Course Name: Time series Analysis and Modeling DATS 6313

Instructor: Reza Jafari, PHD

Report: Final Project

student: Mahtab Barkhordarian

Date:01/05/2022

CONTENT

LIST OF TABLE

Abstract:

This is the final project of Time series, which I tried to cover all the knowledge I gained from this course in this project. This project contains the way of checking the stationary data, Decomposition, Holt winter method, Feature selection, Base models, multiple linear regression, ARMA, Diagnostic Analysis and finally compared the results to choose the best model. some of the methods which are covered in this course are not inside this project since it's the real dataset and it wasn't under control to have all the methods together.

Introduction:

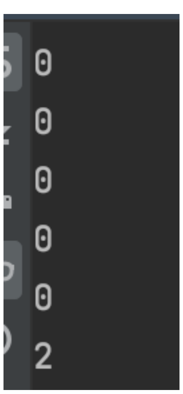
In this project first tried to make the data clean, then tried the potential ways to understand if the data is stationary or not and then tried to use different models to see which one works best on this data set. For this dataset, the feature selection method was covered to shows which methods have the most impact on the final model.

Dataset:

The data set is the underwater surface temperature from the island in Brazil from Kaggle. This dataset had 408639 observations before cleaning. It was the every 20 minutes dataset which was converted to hourly one. Then after the preprocessing, the length of the data is 136213. It's a big dataset but because of the more accuracy, I kept it and tried to work with that for the more accurate prediction. this data set has 8 columns including id. Then there are 7 variables which one of them is dependent which is the Temp and we are going to predict Temp in this dataset. The dataset is structured in seven variables: Site, Latitude, Longitude, Date, Time of Sampling, Temperature (°C) and Depth (meters),Rest of the variables are independent. The time which are in this dataset includes the date between December 2012 till July 2014.

Preprocessing :

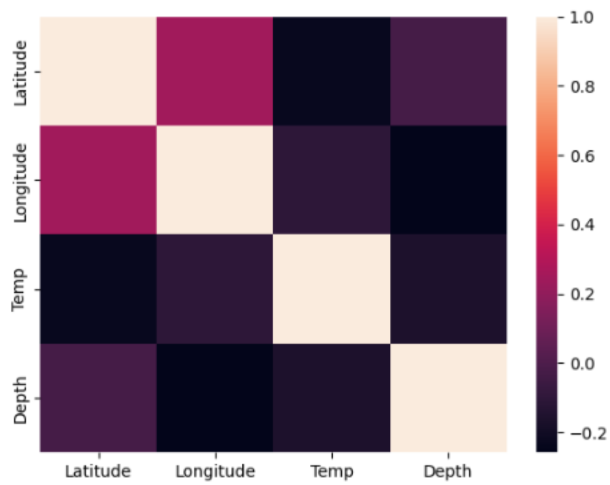
For timeseries datasets, we cannot remove the nulls, then mostly for numeric data we can use mean(average) of that column to substitute by and for the categorical data, we can substitute the null by the mode of that column depends on the situation. In this dataset we just had 2 nulls the Temp column which is filled with the average of that column.



Here is the first 5 rows after preprocessing and as you see there is no nan in this dataset, if you notice the ids you can see, they are chosen hourly not every 20 minutes.

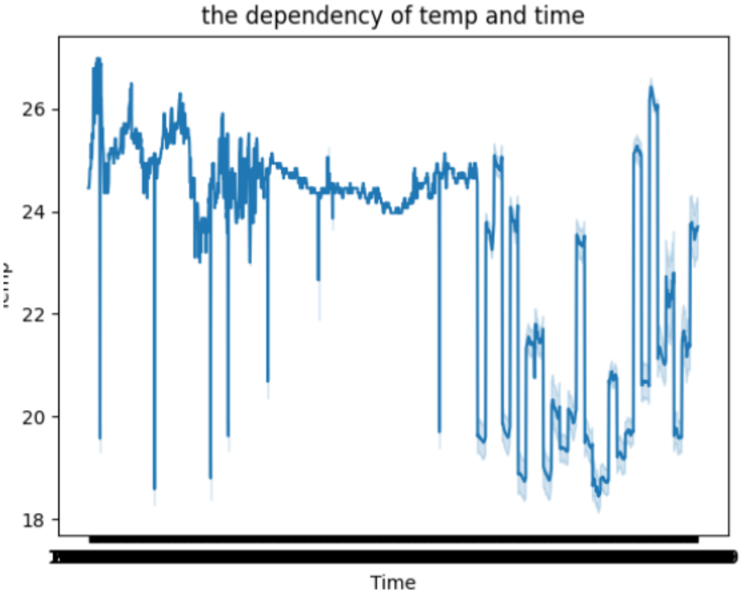
```
>>> df.head(5)
             Site  Latitude  Longitude     Date   Time    Temp  Depth
ID
1   Ilha Deserta   27.2706     48.331  2/20/13  11:40  24.448   12.0
4   Ilha Deserta   27.2706     48.331  2/20/13  12:40  24.448   12.0
7   Ilha Deserta   27.2706     48.331  2/20/13  13:40  24.545   12.0
10  Ilha Deserta   27.2706     48.331  2/20/13  14:40  24.641   12.0
13  Ilha Deserta   27.2706     48.331  2/20/13  15:40  24.835   12.0
>>> df.isnull().sum().sum()
0
```

Heatmap:



This heatmap shows the coefficient between variables, which we can see latitude and longitude has the most correlation together which makes sense, and then depth and Latitude has the most correlation.

Stationarity:
To get to know if the dataset is stationary or not, tried to plot the dependent variable which is temp vs time.

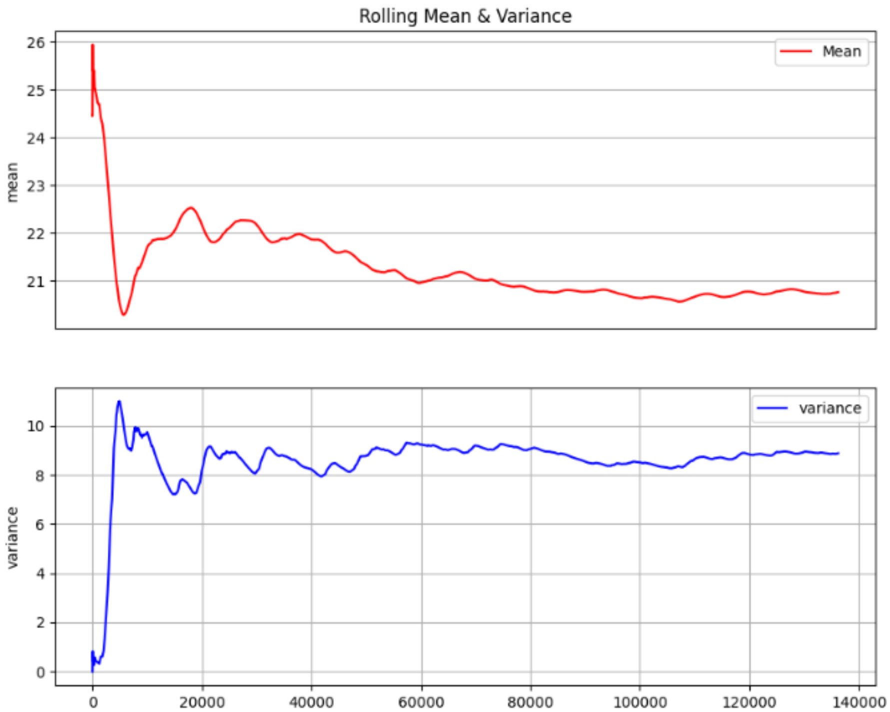


With the optical view, it is not stationary for me, but I am trying to do the ADF test, rolling mean and var and KPSS test since never the optical test is trustable.

ADF/KPSS:
The ADF test shows that the 'Temp' is stationary , however the KPSS shows it is not because the P-value is 0.01 which is less than 0.05. because of that we are doing the rolling mean and var to make sure.

```
ADF Statistic :-9.043268
p_value: 0.000000
Critical Values:
    1%:-3.430
    5%:-2.862
    10%:-2.567
Temp is Stationary
KPSS output for Temp is :
Test Statistic               2.08906
p-value                      0.01000
LagsUsed                   208.00000
Critical Value (10%)         0.34700
Critical Value (5%)          0.46300
Critical Value (2.5%)        0.57400
Critical Value (1%)          0.73900
dtype: float64
>>>
```

This is the rolling mean and var plot which shows the dependent variable is stationary .



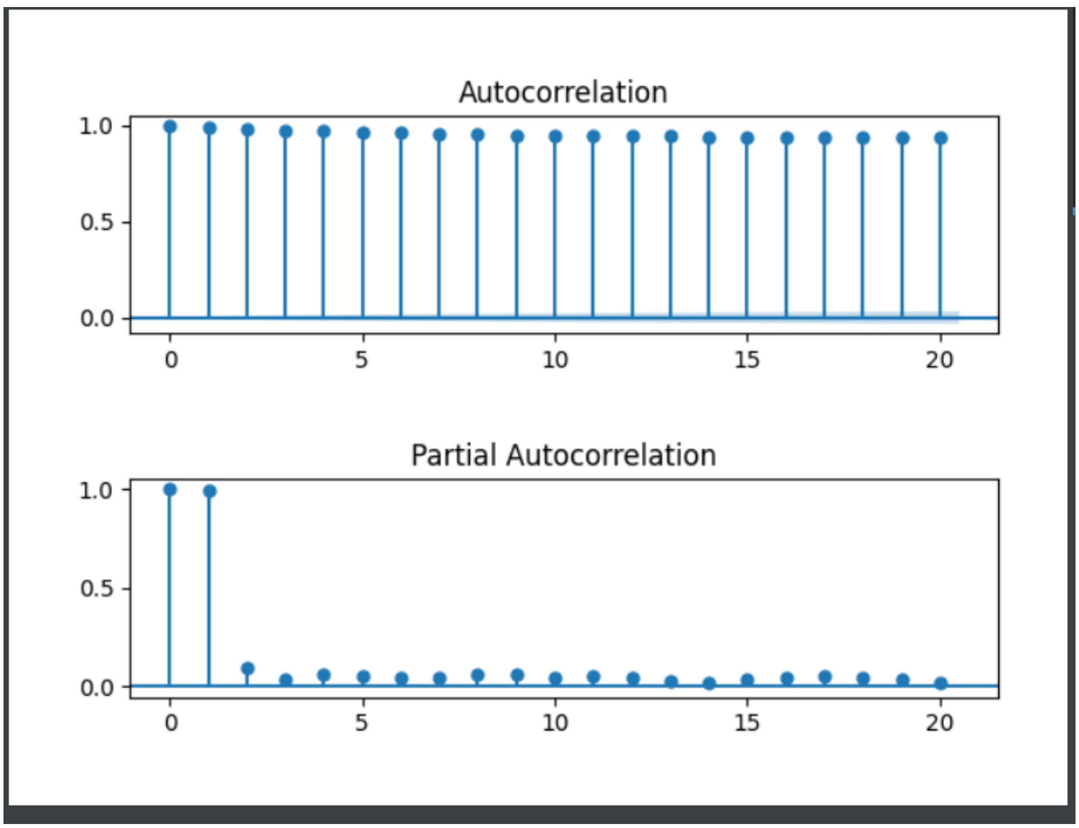
With the ADF test result and rolling mean and var plot, we can realize that the target is stationary, however just because of my KPSS test, I have done the first differencing to make sure that it is stationary and if it is needed for the ARIMA, if my GPAC is not going to work then I have the differenced data which is 100% stationary to work on.

