Time Series Analysis and Forecasting on Minimum Daily Temperature from 1981 to 1990

Souvik Paul [191152]
Ankit Gupta [191016]
Rajat Agarwal [191104]
Sourav Mandal [191149]
Vinay Sharma [191171]

M.Sc. Third Semester November 2020
IIT Kanpur

Acknowledgement

We would like to express our special thanks of gratitude to our Prof. Dr. Amit Mitra to show us various way to complete the project, shared his valuable lecture notes and by clearing all doubts questioned by us.We would also like to thank Sir for providing us the opportunity to implement our knowledge on time series .

Contents

1 Introduction

A time series is a series of data points indexed in time order. Time Series analysis accounts for the fact that the data points taken over time may have an internal structure (such as autocorrelation, trend or seasonal variation) that should be explored and used to forecast future observation.

2 Data Description

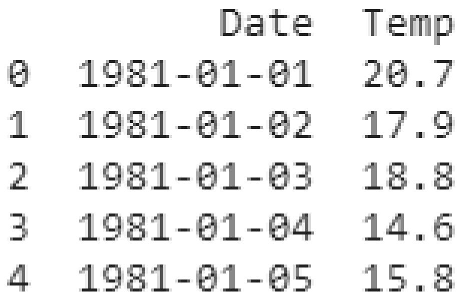
We have taken this data on Daily Minimum Temperatures over 10 years (1981-1990) in the city Melbourne, Australia. This data consists of 3560 temperature values (measured on Degree Celsius) from 1st January 1981 to 31 December 1990 with two missing temperatures on 31 December 1984 and 1988. We have divided our data into two parts and performed entire analysis on the observations of first 8 years and keep the rest 730 observations for prediction purpose.

2.1 Source of data

Data can be gained through following link :
`https://raw.githubusercontent.com/jbrownlee/Datasets/master/daily-min-temperatures.csv`

3 Data at a glance

Let's see the first five observation of our data



```
         Date  Temp
0  1981-01-01  20.7
1  1981-01-02  17.9
2  1981-01-03  18.8
3  1981-01-04  14.6
4  1981-01-05  15.8
```

Figure 1: First Five Observations

3.1 Plot of data

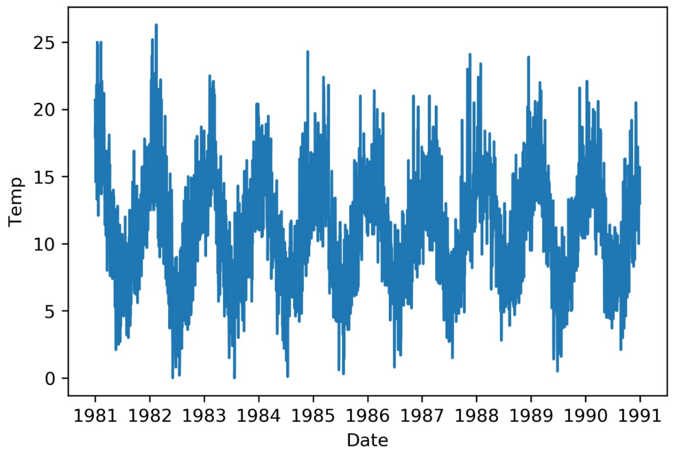
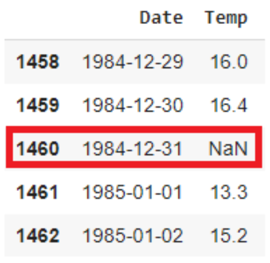


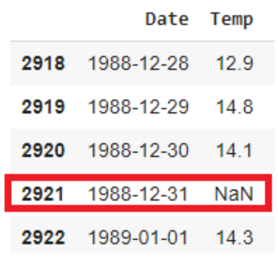
Figure 2: Plotting of Orignal data

4 Missing Value Imputation

Our data have two missing values in 31 December of 1984 and 1988.



	Date	Temp
1458	1984-12-29	16.0
1459	1984-12-30	16.4
1460	1984-12-31	NaN
1461	1985-01-01	13.3
1462	1985-01-02	15.2



	Date	Temp
2918	1988-12-28	12.9
2919	1988-12-29	14.8
2920	1988-12-30	14.1
2921	1988-12-31	NaN
2922	1989-01-01	14.3

