



머신러닝 기반 빅데이터 분석 & 시각화 과정

2021. 3. 29 ~ 4.2 (5일 과정)

정준수 Ph.D

과정 목표

복잡한 데이터 구조 패턴을 기계(컴퓨터)로 하여금 스스로 학습하게 하는 인공지능 기계학습 기술을 활용하여 현업의 빅데이터를 지도학습이나 자율학습형태로 분석하고 성능을 평가하여 그 결과를 실제 업무에 적용

1. 분석 방법론, 빅데이터 분석 사례, 파이썬을 이용한 빅데이터 분석 환경 구성
2. 지도학습 모델, 회귀분석, 시계열 분석, 예측 모델 개발, HMM, SVM
3. 지도학습 모델, 분류 분석, 인공신경망, 앙상블모델(XGBoost, LGBM)
4. 의사결정나무, SVM, 자율학습 모델, 데이터 군집 모델 개발, K-means Clustering
5. 빅데이터 모델 평가, 혼돈 메트릭스, AUC, RMSE, F-measure
6. 시각화 도구, 시각화 스토리 텔링, 분석 정보 시각화하기, 시각화 보정

Deep learning is not like pure mathematics. It is a heavily experimental field, so it's important to be a strong practitioner, not just a theoretician.

딥 러닝은 순수한 수학과는 다릅니다. 매우 실험적인 분야이기 때문에 이론가가 아닌 실무자가 되는 것이 중요합니다.

<실습 예제 프로그램>

<https://github.com/JSJeong-me/KOSA-ML-BigData-Analysis>

머신러닝이란?

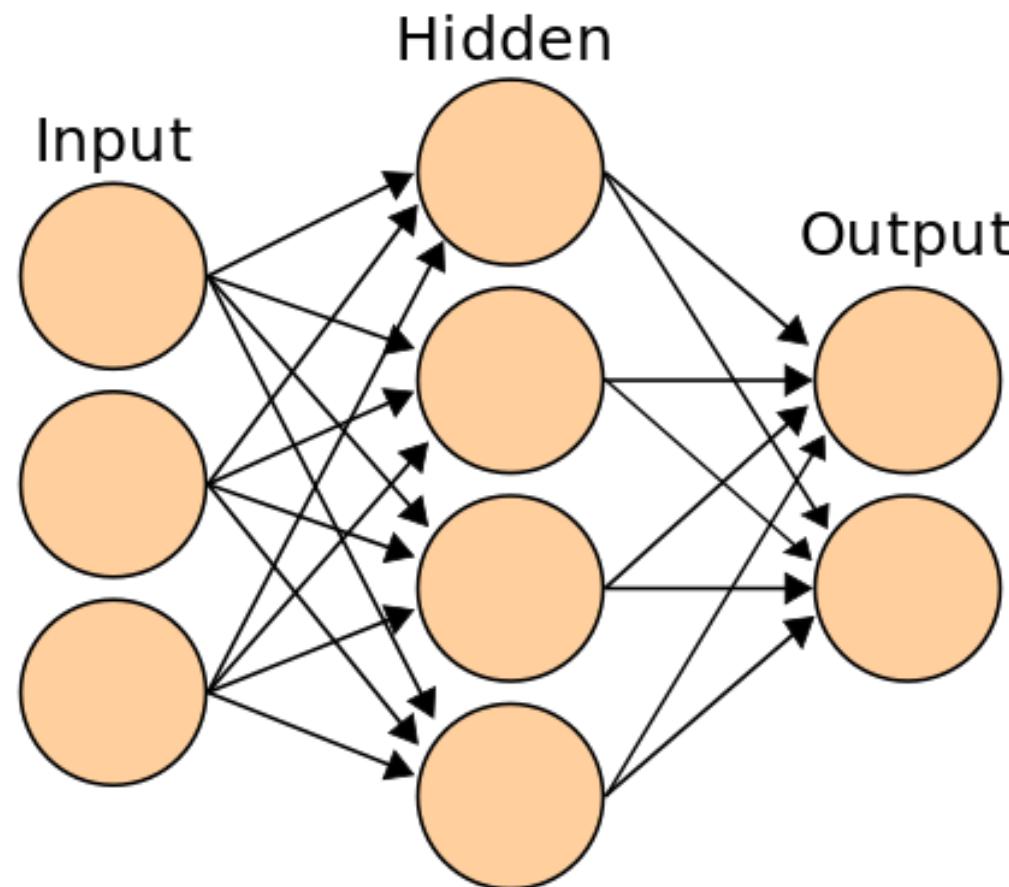
인간이 다양한 경험과 시행착오를 통해 지식을 배우는 것처럼, 컴퓨터에게 충분히 많은 데이터를 주고, 거기에서 일반적인 패턴을 찾아내게 하는 방법을 말합니다. (머신러닝의 대표적인 알고리즘이 딥러닝입니다.)

딥러닝이란?

머신러닝의 대표적인 학습법이 다음장 그림처럼 여러 층을 거쳐 점점 추상화 단계로 접어드는 알고리즘 형태인 '딥 러닝(Deep Learning)'입니다.

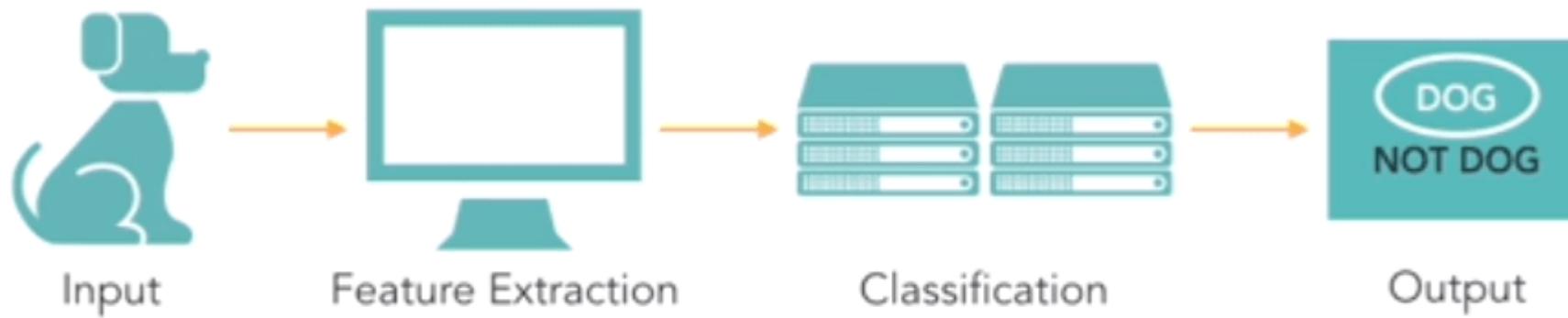
패턴을 찾기 위해선, 수많은 데이터가 있어야 되겠지요. 패턴을 견고하게 만드는 일종의 학습용 훈련데이터 말입니다.

