nipa 정보통신산업진흥원

스마트팩토리 AI교육

공급망관리지능화 : 수요예측 최적화

재단법인 경남테크노파크

KBCI 한국블록체인연구교육원 Korea Blockchain Institute

nipa 정보통신산업진흥원

CONTENTS

이곳에 제목을 입력하세요

- 1. 수요예측 이해
- 2. 시계열분석 기반 수요예측
- 3. 시계열 예측분석
- 4 수요 예측분석
- 5. 시세 예측분석
- 6. 판매량 예측분석

재단법인 경남테크노파크

KBCI 한국블록체인연구교육원 Korea Blockchain Institute

1

수요예측 이해

수요예측이란

수요예측 (demand forecasting)이란?	•기업의 제품과 서비스에 대한 수요의 양과 시기를 예측하는 것 •수요예측이 이루어지면 수요를 충족시키기 위해 필요한 자원에 대한 예측이 이루어지는데 이는 구매되는 부품과 원자재뿐만 아니라 기업의 설비, 기계, 노동력에 대한 양과 시기를 예측하는 것

"기업의 장기계획 수립을 위한 전제가 되며
단기적인 운영관리에 지대한 영향을 주는 활동이다."

예산수립과 비용통제 등에 이용

재무/회계부서

신제품계획수립, 판매원에 대한 보상계획수립 등에 이용

마케팅부서

수요예측 기간

수요예측은 예측되는 기간의 길이에 따라서 단기예측, 중기예측, 장기예측으로 나누어진다.

단기예측 (3개월 ~ 1년)	→	중기예측 (1년 ~ 3년 미만)	→	장기예측 (3년 이상~)

2) 수요자료의 종류

수요자료는 그 자료가 어떤 시간을 기준으로 기록되었는가에 따라서
시계열자료(time series data)와 종단자료(cross-section data)로 나눌 수 있다.

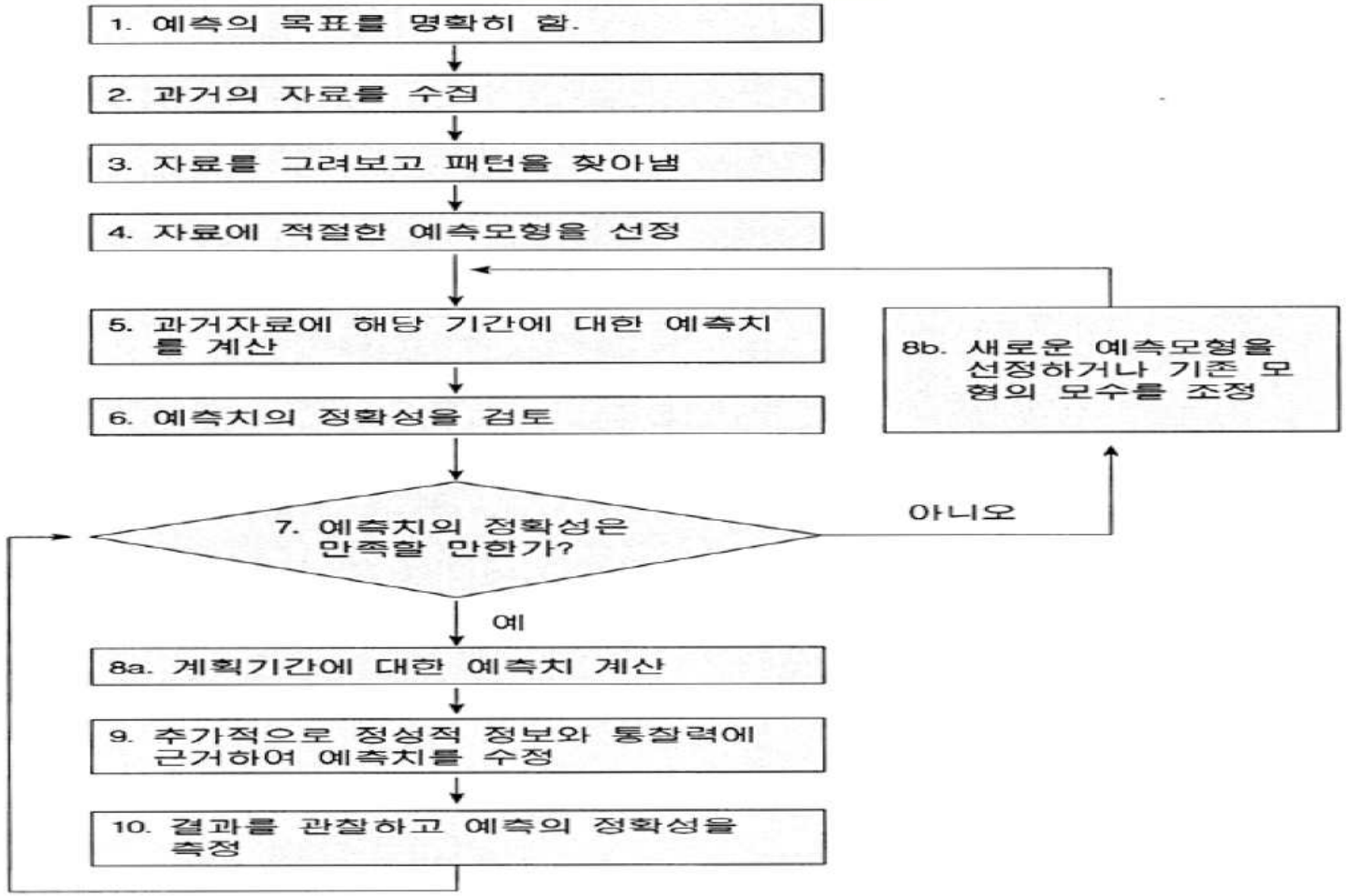
시계열자료 (time series data)	일정한 간격의 시간을 두고 자료를 수집하여 시간이 흐름에 따른 변화를 파악할 수 있도록 기록
종단자료 (cross-section data)	어느 한 시점을 기준으로 자료가 수집되어 그 시점의 상황을 파악할 수 있도록 기록

수요예측의 의의

- 대부분의 제품 및 서비스 수요변동은 다음의 6가지로 설명할 수 있다. (1) 평균수요, (2) 추세변동, (3) 계절변동, (4) 경기순환변동, (5) 자기상관성, (6) 불규칙 변동.
- 경기순환 변동은 선거, 전쟁, 경제적 상황, 사회적 현상들에 의해 영향을 받는다. 불규칙 변동은 설명할 수 없는 수요변동을 가리킨다.
- 추세선의 형태에는 (1) 선형추세선, (2) S-곡선, (3) 점근추세선, (4) 지수곡선이 있다.
- 자기상관성은 "어떤 시점에서의 수요는 그 자체의 과거값과 매우 높은 상관관계가 있다"라는 점을 가리킨다. 예를 들어 만약 어느 대기라인이 길다고 한다면, 잠시 후에도 그 대기라인은 여전히 길다고 예측할 수 있다.

수요예측 절차

수요예측 과정의 단계



수요예측 접근방법

● **학습 데이터 수집**

학습 단계에서는 과거 해당 시점에 영향을 미칠 가능성이 큰 다양한 학습 데이터를 수집해야 합니다. 무엇보다도 중요한 점은 해당 데이터들은 미래에도 수집할 수 있어야 한다는 것 입니다. 미래에 수집할 수 없는 데이터로 학습한 모형은 결국 예측하는 데 사용할 수 없기 때문입니다.

● **변수 생성**

이렇게 다양하게 수집된 학습 데이터들을 이용해 EDA(Exploratory Data Analysis, 탐색적 데이터 분석)와 상관 계수와 같은 검증 통계량을 통해 변수 간의 관계를 파악합니다. 또한 파생 변수들을 생성해 그 파생 변수와 예측하고자 하는 수요와의 관계를 파악하는 과정들을 통해 수요에 큰 영향을 미칠 수 있는 변수들을 생성합니다.

● **학습**

생성한 다양한 변수들이 존재하는 과거 데이터로 시계열 모형(ARIMAX)과 Random Forest, XGBoost 및 LGBM regression 등의 머신러닝 모형을 학습시킵니다.

● **평가**

학습된 모형들을 사용해 과거의 특정 기간을 평가 대상으로 예측하고 각 모형의 수요 예측값과 해당 시점의 실제 수요값을 비교해 최소 오차를 가지는 모형을 최적 모형으로 선정합니다. 정답지를 사용해 예측 정확도를 평가하는 것은 중요합니다.

수요예측 기법유형(1/2)

수요예측기법

정성적 예측기법 (qualitative forecasting method)

- 조직 내외의 사람들의 경험이나 견해와 같은 주관적 요소를 사용하는 예측기법

- 시장조사법
- 패널동의법
- 중역의견법
- 판매원의견합성법
- 수명주기 유추법
- 델파이법

계량적 예측기법 (quantitative forecasting method)

시계열 예측기법 (time series forecasting method)

- 시계열 자료를 이용하여 예측하는 방법

- 이동평균법,
- 가중이동평균법
- 지수평활법 등

인과형 예측기법 (causal forecasting method)

- 어떤 원인에 의하여 수요자료가 영향을 받는다고 생각하여 그 원인과 결과 사이의 관계를 찾아서 예측을 하는 방법

- 회귀분석법 등

수요예측 기법유형(2/2)

수요예측의 방법

(1) 정성적 방법(현장자료 집결화, 시장조사, 패널동의법, 유추법, 델파이 기법 등)
(2) 시계열분석(단순이동평균법, 가중이동평균법, 지수평활법, 회귀분석 등)
(3) 인과분석(회귀분석, 계량경제 모델 등)
(4) 시뮬레이션 모델(컴퓨터에 근거한 동적모델로 만약 가격이 10% 상승하면 수요는 어느 정도 변화될지를 예측함)

수요예측 알고리즘

전통적인 시계열 모형으로는 예측하고자 하는 값이 과거 시점에 영향을 받는 자기 회귀 모형(Autoregression model), 모형이 설명하지 못하는 오차 혹은 변동값을 이용해 현시점의 값을 예측하는 이동 평균 모형(Moving Average model), 차분, 자기 회귀 및 이동 평균을 고려하는 자동 회귀 누적 이동 평균(ARIMA: Autoregressive integrated moving average) 등이 존재합니다.

위와 같은 전통적인 시계열 모형들은 과거 데이터의 패턴과 오차만을 가지고 미래를 예측하려는 점에서 공통점을 가지고 있습니다. 최근에는 과거 데이터 외에도 해당 과거 시점별 수요에 영향을 미치는 독립 변수들을 학습하는 다양한 예측 모형이 만들어졌습니다.

대표적으로 ARIMAX 모형이 있습니다. 이 모형은 예측하려는 수요에 영향을 미치는 독립 변수를 발굴해 미래 시점의 독립 변수 바탕으로 미래 수요를 예측하는 모형입니다.

딥러닝을 이용한 RNN(Recurrent Neural Network, 순환 신경망)의 한 종류인 LSTM(Long Short-Term Memory)도 있으며, Facebook의 forecasting 라이브러리인 Prophet은 GAM(Generalized Additive Model, 일반화 가법 모형)에 시계열 예측으로 발전시킨 모형도 있습니다.

2

시계열분석 기반 수요예측

시계열분석

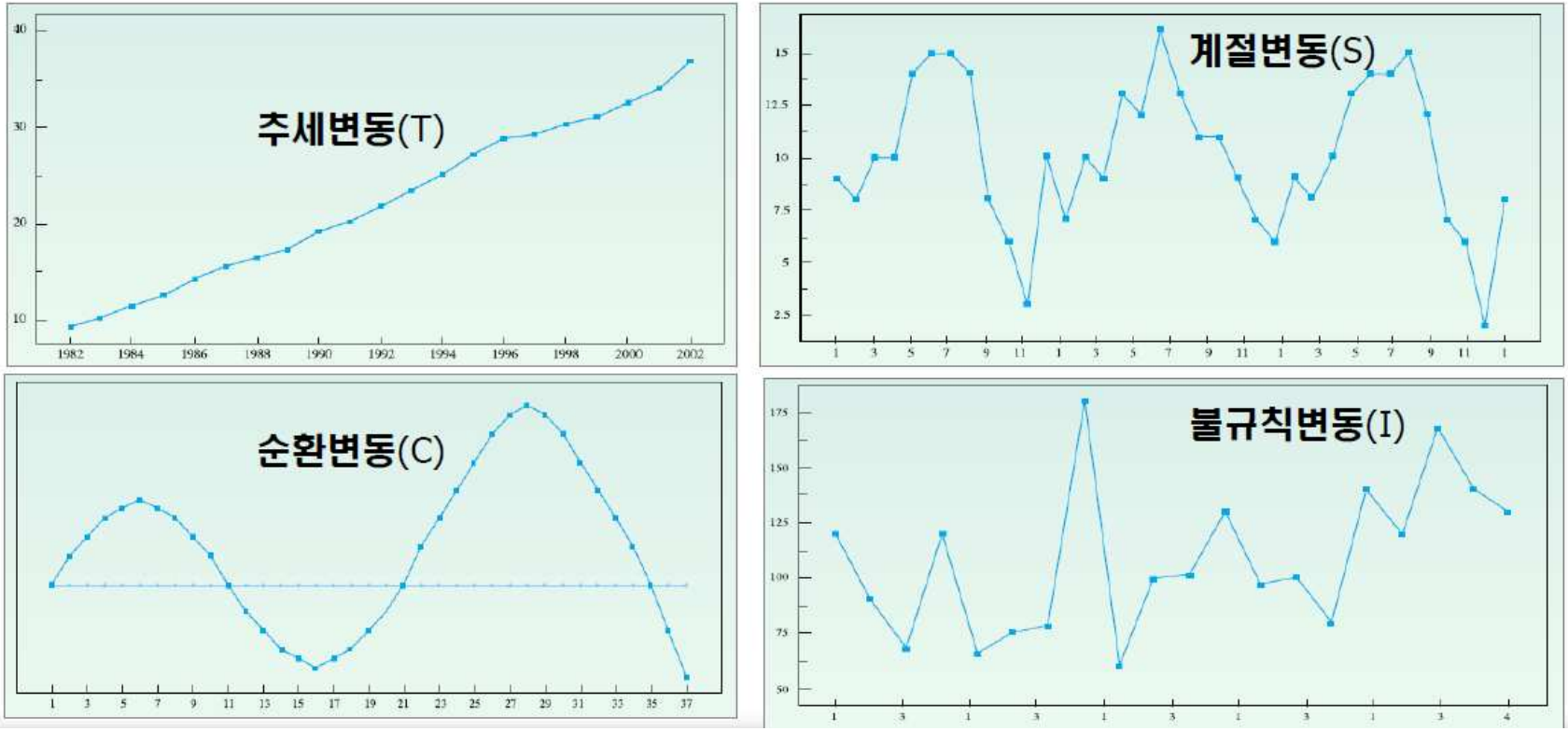
- 단순이동평균법: 제품수요 증감의 변화가 빠르지 않고 계절적 요인이 없을 경우에 활용한다.
- 가중이동평균법: 미래를 예측하는데 있어 좀 더 정확한 것은 최근의 정보이다.
- 지수평활법: 적은 계산으로 예측이 가능하고 공식이 상대적으로 쉽다. 지수평활법은 세가지 요소만 있으면 예측이 가능하다. 즉, (1) 최근의 예측치, (2) 실제치, (3) 평활상수 α 이다. 비교적 안정적으로 성장하는 제품을 생산하는 회사에서 이 상수값은 보통 5-10%이며, 반면 성장속도가 빠른 제품인 경우 최근 정보가 중요하므로 상수는 15-30%가진다. 또한 지수평활법에서는 추세효과를 반영할 수 있다. 이 때에는 α이외에 δ가 사용된다.

