DATS 6313_10: TIME SERIES ANALYSIS AND MODELING

Instructor: Reza Jafari

Term Project

Varun .R. Shah
VRS
05/03/2022

Table of Contents:

Abstract

Introduction

Description of the Dataset

Stationarity

Time-Series Decomposition

Forecasting Methods
- **Base Models:**
  - **Holt-Winter Method**
  - **Average Method**
  - **Naïve Method**
  - **Drift Method**
  - **Simple Exponential Smoothing Method**
- **Multiple Linear Regression**
  - **Feature Selection**
  - **OLS Regression Model**
- **ARIMA/SARIMA Model:**

Summary

Conclusion

Abstract:

This project consists of Time-Series Analysis of Bike Sharing Demand over time. In this project, we will first understand the data and achieve stationarity by using various transformation methods. We then use Time-Series Decomposition methods to test if the data is completely detrended and seasonality has been adjusted. This is done by using the STL-Decomposition Method. We then begin by using simple forecasting methods on our original data, including the base models such as Holt-Winter Method, Average Method, Naïve Method, Drift and Simple Exponential Smoothing Method. We then use the OLS Regression Method, here we first check the collinearity among the features and then use Backward Stepwise regression for Feature Selection and to get our best model. We then develop an ARIMA model on the transformed data, we first determine the order of the AR and MA processes using ACF/PACF Plots and GPAC, and then estimate the parameters for the model. We evaluate all these models using the Mean-Squared Error of Forecast error, Variance of errors, Q-Value and Correlation Coefficients. We also plot the ACF Plots for each residual errors to see if the residuals fall under white noise.

Introduction:

The data consists of hourly bike rental data over a period of 2 years. It includes casual bike rentals and registered bike rentals. For simplicity, we look at the total bike rentals (casual + registered combined). The data includes season, weather, temperature, humidity and 12 other features that may impact the bike rentals every hour. We will be cleaning the data and then use different methods to remove seasonality and trend from the data and bring stationarity in the data, and transform it accordingly. We will be then applying different forecasting methods to pick the best model that provides us with better forecasting of the bike rentals over time.

The dataset is sourced from Kaggle Competition: https://www.kaggle.com/competitions/bike-sharing-demand/data

Description of the Dataset:

The dataset includes 10886 observations, with 9 features. The features are as follows:

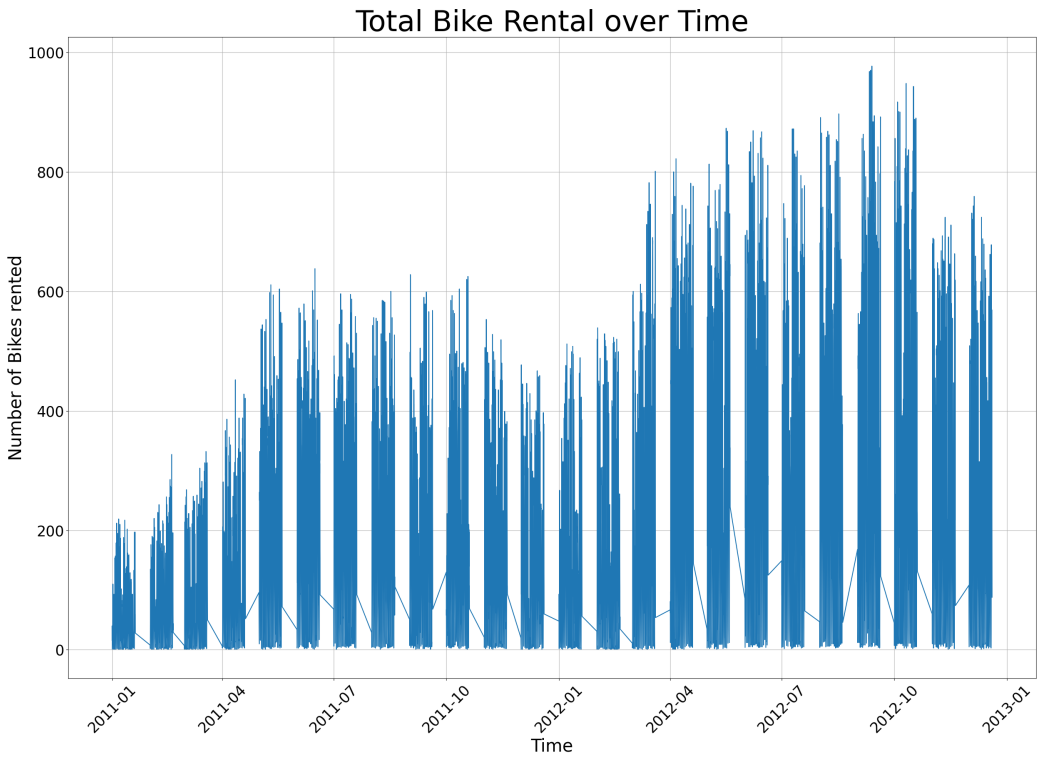
Columns: 'datetime', 'season', 'holiday', 'workingday', 'weather', 'temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered', 'count']

The datetime is an hourly data over a period of two years. It includes seasons such as “spring”, “summer”, “fall” and “winter. It also includes features such as “holiday” and “workingday” to label if the day is a holiday or a weekday. It includes metrics such as temperatures, humidity and windspeed. It has the count of casual bike rentals and registered users along with total count.

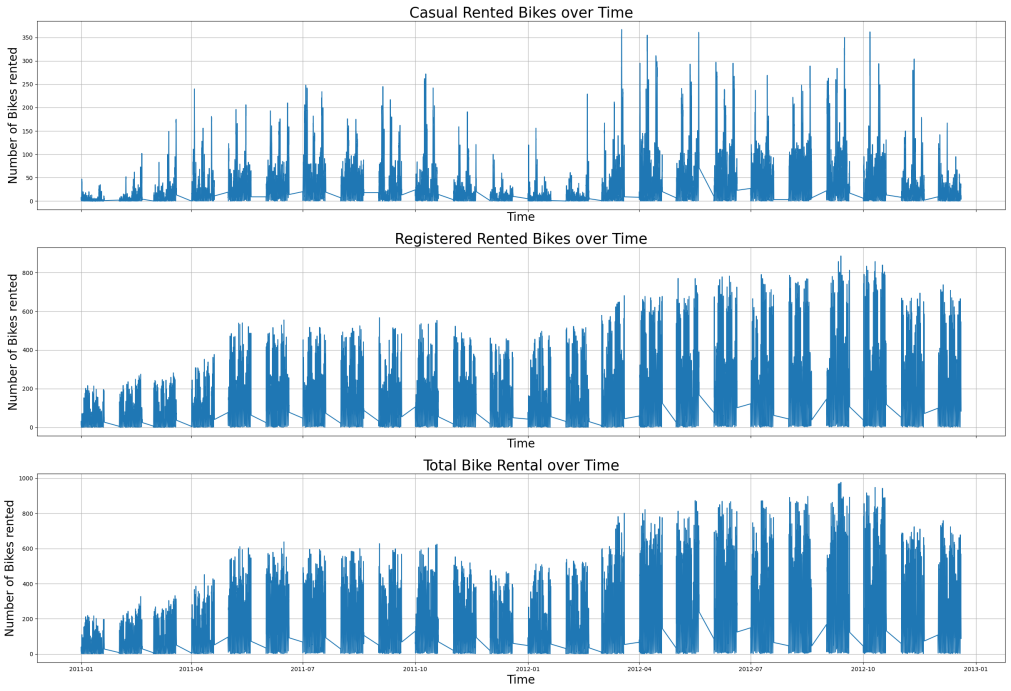
The description are as follows:

Features	***Description***
datetime	Hourly Data with timestamp
season	1: spring, 2: summer, 3:fall, 4:winter
holiday	Whether the day is considered holiday
workingday	Whether the day is neither the weekend or holiday
weather	1: Clear/Partly Cloudy/Few Clouds. 2: Mist + Cloudy, Mist+Few Clouds 3: Light Snow, Light rain + Thunderstorm, Light rain + scattered clouds 4: Heavy rain + ice Pallets + Thunderstorm + Mist/fog, Snow + fog
temp	Temperature in Celsius
atemp	"feels like" temperature in Celsius
humidity	relative humidity
windspeed	wind speed
casual	Number of non-registered user rentals initiated
registered	Number of registered user rentals initiated
count	Number of Total rentals.

Total Bikes Rented over Time:

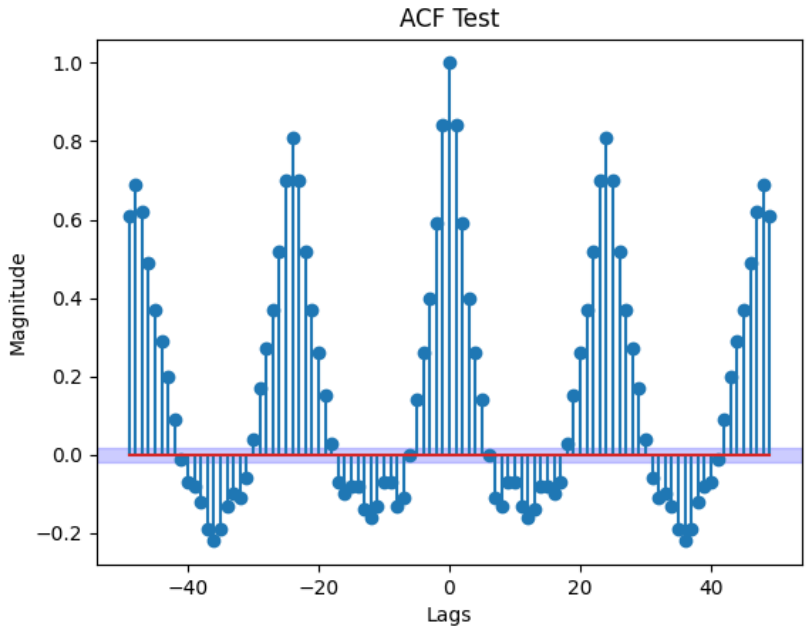


The above shows the hourly data of bike rentals, over a period of two years. The below shows the bike rental data over time for Casual, Registered and combined rentals over time.



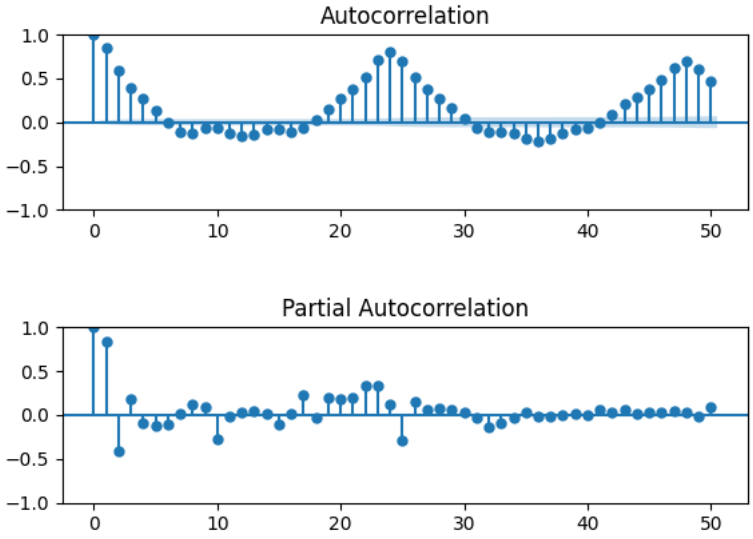
For simplicity, we will only consider the total bikes rented over Time (combining the casual and registered users).

ACF Plot:



The above shows the ACF dropping gradually as the lag increases, showing non-stationarity in the data. The periodic decrease is followed by a gradual increase, indicating seasonality in the data.

ACF/PACF Plot:

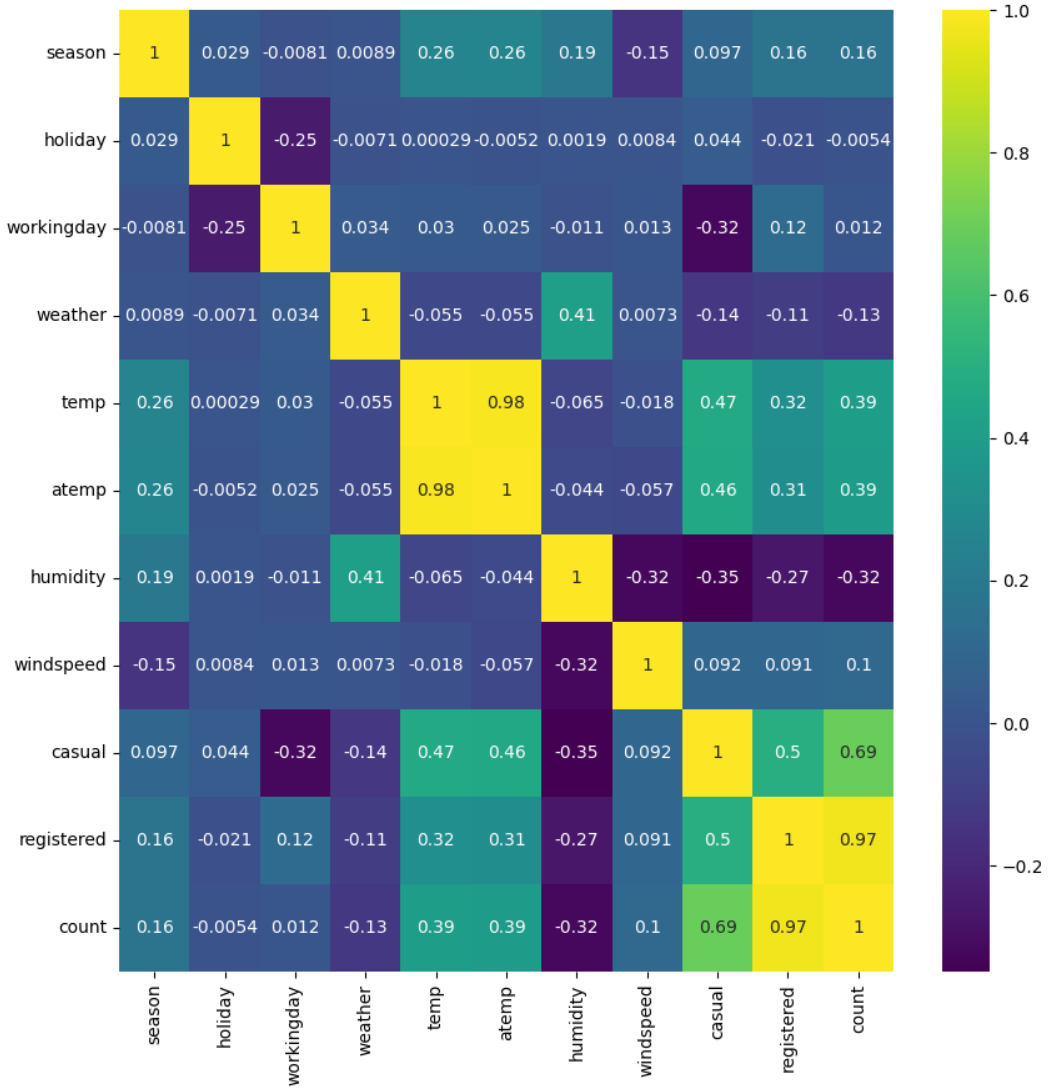


The above shows ACF tails-off slowly, and again increasing after lag 24. This shows non-stationary data. The PACF cuts-off at lag 3, showing a non-seasonal component and we observe the spike after lag 20. We observe a seasonality in the data approximately at lag 24.

