##5차 과제: 202340339 이하늘

공통 사용자 지정 함수

#코드

```
import numpy as np

import matplotlib.pyplot as plt


def ARMA_11(phi0, phi1, th1, ss, n):

    L = []

    w = np.random.normal(0,ss**0.5,n+1) #표준편차 #w[0],...,w[n]

    x0 = 0 #초기값

    for t in range(1, n+1): #t=1,2,...,n

        xt = phi0 + phi1*x0 + w[t] + th1*w[t-1]

        L.append(xt)

        x0 = xt

    plt.plot(L)

    plt.show()

    return L


def ARMA_22(phi0, phi1, phi2, th1, th2, ss, n):

    L = []

    w = np.random.normal(0,ss**0.5,n+2) #표준편차 #w[0],...,w[n+1] 총 n+2개

    x00,x0 = 0,0 #초기값

    for t in range(2, n+2): #t=2,3,...,n+1 총 n개

        xt = phi0 + phi1*x0 + phi2*x00 + w[t] + th1*w[t-1] + th2*w[t-2]

        L.append(xt)

        x00 = x0

        x0 = xt

    plt.plot(L)

    plt.show()
```

```
    return L

def ACF(D):
    x = D
    n = len(D)
    mu = np.mean(D)
    L = []
    for h in range(21): #h=0,1,...,20 - lag 20까지
        Lh = []
        for t in range(0, n-h):
            ac = (x[t+h] - mu)*(x[t] - mu)
            Lh.append(ac)
        autocov_h = sum(Lh)/n
        L.append(autocov_h)
    AutoCov = np.array(L)
    Sacf = AutoCov/AutoCov[0]
    return Sacf

def OLSE_AR_1(D): #D=data
    n = len(D)
    X = np.array(D) - np.mean(D) #평균이 0인 주어진 데이터
    L_num, L_den = [],[]
    for t in range(1,n):
        L_num.append(X[t]*X[t-1])
        L_den.append(X[t-1]**2)
    olse = sum(L_num)/sum(L_den) #for phi = phi_1
    hat_phi0 = np.mean(D)*(1-olse) # mu = phi0/(1-phi_1)
    hat_phi1 = olse
    return hat_phi0, hat_phi1
```

```
def YW_AR_1(D):

    n = len(D)

    X = np.array(D) - np.mean(D)

    sacf = ACF(D) #sm..........acf? #사용자 지정 함수

    hat_phi1 = sacf[1]

    hat_phi0 = np.mean(D)*(1-hat_phi1)

    return hat_phi0, hat_phi1


def OLSE_AR_2(D): #xt = b'zt + wt where zt = (x_{t-1}, x_{t-2}), b = (b1,b2)

    n = len(D)

    X = np.array(D) - np.mean(D)

    Z, xx = [],[]

    for t in range(2,n):

        zt = [X[t-1],X[t-2]]

        Z.append(zt)

        xx.append(X[t])

    Z = np.array(Z)

    xx = np.array(xx) #hat b = (Z'Z)^{-1}(Z'xx)

    First = np.dot(Z.T, Z) #T = transpose

    Second = np.dot(Z.T, xx)

    F_inv = np.linalg.inv(First) #역행렬

    olse = np.dot(F_inv, Second)

    phi1, phi2 = olse

    phi0 = np.mean(D) * (1-phi1-phi2)

    return phi0,phi1,phi2


def YW_AR_2(D): #강의노트 오류있음

    n = len(D)

    X = np.array(D) - np.mean(D)

    rho = ACF(D) #AX = B --> X = A^{-1}B
```

```
    B = np.zeros((2,1))
    A = np.zeros((2.2))
    for i in range(2): #i=0,1
        B[i][0] = rho[i+1]
        for j in range(2): #j=0,1
            if i >= j:
                A[i][j]b = rho[i-j]
            else:
                A[i][j] = rho[j-i]
    A_inv = np.linalg.inv(A)
    phi = np.dot(A_inv,B)
    phi1,phi2 = phi
    phi0 = np.mean(D)*(1-phi1-phi2)
    return phi0,phi1,phi2

def FORE_AR1(D):
    mu = np.mean(D)
    print(mu)
    olse = OLSE_AR_1(D)
    print(olse) #phi0, phi1
    Xt = D[-1] #the last
    Xt1 = mu + olse[1]*(Xt-mu) #one-step forecast
    Xt2 = mu + olse[1]*(Xt1-mu) #two-step forecast
    Xt3 = mu + olse[1]*(Xt2-mu) #three-step forecase
    return Xt1, Xt2, Xt3

def Graph_FORE_AR1(D, ell):
    mu = np.mean(D)
    X = np.array(D) - mu
    olse = OLSE_AR_1(D) #phi0, phi1
```

```
    L = []
