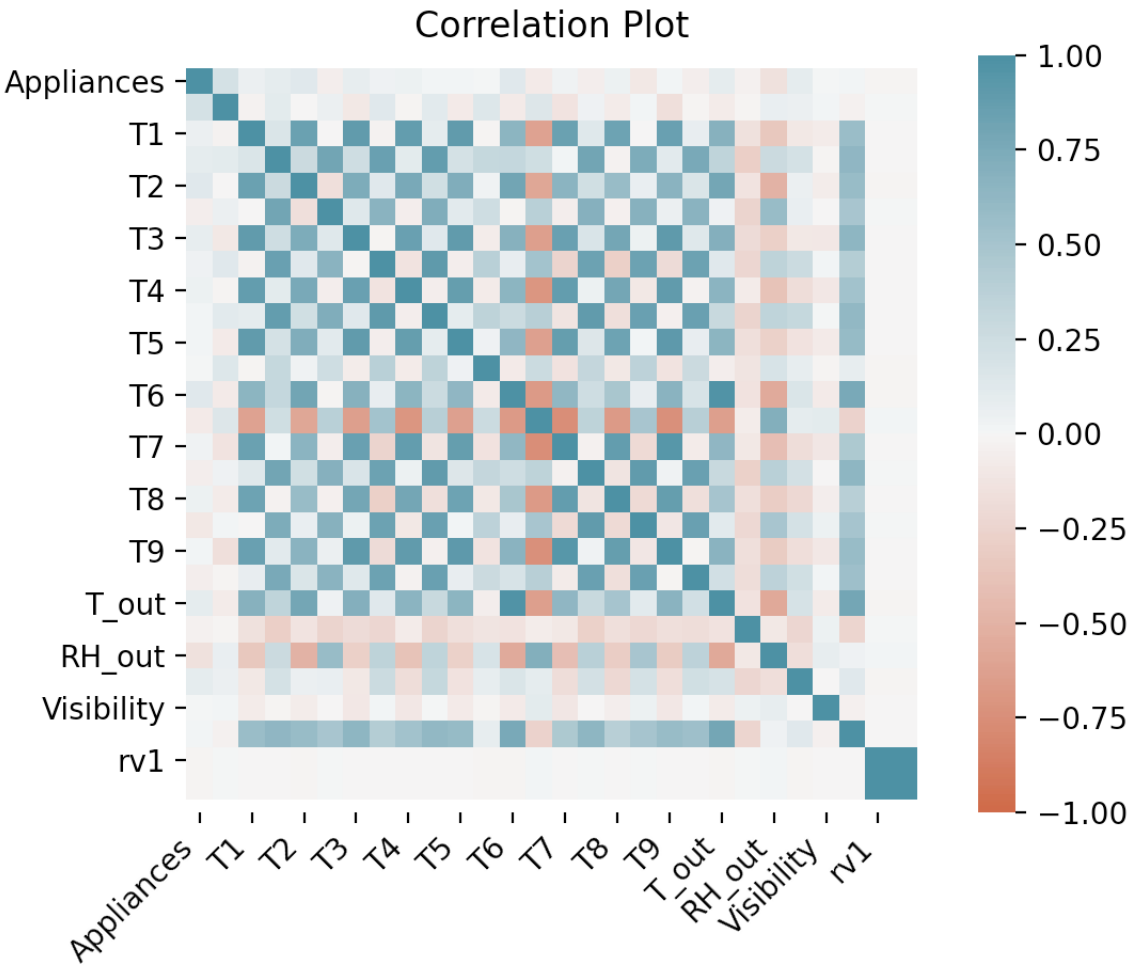
```
ADF Statistic: -20.224148
p-value: 0.000000
Critical Values:
    1%: -3.431
    5%: -2.862
    10%: -2.567
```

Differenced Data

```
ADF Statistic: -20.514060
p-value: 0.000000
Critical Values:
    1%: -3.431
    5%: -2.862
    10%: -2.567
```

(Figure 5: ADF test of raw and differenced Data)

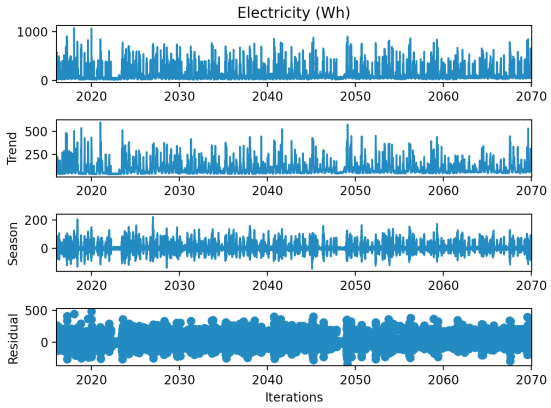
The ADF values were calculated for the original and differenced data and reported to be 0.00000 **(figure 5)**. Both the original data and the differenced data were found to be stationary.

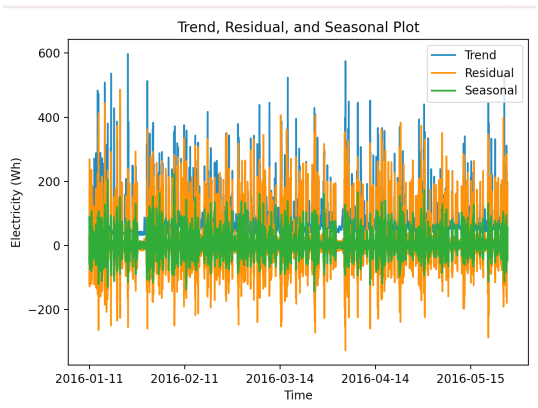


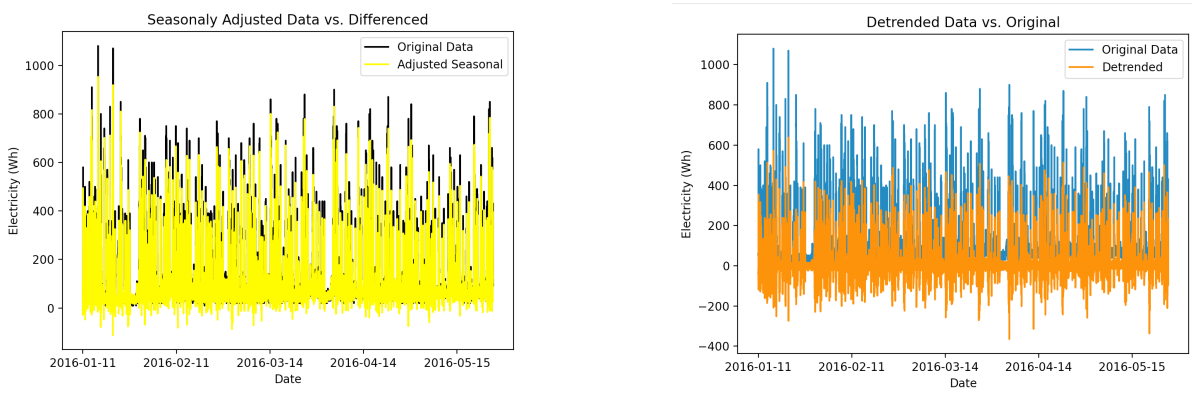
(Figure 6: ADF test of raw and differenced Data)

A correlation matrix was constructed on the data to visualize the correlations between the different variables **(figure 6)**. There was a notable "checkerboard" pattern within the plot which implied that many of the values were highly correlated and uncorrelated with one and other. Further testing was utilized to confirm this belief.

Seasonality:



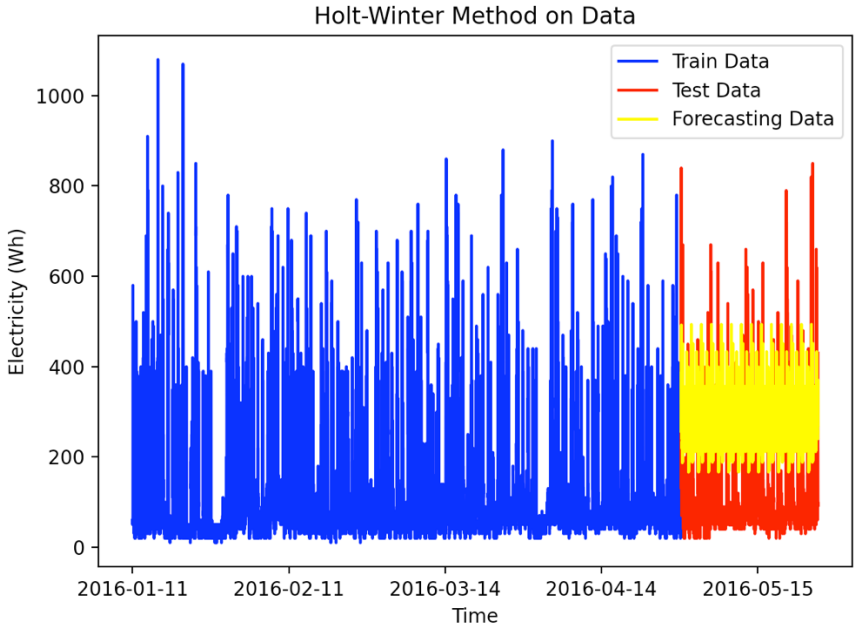


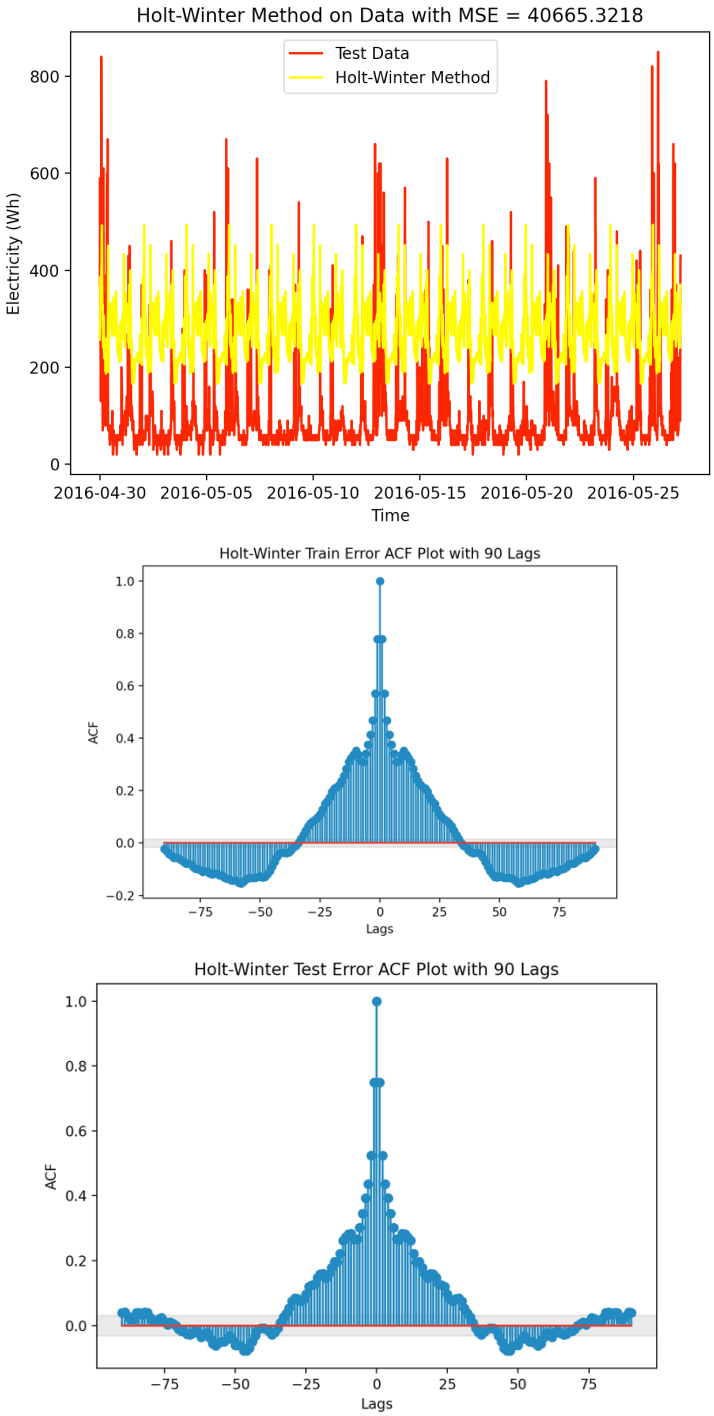


(Figure 7: ADF test of raw and differenced Data)

The plot of the residuals, seasonality, and trend were constructed to analyze the data. The residuals appeared scattered, and the seasonality trend plots showed consistently large increases and decreases in the data indicating some trend and seasonality were present **(figure 7)**. The strength of the trend and the seasonality were calculated to be 0.728 and 0.2883, respectively. The plots of the detrended and adjusted seasonal data appeared to follow the trend in the original data fairly well. The original data noticeably had a slightly higher variance than the detrended and adjusted seasonal data.

Holt- Winter Model:





(Figure 8: Holt Winter model and ACF plot)

A seasonality of 288 was used on the Holt-Winter model as it yielded the lowest MSE. The train data was utilized to develop a predictive function which was ran against the test data **(figure 8)**. The mean square error for the Holt-Winter method prediction on Electricity (Wh) was 43496.2709. The Q value was found to be 23100.1654 with a p-value of 0.0. The mean of the Holt-winter model prediction error was -172.7199. The variance of the Holt-winter model

prediction error was 13664.0900. The RMSE of the Holt-winter model prediction error was 208.5576.

An assessment was also conducted on the forecasting data. The mean square error for the Holt-Winter method forecasting on Electricity (Wh) was 40665.3218. The Q value was found to be 5138.065419 with a p-value of 0.0. The mean of the Holt-winter model forecasting error was -175.0463. The variance of the Holt-winter model forecasting error was 10024.1039. The RMSE of the Holt-winter model forecasting error was 201.6564. The variance for the prediction error was larger than the variance of the forecasting error. Overall, the Holt-winter model did not do an effective job at estimating the test data as the MSE and variance were very large. Also, the ACF plot did not appear to be capturing the variance in the error term. The ACF plots did not immediately decrease towards zero.

Feature Selection and Backwards Regression:

```
this is X dim (15788, 28)
This is H dim (28, 28)
SingularValues =  [9.49180454e+09 1.43321499e+07 6.64639821e+06 2.41644450e+06
 1.71132439e+06 1.32612807e+06 1.14252067e+06 9.44023889e+05
 1.65167716e+05 1.12684780e+05 9.14268363e+04 6.75353423e+04
 4.80002467e+04 4.32048592e+04 4.09875639e+04 2.03833935e+04
 1.26346857e+04 1.07403810e+04 9.83140093e+03 7.27434807e+03
 5.34565853e+03 3.98959028e+03 3.07348563e+03 1.96329401e+03
 9.85144085e+02 7.32816280e+02 7.12515966e-01 9.48420991e-07]
 the condition number for X is =  5.170727040402866e+17
```

Full Model

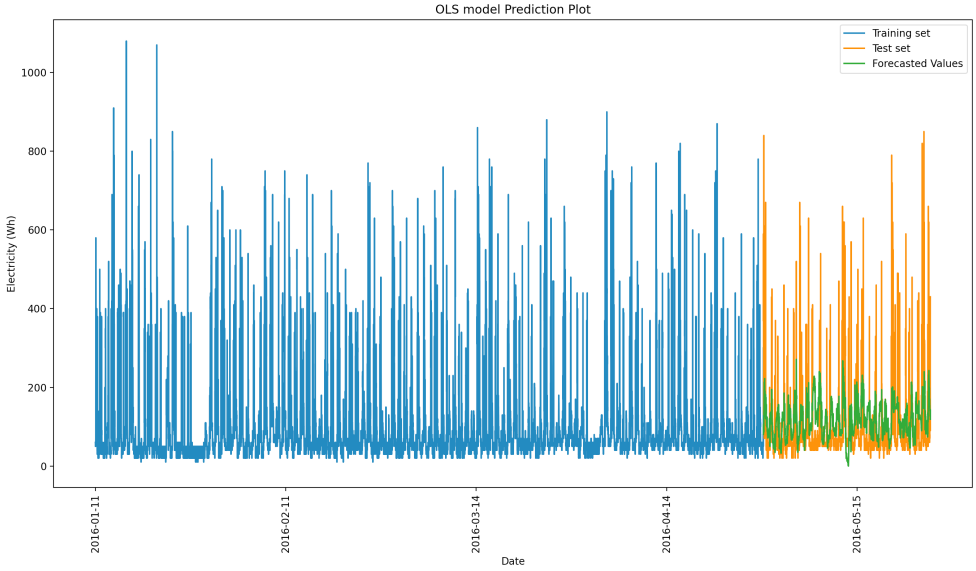
```
                            OLS Regression Results
==============================================================================
Dep. Variable:             Appliances   R-squared:                       0.173
Model:                            OLS   Adj. R-squared:                  0.172
Method:                 Least Squares   F-statistic:                     126.7
Date:                Mon, 03 May 2021   Prob (F-statistic):               0.00
Time:                        17:33:58   Log-Likelihood:                -94410.
No. Observations:               15788   AIC:                         1.889e+05
Df Residuals:                   15761   BIC:                         1.891e+05
Df Model:                          26
Covariance Type:            nonrobust
==============================================================================
```

