시계열 분석 (Auto ARIMA)

#01. 작업준비

패키지 참조

`pmdarima` 패키지의 설치가 필요하다

```
from pandas import read_excel, DataFrame, Series
from matplotlib import pyplot as plt
from matplotlib import dates as mdates
from pmdarima.arima import auto_arima
from datetime import timedelta
import seaborn as sb
import sys
```

데이터 가져오기

```
df = read_excel("https://data.hossam.kr/E06/air_passengers.xlsx", index_
df.head()
```

	Passengers
Month	
1949-01-01	112
1949-02-01	118
1949-03-01	132
1949-04-01	129
1949-05-01	121

그래프 초기화

```
plt.rcParams["font.family"] = 'AppleGothic' if sys.platform == 'darwin'
plt.rcParams["font.size"] = 12
plt.rcParams["axes.unicode_minus"] = False
```

#02. 데이터 분석

데이터 분할

처음부터 70% 위치까지의 데이터

분석모델 구축용(=학습용)

```
# 처음부터 70% 위치 전까지 분할
train = df[:int(0.7*len(df))]
train.head()
```

	Passengers
Month	
1949-01-01	112
1949-02-01	118
1949-03-01	132
1949-04-01	129
1949-05-01	121

검증용 데이터 (나머지 30%)

```
# 70% 위치부터 끝까지 분할
test = df[int(0.7*len(df)):]
test.head()
```

	Passengers
Month	
1957-05-01	355
1957-06-01	422

	Passengers
Month	
1957-07-01	465
1957-08-01	467
1957-09-01	404

모델 구축

시계열 데이터를 계절 ARIMA 모델에 맞추려고 할 때 첫 번째 목표는 측정항목을 최적화하는 $ARIMA(p,d,q)(P,D,Q)M$ 값을 찾는 것