    Xt = D[-1]
    for k in range(1,ell+1): #k=1,2,,,,,ell
        fore_k = mu + olse[1]*(Xt-mu)
        L.append(fore_k)
        Xt = fore_k
    plt.plot(L)
    plt.show()
    return L

def ERROR_one_step_forecast_AR1(D,m): #RMSE, MAE 강의노트 오류
    T = len(D)
    n = T - m
    Lr,Lf,Le = [],[],[]
    for i in range(m): #i = 0,1,...,m-1 총 m개
        INS = D[i:i+n] #of size n, from i, i+1,...,i+n-1
        Real_one = D[i+n] #one-step real value
        mu = np.mean(INS)
        olse = OLSE_AR_1(INS) #phi0, phi1
        Xt = INS[-1]
        Fore_one = mu + olse[1]*(Xt-mu)
        Lr.append(Real_one)
        Lf.append(Fore_one)
        Le.append(Real_one - Fore_one)
    Le = np.array(Le)
    MAE = np.mean(np.abs(Le))
    RMSE = (np.mean(Le**2))**0.5
    print(MAE, RMSE,'=MAE,RMSE')
    plt.plot(Lr, 'b', label = "Real value", marker = 'o')
    plt.plot(Lf, 'r', label = "Forecast", marker = '*')
```

```
    plt.legend()
    plt.show()

def ERROR_one_step_forecast_AR2(D,m): #RMSE, MAE 강의노트 오류
    T = len(D)
    n = T - m
    Lr,Lf,Le = [],[],[]
    for i in range(m): #i = 0,1,...,m-1 총 m개
        INS = D[i:i+n] #of size n, from i, i+1,...,i+n-1
        Real_one = D[i+n] #one-step real value
        mu = np.mean(INS)
        olse = OLSE_AR_2(INS) #phi0, phi1
        Xt = INS[-1]
        Xt1 = INS[-2]
        Fore_one = olse[0]+olse[1]*Xt+olse[2]*Xt1
        Lr.append(Real_one)
        Lf.append(Fore_one)
        Le.append(Real_one - Fore_one)
    Le = np.array(Le)
    MAE = np.mean(np.abs(Le))
    RMSE = (np.mean(Le**2))**0.5
    print(MAE, RMSE,'=MAE,RMSE')
    plt.plot(Lr, 'b', label = "Real value", marker = 'o')
    plt.plot(Lf, 'r', label = "Forecast", marker = '*')
    plt.legend()
    plt.show()

def ERROR_one_step_forecast_AR1_YW(D,m): #RMSE, MAE 계산
    T = len(D)
    n = T - m
```

```
    Lr,Lf,Le = [],[],[]
    for i in range(m): #i = 0,1,...,m-1 총 m개
        INS = D[i:i+n] #of size n, from i, i+1,...,i+n-1
        Real_one = D[i+n] #one-step real value
        yw = YW_AR_1(INS) #phi0, phi1
        Xt = INS[-1]
        Fore_one = yw[0] + yw[1]*Xt
        Lr.append(Real_one)
        Lf.append(Fore_one)
        Le.append(Real_one - Fore_one)
    Le = np.array(Le)
    MAE = np.mean(np.abs(Le))
    RMSE = (np.mean(Le**2))**0.5
    print(MAE, RMSE,'=MAE,RMSE')
    plt.plot(Lr, 'b', label = "Real value", marker = 'o')
    plt.plot(Lf, 'r', label = "Forecast", marker = '*')
    plt.legend()
    plt.show()

def ERROR_one_step_forecast_AR2_YW(D,m): #RMSE, MAE 계산
    T = len(D)
    n = T - m
    Lr,Lf,Le = [],[],[]
    for i in range(m): #i = 0,1,...,m-1 총 m개
        INS = D[i:i+n] #of size n, from i, i+1,...,i+n-1
        Real_one = D[i+n] #one-step real value
        mu = np.mean(INS)
        yw = YW_AR_2(INS) #phi0, phi1, phi2
        Xt = INS[-1]
        Xt1 = INS[-2] if len(INS) > 1 else 0
```

```
        Fore_one = yw[0] + yw[1]*Xt + yw[2]*Xt1
        Lr.append(Real_one)
        Lf.append(Fore_one)
        Le.append(Real_one - Fore_one)
    Le = np.array(Le)
    MAE = np.mean(np.abs(Le))
    RMSE = (np.mean(Le**2))**0.5
    print(MAE, RMSE,'=MAE,RMSE')
    plt.plot(Lr, 'b', label = "Real value", marker = 'o')
    plt.plot(Lf, 'r', label = "Forecast", marker = '*')
    plt.legend()
    plt.show()
```