Figure 3: Missing value in 1984 and 1988

Ploting near missing data we get

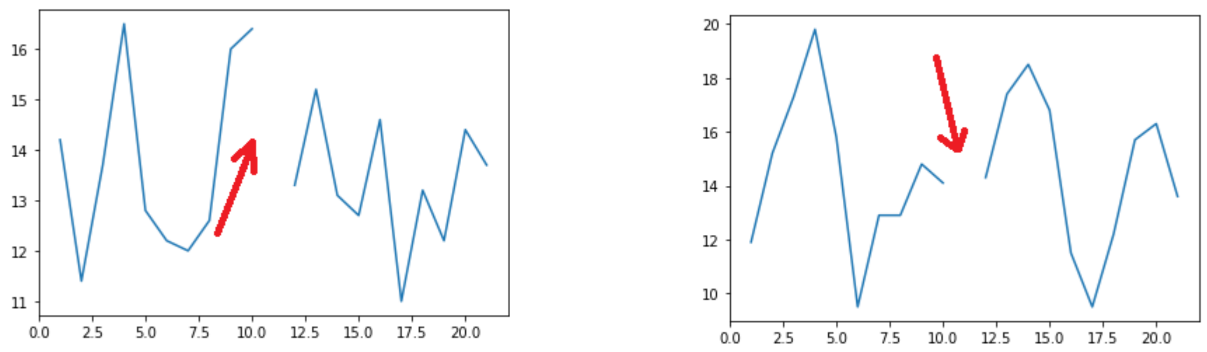


Figure 4: Missing value in 1984 and 1988

We see that around the missing observations the nature of the graph is of random type. So, we take the average value of five observations before missing value and five observations after the missing value to impute this missing values. For 1984 we find this imputed value as 16.67 and for 1988 we find this value as 14.8

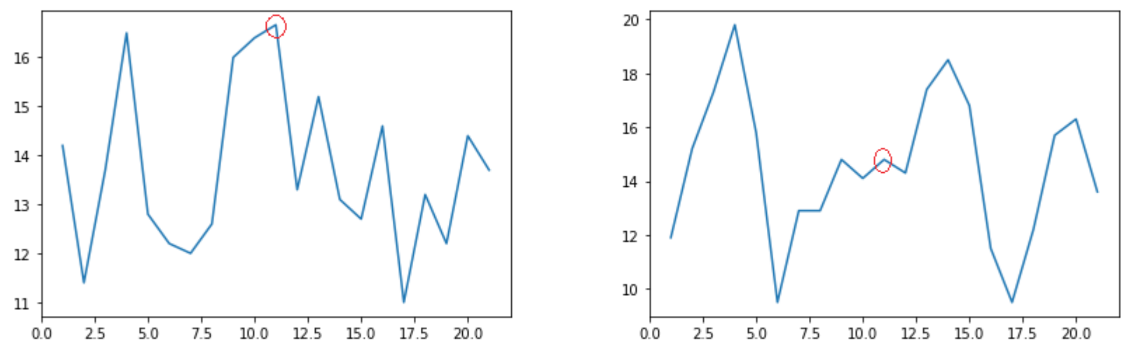


Figure 5: Imputed values in 1984 and 1988

5 Objectives of the Project

1. Firstly Check whether the data is purely random or not.
2. Check whether trend is present in the data and accordingly estimation and elimination of it.
3. Check whether seasonal component is present or not and accordingly estimation and elimination of it.
4. From de-trended and de-seasonalized data checking randomness .
5. Check stationarity and accordingly fitting model on the residual
6. Finally Prediction of temperature of last two years and observe its accuracy.

6 Randomness of the Data

Here we used the " turning point test " a non parametric test to claim whether data is purely random or not . the turning point test details are as follows:-
Test Hypothesis:-
H0 :- Series is purely random against
H1 :- Series is not purely random.
Let Y_t be the temperature at time index t . Now, Y_t is said to be a turning point if $Y_t > Y_{t-1}$ and $Y_t > Y_{t+1}$ or $Y_t < Y_{t-1}$ and $Y_t < Y_{t+1}$
Now , T $= \sum Y_t$, we know by standard calculation and under the null hypothesis :-
E(T)$=\frac{2(n-2)}{3}$, Var(T)$=\frac{(16n-29)}{90}$
The **test statistic** is given by :-

$$Z=\frac{T-E(T)}{\sqrt{var(T)}}; Z{\sim}N(0,1)$$

Decision rule is :-
Reject H0 against H1 at level of significance if observed |Z| $> Z_{\frac{\alpha}{2}}$; where $Z_{\frac{\alpha}{2}}$ is upper $\frac{\alpha}{2}$ critical point standard normal distrbution .
In our data we have n=2922 and we find obs(|Z|)= -15.83 and at 5% level $Z_{\frac{\alpha}{2}}$=1.96
So, we reject the null hypothesis, i.e. after performing turning point test we found that

our series is not purely random . Also from figure 2 it is clear that the data is periodic and this pattern of period repeats every year. So, data is not purely random.