M값을 고정한 상태에서 0부터 $(p,d,q)(P,D,Q)$로 주어진 값의 범위 안에서 최적의 값을 검색한다.

```
my_p = 5     # 적절히 넉넉히
my_d = 2     # 차분 횟수 (검증한 결과를 활용)
my_q = 5     # 적절히 넉넉히
my_s=12      # 계절성 주기 (분석가가 판단)

model = auto_arima(
    y=train,                # 모델링하려는 시계열 데이터 또는 배열
    start_p=0,              # p의 시작점
    max_p=my_p,                # p의 최대값
    d=my_d,                    # 차분 횟수
    start_q=0,              # q의 시작점
    max_q=my_q,                # q의 최대값
    seasonal=True,          # 계절성 사용 여부
```

```
    m=my_s,                    # 계절성 주기
    start_P=0,                 # P의 시작점
    max_P=my_p,                # P의 최대값
    D=my_d,                    # 계절성 차분 횟수
    start_Q=0,                 # Q의 시작점
    max_Q=my_q,                # Q의 최대값
    trace=True                 # 학습 과정 표시 여부
)
print(model.summary())
```

```
Performing stepwise search to minimize aic
 ARIMA(0,2,0)(0,2,0)[12]             : AIC=700.560, Time=0.04 sec
 ARIMA(1,2,0)(1,2,0)[12]             : AIC=628.108, Time=0.27 sec
 ARIMA(0,2,1)(0,2,1)[12]             : AIC=inf, Time=0.50 sec
 ARIMA(1,2,0)(0,2,0)[12]             : AIC=656.611, Time=0.06 sec
 ARIMA(1,2,0)(2,2,0)[12]             : AIC=618.256, Time=0.45 sec
 ARIMA(1,2,0)(3,2,0)[12]             : AIC=614.066, Time=0.87 sec
 ARIMA(1,2,0)(4,2,0)[12]             : AIC=609.992, Time=3.41 sec
 ARIMA(1,2,0)(5,2,0)[12]             : AIC=inf, Time=7.91 sec
 ARIMA(1,2,0)(4,2,1)[12]             : AIC=inf, Time=10.24 sec
 ARIMA(1,2,0)(3,2,1)[12]             : AIC=inf, Time=2.72 sec
 ARIMA(1,2,0)(5,2,1)[12]             : AIC=inf, Time=7.96 sec
 ARIMA(0,2,0)(4,2,0)[12]             : AIC=651.167, Time=2.53 sec
 ARIMA(2,2,0)(4,2,0)[12]             : AIC=605.939, Time=3.59 sec
 ARIMA(2,2,0)(3,2,0)[12]             : AIC=606.557, Time=1.08 sec
 ARIMA(2,2,0)(5,2,0)[12]             : AIC=inf, Time=8.47 sec
 ARIMA(2,2,0)(4,2,1)[12]             : AIC=inf, Time=11.74 sec
 ARIMA(2,2,0)(3,2,1)[12]             : AIC=inf, Time=3.96 sec
 ARIMA(2,2,0)(5,2,1)[12]             : AIC=inf, Time=14.84 sec
 ARIMA(3,2,0)(4,2,0)[12]             : AIC=601.906, Time=4.37 sec
```

```
 ARIMA(3,2,0)(3,2,0)[12]             : AIC=604.074, Time=2.75 sec
 ARIMA(3,2,0)(5,2,0)[12]             : AIC=inf, Time=8.89 sec
 ARIMA(3,2,0)(4,2,1)[12]             : AIC=inf, Time=15.03 sec
 ARIMA(3,2,0)(3,2,1)[12]             : AIC=inf, Time=10.46 sec
 ARIMA(3,2,0)(5,2,1)[12]             : AIC=inf, Time=18.88 sec
 ARIMA(4,2,0)(4,2,0)[12]             : AIC=596.152, Time=4.44 sec
 ARIMA(4,2,0)(3,2,0)[12]             : AIC=597.717, Time=3.31 sec
 ARIMA(4,2,0)(5,2,0)[12]             : AIC=inf, Time=8.83 sec
 ARIMA(4,2,0)(4,2,1)[12]             : AIC=inf, Time=13.65 sec
 ARIMA(4,2,0)(3,2,1)[12]             : AIC=inf, Time=12.79 sec
 ARIMA(4,2,0)(5,2,1)[12]             : AIC=inf, Time=17.03 sec
 ARIMA(5,2,0)(4,2,0)[12]             : AIC=590.926, Time=7.23 sec
 ARIMA(5,2,0)(3,2,0)[12]             : AIC=590.534, Time=5.29 sec
 ARIMA(5,2,0)(2,2,0)[12]             : AIC=597.117, Time=1.24 sec
 ARIMA(5,2,0)(3,2,1)[12]             : AIC=inf, Time=12.09 sec
 ARIMA(5,2,0)(2,2,1)[12]             : AIC=inf, Time=3.40 sec
 ARIMA(5,2,0)(4,2,1)[12]             : AIC=inf, Time=15.32 sec
 ARIMA(5,2,1)(3,2,0)[12]             : AIC=585.655, Time=8.15 sec
 ARIMA(5,2,1)(2,2,0)[12]             : AIC=inf, Time=3.48 sec
 ARIMA(5,2,1)(4,2,0)[12]             : AIC=inf, Time=14.55 sec
 ARIMA(5,2,1)(3,2,1)[12]             : AIC=inf, Time=12.60 sec
 ARIMA(5,2,1)(2,2,1)[12]             : AIC=inf, Time=4.03 sec
 ARIMA(5,2,1)(4,2,1)[12]             : AIC=inf, Time=16.09 sec
 ARIMA(4,2,1)(3,2,0)[12]             : AIC=inf, Time=10.63 sec
 ARIMA(5,2,2)(3,2,0)[12]             : AIC=587.252, Time=8.82 sec
 ARIMA(4,2,2)(3,2,0)[12]             : AIC=inf, Time=11.74 sec
 ARIMA(5,2,1)(3,2,0)[12] intercept   : AIC=587.606, Time=10.51 sec

Best model:  ARIMA(5,2,1)(3,2,0)[12]
Total fit time: 346.290 seconds
```

시계열 분석 (Auto ARIMA)

#01. 작업준비

패키지 참조

데이터 가져오기

그래프 초기화

#02. 데이터 분석

데이터 분할

처음부터 70% 위치까지의 데이터

검증용 데이터 (나머지 30%)

모델 구축

모델 학습

잔차 플롯 검토

예상치 생성

관측치와 예상치 비교

SARIMAX Results

Dep. Variable:	y	No. Observations:	
Model:	SARIMAX(5, 2, 1)x(3, 2, [], 12)	Log Likelihood	
Date:	Fri, 04 Aug 2023	AIC	
Time:	14:52:23	BIC	
Sample:	01-01-1949	HQIC	
	- 04-01-1957		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	
ar.L1	-0.5251	0.210	-2.498	0.012	-0.937	
ar.L2	-0.2593	0.225	-1.155	0.248	-0.700	
ar.L3	-0.3176	0.211	-1.505	0.132	-0.731	
ar.L4	-0.3068	0.233	-1.314	0.189	-0.764	
ar.L5	-0.1050	0.205	-0.511	0.609	-0.507	
ma.L1	-0.8206	0.172	-4.768	0.000	-1.158	
ar.S.L12	-1.1211	0.117	-9.567	0.000	-1.351	
ar.S.L24	-0.8605	0.171	-5.026	0.000	-1.196	
ar.S.L36	-0.4895	0.157	-3.110	0.002	-0.798	
sigma2	88.3178	18.953	4.660	0.000	51.171	1