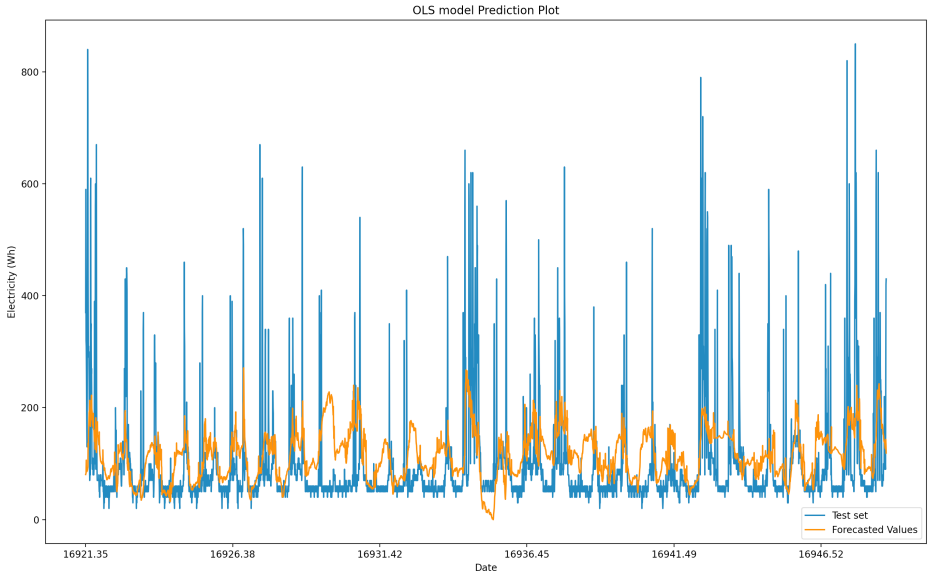
Reduced Model

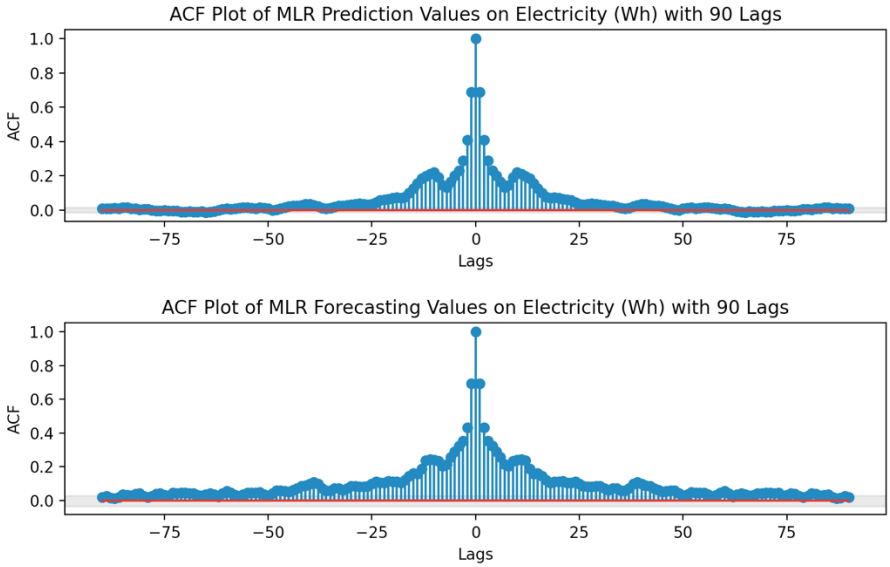
OLS Regression Results

Dep. Variable:	Appliances	R-squared (uncentered):	0.554
Model:	OLS	Adj. R-squared (uncentered):	0.554
Method:	Least Squares	F-statistic:	1309.
Date:	Tue, 04 May 2021	Prob (F-statistic):	0.00
Time:	08:12:46	Log-Likelihood:	-94460.
No. Observations:	15788	AIC:	1.890e+05
Df Residuals:	15773	BIC:	1.891e+05
Df Model:	15		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
lights	2.4038	0.098	24.407	0.000	2.211	2.597
RH_1	14.5246	0.630	23.064	0.000	13.290	15.759
T2	-18.1917	1.400	-12.997	0.000	-20.935	-15.448
RH_2	-15.3846	0.699	-22.015	0.000	-16.754	-14.015
T3	22.5514	1.022	22.056	0.000	20.547	24.555
RH_3	9.2034	0.685	13.431	0.000	7.860	10.546
T4	-7.1188	0.987	-7.214	0.000	-9.053	-5.184
RH_4	-2.8166	0.678	-4.153	0.000	-4.146	-1.487
T6	7.1959	0.744	9.678	0.000	5.739	8.653
RH_6	0.4739	0.069	6.902	0.000	0.339	0.609
T8	4.6182	0.760	6.074	0.000	3.128	6.108
RH_8	-6.1775	0.321	-19.234	0.000	-6.807	-5.548
T_out	-4.8133	0.812	-5.928	0.000	-6.405	-3.222
Windspeed	0.8224	0.359	2.289	0.022	0.118	1.527
Visibility	0.1913	0.062	3.085	0.002	0.070	0.313


OLS model Prediction Plot
Training set
Test set
Forecasted Values
Electricity (Wh)
0
200
400
600
800
1000
2016-01-11
2016-02-11
2016-03-14
2016-04-14
2016-05-15
Date


OLS model Prediction Plot
Test set
Forecasted Values
Electricity (Wh)
0
200
400
600
800
16921.35
16926.38
16931.42
16936.45
16941.49
16946.52
Date



(Figure 9: Singular Value Decomposition, Condition Number, and Backward Selection)

The singular value decomposition (SVD) and condition number were constructed and reported **(figure 9)**. Only the last value in the SVD was found to be close to zero. The condition number was greater than 1000 so there may be multicollinearity within the data set. Backwards regression was employed as the method for simplifying the full model. The number of independent variables dropped from 27 to 15 and the R^2 moved from 0.172 to 0.554. The reduced model had p-values for each coefficient under the designated alpha of 0.05. So, every independent variable found in the model was useful for the model construction.

Multiple regression:

```
b_12= df_x_t[['lights', 'RH_1', 'T2', 'RH_2', 'T3',
        'RH_3', 'T4', 'RH_4', 'T6', 'RH_6', 'T8',
        'RH_8', 'T_out', 'Windspeed',
        'Visibility']]
model = sm.OLS(df_y_t_n,b_12)
results_t = model.fit()
print(results_t.summary())
```

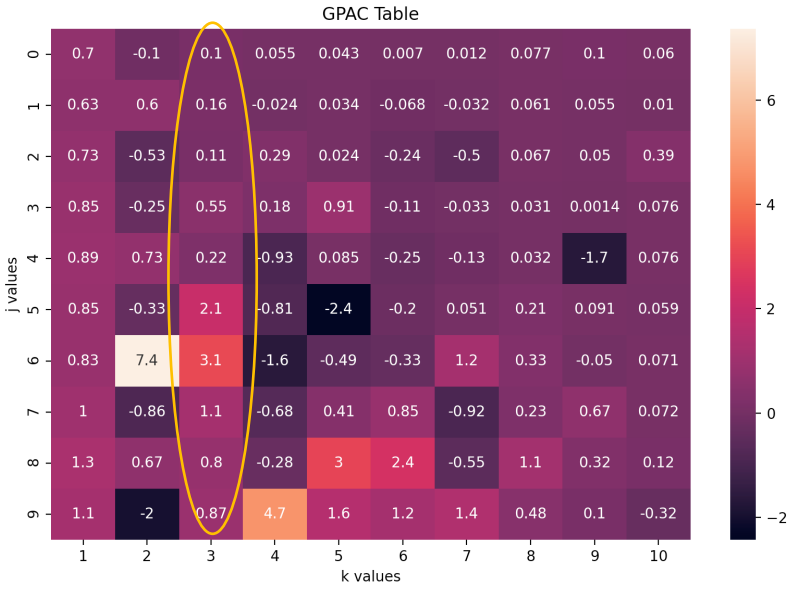
(Figure 10: Final Regression Parameters)

After experimenting with different combinations of parameters, it was found that the most accurate model was the backward reduced model with 0.554 R^2 and an adjusted R^2 of 0.554 **(figure 10)**. The AIC and BIC values were 1.890e+05 and 1.891e+5, respectively. The F-stat was 1309 which was significant and led the researcher to conclude that this model was more effective at predicting Appliance Electricity (Wh) than an empty model. The corresponding p-values for each parameter in the reduced model were below alpha of 0.05 which indicated they were useful terms in predicting Appliance Electricity (Wh). The Mean square error for the regression method prediction on Electricity (Wh) was 9211.4584. The Q value was found to be

12906.653 with a p-value of 0.0. The mean of the regression model prediction error is -0.0103. The variance of the regression model prediction error is 9211.4582. The RMSE of the regression model prediction error was 107.0667.

The Mean square error for the regression method forecasting on Electricity (Wh) was 7925.3884. The Q value was found to be 3890.720017 with a p-value of 0.0. The mean of the regression model forecasting error is -17.6336. The variance of the regression model forecasting error is 7614.4434. The variance of the prediction error appeared larger than the variance of the testing error. The RMSE of the regression model forecasting error was 121.6338. Since the Q value had a corresponding p value of 0.0, the model was not doing an effective job at capturing the variance in the data. This was confirmed visually by the ACF plots of the test and train data as although the ACF values did eventually go towards zero, there were many highly ACF values in the early lags.

ARMA(3,0):



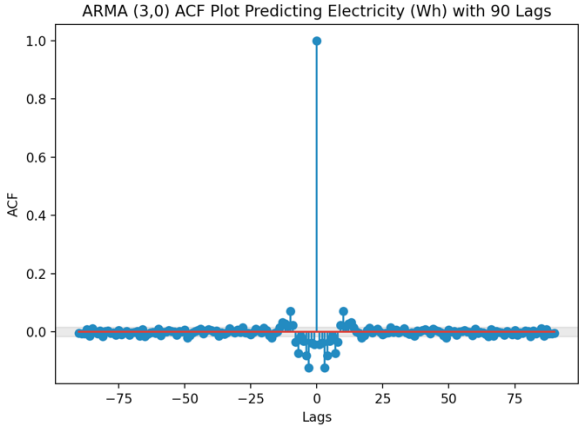
ARMA Model Results

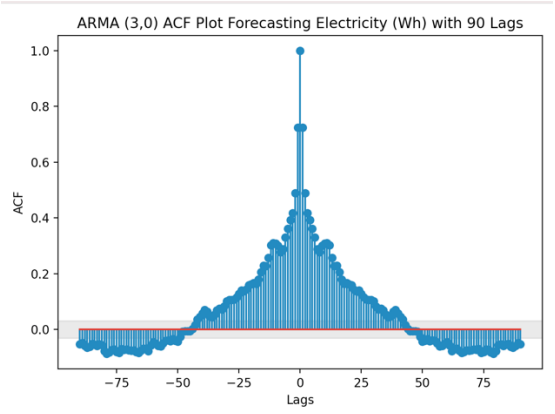
Dep. Variable:	Appliances	No. Observations:	15788
Model:	ARMA(3, 0)	Log Likelihood	-89390.984
Method:	css-mle	S.D. of innovations	69.614
Date:	Tue, 04 May 2021	AIC	178789.969
Time:	15:32:32	BIC	178820.637
Sample:	01-11-2016 - 04-30-2016	HQIC	178800.119

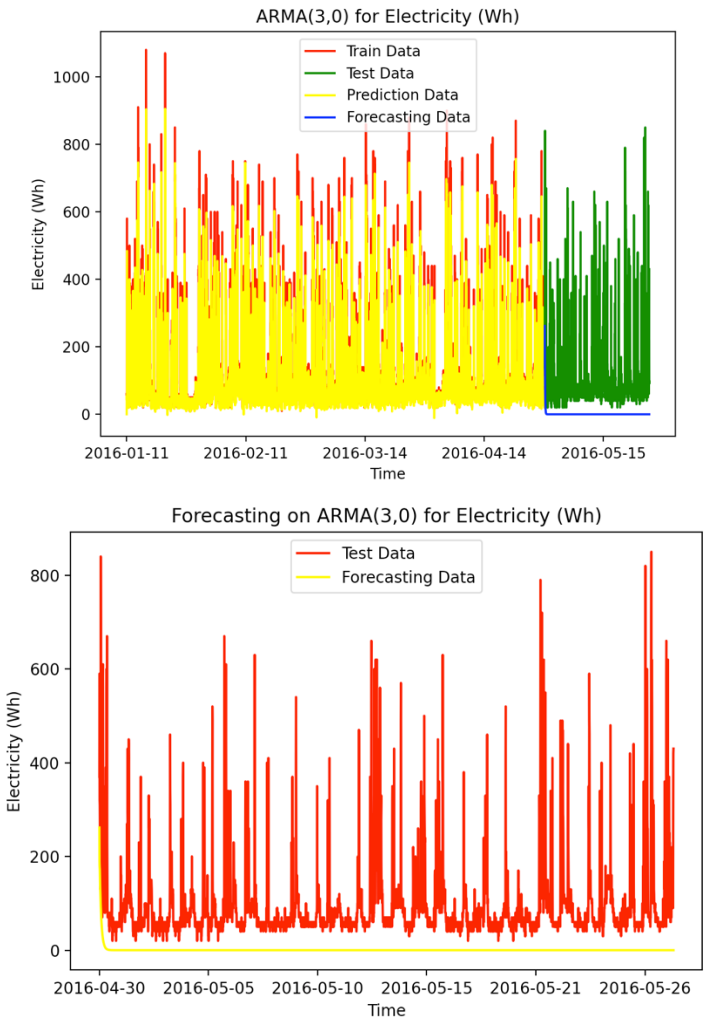
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1.Appliances	0.8826	0.008	112.958	0.000	0.867	0.898
ar.L2.Appliances	-0.1806	0.010	-17.350	0.000	-0.201	-0.160
ar.L3.Appliances	0.1915	0.008	24.515	0.000	0.176	0.207

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.0938	-0.0000j	1.0938	-0.0000
AR.2	-0.0755	-2.1834j	2.1847	-0.2555
AR.3	-0.0755	+2.1834j	2.1847	0.2555







(Figure 11: ARMA(3,0) and plots)

A generalized partial autocorrelation plot was constructed to assist in selected the ARMA parameters that would predict the data **(figure 11)**. There appeared to be a row of constant zeros at (3,0), (3,1) and (3,3). So 3 ARMA models were developed for each instance. The coefficients for the models were reported in the screenshot above. The estimate appeared to be biased as it did not follow the original data exactly. There did not appear to be any zero/pole cancelation.

For ARMA(3,0) descriptive statistics were calculated to assess the efficiency of the model. The Mean square error for the ARMA(3,0) method prediction on Electricity (Wh) was 4846.3313. The Q value was found to be 414.214595 with a p-value of 2.560013e-87. The mean for the ARMA(3,0) model error was 10.4497. The variance for the ARMA(3,0) model error was 4737.1346. the RMSE for the ARMA(3,0) model error was 69.61560.

the covariance Matrix for the data was:

	ar.L1.Appliances	ar.L2.Appliances	ar.L3.Appliances
ar.L1.Appliances	6.105410e-05	-0.000054	7.277838e-07
ar.L2.Appliances	-5.374990e-05	0.000108	-5.374481e-05
ar.L3.Appliances	7.277838e-07	-0.000054	6.104928e-05

The standard error coefficients were:

ar.L1.Appliances	0.007814
ar.L2.Appliances	0.010410
ar.L3.Appliances	0.007813

The confidence intervals did not contain zero and therefore were statistically significant. There were

	Upper Bound	Lower Bound
ar.L1.Appliances	0.867311	0.897940
ar.L2.Appliances	-0.201005	-0.160200
ar.L3.Appliances	0.176234	0.206862

An analysis was also conducted on the forecasting data as well. The MSE for the forecasting data was found to be 17223.5194. The Q value was found to be 4824.9665 with a p-value of 0.0. The mean for the ARMA(3,0) model forecasting error was 95.7142. The variance for the ARMA(3,0) model forecasting error was 8062.3070. The RMSE for the ARMA(3,0) model error was 131.2384. The variance of the prediction error was less than the variance of the forecasting error. The model had a lower MSE than the other prior models. The Q statistics p-value was still around 0 but it was larger than any of the prior models. The ACF plot of the train data appeared to be capturing much of the variances the ACF values quickly went towards zero. The ACF plot of the test data indicated that the model was not capturing the variance in the data. More ARMA models were explored.