딥러닝 레이어



Machine Learning vs Deep Learning

TRADITIONAL MACHINE LEARNING

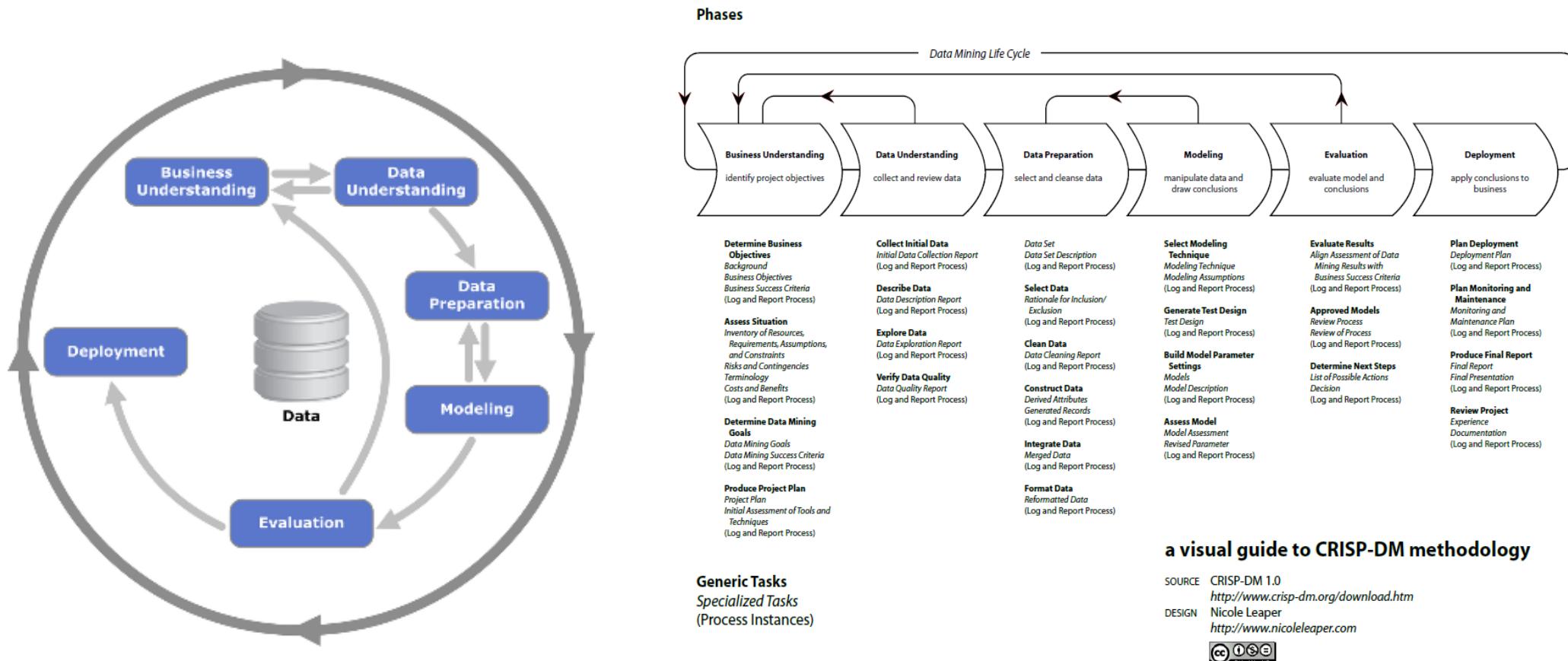


DEEP LEARNING

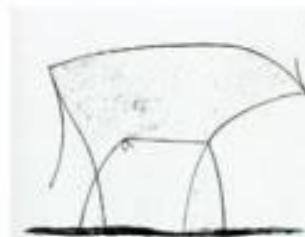
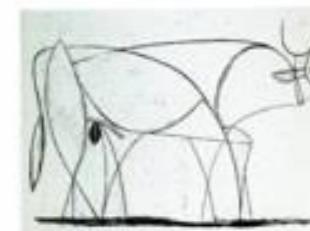
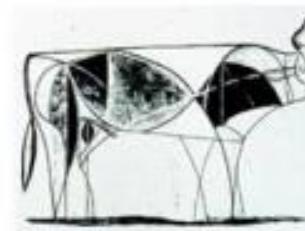
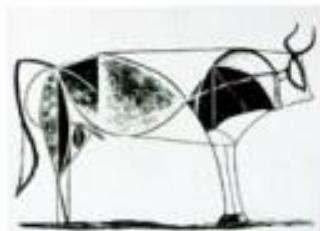
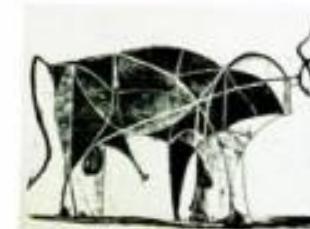
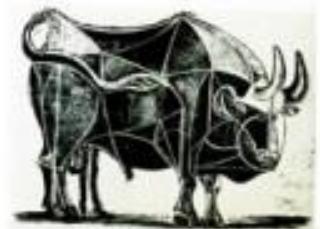
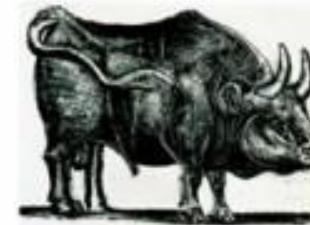
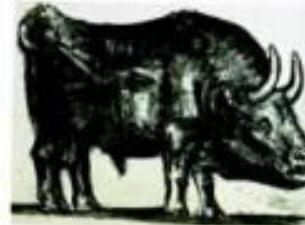
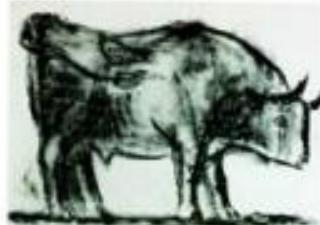


CRISP-DM (Cross Industry Standard Process for Data Mining)

CRISP-DM(Cross Industry Standard Process for Data Mining)은 데이터 마이닝 전문가가 사용하는 일반적인 접근 방식을 설명한 가장 널리 사용되는 공개 표준 분석 모델입니다.



◆ 추상화 (Abstract)