Ljung-Box (L1) (Q):	0.00	Jarque-Bera (JB):	
Prob(Q):	0.96	Prob(JB):	
Heteroskedasticity (H):	0.79	Skew:	
Prob(H) (two-sided):	0.55	Kurtosis:	

```
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (c
```

모델 학습

관측치를 모델에 적제하여 학습을 수행한다.

```
model.fit(train)
```

```
ARIMA
ARIMA(5,2,1)(3,2,0)[12]
```

잔차 플롯 검토

왼쪽 상단: 잔차 오차는 평균 0을 중심으로 변동하고 균일한 분산을 갖는 것으로 보임

오른쪽 상단: 밀도 도표는 평균이 0인 정규 분포를 나타냄

왼쪽 하단: 모든 점이 빨간색 선과 완벽하게 일치해야 함. 편차가 크면 분포가 왜곡되었음을 의미합니다.

오른쪽 아래: 상관관계도(ACF 플롯이라고도 함)는 잔차 오류가 자동 상관되지 않음을 보여줌. 모든 자기상관은 모델에서 설명되지 않는 잔차 오류에 일부 패턴이 있음을 의미하기 때문에 모델에 대해 더 많은 X(예측 변수)를 찾아야 함.

```
model.plot_diagnostics(figsize=(15,10))
plt.show()
```

시계열 분석 (Auto ARIMA)

#01. 작업준비

패키지 참조

데이터 가져오기

그래프 초기화

#02. 데이터 분석

데이터 분할

처음부터 70% 위치까지의 데이터

검증용 데이터 (나머지 30%)