ARMA(3,1)

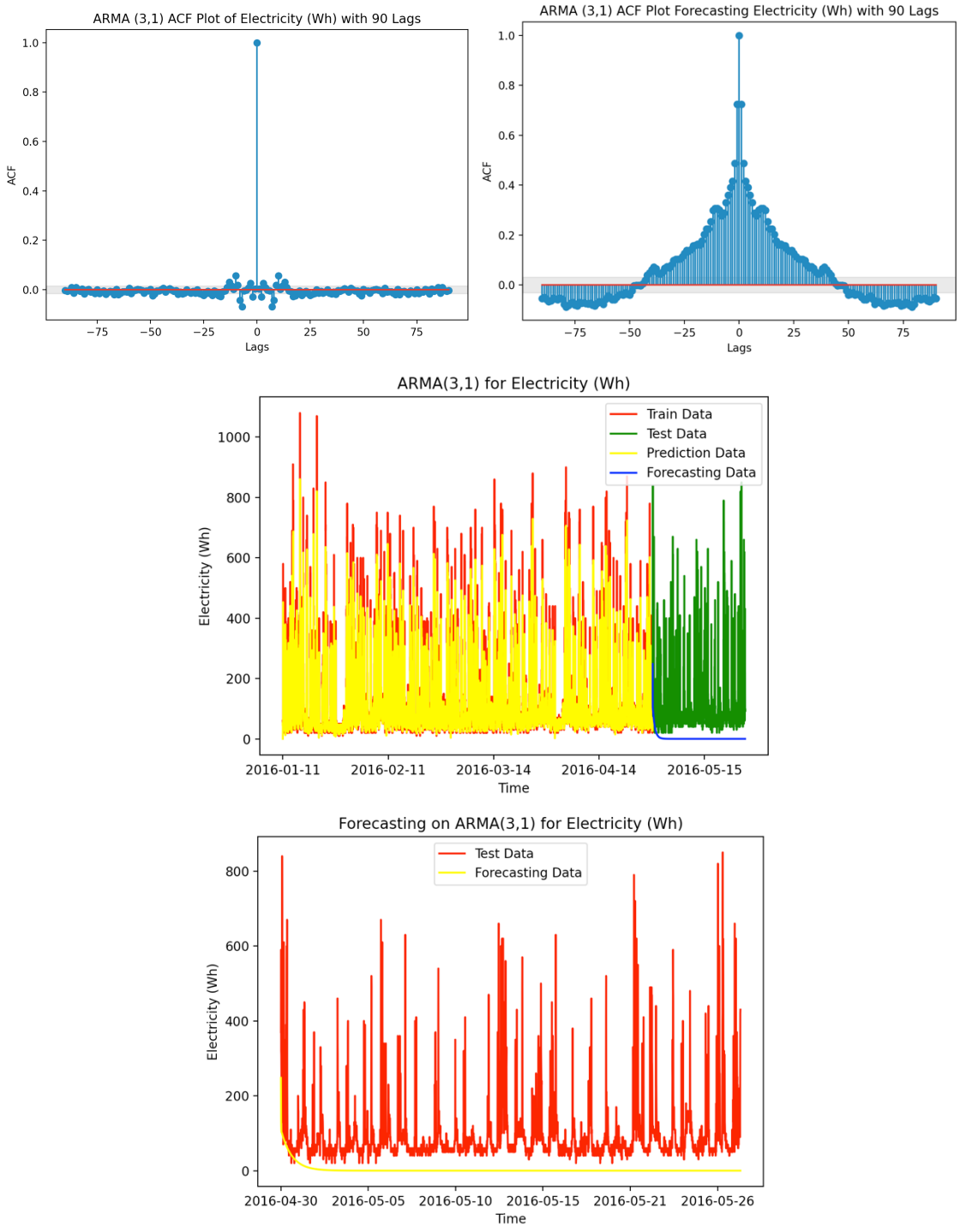
ARMA Model Results

Dep. Variable:	Appliances	No. Observations:	15788
Model:	ARMA(3, 1)	Log Likelihood	-88969.936
Method:	css-mle	S.D. of innovations	67.781
Date:	Tue, 04 May 2021	AIC	177949.873
Time:	16:44:26	BIC	177988.208
Sample:	01-11-2016 - 04-30-2016	HQIC	177962.560

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1.Appliances	1.6812	0.010	160.404	0.000	1.661	1.702
ar.L2.Appliances	-0.8576	0.014	-59.938	0.000	-0.886	-0.830
ar.L3.Appliances	0.1712	0.008	20.208	0.000	0.155	0.188
ma.L1.Appliances	-0.8749	0.008	-114.263	0.000	-0.890	-0.860

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.0109	-0.0000j	1.0109	-0.0000
AR.2	1.9995	-1.3344j	2.4039	-0.0937
AR.3	1.9995	+1.3344j	2.4039	0.0937
MA.1	1.1430	+0.0000j	1.1430	0.0000



(Figure 12: ARMA(3,1) and plots)

An ARMA(3,1) was also constructed **(figure 12)**. The coefficients for the model was reported in the screenshot above. There did not appear to be any zero/pole cancelation. The estimate was biased as it did not follow the original data exactly. The ACF plot of the training data appeared to capture the data as many of the ACF values were close to zero, The ACF plot for the test data appeared to indicate that the model was not capturing much of the variance in the

test data. The prediction data followed the training data fairly well, but the forecasting data did not follow the test data very well as indicated by the charts. The mean square error for the ARMA(3,1) method prediction on Electricity (Wh) was 4594.581. The Q value was found to be 24.179986 with a p-value of 0.0002. The mean for the ARMA(3,1) model error is 4.09056. The variance for the ARMA(3,1) model error was 4577.8491. The RMSE for the ARMA(3,1) model error was 67.7833. The covariance Matrix for the data was reported below with only the first and last columns of the 4x4 table:

	ar.L1.Appliances	...	ma.L1.Appliances
ar.L1.Appliances	0.000110	...	-0.000053
ar.L2.Appliances	-0.000120	...	0.000025
ar.L3.Appliances	0.000014	...	0.000024
ma.L1.Appliances	-0.000053	...	0.000059

the standard error coefficients were:

ar.L1.Appliances	0.010481
ar.L2.Appliances	0.014308
ar.L3.Appliances	0.008471
ma.L1.Appliances	0.007657

The confidence intervals were:

	Lower Bound	Upper Bound
ar.L1.Appliances	1.660669	1.701755
ar.L2.Appliances	-0.885630	-0.829544
ar.L3.Appliances	0.154574	0.187779
ma.L1.Appliances	-0.889873	-0.859860

The model had a lower MSE than the other ARMA(3,0). An analysis was also conducted on the forecasting data as well. The MSE for the forecasting data was found to be 16830.2026. The Q value was found to be 4809.263056 with a p-value of 0.0. The mean for the ARMA(3,1) model forecasting error was 93.7375. The variance for the ARMA(3,1) model forecasting error was 8043.466. The variance of the prediction error was less than the variance of the forecasting error. The RMSE for the ARMA(3,1) model error was 129.7313. The Q statistics p-value for the train data was greater than the other models explored thus far but it was still below the alpha of 0.05. One more ARMA model was explored.

ARMA(3,3):

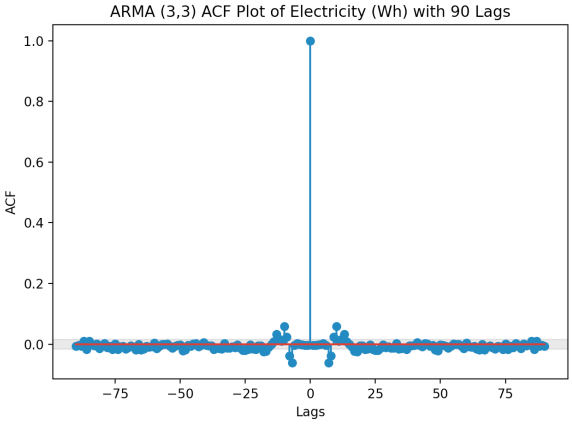
ARMA Model Results

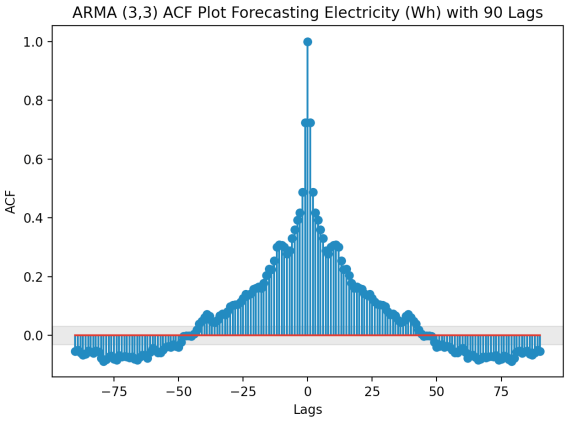
Dep. Variable:	Appliances	No. Observations:	15788
Model:	ARMA(3, 3)	Log Likelihood	-88950.819
Method:	css-mle	S.D. of innovations	67.699
Date:	Tue, 04 May 2021	AIC	177915.638
Time:	16:59:02	BIC	177969.307
Sample:	01-11-2016 - 04-30-2016	HQIC	177933.400

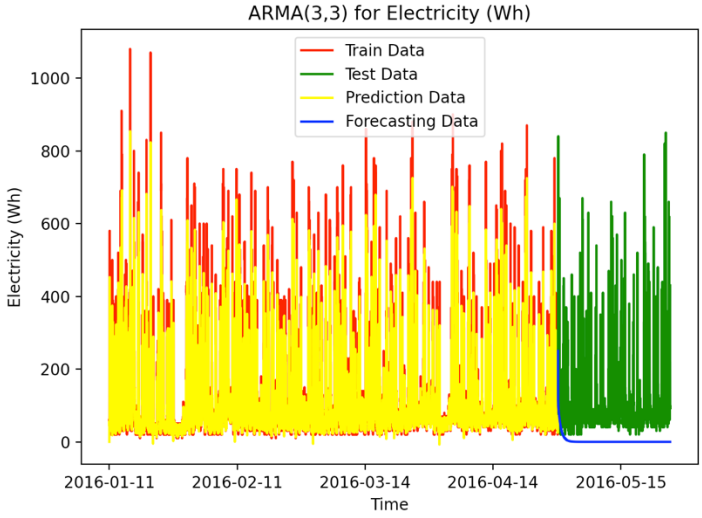
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1.Appliances	0.4280	0.018	23.784	0.000	0.393	0.463
ar.L2.Appliances	0.9865	0.015	64.236	0.000	0.956	1.017
ar.L3.Appliances	-0.4256	0.017	-24.506	0.000	-0.460	-0.392
ma.L1.Appliances	0.3858	0.018	21.359	0.000	0.350	0.421
ma.L2.Appliances	-0.8553	0.013	-67.896	0.000	-0.880	-0.831
ma.L3.Appliances	-0.2519	0.011	-22.811	0.000	-0.274	-0.230

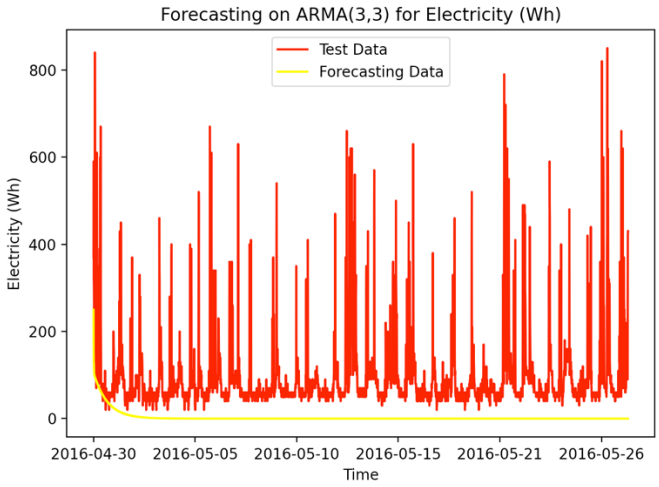
Roots

	Real	Imaginary	Modulus	Frequency
AR.1	-1.0056	+0.0000j	1.0056	0.5000
AR.2	1.0099	+0.0000j	1.0099	0.0000
AR.3	2.3136	+0.0000j	2.3136	0.0000
MA.1	1.1222	+0.0000j	1.1222	0.0000
MA.2	-1.0080	+0.0000j	1.0080	0.5000
MA.3	-3.5094	+0.0000j	3.5094	0.5000









(Figure 13: ARMA(3,3) and plots)

An ARMA(3,3) was also constructed **(figure 13)**. The coefficients for the model were reported in the screenshot above. There did not appear to be any zero/pole cancelation. This estimate was biased as it did not follow the train data exactly. The ACF plot for the train data appeared to be capturing much of the variance in the data as the ACF values quickly went to zero. The plot of the train data showed the prediction values following the data fairly well. The forecasting data did not follow the test data very well as it quickly declined and was stationary right above zero. The mean square error for the ARMA(3,3) method predicting on Electricity (Wh) was 4583.4875. The Q value was found to be 0.637048 with a p-value of 0.986244. The mean for the ARMA(3,3) model error was 3.9264. The variance for the ARMA(3,3) model error was 4568.071. The RMSE for the ARMA(3,3) model error was 67.70146. The covariance Matrix for the data was constructed below with only the first and last columns displaying of the 6x6 table.

	ar.L1.Appliances	...	ma.L3.Appliances
ar.L1.Appliances	0.000324	...	0.000114
ar.L2.Appliances	-0.000116	...	0.000016
ar.L3.Appliances	-0.000195	...	-0.000127
ma.L1.Appliances	-0.000294	...	-0.000148
ma.L2.Appliances	-0.000040	...	-0.000043
ma.L3.Appliances	0.000114	...	0.000122

The standard error coefficients were

ar.L1.Appliances	0.017993
ar.L2.Appliances	0.015358
ar.L3.Appliances	0.017367
ma.L1.Appliances	0.018064
ma.L2.Appliances	0.012597
ma.L3.Appliances	0.011043

The confidence intervals was

	Lower Bound	Upper Bound

ar.L1.Appliances 0.392684 0.463217
ar.L2.Appliances 0.956445 1.016648
ar.L3.Appliances -0.459647 -0.391569
ma.L1.Appliances 0.350428 0.421237
ma.L2.Appliances -0.879957 -0.830579
ma.L3.Appliances -0.273543 -0.230255

ARMA(3,3) had a significant p-value and therefore was explaining much of the variation in the data. It was the most effective ARMA model, and its coefficient was used in the SARIMA model as well.

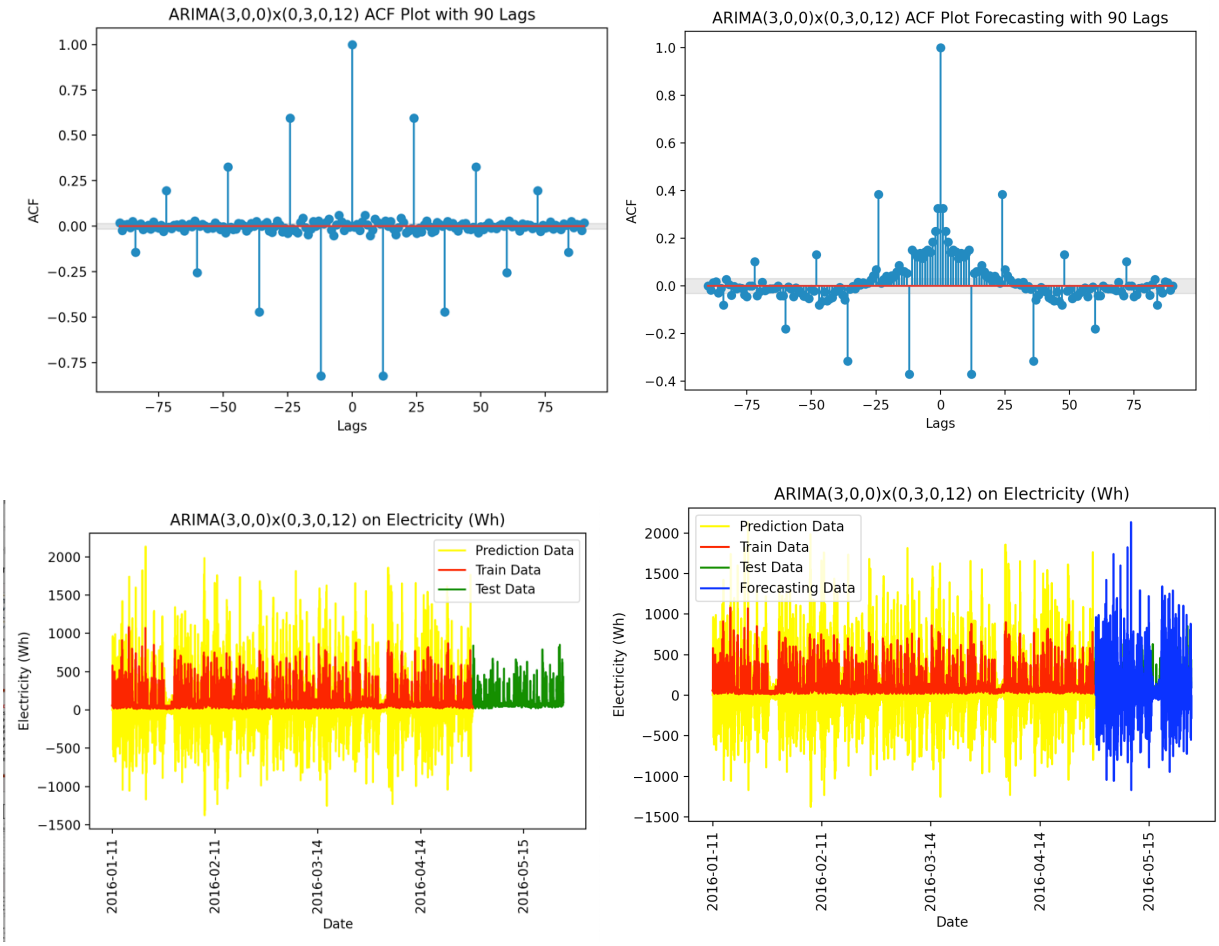
An analysis was also conducted on the forecasting data. The MSE for the forecasting data was found to be 16809.6514. The Q value was found to be 4820.00526 with a p-value of 0.0 . The mean for the ARMA(3,3) model forecasting error was 93.5672. The variance for the ARMA(3,3) model forecasting error was 8054.8379. The variance of the prediction error was less than the variance of the forecasting error. The RMSE for the ARMA(3,3) model error was 129.6503. The Q value for the train data showed that the model was capturing enough of the data to explain the variance but when the model was introduced on the testing data, it did not perform as well.

