스마트테크놀로지AIR

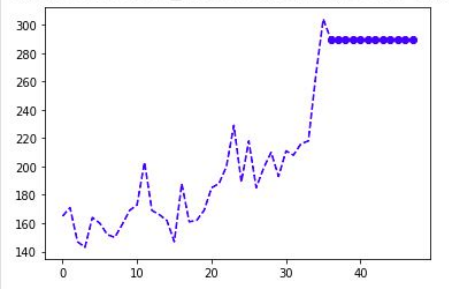
HW-Forecasting

2016147530
권예린

SES 단순평활예측법

1. SimpleExpSmoothing 모듈을 사용하여 모델링하기+plot

   SimpleExpSmoothing 모듈을 이용하여 initial level을 초반 12개월치의 데이터의 평균값으로 설정한 뒤 36개월 이후 12개월을 예측했다.



2. forecast 함수로 12개월 예측하기 , sse로 standard error 판단하기

```
36    289.901044
37    289.901044
38    289.901044
39    289.901044
40    289.901044
41    289.901044
42    289.901044
43    289.901044
44    289.901044
45    289.901044
46    289.901044
47    289.901044
dtype: float64
```

```
#standard error
math.sqrt(SES.sse/(36-1))
```

```
20.393115830049346
```

3. forecast로 예측값 구하기 (모듈 사용하지 않고 알파값을 임의로 설정할 경우)

   initial level을 앞선 12개월의 평균으로 설정한 뒤, 알파값을 0.5로 주었을 때의 예측 결과이다.

```
#initial level : 12개월 평균
initial_level = DATA[:12].mean()

level = initial_level
alpha = 0.5
for i in range(36):
  level = level+alpha*(demands[i]-level)
print(level)
```

```
271.648188492778
```

```
[271.648188492778,
 271.648188492778,
 271.648188492778,
 271.648188492778,
 271.648188492778,
 271.648188492778,
 271.648188492778,
 271.648188492778,
 271.648188492778,
 271.648188492778,
 271.648188492778,
 271.648188492778]
```

Holt 이중평활예측법

1. Trend 확인

```
[ ] linreg = linregress(x=DATA.index, y=DATA)
    print(linreg.slope, linreg.pvalue)
```

```
2.5386100386100385 1.1694156537818794e-08
```

기울기가 약 2.53으로 존재하고 pvalue가 0에 가깝기 때문에 trend가 존재한다고 볼 수 있다.

2. 데이터 절반의 initial level과 initial slope확인

```
#데이터 절반의 initial level, initial slope 확인
lin = linregress(x=DATA1[:18].index, y=DATA1[:18])
initial_slope = lin.slope
initial_level = lin.intercept
print(initial_level,initial_slope)
```

```
155.8823529411765 0.8369453044375644
```

3. 초기값 토대로 level, trend 구하기 (alpha, gamma 둘 다 0.5일 때), 12개월 예측결과

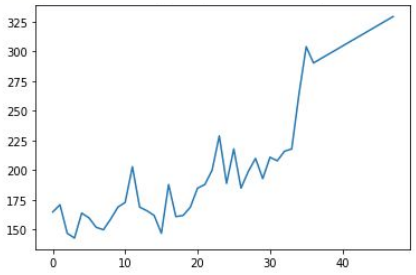
식)

```
one_step_forecast[i] = level[i-1]+slope[i-1]
level[i] =
level[i-1]+slope[i-1]+alpha*(demands[i]-one_step_forecast[i])
slope[i] =
slope[i-1]+alpha*gamma*(demands[i]-one_step_forecast[i])
```

결과)

```
[307.633073002067,
 334.2595825478553,
 360.8860920936436,
 387.51260163943186,
 414.13911118522014,
 440.7656207310084,
 467.3921302767967,
 494.018639822585,
 520.6451493683733,
 547.2716589141615,
 573.8981684599498,
 600.5246780057381]
```

4. Exponential Smoothing으로 최적화



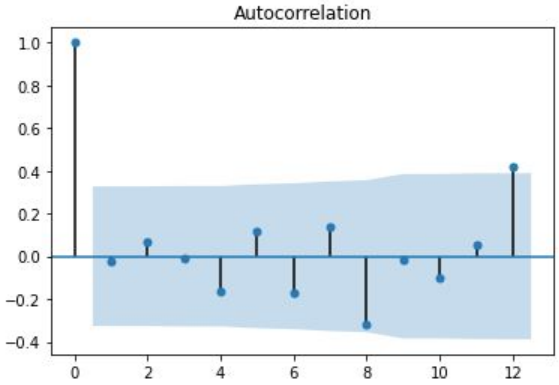
```
#standard error
math.sqrt(Hz.sse/(36-1))

19.745150462645533
```

앞선 SES보다 standard error값이 줄었다. 예측 에러값이 줄어들었기 때문에 보다 정확한 결과라 볼 수 있다.