First Differencing:

After the first differencing the kpss test is stationary because the p-value is 0.1 which is greater than 0.05.

```
KPSS output for Temp is :
Test Statistic              0.017785
p-value                     0.100000
LagsUsed                  457.000000
Critical Value (10%)        0.347000
Critical Value (5%)         0.463000
Critical Value (2.5%)       0.574000
Critical Value (1%)         0.739000
dtype: float64
```

ACF/PACF:

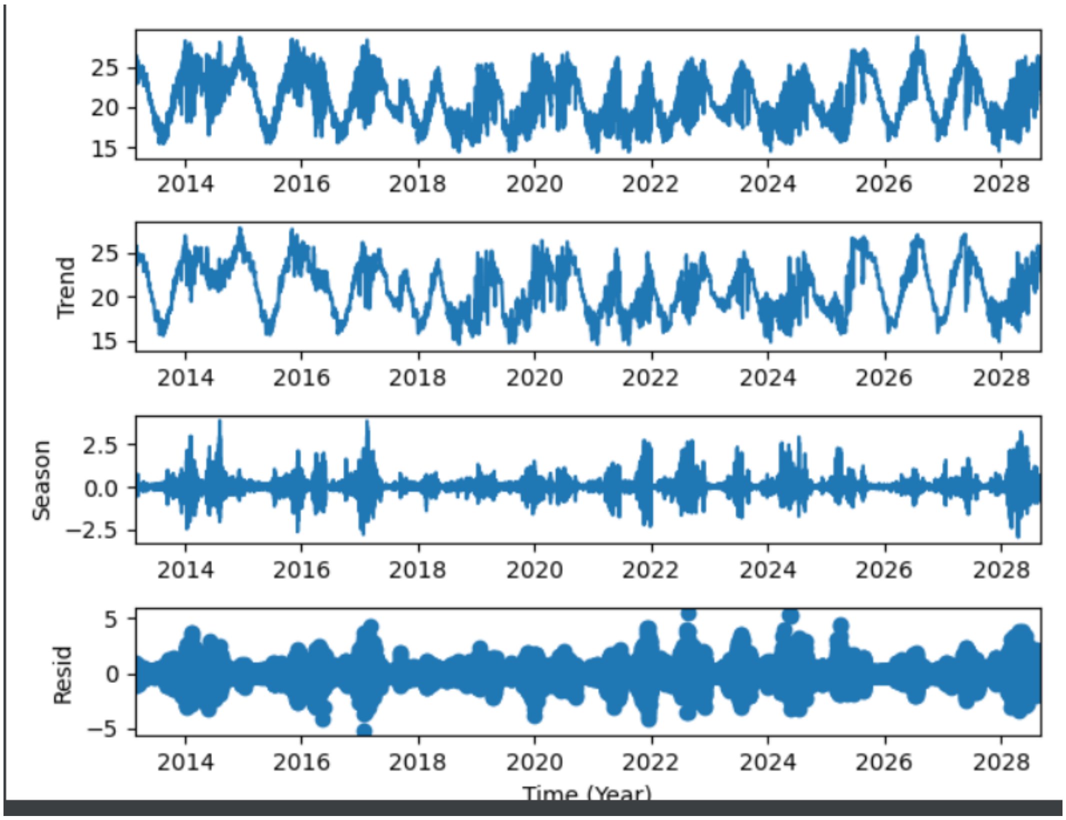


This is our ACF and PACF plot , from the PACF we can guess probable the ARMA(1,0) is going to work in our GPAC.

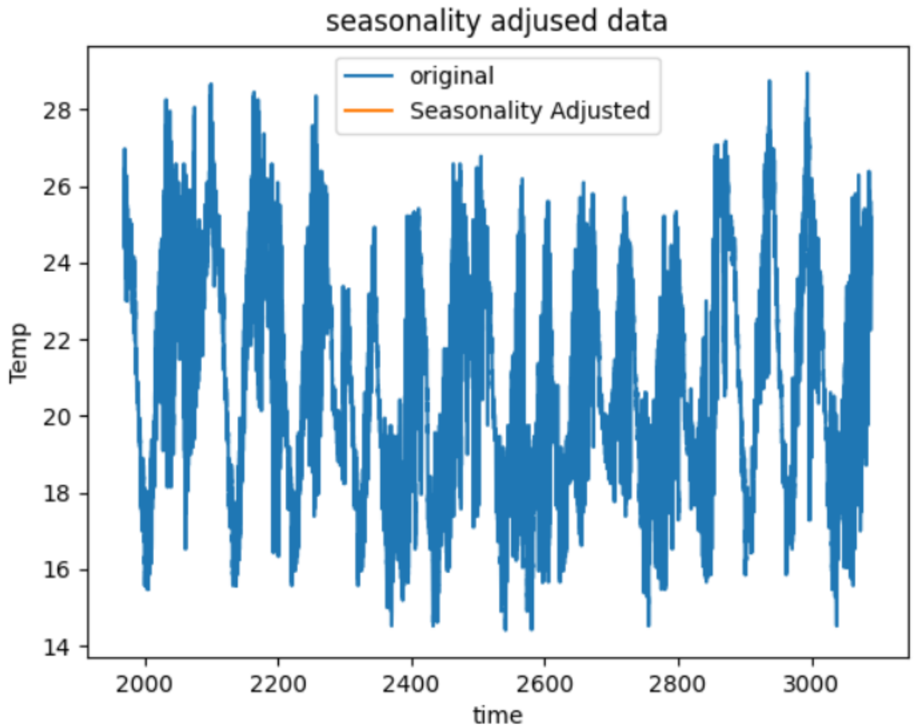
Time Series Decomposition:

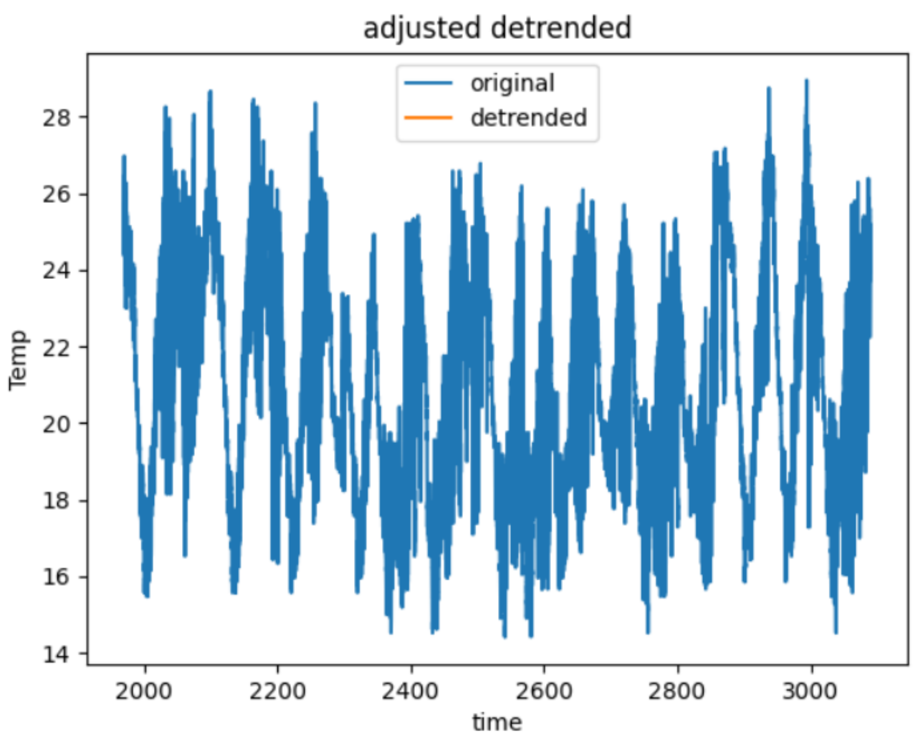
```
the strength of Trend is0.9827413173156924
the strength of seasonality is0.465138448789971
```

After the decomposition we realized that it is highly trended, but weak seasonality.



This plot which also shows that it is highly trended but not seasonal.





This is the plot after detrended.

Holt-winters method:

I used the holt-trended methods instead of holt-winters because my data was highly trended.

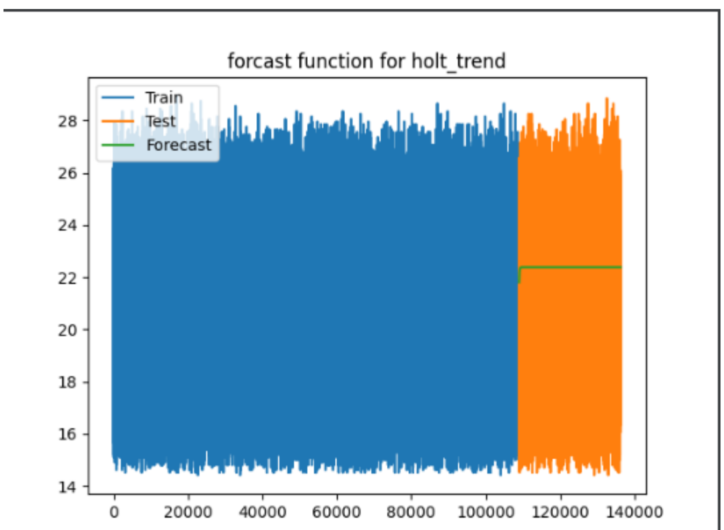
```
      lb_stat     lb_pvalue
20  241.978138  4.713375e-40
The residual is not white
From acorr_ljungbox test
```

The p value for the holt trended is less than 0.05 which shows that the residual error is not white, however I did the Q square test here to realize that.

This is the result for the Holt-trend method:

```
the estimated variance error of residual for holt is : 9.033590233391696
the estimated variance error of forcast for holt is : 8.938910271777898
the ratio bet var of pred and forcast in holt-trend: 1.0105918908161236
The Mean Square Error of predict for holt_trend is:  8.52268735021395
The Mean Square Error of forcast for holt-trend method is:  11.493248926462863
```

The ratio of the variance of the error for predict divided by forecast is almost 1 which shows that it is not a bad model, but let's see all the models and compare the results together, then decide.



Feature Selection:

With OLS regression, I did feature selection. with that , I added the constant to the features in the beginning and then after I observed the result, I should remove the highest p value until there is the big change in the R squared error. However , in this case all the p values were 0 and the condition number was so low , then I decided with std error which was high for the constant. Then I tried to remove the constant and it shows that the model is not linear, it is centered.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.092
Model:                            OLS   Adj. R-squared:                  0.092
Method:                 Least Squares   F-statistic:                     4609.
Date:                Mon, 02 May 2022   Prob (F-statistic):               0.00
Time:                        08:00:59   Log-Likelihood:            -3.3542e+05
No. Observations:              136213   AIC:                         6.709e+05
Df Residuals:                  136209   BIC:                         6.709e+05
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        217.4831      4.064     53.515      0.000     209.518     225.448
Latitude      -1.2250      0.016    -78.611      0.000      -1.256      -1.194
Longitude     -3.3456      0.086    -39.101      0.000      -3.513      -3.178
Depth         -0.0952      0.001    -73.772      0.000      -0.098      -0.093
==============================================================================
Omnibus:                    20033.296   Durbin-Watson:                   0.020
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4740.369
Skew:                          -0.016   Prob(JB):                         0.00
Kurtosis:                       2.087   Cond. No.                     3.01e+04
==============================================================================
>>>
```

Here is the result after we removed the constant:

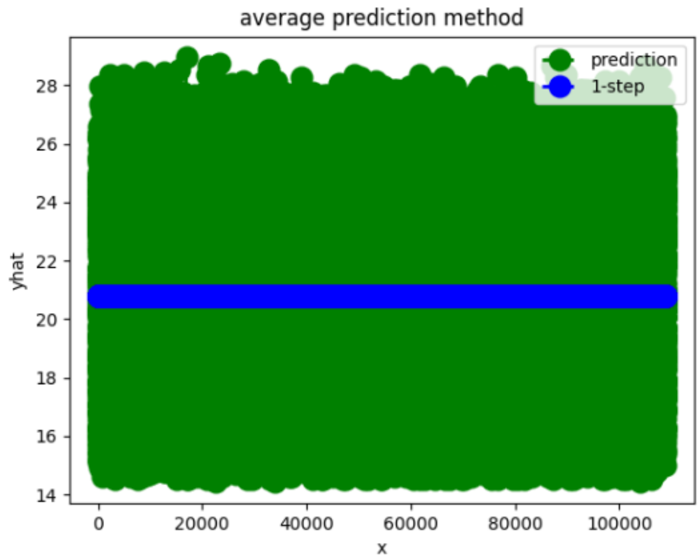
```
                            OLS Regression Results
=======================================================================================
Dep. Variable:                      y   R-squared (uncentered):                   0.981
Model:                            OLS   Adj. R-squared (uncentered):              0.981
Method:                 Least Squares   F-statistic:                          2.381e+06
Date:                Mon, 02 May 2022   Prob (F-statistic):                        0.00
Time:                        08:00:59   Log-Likelihood:                     -3.3684e+05
No. Observations:              136213   AIC:                                  6.737e+05
Df Residuals:                  136210   BIC:                                  6.737e+05
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Latitude      -1.3471      0.016    -86.485      0.000      -1.378      -1.317
Longitude      1.2094      0.009    137.197      0.000       1.192       1.227
Depth         -0.0769      0.001    -61.195      0.000      -0.079      -0.074
==============================================================================
Omnibus:                    23748.173   Durbin-Watson:                   0.019
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             5108.958
Skew:                          -0.028   Prob(JB):                         0.00
Kurtosis:                       2.053   Cond. No.                         131.
==============================================================================
```

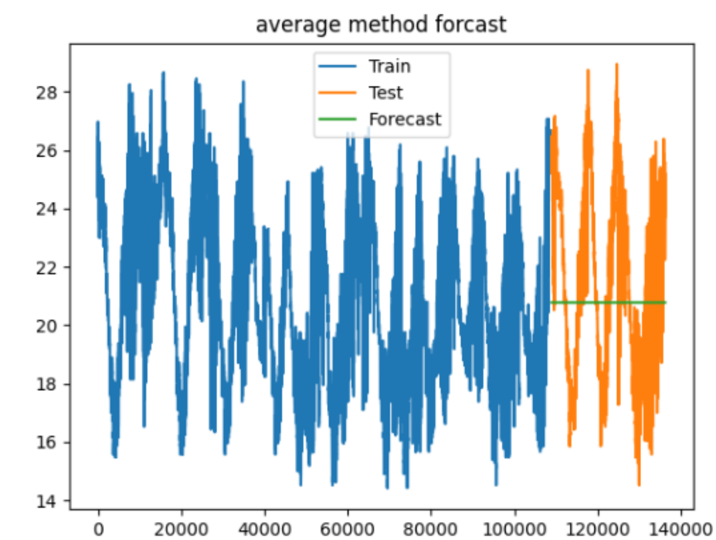
R squared increased and all the std errors are so low. Then the best features are Latitude, longitude and depth.

Base Models

Average Method:

These are the result for the average method prediction and forcast.



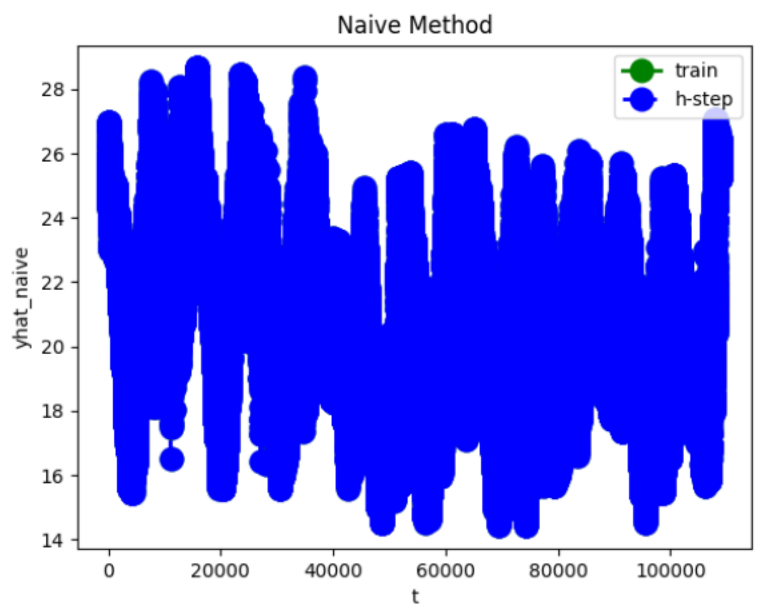


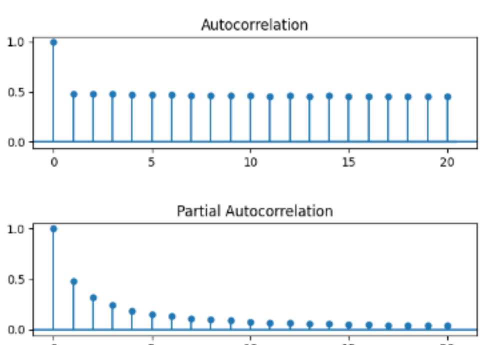
```
The Mean Square Error of predict for Average method is:  8.866157206896837
The Mean Square Error of forcast for Average method is:  8.937020987458277
        lb_stat  lb_pvalue
100  104.303532   0.364279
the estimated variance error of residual for average is : 8.865549730261344
the estimated variance error of forcast for average is : 8.937397745286397
the ratio bet var of pred and forcast in average: 0.9919609692806897
>>> forcast=[]
```

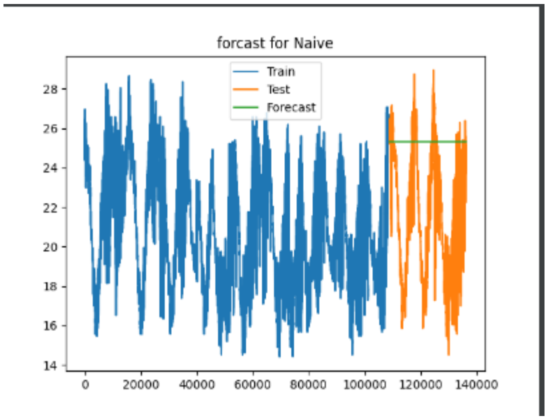
This result shows that the residual error is white which is almost 0.3 which is greater than 0.05.

Also the error is less than holt-trend and the ratio is almost 0.99 percent which is so close to 1.

Naive Method:



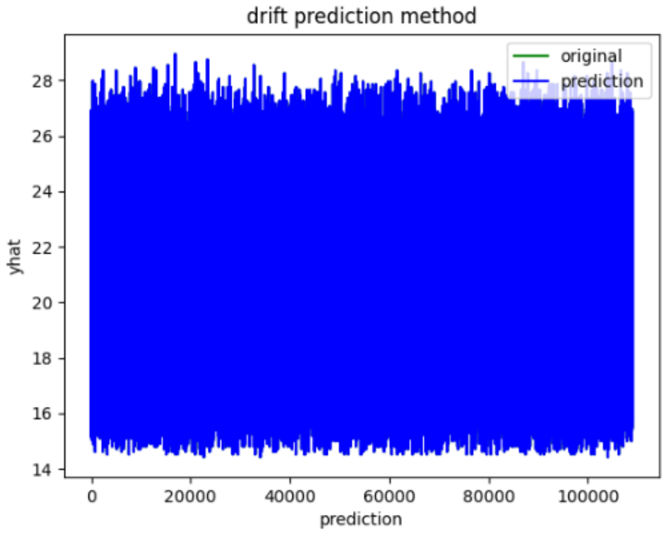


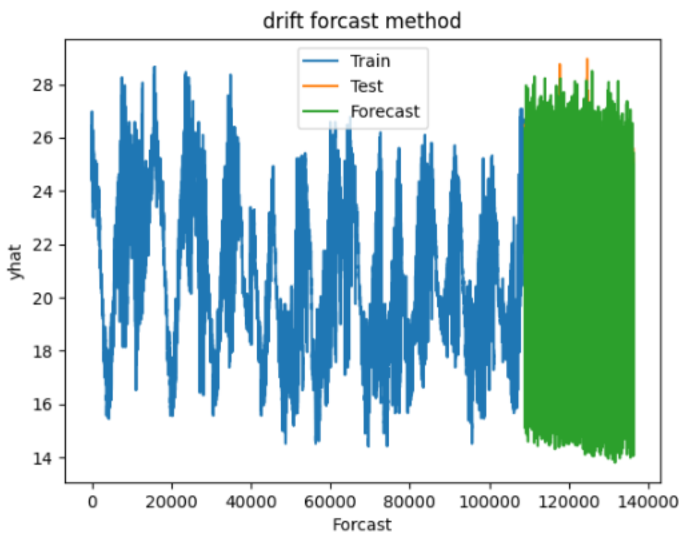


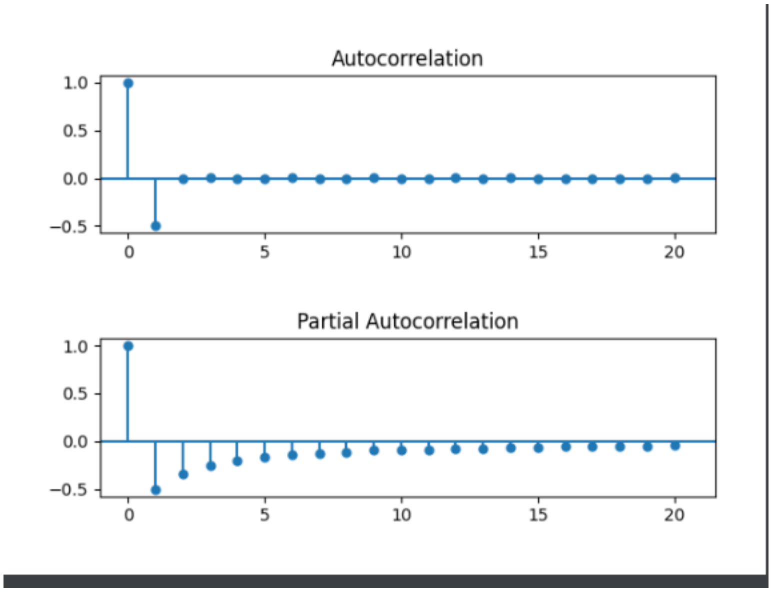