시계열 기반 수요예측

- **시계열분석** (Time Series Analysis)

  시계열 데이터로부터 추세나 경향(일정한 패턴 : 규칙성/시계열적 변동)을 파악하여 장래 수요를 예측

- **시계열적 변동의 종류**



시계열자료 특성

시계열 자료

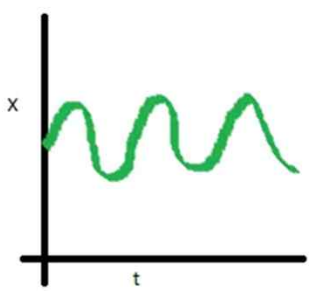
- 시간의 흐름에 따라서 관측된 데이터를 사용

- 주간/월간 단위로 지표를 리뷰해 트렌드에 이상치는 없는 지 모니터링 하는데 사용
- 보통 분기/반기/년간 단위로 다음 지표를 예측하는 데 이용

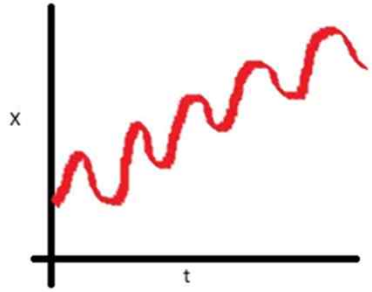
시계열데이터가 정상성을 만족하는 게 중요함

- 비정상성 시계열자료(Non Stationary Series)
  - 시계열 분석하는데 다루기 어려운 시계열 자료(대부분의 시계열자료)
- 정상성 시계열자료(Stationary Series)
  - 비정상 시계열을 핸들링해 다루기 쉬운 시계열 자료로 변환한 자료

시간의 추이에 상관없이 평균이 불변

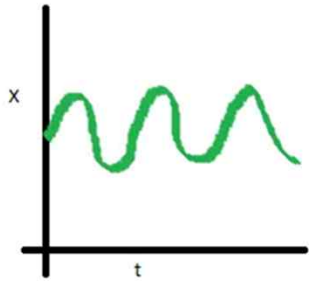


Stationary series

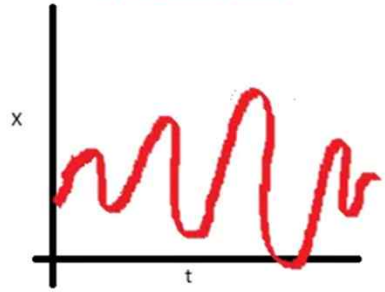


Non-Stationary series

시간의 추이에 상관없이 분산이 불변

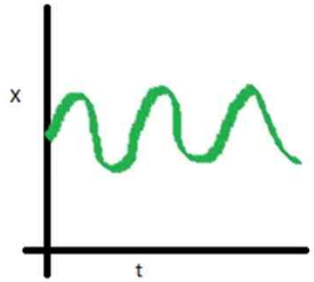


Stationary series

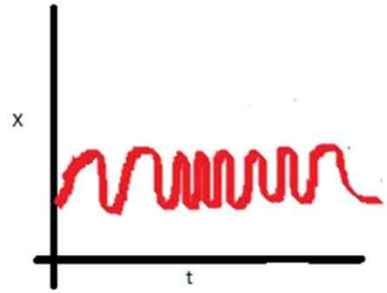


Non-Stationary series

두 시점 간의 공분산이 기준시점과 무관



Stationary series



Non-Stationary series

정상 시계열

- 정상시계열의 특징
  - 어떤 시점에서 평균과 분산, 그리고 특정한 시차의 길이를 갖는 자기공분산을 측정하더라고 동일한 값을 가짐
  - 항상 그 평균값으로 회귀하려는 경향이 있으며, 그 평균값 주변에서의 변동은 대체로 일정한 폭을 가짐
  - 정상시계열이 아닌 경우 특정기간의 시계열자료로부터 얻는 데이터를 다른 시기로 일반화 할 수 없음
- 정상시계열로 변환방법
  - 평균이 일정하지 않는 시계열은 차분(difference)를 통해 정상화
  - 차분은 현시점 자료에서 전 시점 자료를 빼는 것을 의미
  - 일반차분(regular difference): 바로 이전 시점의 자료를 빼는 것
  - 계절차분(seasonal difference): 여러 시점 전의 자료를 빼는 것. 주로 계절성을 갖는 자료를 정상화하는 데 사용
  - 분산이 일정하지 않는 시계열은 변환(transformation)으로 정상화
  - 변환은 로그변환, 제곱근 변환 등을 통해 데이터를 안정화함

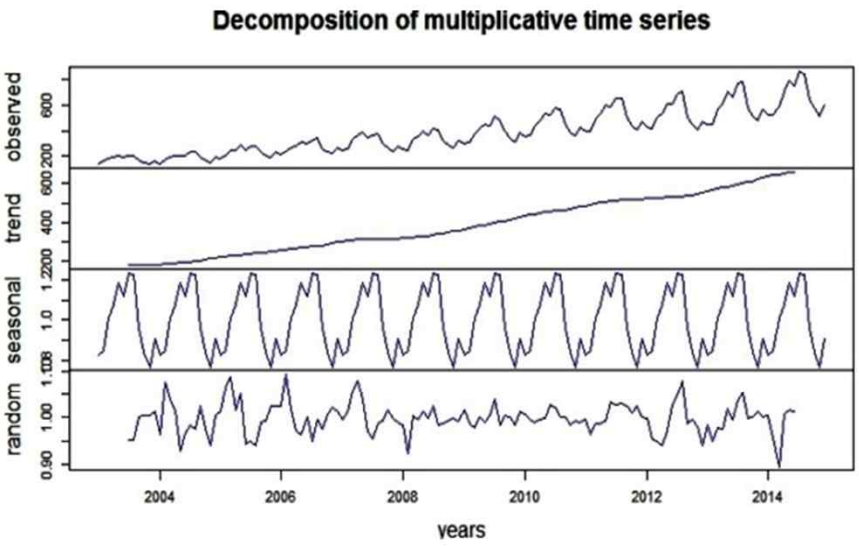
시계열자료 분석방법

- 회귀분석(계량경제)방법, 박스-젠킨스 방법, 지수평활법, 시계열 분해법 등이 있음
- 수학적 이론모형
  - 직관적 방법 : 지수평활법, 시계열 분해법, 시간에 따른 변동이 느린 데이터에 활용
  - 장기 예측: 회귀분석 활용
  - 단기 예측: 박스-젠킨스 방법, 지수평활법, 시계열 분해법 활용
  - 자기회귀 모형(AR모형, AutoRegressive model)
- 정상성을 가정하고 사용하는 모형
  - 자기회귀 모형(AR모형, AutoRegressive model)
  - 이동평균모형 (MA모형, Moving Average model)
- 비정상 시계열에 사용하는 모형
  - 자기회귀누적이동평균 모형 ARIMA(p,d,q) 모형 (AutoRegressive Interated Moving Average model)
  - ARIMA 모형은 비정상시계열 모형
  - ARIMA모형을 차분이나 변환을 통해 AR모형/MA모형/ARMA모형으로 정상화 가능

분해 시계열

분해 시계열

- 분해시계열이란
  - 시계열에 영향을 주는 일반적인 요인을 시계열에서 분리해 분석하는 방법을 의미함. 보통 4가지 요소로 구성됨



- 추세요인(Trend factor)
  - 시계열 자료에 영향을 주는 장기 변동요인. 큰그림
  - 시계열 데이터로 그림을 그렸을 때 그 형태가 오르거나 내리는 추세를 따르는 경우
  - 장기간에 걸쳐 지속적으로 증가/감소 또는 일정한 상태를 유지하려는 성향
- 계절요인(Seasonal factor)
  - 주로 주/월/분기/년 등 고정된 단위로 발생하는 단기변동요인
  - 요일마다 반복되거나 일년 중 각 월에 의한 변화, 사분기 자료에서 각 분기에 의한 변화 등 고정된 주기에 따라 자료가 변화하는 경우
  - 계절역 영향과 사회적관습에 따라 발생
- 순환요인(Cyclical factor)
  - 통상적으로 2~10년의 주기로 순환하는 이유모를 중기 변동요인
  - 경제적이나 자연적인 특별한 이유없이 알려지지 않은 주기를 가지고 변화하는 자료인 경우
  - 추세선을 따라 상하로 반복되는 변동으로 나타남
- 불규칙요인(Irregular factor)
  - 위 3가지 요인으로 설명할 수 없는 오차에 해당하는 요인
  - 규칙성 없이 예측불가능하게 우연적으로 발생하는 변동

시계열 데이터 이해

● 시계열적 변동의 종류

- **추세변동** (Trend Movement : T)
  장기변동의 전반적인 추세(경향), 추세선의 추정을 위해 최소자승법 사용
- **순환변동** (Cyclical Fluctuation : C)
  일정한 주기 없이 장기적 (1년 이상의 간격)으로 나타나는 유사한 진동
- **계절변동** (Seasonal Variation : S)
  1년 주기로 계절에 따라 되풀이되는 변동
- **불규칙/우연변동** (Irregular Movement / Random Variation : I)
  예측 불가능한 임의 변동으로 돌발적, 불명 원인에 의해 출현

- **가법모델** (Additive Model) Y (**수요량**) = T + C + S + I
- **승법모델** (Multiplicative Model) Y (**수요량**) = T x C x S x I

추세분석 방법

추세분석 (Trend Analysis)

시계열의 장기적 변동 경향(추세선)을 도출하여 미래의 수요를 예측하는 방법
→ 추세선의 형태를 선정하는 것이 가장 중요

● 추세분석법의 종류

① **목측법** (Freehand Method)
② **이동평균법** (Moving Average Method)
③ **지수평활법** (Exponential Smoothing Method)
④ **최소자승법** (Least-squares Method)

이동평균법(1/2)

[1] 이동평균법 (Moving Average Method)

과거 일정기간 동안의 실적을 평균해서 다음 기의 값을 예측하는 방법

– 특별한 추세변동, 계절변동, 순환변동 등의 요인이 없을 때 적용 가능

단순 이동평균법 (Simple Moving Average Method)

$$F_t = \frac{\sum A_{t-i}}{n}$$

F_t : **기간** t **의 수요 예측량**
A_{t-i} : **기간** (t-i) **의 실제 수요량**
n : **고려하는 기간 수**

가중 이동평균법 (Weighted Moving Average Method)

$$F_t = \frac{w_1 A_{t-1} + w_2 A_{t-2} + \cdots + w_n A_{t-n}}{w_1 + w_2 + \cdots + w_n}$$

F_t : **기간** t **의 수요 예측량**
A_{t-i} : **기간** (t-i) **의 실제 수요량**
w_i : **가중치**
[**단,** $w_1 > w_2 > \cdots > w_n$]

이동평균법(2/2)

다음은 공기정화기를 제조, 판매하는 (주)갑을의 지난 5개월 동안의 판매 자료이다.
6월의 수요를 예측하면 다음과 같다.