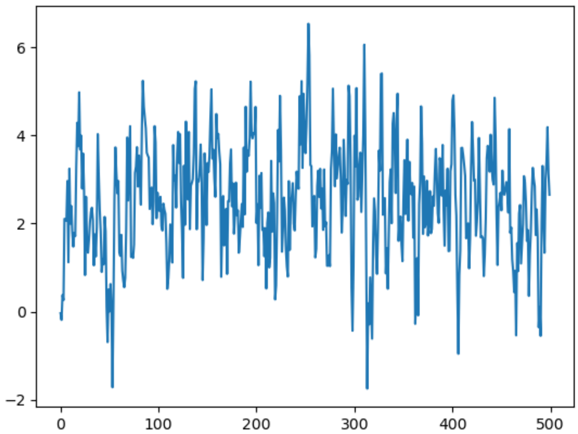
###HW: [1]

(a) 위의 Ar1 데이터에서 Yule-Walker estimator of AR(1)를 사용하여 one-step 예측 그래프를 그리고, MAE, RMSE를 계산하시오.

#코드

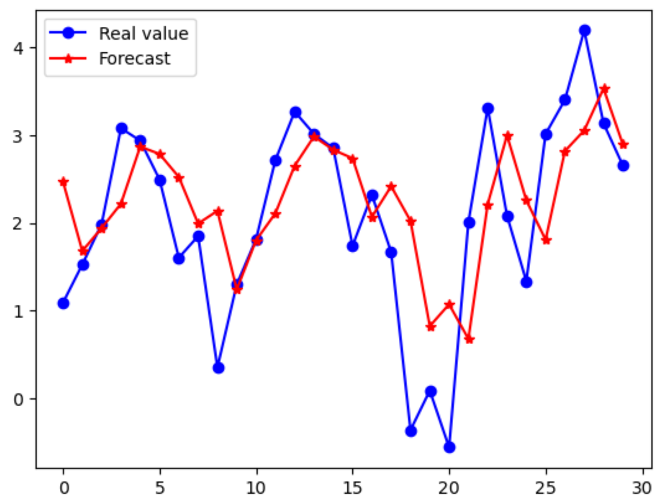
#모델 설정

Ar1 = ARMA_11(1,0.6,0,1,500)



ERROR_one_step_forecast_AR1_YW(Ar1, 30)

0.7179998638845632 0.9305242895600543 =MAE,RMSE

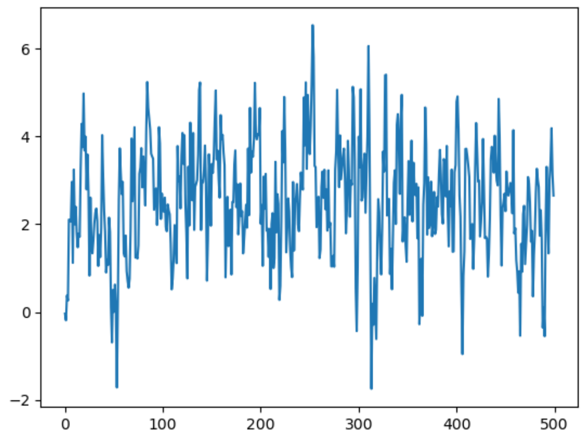


(b) 위의 Ar1 데이터에서 Yule-Walker estimator of AR(2) 를 사용하여 one-step 예측 그래프를 그리고, MAE, RMSE를 계산하시오.

#코드

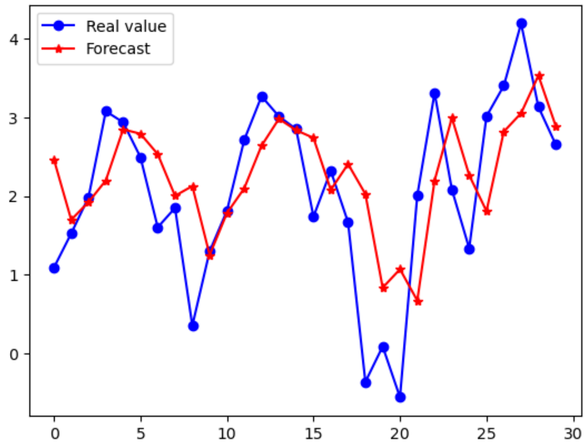
#모델 설정

Ar1 = ARMA_11(1,0.6,0,1,500)