7 Checking for the presence of Trend

Now our objective is to test whether there is trend component present in the data or not. Hence we will do it using **Relative Ordering Test**. Here -
H_0 : There is no trend in our data
H_1 : Trend is present in the data
Let, **Q** : Number of discordant pairs in the data ((y_i, y_j) is said to be discordant if for i<j $y_i > y_j$)
Now, under H_0, E(**Q**)=$\frac{n(n-1)}{4}$
If obs(Q) » E(Q) that implies decreasing trend
If obs(Q) « E(Q) that implies increasing trend
Now, Q is related to Kendall's τ, the rank correlation coefficient $\tau = 1 - \frac{4Q}{n(n-1)}$
Now, under H_0, E(τ)=0 and var(τ)=$\frac{2(2n+5)}{9n(n-1)}$
Test Statistic, Z= $\frac{\tau - E(\tau)}{\sqrt{var(\tau)}} \overset{asymp}{\sim} N(0,1)$
Test Criterion:
Reject H_0 at level α if obs(|Z|) $> \tau_{\alpha/2}$
In our data we have n=2922 and we find obs(|Z|)=0.54 and at 5% level $\tau_{\alpha/2} = \tau_{0.025}$=1.96
So, we fail to reject the null hypothesis, i.e. our data has no trend.

8 Checking for presence of Seasonality in the Data

Now we will check by Friedman test if there is any seasonality present in our data. Here -
H_0 : Data has no seasonality
H_1 : H_0 is not true
Here we have period of seasonality d=r=365 and we have c=8 years. Let M_{ij} denotes the rank corresponding to ith day for the jth year. Let M_i denotes the total rank for day i (total over the c years) i=1(1)365
i.e., $M_i = \sum_{j=1}^{c} M_{ij}$
Under H_0, E(M_i) = $\frac{c(r+1)}{2}$
Test statistic is x=365$\frac{\sum_{i=1}^{r}(M_i - \frac{c(r+1)}{2})^2}{cr(r+1)}$
Under H_0, $X \overset{asymp}{\sim} \chi^2_{r-1}$

Test Criterion:
We reject H_0 at α level of significance if obs(X)>$\chi^2_{\alpha;r-1}$
In our data we observed X=1792.42 and at 5% level $\chi^2_{\alpha;r-1} = \chi^2_{0.05;364}$=409.49
So, clearly we reject the null at level 5% i.e. data has seasonality.

8.1 Estimation of Seasonality:

We use a 4 degree polynomial to estimate the seasonality. We want to estimate Y_t by $\hat{Y_t} = at^4 + bt^3 + ct^2 + dt + e$. Using least square estimates we find $\hat{a} = -1.09 \times 10^{-8}, \hat{b} = 8.68 \times 10^{-6}, \hat{c} = -2.003 \times 10^{-3}, \hat{d} = 1.06 \times 10^{-1}, \hat{e} = 14$
i.e. These $\hat{a}, \hat{b}, \hat{c}, \hat{d}, \hat{e}$ *minimizes* $\sum_t (Y_t - \hat{y_t})^2$
After estimating the seasonality we get the following kind of graph. This red curve shows the estimates of seasonality

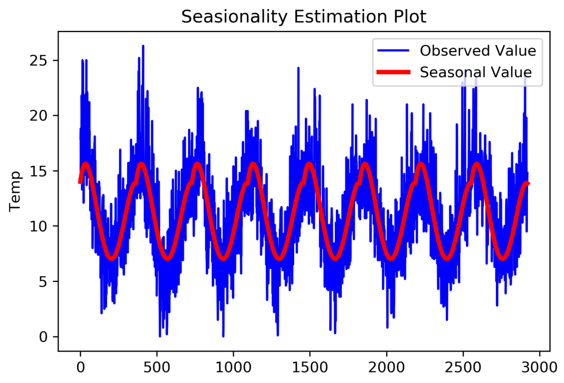


Figure 6: Estimates of seasonality

So, now we have estimates of seasonality. By substraction we get the de-seasonalized data (or residual).