월	1	2	3	4	5
판매량	900	700	800	850	820

단순이동평균법의 적용 (4개월)

$$F_t = \frac{\sum A_{t-i}}{n} = \frac{700 + 800 + 850 + 820}{4} = 792.5$$

가중이동평균법의 적용 (4개월)
4개월간의 자료에 대한 가중치가 각각 0.4, 0.3, 0.2, 0.1

$$F_t = \frac{w_1 A_{t-1} + w_2 A_{t-2} + \cdots + w_n A_{t-n}}{w_1 + w_2 + \cdots + w_n}$$

$$= \frac{(0.4 \times 820)+(0.3 \times 850)+(0.2 \times 800)+(0.1 \times 700)}{0.4+0.3+0.2+0.1} = 813$$

지수평활법

[2] 지수평활법 (Exponential Smoothing Method)

- 가중이동평균법의 일종, 단기예측에 적합
- 가장 최근의 실적치에 가장 큰 가중치를 부여하고 오래된 데이터의 가중치는 지수함수적으로 적게 적용하는 것
- 장기적인 상승/하강 움직임(추세변동)을 포함하고 있는 자료에는 적용 부적합

$$F_t = F_{t-1} + \alpha(A_{t-1} - F_{t-1}) = \alpha A_{t-1} + (1 - \alpha) F_{t-1}$$

[단, α : 평활계수 ($0 \le \alpha \le 1$)]

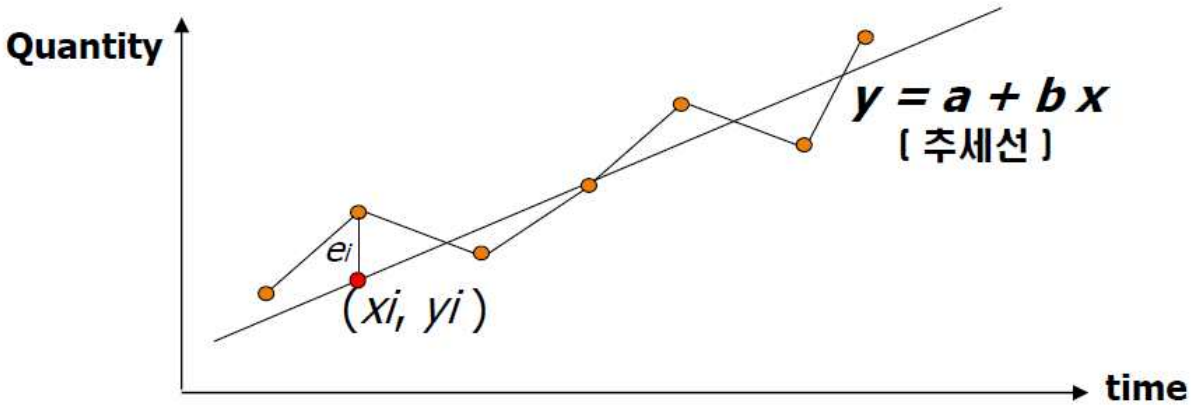
< 평활계수의 조정 >

- 제품수요가 불안정한 경우 (신제품의 생산판매, 고객취향의 변동심화품목)
  α : 0.7 – 0.8 ⇨ 예측의 감응도를 높인다
- 제품수요가 안정된 경우
  α : 0.01 – 0.5 ⇨ 예측의 안정도를 높인다

최소자승법(1/2)

[3] 최소자승법 (Least-squares Method)

- 실제치와 예측치와의 편차 [e_i] 자승의 총합이 최소가 되는 추세선을 찾고 이를 통해 미래 수요를 예측하는 방법
- 장래의 수요변동을 추세변동만으로 예측하려는 방법



$$b = \frac{n\Sigma x_i y_i - \Sigma x_i \Sigma y_i}{n\Sigma x_i^2 - (\Sigma x_i)^2} = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\Sigma(x_i - \bar{x})^2} \qquad \frac{Sxy}{Sxx}$$

$$a = \frac{\Sigma y_i - b\Sigma x_i}{n} = \bar{y} - b\bar{x}$$

최소자승법(2/2)

식 (1)과 식 (2)를 다시 정리하면

$$\Sigma y_i = an + b\,\Sigma x_i \quad \text{----------} \quad \text{식 (1)}'$$

$$\Sigma x_i y_i = a\Sigma x_i + b\Sigma^2 x_i \quad \text{---------} \quad \text{식 (2)}'$$

이상의 식을 연립하여 풀면

$$b = \frac{n\,\Sigma x_i y_i - \Sigma x_i \Sigma y_i}{n\,\Sigma x_i^2 - (\Sigma x_i)^2} \leftarrow \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\boxed{\Sigma(x_i - \bar{x})^2}}$$

$$a = \frac{\Sigma y_i - b\,\Sigma x_i}{n} = \bar{y} - b\bar{x}$$

$$\sum x_i^2 - 2\frac{(\sum x_i)^2}{n} + n\left(\frac{\sum x_i}{n}\right)^2$$
$$= \sum x_i^2 - \frac{(\sum x_i)^2}{n} = n\sum x_i^2 - (\sum x_i)^2$$

3

시계열 예측분석

시계열 분석 예제

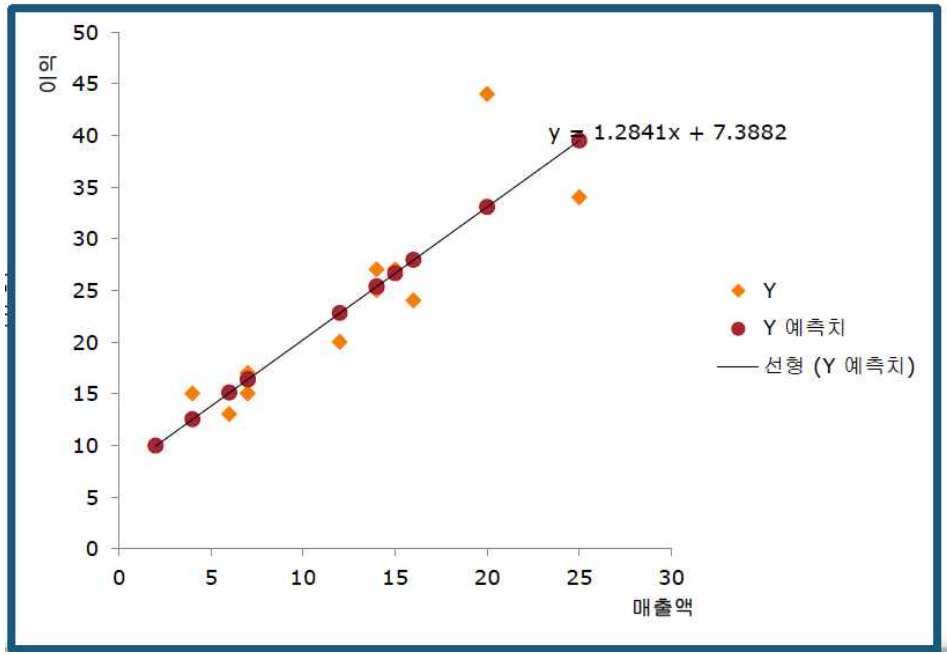
1월 실제수요 200
2월 실제수요 160
3월 실제수요 300
4월 실제수요 230

3개월 단순이동평균법: 5월 예측수요= (160+300+230)/3=230
가중이동평균법: 5월 예측수요=160*0.1+300*0.2+230*0.7
지수평할법: 5월 예측수요=
4월예측수요+0.1(4월 실제수요-4월 예측수요)

사례분석 B (1/3)

(주)천안전자의 PDP-TV의 매출액과 이익의 관계가 다음과 같았다.
이 자료를 토대로 매출액과 이익 사이에 적용할 수 있는 추세선을 구하라.

n	1	2	3	4	5	6	7	8	9	10	11	12
매출액(백만원) x	7	2	6	4	14	15	16	12	14	20	25	7
이 익(만원) y	15	10	13	15	25	27	24	20	27	44	34	17



$+ b x$

사례분석 B (2/3)

n	x_i	y_i	$x_i y_i$	x_i^2
1	7	15	105	49
2	2	10	20	4
3	6	13	78	36
4	4	15	60	16
5	14	25	350	196
6	15	27	405	225
7	16	24	384	256
8	12	20	240	144
9	14	27	378	196
10	20	44	880	400
11	15	34	510	225
12	7	17	119	49
합 계	132	271	3529	1796

$$b = \frac{n \Sigma x_i y_i - \Sigma x_i \Sigma y_i}{n \Sigma x_i^2 - (\Sigma x_i)^2} = \frac{(12 \times 3529) - (132 \times 271)}{(12 \times 1796) - (132 \times 132)} = 1.593$$

$$a = \frac{\Sigma y_i - b \Sigma x_i}{n} = \frac{271 - (1.593 \times 132)}{12} = 5.060$$

$$\therefore \; y = a + b x = 5.060 + 1.593 x$$

사례분석 B (3/3)

☑ 자료가 홀수개인 경우 "간편법" 에 의한 접근

년도	결과
1996	436
1997	470
1998	519
1999	578
2000	639

년도	xi	yi	xi^2	xi yi
1996	-2	436	4	-872
1997	-1	470	1	-470
1998	0	519	0	0
1999	1	578	1	578
2000	2	639	4	1278
합계	0	2642	10	514

$$\sum_{i=1}^{n} x_i = 0$$

$$\sum_{i=1}^{n} y_i - na - b\sum_{i=1}^{n} x_i = 0 \longrightarrow \therefore a = \frac{\sum_{i=1}^{n} y_i}{n} = \bar{y} = 528.4$$

$$\sum_{i=1}^{n} x_i y_i - a\sum_{i=1}^{n} x_i - b\sum_{i=1}^{n} x_i^2 = 0 \longrightarrow \therefore b = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2} = 51.4$$

$$\therefore y = 528.4 + 51.4x$$

수요예측 성능평가(1/3)

[1] 수요예측의 정확성 판정

- 예측오차 (E) = 실제수요(At)와 예측수요(Ft)와의 차이
- 오차가 크면 클수록 그때 채용된 예측기법의 유효성이 떨어짐을 의미

예측오차의 편의

- **누적예측오차** (RSFE : Running Sum of Forecast Error)

$$\mathbf{RSFE} = \Sigma(At - Ft)$$

- **평균예측오차** (MFE : Mean Forecast Error)

$$\mathbf{MFE} = \frac{\Sigma(At - Ft)}{n}$$

예측오차의 크기

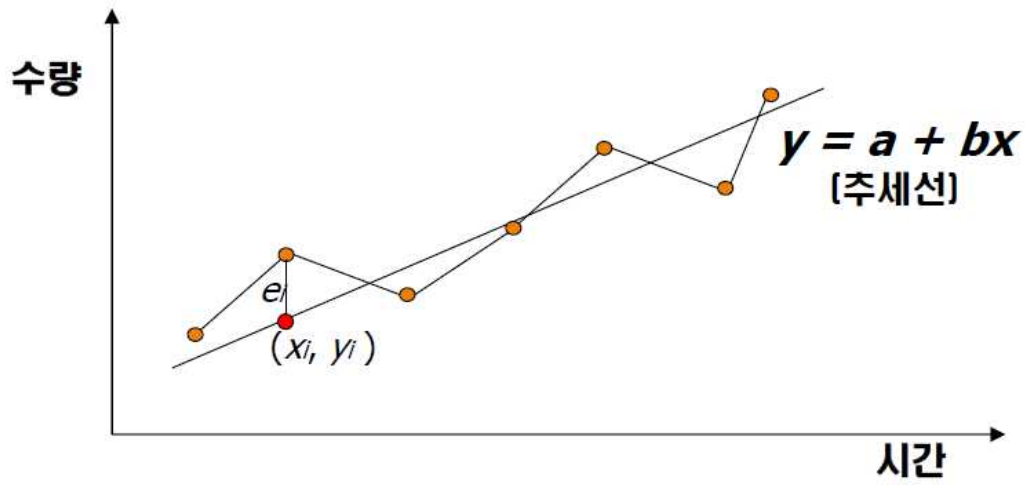
- **절대평균오차** (MAD : Mean Absolute Deviation)

$$\mathbf{MAD} = \frac{\Sigma \mid At - Ft \mid}{n}$$

- **평균절대백분률오차** (MAPE : Mean Absolute Percent Error)

$$\mathbf{MAPE} = \frac{1}{n}\left\{\Sigma \frac{\mid At - Ft \mid}{At}\right\} X\ 100\ (\%)$$

수요예측 성능평가(2/3)



$$y_i = a + bx_i + e_i$$

$$e_i = y_i - a - bx_i$$

$$\Sigma e_i^2 = \Sigma(y_i - a - bx_i)^2$$

$$\frac{\partial \Sigma e_i^2}{\partial a} = \frac{\partial}{\partial a}\Sigma(y_i - a - bx_i)^2 = 0 \quad \Rightarrow \quad \Sigma y_i - an - b\Sigma x_i = 0 \text{ ---------- 식 (1)}$$

$$\frac{\partial \Sigma e_i^2}{\partial b} = \frac{\partial}{\partial b}\Sigma(y_i - a - bx_i)^2 = 0 \quad \Rightarrow \quad \Sigma x_i y_i - a\Sigma x_i - b\Sigma x_i^2 = 0 \text{ --- 식 (2)}$$

수요예측 성능평가(3/3)

[2] 추적지표를 이용한 예측과정 관리

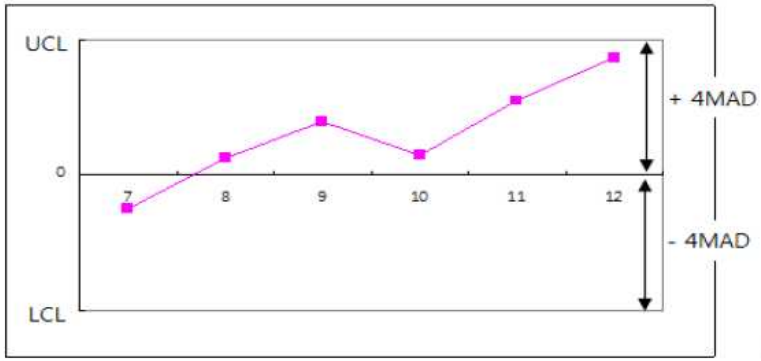
- **추적지표** : Tracking Signal
- **예측치의 평균이 일정한 진로를 유지하고 있는지를 나타내는 척도**
- **누적예측오차를** MAD**로 나눈 것**

$$TS = \frac{\Sigma\,(At - Ft)}{MAD} = \frac{RSFE}{MAD} = \frac{(n \times Bias)}{MAD}$$

※ **추적지표의 값이 「0」에서 크게 이탈한다면 편기가 발생하고 있음을 의미.**

- **양의 값 : 과소예측의 편기 존재**
- **음의 값 : 과대예측의 편기 존재**

※ **추적지표는 예측의 정확도가 의심스러울 경우에 사용**



$\pm 4MAD$

예측오차와 계절변동분석

- 예측오차: 예측오차는 예측치와 실제치와의 차이를 의미한다. 어느 일정 기간 동안 예측치와 실제치 자료를 가지고 (1) 평균절대오차(MAD, mean absolute deviation)와 (2) 누적예측오차(RSFE)를 비교하여 추적지표(tracing signal)를 계산한다. 추적지표는 어떤 시계열 분석방법이 실제수요의 상향과 하향의 변화를 얼마나 잘 반영하고 있는지를 보여준다.
- 계절변동분석: 계절변동의 형태에는 두 가지 (1) 가법계절변동(additive seasonal variation)과 (2) 승법계절변동(multiplicative seasonal variation)이 있다. (1) 가법계절변동에서는 "추세 + 계절" 공식을 이용하여 수요를 예측한다. (2) 승법계절변동에서는 "추세 × 계절" 공식을 이용하여 수요를 예측한다.