ERROR_one_step_forecast_AR2_YW(Ar1, 30)

0.722382352547672 0.9325488319136132 =MAE,RMSE



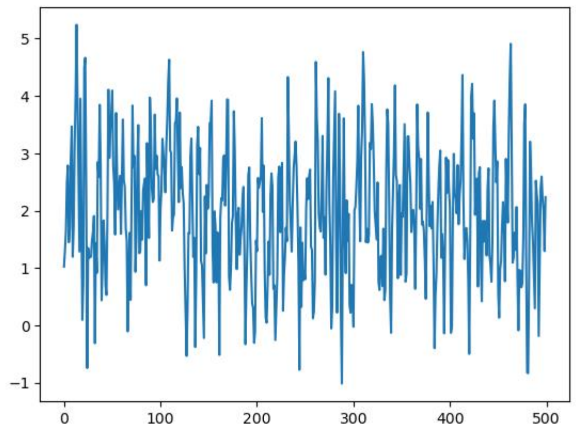
###HW: [2]

(a) 위의 Ar2 데이터에서 OLSE of AR(2) 를 사용하여 one-step 예측 그래프를 그리고, MAE, RMSE를 계산하시오.

#코드

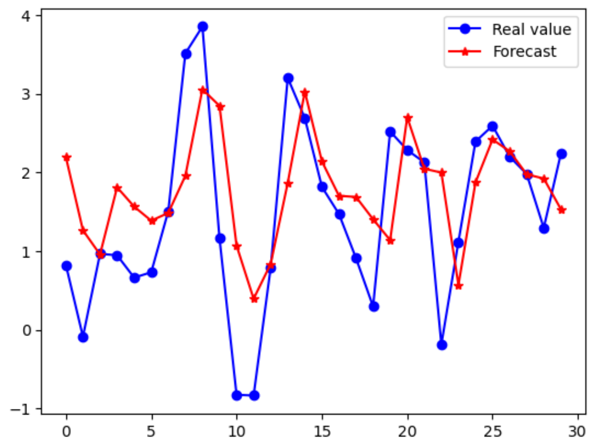
#모델 설정

Ar2 = ARMA_22(1,0.7,-0.2,0,0,1,500)



ERROR_one_step_forecast_AR2(Ar2, 30)

0.773325704990766 0.9817058433692196 =MAE,RMSE

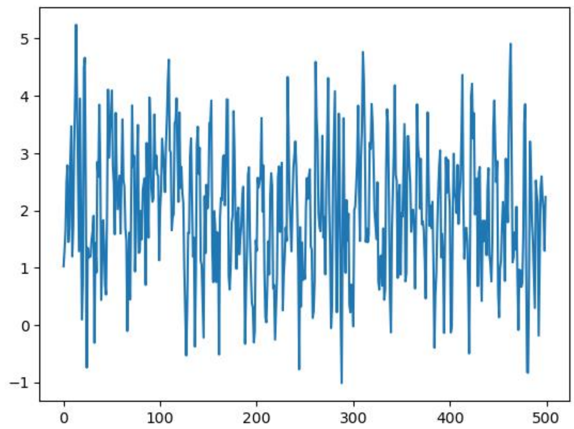


(b) 위의 Ar2 데이터에서 Yule-Walker estimator of AR(2) 를 사용하여 one-step 예측 그래프를 그리 고, MAE, RMSE를 계산하시오.

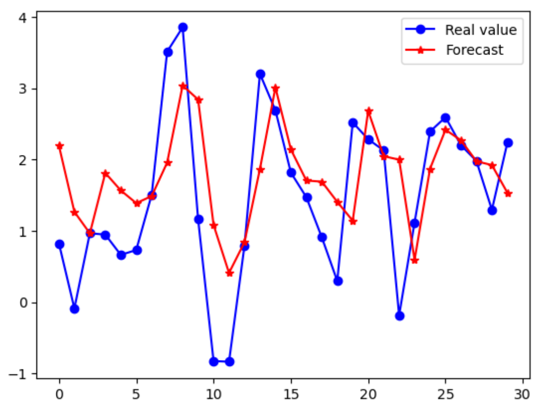
#코드

#모델 설정

Ar2 = ARMA_22(1,0.7,-0.2,0,0,1,500)



ERROR_one_step_forecast_AR2_YW(Ar2, 30)