9 Checking Randomness in de-seasonalized data

We use Turning point test to check randomness in the de-seasonalized data. If it is random our analysis of estimation will end here, otherwise we will check stationarity and accordingly fit appropriate model.
Plot of de-seasonalied data -

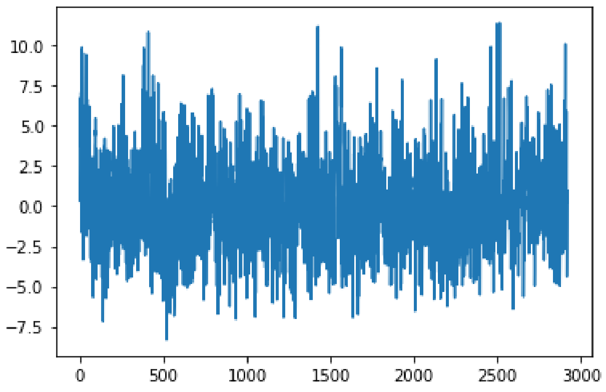


Figure 7: Plot of de-seasonalized data

Turning point test details are as follows:-
Test Hypothesis:-
H0 :- Series is purely random against
H1 :- Series is not purely random.
Let Y_t be the deseasonalized temperature data at time index t . Now, Y_t is said to be a turning point if $Y_t > Y_{t-1}$ and $Y_t > Y_{t+1}$ or $Y_t < Y_{t-1}$ and $Y_t < Y_{t+1}$
Let, U_t=1 if Y_t is a turning point , 0 otherwise
Now , T = $\sum_{t=2}^{n-1} U_t$, we know by standard calculation and under the null hypothesis :-
E(T)=$\frac{2(n-2)}{3}$, Var(T)=$\frac{(16n-29)}{90}$
The **test statistic** is given by :-

Z=$\frac{T-E(T)}{\sqrt{var(T)}}$; $Z\sim$N(0,1)
Decision rule is :-
Reject H0 against H1 at level of significance if observed |Z| $> Z_{\frac{\alpha}{2}}$; where $Z_{\frac{\alpha}{2}}$ is upper $\frac{\alpha}{2}$ critical point standard normal distrbution .
In our data we have n=2922 and we find obs(|Z|)= -13.94 and at 5% level $Z_{\frac{\alpha}{2}}$=1.96 thus we rejct the null hypothesis H0 and conclude that the deseasonalized data is not purley random . thus we can move forward to check whether the data is stationary or not .

10 Checking for Stationarity in the random part

AUGMENTED DICKEY–FULLER TEST (ADF)
An Augmented Dickey–Fuller test (ADF) is a test for a unit root in a time series sample. It is an augmented version of the Dickey–Fuller test for a larger and more complicated set of time series models. The augmented Dickey–Fuller (ADF) statistic, used in the test, is a negative number. The more negative it is, the stronger the rejection of the hypothesis that there is a unit root at some level of confidence.
Let $\{X_t\}$ follows the AR(p) model with mean μ given by
$X_t - \ \mu = \phi_1(X_{t-1} - \ \mu) + ... + \phi_p(X_{t-p} - \ \mu + Z_t, \ \{Z_t\} \sim WN(0, \sigma^2)$
Above equation can be written as
$\nabla X_t = \phi_0^* + \phi_1^* X_{t-1} + \sum_{j=2}^{p} \phi_j^* \nabla X_{t-j+1} + Z_t$
where $\phi_0^* = \mu(1 - \sum_{i=1}^{p} \phi_i), \ \phi_1^* = \sum_{i=1}^{p} \phi_i - 1, \ \phi_j^* = -\sum_{i=j}^{p} \phi_i$
Now if $X_t \ follows \ AR(p) \ then \ \nabla X_t \ follows \ AR(p-1)$
Hypothesis of Interest:
H_0 : Data is non stationary or $\phi_1^* = 0$
H_1 : Data is Stationary or $\phi_1^* < 0$
The estimated standard error of ϕ_1^* is $\hat{SE(\phi_1^*)} = S \ (\sum_{i=2}^{p}(X_{i-1} - \overline{X}))^{-0.5}$
where $S^2 = \sum_{t=2}^{n}(\nabla X_t = \phi_0^* + \phi_1^* X_{t-1} + \sum_{j=2}^{p} \phi_j^* \nabla X_{t-j+1})^2/(n-p-2)$
The test statistic for calculating the ADF test is $\hat{\tau_\mu} = \phi_1^*/\hat{SE(\phi_1^*)}$
Test Criterion:
we reject H_0 if $\hat{\tau_\mu} < DF_\alpha$ From the test we have found out that $\hat{\tau_{mu}} = -16.04$ and $DF_{0.05} = -2.87$, so, we reject the null at 5% level, i.e. the residual data we have derived is stationary one.

11 Model Identification

11.1 Moving Average/Auto-Regressive/Auto-regressive moving average:

In order to find whether it is AR/MA/ARMA model, we'll look at Auto-Correlation Plot and Partial Auto-Correlation Plot and we assume it to be ARMA(p,q).