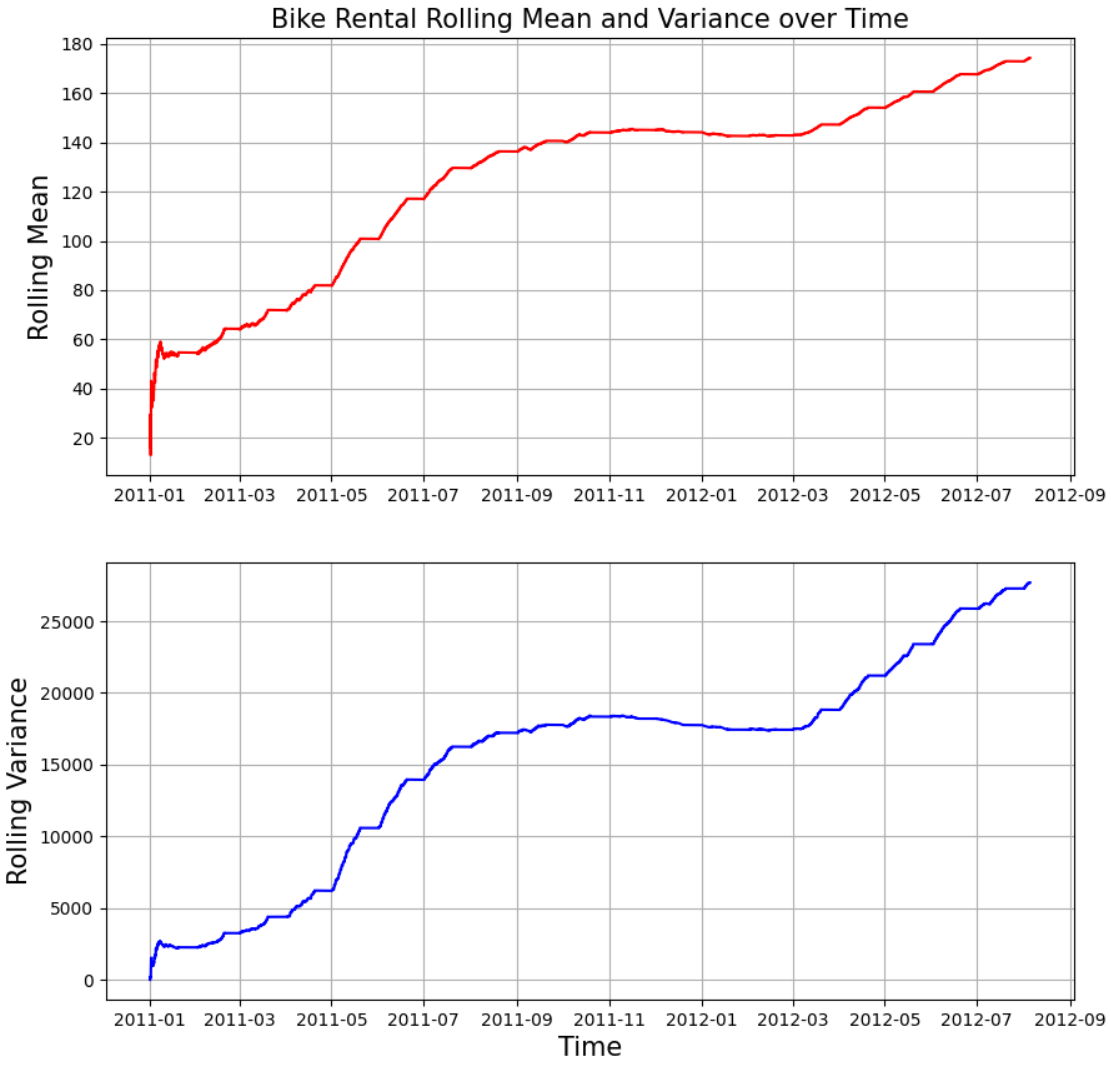
Correlation Matrix:

The correlation matrix shows the correlation coefficients between all the features. Each cell in the table shows the correlation between two features.

From the table, we can see that “humidity” is negatively correlated with rental count, and temperature is positively correlated with the count. This means that the rentals increase as the temperature increases. We also notice, the casual renters significantly increase as the temperature increases, compared to the already registered users.



Stationarity:



The above shows an upward trend in rolling mean and variance, with a brief period of stationary data. We notice the trend is upwards from January to September, it stays stationary until March of the following year followed by trending upwards.

ADF Test:

In *ADF-Test,* the null and alternate hypothesis are as follows:

Null Hypothesis: The series has a unit root (it is non-stationary)
Alternate Hypothesis: The series is no unit root (it is stationary)

ADF Statistic: -5.611623
p-value: 0.000001
Critical Values:
1%: -3.431
5%: -2.862
10%: -2.567

Based on the above, we see that the p-value is less than α at 0.05. Hence, we are able to the null hypothesis and adopt the alternate hypothesis and state that the data is stationary.

KPSS Test:

In *KPSS-Test,* the null and alternate hypothesis are opposite of that of ADF-Test. They are as follows:

Null Hypothesis: The series trends stationary
Alternate Hypothesis: The series has a unit root (it is non-stationary)

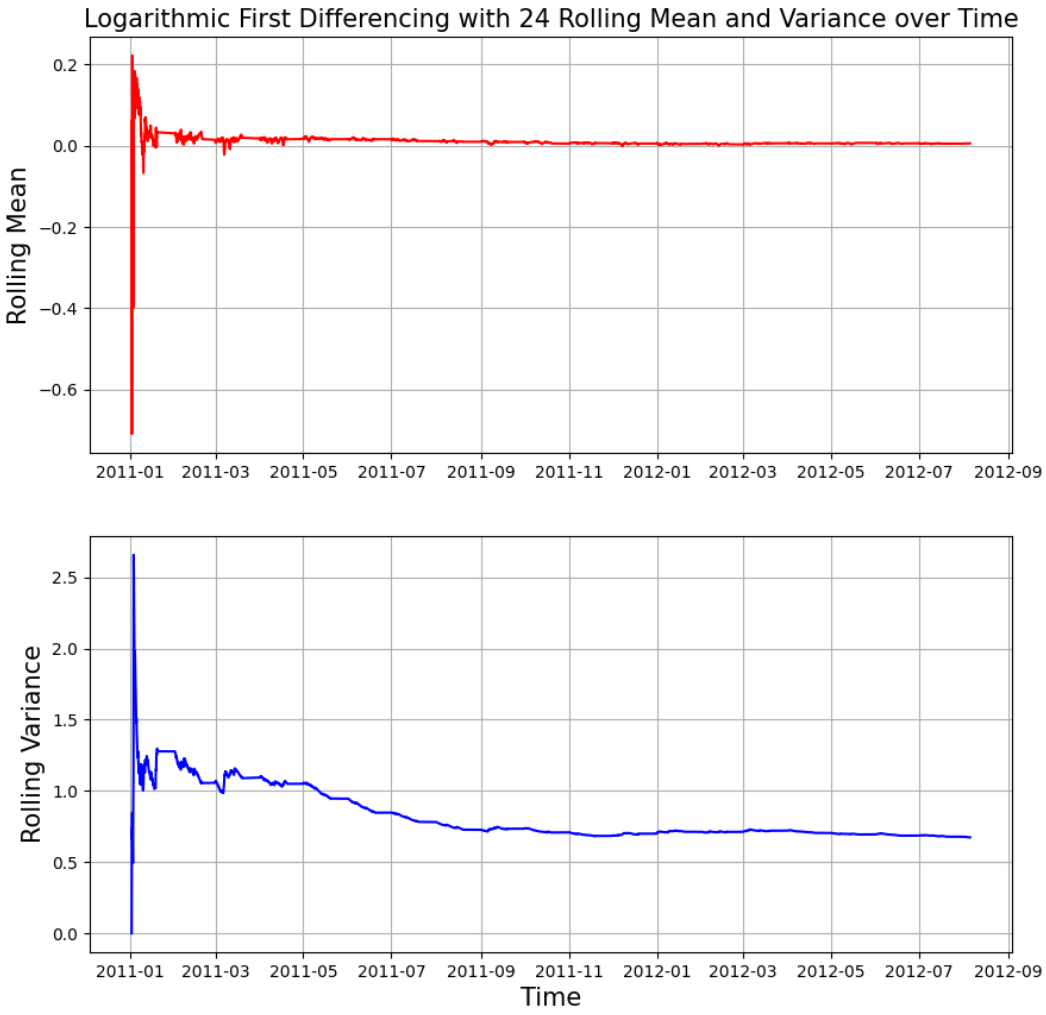
Results of KPSS Test:
Test Statistic 11.798041
p-value 0.010000
Lags Used 33.000000
Critical Value (10%) 0.347000
Critical Value (5%) 0.463000
Critical Value (2.5%) 0.574000
Critical Value (1%) 0.739000

Based on the above, we see that that the p-value is less than α at 0.05. Hence, here we reject the Null Hypothesis and can state that the data is non-stationary.

Stationarity by Logarithmic Differencing:

Logarithmic First Differencing with Seasonality: 24

Based on the above plots, we notice the seasonality is at 24 (hourly data at 24-hour cycles).



ADF Test:

ADF Statistic: -18.974139
p-value: 0.000000
Critical Values:
1%: -3.431
5%: -2.862
10%: -2.567

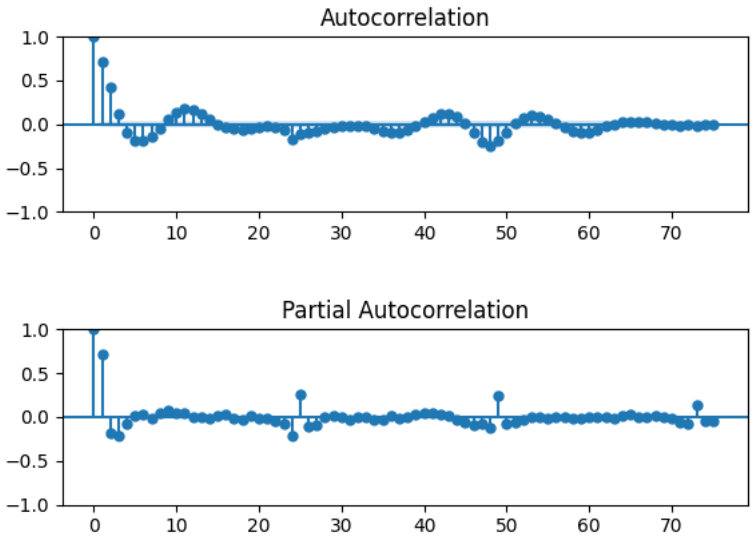
With p-value less than 0.05, we reject the Null and can state that the data is stationary.

KPSS Test:

Results of KPSS Test:
Test Statistic 0.010472
p-value 0.100000
Lags Used 27.000000
Critical Value (10%) 0.347000
Critical Value (5%) 0.463000
Critical Value (2.5%) 0.574000
Critical Value (1%) 0.739000

With p-value at 0.1, greater than 0.05, we are unable to reject the Null and state the data is stationary.

ACF/PACF Plot:



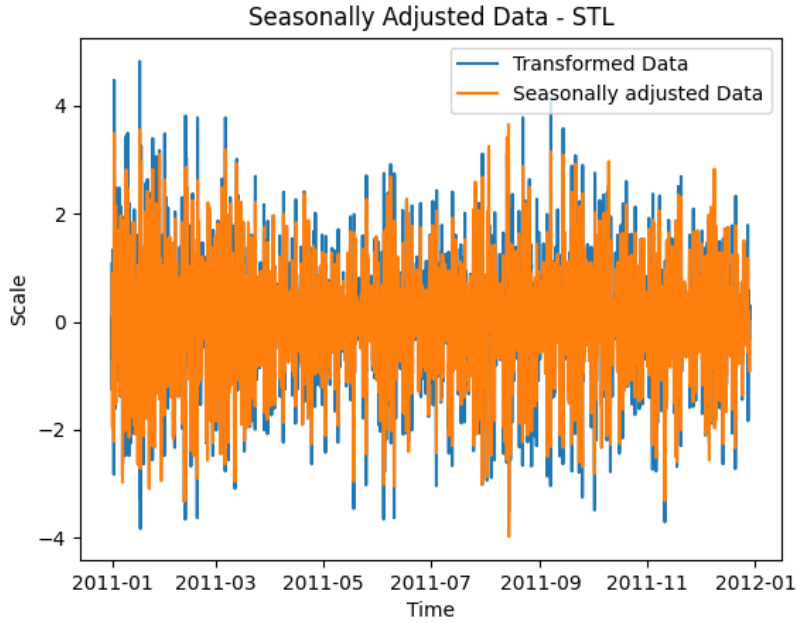
The above shows a decay after lag-2 across seasonal lags of 24, 48, and so on in ACF. The significant spike at lag 1 in PACF suggests a non-seasonal AR(1) component, the significant spike at ACF until lag 2 suggests a non-seasonal MA(2) component.

The ACF also shows a seasonal spike at lag 24 and at 48 which is achieved by seasonal differencing at 24.