SARIMA model:

SARIMAX Results

Dep. Variable:	Appliances	No. Observations:	15788
Model:	SARIMAX(3, 0, 0)x(0, 3, 0, 12)	Log Likelihood	-108653.972
Date:	Tue, 04 May 2021	AIC	217315.943
Time:	19:00:08	BIC	217346.601
Sample:	01-11-2016	HQIC	217326.091
	- 04-30-2016		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.7017	0.005	135.342	0.000	0.692	0.712
ar.L2	-0.2475	0.006	-40.828	0.000	-0.259	-0.236
ar.L3	0.1099	0.006	19.758	0.000	0.099	0.121
sigma2	3.276e+04	178.254	183.800	0.000	3.24e+04	3.31e+04



(Figure 14: ARIMA(3,0,0)x(0,3,0,12) and plots)

An ARIMA(3,0,0)x(0,3,0,12) was constructed for modeling the data **(figure 14)**. The values of 3,3 were use as they had yielded the lowest MSE score out of all the ARMA models. A ARIMA(3,0,0)x(0,3,0,144) was attempted but the researcher's computer was not capable of constructing a chart and the computer would crash. So, an alternative of seasonal differencing by 12 was introduced as a substitute. The parameter estimates were displayed in the SARIMA screenshot above. The data appeared biased as the prediction values did not follow the train data exactly. The ACF plot for the prediction error showed ACF values close to zero and there were cases of seasonality where the ACF values would increase or decrease every twelfth lag. The ACF plot for the forecasting errors appeared had a similar pattern in significant lags for every twelfth iteration. Eventually many of the values came close to zero as the lags increased. The plot of the training, predicting, testing and forecasting showed the predicting data and forecasting data to have a much greater variance in values than the train or test data.

The mean square error for the ARIMA(3,0,0)x(0,3,0,12) method prediction on Electricity (Wh) was 57571.3242. The Q value was found to be 78.976724 with a p-value of 1.373662e-15. The mean for the ARIMA(3,0,0)x(0,3,0,12) model error was 0.04235. The variance for the ARIMA(3,0,0)x(0,3,0,12) model error was 57571.3224. The RMSE for the ARIMA(3,0,0)x(0,3,0,12) model error is 239.94025.

the covariance matrix for the data was

	ar.L1	ar.L2	ar.L3	sigma2
ar.L1	0.000027	-0.000019	0.000002	-0.065337
ar.L2	-0.000019	0.000037	-0.000017	0.170987
ar.L3	0.000002	-0.000017	0.000031	-0.068476
sigma2	-0.065337	0.170987	-0.068476	31774.395873

The standard error for the coefficients were

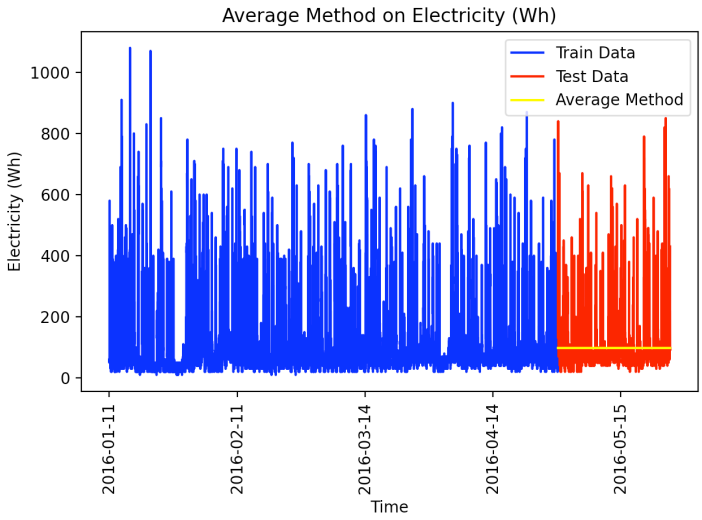
ar.L1	0.005185
ar.L2	0.006062
ar.L3	0.005561
sigma2	178.253740

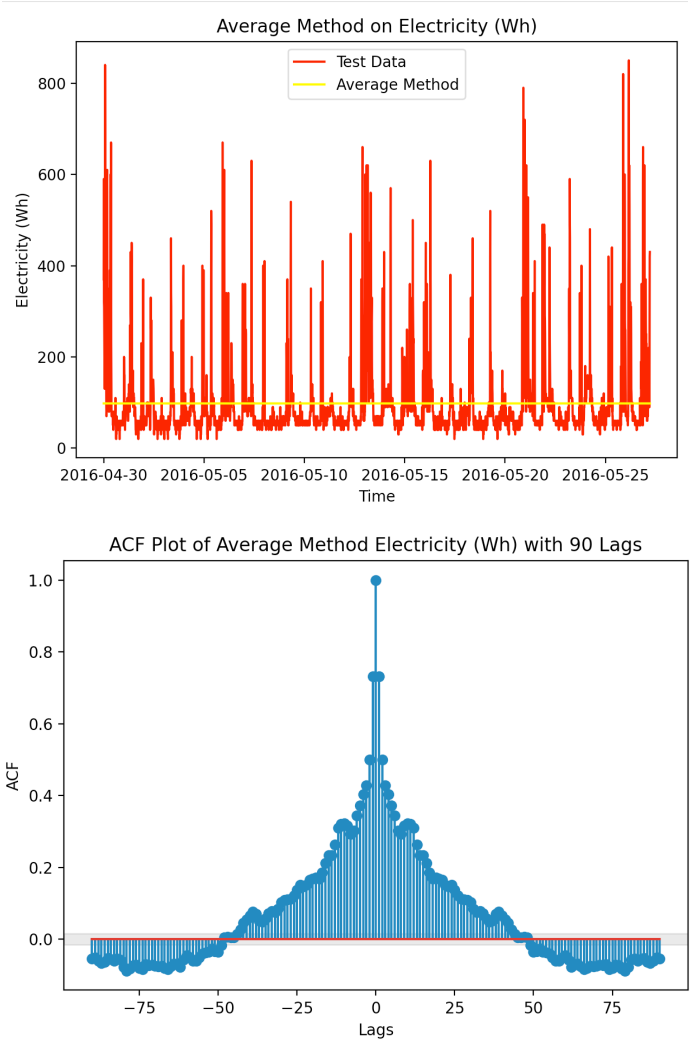
The confidence intervals were

	Lower Bound	Upper Bound
ar.L1	0.691534	0.711858
ar.L2	-0.259363	-0.235602
ar.L3	0.098975	0.120774
sigma2	32413.665709	33112.407530

An analysis was also conducted on the testing data for the SARIMA model. The MSE for the forecasting data was found to be 75328.2380. The Q value was found to be 923.114 with a p-value of 2.648581e-197. The mean for the ARIMA(3,0,0)x(0,3,0,12) model forecasting error was -0.796476. The variance for the ARIMA(3,0,0)x(0,3,0,12) model forecasting error was 75327.6037. The variance of the prediction error was less than the variance of the forecasting error. The RMSE for the ARIMA(3,0,0)x(0,3,0,12) model error was 274.4599. Seasonal differencing of twelve was not an effective measure at predicting variance. The Q statistic p-value was very close to zero. Also, the MSE was very large for the model.

Base Models (Average Method):

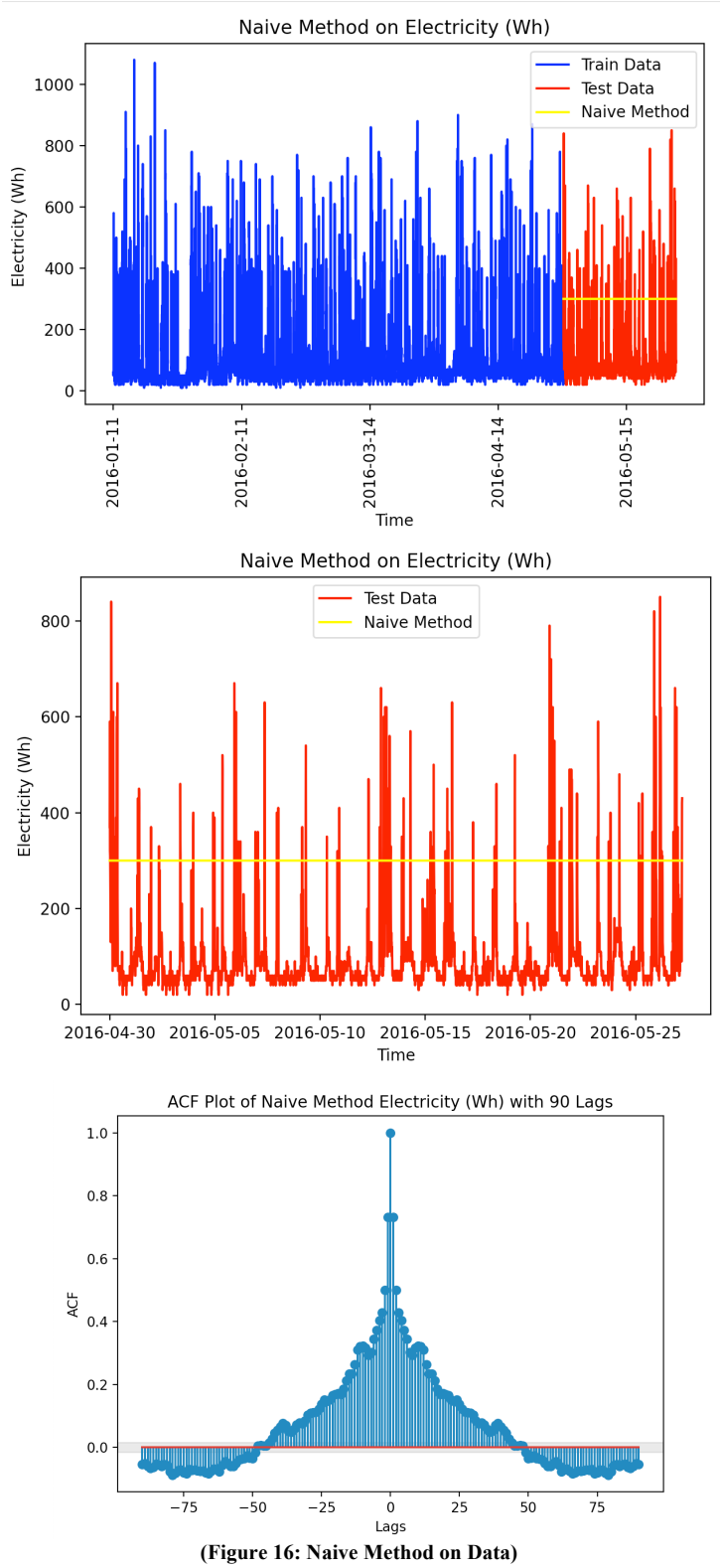




(Figure 15: Average Method on Data)

An analysis was conducted on the base model of the average method **(figure 15)**. The mean square error for the average method training set was 11074.3459. The mean square error for the average method testing set was 8290.728. The variance of the error of the average method training set was 11070.31250. The variance of the error of the average method testing set was 8288.01394. The variance for the prediction error was greater than the variance of the forecasting error. The RMSE of the Average model forecasting error was 91.0534. The mean of the Average model forecasting error was -1.6475. The Q value was found to be 5014.178759 with a p-value of 0.0. Since the model had a high Q statistic and low corresponding p-value, it was not doing an effective job at capturing the variance in the dataset. The ACF plot confirmed this assessment as the ACF values showed signs of seasonality as the values appeared to follow a seasonal trend.

Base Models (Naive Method):



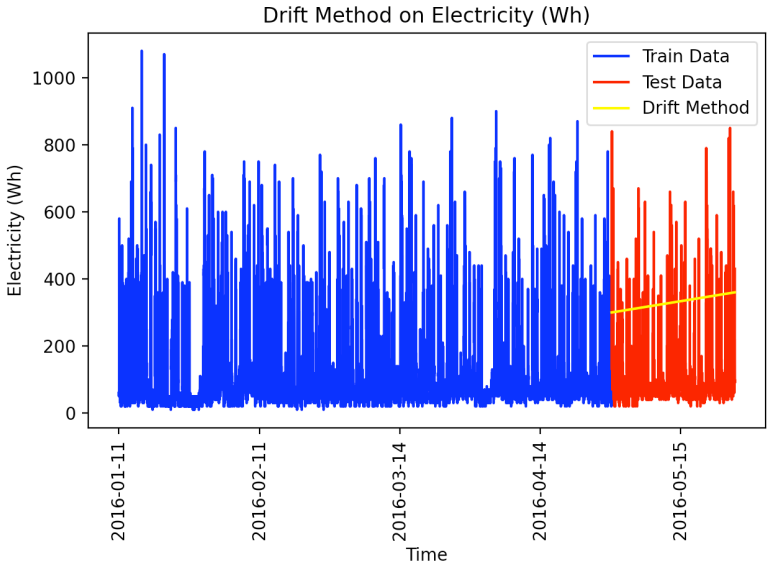
(Figure 16: Naive Method on Data)

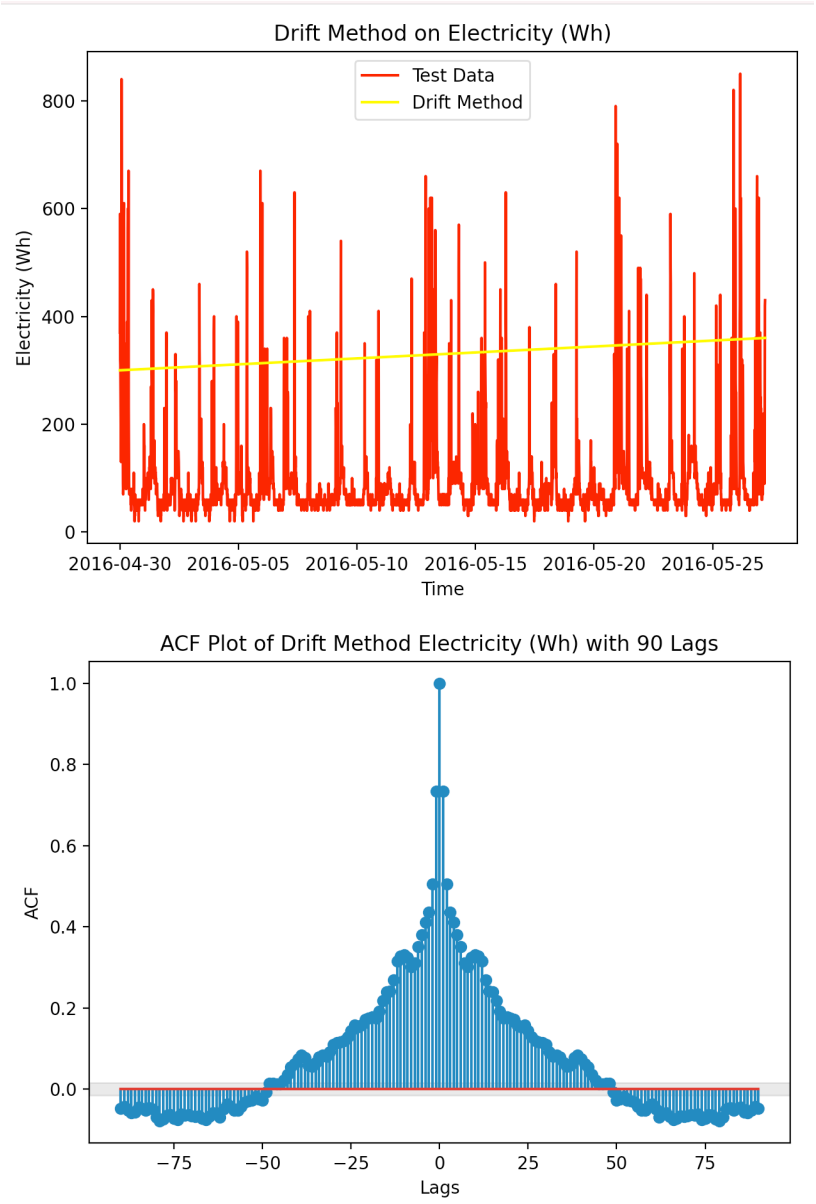
A Naïve model was constructed on the test dataset **(figure 16)**. The mean square error for the Naive method training set was 5381.1617. The mean square error for the Naive method

testing set was 49750.3420. The variance of the error of the average method training set was 5381.1615.