```
The Mean Square Error of predict for Naive method is:  0.16282761782385174
The Mean Square Error of predict for Naive method is:  26.059722387540834
          lb_stat  lb_pvalue
100  2.054394e+06        0.0
The residual is not white
From acorr_ljungbox test
[241.97813781]
[4.71337477e-40]
the estimated variance error of residual for naive is : 17.2968652246578
the estimated variance error of forcast for naive is : 8.937397745286397
the ratio bet var of pred and forcast in naive: 1.9353357339142934
```

These result is for Naïve method, as you see the p value of Q is less than 0.05 which with the Q square test , we can realize that It is not white, the estimated variance error is high and the ratio is too far from 1. Then it is not a good model.

Drift Method:


drift prediction method
original
prediction
yhat
prediction

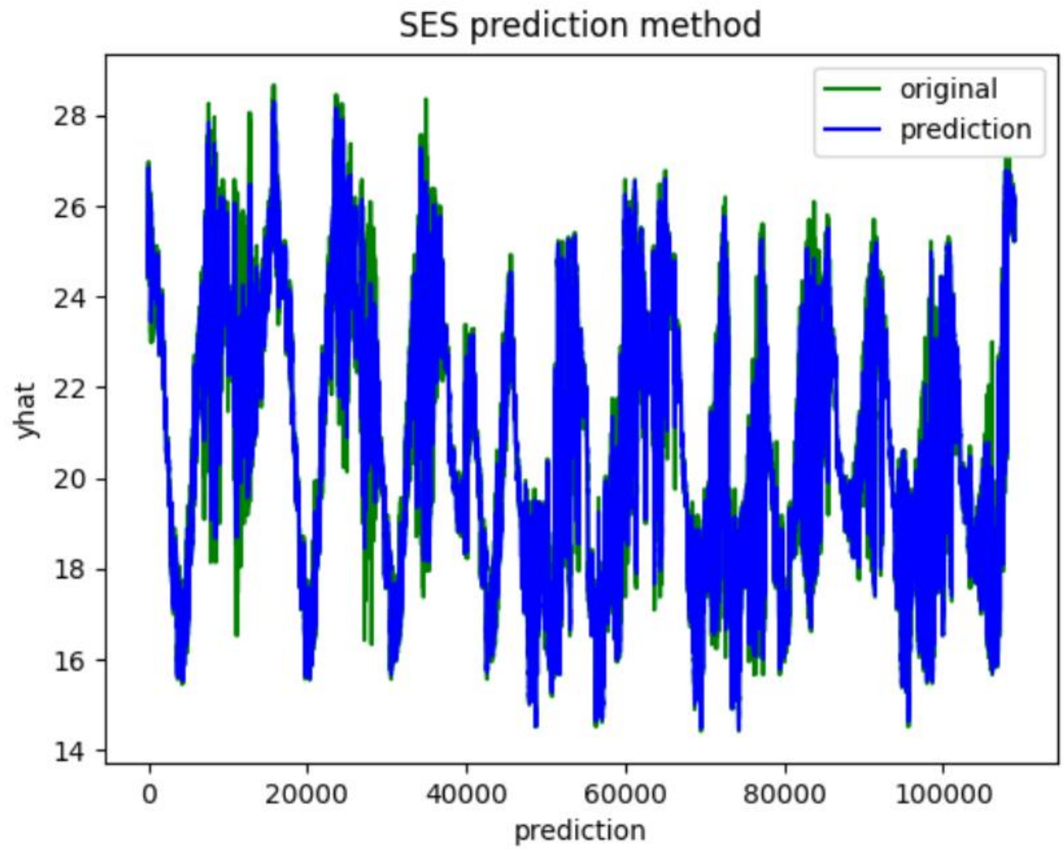

drift forcast method
Train
Test
Forecast
yhat
Forcast

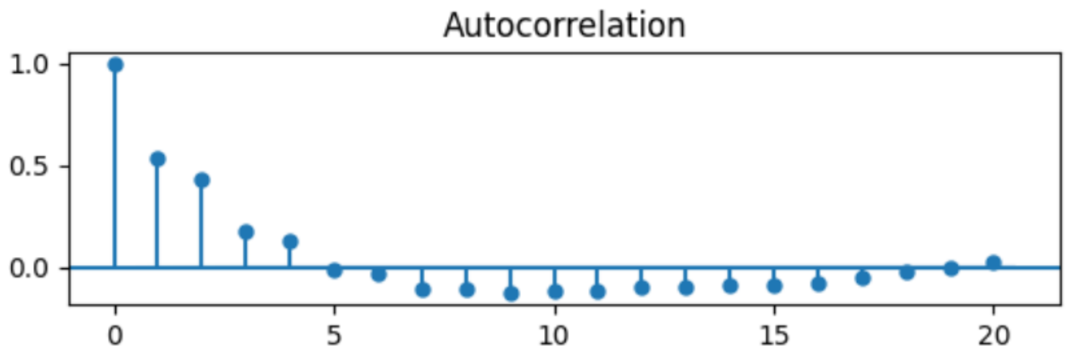


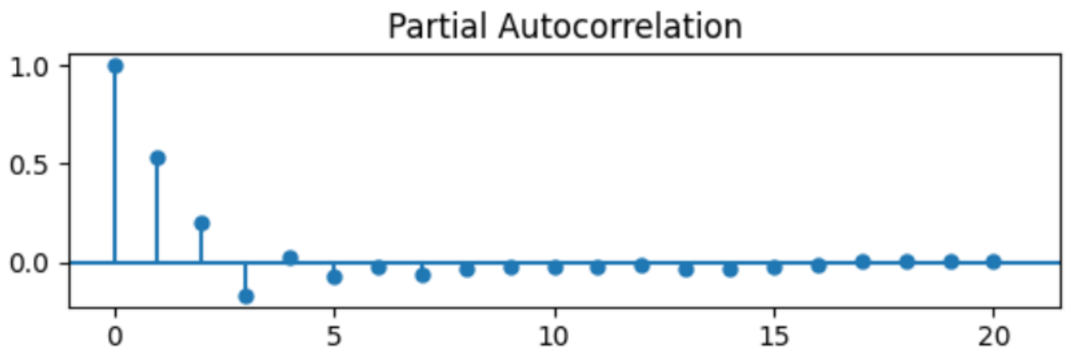

```
The Mean Square Error of predict for drift method is:  17.816213318091755
          lb_stat   lb_pvalue
100  27507.384804        0.0
The residual is not white
From acorr_ljungbox test
[27377.02087613]
[0.]
The Mean Square Error of forcast for drift method is:  17.816213318091755
the estimated variance error of residual for drift is : 17.81621329713728
the estimated variance error of forcast for drift is : 17.904191825168713
the ratio bet var of pred and forcast in drift: 0.9950861491604577
```

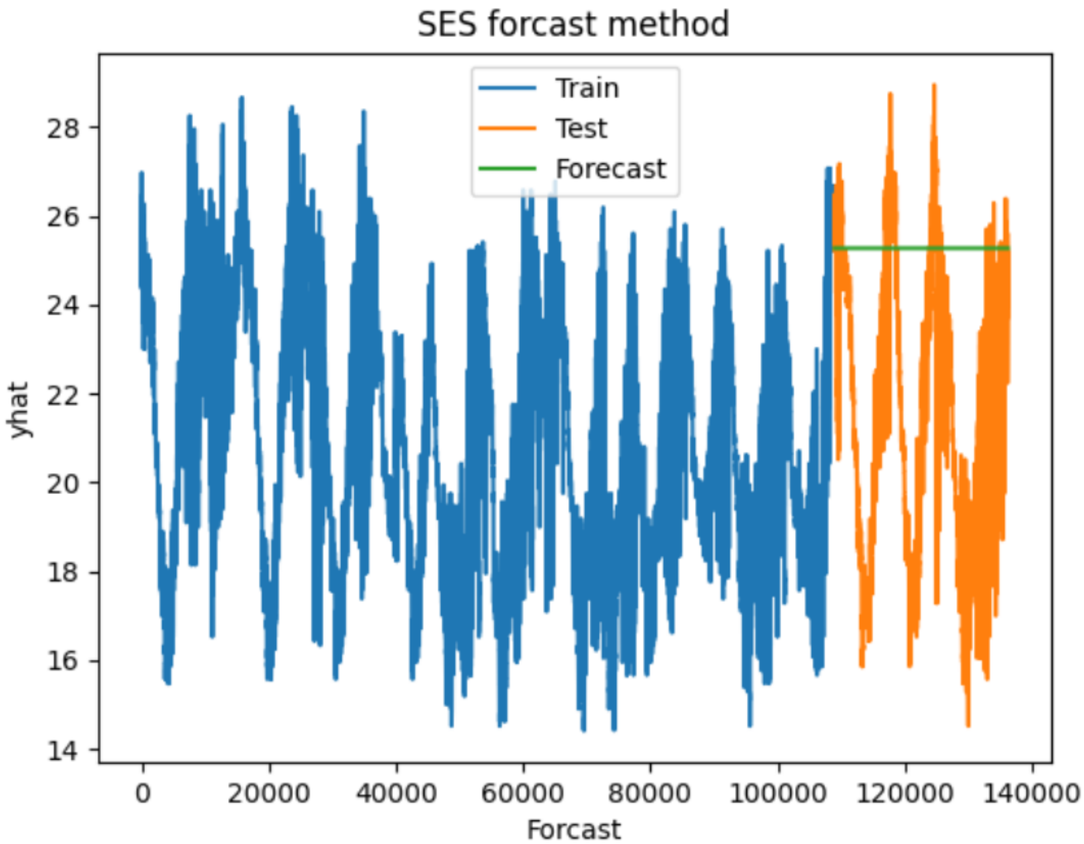
The residual error for the Drift method also is not white and the errors are so high, however the ratio is close to one, but overall it's not a good model.

SES Method:


SES prediction method
original
prediction
yhat
prediction
28
26
24
22
20
18
16
14
0
20000
40000
60000
80000
100000


Autocorrelation
1.0
0.5
0.0
0
5
10
15
20

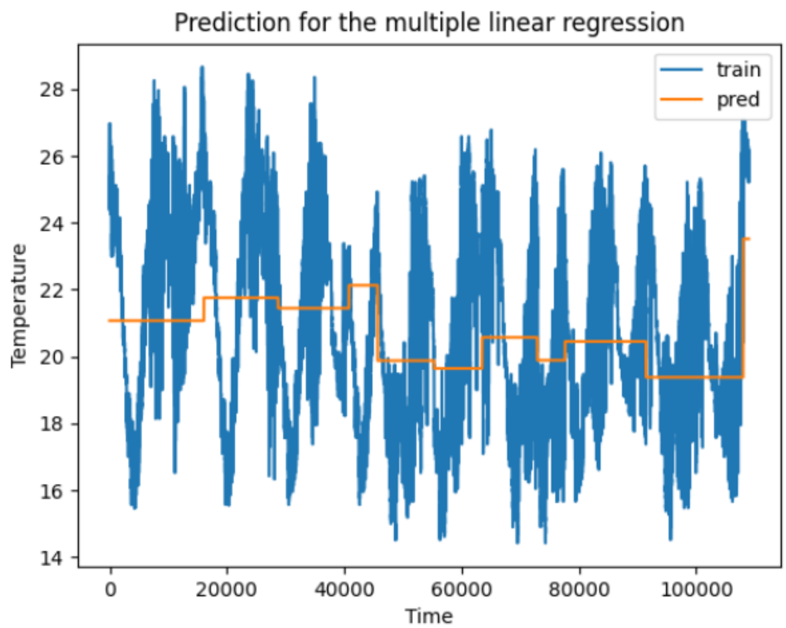

Partial Autocorrelation
1.0
0.5
0.0
0
5
10
15
20

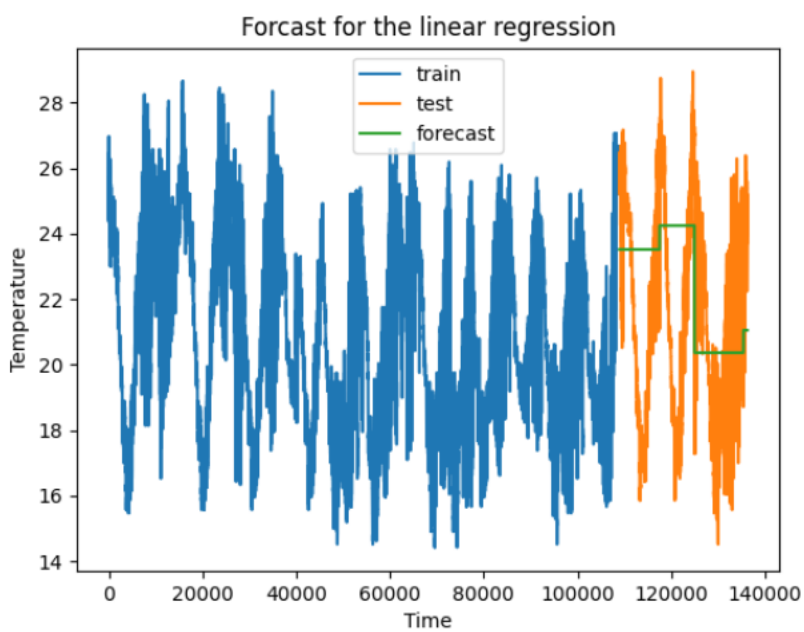


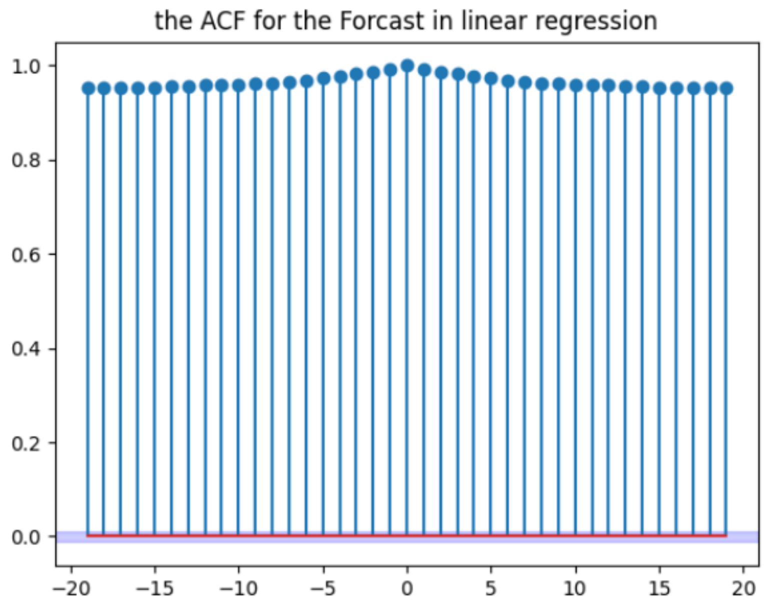