Hence, our potential model looks to be: ARIMA(1, 0, 2) x $(0, 1, 2)_{24}$

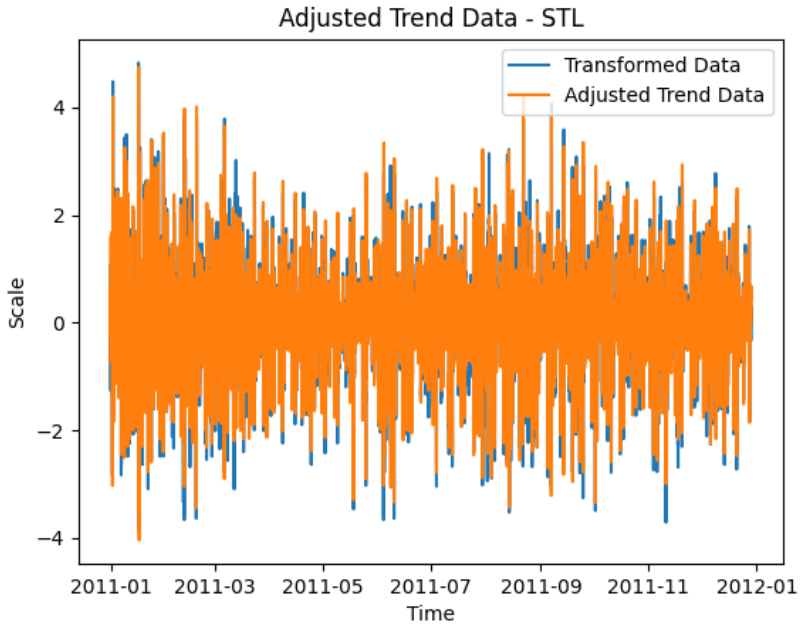
Time-Series Decomposition:

STL-Decomposition of the Transformed Data:



Strength of Seasonality for this data set is: 0.211

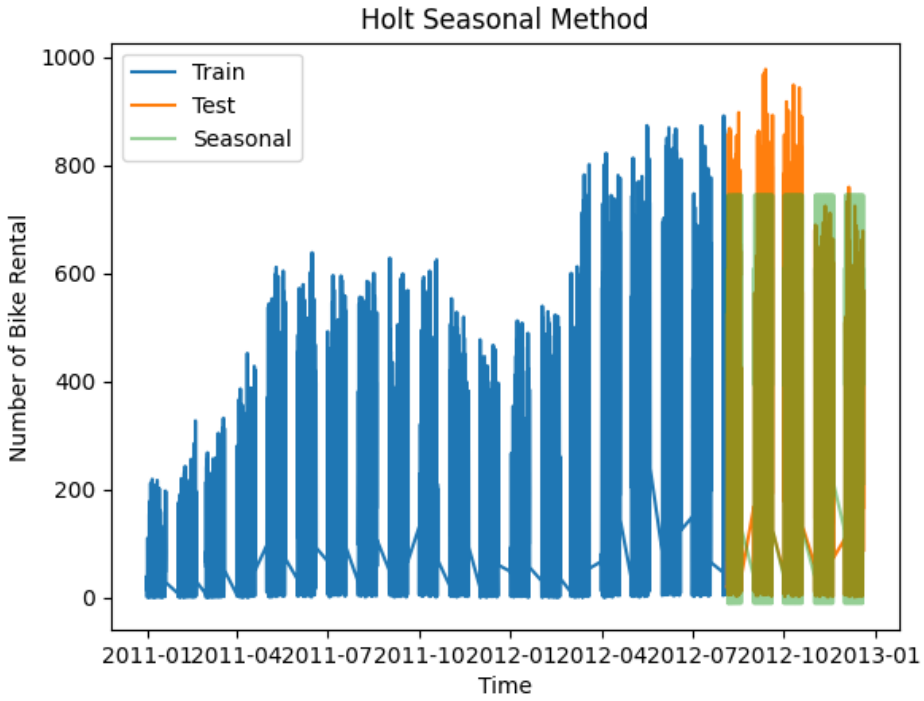
This is very close to zero, hence the data has low seasonality.



Strength of Trend for this data set is: 0.18

This is very close to zero. Hence the transformed data is almost detrended.

Holt-Winter Method (On Original Data):



Mean Squared Error:

MSE of Forecast Errors with Holt Seasonal Method: 25715.75

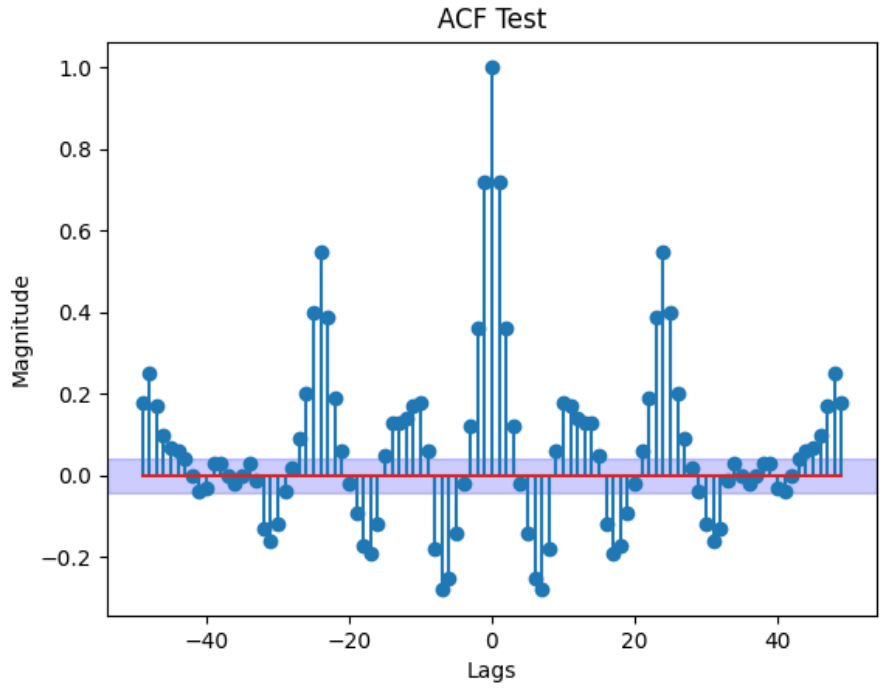
Variance of Errors:

Variance of Prediction Error with Holt Seasonal Method 5540.46

Variance of Forecast Error with Holt Seasonal Method 21878.98

The Ratio of Variance is: 0.25

ACF Plot of the residuals from Holt-Winter Seasonal Method:



The above ACF Plot for the forecast errors shows the residuals are autocorrelated, and not in white-noise completely.

Q-Value:

The Q value (Box-Pierce Test) with Holt Seasonal, for the prediction error is: 8494.35

Correlation Coefficient between forecast error and Test set:

The Correlation Coefficient with Holt Seasonal Method is 0.304

As per above, we see that correlation coefficient between forecast error and test set is positive. Meaning as the Test set values increase, the forecast error increases. Note, the degree is much smaller, implying prediction values with smaller forecast errors.

Feature Selection:

Here, we begin by selecting the below features as X:

'season', 'holiday', 'workingday', 'weather', 'temp', 'atemp', 'humidity', 'windspeed'

and we drop the ‘datetime’, ‘casual’ and ‘registered’.

Singular Values of X:

Singular Values of X are:

[8.77810699e+04 5.78387294e+03 2.11047884e+03 9.13134656e+02
4.52371380e+02 2.80022234e+02 2.12880432e+02 1.10645833e+02
2.24097174e+00]

Based on the singular values, we can state that there is collinearity between at-least 4 features. This is based on the lowest values from SVD, which are much closer to zero compared to the top 5 features.

Condition Number:

The condition Number for X is 197.92

The Condition Number is between 100 and 1000, which suggests Moderate to Strong Degree of Collinearity.

Model Summary using OLS Function:

Full Model:

```
    ....
                            OLS Regression Results
==============================================================================
Dep. Variable:                  count   R-squared:                       0.279
Model:                            OLS   Adj. R-squared:                  0.278
Method:                 Least Squares   F-statistic:                     421.0
Date:                Tue, 03 May 2022   Prob (F-statistic):               0.00
Time:                        16:54:17   Log-Likelihood:                -55469.
No. Observations:                8708   AIC:                         1.110e+05
Df Residuals:                    8699   BIC:                         1.110e+05
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        133.5425      8.809     15.159      0.000     116.274     150.811
season         2.5354      1.622      1.563      0.118      -0.643       5.714
holiday      -10.3852      9.605     -1.081      0.280     -29.213       8.443
workingday    -2.9481      3.366     -0.876      0.381      -9.547       3.650
weather        1.4625      2.662      0.549      0.583      -3.755       6.680
temp         -84.4448     66.100     -1.278      0.201    -214.016      45.126
atemp        435.3437     67.553      6.444      0.000     302.923     567.764
humidity    -255.7644      9.272    -27.586      0.000    -273.939    -237.590
windspeed     44.6485     11.632      3.838      0.000      21.846      67.451
==============================================================================
Omnibus:                     1625.545   Durbin-Watson:                   0.434
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3100.036
Skew:                           1.148   Prob(JB):                         0.00
Kurtosis:                       4.809   Cond. No.                         198.
```

In the above summary with all the features included, we see the Adjusted R^2 at 0.278, AIC and BIC at 11,100 (highlighted) and a Condition Number at 198, showing the moderate to high degree of collinearity.

We use Backward stepwise regression here, hence we include all the features at the beginning and eliminate the features one by one.

Feature Selection/Elimination:

We observe the p-values of the features and eliminate the features that has the highest p-value. The features that have p-values less than 0.05 are the ones that are statistically significant for our model.

We begin by removing the feature **'weather'** and get the below result:

OLS Regression Results

Dep. Variable:	count	R-squared:	0.279
Model:	OLS	Adj. R-squared:	0.278
Method:	Least Squares	F-statistic:	481.1
Date:	Tue, 03 May 2022	Prob (F-statistic):	0.00
Time:	16:56:12	Log-Likelihood:	-55469.
No. Observations:	8708	AIC:	1.110e+05
Df Residuals:	8700	BIC:	1.110e+05
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	134.1663	8.736	15.359	0.000	117.042	151.290
season	2.4574	1.615	1.521	0.128	-0.709	5.624
holiday	-10.3378	9.604	-1.076	0.282	-29.165	8.489
workingday	-2.8643	3.363	-0.852	0.394	-9.456	3.727
temp	-83.4531	66.073	-1.263	0.207	-212.971	46.065
atemp	434.3690	67.527	6.432	0.000	302.000	566.738
humidity	-253.5166	8.320	-30.471	0.000	-269.826	-237.208
windspeed	45.5280	11.521	3.952	0.000	22.944	68.112

Omnibus:	1623.617	Durbin-Watson:	0.433
Prob(Omnibus):	0.000	Jarque-Bera (JB):	3092.651
Skew:	1.147	Prob(JB):	0.00
Kurtosis:	4.805	Cond. No.	178.

Here, we see no change in Adjusted R^2, and some reduction in AIC and BIC. We repeat the process of elimination one by one, again by looking at the highest p-values.

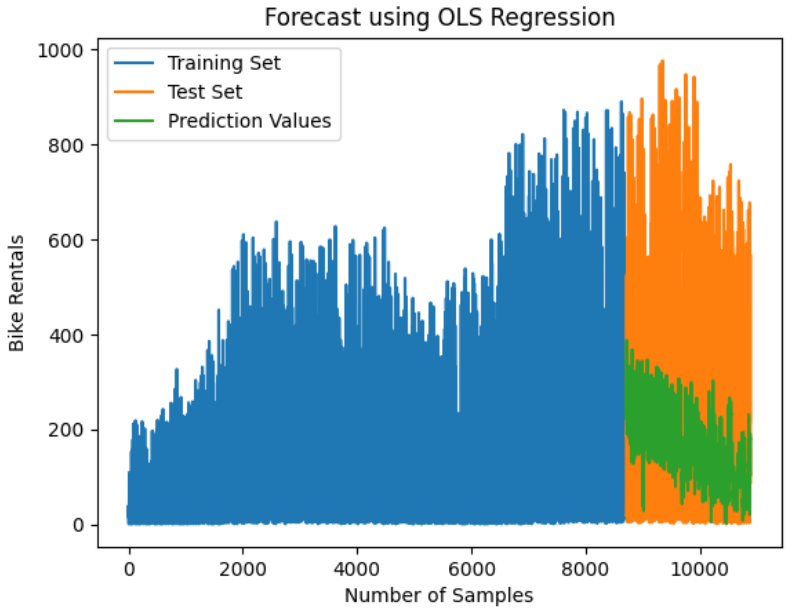
We remove **'workingday** as the next feature which has a p-value of 0.394, followed by **'holiday'**, **'temp'**, **'season'**. We get the below after removing these features one by one.

Final Model:

```
...
                            OLS Regression Results
==============================================================================
Dep. Variable:                  count   R-squared:                       0.279
Model:                            OLS   Adj. R-squared:                  0.278
Method:                 Least Squares   F-statistic:                     1121.
Date:                Tue, 03 May 2022   Prob (F-statistic):               0.00
Time:                        16:57:31   Log-Likelihood:                -55472.
No. Observations:                8708   AIC:                         1.110e+05
Df Residuals:                    8704   BIC:                         1.110e+05
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        137.1536      7.840     17.495      0.000     121.786     152.521
atemp        354.7454      7.800     45.483      0.000     339.456     370.034
humidity    -250.1324      8.131    -30.764      0.000    -266.070    -234.194
windspeed     39.9204     10.991      3.632      0.000      18.374      61.466
==============================================================================
Omnibus:                     1608.768   Durbin-Watson:                   0.432
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3038.855
Skew:                           1.142   Prob(JB):                         0.00
Kurtosis:                       4.777   Cond. No.                         11.5
==============================================================================
```

From the above, we finalize on the **'constant'** + the 3 features, ***'atemp***, ***'humidity'*** and ***'windspeed'*** as statistically significant for our model with p-values less than 0.05. Our final model has the largest Adjusted R^2 achieved at 0.278, with AIC and BIC constant at to 11,100. The Condition Number is reduced to 11.5, which is still very low degree of collinearity. Note, any further elimination of features results in reduction of Adjusted R^2, negatively impacting our model and we do not remove the constant as it is the term for the intercept in our linear regression model.