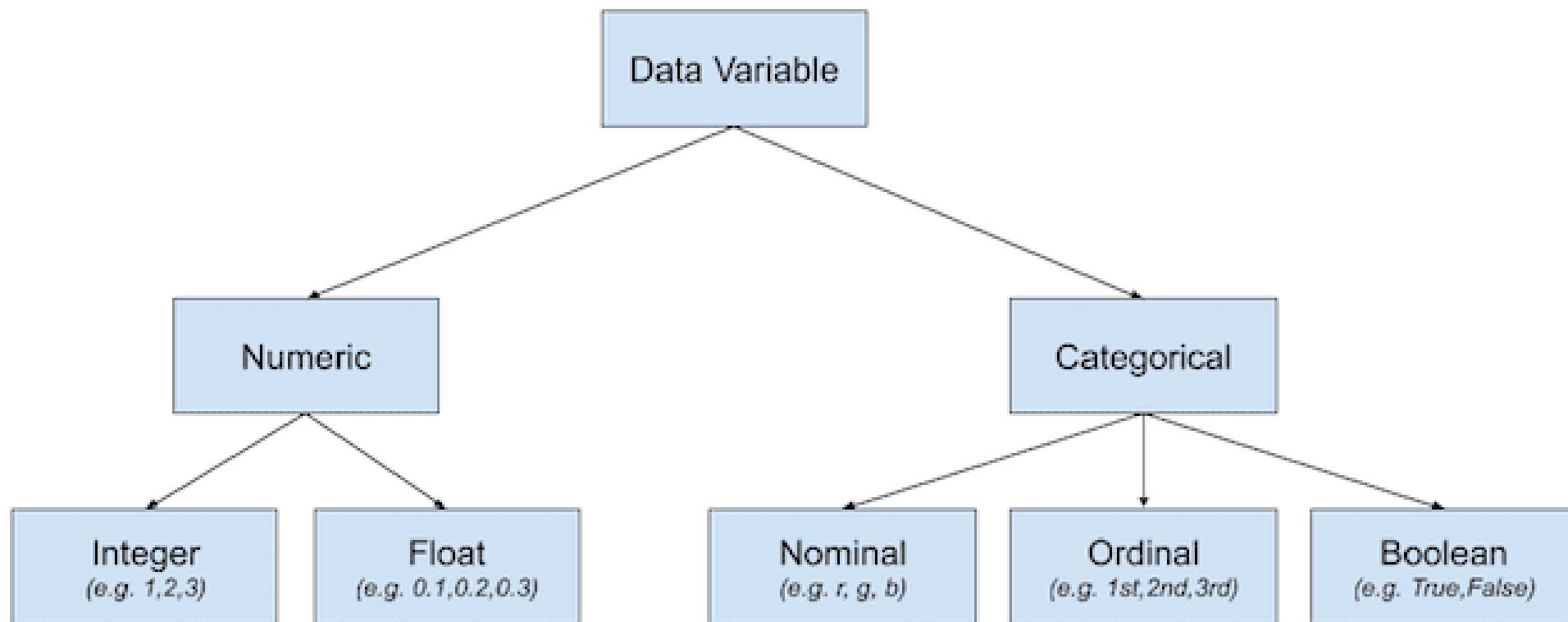


Picasso

Pablo Picasso, Bull (plates I - XI) 1945

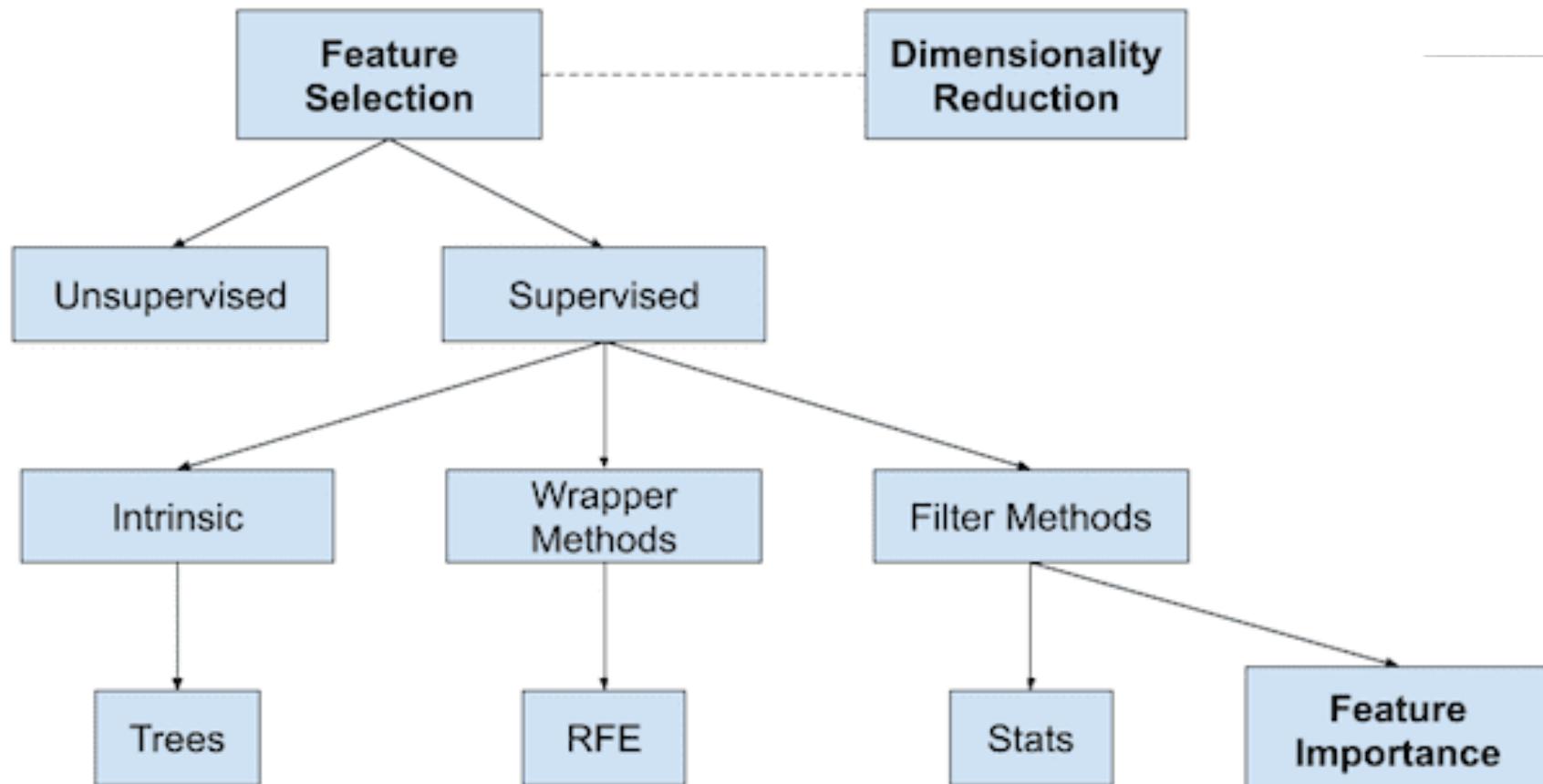
Data 변수 타입 분류

Overview of Data Variable Types



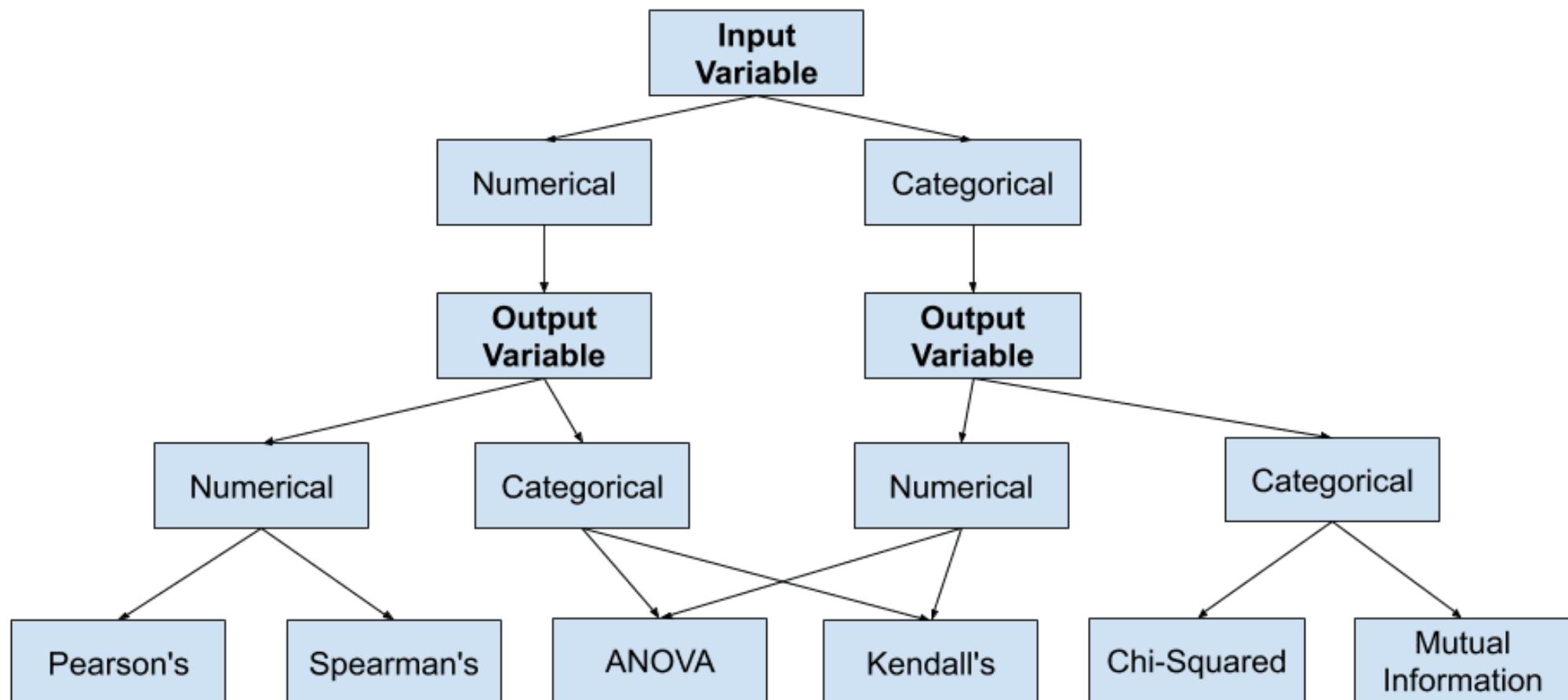
Feature Selection 분류 과정

Overview of Feature Selection Techniques



Feature Selection 분류 방법

How to Choose a Feature Selection Method



머신러닝에서 일반적 Data Preparation 과정 정리

1. 데이터 준비 과정의 중요성
2. 결측치의 처리방법
3. 특징 추출 (Recursive Feature Elimination)
4. 데이터 정규화
5. 원 핫 인코딩으로 범주 변환 (One Hot Encoding)
6. 숫자 변수의 범주형 변수로 변환
7. PCA를 통한 차원 축소

1. 데이터 준비 과정의 중요성

데이터 준비 과정은 원시 데이터를 모델링에 적합한 형식으로 변환하는 작업이 중심 내용이며, 데이터를 준비하는 것은 예측 모델 생성 프로젝트에서 가장 중요한 부분이며 가장 많은 시간이 소요된다. 모든 데이터 준비 과정은 기계 학습 알고리즘에 맞추어져 진행 되며, 실제 데이터와 매개 변수 등을 활용한다. 실질적인 데이터 준비 과정에는 데이터 정리, 특징 추출, 데이터 변환, 차원 축소 등에 대한 지식이 필요하다.

예측 모델링 프로젝트에는 데이터로부터 학습이 진행됩니다. 데이터는 해결하려는 문제를 특징 짓는 도메인으로부터 경험 할 수 있는 사례를 제공하지만, 분류 또는 회귀와 같은 예측 모델링 프로젝트에서 원시 데이터는 일반적으로 직접 사용할 수 없습니다. 이것이 사실인 네 가지 주요 이유가 있습니다.

원시 데이터는 기계 학습 모델을 맞추고 평가하는 데 사용되기 전에 사전 처리 되어야 합니다. 기계 학습 프로젝트의 데이터 준비 단계에서 사용하거나 탐색 할 수 있는 일반 또는 표준 작업이 있습니다.