```
The Mean Square Error of predict for SES method is:  0.2382577614929861
         lb_stat  lb_pvalue
100  84174.736162        0.0
The residual is not white
From acorr_ljungbox test
[69208.87152208]
[0.]
The Mean Square Error of forcast for SES method is:  29.127526756795444
the estimated variance error of residual for SES is : 0.2382577605717134
the estimated variance error of forcast for SES is : 8.937397745286397
the ratio bet var of pred and forcast in SES: 0.026658515975455058
```

These are the results for SES method, this model is not white also, the mean square error is so high and the ratio is so far from 1. Then it's not the good model at all.

Multiple Linear Regression Method:








```
                            OLS Regression Results
=======================================================================================
Dep. Variable:                      y   R-squared (uncentered):                   0.982
Model:                            OLS   Adj. R-squared (uncentered):              0.982
Method:                 Least Squares   F-statistic:                          2.008e+06
Date:                Mon, 02 May 2022   Prob (F-statistic):                        0.00
Time:                        10:43:05   Log-Likelihood:                     -2.6587e+05
No. Observations:              108970   AIC:                                  5.318e+05
Df Residuals:                  108967   BIC:                                  5.318e+05
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Latitude      -2.3770      0.025    -96.356      0.000      -2.425      -2.329
Longitude      1.8015      0.014    127.731      0.000       1.774       1.829
Depth         -0.0978      0.001    -74.540      0.000      -0.100      -0.095
==============================================================================
Omnibus:                    11657.899   Durbin-Watson:                   0.021
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3277.392
Skew:                           0.027   Prob(JB):                         0.00
Kurtosis:                       2.152   Cond. No.                         193.
```

```
the Mean square error is: 7.70528191915556
          lb_stat  lb_pvalue
100  8.456688e+06        0.0
ID
326911    1.701455
326914    1.701455
326917    1.701455
326920    1.798455
326923    1.895455
            ...
408625    3.492314
408628    3.588314
408631    3.588314
408634    3.878314
408637    3.878314
Length: 27243, dtype: float64
          lb_stat  lb_pvalue
100  2.320774e+06        0.0
the Mean square error is: 11.993950849917523
the estimated variance error of residual for linea is : 7.705281833833325
the estimated variance error of forcast for linear is : 10.72304455843335
the ratio bet var of pred and forcast in linear: 0.7185722107042216
>>>
```

These are the results of the multiple linear regression.

From the ACF we can see that model is not white and also from the p value which is 0, the AIC is low in this model which is good but also the BIC is low. the R square getting increased after I removed the feature which shows the model got better, however it doesn't show it's a good model at all. The Mean square error and MSE is high and the ratio is almost 0.7 which is far from 1. Overall it's not a good model for the prediction.

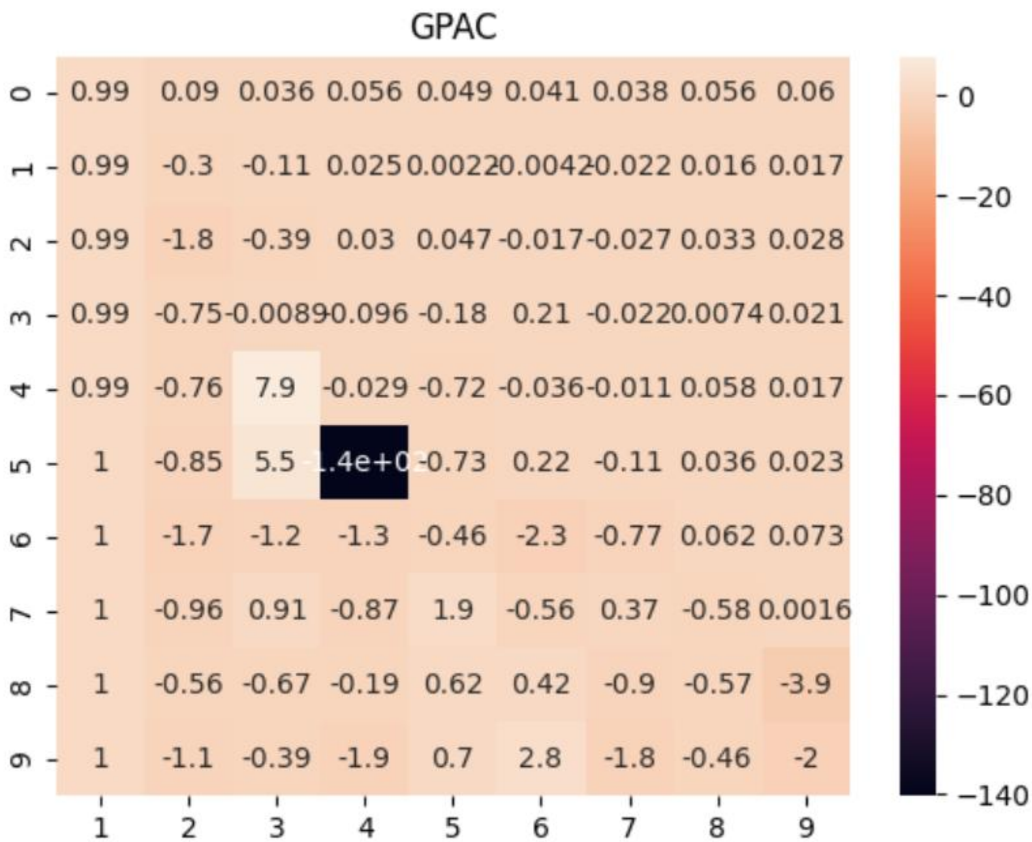
The Q value for this method is lb-stat here.

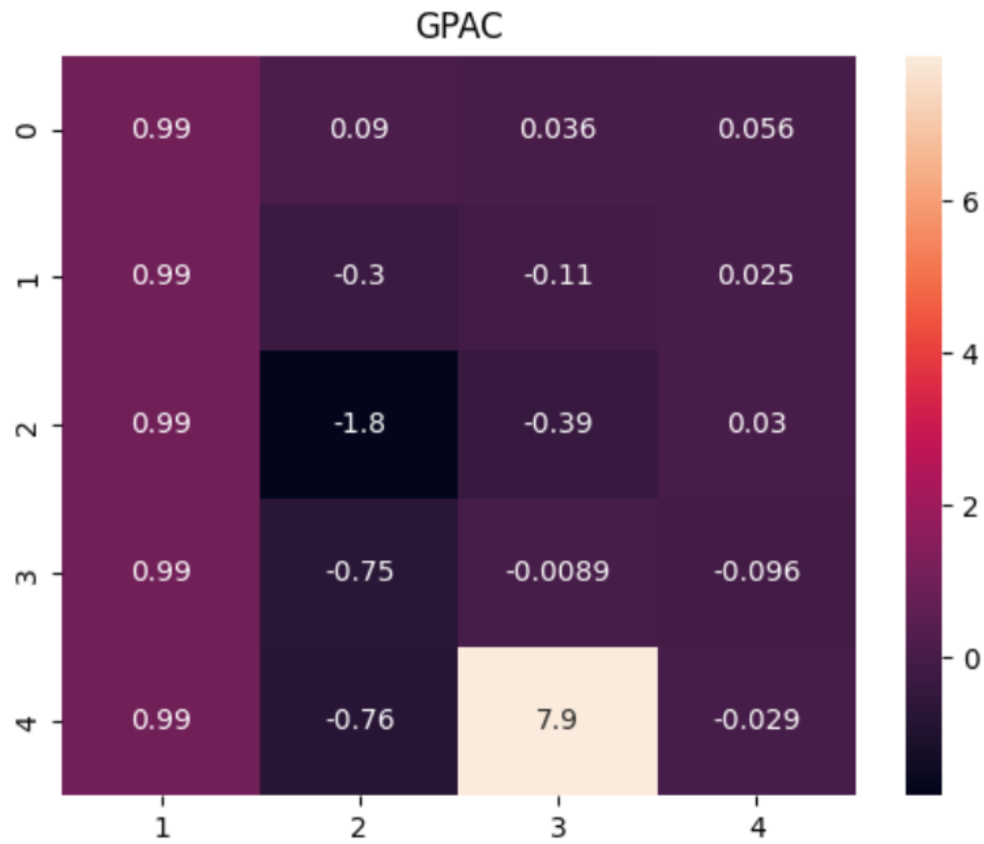
```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
c0            -2.3770      0.025    -96.356      0.000      -2.425      -2.329
c1             1.8015      0.014    127.731      0.000       1.774       1.829
c2            -0.0978      0.001    -74.540      0.000      -0.100      -0.095
==============================================================================
```

This is the result of T-test and as it shows all the rest variables are important and I am not going to remove anymore. The std error for all of them are low.

GPAC:

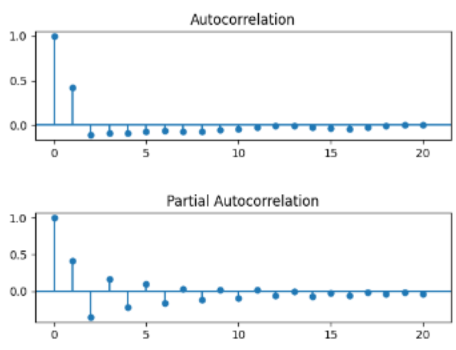
I calculated the GPAC(10,10) and GPAC(5,5) and there is the obvious pattern of 1 and 0 in this GPAC which shows the na of 1 and nb of 0.

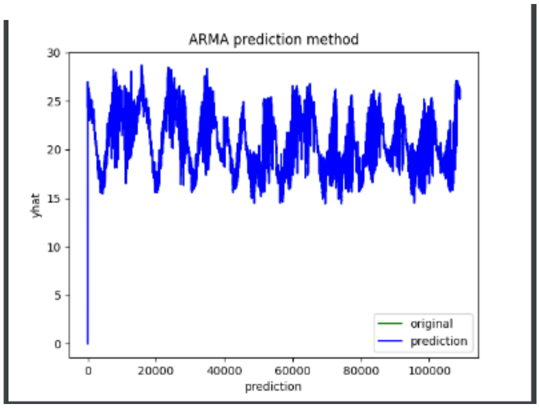


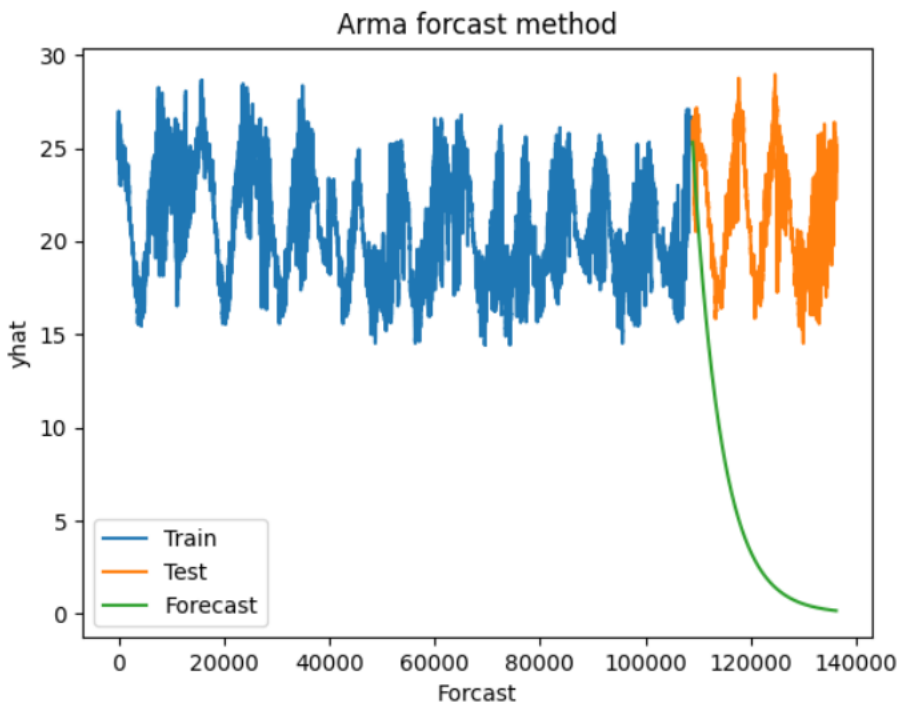


ARMA:

For the ARMA model I need to know the GPAC order which is 1, 0 for this raw dataset.(I haven't used the differenced one since my data is stationary).