Plotting Training set, Test set and the prediction values:



Mean-Squared-Error of Forecast Error:

MSE of Forecast Errors with OLS Method: 46058.63

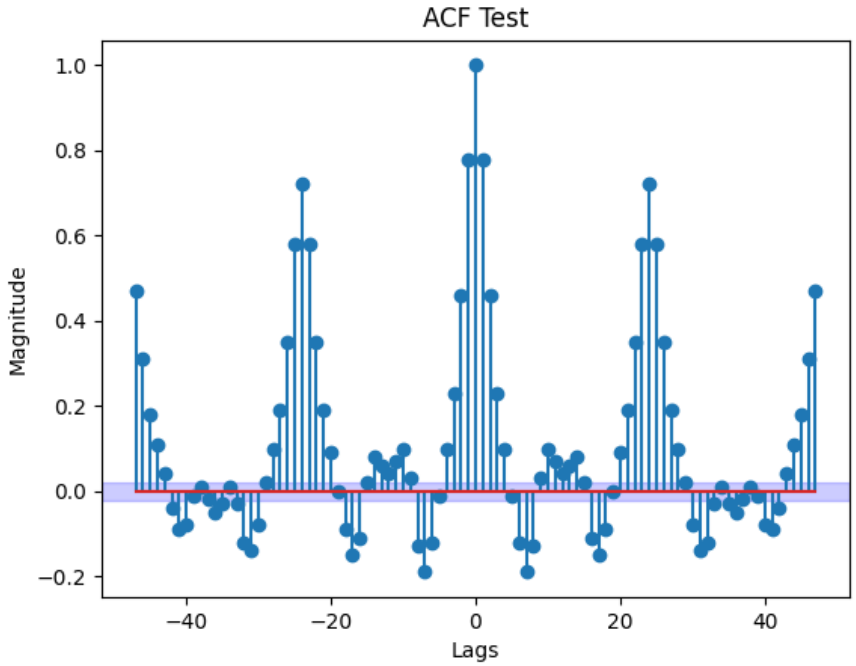
Variance of Errors:

The Estimated Variance for Prediction Errors is: 141.4

The Estimated Variance for Forecast Errors is: 214.86

The Variance Ratio is: 0.65

ACF Plot for the Residual Errors:



The above ACF plot suggests that the residual errors are highly autocorrelated.

Q-Value:

The Q value (Box-Pierce Test) with OLS Method, for the prediction error is: 32357.55

Correlation Coefficient between forecast error and Test set:

The Correlation Coefficient with OLS Method is 0.941

As per above, we see that correlation coefficient between forecast error and test set is positive and highly correlated. Meaning as the Test set values increase, the forecast error increases. The degree is very high, implying prediction values with larger forecast errors.

T-Test:

We use the T-test to conduct Hypothesis tests on the regression coefficients of our final model. Here, the Null and Alternate Hypothesis are as follows:

$$H_0 : \beta_i = 0$$
$$H_a : \beta_i \neq 0$$

Null Hypothesis: the slope of the coefficients is equal to zero.
Alternate Hypothesis: the slope of the coefficients is not equal to zero

X(Predictors)	Coefficients	Std. Error	t-values	p-values:
const	137.153618	7.917	16.561453	2.215712e-67
atemp	354.745419	0.175	45.482724	0.000000e+00
humidity	-250.132382	0.081	-30.763904	2.227031e-197
windspeed	39.920403	0.193	3.631940	2.829160e-04

Since the p-value is less than 0.05, we reject the Null Hypothesis. This means that “atemp”, “humidity” and “windspeed” does have a statistically significant relationship with the total bike rentals.

F-Test:

For F-Test, our Null and Alternative Hypothesis are as follows:

Null hypothesis: The fit of the intercept-only model and our final model are equal.

Alternative hypothesis: The fit of the intercept-only model is significantly reduced compared to our final model.

F test: F=1120.573348035355, p=0.0, df_denom=8.7e+03, df_num=3

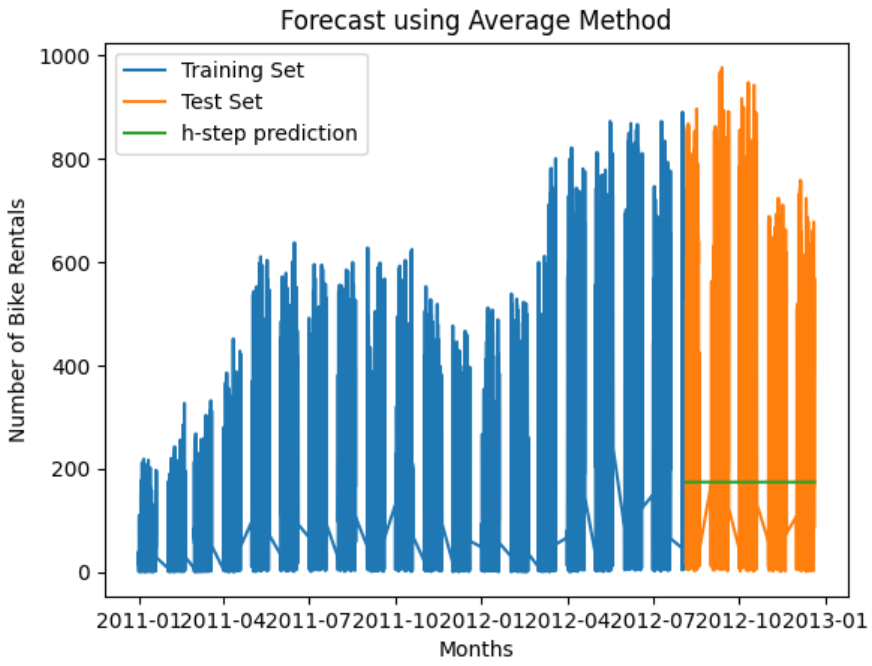
F_value: 1120.5733480353558

F_P-value: 0.0

The above shows the p-value less than 0.05, hence we can reject the Null Hypothesis (that the intercept and our final model are equal) and statistically adopt the Alternate Hypothesis, that the fit of the intercept-only model is significantly reduced compared to our final model.

Base models:

Average Method:



Mean Squared Error for Forecast Error:

MSE of Forecast Errors with Average Method: 54724.48

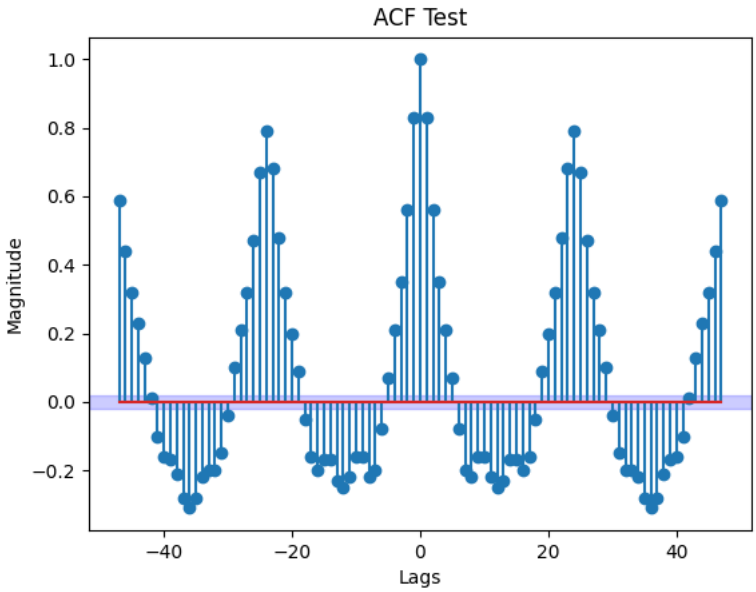
Variance of Errors:

Variance of Prediction Error with Average Method 25243.23

Variance of Forecast Error with Average Method 47348.4

The Variance Ratio is: 0.53

ACF Plot for Residual Error:



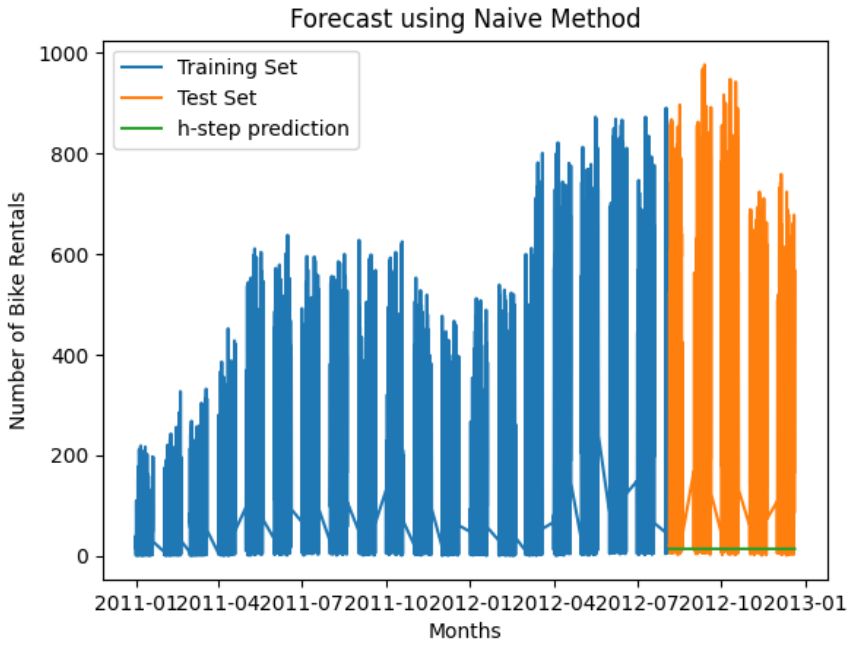
Q – Value:

The Q value (Box-Pierce Test) with Average Method, for the prediction error is: 55860.42

Correlation Coefficients for Forecast Errors and Test Set:

The Correlation Coefficient with Average Method is 0.999

Naïve Method:



Mean Squared Error for Forecast Error:

MSE of Forecast Errors with Naive Method: *107999.34*

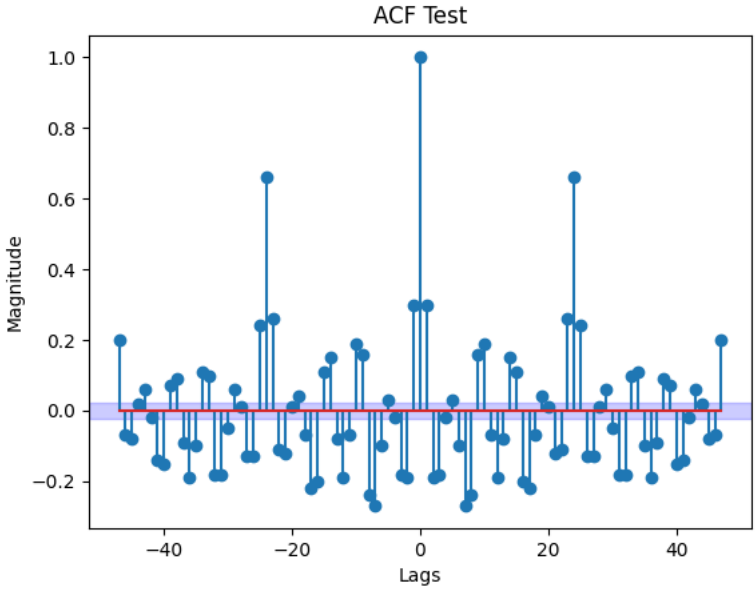
Variance of Errors:

Variance of Prediction Error with Naïve Method 8562.53

Variance of Forecast Error with Naïve Method 47348.4

The Variance Ratio is: 0.18

ACF Plot for Residual Errors:



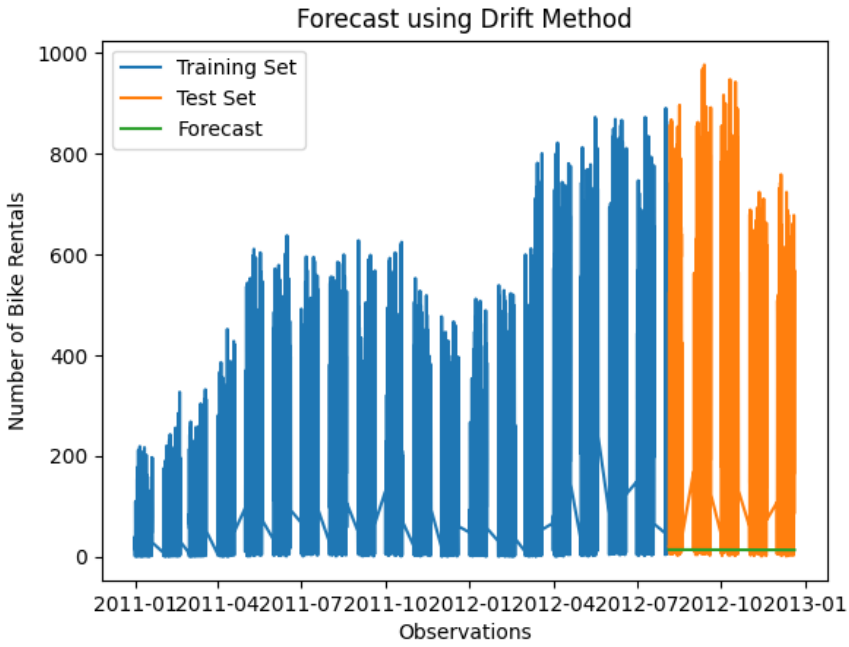
Q-Value:

The Q value (Box-Pierce Test) with Naive Method, for the prediction error is: 15438.53

Correlation Coefficients for Forecast Errors and Test Set:

The Correlation Coefficient with Naive Method is 0.99

Drift Method:



Mean Squared Error of Forecast Errors:

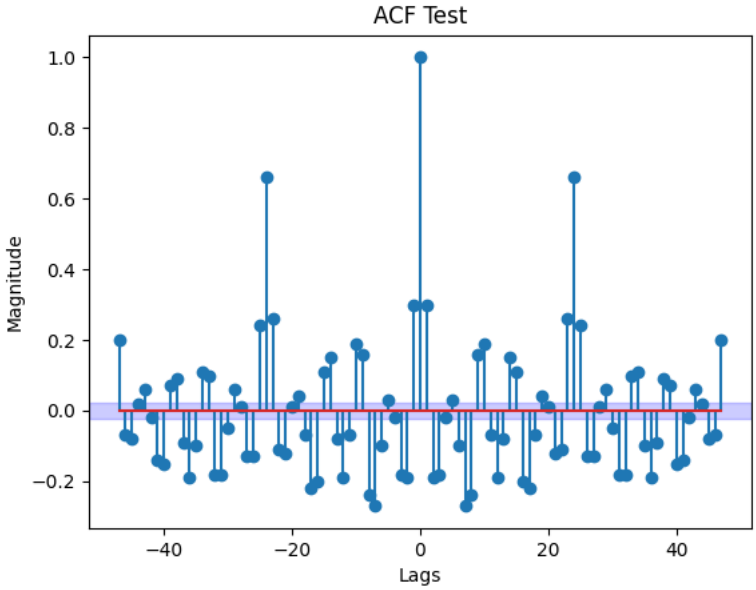
MSE of Forecast Errors with Drift Method: 108115.02

Variance of Errors:

Variance of Prediction Error with Drift Method 8566.1
Variance of Forecast Error with Drift Method 47340.76

The Variance Ratio is: 0.18

ACF Plot:



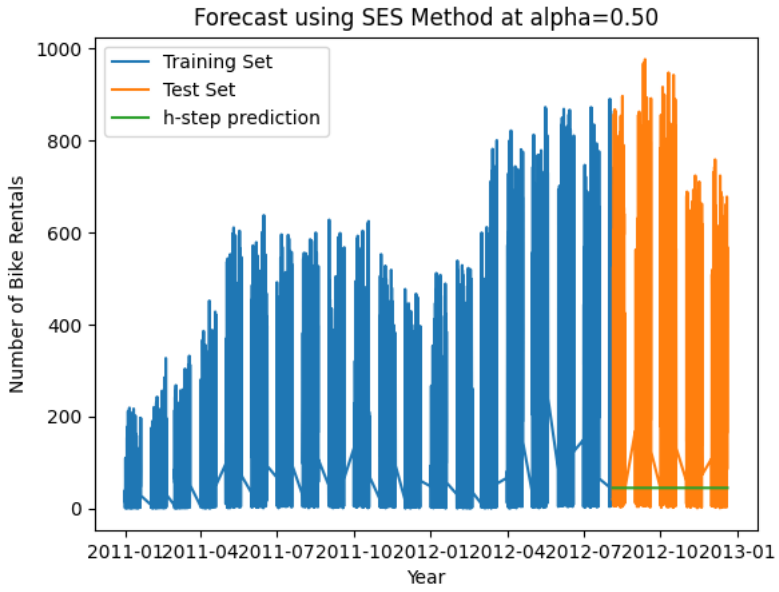
Q-Value:

The Q value (Box-Pierce Test) with Drift Method, for the prediction error is: 15438.53

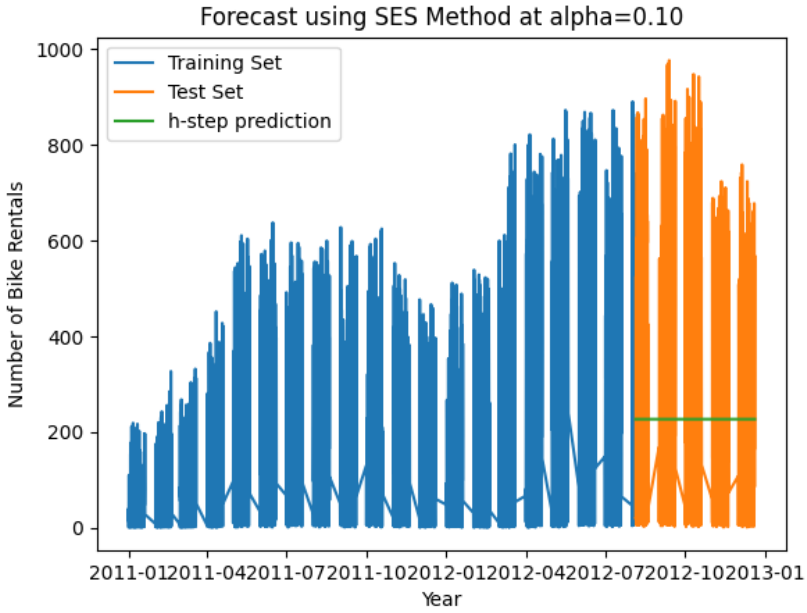
Correlation Coefficient of Forecast Errors with Test Set:

The Correlation Coefficient with Drift Method is 0.99

SES Method (at alpha: 0.50)



SES Method (at alpha: 0.10)



Mean Squared Error for forecast Errors:

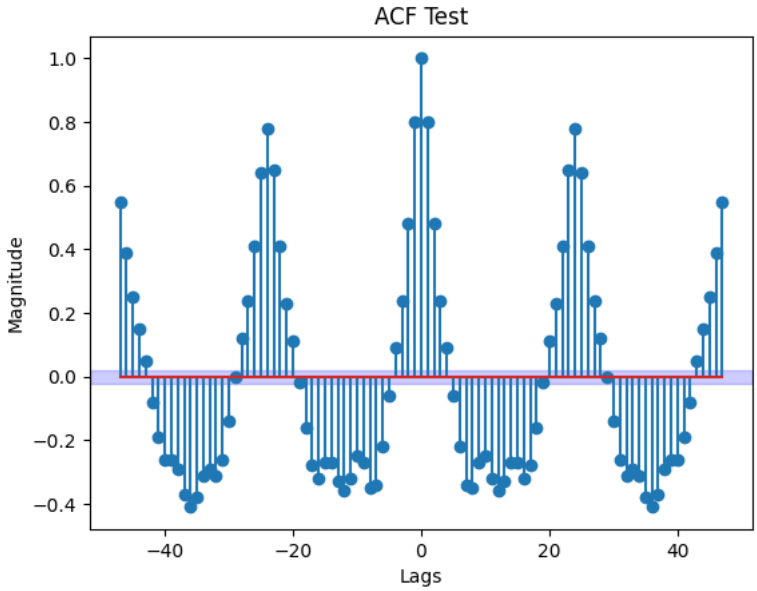
MSE of Forecast Errors with SES Method: 48456.9

Variance of Errors:

Variance of Prediction Error with SES Method 22835.57
Variance of Forecast Error with SES Method 47348.4

The Variance Ratio is: 0.48

ACF Plot for residual errors:



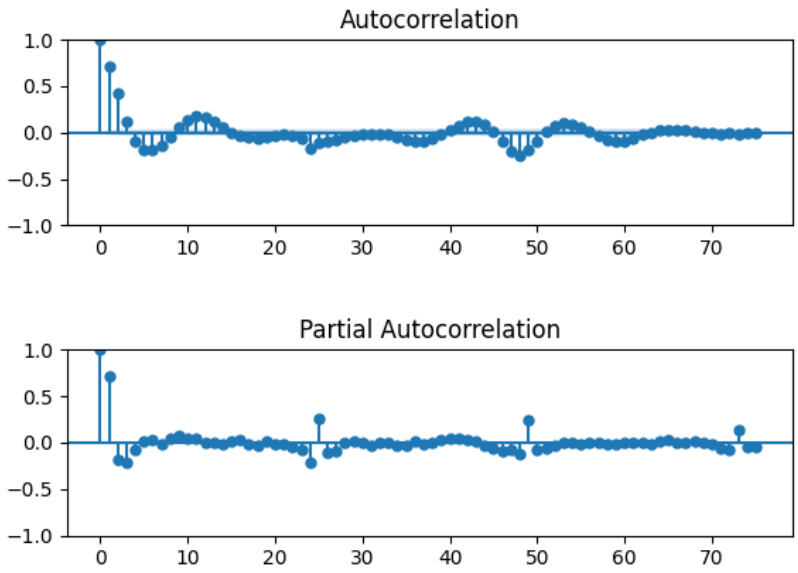
Q-Value:

The Q value (Box-Pierce Test) with SES Method (at alpha=0.10), for the prediction error is: 60804.84

Correlation Coefficient of forecast errors with test set:
The Correlation Coefficient with SES Method is 0.99

ARIMA/SARIMA Process:

Based on the transformed data we got when looking for stationarity, we now proceed with the ARIMA process we got in the beginning.



As indicated earlier, the above shows a decay after lag-2 across seasonal lags of 24, 48, and so on in PACF. The significant spike at lag 1 in PACF suggests a non-seasonal AR(1) component, the significant spike at ACF until lag 2 suggests a non-seasonal MA(2) component.

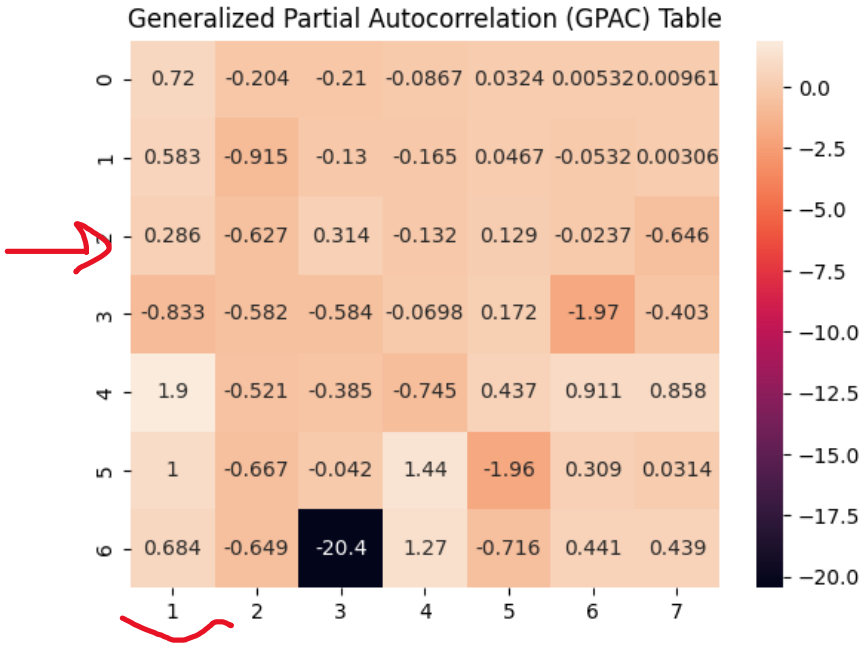
The ACF also shows a seasonal spike at lag 24 and at 48 which is achieved by seasonal differencing at 24.

ARIMA $(p, d, q)\ (P,D,Q)_s$
$p = 1$
$d = 0$
$q = 2$
$P = 0$
$D = 1$
$Q = 2$
$s = 24$

Hence, our potential model looks to be: ARIMA$(1, 0, 2) \times (0, 1, 2)_{24}$

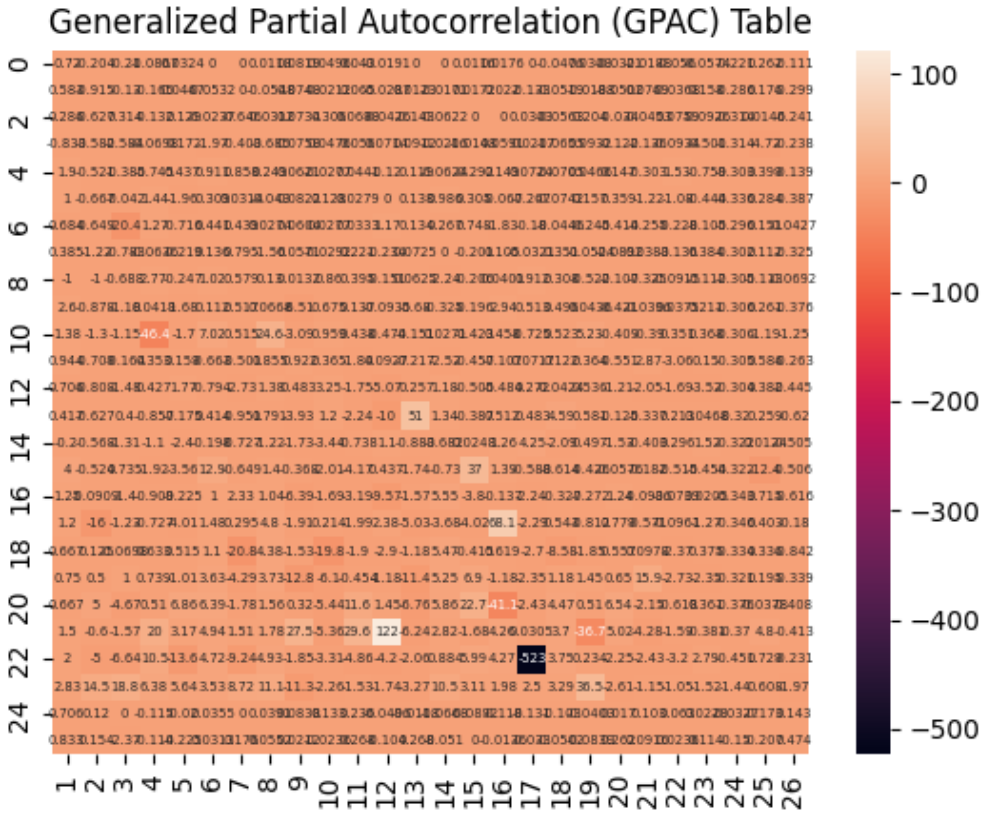
GPAC:

To check the non-seasonal component, we first plot a 7x7 GPAC:



The above GPAC shows the non-seasonal AR(1) and MA(2) component.

GPAC with 26 x 26 lags:



The above GPAC shows the seasonal component with MA of order 1 at lag 24. Because of its size, lag 48 cannot be plot.