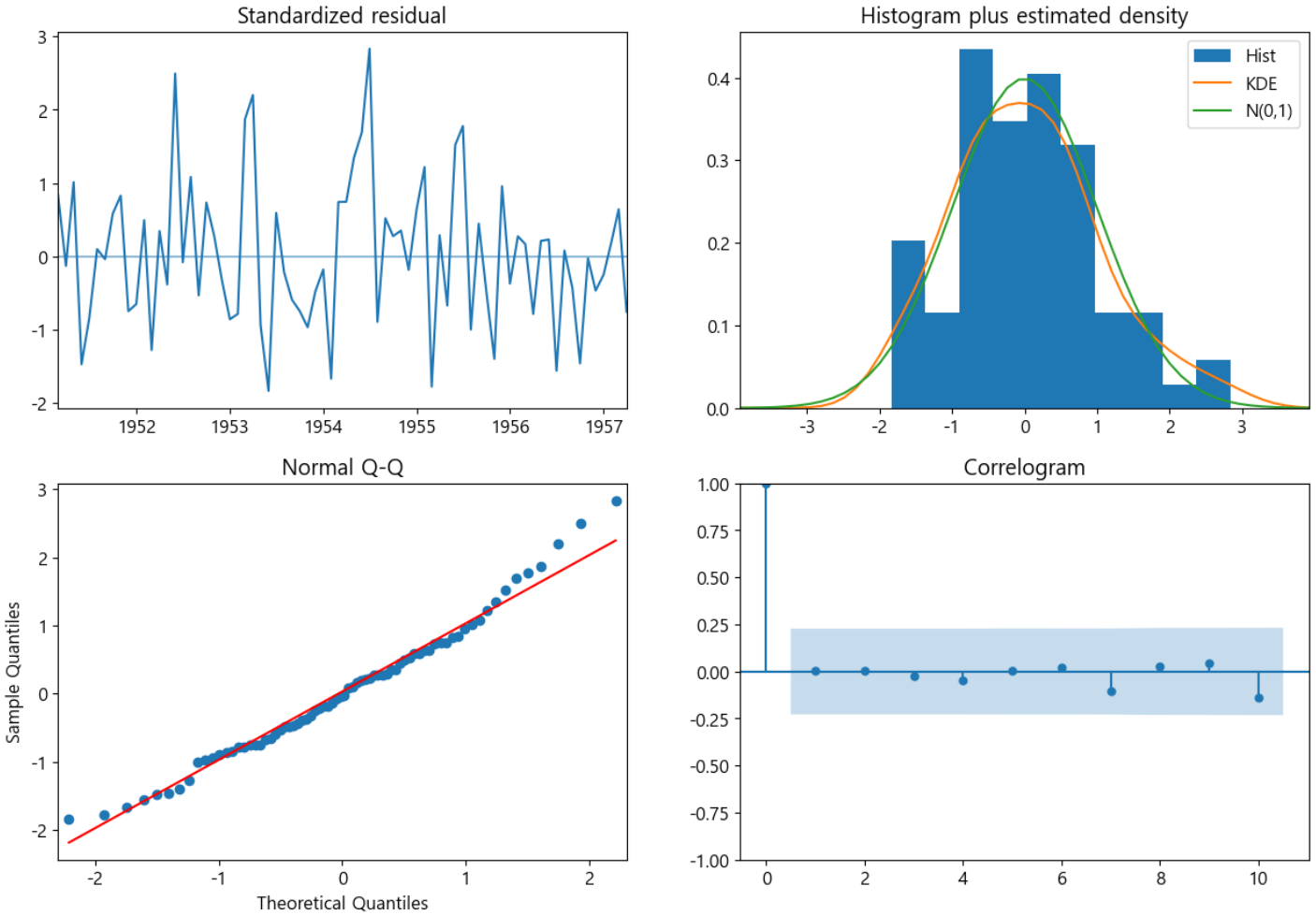
모델 구축

모델 학습

잔차 플롯 검토

예상치 생성

관측치와 예상치 비교



> 전반적으로 잘 맞는것 같다!!!

예상치 생성

학습결과를 토대로 주어진 `n_periods` 수 만큼의 이후 데이터를 예상하여 결과를 반환한다.

```
# 원본 데이터 이후 10단계 까지 예측
y_predict = model.predict(n_periods=int(len(test)+10))
```

```
y_predict
```

```
1957-05-01    353.001884
1957-06-01    404.404671
1957-07-01    458.201793
1957-08-01    438.689621
1957-09-01    388.085053
1957-10-01    331.313503
1957-11-01    287.557417
1957-12-01    327.148708
1958-01-01    337.122339
1958-02-01    318.144484
1958-03-01    370.266764
1958-04-01    360.138673
1958-05-01    368.959373
1958-06-01    435.883864
1958-07-01    496.438803
1958-08-01    468.139086
1958-09-01    412.561605
1958-10-01    347.657604
1958-11-01    302.561163
1958-12-01    346.764795
1959-01-01    358.430437
1959-02-01    341.134185
1959-03-01    392.107616
1959-04-01    382.591081
1959-05-01    388.221442
1959-06-01    462.217613
1959-07-01    526.212535
1959-08-01    494.128404
```

```
1959-09-01    431.669748
1959-10-01    356.456026
1959-11-01    303.324582
1959-12-01    352.010052
1960-01-01    360.138340
1960-02-01    338.451191
1960-03-01    396.043429
1960-04-01    380.204893
1960-05-01    387.902897
1960-06-01    467.494133
1960-07-01    532.155387
1960-08-01    497.101941
1960-09-01    425.341205
1960-10-01    339.716408
1960-11-01    283.666773
1960-12-01    330.505222
1961-01-01    340.060004
1961-02-01    312.796958
1961-03-01    374.291074
1961-04-01    355.357385
1961-05-01    362.118872
1961-06-01    445.110866
1961-07-01    519.065815
1961-08-01    473.217885
1961-09-01    397.377063
1961-10-01    302.232564
Freq: MS, dtype: float64
```

관측치와 예상치 비교

시계열 분석 (Auto ARIMA)

#01. 작업준비

패키지 참조

데이터 가져오기

그래프 초기화

#02. 데이터 분석

데이터 분할

처음부터 70% 위치까지의 데이터

검증용 데이터 (나머지 30%)

모델 구축

모델 학습

잔차 플롯 검토

예상치 생성

관측치와 예상치 비교

```
plt.figure(figsize=(20,8))

# 앞 70%의 원본 데이터
sb.lineplot(data=train, x=train.index, y='Passengers', label='Original(T

# 뒤 30%의 원본 데이터
sb.lineplot(data=test, x=test.index, y='Passengers', label='Original(Tes

# 뒤 30% + 10단계에 대한 예측 데이터
sb.lineplot(x=y_predict.index, y=y_predict.values, label='Predict(Test)'

plt.xlabel('Month')
plt.ylabel('Passengers')

# 그래프의 x축이 날짜로 구성되어 있을 경우 형식 지정
monthyearFmt = mdates.DateFormatter('%y.%m.%d')
plt.gca().xaxis.set_major_formatter(monthyearFmt)

plt.grid()
plt.show()
plt.close()
```

시계열 분석 (Auto ARIMA)

#01. 작업준비

패키지 참조

데이터 가져오기

그래프 초기화

#02. 데이터 분석

데이터 분할

처음부터 70% 위치까지의 데이터

검증용 데이터 (나머지 30%)

모델 구축

모델 학습

잔차 플롯 검토

예상치 생성

관측치와 예상치 비교