- 데이터 정리 : 데이터의 오류 또는 오류를 식별하고 수정
- 특징 선택 : 작업과 가장 관련된 입력 변수 식별
- 데이터 변환 : 변수의 척도 또는 분포 파악
- 특징 엔지니어링 : 사용 가능한 데이터에서 새로운 변수 도출
- 차원 감소 : 데이터의 간결한 예측 생성

2. 결측치의 처리방법

실제 데이터에는 종종 결 측값이 있습니다. 데이터에는 기록되지 않은 관찰 및 데이터 손상과 같은 여러 가지 이유로 인해 누락 된 값이 있을 수 있습니다. 누락 된 데이터 처리가 중요합니다. 많은 기계 학습 알고리즘이 결 측값이 있는 데이터를 지원하지 않기 때문입니다. 누락 된 값을 데이터로 채우는 것을 데이터 대치라고 하며 데이터 대치에 대한 일반적인 접근 방식은 각 열 (예 : 평균)에 대한 통계 값을 계산하고 해당 열의 모든 누락 된 값을 통계로 바꾸는 것입니다.

(결측치 처리 예제 – 단순 평균값 입력)

SimpleImputer() 클래스를 사용하여 NaN 값으로 표시된 모든 누락 된 값을 열의 평균으로 변환 할 수 있습니다.

<실습 프로그램>

<https://gist.github.com/JSJeong-me/fdbba476a9cff9400ba32064a92f54e8>

3. 특징 추출 (Recursive Feature Elimination)

특징 선택은 예측 모델을 개발할 때 입력 변수의 수를 줄이는 프로세스입니다. 모델링의 계산 비용을 줄이고 경우에 따라 모델의 성능을 향상시키기 위해 입력 변수의 수를 줄이는 것이 바람직합니다. 간단히 RFE는 인기있는 기능 선택 알고리즘입니다.

RFE는 구성 및 사용이 쉽고 대상 변수를 예측하는 데 더 많거나 가장 관련성이 높은 학습 데이터 세트에서 이러한 기능 (열)을 선택하는 데 효과적이기 때문에 널리 사용됩니다.

(특징 추출 예제 – scikit-learn 사용)

scikit-learn Python 기계 학습 라이브러리는 기계 학습을 위한 RFE 구현을 제공합니다. RFE 변환을 사용하려면 먼저 추정치 인수를 통해 지정된 선택한 알고리즘과 인수를 선택하기 위해 n 개의 기능을 통해 선택할 기능수로 클래스를 구성합니다. 다음의 예는 5 개의 중복 입력 기능이 있는 합성 분류 데이터 세트를 정의합니다. 그런 다음 RFE를 사용하여 의사 결정 트리 알고리즘을 사용하여 5 개의 기능을 선택합니다.

<실습 프로그램>

<https://gist.github.com/JSeong-me/7d6a3f852eb0e9eb451e4c153af6cc6f>

4. 데이터 정규화

기계 학습을 위해 숫자 데이터를 확장하는 방법을 알아 봅니다. 많은 기계 학습 알고리즘은 숫자 입력 변수가 표준 범위로 조정될 때 더 잘 수행됩니다. 여기에는 선형 회귀와 같은 입력의 가중 합계를 사용하는 알고리즘과 k- 최근 접 이웃과 같은 거리 측정을 사용하는 알고리즘이 포함됩니다.

모델링 전에 수치 데이터를 스케일링하는 가장 널리 사용되는 기술 중 하나는 정규화입니다. 정규화는 각 입력 변수를 0-1 범위로 개별적으로 조정합니다. 이는 가장 정밀도가 높은 부동 소수점 값의 범위입니다. 각 변수에 대한 최소 및 최대 관찰 가능 값을 알고 있거나 정확하게 추정 할 수 있어야합니다.

(데이터 정규화 예제 – MinMaxScaler 사용)

scikit-learn 객체 MinMaxScaler를 사용하여 데이터 세트를 정규화 할 수 있습니다. 아래 예제는 합성 분류 데이터 세트를 정의한 다음 MinMaxScaler를 사용하여 입력 변수를 정규화합니다.

<실습 프로그램>

<https://gist.github.com/JSJeong-me/fa8e38c0d0960f520731de45f7e1b6eb>

5. 원 핫 인코딩으로 범주 변환 (One Hot Encoding)

범주 형 입력 변수를 숫자로 인코딩하는 방법을 알아 봅니다. 기계 학습 모델에서는 모든 입력 및 출력 변수가 숫자여야 합니다. 즉, 데이터에 범주형 데이터가 포함된 경우 모델을 적합하고 평가하기 전에 이를 숫자로 인코딩 해야합니다. 범주 형 변수를 숫자로 변환하는 가장 널리 사용되는 기술 중 하나는 원 핫 인코딩입니다. 범주 형 데이터는 숫자 값이 아닌 레이블 값을 포함하는 변수입니다.

범주 형 변수에 대한 각 레이블은 서수 인코딩이라고하는 고유 한 정수에 매핑 될 수 있습니다. 그런 다음 서수 표현에 원 핫 인코딩을 적용 할 수 있습니다. 여기서 변수의 고유 한 정수 값 각각에 대해 하나의 새 이진 변수가 데이터 세트에 추가되고 원래 범주 형 변수가 데이터 세트에서 제거됩니다.

(데이터 Encoding 예제 – One Hot Encoding 사용)

이 핫 인코딩 변환은 OneHotEncoder 클래스를 통해 scikit-learn Python 기계 학습 라이브러리에서 사용할 수 있습니다. 유방암 데이터 세트에는 범주형 입력 변수만 포함됩니다. 아래 예제는 데이터 세트를 로드하고 하나의 핫 인코딩은 각 범주형 입력 변수를 인코딩합니다.

<실습 프로그램>

<https://gist.github.com/JSeong-me/664af116682e32f0e1fb0141c280d879>

6. 숫자 변수의 범주형 변수로 변환

숫자 변수를 범주형 변수로 변환하는 방법을 알아 봅니다. 일부 기계 학습 알고리즘은 일부 의사 결정 트리 및 규칙 기반 알고리즘과 같은 범주 형 또는 순서 형 입력 변수를 선호하거나 요구할 수 있습니다. 이것은 데이터, 다중 입력 데이터 분포, 고도의 지수 분포 등. 많은 기계 학습 알고리즘은 비표준 분포를 가진 숫자 입력 변수가 새로운 분포 또는 완전히 새로운 데이터 유형을 갖도록 변환 될 때 더 나은 성능을 제공합니다.

한 가지 접근 방식은 숫자 변수의 변환을 사용하여 각 숫자 값에 레이블이 할당되고 레이블에 순서가 지정된 (순서적) 관계가 있는 이산 확률 분포를 갖는 것입니다. 이를 이산화 변환이라고 하며 숫자 입력 변수의 확률 분포를 이산 적으로 만들어 데이터 세트에 대한 일부 기계 학습 모델의 성능을 향상시킬 수 있습니다.

(숫자 변수의 범주형 변수 변환 예제 – KBinsDiscretizer 사용)

이산화 변환은 KBinsDiscretizer 클래스를 통해 scikit-learn Python 기계 학습 라이브러리에서 사용할 수 있습니다. 생성 할 이산 구간의 수 (n 구간), 변환 결과가 서수인지 아니면 하나의 핫 인코딩 (인코딩)인지, 변수 값을 나누는 데 사용되는 분포 (전략)를 지정할 수 있습니다. 아래 예제에서는 10 개의 숫자 입력 변수가 있는 합성 입력 변수를 생성 한 다음 각각을 서수 인코딩을 사용하여 10 개의 개별 Bin(바구니)으로 인코딩합니다.

<실습 프로그램>

<https://gist.github.com/JSJeong-me/34721c717cf4a4f9e5eaee4f33124c2>

7. PCA를 통한 차원 축소

차원 감소를 사용하여 데이터 세트의 입력 변수를 줄이는 방법을 알아 봅니다. 데이터 세트에 대한 입력 변수 또는 기능의 수를 차원이라고 합니다. 차원 감소는 Features 수를 줄이는 기술을 의미합니다. 데이터 세트의 입력 변수. 더 많은 입력 기능은 종종 예측 모델링 작업을 모델링하기 더 어렵게 만들고, 일반적으로 차원의 저주라고 합니다. 고차원 통계에서는 차원 감소 기술이 데이터 시각화에 자주 사용되지만, 이러한 기술은 예측 모델에 더 적합하도록 분류 또는 회귀 데이터 세트를 단순화하기 위해 적용된 기계 학습에서 사용할 수 있습니다. 기계 학습에서 차원 감소를 위한 가장 인기있는 기술은 주성분 분석 (줄여서 PCA) 일 것입니다.