The Variance of the error of the average method testing set was 8288.0139. The variance for the prediction error was less than the variance of the forecasting error. The RMSE of the Naive model forecasting error was 223.0478. The mean of the Naive model forecasting error is -203.6230. The Q value was found to be 5014.178759 with a p-value of 0.0. Since the model had a high Q statistic and low corresponding p-value, it was not doing an effective job at capturing the variance in the dataset. The ACF plot appeared to not be capturing the variance in the data as there appeared to be a seasonal trend in the ACF values where there was a steady decline followed by an incline in ACF values.

Base Models (Drift Method):

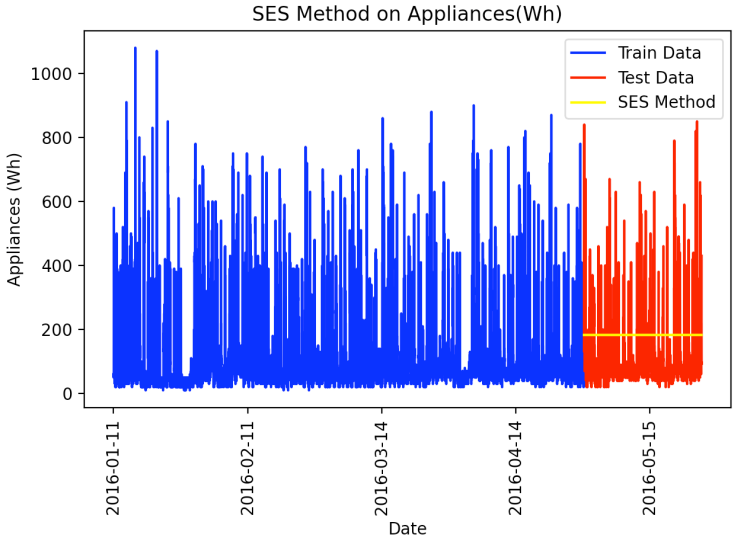


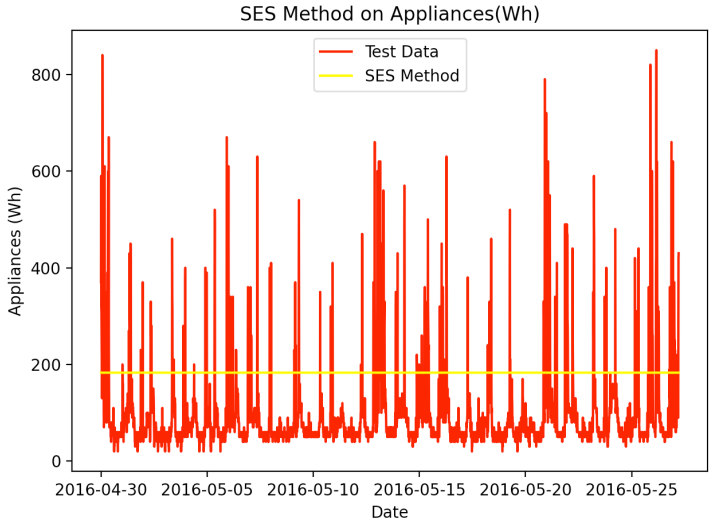


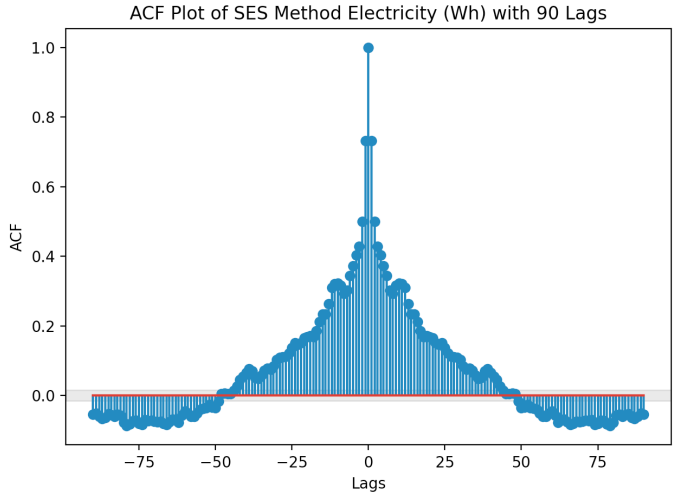
(Figure 17: Drift Method on Data)

A Drift model was constructed for the dataset **(figure 17)**. The mean square error for the drift method training set was 5385.1842. The mean square error for the drift method testing set was 62980.3347. The variance of the error of the Drift method training set was 5385.18408. The Variance of the error of the Drift method testing set was 8396.1867. The variance of the prediction error was less than the variance of the forecasting error. The RMSE of the Drift model forecasting error was 250.9588. The mean of the Drift model forecasting error was -233.6325. The Q value was found to be 5129.25267 with a p-value of 0.0. Since the model had a high Q statistic and low corresponding p-value, it was not doing an effective job at capturing the variance in the dataset. The ACF plot appeared to not be capturing the variance in the data as there appeared to be a seasonal trend in the ACF values where there was a steady decline followed by an incline in ACF values.

Base Models (SES Method):





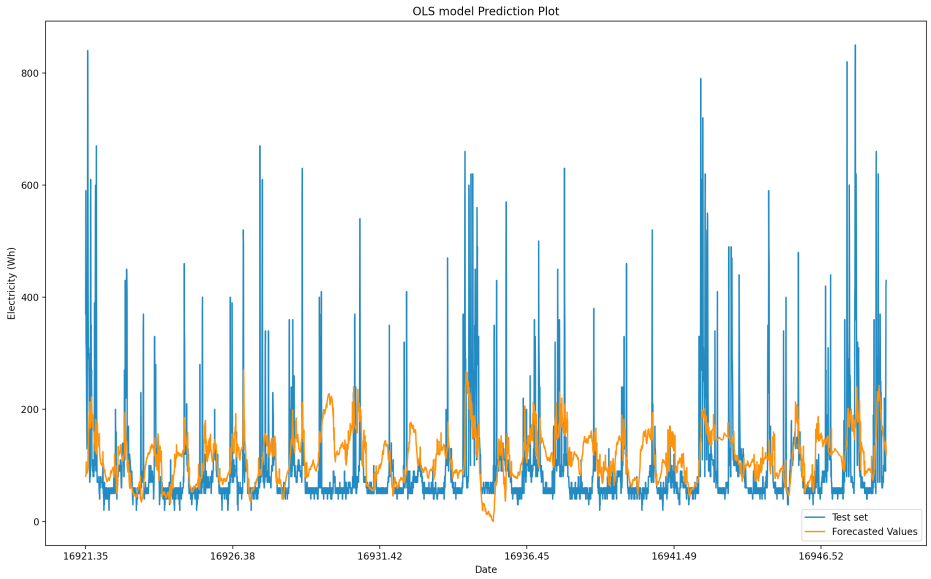


(Figure 18: SES Method on Data)

A singular exponential smoothing (SES) plot was constructed for the data **(figure 18)**. The mean square error for the SES method training set was 5717.6955. The mean square error for the SES method testing set was 15784.7008. The variance of the error of the SES method training set was 5706.26816. The variance of the error of the SES method testing set was 8288.0139. The RMSE of the SES model error was 125.6372. The mean of the SES model error is -86.5834. The Q value was found to be 5014.178759 with a p-value of 0.0. Since the model had a high Q statistic and low corresponding p-value, it was not doing an effective job at capturing the variance in the dataset. The ACF plot appeared to not be capturing the variance in the data as there appeared to be a seasonal trend in the ACF values where there was a steady decline followed by an incline in ACF values.

Forecasting function and h-step prediction:

$$\widehat{y_t} = lights(2.4038) + * RH_1(14.5246) + T_2(-18.1917) + RH_2(-153846) + T_3(22.5514) + RH_3(9.2034) + T_4(-7.1188) + RH_4(-2.8166) + T_6(7.1959) + RH_6(0.4739) + T_8(4.6182) + RH_8(-6.1775) + T_{out}(-4.8133) + Windspeed(0.8224) + Visibility(0.1913)$$



(Figure 19: Forecasting Function and Plot of Data)

Within the forecast function, a total of 15 variables were included to make the prediction on the data **(figure 19)**. The h-step prediction was constructed and displayed. The chart of the forecast function and the test data showed the values to be following the same general path. The forecasting function did not dramatically increase or decrease in values when compared to the test data.

Summary and Conclusion:

Model	MSE of Testing Data
Holt-winter	40665.3218
Linear Regression	7925.3884
ARMA(3,0)	17223.5195
ARMA(3,1)	16830.2027
ARMA(3,3)	16809.65143
ARIMA(3,0,0)x(0,3,0,12)	75328.2380
Average	8290.728
Naive	49750.342
Drift	62980.3347
SES	15784.7008

(Figure 20: MSE of Testing Chart)

The model that had the lowest MSE value for the forecasting data was the linear regression model **(figure 20)**. Therefore, the linear regression model was selected as the final model. The R^2 value of 0.554 only explained about half of the variation in the data but any other model that was explored by the researcher only did marginally better. This was corroborated by its ACF plot **(figure 9)** as the ACF values did not immediately go down towards zero. The model that had the lowest MSE for the training data was the ARMA(3,3). However, when this model was forecasted on the test data, it did not perform as well as the linear regression model. The model that came in second for the forecasting data was the Average method approach. The only major issue encountered in this assessment was that SARIMA modeling could not be completed for differencing at lag 144 as the researcher's computer would crash after the code ran for a while. The alternative of seasonal differencing by 12 was substituted in. The researcher would've liked to have seen a SARIMA model with seasonal differencing of 144 but was not able to construct this. For future analysis on Appliance energy consumption, more homes should be included in the sample size. Also, there should be a recording of how many people were present in the house at each 10-minute interval and if they were asleep or awake. By knowing these factors, energy usage could potentially be better approximated.