4

수요예측 분석

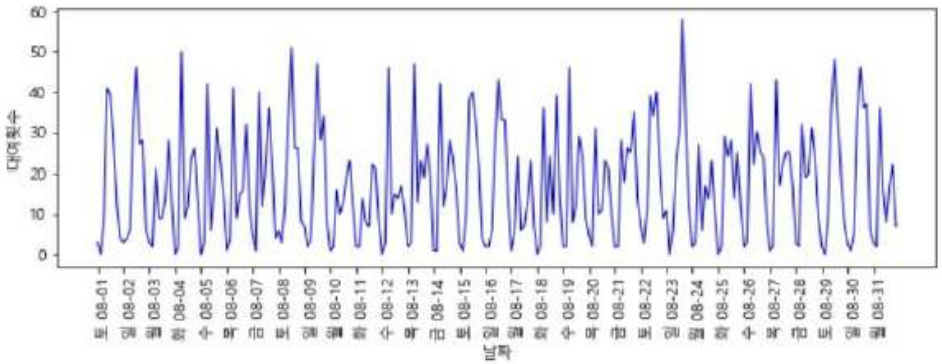
수요예측 지식활용

- **전문가의 지식이 유용한 특징 선택 시 도움이 됨**
  - 여행을 위한 항공료 예측
  - 날짜, 항공사, 출발지, 도착지
  - 공휴일, 성수기, 방학기간에 대한 정보
- A가 집 앞 대여소 B에서 자전거 대여 가능?
  - **특정 시간과 날짜에 B 대여소에서 자전거 대여 횟수 예측**
  - **남아있는 자전거가 있는가?**

tripduratio	starttime	stoptime	start station id	start station name	start station latitu	start station longitu	end statio	end statio	end statio	end statio	bikeid	usertype	birth year	gender
3117	8/1/2015 01:19:15	8/1/2015 02:11:12	301	E 2 St & Avenue B	40.72217444	-73.98368779	301	E 2 St & A	40.72217	-73.9837	18070	Subscriber	1986	1
690	8/1/2015 01:27:30	8/1/2015 01:39:00	301	E 2 St & Avenue B	40.72217444	-73.98368779	349	Rivington	40.7185	-73.9833	19699	Subscriber	1985	1
727	8/1/2015 01:38:49	8/1/2015 01:50:57	301	E 2 St & Avenue B	40.72217444	-73.98368779	2010	Grand St &	40.72165	-74.0023	20953	Subscriber	1982	1
698	8/1/2015 06:06:41	8/1/2015 06:18:20	301	E 2 St & Avenue B	40.72217444	-73.98368779	527	E 33 St &	40.74402	-73.9761	23566	Subscriber	1976	1
351	8/1/2015 06:24:29	8/1/2015 06:30:21	301	E 2 St & Avenue B	40.72217444	-73.98368779	250	Lafayette S	40.72456	-73.9957	17545	Subscriber	1959	1
597	8/1/2015 07:46:09	8/1/2015 07:56:06	301	E 2 St & Avenue B	40.72217444	-73.98368779	497	E 17 St &	40.73705	-73.9901	17435	Subscriber	1979	1
1248	8/1/2015 08:02:58	8/1/2015 08:23:47	301	E 2 St & Avenue B	40.72217444	-73.98368779	505	6 Ave & W	40.74901	-73.9885	18236	Subscriber	1987	1

Kagge Citi bike 수요예측 경진대회

- 2015년 8월 데이터
- Citibike = mglearn.datasets.load_citibike() #248 샘플
- Print(" 시티 바이크 데이터:\n{} " .format(citibike.head()))
  - 시티 바이크 데이터: starttime #3시간 단위로 대여 횟수 계산
  - Index , 횟수
  - 2015-08-01 00:00:00 3
  - 2015-08-01 03:00:00 0
  - 2015-08-01 06:00:00 9
  - 2015-08-01 09:00:00 41
  - 2015-08-01 12:00:00 39
  - Freq: 3H, Name: one, dtype: int64
  - 8월 데이터 중, 주말이 높고, 월요일, 목요일 등이 횟수가 줄어드는 패턴

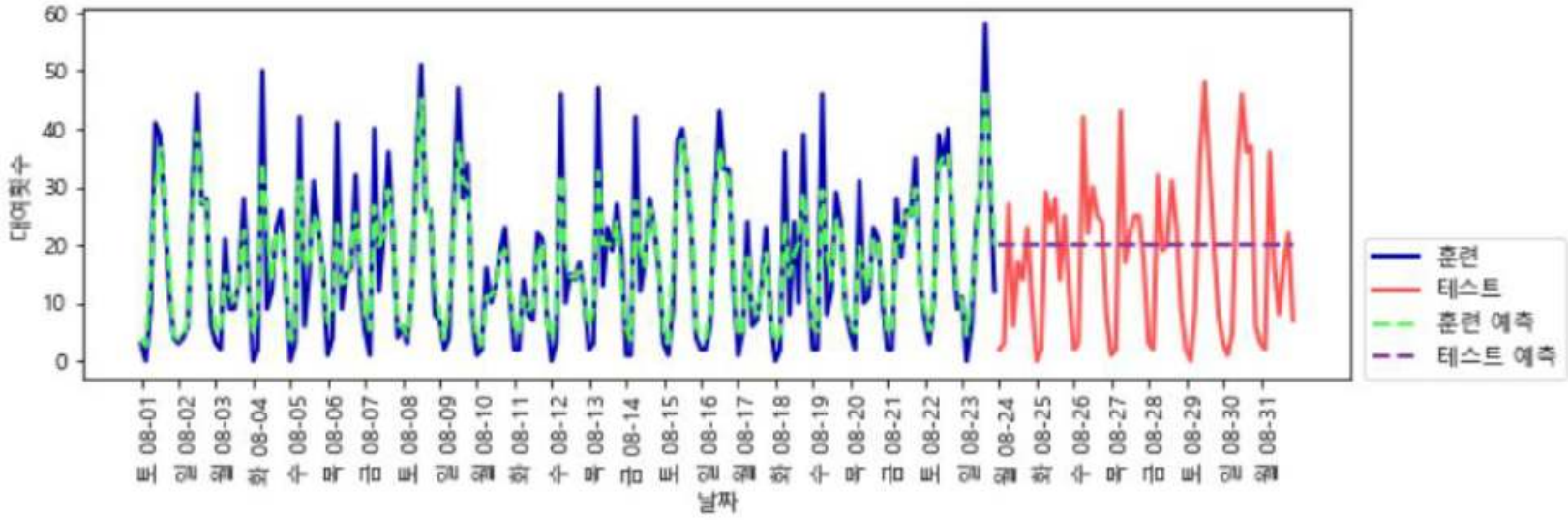


Citi bike data 훈련

- 시계열 데이터
  - 이전 날짜의 데이터를 훈련으로 사용하여 다음날 값을 예측
- y = citibike.values
  - # 타깃값 추출 (대여 횟수)
- X = citibike.index.astype("int64").values.reshape(-1, 1)
  - # astype(x)=>값을 x라는 타입으로 변경
  - 인덱스 시간을 64비트 정수로 변환 후 훈련데이터로 사용
  - [[1438387200] [1438398000] [1438408800] [1438419600] [1438430400] [1438441200] [1438452000] [1438462800] [1438473600] [1438484400] [1438495200]
  - 훈련 : 184(23일) 사용, 테스트 64(8일) 사용

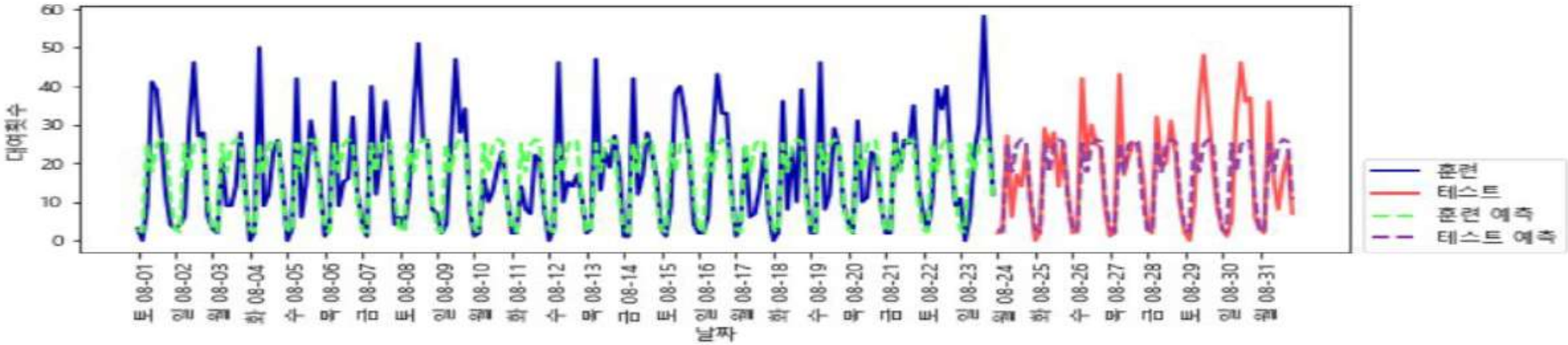
회귀비교

- regressor = RandomForestRegressor(n_estimators=100, random_state=0)
  - 테스트 세트 R^2: -0.04 #거의 학습되지 않음
  - Random forest: 외삽(extrapolation)이 안됨
  - 마지막 훈련세트 데이터의 타깃값을 예측으로
  - 테스트 데이터의 특성값이 훈련데이터의 특성값의 범위 밖



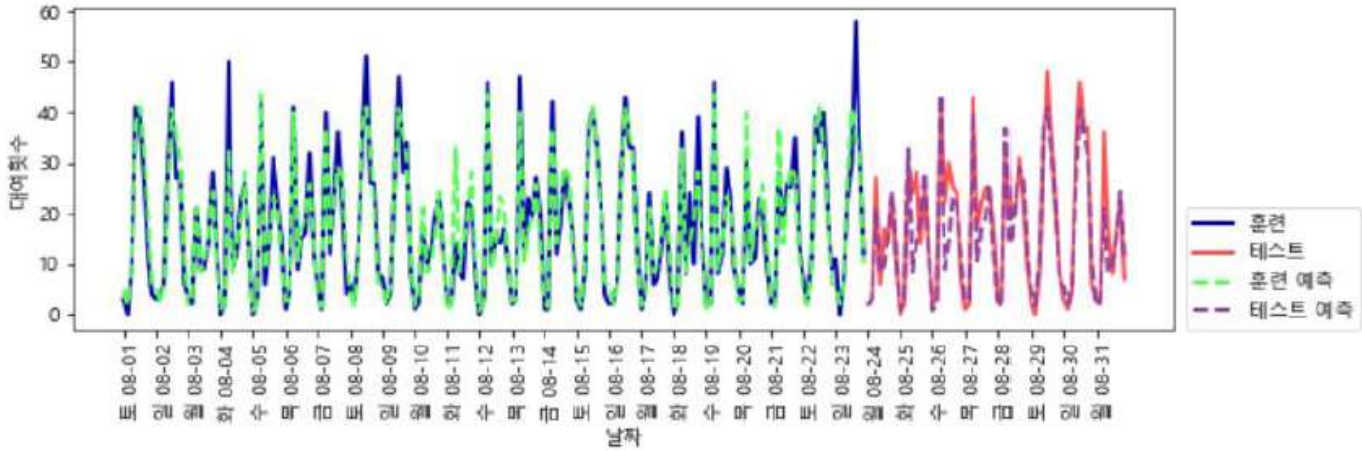
시간으로 회귀예측

- RandomForestRegressor
- X_hour = citibike.index.hour.values.reshape(-1, 1)
- eval_on_features(X_hour, y, regressor)
- print(X_hour)
  - [0] [3] [6] [9] [12] [15] [18] [21]
  - [0] [3] [6] [9] [12] [15] [18] [21]
  - 테스트 세트 R^2: 0.60 #훈련데이터의 주기적 시간으로 매일 같은 패턴