(PCA를 통한 차원 축소 예제 – PCA)

scikit-learn 라이브러리는 데이터 세트에 적합하고 향후 학습 데이터 세트 및 추가 데이터 세트를 변환하는 데 사용할 수 있는 PCA 클래스를 제공합니다. 아래 예제는 10 개의 입력 변수가 있는 합성 이진 분류 데이터 세트를 만든 다음 PCA를 사용하여 데이터 세트의 차원을 가장 중요한 세 가지 구성 요소로 줄입니다.

<실습 프로그램>

<https://gist.github.com/JSJeong-me/ec0826fc9da3f7e1bac19bcf95aef47f>

Machine Learning에서 “예측”이란?

이전에 본적 없는 새로운 데이터에 대한 정확한 출력 예측

Predictive Analytics

1 무엇을 예측하는가?

2 무엇을 할 것인가?

Machine Learning - Training

	Input			Output
Example 1	0	0	1	0
Example 2	1	1	1	1
Example 3	1	0	1	1
Example 4	0	1	1	0



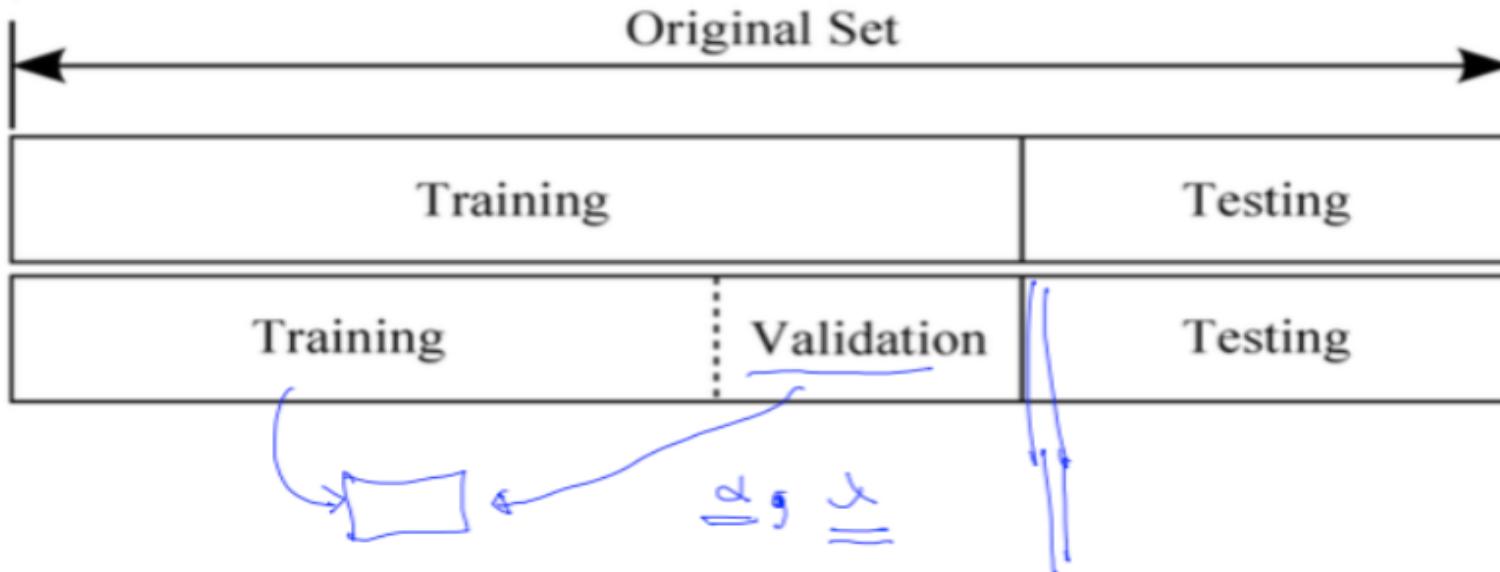
New situation	1	0	0	?
----------------------	---	---	---	---

```
from numpy import exp, array, random, dot
training_set_inputs = array([[0, 0, 1], [1, 1, 1], [1, 0, 1], [0, 1, 1]])
training_set_outputs = array([[0, 1, 1, 0]]).T
random.seed(1)
synaptic_weights = 2 * random.random((3, 1)) - 1
for iteration in range(10000):
    output = 1 / (1 + exp(-(dot(training_set_inputs, synaptic_weights))))
    synaptic_weights += dot(training_set_inputs.T, (training_set_outputs - output) * output * (1 - output))
print (1 / (1 + exp(-(dot(array([1, 0, 0]), synaptic_weights)))))
```

출처: <https://medium.com/technology-invention-and-more/how-to-build-a-simple-neural-network-in-9-lines-of-python-code-cc8f23647ca1>

머신러닝 학습 데이터의 구성

Training, validation and test sets



데이터 영역별 구성도다. 데이터가 충분하다면 validation set을 구성하지 않을 이유가 없다. 쉽게 보면 test를 한번 더 한다고 생각해도 좋을 것이다. 머신러닝의 교과서에 해당하는 mnist 예제는 train 55,000개, validation 5,000개, test 10,000개의 dataset으로 구성되어 있다.

예측 분석 응용: 이탈 모델링으로 고객 이탈 방지하기

1 무엇을 예측하는가?

어느 고객이 떠나갈 것인가.

2 무엇을 할 것인가?

떠날 위기에 있는 고객들을 타깃으로 한 고객 유지 마케팅을 수행한다.

예측 분석 응용: 부동산 담보대출 채권 가치 추산

1 무엇을 예측하는가?

부동산 담보대출 고객 중에서 누가 향후 90일 내에 조기 상환할 것인가.

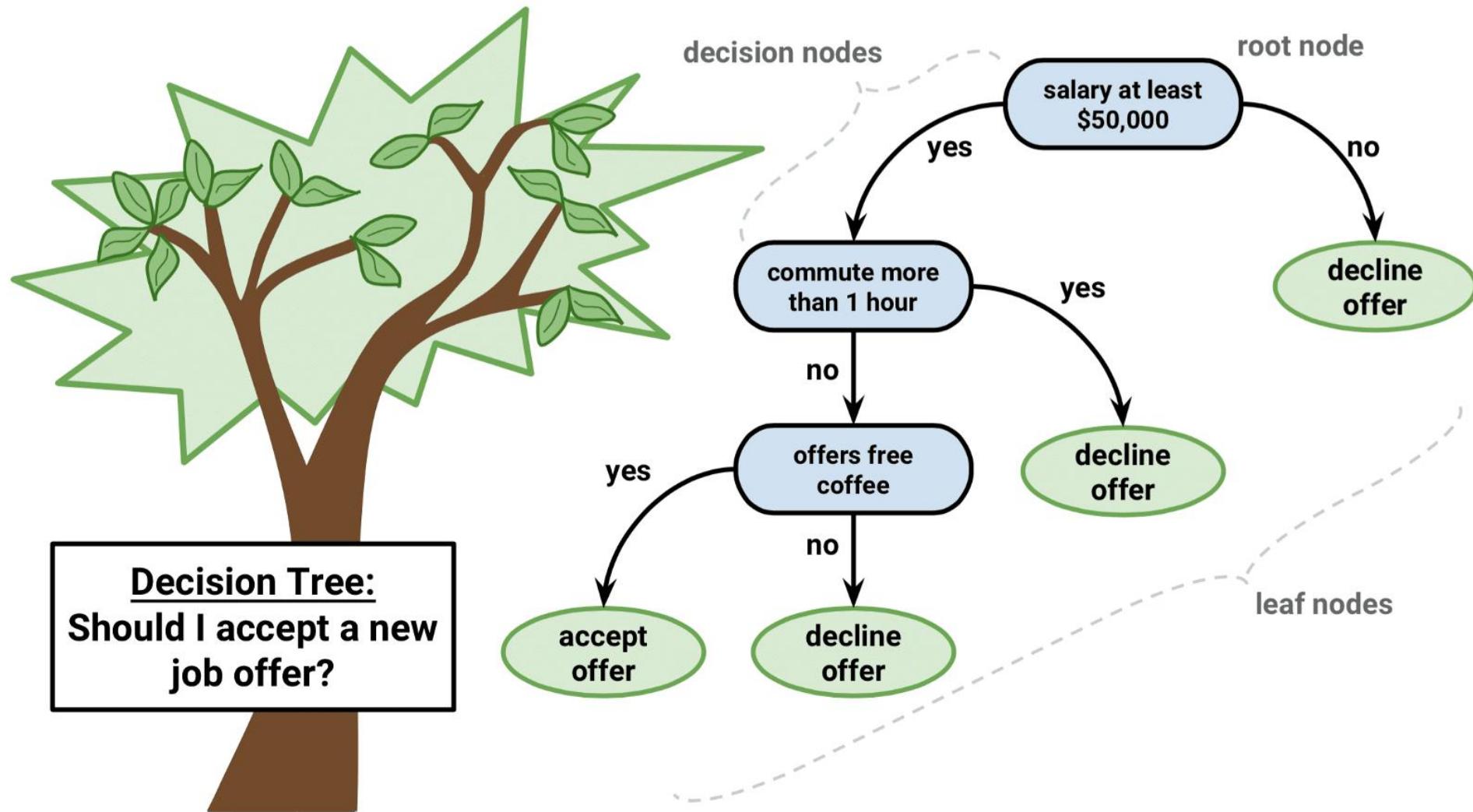
2 무엇을 할 것인가?