```
          lb_stat  lb_pvalue
100  30458.815462        0.0
The residual is not white
From acorr_ljungbox test
[25561.72585702]
[0.]
The Mean Square Error of predict for ARMA is:  0.2964348919431837
The Mean Square Error of forcast for ARMA is:  304.76433641814725
the variance of residual error 1.2650828859939426e-06
the variance of forcast error 0.0002821356875401197
The residual is not white
From acorr_ljungbox test
[25561.72585702]
[0.]
Confidence interval:
0.9998160185026328: ar.L1.Temp    0.999704
Name: 0, dtype: float64
ratio residual forcast 0.004483952019767254
```

These are results for ARMA model, from the ACF of residual it seems that after the second lag it should be white, however it's not. The forecast plot shows that it's not forecasting good at all. The green line which is forecast is out completely. The mean square error of forecast is so high and the ratio is almost 0 which is so bad. ARMA is the worst model for me.

ARIMA:

Arima is not capable for this data since we are not using our differencing data since our data is stationary.

SARIMA:

Sarima also is not applicable for this data since this data has the weak seasonality and we use the SARIMA for the high seasonal data.

Levenberg Marquardt algorithm :

However I covered the LM algorithm during this semester, I preferred to use the stats model for displaying the parameters.

```
------------------------------------------------------------------------------
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1.Temp     0.9998   5.73e-05   1.75e+04      0.000       1.000       1.000
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.0002           +0.0000j            1.0002            0.0000
-----------------------------------------------------------------------------
```

It shows the coefficient of 0.9998 which is our parameter.

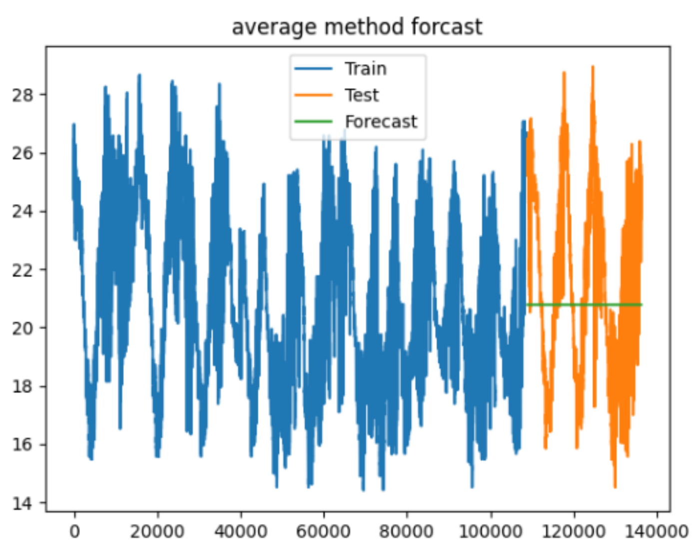
The model is not unbiased because the difference between 1(estimate parameter) and 0.9998 is not zero.

Final Model Selection:

	Q value	MSE residual	MSE Forecast	Er_Var_prediction	Er_Var_forcast	Variance ratio	residu White
Holt-trend	241.978138	8.5226	11.4932	9.0335	8.9389	1.0105	No
Average	104.303532	8.8661	8.9370	8.8655	8.93739774	0.9919	Yes
Naïve	2.054394e+06	0.1628	26.0597	17.2968	8.93739774	1.93	No
Drift	27507.384	17.8162	17.8162	17.8162	17.904191	0.9950	No
SES	84174.7361	0.2382	29.1275	0.238257	8.937397	0.02	No
OLS	8.456688e+06	7.7052	11.9939	7.705281	10.723044	0.7185	No
ARMA	30458.815462	0.296434	304.764	1.265082	0.0002821	0.0044	No

The best model of the table is Average since the residual error is white and the ratio is 0.99 which is close to and the mean square error forecast is less than the other models. But after that I would choose the Holt-trend, however it is not white.

Forecast Function of the Average:



The forecast method and h steps prediction have covered in the average part and as you see the forecast function is flat which make sense because the temperature of the sea underwater surface is around 24 most of the time.

Summary and Conclusion:

The best model for this dataset was Average method, the other types of model which may improve the performance is LSTM, which I haven't tried on this dataset yet.

Appendix:

```
import matplotlib.pyplot as plt
import pandas as pd
from statsmodels.tsa.stattools import adfuller
import os
import numpy as np
import seaborn as sns
import statsmodels
import matplotlib
import numpy as np
import statsmodels.api as sm
import statsmodels.tsa.arima_model
from sklearn.model_selection import train_test_split
from pandas import Series
from statsmodels.tsa.seasonal import STL
import statsmodels.tsa.holtwinters as ets
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from sklearn import metrics
from numpy import linalg as LA
import math
from scipy import signal
import warnings
warnings.filterwarnings('ignore')

data = pd.read_csv('underwater_temperature.csv',index_col='ID',
parse_dates=True,encoding= 'unicode_escape')
z=data.copy(deep=True)
df=z.iloc[::3,:]

print(df.isnull().sum().sum())
print(df.columns)
#Preprocessing
print(df['Site'].isnull().sum().sum())
print(df['Latitude'].isnull().sum().sum())
print(df['Longitude'].isnull().sum().sum())
print(df['Date'].isnull().sum().sum())
print(df['Time'].isnull().sum().sum())
print(df['Temp (°C)'].isnull().sum().sum()) #2 missing values

#preproccesing the Data›
df.rename(columns={'Temp (°C)':'Temp'},inplace=True)
mean_value=df['Temp'].mean()
df['Temp'].fillna(value=mean_value, inplace=True)
print(df['Temp'].isnull().sum().sum()) #2 missing values

#now we don't have any missing values
#plot the data vs time
sns.lineplot(data=df, x="Time",y="Temp")
plt.title('the dependency of temp and time')
plt.show()
#ACF/PACF of the dependent variable
from statsmodels.graphics.tsaplots import plot_acf , plot_pacf
```

```
def ACF_PACF_Plot(y,lags):
    acf = sm.tsa.stattools.acf(y, nlags=lags)
    pacf = sm.tsa.stattools.pacf(y, nlags=lags)
    fig = plt.figure()
    plt.subplot(211)
    plt.title('ACF/PACF of the raw data')
    plot_acf(y, ax=plt.gca(), lags=lags)
    plt.subplot(212)
    plot_pacf(y, ax=plt.gca(), lags=lags)
    fig.tight_layout(pad=3)
    plt.show()
ACF_PACF_Plot(df['Temp'],20)
print(pd.unique(df['Site']))
#because of PACF the Ar is going to be 1

corr=df.corr()
ax =sns.heatmap(corr)
plt.show()
df['Time']=df["Time"].str[:-3]
y=df['Temp']
x=df.drop(['Temp','Site','Date','Time'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
#rolling mean
def rolling_mean_var(x):
    df_rolling = pd.DataFrame()
    mean=[]
    var=[]
    for i in range(len(x)):
        mean.append(x[0:i+1].mean())
        var.append(x[0:i+1].var())
    df_rolling['mean'] = mean
    df_rolling['var'] = var
    df_rolling.fillna(0, inplace=True)
    #====
    fig = plt.figure(figsize=(10, 8))
    plt.subplot(2, 1, 1)
    plt.plot('mean', data=df_rolling, label="Mean", color='red')
    plt.title("Rolling Mean & Variance")
    plt.xticks([])
    plt.grid()
    plt.ylabel('mean')
    plt.legend()
    plt.subplot(2, 1, 2)
    plt.plot('var', data=df_rolling, label="variance", color='blue')
    plt.grid()
    plt.ylabel('variance')
    plt.legend()
    plt.show()
rolling_mean_var(df['Temp'])

#Stationary
#ADF test for temp

x=df['Temp'].values
resault=adfuller(x)
print('ADF Statistic :%f'% resault[0])
print('p_value: %f'% resault[1])
```

```
print('Critical Values:')
for key, value in resault[4].items():
    print('\t%s:%.3f'%(key,value))
if resault[0] < resault[4]['5%']:
    print('Temp is Stationary')
else:
    print('Temp is not stationory')
#KPS Test
from statsmodels.tsa.stattools import kpss
x=df['Temp'].values
def kpss_test(x):
    print ('Results of KPSS Test:')
kpsstest = kpss(x, regression='c', nlags="auto")
kpss_output = pd.Series(kpsstest[0:3], index=['Test Statistic','p-
value','LagsUsed'])
for key,value in kpsstest[3].items():
    kpss_output['Critical Value (%s)'%key] = value
print('KPSS output for Temp is :')
print (kpss_output)

#first diferencing because the KPSS is not stationary
def f_difference(dataset, interval):
    diff = []

    for i in range(interval, len(dataset),interval):
        value = dataset[i] - dataset[i - interval]
        if i == 1:
            diff.append(0)
        elif i == 2 and interval == 2:
            diff.append(0)
            diff.append(0)
        elif i == 3 and interval == 3:
            diff.append(0)
            diff.append(0)
            diff.append(0)

        diff.append(value)
    return diff
x=f_difference(df['Temp'].values,1)