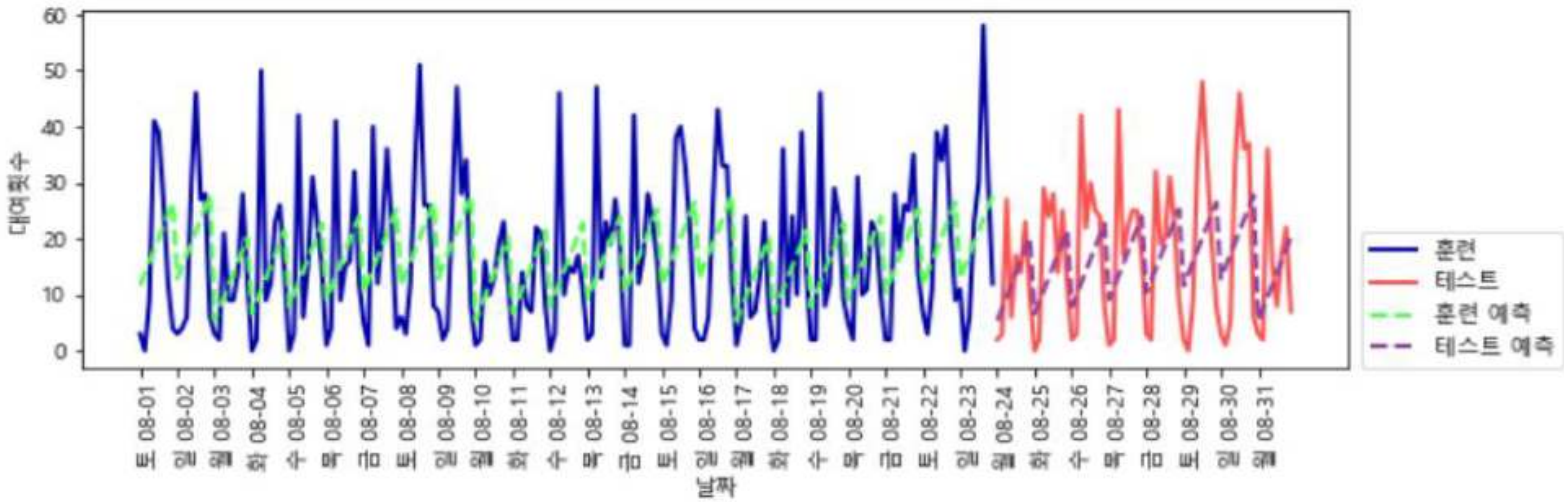
시간·요일 특성으로 예측

- RandomForestRegressor
- X_hour_week = np.hstack([citibike.index.dayofweek.values.reshape(-1, 1),
- citibike.index.hour.values.reshape(-1, 1)])
- eval_on_features(X_hour_week, y, regressor)
  - [5 0] [5 3] [5 6] [5 9] [5 12] [5 15] [5 18] [5 21]
  - [6 0] [6 3] [6 6] [6 9] [6 12] [6 15] [6 18] [6 21]
  - 테스트 세트 R^2: 0.84 #요일과 시간의 주기성을 찾아냄
  - 24일 월, 31일 월 예측이 동일



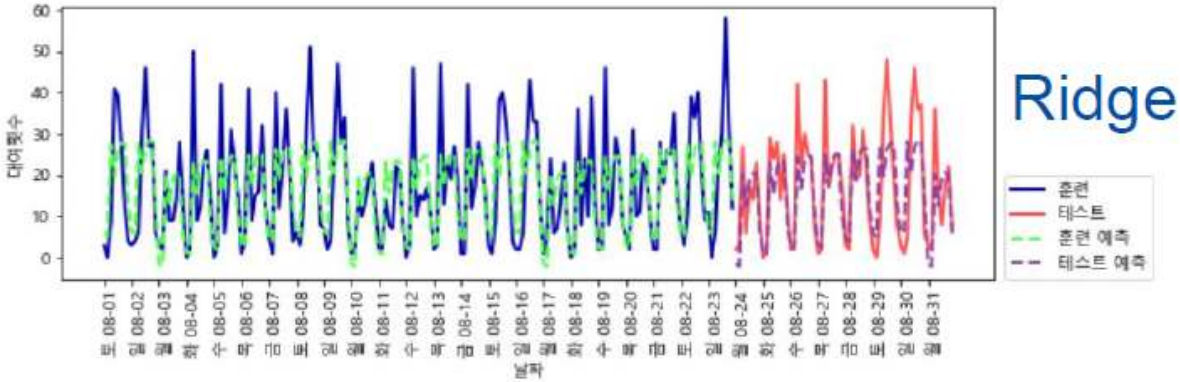
Linear regression

- eval_on_features(X_hour_week, y, LinearRegression())
  - 시간 요일 특성
  - 테스트 세트 R^2: 0.13
  - 성능이 떨어지고, 주기 패턴도 이상
  - 요일, 시간이 정수이므로 연속 값으로 해석
  - 하루의 시간이 증가할수록 대여수가 증가하는 선형 증가로 학습



OneHotEncoding

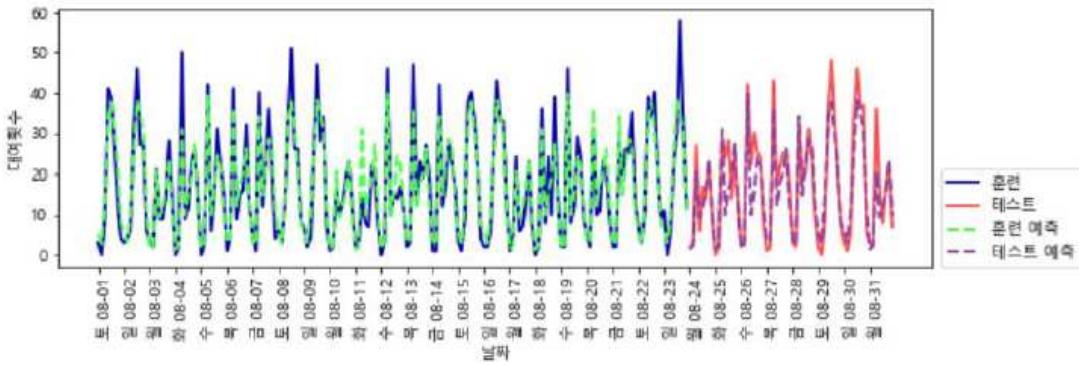
- enc = OneHotEncoder()
- X_hour_week_onehot = enc.fit_transform(X_hour_week).toarray()
  - [0. 0. 0. 0. 0. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0.] # 7개 요일, 8개 시간구간
  - [0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0. 0.] # 요일마다 시간마다 하나의 계수를 학습
  - [0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0.] # 시간패턴이 모든 날에 공유
  - [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
  - [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
  - [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
  - [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
- Ridge/linear
  - R^2: 0.62
- RF: 0.84



상호작용특징

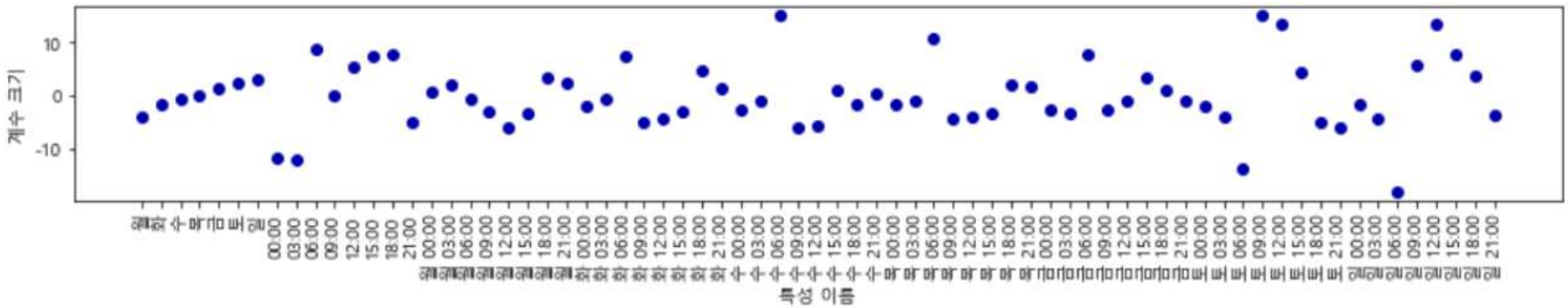
- poly_transformer = PolynomialFeatures(degree=2, interaction_only=True,
- include_bias=False) # (15,2)조합 + 15 = 105+15=120
- X_hour_week_onehot_poly = poly_transformer.fit_transform(X_hour_week_onehot)
- lr = Ridge()
- eval_on_features(X_hour_week_onehot_poly, y, lr)
- Ridge :시간과 요일 조합별 계수 학습
  - R^2: 0.85
- RF: 0.84
- LR: 0.4

Ridge



Ridge coef_

- hour = ["%02d:00" % i for i in range(0, 24, 3)]
- day = ["월", "화", "수", "목", "금", "토", "일"]
- features = day + hour
  - '월', '화', '수', '목', '금', '토', '일', '00:00', '03:00', '06:00', '09:00', '12:00', '15:00', '18:00', '21:00']
- features_poly =



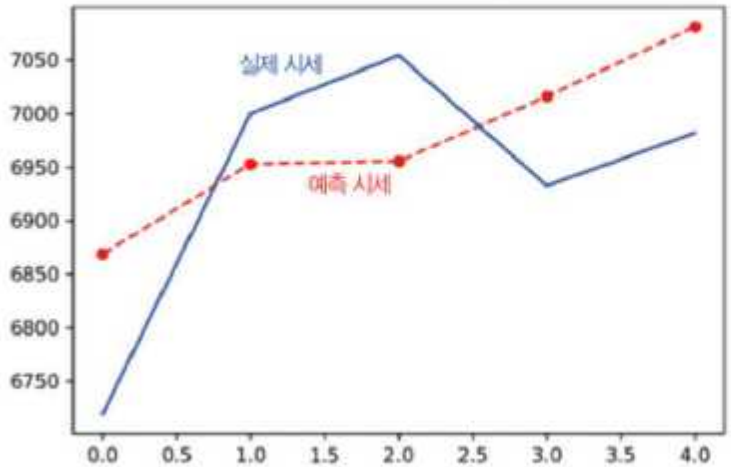
5

시세예측 분석

예측 미리보기

재미로 알아보는 내일의 비트코인 시세

아래 그래프는 1년간의 비트코인 데이터를 기반으로 향후 5일간 비트코인 데이터의 시세 변화를 미리 예측한 것입니다.



분석 결과: 5일간의 비트코인 시세 예측 그래프

주요 키워드	핵심 내용	설명
시계열 데이터 분석	시계열 데이터 분석의 정의	연도, 월, 일자 같은 시간성을 띠는 데이터를 학습하여 미래의 수치를 예측하는 분석입니다.
ARIMA 모델링	시계열 예측 분석 방법	ARIMA 모델은 '얼마만큼의 나를 돌아보며', '얼마만큼의 차분을 이용하여 규칙성을 줄 것이며', '얼마만큼의 오차를 관찰할 것인지'를 활용한 시계열 예측 분석의 방법입니다.
모델의 평가	학습 데이터셋과 테스트 데이터셋의 분리	특정한 시간 혹은 일자를 기준으로 학습 데이터셋과 테스트 데이터셋을 분리합니다.
모델 개선의 방법	상한선과 하한선 설정, 이상치 제거	더 나은 분석을 위해 상한선과 하한선을 설정하거나 이상치를 제거하는 방법을 사용할 수 있습니다.

시간정보 데이터 탐색(1/2)

>>> market price 데이터셋의 기본 정보 구하기

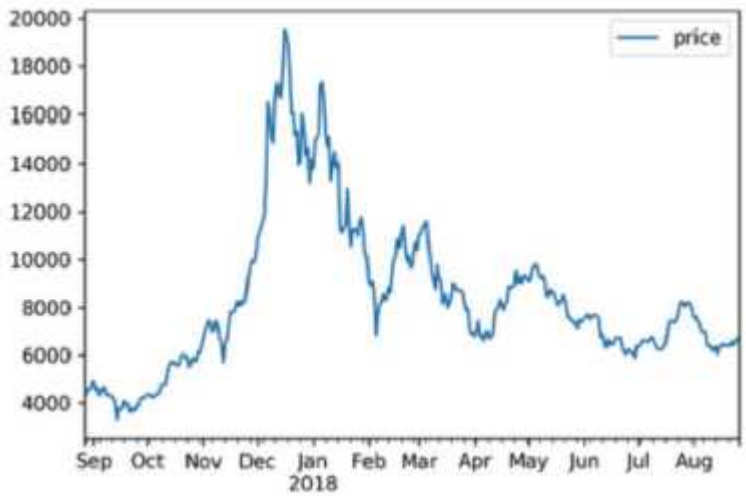
```
# -*- coding: utf-8 -*-
%matplotlib inline

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Data Source: https://www.blockchain.com/ko/charts/market-price?timespan=60days
file_path = '../data/market-price.csv'
bitcoin_df = pd.read_csv(file_path, names = ['day', 'price'])

# 기본 정보를 출력합니다.
print(bitcoin_df.shape)
print(bitcoin_df.info())
bitcoin_df.tail()
```

```
(365, 2)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 365 entries, 0 to 364
Data columns (total 2 columns):
day      365 non-null object
price    365 non-null float64
dtypes: float64(1), object(1)
memory usage: 5.8+ KB
None
```

	day	price
360	2018-08-22 00:00:00	6575.229167
361	2018-08-23 00:00:00	6434.881667
362	2018-08-24 00:00:00	6543.645714
363	2018-08-25 00:00:00	6719.429231
364	2018-08-26 00:00:00	6673.274167

시간정보 데이터 탐색(2/2)

```
# to_datetime으로 day 피처를 시계열 피처로 변환합니다.
bitcoin_df['day'] = pd.to_datetime(bitcoin_df['day'])

# day 데이터 프레임의 index로 설정합니다.
bitcoin_df.index = bitcoin_df['day']
bitcoin_df.set_index('day', inplace=True)

# 일자별 비트코인 시세를 시각화합니다.
bitcoin_df.plot()
plt.show()
```