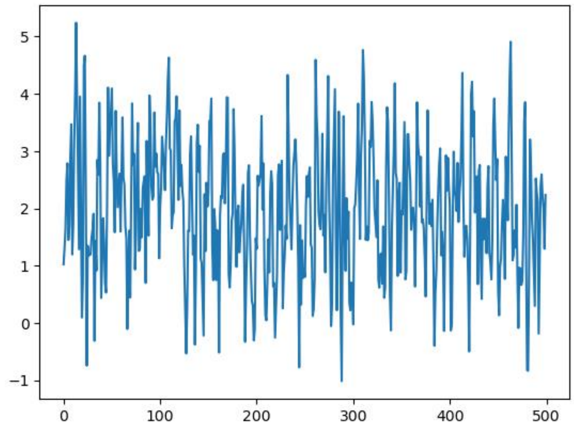


(c) 위의 Ar2 데이터에서 Yule-Walker estimator of AR(1) 를 사용하여 one-step 예측 그래프를 그리고, MAE, RMSE를 계산하시오.

#코드

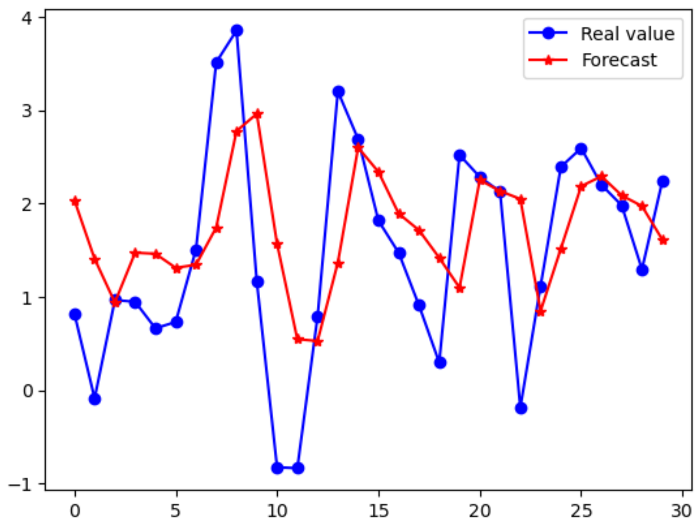
#모델 설정

Ar2 = ARMA_22(1,0.7,-0.2,0,0,1,500)



ERROR_one_step_forecast_AR1_YW(Ar2, 30)

0.8339496470290547 1.0772698172013508 =MAE,RMSE



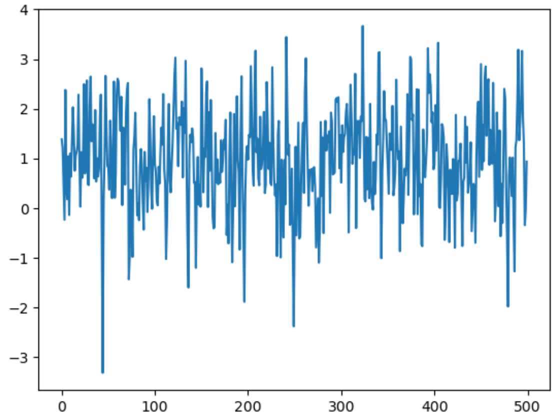
###HW: [3]

(a) 위의 Ma1 데이터에서 MA estimator (< 1 in absolute)를 사용하여 one-step 예측 그래프를 그리고, MAE, RMSE를 계산하시오.

#코드

```
def MA_1_estimator(D):

    rho = ACF(D)

    if np.abs(rho[1]) < 1:

        hat_th_11 = (1+np.sqrt(1-4*rho[1]**2))/2/rho[1]

        hat_th_12 = (1-np.sqrt(1-4*rho[1]**2))/2/rho[1]

    else:

        print('DNE')

    return hat_th_11, hat_th_12


def ERROR_one_step_forecast_MA1(D,m): #RMSE, MAE

    T = len(D)

    n = T - m

    Lr,Lf,Le = [],[],[]

    for i in range(m): #i = 0,1,...,m-1 총 m개

        INS = D[i:i+n] #of size n, from i, i+1,...,i+n-1

        Real_one = D[i+n] #one-step real value

        ma = min(MA_1_estimator(INS))

        Fore_one=0

        for j in range(1,len(INS)+1): #j=1,...,n

            Fore_one+=(-1)**(j-1)*(ma**j)*INS[-j]

        Lr.append(Real_one)

        Lf.append(Fore_one)

        Le.append(Real_one - Fore_one)

    Le = np.array(Le)

    MAE = np.mean(np.abs(Le))

    RMSE = (np.mean(Le**2))**0.5

    print(MAE, RMSE,'=MAE,RMSE')

    plt.plot(Lr, 'b', label = "Real value", marker = 'o')

    plt.plot(Lf, 'r', label = "Forecast", marker = '*')
```

```
    plt.legend()

    plt.show()

#모델 설정

Ma1 = ARMA_11(1,0,0.4,1,500)
```



```
ERROR_one_step_forecast_MA1(Ma1, 30)
1.1071875440496513 1.2820995714947057 =MAE,RMSE
```