#Kpss after first differnecing
def kpss_test(x):
    print ('Results of KPSS Test:')
kpsstest = kpss(x, regression='c', nlags="auto")
kpss_output = pd.Series(kpsstest[0:3], index=['Test Statistic','p-
value','LagsUsed'])
for key,value in kpsstest[3].items():
    kpss_output['Critical Value (%s)'%key] = value
print('KPSS output for Temp is :')
print (kpss_output)

#Decomposition

Temp = df['Temp']
# df.replace(to_replace =["/"],
#              value ="-")
dates= pd.date_range(start='2013-02-20 11:40:00', periods=len(df),freq='1H')
```

```
temp_volume= pd.Series(df['Temp'].ravel(),index=dates)
STL = STL(temp_volume)
res = STL.fit()
fig = res.plot()
plt.xlabel("Time (Year)")
plt.suptitle('STL Decomposition', y=1.05)
plt.show()
#
# #seasonality and trend
T=res.trend
S=res.seasonal
R=res.resid
adj_seasonal=Temp - S
plt.plot(Temp,label='original')
plt.plot(adj_seasonal,label='Seasonality Adjusted')
plt.xlabel('time')
plt.ylabel('Temp')
plt.title('seasonality adjused data')
plt.legend()
plt.show()
#strength of tren
F=np.maximum(0,1-np.var(R)/np.var(np.array(T)+np.array(R)))
print(f'the strength of Trend is{F}')
#dtrended
T=res.trend
S=res.seasonal
R=res.resid
detrended=Temp - T
plt.plot(Temp,label='original')
plt.plot(detrended,label='detrended')
plt.xlabel('time')
plt.ylabel('Temp')
plt.title('adjusted detrended')
plt.legend()
plt.show()
#strength of seasonality
F=np.maximum(0,1-np.var(R)/np.var(np.array(S)+np.array(R)))
print(f'the strength of seasonality is{F}')

#holt_trend
n1=len(y_train)
n2 = len(y_test)
# train= df['Temp'][:n1]
# test= df['Temp'][n1:]
model=ets.ExponentialSmoothing(y_train, trend='add', damped_trend=True,
seasonal=None).fit()
fitted= model.fittedvalues
forecast_holt= model.forecast(steps=len(y_test))
residual_error= y_train[1:].values-fitted[:-1].values
ACF_PACF_Plot(residual_error,20)
plt.hist(residual_error)
plt.show()
Q_holt=sm.stats.acorr_ljungbox(residual_error, lags=[20],return_df=True)
print(Q_holt)
plt.plot(list(range(0,n1)),y_train,label='Train')
plt.plot(list(range(n1,n1+n2)),y_test,label='Test')
plt.plot(list(range(n1,n1+n2)),forecast_holt,label='Forecast')
```

```
plt.title('forcast function for holt_trend')
plt.legend()
plt.show()
forcast_error_holt=y_test[2:].values-forecast_holt[:-2].values
#when the p value for Q is more than 0.05 is white which is not here now
#Chi square test
from scipy.stats import chi2
def chi_test(na,nb,lags,Q,e):
    DOF=lags-na-nb
    alpha=0.01
    chi_critical=chi2.ppf(1-alpha,DOF)
    if Q<chi_critical:
        print('The residuals are white')
    else:
        print('The residual is not white')
    lbvalue,pvalue=sm.stats.acorr_ljungbox(e,lags=[lags])
    print('From acorr_ljungbox test')
    print(lbvalue)
    print(pvalue)
chi_test(1,0,20,1257.221097,residual_error)
print('the estimated variance error of residual for holt is
:',np.var(residual_error))
print('the estimated variance error of forcast for holt is
:',np.var(forcast_error_holt))
ratio_variance_holt=np.var(residual_error)/np.var(forcast_error_holt)
print('the ratio bet var of pred and forcast in holt-
trend:',ratio_variance_holt)

#MSE holt_tren for pred
mse_holt_trend_pred= metrics.mean_squared_error(fitted[1:],y_train[:-1])
print( "The Mean Square Error of predict for holt_trend is: " ,
(mse_holt_trend_pred))

#MSE holt_tren for forcast
mse_holt_trend_forcast=
metrics.mean_squared_error(forecast_holt[2:],y_test[:-2])
print( "The Mean Square Error of forcast for holt-trend method is: " ,
(mse_holt_trend_forcast))

#Feature Selection
x=df.drop(['Temp','Site','Date','Time'],axis=1)
y=df['Temp']
x = sm.add_constant(x)
y=y.values
model=sm.OLS(y,x).fit()
print(model.summary())
x=x.drop(columns='const')
model2=sm.OLS(y,x).fit()
print(model2.summary())
#we remove contant because it's high std error
print('This is the final model',model2.summary())

#Average Method
#Average method and predict
t1= list(range(len(x_train))) #x_train
t2= list(range(len(x_train),len(x_test))) #x_test
n1=len(y_train)
```

```
n2 = len(y_test)
yhath_Average=[]
yhath_Average_val = np.mean(y_test)
#Prediction
for i in range(1,n1+1):
    yhath_Average.append(yhath_Average_val)
plt.plot(t1, y_train, color='green', marker='o', linestyle='dashed',
linewidth=2, markersize=12, label='prediction')
# plt.plot(t2, y_test, color='red', marker='x', linestyle='dashed',
linewidth=2, markersize=12,label='forcast')
plt.plot(t1, yhath_Average, color='blue', marker='o', linestyle='dashed',
linewidth=2, markersize=12, label='1-step')
plt.legend()
plt.xlabel('x')
plt.ylabel('yhat')
plt.title('average prediction method')
plt.show()
#calculating error for the average method
mse_Average_pred= metrics.mean_squared_error(yhath_Average[1:],y_train[:-1])
print( "The Mean Square Error of predict for Average method is: " ,
(mse_Average_pred))

#forcast
forcast=[]
for i in range(n2):
    forcast.append(yhath_Average[-1])
forcast=np.array(forcast)

train= df['Temp'][:n1]
test= df['Temp'][n1:]
plt.plot(list(range(0,n1)),train,label='Train')
plt.plot(list(range(n1,n1+n2)),test,label='Test')
plt.plot(list(range(n1,n1+n2)),forcast,label='Forecast')
plt.legend()
plt.title('average method forcast')
plt.show()
#calculating error for the average method forcast
mse_Average_forcast= metrics.mean_squared_error(forcast[2:],y_test[:-2])
print( "The Mean Square Error of forcast for Average method is: " ,
(mse_Average_forcast))

#Q value for average
residual_error_average=y_train[1:].values-yhath_Average[:-1]
ACF_PACF_Plot(residual_error_average,20)
Q=sm.stats.acorr_ljungbox(residual_error_average, lags=[100],return_df=True)
print(Q)
#forcast error for average
forcast_error_average=y_test[2:].values-forcast[:-2]
#variance of residual and forcast and the rate
print('the estimated variance error of residual for average is
:',np.var(residual_error_average))
print('the estimated variance error of forcast for average is
:',np.var(forcast_error_average))
ratio_variance_average=np.var(residual_error_average)/np.var(forcast_error_av
erage)
print('the ratio bet var of pred and forcast in
average:',ratio_variance_average)
```

```
#Naïve Method
pred_naive=[]
for T in range(len(train)):
    pred_naive.append(train.ravel()[T])
pred_naive=np.array(pred_naive)
#Plot NAive_prediction

t1= list(range(len(x_train))) #x_train
t2= list(range(len(y_test))) #x_test
plt.plot(t1, train, color='green', marker='o',  linewidth=2, markersize=12,
label='train')
plt.plot(t1, pred_naive, color='blue', marker='o', linestyle='dashed',
linewidth=2, markersize=12, label='h-step')
plt.legend()
plt.xlabel('t')
plt.ylabel('yhat_naive')
plt.title('Naive Method')
plt.show()
#error for naive_prediction
mse_Naive= metrics.mean_squared_error(train[1:],pred_naive[:-1])
print( "The Mean Square Error of predict for Naive method is: " ,
(mse_Naive))

#forecast
forcast_naive=[]
for i in range(n2):
    forcast_naive.append(pred_naive.ravel()[-1])
forcast=np.array(forcast_naive)

train= df['Temp'][:n1]
test= df['Temp'][n1:]
plt.plot(list(range(0,n1)),train,label='Train')
plt.plot(list(range(n1,n1+n2)),test,label='Test')
plt.plot(list(range(n1,n1+n2)),forcast,label='Forecast')
plt.title('forcast for Naive')
plt.legend()
plt.show()
#mse error for naive_Forcast
mse_Naive_forcast= metrics.mean_squared_error(test[2:],forcast_naive[:-2])
print( "The Mean Square Error of predict for Naive method is: " ,
(mse_Naive_forcast))

#Q value for Naive
residual_error_naive=y_train[1:].values-pred_naive[:-1]
ACF_PACF_Plot(residual_error_naive,20)
Q_naive=sm.stats.acorr_ljungbox(residual_error_naive,
lags=[100],return_df=True)
print(Q_naive)
chi_test(1,0,20,2.084765e+06,residual_error)
#forcast error for naive
forcast_error_naive=y_test[2:].values-forcast[:-2]
#variance of residual and forcast and the rate
print('the estimated variance error of residual for naive is
:',np.var(residual_error_naive))
print('the estimated variance error of forcast for naive is
:',np.var(forcast_error_naive))
```

```
ratio_variance_naive=np.var(residual_error_naive)/np.var(forcast_error_naive)
print('the ratio bet var of pred and forcast in naive:',ratio_variance_naive)

#Drift Method
h=1
drift_prediction=[]
for i in range (len(y_train.values)):
    drift_prediction.append(y_train.values[i]+h*(y_train.values[i]-
y_train.values[0])/i)
drift_prediction=drift_prediction[1:-1]
plt.plot(y_train.values, color='green', label='original')
plt.plot(np.array(drift_prediction), color='blue',label='prediction')
plt.legend()
plt.xlabel('prediction')
plt.ylabel('yhat')
plt.title('drift prediction method')
plt.show()

# mse error for drift
mse_Drift= metrics.mean_squared_error(y_train[2:],drift_prediction)
print( "The Mean Square Error of predict for drift method is: " ,
(mse_Drift))

#Q value for drift
residual_error_drift=y_train[2:].values-drift_prediction
ACF_PACF_Plot(residual_error_drift,20)
Q_drift=sm.stats.acorr_ljungbox(residual_error_drift,
lags=[100],return_df=True)
print(Q_drift)
chi_test(1,0,20,27782.818899,residual_error_drift)

#forcast for drift
drift_forcast=[]
for i in range (len(y_test.values)):
    drift_forcast.append(y_train.values[i]+h*(y_train.values[-1]-
y_train.values[0])/len(y_train-1))
    h+=1

plt.plot(list(range(0,n1)),train.values,label='Train')
plt.plot(list(range(n1,n1+n2)),test.values,label='Test')
plt.plot(list(range(n1,n1+n2)),np.array(drift_forcast),label='Forecast')

plt.legend()
plt.xlabel('Forcast')
plt.ylabel('yhat')
plt.title('drift forcast method')
plt.show()

# mse error for drift of forcast
mse_Drift_forcast= metrics.mean_squared_error(y_test,drift_forcast)
print( "The Mean Square Error of forcast for drift method is: " ,
(mse_Drift))

#forcast error for drift
forcast_error_drift=y_test[2:].values-drift_forcast[:-2]
#variance of residual and forcast and the rate
print('the estimated variance error of residual for drift is
```

```
:',np.var(residual_error_drift))
print('the estimated variance error of forcast for drift is
:',np.var(forcast_error_drift))
ratio_variance_drift=np.var(residual_error_drift)/np.var(forcast_error_drift)
print('the ratio bet var of pred and forcast in drift:',ratio_variance_drift)