ACF and PACF plot

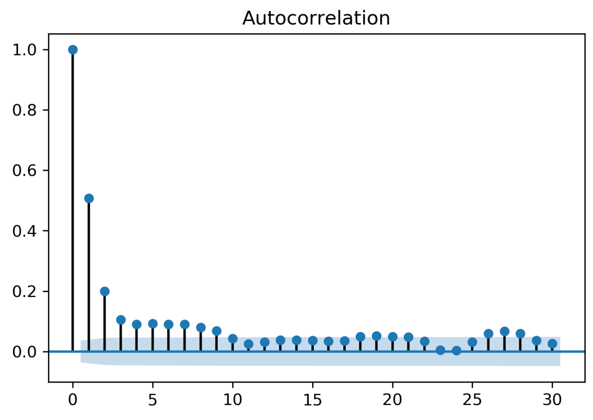


Figure 8: ACF Plot

From the plot, we realize that MA of any order cannot fit the data as the spikes do not tail after certain lag, so AR or ARMA model will be appropriate for the data, which can be confirmed from PACF plot. Hence we look at PACF plot.

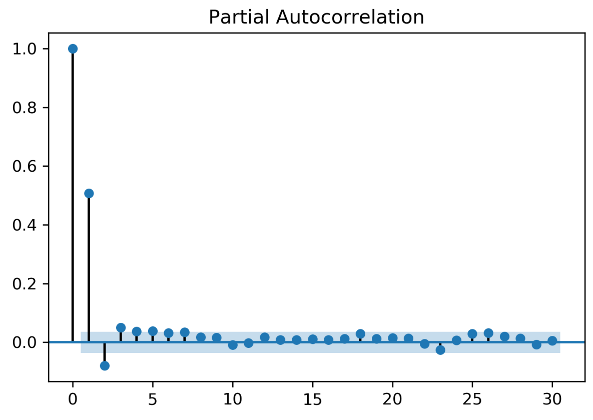


Figure 9: PACF Plot

From the PACF plot, we can see that the spike never cuts off after any lag, i.e. lies within the band. So from the ACF and PACF plot we suspect that ARMA(p,q) will fit the model well.

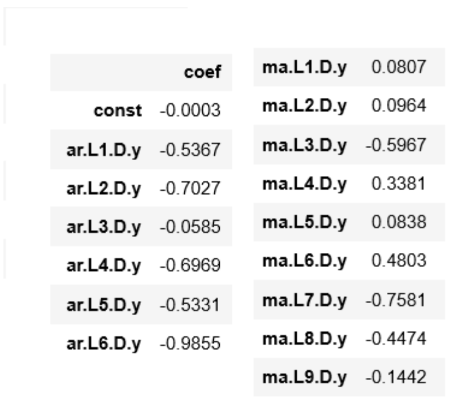
Estimates of the order of ARMA model:
We see that the data is stationary and not random.So We want to fit a ARMA model of order (p,q) to our data.

ARMA(p,q): $\{X_t\}$, a stationary time series data, follows ARMA(p,q) if
$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + ... + \phi_p X_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_q \epsilon_{t-q}$, $\{\epsilon_t\} \sim WN(0, \sigma^2)$
Now, to find out the appropriate order of ARMA(p,q), we will calculate the value of Akaike Information Criterion and consider the corresponding value of p and q for which AIC is minimum.
For ARMA(p,q), AIC$=-2\ln L(\theta) + 2(p+q+1)$

We perform adfuller test and find p=6 where Akaike Information Criteria(AIC) is minimum. Again, from pacf graph, we find the value of p is around 6, so we consider p=6 in ARMA(p,q) model and from acf graph we find the value of q=9.
So Our desired ARMA model is ARMA(6,9)

$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \phi_3 X_{t-3} + \phi_4 X_{t-4} + \phi_5 X_{t-5} + \phi_6 X_{t-6} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_9 \epsilon_{t-9}$, $\{\epsilon_t\} \sim WN(0, \sigma^2)$

by estimating the value of ϕ_i and θ_j for i =1 to 6 and j=1 to 9 we get our final model :



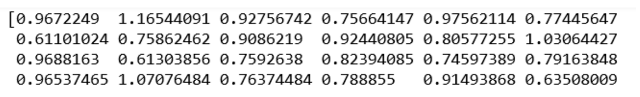
	coef
const	-0.0003
ar.L1.D.y	-0.5367
ar.L2.D.y	-0.7027
ar.L3.D.y	-0.0585
ar.L4.D.y	-0.6969
ar.L5.D.y	-0.5331
ar.L6.D.y	-0.9855

ma.L1.D.y	0.0807
ma.L2.D.y	0.0964
ma.L3.D.y	-0.5967
ma.L4.D.y	0.3381
ma.L5.D.y	0.0838
ma.L6.D.y	0.4803
ma.L7.D.y	-0.7581
ma.L8.D.y	-0.4474
ma.L9.D.y	-0.1442