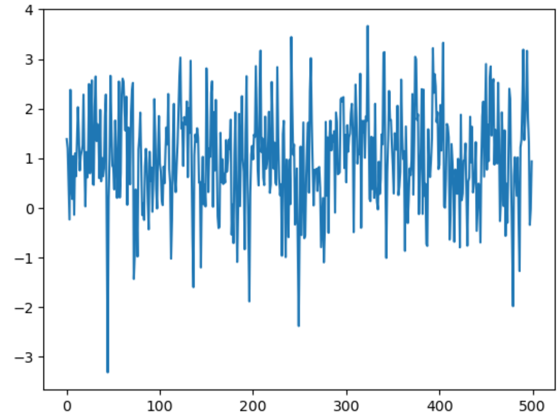
(b) 위의 Ma1 데이터에서 OLSE of AR(1) 를 사용하여 one-step 예측 그래프를 그리고, MAE, RMSE 를 계산하시오.

```
#코드

def OLSE_AR_1(D): #D=data

    n = len(D)

    X = np.array(D) - np.mean(D) #평균이 0인 주어진 데이터

    L_num, L_den = [],[]

    for t in range(1,n):
```

```
        L_num.append(X[t]*X[t-1])
        L_den.append(X[t-1]**2)
    olse = sum(L_num)/sum(L_den) #for phi = phi_1
    hat_phi0 = np.mean(D)*(1-olse) # mu = phi0/(1-phi_1)
    hat_phi1 = olse
    return hat_phi0, hat_phi1

def ERROR_one_step_forecast_AR1(D,m): #RMSE, MAE
    T = len(D)
    n = T - m
    Lr,Lf,Le = [],[],[]
    for i in range(m): #i = 0,1,...,m-1 총 m개
        INS = D[i:i+n] #of size n, from i, i+1,...,i+n-1
        Real_one = D[i+n] #one-step real value
        mu = np.mean(INS)
        olse = OLSE_AR_1(INS) #phi0, phi1
        Xt = INS[-1]
        Fore_one = mu + olse[1]*(Xt-mu)
        Lr.append(Real_one)
        Lf.append(Fore_one)
        Le.append(Real_one - Fore_one)
    Le = np.array(Le)
    MAE = np.mean(np.abs(Le))
    RMSE = (np.mean(Le**2))**0.5
    print(MAE, RMSE,'=MAE,RMSE')
    plt.plot(Lr, 'b', label = "Real value", marker = 'o')
    plt.plot(Lf, 'r', label = "Forecast", marker = '*')
    plt.legend()
    plt.show()
```

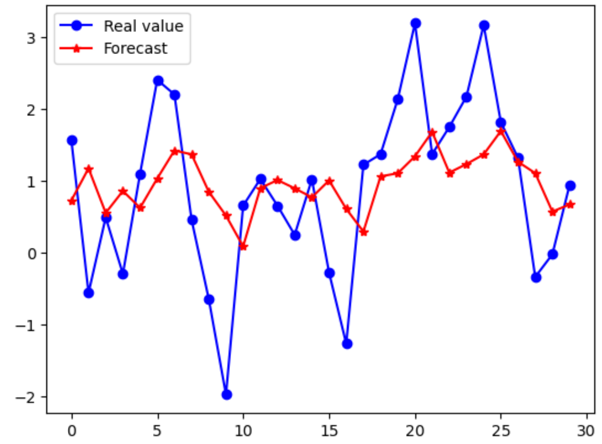
#모델 설정

Ma1 = ARMA_11(1,0,0.4,1,500)



ERROR_one_step_forecast_AR1(Ma1, 30)