파이썬 ARIMA 라이브러리 활용

>>> ARIMA 모델 활용하기: 모델 학습

```
from statsmodels.tsa.arima_model import ARIMA
import statsmodels.api as sm

# (AR=2, 차분=1, MA=2) 파라미터로 ARIMA 모델을 학습합니다.
model = ARIMA(bitcoin_df.price.values, order=(2,1,2))
model_fit = model.fit(trend='c', full_output=True, disp=True)
print(model_fit.summary())
```

ARIMA Model Results

Dep. Variable:	D.y	No. Observations:	364
Model:	ARIMA(2, 1, 2)	Log Likelihood	-2780.074
Method:	css-mle	S.D. of innovations	501.536
Date:	Thu, 04 Jul 2019	AIC	5572.148
Time:	21:41:22	BIC	5595.531
Sample:	1	HQIC	5581.442

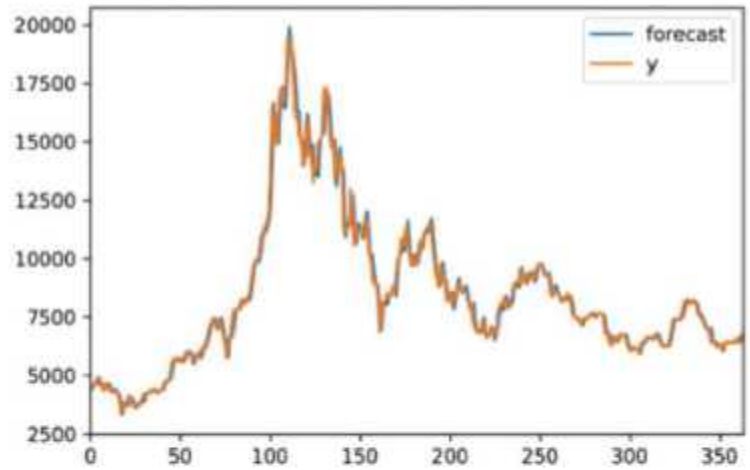
	coef	std err	z	P>|z|	[0.025	0.975]
const	6.3813	26.944	0.237	0.813	-46.428	59.191
ar.L1.D.y	0.4496	0.029	15.512	0.000	0.393	0.506
ar.L2.D.y	-0.9622	0.023	-42.313	0.000	-1.007	-0.918
ma.L1.D.y	-0.3889	0.031	-12.697	0.000	-0.449	-0.329
ma.L2.D.y	0.9393	0.039	23.874	0.000	0.862	1.016

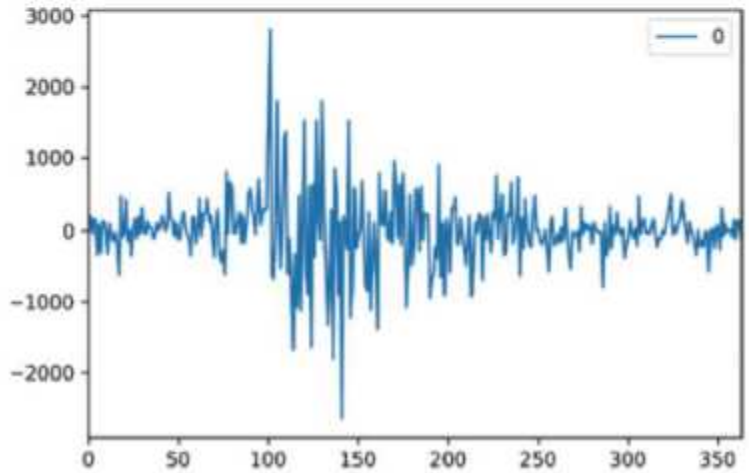
Roots

	Real	Imaginary	Modulus	Frequency
AR.1	0.2336	-0.9923j	1.0194	-0.2132
AR.2	0.2336	+0.9923j	1.0194	0.2132
MA.1	0.2070	-1.0108j	1.0318	-0.2178
MA.2	0.2070	+1.0108j	1.0318	0.2178

ARIMA모델 성능·예측

```
>>> ARIMA 모델 활용하기: 모델의 성능 & 예측 결과 시각화

fig = model_fit.plot_predict() # 학습 데이터에 대한 예측 결과입니다. (첫 번째 그래프)
residuals = pd.DataFrame(model_fit.resid) # 잔차의 변동을 시각화합니다. (두 번째 그래프)
residuals.plot()
```



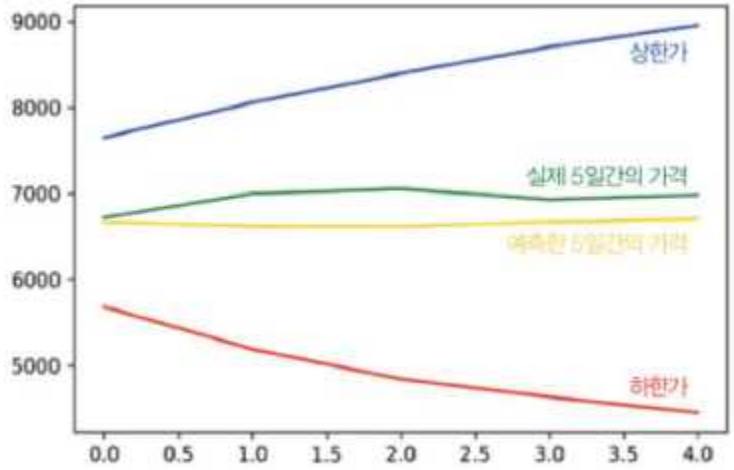


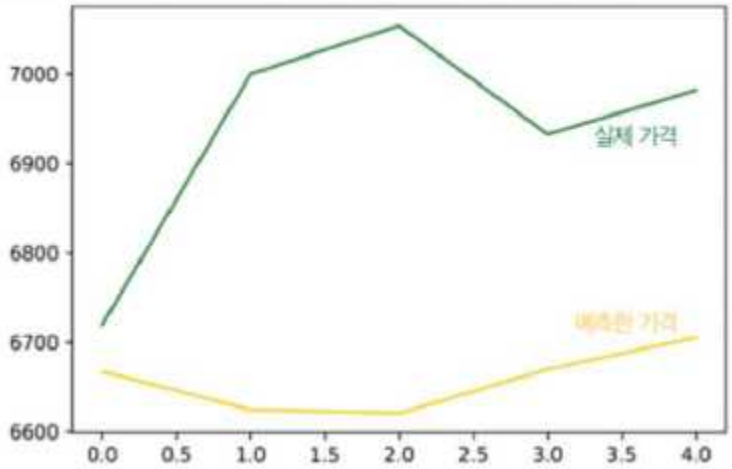
ARIMA모델 활용

>>> ARIMA 모델 활용하기: 실제 데이터와의 비교

```
forecast_data = model_fit.forecast(steps=5) # 학습 데이터셋으로부터 5일 뒤를 예측합니다.

# 테스트 데이터셋을 불러옵니다.
test_file_path = '../data/market-price-test.csv'
bitcoin_test_df = pd.read_csv(test_file_path, names=['ds', 'y'])

pred_y = forecast_data[0].tolist() # 마지막 5일의 예측 데이터입니다. (2018-08-27 - 2018-08-31)
test_y = bitcoin_test_df.y.values # 실제 5일 가격 데이터입니다. (2018-08-27 - 2018-08-31)
pred_y_lower = [] # 마지막 5일의 예측 데이터의 최소값입니다.
pred_y_upper = [] # 마지막 5일의 예측 데이터의 최대값입니다.
for lower_upper in forecast_data[2]:
    lower = lower_upper[0]
    upper = lower_upper[1]
    pred_y_lower.append(lower)
    pred_y_upper.append(upper)
```

```
plt.plot(pred_y, color="gold") # 모델이 예측한 가격 그래프입니다.
plt.plot(pred_y_lower, color="red") # 모델이 예측한 최저 가격 그래프입니다.
plt.plot(pred_y_upper, color="blue") # 모델이 예측한 최고 가격 그래프입니다.
plt.plot(test_y, color="green") # 실제 가격 그래프입니다.
```





Facebook 라이브러리 활용

```
[Anaconda Powershell Prompt]

(pybook) C:\Users\yoonk> conda install -c conda-forge fbprophet
```

>>> Facebook Prophet 활용하기

```
from fbprophet import Prophet

# prophet을 사용하기 위해서는 다음과 같이 피처명을 변경해야 합니다: 'ds', 'y'
bitcoin_df = pd.read_csv(file_path, names=['ds', 'y'])
```

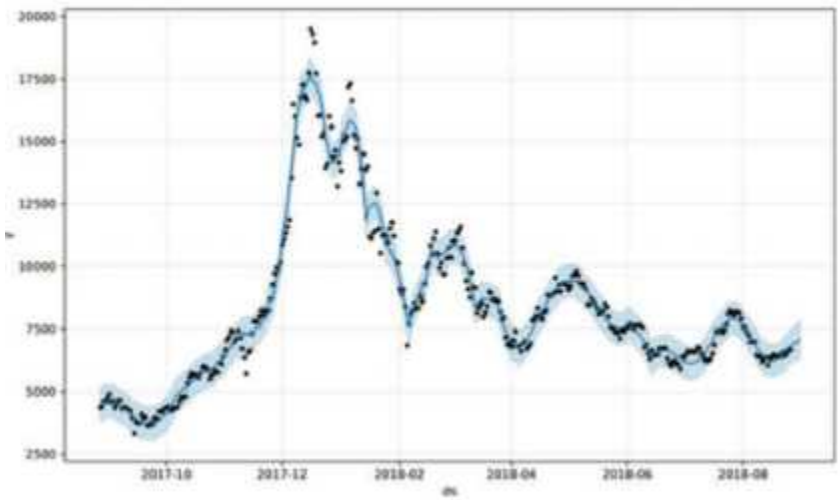
```
prophet = Prophet(seasonality_mode='multiplicative',
            yearly_seasonality=True,
            weekly_seasonality=True, daily_seasonality=True,
            changepoint_prior_scale=0.5)
prophet.fit(bitcoin_df)
```

```
# 5일을 내다보며 예측합니다.
future_data = prophet.make_future_dataframe(periods=5, freq='d')
forecast_data = prophet.predict(future_data)
forecast_data[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail(5)
```

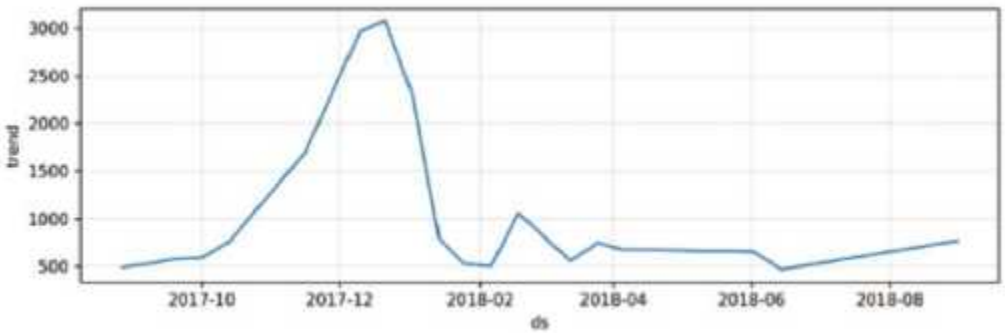
	ds	yhat	yhat_lower	yhat_upper
365	2018-08-27	6868.850910	6221.567267	7562.957505
366	2018-08-28	6953.100528	6316.073545	7612.836267
367	2018-08-29	6955.494928	6255.320153	7709.794007
368	2018-08-30	7015.995983	6250.178249	7805.851140
369	2018-08-31	7081.080511	6356.615182	7903.602542

Facebook 예측모델 결과(1/2)

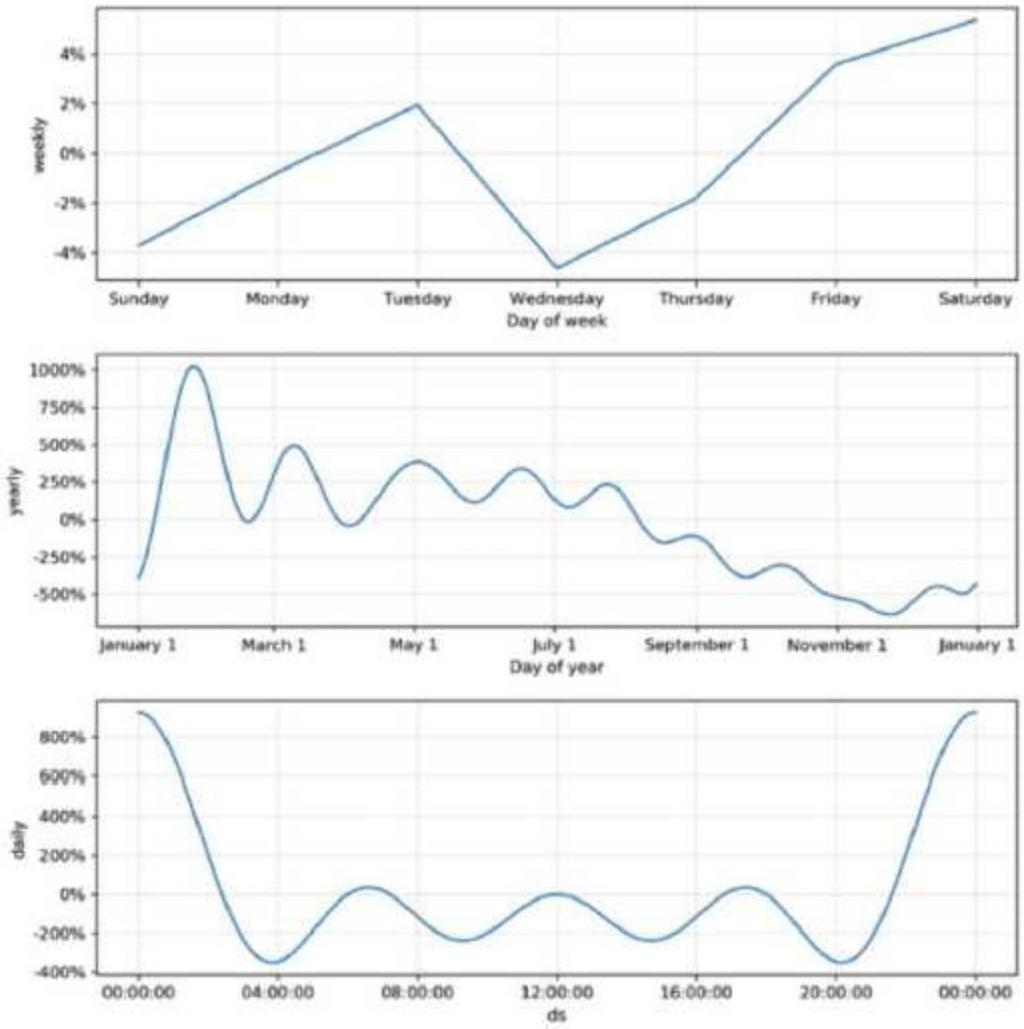
```
fig1 = prophet.plot(forecast_data)
```




```
fig2 = prophet.plot_components(forecast_data)
```