Appendix:

```
import numpy as np
import pandas as pd
from numpy import linalg as LA
import statsmodels.api as sm
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from statsmodels.tsa.seasonal import STL
import statsmodels.tsa.holtwinters as ets
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.holtwinters import ExponentialSmoothing as HWES
```

```
from statsmodels.tsa.arima_model import ARIMA
from scipy.stats import boxcox
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_pacf, plot_acf
from sklearn.metrics import mean_squared_error
import pmdarima as pmd
from pmdarima.metrics import smape
from pmdarima.arima import ndiffs
from sklearn.metrics import mean_squared_error
from scipy.stats import chi2
from scipy import signal
from pandas import DataFrame
import seaborn as sns
import warnings
warnings.filterwarnings('ignore',
'statsmodels.tsa.arima_model.ARMA',FutureWarning)

# read the data
df = pd.read_csv('KAG_energydata_complete.csv')
df['Date']=pd.to_datetime(df['date'])
del df['date']
df.set_index('Date', inplace = True)


# 5.a: plot the Appliance vs time
def difference(dataset, interval):
   diff = []
   for i in range(interval, len(dataset)):
      value = dataset[i] - dataset[i - interval]
      diff.append(value)
   return diff


# applied 144 differencing as
# there were 10 minute intervals in the data and 6 intervals/hr * 24 hr/day =
144
Diff_1 = difference(df.Appliances, 144)
DF = pd.DataFrame(Diff_1, index=df.index[ 144:])
DF.rename(columns={0:'Appliances'}, inplace=True)


plt.figure()
plt.subplot(211)
plt.plot(df.index, df.Appliances)
plt.xlabel('Date')
plt.xticks(df.index[::4500], fontsize= 10)
plt.ylabel('Electricity (Wh)')
plt.title('Appliance Electricity (Wh) Over Time')
plt.subplot(212)
plt.plot(DF.index, DF.Appliances)
plt.xlabel('Date')
plt.xticks(DF.index[::4500], fontsize= 10)
plt.ylabel('Electricity (Wh)')
plt.title('Differenced Electricity (Wh) Over Time')
plt.tight_layout()
plt.show()
```

```
# =============== #
# Rolling Average #
# =============== #
def rolling_mv(y, title):
    means = []
    vars = []

    for i in range(1, len(y)):
        means.append(y[:i].mean())
        vars.append(y[:i].var())

    # plot rolling mean
    plt.figure()
    plt.subplot(211)
    plt.plot(means, label= 'Mean')
    plt.title("Rolling Mean "  + str(title))
    plt.xlabel('Time')
    plt.ylabel("Rolling Mean")
    plt.subplot(212)
    plt.plot(vars, label= 'Variance')
    plt.title("Rolling Variance " + str(title))
    plt.xlabel('Time')
    plt.ylabel("Rolling Variance")
    plt.tight_layout()
    plt.show()

# plot rolling average
rolling_mv(df.Appliances, 'of Original Data')
rolling_mv(DF.Appliances, 'of Differenced Data')


 # 5.b: ACF/PACF plots ACF Plot
def autocorr(x,lag):
     l = range(lag+1)
     x_br = np.mean(x)
     autocorr = []
     for i in l:
         num = 0
         var = 0
         for j in range(i, len(x)):
             num += np.sum(x[j]- x_br) *(x[j-i] -x_br)
         var = np.sum((x - x_br) ** 2)
         autocorr.append(num/var)
     return autocorr

def ACF_Plot(x,lag):
    lg = np.arange(-lag, lag + 1)
    x = x[0:lag + 1]
    rx = x[::-1]
    rxx = rx[:-1] + x
    plt.stem(lg, rxx)



# ACF plot of raw data
plt.figure()
bound = 1.96/np.sqrt(len(df.Appliances))
```

```
ACF_Plot(autocorr(df.Appliances,90),90)
plt.xlabel('Lags')
plt.ylabel('ACF')
plt.title('ACF Plot of Electricity (Wh) with 90 Lags')
plt.axhspan(-bound,bound,alpha = .1, color = 'black')
plt.tight_layout()
plt.show()

# ACF plot of differenced data
plt.figure()
bound = 1.96/np.sqrt(len(DF.Appliances))
ACF_Plot(autocorr(DF.Appliances,90),90)
plt.xlabel('Lags')
plt.ylabel('ACF')
plt.title('ACF Plot of Differenced Electricity (Wh) with 90 Lags')
plt.axhspan(-bound,bound,alpha = .1, color = 'black')
plt.tight_layout()
plt.show()


def PACF_ACF_Plot(x,lags,title):
    plt.figure()
    plt.subplot(211)
    plt.xlabel('Lags')
    plt.ylabel('ACF Value')
    plt.title('ACF and PACF plot ' + str(title))
    sm.graphics.tsa.plot_acf(x, ax=plt.gca(), lags=lags)
    plt.subplot(212)
    plt.xlabel('Lags')
    plt.ylabel('PACF Value')
    sm.graphics.tsa.plot_pacf(x, ax=plt.gca(), lags=lags)
    plt.tight_layout()
    plt.show()

PACF_ACF_Plot(df.Appliances,500, 'of Raw Data')
PACF_ACF_Plot(DF.Appliances,500, 'of Differenced Data')

 # 5c: Matrix Correlation
# Q2: Heat Map
plt.figure()
corr_t= df.corr()
# create correlation coefficient heatmap for x,y,g, and z
ax = sns.heatmap(corr_t, vmin=-1, vmax= 1, center=0, cmap =
sns.diverging_palette(20,220, n=200), square= True)
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, horizontalalignment=
'right')
plt.title('Correlation Plot')
plt.tight_layout()
plt.show()
 # 5d:Cleaning

# tested other forms of differencing


 # 5e:Split data of differenced data
```

```
df2= df[[ 'Appliances', 'lights', 'T1', 'RH_1', 'T2', 'RH_2', 'T3',
        'RH_3', 'T4', 'RH_4', 'T5', 'RH_5', 'T6', 'RH_6', 'T7', 'RH_7', 'T8',
        'RH_8', 'T9', 'RH_9', 'T_out', 'Press_mm_hg', 'RH_out', 'Windspeed',
        'Visibility', 'Tdewpoint', 'rv1', 'rv2']]
# df.set_index('Date')


df_y= df2[['Appliances']]

df_x = df2[['T1', 'lights','RH_1', 'T2', 'RH_2', 'T3',
        'RH_3', 'T4', 'RH_4', 'T5', 'RH_5', 'T6', 'RH_6', 'T7', 'RH_7', 'T8',
        'RH_8', 'T9', 'RH_9', 'T_out', 'Press_mm_hg', 'RH_out', 'Windspeed',
        'Visibility', 'Tdewpoint', 'rv1', 'rv2']]


# x data
df_x_t, df_x_f= train_test_split(df_x, shuffle= False, test_size=0.2)
# differenced
df_y_t, df_y_f = train_test_split(DF, shuffle= False, test_size=0.2)
# undifferenced
df_y_t_n, df_y_f_n = train_test_split(df_y, shuffle= False, test_size=0.2)
#first date 2016-01-11 17:00:00
#last date  2016-05-27 18:00:00

 #  6: Stationary
def ADF_Cal(x):
    result = adfuller(x)

    print("ADF Statistic: %f" %result[0])
    print('p-value: %f' % result[1])
    print('Critical Values:')
    for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))

# Data appeared to be stationary with p-value of 0.00000
ADF_Cal(df_y_t.Appliances) #seasonal data
ADF_Cal(DF.Appliances) # non-seasonal

# 7: Time series decomposition

Appliances = df['Appliances']
Appliances = pd.Series(np.array(df['Appliances']),
                       index = pd.date_range('2016-01-11 17:00:00', periods=
len(Appliances)), name= 'Electricity (Wh)')

res = STL(Appliances).fit()
fig = res.plot()
plt.ylabel('Residual')
plt.xlabel('Iterations')
plt.tight_layout()
plt.show()

T = res.trend
S = res.seasonal
R = res.resid
```

```
R = np.array(R)
S = np.array(S)
T = np.array(T)

# strength of seasonality and Trend
Ft = np.max([0,1 - np.var(R)/np.var(T+R)])
Fs = np.max([0,1 - np.var(R)/np.var(S+R)])

print('The strength of trend for this data set is', Ft.round(4))

print('The strength of seasonality for this data set is ', Fs.round(4))

plt.figure()
plt.plot(df.index,T, label= 'Trend')
plt.plot(df.index,R, label= 'Residual')
plt.plot(df.index,S, label= 'Seasonal')
plt.title('Trend, Residual, and Seasonal Plot')
plt.xticks(df.index[::4500], fontsize= 10)
plt.ylabel('Electricity (Wh)')
plt.xlabel('Time')
plt.legend()
plt.tight_layout()
plt.show()

adjusted_seasonal = Appliances - S # Adjusted Seasonal Dataset
adjusted_seasonal.to_frame(name = 'Val')
detrended = Appliances - T # detrended data
detrended.to_frame(name = 'Val')


R = np.array(R)
A_S = np.array(adjusted_seasonal)
D_T = np.array(detrended)



plt.figure()
plt.plot(df.index,Appliances, label= 'Original Data', color = 'black')
plt.plot(df.index,adjusted_seasonal.values, label= 'Adjusted Seasonal', color
= 'yellow')
plt.xticks(df.index[::4500], fontsize= 10)
plt.title('Seasonaly Adjusted Data vs. Differenced')
plt.xlabel('Date')
plt.ylabel('Electricity (Wh)')
plt.legend()
plt.tight_layout()
plt.show()

plt.figure()
plt.plot(df.index,Appliances, label= 'Original Data')
plt.plot(df.index,detrended.values, label= 'Detrended')
plt.xticks(df.index[::4500], fontsize= 10)
plt.title('Detrended Data vs. Original')
plt.xlabel('Date')
plt.ylabel('Electricity (Wh)')
plt.legend()
plt.tight_layout()
```

```
plt.show()

# 8: Holt-Winter method

# use training data to fit model
model = ets.ExponentialSmoothing(df_y_t_n['Appliances'], damped_trend= True,
seasonal_periods=288, trend='add', seasonal='add').fit()

# prediction on train set
df_y_t_HW = model.forecast(steps=len(df_y_t_n['Appliances']))
df_y_t_HW = pd.DataFrame(df_y_t_HW,
columns=['Appliances']).set_index(df_y_t_n.index)
# made prediction on test set
df_y_f_HW = model.forecast(steps=len(df_y_f_n['Appliances']))
df_y_f_HW = pd.DataFrame(df_y_f_HW,
columns=['Appliances']).set_index(df_y_f_n.index)



# print the summary
print(model.summary())

# model assessment
def mse(errors):
   return  np.sum(np.power(errors,2))/len(errors)
# tain data
df_y_t_HW_error = np.array(df_y_t_n['Appliances'] - df_y_t_HW['Appliances'])
print("Mean square error for the Holt-Winter method prediction on Electricity
(Wh) is ", mse(df_y_t_HW_error).round(4))
print(sm.stats.acorr_ljungbox(df_y_t_HW_error, lags=[5], boxpierce=True,
return_df=True))
print('The Q value was found to be 23100.165439 with a p-value of 0.0')
print('the mean of the Holt-winter model prediction error is',
np.mean(df_y_t_HW_error))
print('the variance of the Holt-winter model prediction error is',
np.var(df_y_t_HW_error))
print('the RMSE of the Holt-winter model prediction error is, ',
mean_squared_error(df_y_t_n['Appliances'], df_y_t_HW['Appliances'],
squared=False))

# test data
df_y_f_HW_error = np.array(df_y_f_n['Appliances'] - df_y_f_HW['Appliances'])
print("Mean square error for the Holt-Winter method forecasting on
Electricity (Wh) is ", mse(df_y_f_HW_error).round(4))
print(sm.stats.acorr_ljungbox(df_y_f_HW_error, lags=[5], boxpierce=True,
return_df=True))
print('The Q value was found to be 5138.065419 with a p-value of 0.0')
print('the mean of the Holt-winter model error is', np.mean(df_y_f_HW_error))
print('the variance of the Holt-winter model error is',
np.var(df_y_f_HW_error))
print('the RMSE of the Holt-winter model error is, ',
mean_squared_error(df_y_f_n['Appliances'], df_y_f_HW['Appliances'],
squared=False))
print('the variance of the prediction error appeared larger than the variance
of the testing error')
```

```
# plot Holt-Winter model

# plot of full model
plt.figure()
plt.xlabel('Time')
plt.ylabel('Electricity (Wh)')
plt.title('Holt-Winter Method on Data')
plt.plot(df_y_t_n.index,df_y_t_n.Appliances,label= "Train Data", color =
'blue')
plt.plot(df_y_f_n.index,df_y_f_n.Appliances,label= "Test Data", color =
'red')
plt.plot(df_y_f_HW.set_index(df_y_f_HW.index), label = 'Forecasting Data',
color = 'yellow')
plt.xticks(df_y.index[::4500], fontsize= 10)
plt.legend()
plt.tight_layout()
plt.show()


# plot of test data
plt.figure()
plt.plot(df_y_f_n.index,df_y_f_n.Appliances,label= "Test Data", color =
'red')
plt.plot(df_y_f_HW.set_index(df_y_f_HW.index), label = 'Forecasting Data',
color = 'yellow')
plt.xlabel('Time')
plt.ylabel('Electricity (Wh)')
plt.title(f'Holt-Winter Method on Data with MSE =
{mse(df_y_f_HW_error).round(4)}')
plt.xticks(df_y_f_n.index[::725], fontsize= 10)
plt.legend()
plt.tight_layout()
plt.show()

# note
# mse 87444 # s 288

# holt-winter train data
plt.figure()
m_pred_f = 1.96/np.sqrt(len(df_y_t_n.Appliances))
ACF_Plot(autocorr(df_y_t_HW_error,90),90)
plt.xlabel('Lags')
plt.ylabel('ACF')
plt.title('Holt-Winter Train Error ACF Plot with 90 Lags')
plt.axhspan(-m_pred_f,m_pred_f,alpha = .1, color = 'black')
plt.tight_layout()
plt.show()


# holt winter test data
plt.figure()
m_pred_f = 1.96/np.sqrt(len(df_y_f_n.Appliances))
ACF_Plot(autocorr(df_y_f_HW_error,90),90)
plt.xlabel('Lags')
plt.ylabel('ACF')
plt.title('Holt-Winter Test Error ACF Plot with 90 Lags')
plt.axhspan(-m_pred_f,m_pred_f,alpha = .1, color = 'black')
```

```
plt.tight_layout()
plt.show()

# 9: Feature Selection collinearity

X = df_x_t[['T1', 'lights','RH_1', 'T2', 'RH_2', 'T3',
        'RH_3', 'T4', 'RH_4', 'T5', 'RH_5', 'T6', 'RH_6', 'T7', 'RH_7', 'T8',
        'RH_8', 'T9', 'RH_9', 'T_out', 'Press_mm_hg', 'RH_out', 'Windspeed',
        'Visibility', 'Tdewpoint', 'rv1', 'rv2']].values
Y = df_y_t_n.values
X = sm.add_constant(X)
print("this is X dim", X.shape)
H = np.matmul(X.T,X)
print('This is H dim', H.shape)
s,d,v = np.linalg.svd(H)
print('SingularValues = ', d)
#Condition number
print(" the condition number for X is = ", LA.cond(X))
# the condition number for X is =  1.5823924406110067e+17

# Q:10 model building

df_x_t = sm.add_constant(df_x_t)
df_x_f = sm.add_constant(df_x_f)

# full model
model = sm.OLS(df_y_t_n,df_x_t)
results = model.fit()
print(results.summary())

#removed 'rv1 and rv2' as it had p-value of 0.700
b_1= df_x_t[['const','T1', 'lights','RH_1', 'T2', 'RH_2', 'T3',
        'RH_3', 'T4', 'RH_4', 'T5', 'RH_5', 'T6', 'RH_6', 'T7', 'RH_7', 'T8',
        'RH_8', 'T9', 'RH_9', 'T_out', 'Press_mm_hg', 'RH_out', 'Windspeed',
        'Visibility', 'Tdewpoint']]
model = sm.OLS(df_y_t_n,b_1)
results = model.fit()
print(results.summary())



#removed 'const' as it had p-value of 0.553
b_2= df_x_t[['lights','T1', 'RH_1', 'T2', 'RH_2', 'T3',
        'RH_3', 'T4', 'RH_4', 'T5', 'RH_5', 'T6', 'RH_6', 'T7', 'RH_7', 'T8',
        'RH_8', 'T9', 'RH_9', 'T_out', 'Press_mm_hg', 'RH_out', 'Windspeed',
        'Visibility', 'Tdewpoint']]
model = sm.OLS(df_y_t_n,b_2)
results = model.fit()
print(results.summary())

#removed 'T7' as they had p-value of 0.426
b_3= df_x_t[['lights','T1', 'RH_1', 'T2', 'RH_2', 'T3',
        'RH_3', 'T4', 'RH_4', 'T5', 'RH_5', 'T6', 'RH_6', 'RH_7', 'T8',
        'RH_8', 'T9', 'RH_9', 'T_out', 'Press_mm_hg', 'RH_out', 'Windspeed',
        'Visibility', 'Tdewpoint']]
model = sm.OLS(df_y_t_n,b_3)
results = model.fit()
```

```
print(results.summary())

#removed 'RH_7' as they had p-value of 0.415
b_4= df_x_t[['lights','T1', 'RH_1', 'T2', 'RH_2', 'T3',
        'RH_3', 'T4', 'RH_4', 'T5', 'RH_5', 'T6', 'RH_6', 'T8',
        'RH_8', 'T9', 'RH_9', 'T_out', 'Press_mm_hg', 'RH_out', 'Windspeed',
        'Visibility', 'Tdewpoint']]
model = sm.OLS(df_y_t_n,b_4)
results = model.fit()
print(results.summary())

#removed 'RH_5' as they had p-value of 0.308
b_6= df_x_t[['lights','T1', 'RH_1', 'T2', 'RH_2', 'T3',
        'RH_3', 'T4', 'RH_4', 'T5', 'T6', 'RH_6', 'T8',
        'RH_8', 'T9', 'RH_9', 'T_out', 'Press_mm_hg', 'RH_out', 'Windspeed',
        'Visibility', 'Tdewpoint']]
model = sm.OLS(df_y_t_n,b_6)
results = model.fit()
print(results.summary())

#removed 'T5' as they had p-value of 0.366
b_7= df_x_t[['lights','T1', 'RH_1', 'T2', 'RH_2', 'T3',
        'RH_3', 'T4', 'RH_4', 'T6', 'RH_6', 'T8',
        'RH_8', 'T9', 'RH_9', 'T_out', 'Press_mm_hg', 'RH_out', 'Windspeed',
        'Visibility', 'Tdewpoint']]
model = sm.OLS(df_y_t_n,b_7)
results = model.fit()
print(results.summary())

#removed 'RH_out' as they had p-value of 0.109
b_8= df_x_t[['lights','T1', 'RH_1', 'T2', 'RH_2', 'T3',
        'RH_3', 'T4', 'RH_4', 'T6', 'RH_6', 'T8',
        'RH_8', 'T9', 'RH_9', 'T_out', 'Press_mm_hg', 'Windspeed',
        'Visibility', 'Tdewpoint']]
model = sm.OLS(df_y_t_n,b_8)
results = model.fit()
print(results.summary())

#removed 'Tdewpoint' as they had p-value of 0.153
b_9= df_x_t[['lights','T1', 'RH_1', 'T2', 'RH_2', 'T3',
        'RH_3', 'T4', 'RH_4', 'T6', 'RH_6', 'T8',
        'RH_8', 'T9', 'RH_9', 'T_out', 'Press_mm_hg', 'Windspeed',
        'Visibility']]
model = sm.OLS(df_y_t_n,b_9)
results = model.fit()
print(results.summary())

#removed 'RH_9' as they had p-value of 0.050
b_10= df_x_t[['lights','T1', 'RH_1', 'T2', 'RH_2', 'T3',
        'RH_3', 'T4', 'RH_4', 'T6', 'RH_6', 'T8',
        'RH_8', 'T9', 'T_out', 'Press_mm_hg', 'Windspeed',
        'Visibility']]
model = sm.OLS(df_y_t_n,b_10)
results = model.fit()
print(results.summary())

#removed 'T1' as they had standard error of 2.215