Backshift equation:

ARIMA(1, 0, 2) x $(0, 1, 2)_{24}$

$(1 - a_1q^{-1})\ _y(t) = (1 - b_1q^{-1} - b_2q^{-2})\ (1 - b_{24}q^{-24} - b_{48}\text{-}q^{-48})e(t)$

SARIMA Model:

```
                                SARIMAX Results
==========================================================================================
Dep. Variable:                             count   No. Observations:                 8708
Model:             SARIMAX(1, 0, 2)x(0, 1, 2, 24)   Log Likelihood              -47390.533
Date:                            Tue, 03 May 2022   AIC                           94793.067
Time:                                    15:51:39   BIC                           94835.482
Sample:                                         0   HQIC                          94807.525
                                             - 8708
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.4362      0.018     24.510      0.000       0.401       0.471
ma.L1          0.5690      0.020     28.416      0.000       0.530       0.608
ma.L2          0.1482      0.018      8.045      0.000       0.112       0.184
ma.S.L24      -0.5440      0.006    -87.826      0.000      -0.556      -0.532
ma.S.L48      -0.2921      0.006    -46.281      0.000      -0.304      -0.280
sigma2      3208.3216     28.366    113.106      0.000    3152.726    3263.917
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):              8159.79
Prob(Q):                              0.98   Prob(JB):                         0.00
Heteroskedasticity (H):               2.30   Skew:                             0.01
Prob(H) (two-sided):                  0.00   Kurtosis:                         7.75
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

The estimated parameters are:

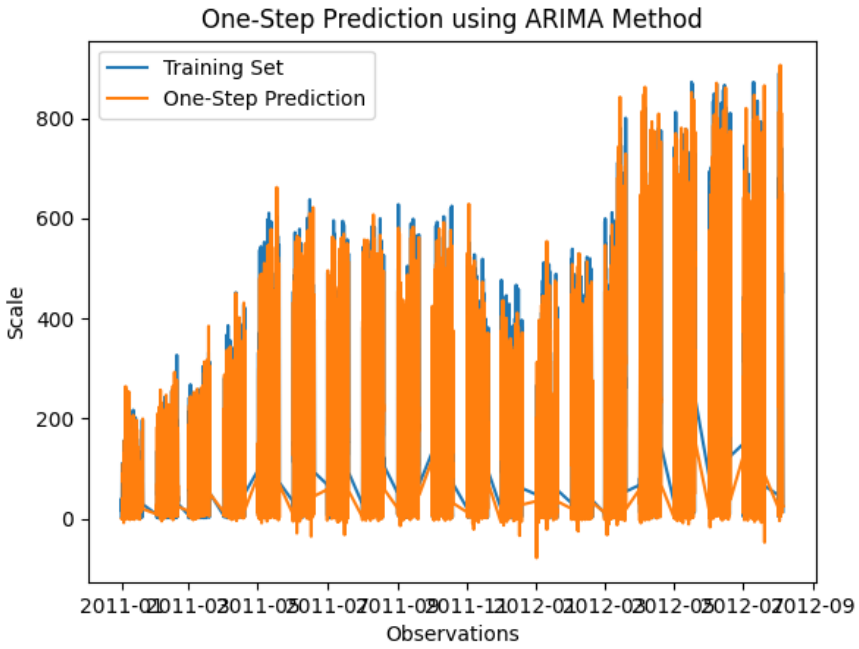
ar.L1	0.436238
ma.L1	0.569048
ma.L2	0.148204
ma.S.L24	-0.543976
ma.S.L48	-0.292072

The Confidence Interval is:

	0	1
ar.L1	0.401354	0.471123
ma.L1	0.529799	0.608297
ma.L2	0.112097	0.184311
ma.S.L24	-0.556116	-0.531837
ma.S.L48	-0.304441	-0.279703

The above Confidence Interval shows, all the parameters are significant as it does not pass 0.

One-Step Predictions:



Q-Value:

The Q Value is 22240.74

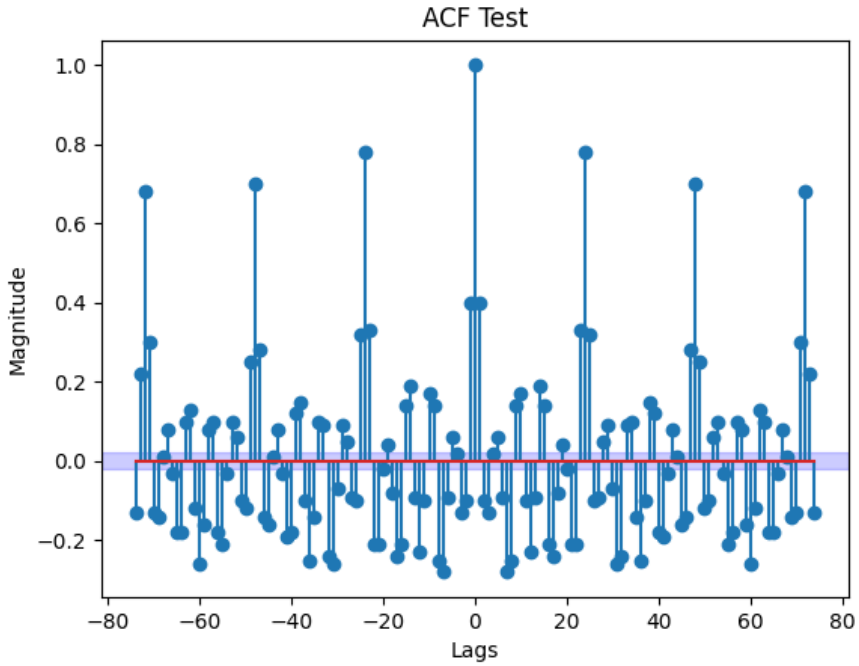
Chi-Test:

The chi critical value is: 74.91947430847816
The residual does not pass the whiteness test

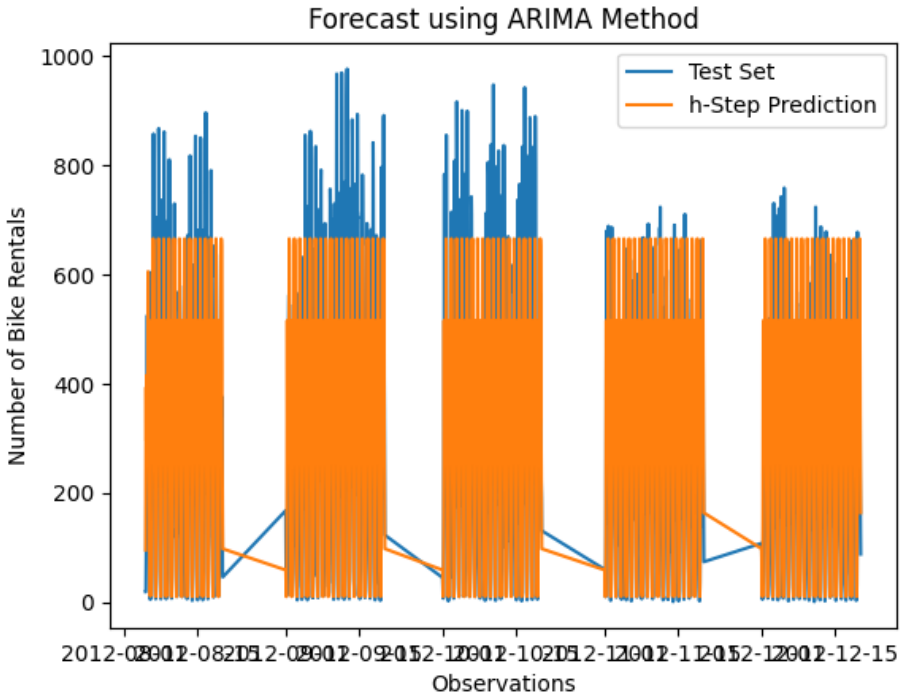
Variance of residual Error is:

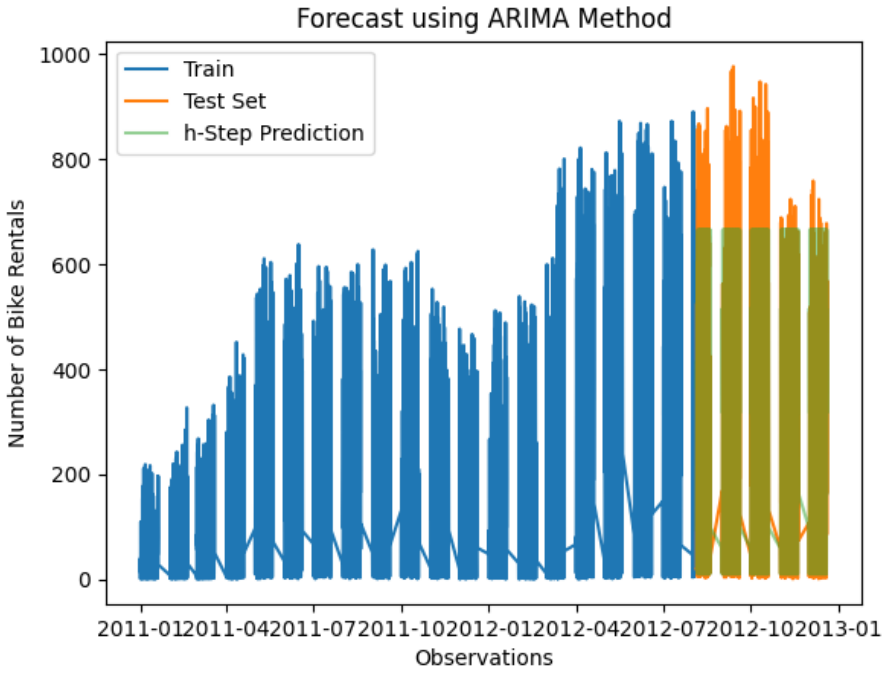
The variance of residual error is:
5587.78

ACF Plot for Residual Errors:



h-step prediction:



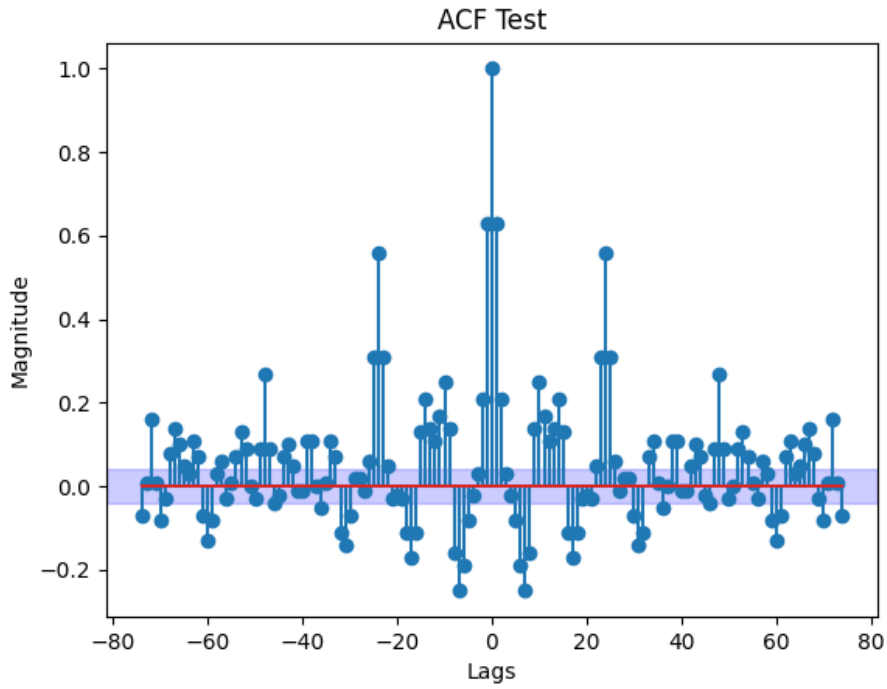


MSE of Forecast Error:
MSE of Forecast Errors with ARIMA Method: 22649.49

Variance of Forecast Error is:

The variance of forecast error is: 22087.92

ACF Plot for forecast errors:



Correlation Coefficient of Forecast Error with Test Set:
The Correlation Coefficient with SES Method is 0.54

Summary

Model Comparison:

Models	*MSE – Forecast Error*	*Variance Ratios*	*Q-Value*	*Correlation Coefficients*
Holt-Winter Method	25715.75	0.25	**8494.35**	**0.304**
OLS Method	46058.63	**0.65**	32357.55	0.94
Average Method	54724.40	0.53	55860.42	0.99
Naïve Method	107999.34	0.18	15438.53	0.99
Drift Method	108115.02	0.18	15438.53	0.99
SES (alpha=0.10)	48456.9	0.48	60804.84	0.99
ARIMA/SARIMA	**22649.49**	0.25	22240.74	0.54

The OLS Method is a close contender however, the Adjusted R^2 did not increase more than 0.278 even after eliminating the insignificant, the AIC and BIC did not reduce either from the Full model. Although the coefficients were significant based on the T-test and F-tests, the model did not improve. A notable observation is that the Variance Ratio is the highest for this model.

The **Holt-Winter** has the second lowest MSE compared to all the other models. The Q-value is also the lowest among all the models. The correlation coefficients of the forecast error with test set, is also the lowest implying the increase in forecast error with test data is the least among the models.

The ARIMA/SARIMA model has the lowest MSE, a low Correlation Coefficients and low Variance of error. We can perhaps optimize the ARIMA model by further differencing and looking for a better process.

Based on all the results, we pick the Holt-Winter Method as our best model. The ACF plot for the residual errors also shows us that most of the residuals fall in white-noise.

Conclusion:

In conclusion, we review all the forecasting methods for the Bike Sharing data. The data is a seasonally trended data with 24 hour cycles and we achieve stationarity by transforming the data by Logarithmic First Seasonal Differencing (of 24 cycles). The transformed data was then gave us a detrended and seasonally adjusted data and the potential ARIMA process looked to be: ARIMA(1, 0, 2) x $(0, 1, 2)_{24}$. We first applied Simple Forecasting Models such as Average Method, Naïve Method, Drift Method, SES Method and Holt-Winter Method. We also looked at the Multiple Linear Regression model using the OLS Method. We then developed the SARIMA model of the initially transformed data to get our forecasts. The models were evaluated by checking the Mean Squared Error of Forecast Errors, Variance of Errors, Q-Value and Correlation Coefficients of the models and pick the Holt-Winter model as our best model based on the results.