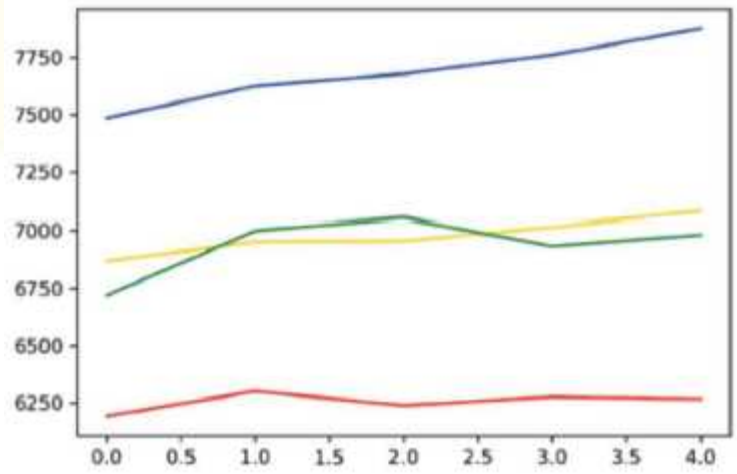
Facebook 예측모델 결과(2/2)



Facebook 예측모델 활용(1/2)

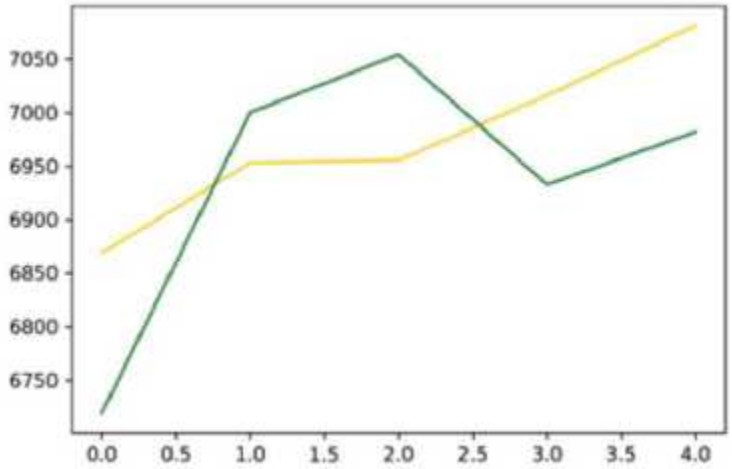
>>> Facebook Prophet 활용하기: 실제 데이터와의 비교

```
bitcoin_test_df = pd.read_csv(test_file_path, names=['ds', 'y'])

# 마지막 5일의 예측 데이터입니다. (2018-08-27 ~ 2018-08-31)
pred_y = forecast_data.yhat.values[-5:]

# 실제 5일 가격 데이터입니다. (2018-08-27 ~ 2018-08-31)
test_y = bitcoin_test_df.y.values
# 마지막 5일 예측 데이터의 최소값입니다.
pred_y_lower = forecast_data.yhat_lower.values[-5:]
# 마지막 5일의 예측 데이터의 최대값입니다.
pred_y_upper = forecast_data.yhat_upper.values[-5:]

plt.plot(pred_y, color="gold") # 모델이 예상한 가격 그래프입니다.
plt.plot(pred_y_lower, color="red") # 모델이 예상한 최저 가격 그래프입니다.
plt.plot(pred_y_upper, color="blue") # 모델이 예상한 최고 가격 그래프입니다.
plt.plot(test_y, color="green") # 실제 가격 그래프입니다.
```



Facebook 예측모델 활용(2/2)

```
plt.plot(pred_y, color="gold") # 모델이 예상한 가격 그래프입니다.
plt.plot(test_y, color="green") # 실제 가격 그래프입니다.
```



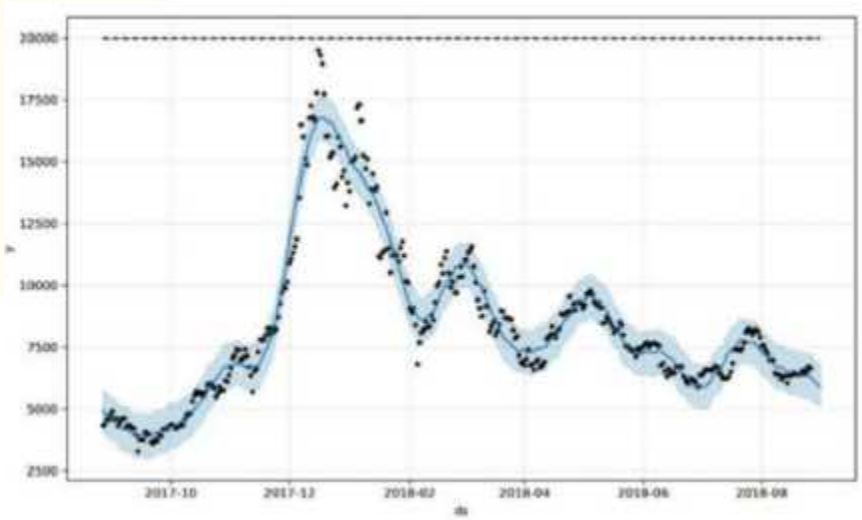

```
rmse = sqrt(mean_squared_error(pred_y, test_y))
print(rmse)
```

```
101.10270853692928
```

예측모델 성능향상(1/3)

>>> 상한가 및 하한가 설정하기

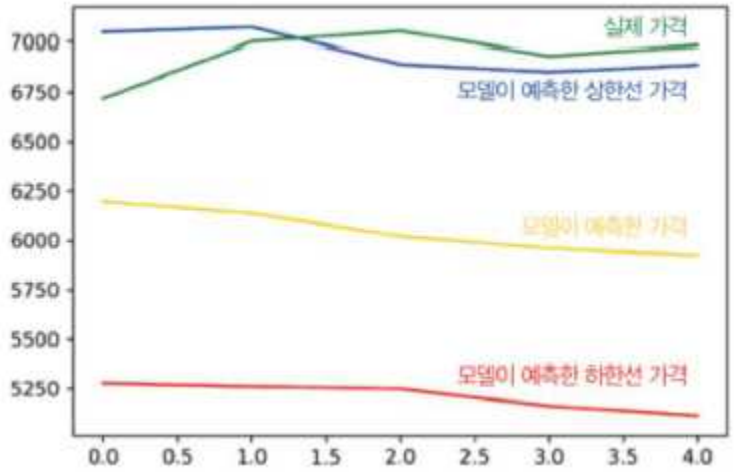
```
bitcoin_df = pd.read_csv(file_path, names=['ds', 'y'])

# 상한가를 설정합니다.
bitcoin_df['cap'] = 20000

# 상한가 적용을 위한 파라미터를 다음과 같이 설정합니다.
prophet = Prophet(seasonality_mode='multiplicative',
            growth='logistic',
            yearly_seasonality=True,
            weekly_seasonality=True, daily_seasonality=True,
            changepoint_prior_scale=0.5)
prophet.fit(bitcoin_df)

# 5일을 내다보며 예측합니다.
future_data = prophet.make_future_dataframe(periods=5, freq='d')

# 상한가를 설정합니다.
future_data['cap'] = 20000
forecast_data = prophet.predict(future_data)

fig = prophet.plot(forecast_data)
```



예측모델 성능향상(2/3)

>>> 예측과 실제 비교 그래프

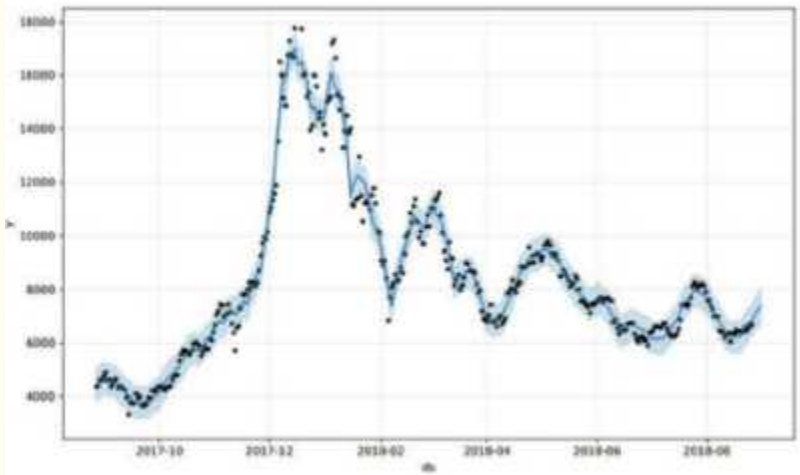
```
bitcoin_test_df = pd.read_csv(test_file_path, names=['ds', 'y'])

# 모델이 예측한 마지막 5일의 가격 데이터를 가져옵니다.
pred_y = forecast_data.yhat.values[-5:]
test_y = bitcoin_test_df.y.values
pred_y_lower = forecast_data.yhat_lower.values[-5:]
pred_y_upper = forecast_data.yhat_upper.values[-5:]

plt.plot(pred_y, color="gold") # 모델이 예측한 가격 그래프입니다.
plt.plot(pred_y_lower, color="red") # 모델이 예측한 하한가 그래프입니다.
plt.plot(pred_y_upper, color="blue") # 모델이 예측한 상한가 그래프입니다.
plt.plot(test_y, color="green") # 실제 가격 그래프입니다.
```



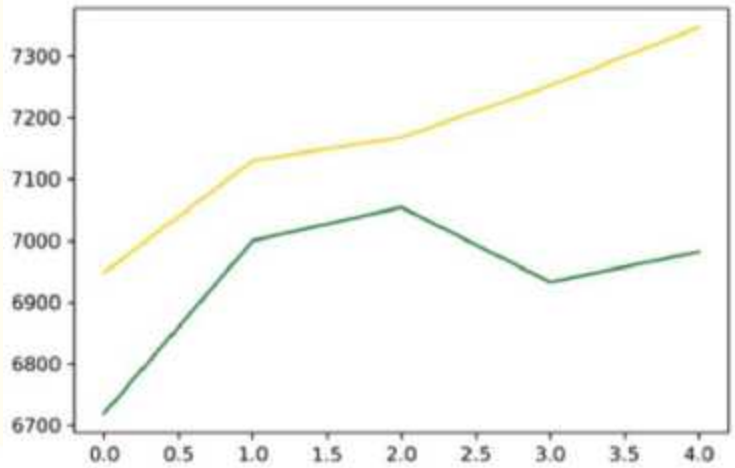
예측모델 성능향상(3/3)

>>> 이상치 제거하기

```
# 18,000 이상의 데이터는 이상치라고 판단합니다.
bitcoin_df = pd.read_csv(file_path, names=['ds', 'y'])
bitcoin_df.loc[bitcoin_df['y'] > 18000, 'y'] = None

# prophet 모델을 학습합니다.
prophet = Prophet(seasonality_mode='multiplicative',
            yearly_seasonality=True,
            weekly_seasonality=True, daily_seasonality=True,
            changepoint_prior_scale=0.5)
prophet.fit(bitcoin_df)

# 5일 단위의 미래를 예측합니다.
future_data = prophet.make_future_dataframe(periods=5, freq='d')
forecast_data = prophet.predict(future_data)

# 예측 결과를 그래프로 출력합니다.
fig = prophet.plot(forecast_data)
```




```
plt.plot(pred_y, color="gold") # 모델이 예측한 가격 그래프입니다.
plt.plot(test_y, color="green") # 실제 가격 그래프입니다.

# 테스트 데이터의 RMSE를 출력합니다.
rmse = sqrt(mean_squared_error(pred_y, test_y))
print(rmse)
```



Test RMSE : 252.17892671748956

6

판매량 예측 분석

판매량 데이터 예측

Shampoo Sales Dataset

3년간 월별 샴푸 판매량에 대한 데이터셋

The units are a sales count and there are 36 observations. The original dataset is credited to Makridakis, Wheelwright, and Hyndman (1998).

- Download the dataset.

아래의 예제는 로드된 데이터셋의 Plot을 로드하고 생성한다.