부동산 담보대출 채권의 가치를 계산한 후 다른 은행에 팔아 넘길지 여부를 결정한다.

나무에서 돈이 자란다!

체이스 은행 예측모델은 부동산 담보대출 중에서 실제로
조기상환된 대출 건들 중 74%를 정확하게 인식해 내어서
부동산 담보대출 포트폴리오를 성공적으로 관리함.

Decision Tree



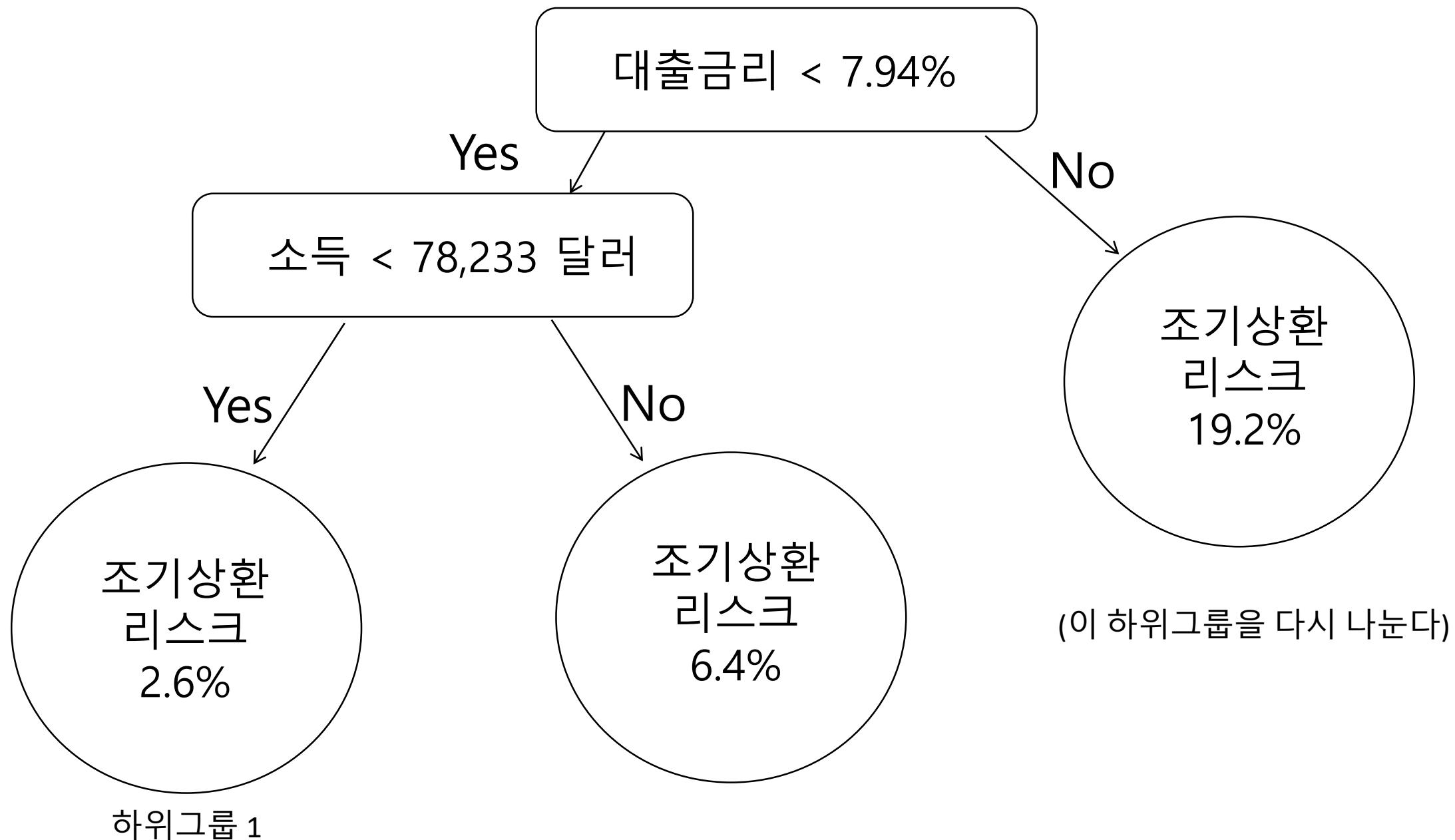
대출금리가
7.94%보다 낮은가?

Yes

No

조기상환
리스크
3.8%

조기상환
리스크
19.2%



대출금리 < 7.94%

Yes

소득 < 78,233 달러

Yes

조기상환
리스크
2.6%

No

조기상환
리스크
6.4%

No

대출금액 < 182,926달러

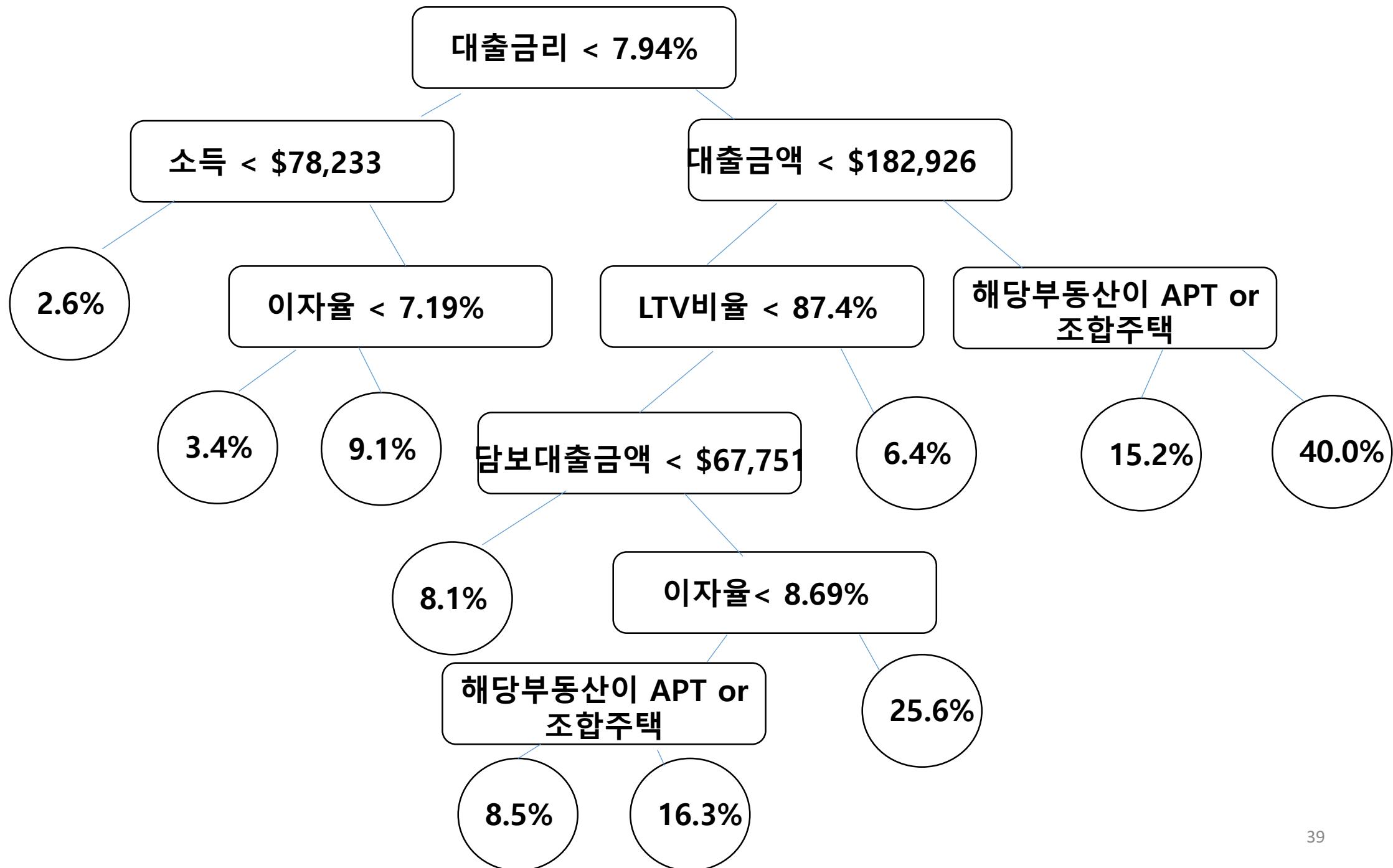
Yes

조기상환
리스크
13.9%

No

조기상환
리스크
36%

하위그룹4



만약(IF):

부동산 담보대출 금액이 67,751 달러와 같거나 그보다 더 많고 182,926 달러보다 작다.