Figure 10: Estimate of coefficient of ARMA(6,9)

12 Forecasting

As we have splitted the data into train and test before and build the model on train data, which is ARMA(6,9). Length of test data is 730. Now we predict the minimum daily temperature for next two year (for test data) using this model.
Some of the predicted value using ARMA(6,9) is given as



```
[0.9672249  1.16544091 0.92756742 0.75664147 0.97562114 0.77445647
 0.61101024 0.75862462 0.9086219  0.92440805 0.80577255 1.03064427
 0.9688163  0.61303856 0.7592638  0.82394085 0.74597389 0.79163848
 0.96537465 1.07076484 0.76374484 0.788855   0.91493868 0.63508009
```

Figure 11: Predicted value using ARMA(6,9)

As we have fitted model on de-seasonaliesd data, and get predicted value according to that model.Now to get actual predicted value of minimum daily temperature we add seasonality to predicted value got by ARMA(6,9).

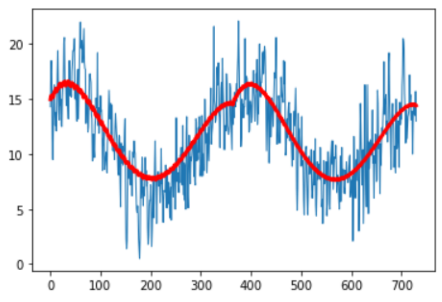


Figure 12: Actual temp vs predicted temp

In this graph, red curve represent predicted value,and blue represent the actual value.

12.1 Root Mean Square Error(RMSE)

RMSE of data given by a model is formulated as
RMSE=$(\frac{1}{n}\sum(y_i - \hat{y})^2)^{\frac{1}{2}}$
where, n=total no observation in test data
y_i= ith actual value of test data.
$\hat{y}$= ith predicted value of test data.
In our case RMSE is given by 2.55 approximately which imply that on average there is around 2.55 degree fluctuation between actual temperature and predicted temperature.

13 Conclusions

...There are two missing observation in data, first we estimate these observations.
...Trend is not present in data but seasonality is present in data
... Estimate seasonality using appropriate order polynomial and de-seasonalize the data.
...After de-seasonalization data found to be Stationary.
...Fit appropriate order of ARMA model on trend data using PACF and ACF graph.
...ARMA(6,9) is best model for given data.
...Forecast minimum temperature using model ARMA(6,9) for test data.
...RMSE of model on test data is around 2.55

14 Reference

1. Time Series Class Notes by Dr. Amit Mitra under MTH 517A.
2. Forecasting: Principles and Practice by Rob J Hyndman and George Athanasopoulos.
3. Introduction to Time Series and Forecasting by Peter J. Brockwell Richard A. Davis.