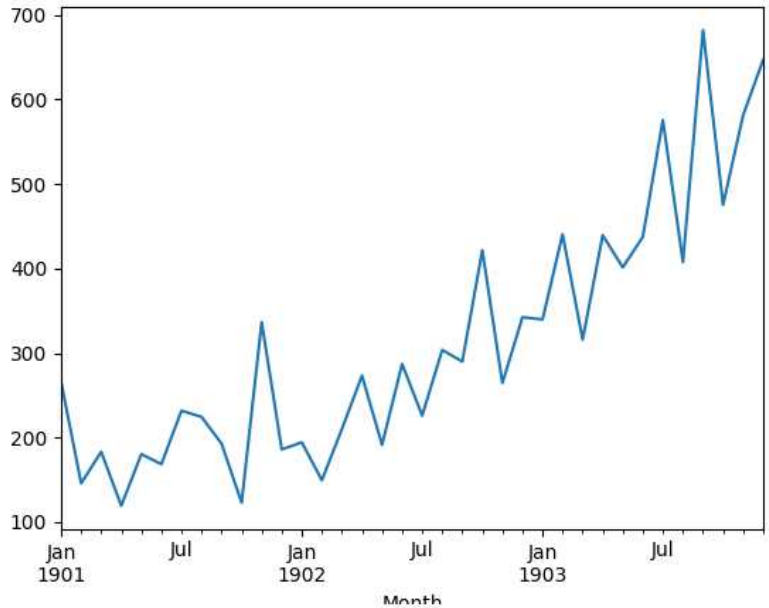
```
# load and plot dataset
from pandas import read_csv
from pandas import datetime
from matplotlib import pyplot
# load dataset
def parser(x):
        return datetime.strptime('190'+x, '%Y-%m')
series = read_csv('shampoo-sales.csv', header=0, parse_dates=[0], index_col=0, squeeze=True, date_parse
r=parser)
# summarize first few rows
print(series.head())
# line plot
series.plot()
pyplot.show()
```

데이터 탐색

예제의 실행 결과로 Pandas Series로써 데이터셋을 로드하고 처음 5행을 출력한다.

```
Month
1901-01-01    266.0
1901-02-01    145.9
1901-03-01    183.1
1901-04-01    119.3
1901-05-01    180.3
Name: Sales, dtype: float64
```

명확한 증가 추세를 보여주는 일련의 line plot이 생성된다.



Line Plot of Shampoo Sales Dataset

다음으로 실험에서 사용된 모델의 구성과 테스트 하네스를 살펴본다.

(test harness: 시스템 및 시스템 컴포넌트를 시험하는 환경의 일부분으로 시험을 지원하는 목적하에 생성된 코드와 데이터.)

데이터 준비

Data Preparation and Model Evaluation

이번 섹션은 데이터 준비와 모델 평가를 설명한다.

Data Split

샴푸 판매 데이터셋을 트레이닝셋과 테스트셋 2개로 나눈다.

처음 2년의 데이터는 트레이닝 데이터셋으로 취하고 남은 1년의 데이터는 테스트셋으로 취한다.

모델은 트레이닝 데이터셋을 사용하면서 발전될 것이고 테스트셋에 대한 예측을 할 것이다.

참고로 마지막 12달의 관측은 다음과 같다:

```
"3-01",339.7
"3-02",440.4
"3-03",315.9
"3-04",439.3
"3-05",401.3
"3-06",437.4
"3-07",575.5
"3-08",407.6
"3-09",682.0
"3-10",475.3
"3-11",581.3
"3-12",646.9
```

모델 평가

Multi-Step Forecast

우리는 multi-step 예측을 할 것이다.

데이터셋의 마지막 12개월 중 지정된 달에 대한 3개월 예측을 해야한다.

주어진 과거의 관측(t-1, t-2, ... t-n)으로 t, t+1, t+2를 예측한다.

특히, 2년차의 12월로부터는 1월, 2월, 3월을 예측해야 한다. 1월로부터는 2월과 3월, 4월을 예측한다. All the way to an October, November, December forecast from September in year 3.

다음과 같이 총 10개의 3개월 예측이 필요하다.

```
Dec,    Jan, Feb, Mar
Jan,    Feb, Mar, Apr
Feb,    Mar, Apr, May
Mar,    Apr, May, Jun
Apr,    May, Jun, Jul
May,    Jun, Jul, Aug
Jun,    Jul, Aug, Sep
Jul,    Aug, Sep, Oct
Aug,    Sep, Oct, Nov
Sep,    Oct, Nov, Dec
```

Model Evaluation

롤링-예측 시나리오가 사용될 것이다. (also called walk-forward model validation.)

테스트 데이터셋의 각각의 time step은 한 번에 하나씩 진행된다.

모델은 time step에 대한 예측에 사용되는데 테스트 세트로부터 실제로 다음달에 기대되는 값(예측값)은 그 다음의 time step에 예측을 위한 모델에 사용된다.

이것은 새로운 샴푸 판매 관측치들이 각각의 달에 사용될 수 있고 다음달의 예측에 사용되는 현실 세계의 시나리오를 모방한다. 이것은 훈련 및 시험 데이터 세트의 구조에 의해 시뮬레이션될 것이다.

테스트 데이터 집합에 대한 모든 예측을 수집하고 각 예측 시간 단계에 대한 모델의 기술을 요약하기 위해 오류 점수를 계산한다.

root mean squared error (RMSE)는 큰 오류를 잡는데 사용되고 예측 데이터와 같은 유닛에 있는 점수의 결과로 사용된다. 즉, 월간 샴푸 판매량

모델적용

Persistence Model

시계열 예측을 위한 좋은 Baseline은 지속성 모델이다.

이것은 마지막 관측이 앞으로도 유지되는 예측 모델이다.

이것은 단순성때문에 종종 navie 예측이라 불린다.

아래의 포스트에서 시계열 예측을 위한 **지속성** 모델을 더 공부할 수 있다:

- How to Make Baseline Predictions for Time Series Forecasting with Python

Prepare Data

첫 단계는 시계열 데이터를 지도된 학습 problem으로 바꾸는 것이다.

숫자들의 리스트에서 입출력 패턴의 리스트로 가는 것이다.

우리는 series_to_supervised()라고 불리는 pre-prepared function을 사용하여 이를 달성할 수 있다.

series_to_supervised() 함수에 대한 더 많은 정보가 담긴 포스트:

- How to Convert a Time Series to a Supervised Learning Problem in Python

데이터준비 함수

이 함수는 아래에 리스트되어 있다.

```
# convert time series into supervised learning problem
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
        n_vars = 1 if type(data) is list else data.shape[1]
        df = DataFrame(data)
        cols, names = list(), list()
        # input sequence (t-n, ... t-1)
        for i in range(n_in, 0, -1):
                cols.append(df.shift(i))
                names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
        # forecast sequence (t, t+1, ... t+n)
        for i in range(0, n_out):
                cols.append(df.shift(-i))
                if i == 0:
                        names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
                else:
                        names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
        # put it all together
        agg = concat(cols, axis=1)
        agg.columns = names
        # drop rows with NaN values
        if dropnan:
                agg.dropna(inplace=True)
        return agg
```

이 함수는 로드된 시계열 값 n_in value 1과 n_out value 3을 전달함으로써 호출할 수 있다.

For example:

```
supervised = series_to_supervised(raw_values, 1, 3)
```

데이터분할 함수

다음으로 우리는 지도 학습 데이터셋을 훈련 세트와 테스트 세트로 분할할 수 있다.

이 형태에서 마지막 10개의 행은 마지막 해의 데이터를 가지고 있음을 알고 있다. 이 행들은 테스트 세트를 구성하고 나머지 데이터는 훈련 데이터 세트를 구성한다.

우리는 로드된 시계열과 몇몇의 파라미터들을 취하고 모델링 준비를 위한 훈련 및 테스트 세트를 반환하는 새로운 함수에 이것들을 모두 넣을 수 있다.

```
# transform series into train and test sets for supervised learning
def prepare_data(series, n_test, n_lag, n_seq):
        # extract raw values
        raw_values = series.values
        raw_values = raw_values.reshape(len(raw_values), 1)
        # transform into supervised learning problem X, y
        supervised = series_to_supervised(raw_values, n_lag, n_seq)
        supervised_values = supervised.values
        # split into train and test sets
        train, test = supervised_values[0:-n_test], supervised_values[-n_test:]
        return train, test
```

예측 함수

Make Forecasts

다음 단계는 지속성 예측을 하는 것이다.

지속성 예측은 persistence()이라는 이름의 함수에서 마지막 관측과 지속할 예측 단계의 수를 취하는 것으로 쉽게 구현할 수 있다.

이 함수는 예측을 포함하는 배열을 반환한다.

```
# make a persistence forecast
def persistence(last_ob, n_seq):
        return [last_ob for i in range(n_seq)]
```

그러면 우리는 이 함수를 2년차 12월 부터 3년차 9월까지 테스트 데이터세트의 각 타임 스탭별로 호출할 수 있다.

아래는 함수 make_forecasts()이다. 이 함수는 이것을 하고(타임 스탭별 호출?) 인자로 데이터 세트에 대한 구성과 학습 및 테스트 데이터를 취하며 예측 리스트를 반환한다.

```
# evaluate the persistence model
def make_forecasts(train, test, n_lag, n_seq):
        forecasts = list()
        for i in range(len(test)):
                X, y = test[i, 0:n_lag], test[i, n_lag:]
                # make forecast
                forecast = persistence(X[-1], n_seq)
                # store the forecast
                forecasts.append(forecast)
        return forecasts
```

We can call this function as follows:

```
forecasts = make_forecasts(train, test, 1, 3)
```

예측평가 함수

Evaluate Forecasts

마지막 단계는 예측을 평가하는 것이다.

multi-step 예측의 각 time step에 대한 RMSE를 계산함으로써 평가한다. 이번 경우에는 3개의 RMSE 점수가 주어진다.

아래의 함수 evaluate_forecasts()는 각 time step별 예측에 대한 RMSE를 계산하고 출력한다.

```
# evaluate the RMSE for each forecast time step
def evaluate_forecasts(test, forecasts, n_lag, n_seq):
        for i in range(n_seq):
                actual = test[:,(n_lag+i)]
                predicted = [forecast[i] for forecast in forecasts]
                rmse = sqrt(mean_squared_error(actual, predicted))
                print('t+%d RMSE: %f' % ((i+1), rmse))
```

We can call it as follows:

```
evaluate_forecasts(test, forecasts, 1, 3)
```

우리는 원래 데이터 세트안의 실제 지속되는 값에 지속되는 예측을 연결함으로써 더 나은 도표를 만들 수 있다.

이를 위해 예측의 앞에 마지막 관측 값을 추가한다.

아래는 plot_forecasts() 함수의 업데이트된 버전이다.

```
# plot the forecasts in the context of the original dataset
def plot_forecasts(series, forecasts, n_test):
        # plot the entire dataset in blue
        pyplot.plot(series.values)
        # plot the forecasts in red
        for i in range(len(forecasts)):
                off_s = len(series) - 12 + i - 1
                off_e = off_s + len(forecasts[i]) + 1
                xaxis = [x for x in range(off_s, off_e)]
                yaxis = [series.values[off_s]] + forecasts[i]
                pyplot.plot(xaxis, yaxis, color='red')
        # show the plot
        pyplot.show()
```

시각화

예제를 몇 번 실행해보고 평균 결과를 비교해보자.

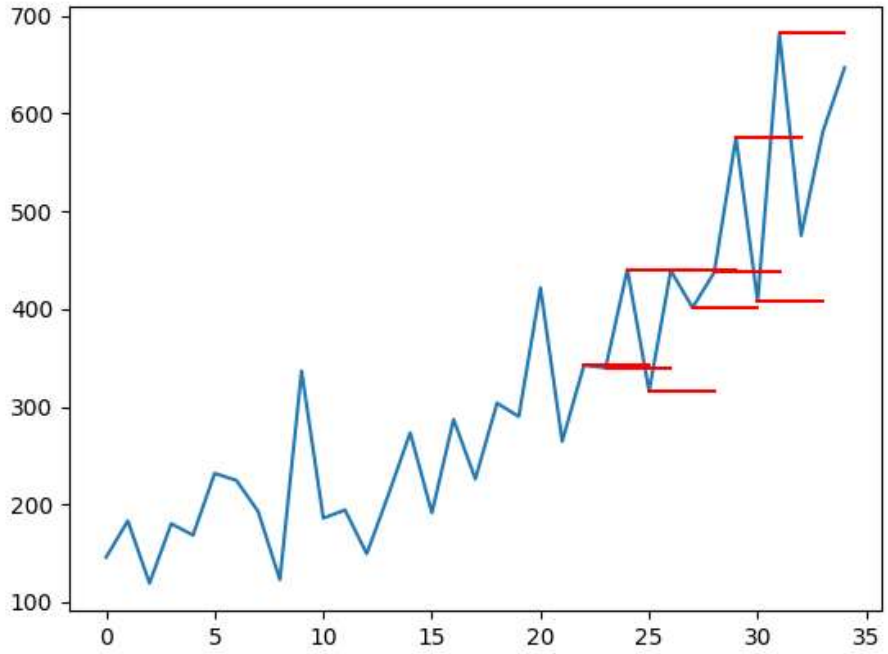
이것은 LSTM이 능가할 수 있는 각 단계별 성능 기준을 제공한다.

```
t+1 RMSE: 144.535304
t+2 RMSE: 86.479905
t+3 RMSE: 121.149168
```

multi-step 지속성 예측이 있는 원래 시계열의 도표도 생성된다.

이 선들은 각 예측에 대한 적절한 입력 값에 연결된다.

이 문맥은 지속성 예측이 실제로 얼마나 naive한가를 보여준다.



End of Documents

Thank You