그리고(AND):

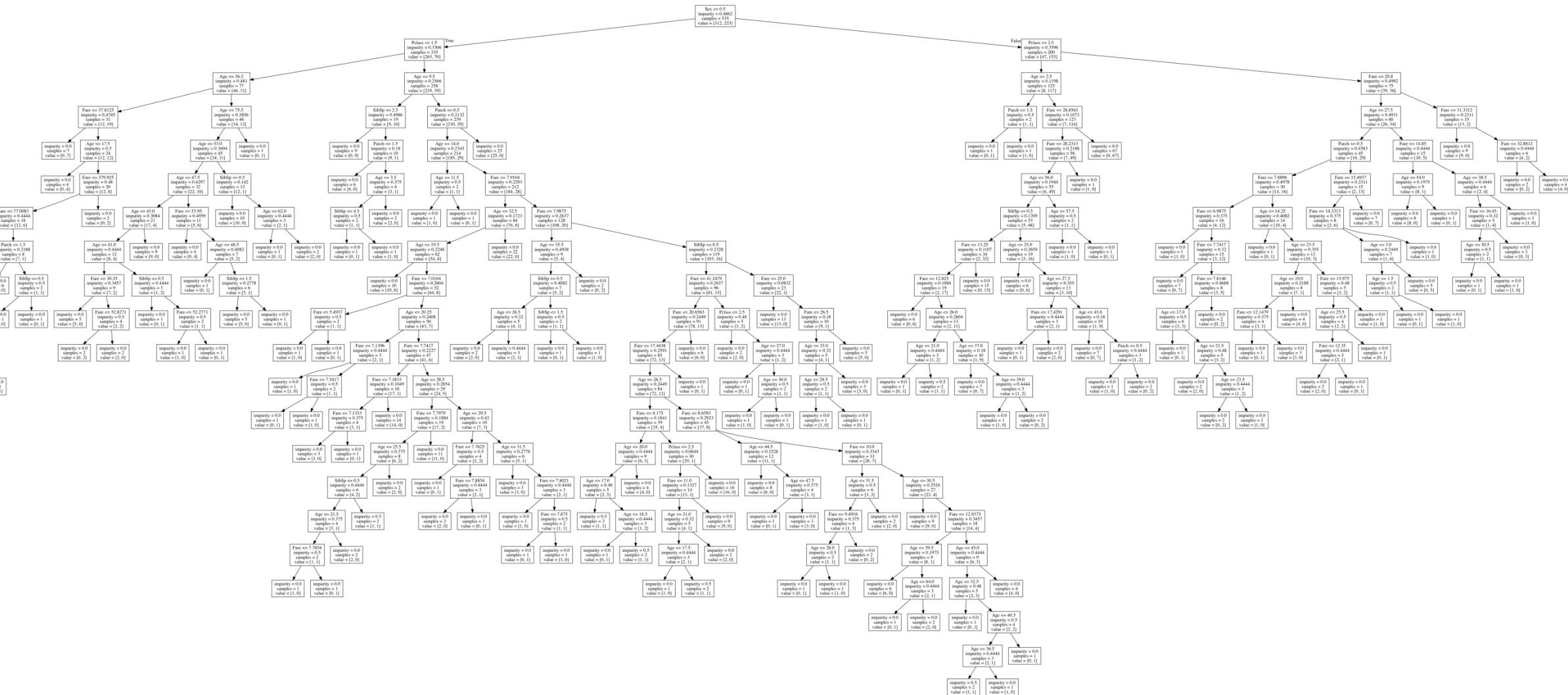
이자율이 8.69%와 같거나 그보다 더 높다.

그리고(AND):

부동산 자산가치 대비 대출금액의 비율이 87.4% 보다 작다.

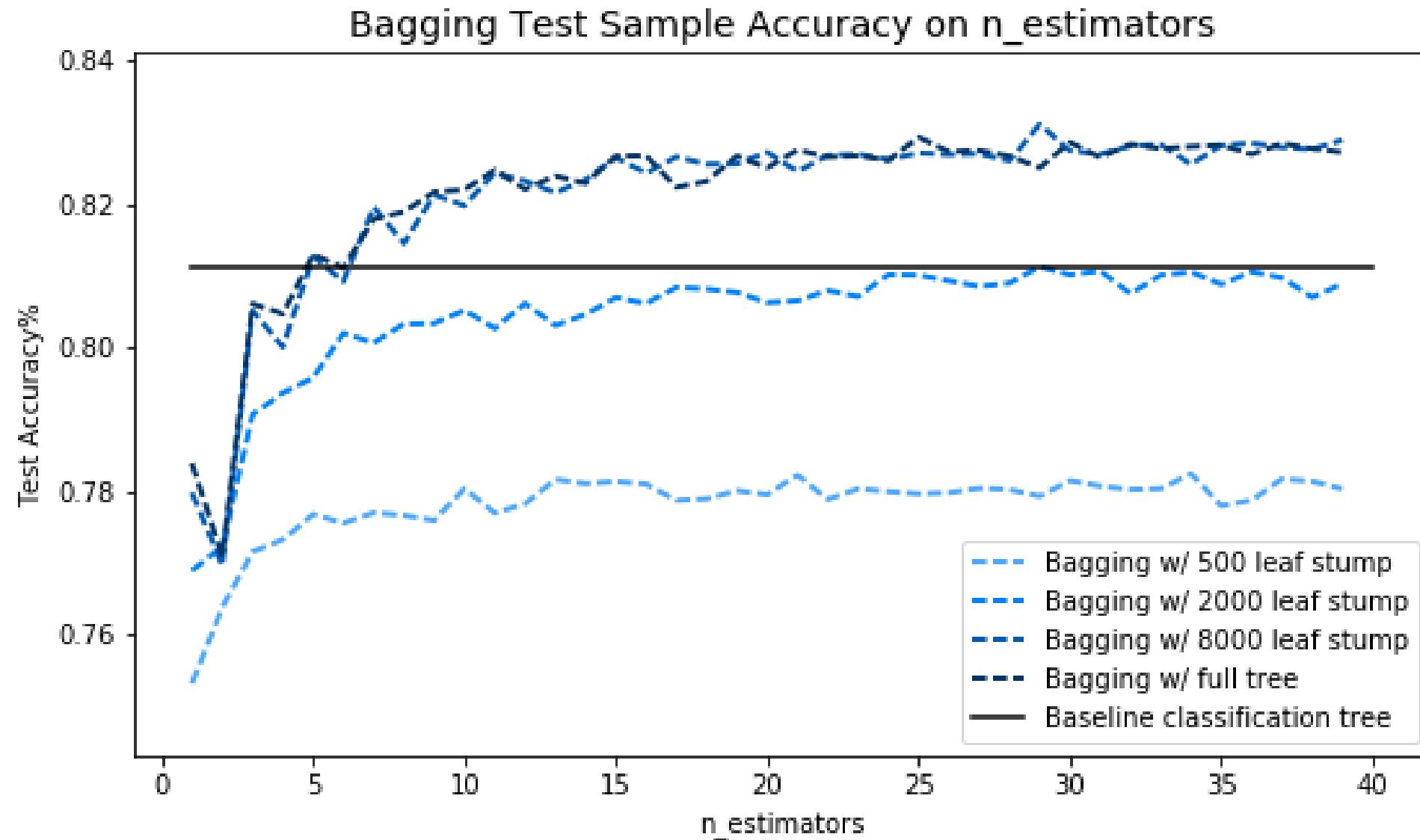
그러면(THEN):

조기상환 확률은 25.6% 이다.