#SES predict
predict_SES=[train.tolist()[0]]
alfa=0.4
for i in range(len(train.values)):
    predict_SES.append(alfa*train.values[i]+(1-alfa)*predict_SES[i-1])

plt.plot(train.values, color='green', label='original')
plt.plot(np.array(predict_SES), color='blue',label='prediction')
plt.legend()
plt.xlabel('prediction')
plt.ylabel('yhat')
plt.title('SES prediction method')
plt.show()

#error for SES for predection
mse_SES= metrics.mean_squared_error(train[1:],predict_SES[1:-1])
print( "The Mean Square Error of predict for SES method is: " , (mse_SES))


#Q value for SES
residual_error_SES=train[1:].values-predict_SES[1:-1]
ACF_PACF_Plot(residual_error_SES,20)
Q_SES=sm.stats.acorr_ljungbox(residual_error_SES, lags=[100],return_df=True)
print(Q_SES)
chi_test(1,0,20,2.141921e+06,residual_error_SES)

#SES Forcast
SES_forcast=[]
for i in range(len(test)):
  SES_forcast.append(predict_SES[-1])

plt.plot(list(range(0,n1)),train.values,label='Train')
plt.plot(list(range(n1,n1+n2)),test.values,label='Test')
plt.plot(list(range(n1,n1+n2)),np.array(SES_forcast),label='Forecast')
plt.legend()
plt.xlabel('Forcast')
plt.ylabel('yhat')
plt.title('SES forcast method')
plt.show()

# mse error for drift of forcast
mse_SES_forcast= metrics.mean_squared_error(y_test,SES_forcast)
print( "The Mean Square Error of forcast for SES method is: " ,
(mse_SES_forcast))

#forcast error for SES
forcast_error_SES=y_test[2:].values-SES_forcast[:-2]
#variance of residual and forcast and the rate
print('the estimated variance error of residual for SES is
:',np.var(residual_error_SES))
```

```
print('the estimated variance error of forcast for SES is
:',np.var(forcast_error_SES))
ratio_variance_SES=np.var(residual_error_SES)/np.var(forcast_error_SES)
print('the ratio bet var of pred and forcast in SES:',ratio_variance_SES)


#Multiple linear Regression
X=df[['Latitude','Longitude','Depth']]
Y=df['Temp'].values
X_train,X_test,Y_train,Y_test=train_test_split(X,Y, shuffle= False,
test_size=0.2)

H=np.matmul(X_train.values.T,X_train.values)
print(H)
s,d,v = np.linalg.svd(H)
print('SingularValues =',d)

Condition_no=LA.cond(X_train)
print(f'Condition Number for train set: {Condition_no}')

model=sm.OLS(Y_train,X_train).fit()
print(model.summary())


#prediction of multiple linear regression
prediction=model.predict(X_train)
import matplotlib.pyplot as plt
plt.plot(list(range(len(Y_train))),Y_train,label='train')
# plt.plot(list(range(len(y_train),len(y_test))),Y_test,label='test_price')
plt.plot(list(range(len(Y_train))),prediction,label='pred')
plt.legend()
plt.xlabel('Time')
plt.ylabel('Temperature')
plt.title('Prediction for the multiple linear regression')
plt.show()
#Forcast_linear model
#Forcast of multiple linear regression
forecast=model.predict(X_test)
import matplotlib.pyplot as plt
plt.plot(list(range(len(y_train))),Y_train,label='train')
plt.plot(list(range(len(y_train),len(y_train)+len(y_test))),Y_test,label='tes
t')
plt.plot(list(range(len(y_train),len(y_train)+len(y_test))),forecast,label='f
orecast')
plt.legend()
plt.xlabel('Time')
plt.ylabel('Temperature')
plt.title('Forcast for the linear regression')
plt.show()
#error MSE
MSE=np.square(np.subtract(Y_train[1:],prediction[:-1])).mean()
print('the Mean square error is:',MSE)
residual_error_linear=Y_train[1:]-prediction[:-1]
Q_linear=sm.stats.acorr_ljungbox(residual_error_linear,
lags=[100],return_df=True)
print(Q_linear)
def autocorrelation(y, k):
```

```
    y_mean = np.mean(y)
    T = len(y)
    resultnumber = 0
    res_den = 0
    for t in range(k,T):
        resultnumber += (y[t] - y_mean) * (y[t-k] - y_mean)

    for t in range(0,T):
        res_den += (y[t] - y_mean)*(y[t] - y_mean)

    res = resultnumber/res_den
    return res

def auto_corr_cal(y, k):
    res = []
    for t in range(0, k):
        result = autocorrelation(y, t)
        res.append(result)
    return res

##forcast error for linear regression
Forcast_error=Y_test-forecast
print(Forcast_error)
Q_linear_forcast=sm.stats.acorr_ljungbox(Forcast_error,
lags=[100],return_df=True)
print(Q_linear_forcast)
#error for the forcast of linear

MSE_linear_forcast=np.square(np.subtract(Y_test[1:],forecast.values[:-
1])).mean()
print('the Mean square error is:',MSE_linear_forcast)

ACF_3_Forecast=auto_corr_cal(Forcast_error.values, 20)
ry= ACF_3_Forecast[::-1][:-1] + ACF_3_Forecast
ac= ACF_3_Forecast
plt.stem(np.linspace(-((len(ac))-1), len(ac)-1, (len(ac)*2-1), dtype=int),
ry, markerfmt='o')
plt.axhspan(-(1.96/np.sqrt(len(Y_test))), (1.96/np.sqrt(len(Y_test))),
alpha=0.2, color='blue')
plt.title('the ACF for the Forcast in linear regression')
plt.show()

#estimated variance
def estimated_variance(T,K,E):#K number of factors # T number of total
samples
    a=1/(T-K-1)
    sum_e=0
    for i in range(1,T):
        sum_e=sum_e+(E[i]**2)
    e_v=np.square(a*sum_e)
    return e_v
print('the estimated variance error of residual for linea is
:',np.var(residual_error_linear))
print('the estimated variance error of forcast for linear is
:',np.var(Forcast_error.values))
ratio_variance_linear=np.var(residual_error_linear)/np.var(Forcast_error.valu
es)
```

```
print('the ratio bet var of pred and forcast in
linear:',ratio_variance_linear)

#T_Test
print(model.summary())
A=np.identity(len(model.params))
print('f test result is',model.f_test(A))
print('t test for the model is',model.t_test(A))

#ACF Function
def autocorrelation(y, k):
    y_mean = np.mean(y)
    T = len(y)
    resultnumber = 0
    res_den = 0
    for t in range(k,T):
        resultnumber += (y[t] - y_mean) * (y[t-k] - y_mean)

    for t in range(0,T):
        res_den += (y[t] - y_mean)*(y[t] - y_mean)

    res = resultnumber/res_den
    return res

def auto_corr_cal(y, k):
    res = []
    for t in range(0, k):
        result = autocorrelation(y, t)
        res.append(result)
    return res
ACF_for_raw=auto_corr_cal(y,100)
ac= ACF_for_raw
acc=ac[::-1]+ac[1:]

#GPAC
#den and NUm
def phi_kk(j,k,ry):
    num = [];a =[];b = []
    den = []
    M = math.ceil(len(ry)/2)
    if k==1:
        phi_kk = ry[M+j]/ry[M+j-1]
    else:
        for m in range(k):

            den.append(ry[M + j + m-1: M + j + m - k-1: -1])
            b.append(ry[M + j+ m])

        den = np.array(den)
        x = np.array(b)
        c = np.transpose([x])
        a = den[...,0:k-1]
        num = np.concatenate((a,c),axis=1)

        if np.linalg.det(den)!=0:
            phi_kk = np.linalg.det(num)/np.linalg.det(den)
        else:
```

```
            phi_kk = 'inf'
    return phi_kk
def pm(s):
    j,k = s.shape
    GPAC = pd.DataFrame(s)
    GPAC.columns = range(1, k+1)
    return GPAC

def Cal_GPAC(ryy, j, k):
    phi = np.zeros((j,k-1))
    for kk in range(1,k):
        for jj in range(j):
            phi_kk1= phi_kk(jj,kk,ryy)
            phi[jj][kk-1]=phi_kk1
    return pm(phi)
gpac_table= Cal_GPAC(acc,10,10)
sns.heatmap(gpac_table,annot=True)
plt.title('GPAC')
plt.show()
gpac_table= Cal_GPAC(acc,5,5)
sns.heatmap(gpac_table,annot=True)
plt.title('GPAC')
plt.show()

#The pattern is 1 and 0 for the GPAc
#ARMA
na=1 #AR auto correlation
nb=0 #moving Average
y=train

model=sm.tsa.ARMA(y,(na,nb)).fit(trend='nc',disp=0)

for i in range(na):
    print('The AR coefficient a{}'.format(i),'is:',model.params[i])
for i in range(nb):
    print('The MA coefficient a{}'.format(i),'is:',model.params[i+na])
print(model.summary())

#Prediction for ARMA
arma10_predict=model.predict(start=0,end=len(train)-1)
arma10_res=train.values[1:]-arma10_predict[:-1]
#Q for ARMA
Q_ARMA_pred=sm.stats.acorr_ljungbox(arma10_res, lags=[100],return_df=True)
print(Q_ARMA_pred)
#Forecast
arma10_test=model.predict(start=len(train), end=len(df))
arma10_fore_error=test.values[2:]-arma10_test[:-3]

#Chi square test
chi_test(1,0,20,30458.815462,arma10_res)
ACF_PACF_Plot(arma10_res,20)

#mse for ARMA for predict
mse_ARMA_predict= metrics.mean_squared_error(train[1:],arma10_predict[:-1])
print( "The Mean Square Error of predict for ARMA is: " , (mse_ARMA_predict))

#predict ARMA
```

```
plt.plot(train.values, color='green', label='original')
plt.plot(np.array(arma10_predict), color='blue',label='prediction')
plt.legend()
plt.xlabel('prediction')
plt.ylabel('yhat')
plt.title('ARMA prediction method')
plt.show()
##ARMA forcast
n1=len(y_train)
n2 = len(y_test)
train= df['Temp'][:n1]
test= df['Temp'][n1:]
plt.plot(list(range(n1)),train.values,label='Train')
plt.plot(list(range(n1,n1+n2)),test.values,label='Test')
plt.plot(list(range(n1,n1+n2)),np.array(arma10_test)[:-1],label='Forecast')
plt.legend()
plt.xlabel('Forcast')
plt.ylabel('yhat')
plt.title('Arma forcast method')
plt.show()
#MSE forcast for ARMA
mse_ARMA_forcast= metrics.mean_squared_error(test[2:],arma10_test[:-3])
print( "The Mean Square Error of forcast for ARMA is: " , (mse_ARMA_forcast))

#estimated variance of error for ARMA oredict and Forcast
print('the variance of residual
error',estimated_variance(5,160,np.array(arma10_res)))
print('the variance of forcast
error',estimated_variance(5,160,arma10_fore_error.values))

#diagnostic test
#chi2 test
chi_test(1,0,20,30458.815462,arma10_res)

#Confidence interval #need to work on that
for i in range(na+nb):
    print(f'Confidence interval:\n{model.params[i]}: {model.conf_int()[i]}')

ratio_residual_forcast=estimated_variance(5,160,np.array(arma10_res))/estimat
ed_variance(5,160,arma10_fore_error.values)
print('ratio residual forcast',ratio_residual_forcast)
#it is a biased model because bet the diffrence estimated parameters are not
zero
```

References:

- https://github.com/rjafari979
- **https://www.statsmodels.org/dev/generated/statsmodels.tsa.holtwinters.ExponentialSmoothing.html**
- **https://python.hotexamples.com/examples/statsmodels.graphics.tsaplots/-/plot_acf/python-plot_acf-function-examples.html**
- **https://www.kaggle.com/datasets/shivamb/underwater-surface-temperature-dataset**