```
#%%
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
from statsmodels.tsa.seasonal import STL
import seaborn as sns
import datetime
from sklearn.model_selection import train_test_split
import statsmodels.tsa.holtwinters as ets
import statsmodels.api as sm
from numpy import linalg as LA
from scipy.stats import chi2

#%%
import sys
sys.path.append(r'/Users/varunshah/PycharmProjects/TimeSeries/toolbox.py')
from toolbox import cal_rolling_mean_var, errors, autocorr, ACF_PACF_Plot, \
    box_pierce, correlation_coefficient, ADF_Cal, kpss_test, \
    difference,calc_gpac, average_forecast, naive_method, drift_method, ses_method, lm_algorithm


#%%
# ================
# Load the data:
# ===============

data = 'https://raw.githubusercontent.com/varunshah111/time_series_analysis/main/project/data/train.csv'
bike = pd.read_csv(data)

#%%
# ==========================
# Data Prepreprocessing
# ==========================

# Printing the first 5 rows
bike.head()

```

```
#%%
bike.info()
#%%
# Converting the object type to datetime
bike['datetime'] = pd.to_datetime(bike['datetime'])

#%%
bike.info()

#%%
# Checking for Null values
bike.isna().sum()
#%%

# Plotting Count against time
x = bike['datetime']
y = bike['count']

f, ax = plt.subplots(figsize=(30,20))
plt.plot(x, y)

plt.xlabel('Time', fontsize = 30)
plt.ylabel('Number of Bikes rented', fontsize=30)
plt.xticks(rotation=45, horizontalalignment="center",
fontsize=25)
plt.yticks(rotation=0, horizontalalignment="right", fontsize
=25)
plt.title('Total Bike Rental over Time', fontsize=50)
plt.grid()
plt.show()

#%%

# Plotting Casual Renters, Registered bike renters and total
 bikes rented(combined)

x = bike['datetime']
y1 = bike['casual']
y2 = bike['registered']
y3 = bike['count']

fig, (ax1, ax2, ax3) = plt.subplots(ncols=1, nrows=3,
figsize=(30, 20), sharex=True)
```

```
78
79 ax1.plot(x, y1)
80 ax1.set_title("Casual Rented Bikes over Time", fontsize=25)
81 ax1.set_xlabel('Time', fontsize = 20)
82 ax1.set_ylabel('Number of Bikes rented', fontsize=20)
83 ax1.grid()
84
85 ax2.plot(x, y2)
86 ax2.set_title("Registered Rented Bikes over Time", fontsize
   =25)
87 ax2.set_xlabel('Time', fontsize = 20)
88 ax2.set_ylabel('Number of Bikes rented', fontsize=20)
89 ax2.grid()
90
91 ax3.plot(x, y3)
92 ax3.set_title("Total Bike Rental over Time", fontsize=25)
93 ax3.set_xlabel('Time', fontsize = 20)
94 ax3.set_ylabel('Number of Bikes rented', fontsize=20)
95
96 ax3.grid()
97 plt.show()
98
99 ____________________________________________________________
100 #%%
101 # ACF Plot
102
103 # For simplicity, we can focus only on the Total count of
    Bike Rentals.
104
105 autocorr(bike['count'], 50)
106 ____________________________________________________________
107 #%%
108 # ACF and PACF
109
110 ACF_PACF_Plot(bike['count'], 50)
111
112
113 ____________________________________________________________
114 #%%
115 # Correlation Heatmap
116
117 plt.subplots(figsize=(10,10))
118 sns.heatmap(bike.drop(columns = 'datetime').corr(), annot=
```

```
True, cmap='viridis' )
#plt.savefig('heatmap.png')
plt.yticks(rotation=0)
plt.show()


#%%
# Converting Season, holiday, workingday and weather to categorical


bike['season'] = bike['season'].astype('category')
bike['holiday'] = bike['holiday'].astype('category')
bike['workingday'] = bike['workingday'].astype('category')
bike['weather'] = bike['weather'].astype('category')
#%%
X = bike.drop(['count'], axis=1)
y = bike['count']
#%%
# Train-Test Split

y_train, y_test = train_test_split(y, shuffle= False, test_size=0.2)
X_train, X_test = train_test_split(X, shuffle= False, test_size=0.2)

h = len(y_test)

#%%
# Stationarity:
cal_rolling_mean_var(y_train, X_train['datetime'], "Bike Rental")

# ADF Test
ADF_Cal(y_train)
# we can reject the Null Hypothesis and adopt the alternative and state that the data is stationary

# KPSS Test
kpss_test(y_train)
# We are unable to reject the Null Hypothesis and can state that the data is stationary

```

```
#%%
# Sampling at seasonality of 24
# Lecture 11 slides
y_24 = y_train[:24:]
ACF_PACF_Plot(y_24, 10)

#%%

# Logarithmic First Differencing - seasonal: 24

Logarithmic = np.log(y_train)
# First Order Differencing for the logarithmic series:

FirstLogDiff = difference(Logarithmic, 24)

datetime_First = X_train['datetime'][24:]

cal_rolling_mean_var(FirstLogDiff, datetime_First, "
Logarithmic First Differencing with 24")


#%%
ACF_PACF_Plot(FirstLogDiff, 75)

# ADF Test
ADF_Cal(FirstLogDiff)
# we can reject the Null Hypothesis and adopt the
alternative and state that the data is stationary

# KPSS Test
kpss_test(FirstLogDiff)
# We are unable to reject the Null Hypothesis and can state
 that the data is stationary

#%%

# 8.
# STL Decomposition

transformed = pd.Series(FirstLogDiff,
                  index=pd.date_range('2011-01-01', periods
=len(FirstLogDiff), freq='h'), name='Transformed Data')

```

```
194 STL = STL(transformed)
195 res = STL.fit()
196 fig = res.plot()
197 plt.show()
198
199 T = res.trend
200 S = res.seasonal
201 R = res.resid
202
203 adj_seasonal = transformed - S
204 adj_trend = transformed - T
205
206 plt.plot(transformed, label = "Transformed Data")
207 plt.plot(adj_seasonal, label = "Seasonally adjusted Data")
208
209 plt.title("Seasonally Adjusted Data - STL")
210 plt.xlabel("Time")
211 plt.ylabel("Scale")
212 plt.legend()
213 plt.show()
214
215 F_S = np.maximum(0,1 - np.var(np.array(R))/np.var(np.array(
    S)+np.array(R)))
216 print("Strength of Seasonality for this data set is: ", F_S
    )
217
218 # F_S is equal to 0 exhibits no seasonality.
219 plt.plot(transformed, label = "Transformed Data") ,
220 plt.plot(adj_trend, label = "Adjusted Trend Data")
221
222 plt.title("Adjusted Trend Data - STL")
223 plt.xlabel("Time")
224 plt.ylabel("Scale")
225 plt.legend()
226 plt.show()
227
228 F_T = np.maximum(0,1 - np.var(np.array(R))/np.var(np.array(
    T)+np.array(R)))
229 print("Strength of Trend for this data set is: ", F_T)
230
231 # F_T is equal to 0, hence its completely detrended
232
233
```

```
234
235 #%%
236 #Holt-Winter Seasonal Method
237
238 holtt_seasonal = ets.ExponentialSmoothing(y_train, trend='
    add', damped_trend=True, seasonal='add', seasonal_periods=
    24).fit()
239 holtf_seasonal = holtt_seasonal.forecast(steps=h)
240 seasonal_predict = holtt_seasonal.fittedvalues
241
242 plt.plot(X_train['datetime'], y_train, label = 'Train')
243 plt.plot(X_test['datetime'], y_test, label = 'Test')
244 plt.plot(X_test['datetime'], holtf_seasonal, label = '
    Seasonal', alpha=0.5)
245 plt.title("Holt Seasonal Method")
246 plt.xlabel("Time")
247 plt.ylabel("Number of Bike Rental")
248 plt.legend()
249 plt.show()
250
251 #%%
252 # MSE of forecast error with Holt-Winter Method
253 mse_forecast_error_HoltS = errors(y_test, holtf_seasonal, h
    , 'mse')
254 print('MSE of Forecast Errors with Holt Seasonal Method: ',
    mse_forecast_error_HoltS)
255
256 # Variance of errors with Holt-Winter Seasonal Method
257 prediction_error_holtS = errors(y_train, seasonal_predict,
    1, 'error')
258 forecast_error_holtS = errors(y_test, holtf_seasonal, 5, '
    error')
259
260 var_pred_holtS = round((np.var(prediction_error_holtS)), 2)
261 var_forecast_holtS = round((np.var(forecast_error_holtS)),
    2)
262
263 print('\nVariance of Prediction Error with Holt Seasonal
    Method', var_pred_holtS)
264 print('Variance of Forecast Error with Holt Seasonal Method
    ', var_forecast_holtS)
265 #%%
266 var_ratio_holt = var_pred_holtS/var_forecast_holtS
```

```
print("The Ratio of Variance is: \n", var_ratio_holt)

#%%
# ACF Plot of the residuals
# Holt-Seasonal Method
autocorr(forecast_error_holtS, 50)

#%%
# Q-Value:
T = len(bike)
box_pierce_holtS = box_pierce(prediction_error_holtS, T, 48
)
print("The Q value (Box-Pierce Test) with Holt Seasonal,
for the prediction error is: ", round((box_pierce_holtS),2
))
#%%

# Correlation-Coefficient:
print("The Correlation Coefficient with Holt Seasonal
Method is ", correlation_coefficient(forecast_error_holtS,
y_test))

#%%
# Feature Selection
X = bike.drop(['count', 'datetime', 'casual','registered'
], axis=1)
y = bike['count']

from sklearn.preprocessing import MinMaxScaler

# The StandardScaler
mms = MinMaxScaler()

#%%
# Normalize the training data


normalized_temp = X['temp'].values.reshape((len(X['temp'
]), 1))
scaler = MinMaxScaler(feature_range=(0, 1))
X['temp'] = scaler.fit_transform(normalized_temp)

normalized_atemp = X['atemp'].values.reshape((len(X['atemp'
```

```
]), 1))
scaler = MinMaxScaler(feature_range=(0, 1))
X['atemp'] = scaler.fit_transform(normalized_atemp)

normalized_humidity = X['humidity'].values.reshape((len(X['humidity']), 1))
scaler = MinMaxScaler(feature_range=(0, 1))
X['humidity'] = scaler.fit_transform(normalized_humidity)

normalized_windspeed = X['windspeed'].values.reshape((len(X['windspeed']), 1))
scaler = MinMaxScaler(feature_range=(0, 1))
X['windspeed'] = scaler.fit_transform(normalized_windspeed)


#%%

# Train-Test Split

# Q1. Train-Test Split
X = sm.add_constant(X) # Adding the constant 1 at the beginning
X_train, X_test, y_train, y_test = train_test_split(X, y, shuffle = False, test_size=0.2)

h = len(y_test)


#%%
# ================================
# Collinearity Detection
# ================================

# Constructing the Matrices
XMat = X_train.values
YMat = y_train.values


#%%

# Singular Value Decomposition
H = np.matmul(XMat.T, XMat)
_, d, _ = np.linalg.svd(H)
```

```
341 print("Singular Values of X are: \n",d)
342
343 # Based on the singular values, we can state that there is
    collinearity between at-least 4 features.
344 # This is based on the lowest values from SVD, which are
    close to zero.
345 # These can be eliminated from our model.
346 #%%
347
348 # Condition Number
349 print(f'The condition Number for X is {round(LA.cond(XMat
    ), 2)}')
350
351 # The Condition Number is greater than 1000, showing severe
     degree of collinearity
352
353 #%%
354 # Model Summary using OLS Function:
355
356 # ========================================
357 # Finding Coefficients using OLS Method
358 # ========================================
359
360
361 model = sm.OLS(y_train, X_train).fit()
362 print(model.summary())
363
364 #%%
365 # ================================
366 # Backward Stepwise Regression
367 # ================================
368
369 # The "weather" feature has the highest p-value at 0.583,
    from the above summary, implying it is not significant.
370 # Removing the "weather" feature
371
372 X_train.drop(['weather'], axis=1, inplace=True)
373 model = sm.OLS(y_train, X_train).fit()
374 print(model.summary())
375
376 #%%
377
378 # Removing "workingday" which is at p-value: 0.394
```

```
379
380 X_train.drop(['workingday'], axis=1, inplace=True)
381 model = sm.OLS(y_train, X_train).fit()
382 print(model.summary())
383
384 #%%
385
386 # Removing "holiday" which is at p-value: 0.371
387 X_train.drop(['holiday'], axis=1, inplace=True)
388 model = sm.OLS(y_train, X_train).fit()
389 print(model.summary())
390
391 #%%
392
393 # Removing "temperature" as it is collinear with "atemp"
394
395 X_train.drop(['temp'], axis=1, inplace=True)
396 model = sm.OLS(y_train, X_train).fit()
397 print(model.summary())
398
399 #%%
400 # Removing "season" at p-value: 0.149
401 # Final Model
402
403 X_train.drop(['season'], axis=1, inplace=True)
404 model = sm.OLS(y_train, X_train).fit()
405 print(model.summary())
406
407 #%%
408 # ============================
409 # Q8. Predictions and plots.
410 # ============================
411
412 X_test = X_test[['const','atemp','humidity','windspeed']]
413
414 predictionOLS = model.predict(X_train)
415 forecastOLS = model.predict(X_test)
416
417 plt.plot(y_train, label = "Training Set") # X_train,
418 plt.plot(y_test, label = "Test Set") #X_test,
419 plt.plot(forecastOLS, label = "Prediction Values") # X_test
    ,
420
```

```
421 plt.title("Forecast using OLS Regression")
422 plt.xlabel("Number of Samples")
423 plt.ylabel("Bike Rentals")
424 plt.legend()
425 plt.show()
426 
427 #%%
428 
429 # ====================================
430 # Prediction errors and ACF Plot
431 # ====================================
432 prediction_errorsOLS = errors(y_train, predictionOLS, 1, '
    error')
433 autocorr(prediction_errorsOLS, 48, True)
434 
435 #%%
436 
437 # ====================================
438 # Forecast Errors and ACF Plot
439 # ====================================
440 forecast_errorsOLS = errors(y_test, forecastOLS, h, 'error'
    )
441 autocorr(forecast_errorsOLS, 48, True)
442 
443 
444 #%%
445 # ======================
446 # Mean Squared Errors:
447 # ======================
448 
449 # MSE of forecast error with Holt-Winter Method
450 mse_forecast_errorOLS = errors(y_test, forecastOLS, h, 'mse
    ')
451 print('MSE of Forecast Errors with OLS Method: ',
    mse_forecast_errorOLS)
452 
453 #%%
454 T = len(bike)
455 box_pierceOLS = box_pierce(prediction_errorsOLS, T, 48)
456 print("The Q value (Box-Pierce Test) with OLS Method, for
    the prediction error is: ", round((box_pierceOLS),2))
457 #%%
458 
```