15 Python Code

```
#!/usr/bin/env python
# coding: utf-8
# In[1]:
import random
random.seed(100)
# In[2]:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import urllib
from pandas import read_csv
from matplotlib import pyplot
from numpy import polyfit
# In[3]:

import math
from scipy.stats import norm, chi2


# In[4]:

link='https://raw.githubusercontent.com/jbrownlee/Datasets/master
/daily-min-temperatures.csv'
data=pd.read_csv(link)


# In[5]:

data.shape


# In[6]:

type(data)
# Plot the original Data
dat = data
dat.Date = pd.to_datetime(dat.Date)
dat = dat.set_index('Date')
dat=dat.drop(columns='Time')
plt.plot(dat)
plt.xlabel('Date')
plt.ylabel('Temp')
plt.savefig('original_data', dpi=300, bbox_inches='tight')



# In[7]:
```

```
print(data.head(5))
print(data.tail(5))
print(data.iloc[1459:1461]) #location of  the missing value
print(data.iloc[2919:2921]) #location of  the missing value


# In[8]:

Time=['%.3f'%(1981+i/365) for i in range(365)]
for i in range(1982,1991):
  if(i%4==0):
    Time+=['%.3f'%(i+j/366) for j in range(366)]
  else:
    Time+=['%.3f'%(i+j/365) for j in range(365)]


# In[9]:

mis1=pd.DataFrame({'Date':'1984-12-31','Temp':float('NaN')},index=[1460])
mis2=pd.DataFrame({'Date':'1988-12-31','Temp':float('NaN')},index=[2921])
df=pd.concat([data.iloc[:1460],mis1,data.iloc[1460:2920],mis2,data.iloc[292
#d=pd.DataFrame(df,index)
df.shape


# In[10]:

df.reset_index(inplace=True)


# In[11]:

df['Time']=Time


# In[12]:

data=df.iloc[:,1:4]


# In[13]:

data.iloc[1458:1463]


# In[14]:

data.iloc[2918:2923]
```

```
# In[15]:

train=data.iloc[:3287,:]
test=data.iloc[3287:,:]
print(train.shape)
print(test.shape)


# In[16]:

y1=list(data.iloc[1450:1460,1])+list(data.iloc[1460:1471,1])
y2=list(data.iloc[2911:2921,1])+list(data.iloc[2921:2932,1])


# In[17]:

plt.plot(list(range(1,22)),y1);


# In[18]:

plt.plot(list(range(1,22)),y2);


# In[19]:

a=np.array([364,729,1094,1460,1825,2190,2555,2921,3286,3651])
y=data.iloc[a,1].values
miss1 = np.mean( np.concatenate((y[0:3],y[4:7])) )
miss2 = np.mean( np.concatenate((y[4:7],y[8:11])) )
print(miss1,miss2)
data.iloc[1460,1] = miss1
data.iloc[2921,1] = miss2
y1=list(data.iloc[1450:1460,1])+list(data.iloc[1460:1471,1])
y2=list(data.iloc[2911:2921,1])+list(data.iloc[2921:2932,1])
plt.plot(list(range(1,22)),y1);


# In[20]:

print(data.head(5))


# In[21]:

len(data)


# In[22]:
```

```
data.iloc[:,2:3]=[1+i%3652 for i in range(0, len(data))]


# In[23]:

data.iloc[:,2:3]


# In[24]:

data.shape


# In[25]:

train=data.iloc[:2922,:]
test=data.iloc[2922:,:]
print(train.shape,test.shape)


# In[26]:

temp=train.iloc[:,1:2]
n=temp.shape[0]
print(temp)


# In[27]:

temp=[temp.iloc[i] for i in range(n)]


# In[28]:

print(temp[1:5])
print(len(temp))


# In[29]:

def relative_ordering(temp,alpha):
  Q=0
  n=len(temp)
  for i in range(n):
    for j in range(i+1,n):
      if(temp[i]-temp[j]>0):
        Q+=1
  print('Q=',Q)
  T=1-4*Q/(n*(n-1))
```

```
    VT=2*(2*n+5)/(9*n*(n-1))
    Z=T/math.sqrt(VT)
    t=norm.ppf(1-alpha/2)
    print('Z=',Z)
    print('t_alpha/2=',t)
    if(abs(Z)<=t):
        print('So, Trend is not present')
    else:
        print('Trend is present')



# In[30]:

temp=[float(i) for i in temp]


# In[31]:

relative_ordering(temp,0.05)


# In[32]:

def friedman(temp,r_val,c,alpha):
    r=[]
    for i in range(1,9):
        if(i<5):
            year=temp[365*(i-1):i*365]
        else:
            year=temp[365*(i-1)+1:i*365+1]
        year=np.array(year)
        ran=pd.Series(year)
        ran=ran.rank()
        r.append(ran)
    s=np.array(sum(r))
    m=c*(r_val+1)/2
    v=c*r_val*(r_val+1)
    X=12*sum((s-m)**2)/v
    print('X=',X)
    chi_square=chi2.ppf(1-alpha,r_val-1)
    print('Chi square value ',chi_square)
    if(X>chi_square):
        print('Seasonality is present')
    if(X<=chi_square):
        print('No Seasonality')


# In[33]:
```

```
friedman(temp,365,8,0.05)


# In[34]:

type(temp)


# In[35]:

def turning_point(temp,alpha):
  P=0
  n=len(temp)
  for i in range(1,n-1):
    if((temp[i]>temp[i-1] and temp[i]>temp[i+1]) or (temp[i]<temp[i-1] and
      P+=1
  print('P=',P)
  EP=2*(n-2)/3
  VP=(16*n-29)/90
  Z=(P-EP)/math.sqrt(VP)