5. deviations from mean

```
0      2.454629
1      3.283281
2    -26.435652
3    -16.515906
4     11.848735
...
31    -3.725676
32     3.187006
33    -0.468307
34    42.291861
35    50.795523
```

6. 자기상관 그래프



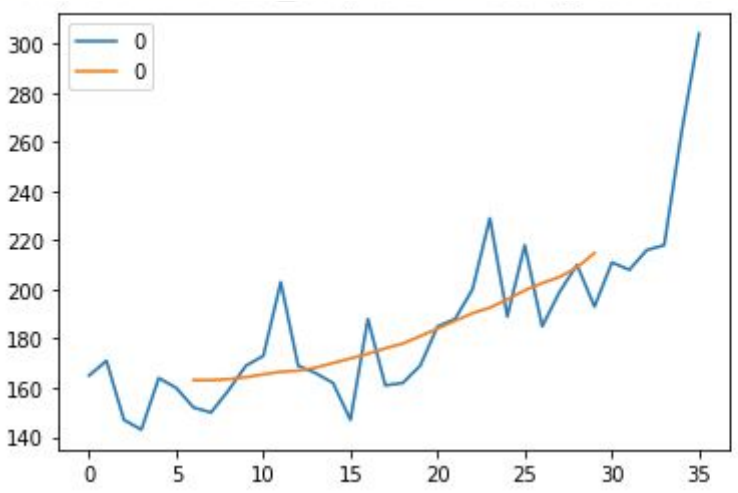
자기상관계수가 12일 때 높은 값을 가진다. 따라서 이 데이터는 12개월을 주기로 변화하는 특징을 가진다고 볼 수 있다.

7. 자기상관계수

```
array([ 1.        , -0.01736212,  0.06753761, -0.00440455, -0.151768  ,
        0.11991857, -0.16848853,  0.13962347, -0.3045567 , -0.0058307 ,
       -0.09231985,  0.05242037,  0.40895114])
```

Holt-winters 삼중평활예측법

1. MA(moving average) smoothed 구하기



MA smoothed: 두 연속한 12개월치 평균값의 중간값
파란색이 기존 데이터, 주황색이 MA smoothed이다.

2. initial seasonal factors X 3 만들기

```
#initian seasonal factors
seasonal_factor_init = []
for i in range(6,12):
  seasonal_factor_init.append((est[i]+est[i+12])/2)
for i in range(6):
  seasonal_factor_init.append((est[i]+est[i+12])/2)
```

x3 : 위 결과를 3배
결과:

```
[0.9882333992444625,    0.9208375893769152,
 1.0394595142086607,    0.9266209436718105,
 0.9329332917124478,    0.9884907528722957,
 0.9125977559105471,    1.0162014531884298,
 1.0430106047420362,    1.0480526558060541,
 0.9064424515366746,    1.2040049076184032]
```
...

3. Deseasonalized Data 만들기

```
deseasonalized[i] = demands[i] / x3[i]
```

```
[166.96460585743006,   229.13921242373823,
 164.50857167840934,   224.47150738443617,
 157.5675359705235,    218.51494247402968,
 156.69554201053379,   214.5243930876147,
 157.2371357054049,    251.89574067436374,
 176.5142395181901,    252.49066517621674]
```
...

4. Level과 Trend와 Seasonal_adj값으로 forecast해보기
    a. Deseasonalized data에 대해 linear regression

```
print(initial_level,initial_slope)
```

```
144.42354225448122 2.2904500049389425
```

    b. alpha, gamma, delta값을 모두 0.5로 설정했을 때 결과

식)

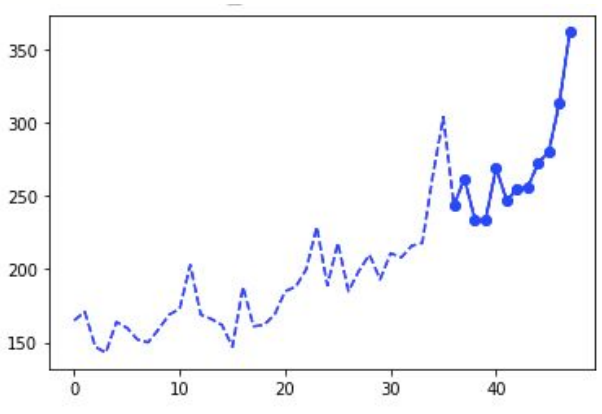
```
prev_level, prev_slope = level,slope
level =
alpha*demands[i]/seasonal_factor_init[i]+(1-alpha)*(prev_level+pr
ev_slope)
slope = gamma*(level-prev_level)+(1-gamma)*prev_slope
seasonal_factor_init.append(delta*(demands[i]/level)+(1-delta)*se
asonal_factor_init[i])
```

결과)

```
[258.3415200040934,
 280.3639523439052,
 253.28021140480934,
 255.96408917080052,
 298.45419301816605,
 275.74146769136377,
 288.1190092946273,
 292.75019901953607,
 318.095985870473,
 336.2107591401663,
 375.27294894525846,
 432.0278289525443]
```

5. 최적화 된 forecast 값, 그래프

```
36    243.653544
37    261.894278
38    233.728698
39    233.769819
40    269.091744
41    247.207091
42    254.738356
43    256.025261
44    272.616274
45    280.410811
46    313.900717
47    362.089388
```



6. standard error

```
math.sqrt(season.sse/(36-1))
```

```
8.382520444295832
```

가장 작은 standard error값을 가지기 때문에 세 방법 중 예측 정확도가 가장 높다고 볼 수 있다.