0.8907553272603717 1.0920236284670575 =MAE,RMSE

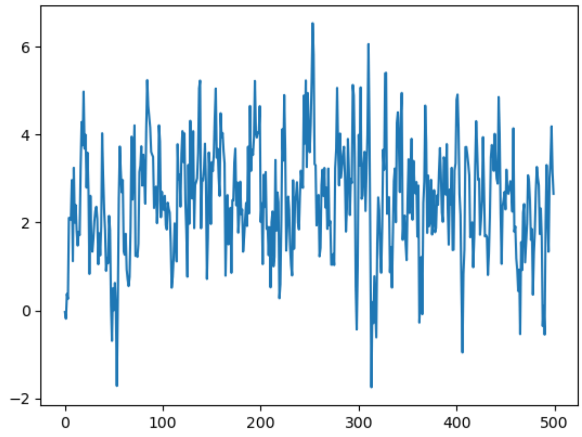


###HW: [4] 위의 Ar1 데이터에서 MLE $\hat{\alpha}, \hat{\mu}\ \hat{\phi}, \hat{\sigma}_w^2$ 를 계산하시오 (파이썬 코딩으로)

#코드

#모델 설정

Ar1 = ARMA_11(1,0.6,0,1,500)




```
import numpy as np


def AR1_estimator(D):
    D = np.array(D)
    n = len(D)
    x_bar_1 = sum(D[:-1])/(n-1)
    x_bar_2 = sum(D[1:])/(n-1)
    numerator = np.sum(D[1:] * D[:-1]) - (n - 1) * x_bar_1 * x_bar_2
    denominator = np.sum(D[1:]**2) - (n - 1) * x_bar_1**2
    phi_hat = numerator / denominator
    alpha_hat = x_bar_2 - phi_hat * x_bar_1
    mu_hat = alpha_hat / (1 - phi_hat)
    Sc = np.sum((D[1:] - alpha_hat - phi_hat * D[:-1]) ** 2)
    sigma_w_hat_sq = Sc / (n - 1)
    return alpha_hat, mu_hat, phi_hat, sigma_w_hat_sq


# AR1 데이터에 대해 함수를 호출
# AR1 데이터에 대해 함수를 호출
alpha_hat, mu_hat, phi_hat, sigma_w_hat_sq = AR1_estimator(Ar1)
print('alpha_hat:', alpha_hat)
print('mu_hat:', mu_hat)
```

```
print('phi_hat:', phi_hat)

print('sigma_w_hat^2:', sigma_w_hat_sq)
```

```
alpha_hat: 1.0146538116120472
mu_hat: 2.5297735072429455
phi_hat: 0.5989151563541117
sigma_w_hat^2: 1.0365975031416284
```

###HW: [5] 위의 Ar1 데이터에서 세 가지 추정법 OLSE, Yule-Walker, MLE을 사용하여 one-step 예측값을 계산하고, MAE, RMSE를 비교하여 어떤 추정법이 적절한지 판단하시오.

```
#코드

def FORE_one_step_AR1(D):

    mu = np.mean(D)

    olse = OLSE_AR_1(D)

    yw = YW_AR_1(D)

    mle = AR1_estimator(D)

    Xt = D[-1] #the last

    Xt1_olse = mu + olse[1]*(Xt-mu) #one-step forecast

    Xt1_yw = yw[0] + yw[1]*Xt

    Xt1_mle = mle[1] + mle[2]*(Xt-mle[1])


    print('olse fore = ',Xt1_olse)

    print('yw fore = ',Xt1_yw)

    print('mle fore = ',Xt1_mle)


FORE_one_step_AR1(Ar1)
```

```
olse fore =  2.6002558261935635
yw fore =  2.6002538803150754
mle fore =  2.6048311173707503
```

```
def ERROR_one_step_forecast(D,m): #RMSE, MAE 강의노트 오류
    T = len(D)
    n = T - m
    Lr,Lf1,Lf2,Lf3,Le1,Le2,Le3 = [],[],[],[],[],[],[]
    for i in range(m):
        INS = D[i:i+n]
        Real_one = D[i+n]
        mu = np.mean(INS)
        olse = OLSE_AR_1(D)
        yw = YW_AR_1(D)
        mle = AR1_estimator(D)
        Xt = INS[-1]
        Fore_one1 = mu + olse[1]*(Xt-mu)
        Fore_one2 = yw[0] + yw[1]*Xt
        Fore_one3 =  mle[1] + mle[2]*(Xt-mle[1])
        Lr.append(Real_one)
        Lf1.append(Fore_one1)
        Lf2.append(Fore_one2)
        Lf3.append(Fore_one3)
        Le1.append(Real_one - Fore_one1)
        Le2.append(Real_one - Fore_one2)
        Le3.append(Real_one - Fore_one3)
    Le1 = np.array(Le1)
    Le2 = np.array(Le2)
    Le3 = np.array(Le3)
    MAE1 = np.mean(np.abs(Le1))
    RMSE1 = (np.mean(Le1**2))**0.5
    MAE2 = np.mean(np.abs(Le2))
    RMSE2 = (np.mean(Le2**2))**0.5
    MAE3 = np.mean(np.abs(Le3))
```

```
    RMSE3 = (np.mean(Le3**2))**0.5

    print('olse MAE = ',MAE1, 'olse RMSE = ',RMSE1)

    print('yw MAE = ',MAE2, 'yw RMSE = ',RMSE2)

    print('mle MAE = ',MAE3, 'mle RMSE = ',RMSE3
```

```
ERROR_one_step_forecast(Ar1, 30)
```

```
olse MAE =  0.7172254212284354 olse RMSE =  0.9291518385332342
yw MAE =  0.716850005984441 yw RMSE =  0.925062848830805
mle MAE =  0.7165626803459162 mle RMSE =  0.9264829615804274
```