```

```
b_11= df_x_t[['lights', 'RH_1', 'T2', 'RH_2', 'T3',
        'RH_3', 'T4', 'RH_4', 'T6', 'RH_6', 'T8',
        'RH_8', 'T9', 'T_out', 'Press_mm_hg', 'Windspeed',
        'Visibility']]
model = sm.OLS(df_y_t_n,b_11)
results = model.fit()
print(results.summary())



#removed 'Press_mm_hg' as they had p-value of 0.780
# no more multicolliniarity present
b_12= df_x_t[['lights', 'RH_1', 'T2', 'RH_2', 'T3',
        'RH_3', 'T4', 'RH_4', 'T6', 'RH_6', 'T8',
        'RH_8', 'T_out', 'Windspeed',
        'Visibility']]
model = sm.OLS(df_y_t_n,b_12)
results_t = model.fit()
print(results_t.summary())


# predictions on backward regression

# #make predictions on training data
ypred = results_t.predict(df_x_t[['lights', 'RH_1', 'T2', 'RH_2', 'T3',
        'RH_3', 'T4', 'RH_4', 'T6', 'RH_6', 'T8',
        'RH_8', 'T_out', 'Windspeed',
        'Visibility']])
# prediction data
pred_t = np.array(df_y_t_n['Appliances']) - ypred


# #make predictions on testing data
ypred_f = results_t.predict(df_x_f[['lights', 'RH_1', 'T2', 'RH_2', 'T3',
        'RH_3', 'T4', 'RH_4', 'T6', 'RH_6', 'T8',
        'RH_8', 'T_out', 'Windspeed',
        'Visibility']])
# forecasting data
pred_f = np.array(df_y_f_n['Appliances']) - ypred_f


# Q8:
df_x_t= df_x_t[['lights', 'RH_1', 'T2', 'RH_2', 'T3',
        'RH_3', 'T4', 'RH_4', 'T6', 'RH_6', 'T8',
        'RH_8', 'T_out', 'Windspeed',
        'Visibility']].copy()

plt.figure()
plt.plot(df_x_t.index, df_y_t_n.values, label = 'Training set')
plt.xticks(df.index[::4500], fontsize= 10)
# I found it difficult to read the data with the prediction data plotted so I
uncommented it.
# if you wish to see the prediction data just uncomment the line below.
# plt.plot(df_x_t.index, pred_t, label = 'Prediction values')
plt.plot(df_x_f.index, df_y_f_n, label = 'Test set')
plt.plot(df_x_f.index,ypred_f, label = 'Forecasted Values')
plt.legend()
```

```
plt.tight_layout()
plt.xlabel('Date')
plt.ylabel('Electricity (Wh)')
plt.title('OLS model Prediction Plot')
plt.show()

plt.figure()
plt.subplot(211)
m_pred_f = 1.96/np.sqrt(len(df_y_t_n.Appliances))
ACF_Plot(autocorr(pred_t.values,90),90)
plt.xlabel('Lags')
plt.ylabel('ACF')
plt.title('ACF Plot of MLR Prediction Values on Electricity (Wh) with 90
Lags')
plt.axhspan(-m_pred_f,m_pred_f,alpha = .1, color = 'black')
plt.subplot(212)
m_pred_f = 1.96/np.sqrt(len(df_y_f_n.Appliances))
ACF_Plot(autocorr(pred_f.values,90),90)
plt.xlabel('Lags')
plt.ylabel('ACF')
plt.title('ACF Plot of MLR Forecasting Values on Electricity (Wh) with 90
Lags')
plt.axhspan(-m_pred_f,m_pred_f,alpha = .1, color = 'black')
plt.tight_layout()
plt.show()



# model assessment
def mse(errors):
   return  np.sum(np.power(errors,2))/len(errors)
# train data
print("Mean square error for the regression method prediction on Electricity
(Wh) is ", mse(pred_t).round(4))
print(sm.stats.acorr_ljungbox(pred_t, lags=[5], boxpierce=True,
return_df=True))
print('The Q value was found to be 12906.653091 with a p-value of  0.0')
print('the mean of the regression model prediction error is',
np.mean(pred_t))
print('the variance of the regression model prediction error is',
np.var(pred_t))
print('the RMSE of the regression model prediction error is, ',
mean_squared_error(df_y_t_n['Appliances'], pred_t.values, squared=False))

# test data
print("Mean square error for the regression method forecasting on Electricity
(Wh) is ", mse(pred_f).round(4))
print(sm.stats.acorr_ljungbox(pred_f, lags=[5], boxpierce=True,
return_df=True))
print('The Q value was found to be 3890.720017  with a p-value of  0.0')
print('the mean of the regression model forecasting error is',
np.mean(pred_f))
print('the variance of the regression model forecasting error is',
np.var(pred_f))
print('the RMSE of the regression model forecasting error is, ',
mean_squared_error(df_y_f_n['Appliances'], pred_f.values, squared=False))
print('the variance of the prediction error appeared larger than the variance
```

```
of the testing error')

# Q11: ARMA models

def Autocorr(df, lg):
    mean = np.mean(df)
    m = len(df)
    num = []
    for i in range(lg, m):
        num_value = (df[i] - mean) * (df[i - lg] - mean)
        num.append(num_value)
    num_sum = sum(num)
    den= []
    for j in range(0, m):
        den_val = (df[j] - mean) ** 2
        den.append(den_val)
    den_sum = sum(den)
    result = round(num_sum / den_sum, 4)
    return result


def GPAC_Matrix(ry, j, k):
    col = []
    for i in range(j):
        col.append(ry[i + 1] / ry[i])
    # convert values to dataframe
    table = pd.DataFrame(col, index=np.arange(0, j).tolist())
    # list for values in the GPAC matrix
    val = []
    # for K in GPAC, you do not want to include column 1 as it will all be
1's
    for a in range(2, k + 1):
        for f in range(j):
            b_val = []
            t_val = []
            for d in range(a):
                den = []
                for h in range(a):
                    den.append(ry[abs(f - h + d)])
                b_val.append(den.copy())
                t_val.append(den.copy())
            mes = a
            for l in range(a):
                t_val[l][mes - 1] = ry[l + 1 + f]

            pac = np.linalg.det(t_val) / np.linalg.det(b_val)
            val.append(pac)
    # reshame the GPAC value so there is no 0 row in k
    GPAC = np.array(val).reshape(k -1, j)
    #correctly transpose the data
    GPAC_T = pd.DataFrame(GPAC.T)
    GPAC_F = pd.concat([table, GPAC_T], axis=1)
    GPAC_F.columns = list(range(1, k + 1))
    return GPAC_F


#ACF for 20 lags
```

```
acf =[]
for n in range(0,40):
    ac = Autocorr(df_y_t['Appliances'],n)
    acf.append(ac)


# GPAC plot

GPAC = GPAC_Matrix(acf,10,10)
plt.figure()
sns.heatmap(GPAC,  annot=True)
plt.title('GPAC Table')
plt.xlabel('k values')
plt.ylabel('j values')
plt.show()

# select parameters from GPAC ARMA:(3,0),(3,1), and (3,2)

na = 3
nb = 0
# ARMA model
model_1 = sm.tsa.ARMA(df_y_t_n.Appliances,(na, nb)).fit(trend='nc',disp=0)
for i in range(na):
    print("The AR coefficient a{}".format(i), "is:", model_1.params[i])
for i in range(nb):
    print("The MA coefficient a{}".format(i), "is:", model_1.params[i + na])
print(model_1.summary())



#MSE calculation
# train data
arma_1_pred_t = model_1.predict(start=0, end=15787)
arma_1_t_error = df_y_t_n.Appliances - arma_1_pred_t.values

# test data
arma_1_pred_f = model_1.predict(start=15788, end=19734)
arma_1_f_error = df_y_f_n.Appliances - arma_1_pred_f.values



plt.figure()
bound = 1.96/np.sqrt(len(arma_1_t_error))
ACF_Plot(autocorr(arma_1_t_error,90),90)
plt.xlabel('Lags')
plt.ylabel('ACF')
plt.title('ARMA (3,0) ACF Plot Predicting Electricity (Wh) with 90 Lags')
plt.axhspan(-bound,bound,alpha = .1, color = 'black')
plt.tight_layout()
plt.show()


plt.figure()
bound = 1.96/np.sqrt(len(arma_1_f_error))
ACF_Plot(autocorr(arma_1_f_error,90),90)
plt.xlabel('Lags')
plt.ylabel('ACF')
```

```
plt.title('ARMA (3,0) ACF Plot Forecasting Electricity (Wh) with 90 Lags')
plt.axhspan(-bound,bound,alpha = .1, color = 'black')
plt.tight_layout()
plt.show()

# train data
print("Mean square error for the ARMA(3,0) method forecasting on Electricity
(Wh) is\n ", mse(arma_1_t_error).round(4))
print(sm.stats.acorr_ljungbox(arma_1_t_error, lags=[5], boxpierce=True,
return_df=True))
print('The Q value was found to be 414.214595  with a p-value of 2.560013e-87
')
print('the mean for the ARMA(3,0) model error is\n', np.mean(arma_1_t_error))
print('the variance for the ARMA(3,0) model error is\n',
np.var(arma_1_t_error))
print("the covariance Matrix for the data is\n", model_1.cov_params())
print("The standard error coefficients are \n", model_1.bse)
print('The confidence intervals are\n', model_1.conf_int())
print('the RMSE for the ARMA(3,0) model error is\n ',
mean_squared_error(df_y_t_n['Appliances'], arma_1_pred_t.values,
squared=False))


# forecasting data
print('The MSE for the forecasting data was found to be',mse(arma_1_f_error))
print(sm.stats.acorr_ljungbox(arma_1_f_error, lags=[5], boxpierce=True,
return_df=True))
print('The Q value was found to be 4824.96647  with a p-value of 0.0 ')
print('the mean for the ARMA(3,0) model forecasting error is\n',
np.mean(arma_1_f_error))
print('the variance for the ARMA(3,0) model forecasting error is\n',
np.var(arma_1_f_error))
print('the RMSE for the ARMA(3,0) model error is\n ',
mean_squared_error(df_y_f_n['Appliances'], arma_1_pred_f.values,
squared=False))
print('The variance of the prediction error was less than the variance of the
forecasting error')

plt.figure()
plt.title('ARMA(3,0) for Electricity (Wh)')
plt.plot(df_y_t_n.index, df_y_t_n.Appliances, color= 'red', label='Train
Data')
plt.plot(df_y_f_n.index, df_y_f_n.Appliances, color= 'green', label='Test
Data')
plt.plot(df_y_t_n.index, arma_1_pred_t, color= 'yellow', label='Prediction
Data')
plt.plot(df_y_f_n.index, arma_1_pred_f, color= 'blue', label='Forecasting
Data')
plt.xticks(df.index[::4500], fontsize= 10)
plt.ylabel('Electricity (Wh)')
plt.xlabel('Time')
plt.legend()
plt.tight_layout()
plt.show()


plt.figure()
```

```
plt.title('Forecasting on ARMA(3,0) for Electricity (Wh)')
plt.plot(df_y_f_n.index, df_y_f_n.Appliances, color= 'red', label='Test
Data')
plt.plot(df_y_f_n.index, arma_1_pred_f, color= 'yellow', label='Forecasting
Data')
plt.xticks(df_y_f_n.index[::750], fontsize= 10)
plt.ylabel('Electricity (Wh)')
plt.xlabel('Time')
plt.legend()
plt.tight_layout()
plt.show()



# test for ARMA(3,1)
na = 3
nb = 1
# ARMA model
model_2 = sm.tsa.ARMA(df_y_t_n.Appliances,(na, nb)).fit(trend='nc',disp=0)
print(model_2.summary())


# training calcualtion
arma_2_pred_t = model_2.predict(start=0, end=15787)
arma_2_error = df_y_t_n.Appliances - arma_2_pred_t.values

# forecasting calculation
arma_2_pred_f = model_2.predict(start=15788, end=19734)
arma_2_f_error = df_y_f_n.Appliances - arma_2_pred_f.values


# acf plot of data
# train
plt.figure()
bound = 1.96/np.sqrt(len(arma_2_error))
ACF_Plot(autocorr(arma_2_error,90),90)
plt.xlabel('Lags')
plt.ylabel('ACF')
plt.title('ARMA (3,1) ACF Plot of Electricity (Wh) with 90 Lags')
plt.axhspan(-bound,bound,alpha = .1, color = 'black')
plt.tight_layout()
plt.show()

# test
plt.figure()
bound = 1.96/np.sqrt(len(arma_2_f_error))
ACF_Plot(autocorr(arma_2_f_error,90),90)
plt.xlabel('Lags')
plt.ylabel('ACF')
plt.title('ARMA (3,1) ACF Plot Forecasting Electricity (Wh) with 90 Lags')
plt.axhspan(-bound,bound,alpha = .1, color = 'black')
plt.tight_layout()
plt.show()

# training data
print("Mean square error for the ARMA(3,1) method prediction on Electricity
```

```
(Wh) is\n ", mse(arma_2_error).round(4))
print(sm.stats.acorr_ljungbox(arma_2_error, lags=[5], boxpierce=True,
return_df=True))
print('The Q value was found to be 24.179986 with a p-value of 0.0002')
print('the mean for the ARMA(3,1) model prediction error is\n',
np.mean(arma_2_error))
print('the variance for the ARMA(3,1) model prediction error is\n',
np.var(arma_2_error))
print("the covariance Matrix for the data is\n", model_2.cov_params())
print('the standard error coefficients are', model_2.bse)
print('The confidence intervals are\n', model_2.conf_int())
print('the RMSE for the ARMA(3,1) model error is\n ',
mean_squared_error(df_y_t_n['Appliances'], arma_2_pred_t.values,
squared=False))

# testing data
print('The MSE for the forecasting data was found to be',mse(arma_2_f_error))
print(sm.stats.acorr_ljungbox(arma_2_f_error, lags=[5], boxpierce=True,
return_df=True))
print('The Q value was found to be 4809.263056  with a p-value of 0.0 ')
print('the mean for the ARMA(3,1) model forecasting error is\n',
np.mean(arma_2_f_error))
print('the variance for the ARMA(3,1) model forecasting error is\n',
np.var(arma_2_f_error))
print('the RMSE for the ARMA(3,1) model error is\n ',
mean_squared_error(df_y_f_n['Appliances'], arma_2_pred_f.values,
squared=False))
print('The variance of the prediction error was less than the variance of the
forecasting error')

plt.figure()
plt.title('ARMA(3,1) for Electricity (Wh)')
plt.plot(df_y_t_n.index, df_y_t_n.Appliances, color= 'red', label='Train
Data')
plt.plot(df_y_f_n.index, df_y_f_n.Appliances, color= 'green', label='Test
Data')
plt.plot(df_y_t_n.index, arma_2_pred_t, color= 'yellow', label='Prediction
Data')
plt.plot(df_y_f_n.index, arma_2_pred_f, color= 'blue', label='Forecasting
Data')
plt.xticks(df.index[::4500], fontsize= 10)
plt.ylabel('Electricity (Wh)')
plt.xlabel('Time')
plt.legend()
plt.tight_layout()
plt.show()

plt.figure()
plt.title('Forecasting on ARMA(3,1) for Electricity (Wh)')
plt.plot(df_y_f_n.index, df_y_f_n.Appliances, color= 'red', label='Test
Data')
plt.plot(df_y_f_n.index, arma_2_pred_f, color= 'yellow', label='Forecasting
Data')
plt.xticks(df_y_f_n.index[::750], fontsize= 10)
plt.ylabel('Electricity (Wh)')
plt.xlabel('Time')
plt.legend()
```

```
plt.tight_layout()
plt.show()


# test for ARMA(3,3)
na = 3
nb = 3
# ARMA model
model_3 = sm.tsa.ARMA(df_y_t_n.Appliances,(na, nb)).fit(trend='nc',disp=0)
print(model_3.summary())

# training calcualtion
arma_3_pred_t = model_3.predict(start=0, end=15787)
arma_3_error = df_y_t_n.Appliances - arma_3_pred_t.values

# forecasting calculation
arma_3_pred_f = model_3.predict(start=15788, end=19734)
arma_3_f_error = df_y_f_n.Appliances - arma_3_pred_f.values


#plot of data
plt.figure()
bound = 1.96/np.sqrt(len(arma_3_error))
ACF_Plot(autocorr(arma_3_error,90),90)
plt.xlabel('Lags')
plt.ylabel('ACF')
plt.title('ARMA(3,3) ACF Forecasting Electricity (Wh) with 90 Lags')
plt.axhspan(-bound,bound,alpha = .1, color = 'black')
plt.tight_layout()
plt.show()

# test
plt.figure()
bound = 1.96/np.sqrt(len(arma_3_f_error))
ACF_Plot(autocorr(arma_3_f_error,90),90)
plt.xlabel('Lags')
plt.ylabel('ACF')
plt.title('ARMA (3,3) ACF Plot Forecasting Electricity (Wh) with 90 Lags')
plt.axhspan(-bound,bound,alpha = .1, color = 'black')
plt.tight_layout()
plt.show()



# statistics
print("Mean square error for the ARMA(3,3) method forecasting on Electricity
(Wh) is\n ", mse(arma_3_error).round(4))
print(sm.stats.acorr_ljungbox(arma_3_error, lags=[5], boxpierce=True,
return_df=True))
print('The Q value was found to be 0.637048 with a p-value of 0.0')
print('the mean for the ARMA(3,3) model error is\n', np.mean(arma_3_error))
print('the variance for the ARMA(3,3) model error is\n',
np.var(arma_3_error))
print("the covariance Matrix for the data is\n", model_3.cov_params())
print('The standard error coefficient are', model_3.bse)
print('The confidence intervals are\n', model_3.conf_int())
print('the RMSE for the ARMA(3,3) model error is\n ',
```

```
mean_squared_error(df_y_t_n['Appliances'], arma_3_pred_t.values,
squared=False))

print('The MSE for the forecasting data was found to be',mse(arma_3_f_error))
print(sm.stats.acorr_ljungbox(arma_3_f_error, lags=[5], boxpierce=True,
return_df=True))
print('The Q value was found to be 4820.005256 with a p-value of 0.0 ')
print('the mean for the ARMA(3,3) model forecasting error is\n',
np.mean(arma_3_f_error))
print('the variance for the ARMA(3,3) model forecasting error is\n',
np.var(arma_3_f_error))
print('the RMSE for the ARMA(3,3) model error is\n ',
mean_squared_error(df_y_f_n['Appliances'], arma_3_pred_f.values,
squared=False))
print('The variance of the prediction error was less than the variance of the
forecasting error')


plt.figure()
plt.title('ARMA(3,3) for Electricity (Wh)')
plt.plot(df_y_t_n.index, df_y_t_n.Appliances, color= 'red', label='Train
Data')
plt.plot(df_y_f_n.index, df_y_f_n.Appliances, color= 'green', label='Test
Data')
plt.plot(df_y_t_n.index, arma_3_pred_t, color= 'yellow', label='Prediction
Data')
plt.plot(df_y_f_n.index, arma_3_pred_f, color= 'blue', label='Forecasting
Data')
plt.xticks(df.index[::4500], fontsize= 10)
plt.ylabel('Electricity (Wh)')