  print("Z=",Z)
  t=norm.ppf(1-alpha/2)
  print('t_alpha/2=',t)
  if(abs(Z)>t):
    print('Series is not purely Random')
  else:
    print('Series is purely Random')



# In[36]:

turning_point(temp,0.05)


# In[39]:

x=[]
# fit polynomial: x^2*b1 + x*b2 + ... + bn
for i in range(1981,1989):
  if(i%4==0):
    for j in range(0,366):
        x.append(j)
  else:
    for j in range(0,365):
        x.append(j)

print(len(x))
y = temp
degree = 4
```

```
coef = polyfit(x, y, degree)
print('Coefficients: %s' % coef)
# create curve
curve = list()
for i in range(len(x)):
        value = coef[-1]
        for d in range(degree):
                value += x[i]**(degree-d) * coef[d]
        curve.append(value)


# In[40]:

# plot curve over original data
plt.plot(data.Temp[1:2922], color='blue', label='Observed Value')
plt.plot(curve, color='red', linewidth=3, label='Seasonal Value')
plt.ylabel('Temp')
plt.title('Seasionality Estimation Plot')
plt.legend(loc='upper right')
plt.savefig('Seasionality Estimation', dpi=300, bbox_inches='tight')
plt.show()


# In[41]:

# create seasonally adjusted
values = y
diff = list()
for i in range(len(values)):
        value = values[i] - curve[i]
        diff.append(value)
pyplot.plot(diff)
pyplot.show()


# In[42]:

print(diff[0:5]) # temperature after removing seasonality (first six observ


# In[43]:

print(curve[0:5])   #  estimates of seasonality (first six observation)


# In[44]:

friedman(diff,365,8,0.05)
```

```
# In[45]:

turning_point(diff,0.05)


# In[46]:

train.shape
print(train.head())


# In[47]:

train.iloc[:,2]=x


# In[48]:

train.iloc[:,1:2]= diff


# In[49]:

print(train.head())


# In[50]:

#  Checking for stationarity


# In[51]:

pyplot.plot(diff)
plt.xlabel('No. of sequential observations')
plt.ylabel('Temp')
plt.title('After Removing the Seasonal Component')
plt.savefig('Removing Seasionality', dpi=300, bbox_inches='tight')
plt.show()
# in plot seems to be stationary as no trend and seasonal component seems h


# In[52]:


import random
from random import sample
#  drawing random sample from diff and checking mean and var
# In[53]:


s1=diff[0:1495]
s2=diff[1495:(n-1)]


# In[54]:


print(s1[0:5])


# In[56]:


# checking equality of mean and var

mean1, mean2 = (sum(s1))/(len(s1)),(sum(s2))/(len(s2))
def vari(list):
    mean=sum(list)/len(list)
    summ=0
    for i in list:
        summ=summ+(i-mean)**2
    variance=summ/len(list)
    return(variance)

print('mean1=%f, mean2=%f' % (mean1,mean2))
var1,var2=vari(s1),vari(s2)
print('variance1=%f, variance2=%f' % ((var1), var2))

# so mean and var are very close so data seems stationarity


# In[57]:

#Augmented Dickey-Fuller test  for testing stationarity
from statsmodels.tsa.stattools import adfuller
X = diff
result = adfuller(X,autolag='AIC')
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Lag Used: %f' % result[2])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
if result[1]<0.05:
    print("data series is stationary")
else:
    print("data serpip install pmdarimaies is not stationary")
```

```
# In[58]:

from statsmodels.graphics.tsaplots import plot_acf,plot_pacf
plot_acf(diff,lags=30)
plot_pacf(diff,lags=30)


# From ACF plot lag q=9, from  PACF plot and ADF test lag p =6

# In[59]:

pip install pmdarima


# In[60]:

from pmdarima.arima import auto_arima


# In[61]:

#stepwise_fit = auto_arima(diff , trace =True ,suppress_warninigs =True)
#stepwise_fit.summary()


# In[62]:

from statsmodels.tsa.arima_model import ARIMA
train.shape
train.head(5)
#test.shape
model = ARIMA(diff,order=(6,1,9))
model = model.fit()
model.summary()


# In[63]:

start =len(train)
end = len(train)+len(test)-1
pred = model.predict(start=start ,end = end ,typ='levels')
print(pred)


# In[64]:

s_comp=curve[0:730]
```

```
# In[65]:

len(s_comp)


# In[66]:

pred1=pred+s_comp


# In[67]:

test = list()
for i in range(730):
        test1 = data.Temp[2922+i]
        test.append(test1)



pyplot.plot(test,linewidth=1)
pyplot.plot(pred1, color='red', linewidth=3)
pyplot.show()


# In[68]:

len(test)


# In[70]:

sum1=0

for i in range(len(test)):
    sum1=sum1+(test[i]-pred1[i])**2


print("sum=",sum1)
RMSE=math.sqrt(sum1/len(test))
print("RMSE=",RMSE)
```