➔ 다 비슷한 결과지만 그 중에서도 mle 추정법이 MAE, RMSE가 가장 낮게 나타난다.

###HW: [6] 위의 Ar1 데이터에서 세 가지 추정법 OLSE, Yule-Walker, MLE을 사용하여 two-step 예측값을 계산하고, MAE, RMSE를 비교하여 어떤 추정법이 적절한지 판단하시오.

```
#코드

def FORE_two_step_AR1(D):

    mu = np.mean(D)

    olse = OLSE_AR_1(D)

    yw = YW_AR_1(D)

    mle = AR1_estimator(D)

    Xt = D[-1] #the last

    Xt1_olse = mu + olse[1]*(Xt-mu) #one-step forecast

    Xt2_olse = mu + olse[1]*(Xt1_olse-mu)

    Xt1_yw = yw[0] + yw[1]*Xt

    Xt2_yw = yw[0] + yw[1]*Xt1_yw

    Xt1_mle = mle[1] + mle[2]*(Xt-mle[1])

    Xt2_mle = mle[1] + mle[2]*(Xt1_mle-mle[1])


    print('olse two fore = ',Xt2_olse)
```

```
    print('yw two fore = ',Xt2_yw)

    print('mle two fore = ',Xt2_mle)


FORE_two_step_AR1(Ar1)


olse two fore =   2.5671304681727105
yw two fore =   2.5671281174494904
mle two fore =   2.574726647548206


def ERROR_two_step_forecast(D,m): #RMSE, MAE 강의노트 오류

    T = len(D)

    n = T - m

    Lr,Lf1,Lf2,Lf3,Le1,Le2,Le3 = [],[],[],[],[],[],[]

    for i in range(m):

        INS = D[i:i+n]

        Real_one = D[i+n]

        mu = np.mean(INS)

        olse = OLSE_AR_1(D)

        yw = YW_AR_1(D)

        mle = AR1_estimator(D)

        Xt = INS[-1]

        Fore_one1 = mu + olse[1]**2*(Xt-mu)

        Fore_one2 = yw[0] + yw[1]**2*Xt

        Fore_one3 =   mle[1] + mle[2]**2*(Xt-mle[1])

        Lr.append(Real_one)

        Lf1.append(Fore_one1)

        Lf2.append(Fore_one2)

        Lf3.append(Fore_one3)

        Le1.append(Real_one - Fore_one1)

        Le2.append(Real_one - Fore_one2)

        Le3.append(Real_one - Fore_one3)
```

```
    Le1 = np.array(Le1)

    Le2 = np.array(Le2)

    Le3 = np.array(Le3)

    MAE1 = np.mean(np.abs(Le1))

    RMSE1 = (np.mean(Le1**2))**0.5

    MAE2 = np.mean(np.abs(Le2))

    RMSE2 = (np.mean(Le2**2))**0.5

    MAE3 = np.mean(np.abs(Le3))

    RMSE3 = (np.mean(Le3**2))**0.5

    print('olse MAE = ',MAE1, 'olse RMSE = ',RMSE1)

    print('yw MAE = ',MAE2, 'yw RMSE = ',RMSE2)

    print('mle MAE = ',MAE3, 'mle RMSE = ',RMSE3)


ERROR_two_step_forecast(Ar1, 30)

olse MAE =   0.7659119229830122 olse RMSE =   0.9946460486547783
yw MAE =   0.8333430776719468 yw RMSE =   0.996196400055147
mle MAE =   0.7640223306879689 mle RMSE =   0.9913948671513667
```

➔ 다 비슷한 결과지만 그 중에서도 mle 추정법이 MAE, RMSE가 가장 낮게 나타난다.