plt.xlabel('Time')
plt.legend()
plt.tight_layout()
plt.show()

plt.figure()
plt.title('Forecasting on ARMA(3,3) for Electricity (Wh)')
plt.plot(df_y_f_n.index, df_y_f_n.Appliances, color= 'red', label='Test
Data')
plt.plot(df_y_f_n.index, arma_3_pred_f, color= 'yellow', label='Forecasting
Data')
plt.xticks(df_y_f_n.index[::750], fontsize= 10)
plt.ylabel('Electricity (Wh)')
plt.xlabel('Time')
plt.legend()
plt.tight_layout()
plt.show()




###############
# SARIMA model
###############


model_s = sm.tsa.statespace.SARIMAX(df_y_t_n.Appliances, order=(3,0,0),
```

```
seasonal_order=(0,3,0,12),
                                    enforce_stationarity=False,
                                    enforce_invertibility=False,)
res = model_s.fit()
print(res.summary())

#train data
ST_pred_t = res.get_prediction(start=pd.to_datetime('2016-01-11 17:00:00'),
end=pd.to_datetime('2016-04-30 08:10:00'), dynamic=False)
ST_pred = ST_pred_t.predicted_mean
pred_plot= ST_pred
ST_error = df_y_t_n.Appliances - ST_pred.values

# test data
#ST_pred_f = res.get_prediction(start=pd.to_datetime('2016-04-30 08:20:00'),
end=pd.to_datetime('2016-05-27 18:00:00 '), dynamic=False)
# ST_pred_f = res.get_forecast(steps=3947, index=df_y_f_n.index)
ST_pred_f = res.predict(start=0, end=(len(df_y_f_n['Appliances'])))
# ST_pred_f = ST_pred_f.predicted_mean
# pred_f_plot= ST_pred_f
ST_error_f = df_y_f_n.Appliances - ST_pred_f.values[1:]

# train
plt.figure()
bound = 1.96/np.sqrt(len(ST_error))
ACF_Plot(autocorr(ST_error,90),90)
plt.xlabel('Lags')
plt.ylabel('ACF')
plt.title('ARIMA(3,0,0)x(0,3,0,12) ACF Plot with 90 Lags')
plt.axhspan(-bound,bound,alpha = .1, color = 'black')
plt.tight_layout()
plt.show()

# test
plt.figure()
bound = 1.96/np.sqrt(len(ST_error_f))
ACF_Plot(autocorr(ST_error_f,90),90)
plt.xlabel('Lags')
plt.ylabel('ACF')
plt.title('ARIMA(3,0,0)x(0,3,0,12) ACF Plot Forecasting with 90 Lags')
plt.axhspan(-bound,bound,alpha = .1, color = 'black')
plt.tight_layout()
plt.show()


# training statistics
print("Mean square error for the  ARIMA(3,0,0)x(0,0,0,12) method forecasting
on Electricity (Wh) is\n ", mse(ST_error).round(4))
print(sm.stats.acorr_ljungbox(ST_error, lags=[5], boxpierce=True,
return_df=True))
print('The Q value was found to be 78.976724  with a p-value of 1.373662e-
15')
print('the mean for the  ARIMA(3,0,0)x(0,3,0,12) model error is\n',
np.mean(ST_error))
print('the variance for the  ARIMA(3,0,0)x(0,3,0,12) model error is\n',
np.var(ST_error))
print("the covariance Matrix for the data is\n", res.cov_params())
```

```
print("the Standard Error for the coefficients are is\n", res.bse)
print('The confidence intervals are\n', res.conf_int())
print('the RMSE for the ARIMA(3,0,0)x(0,3,0,12) model error is\n ',
mean_squared_error(df_y_t_n.Appliances, ST_pred.values, squared=False))

# testing statistics
print('The MSE for the forecasting data was found to be',mse(ST_error_f))
print(sm.stats.acorr_ljungbox(ST_error_f, lags=[5], boxpierce=True,
return_df=True))
print('The Q value was found to be 923.111417 with a p-value of 2.648581e-
197')
print('the mean for the ARIMA(3,0,0)x(0,3,0,12) model forecasting error
is\n', np.mean(ST_error_f))
print('the variance for the ARIMA(3,0,0)x(0,3,0,12) model forecasting error
is\n', np.var(ST_error_f))
print('the RMSE for the ARIMA(3,0,0)x(0,3,0,12) model error is\n ',
mean_squared_error(df_y_f_n['Appliances'], ST_pred_f[1:].values,
squared=False))
print('the variance of the prediction error appeared less than the variance
of the forecasting error')
# without forecasting
plt.figure()
plt.title('ARIMA(3,0,0)x(0,3,0,12) on Electricity (Wh)')
plt.xlabel('Date')
plt.ylabel('Electricity (Wh)')
plt.plot(df_y_t_n.index, pred_plot, color ='yellow', label='Prediction Data')
plt.plot(df_y_t_n.index, df_y_t_n.Appliances, color='red', label='Train
Data')
plt.plot(df_y_f_n.index, df_y_f_n.Appliances, color='green', label='Test
Data')
# plt.plot(df_y_f_n.index, pred_f_plot, color ='blue', label='Forecasting
Data')
plt.xticks(df_y.index[::4500], rotation= 90, fontsize= 10)
plt.legend()
plt.tight_layout()
plt.show()


# with forecasting
plt.figure()
plt.title('ARIMA(3,0,0)x(0,3,0,12) on Electricity (Wh)')
plt.xlabel('Date')
plt.ylabel('Electricity (Wh)')
plt.plot(df_y_t_n.index, pred_plot, color ='yellow', label='Prediction Data')
plt.plot(df_y_t_n.index, df_y_t_n.Appliances, color='red', label='Train
Data')
plt.plot(df_y_f_n.index, df_y_f_n.Appliances, color='green', label='Test
Data')
plt.plot(df_y_f_n.index, ST_pred_f[1:], color ='blue', label='Forecasting
Data')
plt.xticks(df_y.index[::4500], rotation= 90, fontsize= 10)
plt.legend()
plt.tight_layout()
plt.show()


#14: base models
```

```
def one_step_average_method(x):
    x_br = []
    for i in range(1,len(x)):
        m = np.mean(x[0:i])
        x_br.append(m)
    return x_br

one_step_average_method(df_y_t_n.Appliances)

def h_step_average_method(train, test):
    forecast = np.mean(train)
    predictions = []
    for i in range(len(test)):
        predictions.append(forecast)
    return predictions
H_avg= h_step_average_method(df_y_t_n['Appliances'],df_y_f_n['Appliances'])


# Q2 plot train, test, and average

plt.figure()
plt.plot(df_y_t_n.index,df_y_t_n.Appliances,label= "Train Data", color =
'blue')
plt.xticks(df.index[::4500], rotation= 90, fontsize= 10)
plt.plot(df_y_f_n.index,df_y_f_n.Appliances,label= "Test Data", color =
'red')
plt.plot(df_y_f_n.index, H_avg, label = 'Average Method', color = 'yellow')
plt.xlabel('Time')
plt.ylabel('Electricity (Wh)')
plt.title('Average Method on Electricity (Wh)')
plt.legend()
plt.tight_layout()
plt.show()

plt.figure()
plt.plot(df_y_f_n.index,df_y_f_n.Appliances,label= "Test Data", color =
'red')
plt.plot(df_y_f_n.index, H_avg, label = 'Average Method', color = 'yellow')
plt.xticks(df_y_f_n.index[::725], fontsize= 10)
plt.xlabel('Time')
plt.ylabel('Electricity (Wh)')
plt.title('Average Method on Electricity (Wh)')
plt.legend()
plt.tight_layout()
plt.show()



#train
avg_yt_error = np.array(df_y_t_n.Appliances[1:]) -
np.array(one_step_average_method(df_y_t_n.Appliances))
print("Mean square error for the average method training set is ",
mse(avg_yt_error).round(4))
#forecast
avg_yf_error = np.array(df_y_f_n.Appliances) - np.array(H_avg)
print("Mean square error for the average method testing set is ",
mse(avg_yf_error).round(4))
```

```
# Average method statistics
print('the variance of the error of the average method training set is ',
np.var(avg_yt_error))
print('the variance of the error of the average method testing set is ',
np.var(avg_yf_error))
print('the RMSE of the Average forecasting model error is, ',
mean_squared_error(df_y_f_n['Appliances'], np.array(H_avg), squared=False))
print('the mean of the Average forecasting  model error is',
np.mean(avg_yf_error))
print(sm.stats.acorr_ljungbox(avg_yf_error, lags=[5], boxpierce=True,
return_df=True))
print('The Q value was found to be 5014.178759 with a p-value of 0.0')
print('the varaince of hte prediction error appeared larger than the variance
of the forecasting error')

plt.figure()
bound = 1.96/np.sqrt(len(df_y_t_n.Appliances))
ACF_Plot(autocorr(avg_yf_error,90),90)
plt.xlabel('Lags')
plt.ylabel('ACF')
plt.title('ACF Plot of Average Method Electricity (Wh) with 90 Lags')
plt.axhspan(-bound,bound,alpha = .1, color = 'black')
plt.tight_layout()
plt.show()


# naive method
def one_step_naive_method(x):
    forecast = []
    for i in range(len(x)-1):
        forecast.append(x[i])
    return forecast

def h_step_naive_method(test,train):
    forecast = [test[-1] for i in range (len(train))]
    return forecast
N_avg= h_step_naive_method(df_y_t_n.Appliances,df_y_f_n.Appliances)



plt.figure()
plt.plot(df_y_t_n.index,df_y_t_n.Appliances,label= "Train Data", color =
'blue')
plt.plot(df_y_f_n.index, df_y_f_n.Appliances,label= "Test Data", color =
'red')
plt.plot(df_y_f_n.index, N_avg, label = 'Naive Method', color = 'yellow')
plt.xticks(df.index[::4500], rotation= 90, fontsize= 10)
plt.xlabel('Time')
plt.ylabel('Electricity (Wh)')
plt.title('Naive Method on Electricity (Wh)')
plt.legend()
plt.tight_layout()
plt.show()

plt.figure()
plt.plot(df_y_f_n.index, df_y_f_n.Appliances,label= "Test Data", color =
```

```
'red')
plt.plot(df_y_f_n.index, N_avg, label = 'Naive Method', color = 'yellow')
plt.xticks(df_y_f_n.index[::725], fontsize= 10)
plt.xlabel('Time')
plt.ylabel('Electricity (Wh)')
plt.title('Naive Method on Electricity (Wh)')
plt.legend()
plt.tight_layout()
plt.show()


#train
N_yt_error = np.array(df_y_t_n.Appliances[1:]) -
np.array(one_step_naive_method(df_y_t_n.Appliances))
print("Mean square error for the Naive method training set is ",
mse(N_yt_error))
#forecast
N_yf_error = np.array(df_y_f_n.Appliances) - np.array(N_avg)
print("Mean square error for the Naive method testing set is ",
mse(N_yf_error))

# Q6d: Naive method statistics
print('the Variance of the error of the Naive method training set is ',
np.var(N_yt_error))
print('the Variance of the error of the Naive method testing set is ',
np.var(N_yf_error))
print('the RMSE of the Naive model forecasting error is, ',
mean_squared_error(df_y_f_n['Appliances'], np.array(N_avg), squared=False))
print('the mean of the Naive model forecasting error is',
np.mean(N_yf_error))
print(sm.stats.acorr_ljungbox(N_yf_error, lags=[5], boxpierce=True,
return_df=True))
print('The Q value was found to be 5014.178759  with a p-value of 0.0')
print('the variance for the prediction error appeared less than the variance
of the forecasting error')

plt.figure()
bound = 1.96/np.sqrt(len(df_y_t_n.Appliances))
ACF_Plot(autocorr(N_yf_error,90),90)
plt.xlabel('Lags')
plt.ylabel('ACF')
plt.title('ACF Plot of Naive Method Electricity (Wh) with 90 Lags')
plt.axhspan(-bound,bound,alpha = .1, color = 'black')
plt.tight_layout()
plt.show()

# drift method

def one_step_drift_method(x):
    forecast =[]
    for i in range(1,len(x)-1):
        #slope calculation
        prediction = x[i]+(x[i]-x[0])/i
        forecast.append(prediction)
    #gives you every prediction except first bc first is first value of
dataset
    forecast = [x[0]] + forecast
```

```
    return forecast

one_step_drift_method(df_y_t_n.Appliances)

def h_step_drift_method(train,test):
    forecast = []
    # use first and last number in train to calcuate slope for h step
    prediction = (train[-1] - train[0]) / (len(train)-1)
    for i in range(1,len(test) + 1):
        forecast.append(train[-1]+ i*prediction)
    return forecast
H_drift= h_step_drift_method(df_y_t_n.Appliances,df_y_f_n.Appliances)

# Q7b: plot train, test, and drift

plt.figure()
plt.plot(df_y_t_n.index,df_y_t_n.Appliances,label= "Train Data", color =
'blue')
plt.plot(df_y_f_n.index, df_y_f_n.Appliances,label= "Test Data", color =
'red')
plt.plot(df_y_f_n.index, H_drift, label = 'Drift Method', color = 'yellow')
plt.xticks(df.index[::4500], rotation= 90, fontsize= 10)
plt.xlabel('Time')
plt.ylabel('Electricity (Wh)')
plt.title('Drift Method on Electricity (Wh)')
plt.legend()
plt.tight_layout()
plt.show()

plt.figure()
plt.plot(df_y_f_n.index, df_y_f_n.Appliances,label= "Test Data", color =
'red')
plt.plot(df_y_f_n.index, H_drift, label = 'Drift Method', color = 'yellow')
plt.xticks(df_y_f_n.index[::725], fontsize= 10)
plt.xlabel('Time')
plt.ylabel('Electricity (Wh)')
plt.title('Drift Method on Electricity (Wh)')
plt.legend()
plt.tight_layout()
plt.show()



#train
drift_yt_error = np.array(df_y_t_n.Appliances[1:]) -
np.array(one_step_drift_method(df_y_t_n.Appliances))
print("Mean square error for the drift method training set is ",
mse(drift_yt_error))
#forecast
drift_yf_error = np.array(df_y_f_n.Appliances) - np.array(H_drift)
print("Mean square error for the drift method testing set is ",
mse(drift_yf_error))

# Q7d: drift method variance
# Q6d: Naive method statistics
print('the Variance of the error of the Drift method training set is ',
np.var(drift_yt_error))
```

```
print('the Variance of the error of the Drift method testing set is ',
np.var(drift_yf_error))
print('the RMSE of the Drift model forecasting error is, ',
mean_squared_error(df_y_f_n['Appliances'], np.array(H_drift), squared=False))
print('the mean of the Drift model forecasting error is',
np.mean(drift_yf_error))
print(sm.stats.acorr_ljungbox(drift_yf_error, lags=[5], boxpierce=True,
return_df=True))
print('The Q value was found to be 5129.25267 with a p-value of 0.0')
print('the variance for the prediction error appeared less than the variance
of the forecasting error')

plt.figure()
bound = 1.96/np.sqrt(len(df_y_t_n.Appliances))
ACF_Plot(autocorr(drift_yf_error,90),90)
plt.xlabel('Lags')
plt.ylabel('ACF')
plt.title('ACF Plot of Drift Method Electricity (Wh) with 90 Lags')
plt.axhspan(-bound,bound,alpha = .1, color = 'black')
plt.tight_layout()
plt.show()


# Seasonal exponential smoothing

# Q8a: see report

holtt =
ets.ExponentialSmoothing(df_y_t_n.Appliances,trend=None,damped_trend=False,se
asonal=None).fit(smoothing_level=0.5)
holtf = holtt.forecast(steps=len(df_y_f_n))
holtf = pd.DataFrame(holtf)


#plot train, test, and SES

plt.figure()
plt.plot(df_y_t_n.index,df_y_t_n.Appliances,label= "Train Data", color =
'blue')
plt.plot(df_y_f_n.index, df_y_f_n.Appliances,label= "Test Data", color =
'red')
plt.plot(df_y_f_n.index, np.array(holtf), label = 'SES Method', color =
'yellow')
plt.xticks(df.index[::4500], rotation= 90, fontsize= 10)
plt.xlabel('Date')
plt.ylabel('Appliances (Wh)')
plt.title('SES Method on Appliances(Wh)')
plt.legend()
plt.tight_layout()
plt.show()

plt.figure()
plt.plot(df_y_f_n.index, df_y_f_n.Appliances,label= "Test Data", color =
'red')
plt.plot(df_y_f_n.index, np.array(holtf), label = 'SES Method', color =
'yellow')
plt.xticks(df_y_f_n.index[::725], fontsize= 10)
```

```
plt.xlabel('Date')
plt.ylabel('Appliances (Wh)')
plt.title('SES Method on Appliances(Wh)')
plt.legend()
plt.tight_layout()
plt.show()


#train
def SES_train(yt,alpha, initial=430):
    prediction = [initial]
    for i in range(1,len(yt)):
        s= alpha*yt[i-1] + (1-alpha)*prediction[i-1]
        prediction.append(s)
    return prediction

SES_yt_error = np.array(df_y_t_n.Appliances) -
SES_train(df_y_t_n.Appliances,.5)
print("Mean square error for the SES method training set is ",
mse(SES_yt_error))
#forecast
holtf = holtt.forecast(steps=len(df_y_f_n.Appliances))
holtf = pd.DataFrame(holtf)
SES_yf_error = np.array(df_y_f_n.Appliances) - holtf[0]
print("the mean square error for the SES method testing set is ",
mse(SES_yf_error))
print('the Variance of the error of the SES method training set is ',
np.var(SES_yt_error))
print('the Variance of the error of the SES method testing set is ',
np.var(SES_yf_error))
print('the RMSE of the SES model forecasting error is, ',
mean_squared_error(df_y_f_n['Appliances'], holtf[0], squared=False))
print('the mean of the SES model forecasting error is',
np.mean(SES_yf_error))
print(sm.stats.acorr_ljungbox(SES_yf_error, lags=[5], boxpierce=True,
return_df=True))
print('The Q value was found to be 5014.178759 with a p-value of 0.0')
print('The variance of the prediction error appeared less than the variance
of the forecasting error')

plt.figure()
bound = 1.96/np.sqrt(len(df_y_t_n.Appliances))
ACF_Plot(autocorr(SES_yf_error,90),90)
plt.xlabel('Lags')
plt.ylabel('ACF')
plt.title('ACF Plot of SES Method Electricity (Wh) with 90 Lags')
plt.axhspan(-bound,bound,alpha = .1, color = 'black')
plt.tight_layout()
plt.show()
```

Resources:

Candanedo, L. M. (n.d.). *LuisM78/Appliances-energy-prediction-data*. GitHub. https://github.com/LuisM78/Appliances-energy-prediction-data.

Singh, A. (2019, May 18). Appliances_energy_prediction_LSTM_ARIMA_FBPROPHET. Retrieved April 5, 2021, from https://www.kaggle.com/adityasingh22/appliances-energy-prediction-lstm-arima-fbprophet