```
459 # Correlation-Coefficient:
460 print("The Correlation Coefficient with OLS Method is ",
    correlation_coefficient(forecast_errorsOLS, y_test))
461
462 ____________________________________________________________
463 #%%
464 # ============================================================
    =====
465 # Estimated Variance of prediction errors and forecast
    errors
466 # ============================================================
    =======
467
468 SSE_predictionOLS = errors(y_train, predictionOLS, 1, 'sse'
    )
469 SSE_forecastOLS = errors(y_test, forecastOLS, len(y_test),
    'sse')
470
471 # k = Number of predictors
472 # T = Number of observations
473
474 T_trainSet = len(predictionOLS) # Number of samples of
    training set
475 T_testSet = len(forecastOLS) # Number of samples of test
    set
476
477 k = X_train.shape[1] # Number of predictors
478
479 var_predictOLS = ((SSE_predictionOLS/(T_trainSet-k-1))**0.5
    )
480 var_forecastOLS = ((SSE_forecastOLS/(T_testSet-k-1))**0.5)
481
482 print("The Estimated Variance for Prediction Errors is: ",
    round(var_predictOLS, 2))
483 print("The Estimated Variance for Forecast Errors is: ",
    round(var_forecastOLS, 2))
484 ____________________________________________________________
485 #%%
486 var_ratio_ols = var_predictOLS/var_forecastOLS
487 print("The Variance Ratio is: ", var_ratio_ols)
488 ____________________________________________________________
489 #%%
490 # T-Test and F-Test:
```

```
491
492 # T-Test:
493
494 print(model.params)
495 print(model.pvalues)
496 print(model.tvalues)
497
498 #%%
499 # F Test:
500 A = np.identity(len(model.params))
501 A = A[1:,:]
502 print(model.f_test(A))
503
504 print(model.fvalue)
505 print(model.f_pvalue)
506
507 #%%
508
509 # Base Models:
510 # Train-test Split:
511
512 X = bike.drop(['count'], axis=1)
513 y = bike['count']
514
515 y_train, y_test = train_test_split(y, shuffle= False,
    test_size=0.2)
516 X_train, X_test = train_test_split(X, shuffle= False,
    test_size=0.2)
517
518 h = len(y_test)
519 #%%
520
521 # Average Method
522 y_hat_train_avg, y_hat_test_avg = average_forecast(y_train
    , h)
523
524 # Plot
525
526 plt.plot(X_train['datetime'], y_train, label = "Training
    Set")
527 plt.plot(X_test['datetime'], y_test, label = "Test Set")
528 plt.plot(X_test['datetime'], y_hat_test_avg, label = "h-
    step prediction")
```

```
529
530 plt.title("Forecast using Average Method")
531 plt.xlabel("Months")
532 plt.ylabel("Number of Bike Rentals")
533 plt.legend()
534 plt.show()
535
536 #%%
537
538 # MSE for Forecast Error with Average Method
539 mse_forecast_error_avg = errors(y_test, y_hat_test_avg, h,
    'mse')
540 print('MSE of Forecast Errors with Average Method: ',
    mse_forecast_error_avg)
541 #%%
542
543 # Variance of errors with Average Method:
544 prediction_error_avg = errors(y_train, y_hat_train_avg, 1,
    'error')
545 forecast_error_avg = errors(y_test, y_hat_test_avg, h, '
    error')
546
547 var_pred_avg = round((np.var(prediction_error_avg)), 2)
548 var_forecast_avg = round((np.var(forecast_error_avg)), 2)
549
550 print('Variance of Prediction Error with Average Method',
    var_pred_avg)
551 print('Variance of Forecast Error with Average Method',
    var_forecast_avg)
552
553 #%%
554 var_ratio_avg = var_pred_avg/var_forecast_avg
555 print("The Variance Ratio is: ", var_ratio_avg)
556
557 #%%
558 # Average Method - ACF
559 autocorr(prediction_error_avg, 48, plot = True)
560
561 #%%
562
563 T = len(bike)
564 box_pierce_avg = box_pierce(prediction_error_avg, T, 48)
565 print("The Q value (Box-Pierce Test) with Average Method,
```

```
for the prediction error is: ", round((box_pierce_avg),2))

#%%

# Correlation Coefficients for Forecast Errors and Test Set
print("The Correlation Coefficient with Average Method is "
,correlation_coefficient(y_test, forecast_error_avg))

#%%

# Naive Method:

y_hat_train_naive, y_hat_test_naive = naive_method(y_train
, h)

plt.plot(X_train['datetime'], y_train, label = "Training
Set")
plt.plot(X_test['datetime'], y_test , label = "Test Set")
plt.plot(X_test['datetime'], y_hat_test_naive, label = "h-
step prediction")

plt.title("Forecast using Naive Method")
plt.xlabel("Months")
plt.ylabel("Number of Bike Rentals")
plt.legend()
plt.show()

#%%
# MSE of forecast error with Naive Method
mse_forecast_error_naive = errors(y_test, y_hat_test_naive
, h, 'mse')
print('MSE of Forecast Errors with Naive Method: ',
mse_forecast_error_naive)


# Variance of Errors with Naive Method:
prediction_error_naive = errors(y_train, y_hat_train_naive
, 1, 'error')
forecast_error_naive = errors(y_test, y_hat_test_naive, h,
'error')

var_pred_naive = round((np.var(prediction_error_naive)), 2)
var_forecast_naive = round((np.var(forecast_error_naive)),
```

```
2)

print('\nVariance of Prediction Error with Naive method', var_pred_naive)
print('Variance of Forecast Error with Naive Method', var_forecast_naive)

#%%

var_ratio_naive = var_pred_naive/var_forecast_naive
print("The Variance Ratio is: ", var_ratio_naive)

#%%

# Naive Method - ACF
autocorr(prediction_error_naive, 48, plot = True)

box_pierce_naive = box_pierce(prediction_error_naive, T, 48)
print("The Q value (Box-Pierce Test) with Naive Method, for the prediction error is: ", round((box_pierce_naive),2))

print("The Correlation Coefficient with Naive Method is ", correlation_coefficient(y_test, forecast_error_naive))

#%%
# Drift Method

y_hat_train_drift, y_hat_test_drift = drift_method(y_train, h)

plt.plot(X_train['datetime'], y_train, label = "Training Set")
plt.plot(X_test['datetime'], y_test , label = "Test Set")
plt.plot(X_test['datetime'], y_hat_test_drift, label = "Forecast")

plt.title("Forecast using Drift Method")
plt.xlabel("Observations")
plt.ylabel("Number of Bike Rentals")
plt.legend()
plt.show()
```

```
#%%
# MSE of forecast error with Drift Method
mse_forecast_error_drift = errors(y_test, y_hat_test_drift, h, 'mse')
print('MSE of Forecast Errors with Drift Method: ', mse_forecast_error_drift)

#%%
# Variance of errors with Drift method:

prediction_error_drift = errors(y_train, y_hat_train_drift, 1, 'error')
forecast_error_drift = errors(y_test, y_hat_test_drift, 5, 'error')

var_pred_drift = round((np.var(prediction_error_drift)), 2)
var_forecast_drift = round((np.var(forecast_error_drift)), 2)

print('\nVariance of Prediction Error with Drift Method', var_pred_drift)
print('Variance of Forecast Error with Drift Method', var_forecast_drift)

#%%
var_ratio_drift = var_pred_drift/var_forecast_drift
print("The Variance Ratio is: ", var_ratio_drift)


# Drift Method
autocorr(prediction_error_drift, 48, plot = True)

#%%
box_pierce_drift = box_pierce(prediction_error_drift, T, 48)
print("The Q value (Box-Pierce Test) with Drift Method, for the prediction error is: ", round((box_pierce_drift),2))

#%%
print("The Correlation Coefficient with Drift Method is ", correlation_coefficient(y_test, forecast_error_drift))

#%%

```

```
667 # SES Method
668 y_hat_train_ses, y_hat_test_ses = ses_method(y_train, 0.10
    , y_train[0], h)
669 #%%
670 plt.plot(X_train['datetime'], y_train, label = "Training
    Set")
671 plt.plot(X_test['datetime'], y_test , label = "Test Set")
672 plt.plot(X_test['datetime'], y_hat_test_ses, label = "h-
    step prediction")
673 plt.title("Forecast using SES Method at alpha=0.10")
674 plt.xlabel("Year")
675 plt.ylabel("Number of Bike Rentals")
676 plt.legend()
677 plt.show()
678
679 #%%
680 # MSE of forecast error with SES Method at alpha 0.10
681 mse_forecast_error_ses = errors(y_test, y_hat_test_ses, h,
    'mse')
682 print('MSE of Forecast Errors with SES Method: ',
    mse_forecast_error_ses)
683
684 #%%
685
686 # Variance of errors with SES Method with alpha at 0.10
687 prediction_error_ses = errors(y_train, y_hat_train_ses, 1,
    'error')
688 forecast_error_ses = errors(y_test, y_hat_test_ses, h, '
    error')
689
690 var_pred_ses = round((np.var(prediction_error_ses)), 2)
691 var_forecast_ses = round((np.var(forecast_error_ses)), 2)
692
693 print('\nVariance of Prediction Error with SES Method',
    var_pred_ses)
694 print('Variance of Forecast Error with SES Method',
    var_forecast_ses)
695
696 #%%
697 var_ratio_ses = var_pred_ses/var_forecast_ses
698 print("The Variance Ratio is: ", var_ratio_ses)
699
700
```

```
#%%

# SES Method with alpha = 0.10 - ACF
autocorr(prediction_error_ses, 48, plot = True)

#%%
box_pierce_ses = box_pierce(prediction_error_ses, T, 48)
print("The Q value (Box-Pierce Test) with SES Method (at alpha=0.10), for the prediction error is: ", round((box_pierce_ses),2))


#%%
# Correlation Coefficients for Forecast Errors and Test Set
print("The Correlation Coefficient with SES Method is ", correlation_coefficient(forecast_error_ses, y_test))
#%%

# ARIMA/SARIMA Process

X = bike.drop(['count'], axis=1)
y = bike['count']

# Train-Test Split

y_train, y_test = train_test_split(y, shuffle= False, test_size=0.2)
X_train, X_test = train_test_split(X, shuffle= False, test_size=0.2)

h = len(y_test)
#%%
ACF_PACF_Plot(FirstLogDiff, 75)


#%%
# First Log Diff

ry,_, _ = autocorr(FirstLogDiff, 100, plot=False)

# 7x7 GPAC
```

```
gpac = calc_gpac(ry, 7, 7)

#%%

# 30 x 30
gpac = calc_gpac(ry, 26, 26)

#%%
print(gpac.to_string())

#%%
# Fitting the model:

non_seasonalOrder = (1,0,2)
seasonalOrder = (0,1,2,24)

model = sm.tsa.statespace.SARIMAX(y_train, order=
non_seasonalOrder, seasonal_order=seasonalOrder)

#%%
result = model.fit()

#%%
print(result.summary())


#%%
print("The estimated parameters are: \n", result.params)

#%%

# Confidence Interval:
print("The Confidence Interval is: \n",result.conf_int())


#%%
# One-Step Prediction

train_predict=result.get_prediction(start=1, end=len(
y_train), dynamic=False)
sarima_pred=train_predict.predicted_mean

```

```
#%%
#
plt.plot(X_train['datetime'], y_train, label = "Training
    Set")
#plt.plot(X_test['datetime'], y_test , label = "Test Set")
plt.plot(X_train['datetime'], sarima_pred, label = "One-
    Step Prediction")

plt.title("One-Step Prediction using ARIMA Method")
plt.xlabel("Observations")
plt.ylabel("Scale")
plt.legend()
plt.show()
```

```
#%%
residuals = errors(y_train, sarima_pred, 1, 'error')

# ACF Plot:
autocorr(residuals, 75, plot = True)

ACF_PACF_Plot(residuals, 75)
```

```
#%%

# Q Values
T = len(y_train)
Qvalue = box_pierce(residuals, T, 48)
print("The Q Value is", Qvalue)
```

```
#%%

Q_value = sm.stats.acorr_ljungbox(residuals, lags=[48]) #,
    return_df=True)

print("The Q Value is", Q_value)
```

```
#%%
# Chi Test
alpha = 0.01
DOF = 1 + 48
chi_critical = chi2.ppf(1 - alpha, DOF)
chi_critical

```

```
print("The chi critical value is:", chi_critical)
if chi_critical > Qvalue:
    print("The residual passes the whiteness test")
else:
    print("The residual does not pass the whiteness test")


#%%
# Variance:

var_residual_arima = np.var(residuals)
print("The variance of residual error is: \n",
var_residual_arima)


#%%

# h-step prediction

test_forecast=result.get_prediction(start=len(y_train)+1,
end=len(bike), dynamic=False)
sarima_forecast=test_forecast.predicted_mean


#%%

plt.plot(X_test['datetime'], y_test , label = "Test Set")
plt.plot(X_test['datetime'], sarima_forecast, label = "h-
Step Prediction")
plt.title("Forecast using ARIMA Method")
plt.xlabel("Observations")
plt.ylabel("Number of Bike Rentals")
plt.legend()
plt.show()
#%%

plt.plot(X_train['datetime'], y_train, label = 'Train')
plt.plot(X_test['datetime'], y_test , label = "Test Set")
plt.plot(X_test['datetime'], sarima_forecast, label = "h-
Step Prediction", alpha=0.50)
plt.title("Forecast using ARIMA Method")
plt.xlabel("Observations")
```

```
plt.ylabel("Number of Bike Rentals")
plt.legend()
plt.show()

#%%
ACF_PACF_Plot(sarima_forecast, 75)

#%%
# MSE of forecast error with ARIMA method
mse_forecast_error_sarima = errors(y_test, sarima_forecast, len(y_test), 'mse')
print('MSE of Forecast Errors with ARIMA Method: ', mse_forecast_error_sarima)

#%%
forecast_errors_arima = errors(y_test, sarima_forecast, 1, 'error')


# ACF Plot:
autocorr(forecast_errors_arima, 75, plot = True)

#%%
ACF_PACF_Plot(forecast_errors_arima, 75)

#%%
# Variance:
var_forecast_arima = np.var(forecast_errors_arima)
print("The variance of forecast error is: \n", var_forecast_arima)

#%%
var_ratio_arima = var_residual_arima/var_forecast_arima
print("The Variance Ratio is: ", var_ratio_arima)

#%%

# Correlation Coefficients for Forecast Errors and Test Set
print("The Correlation Coefficient with ARIMA Method is ", correlation_coefficient(forecast_errors_arima, y_test))

#%%

```