Decision Tree: 20% 기준 다양한 향상도 성과 기록

Decision Tree(Leafs)	20% 기준 향상도
4개의 그룹	2.5
10개의 그룹	3.0
39개의 그룹	3.0



보·이·는·대·로·믿·지·마·라!

대/반/전/ 음악추리쇼

나의 목소리가 보여

E



Mnet tvN 공동 방송

10월 22일 목요일 | 밤 9시 40분

6명 중에는 음치도 있고 노래를 잘하는 실력자(정상)도 있다.

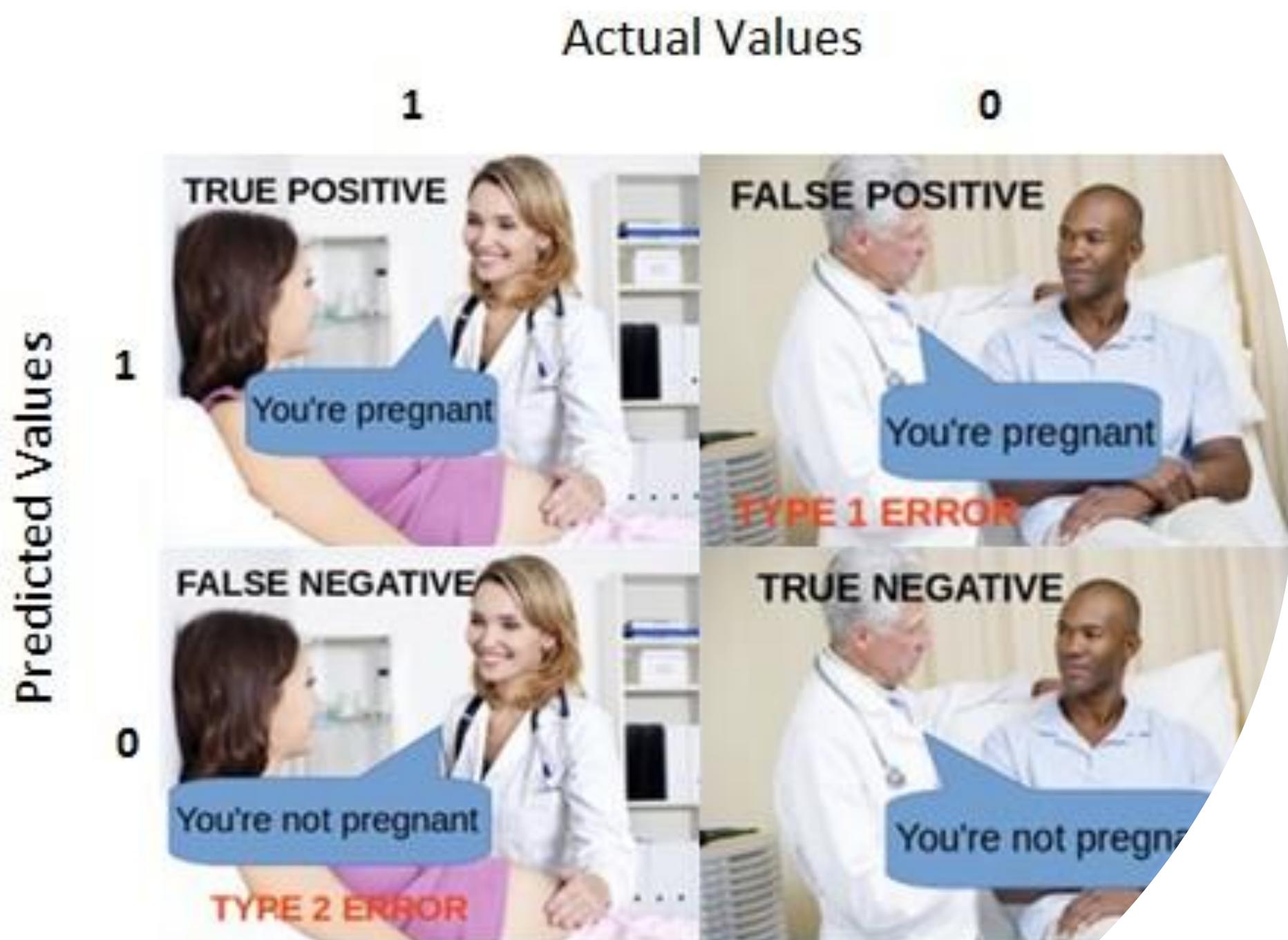
번호 [1, 2, 3, 4, 5, 6]

정답 : [음치, 음치, 음치, 음치, 정상, 정상] 가 있다.

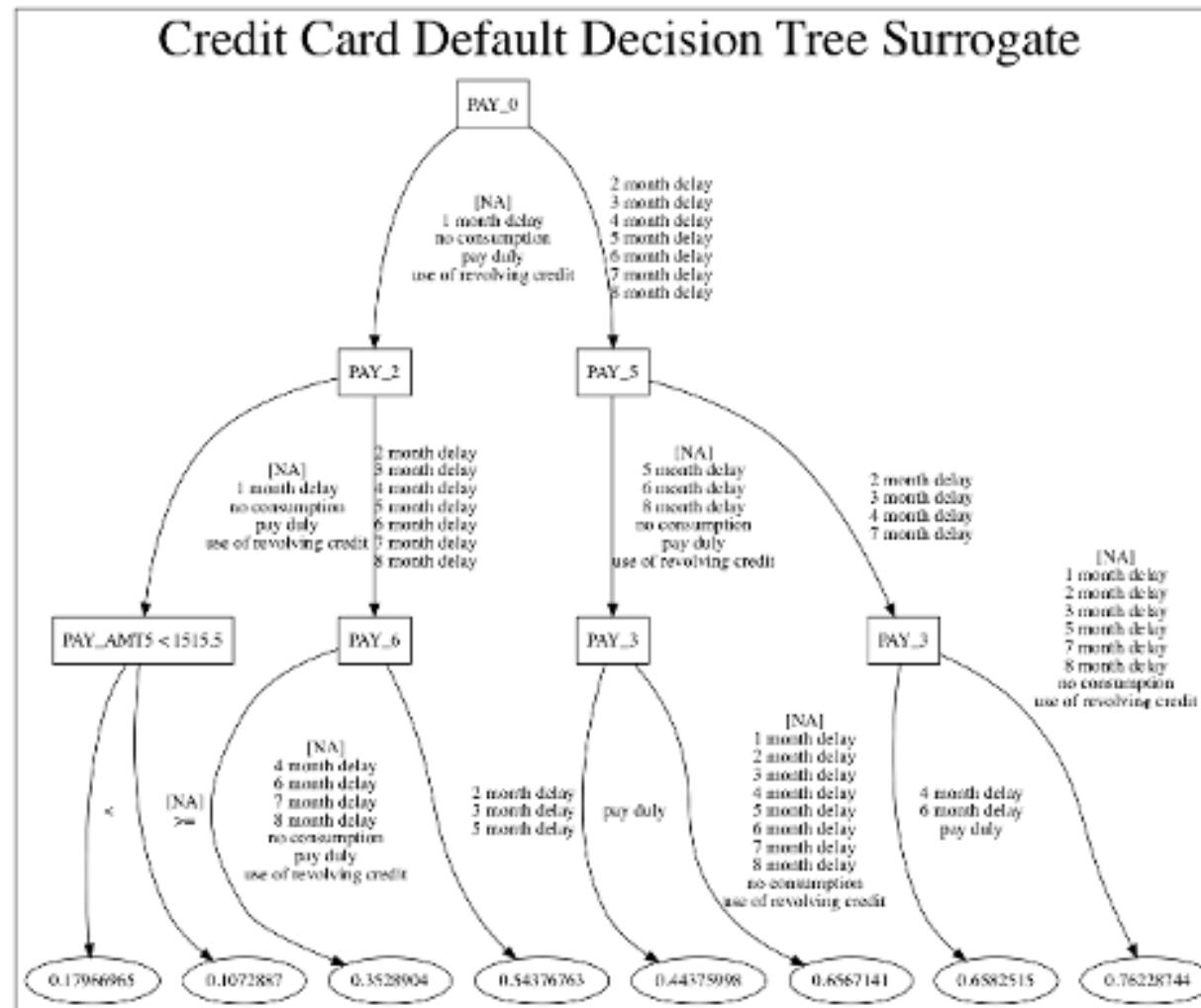
누가 음치인지 겉모습만 보고 맞춰야 한다.

감으로 예측을 한다.

예측 : [음치, 음치, 정상, 정상, 정상, 정상]



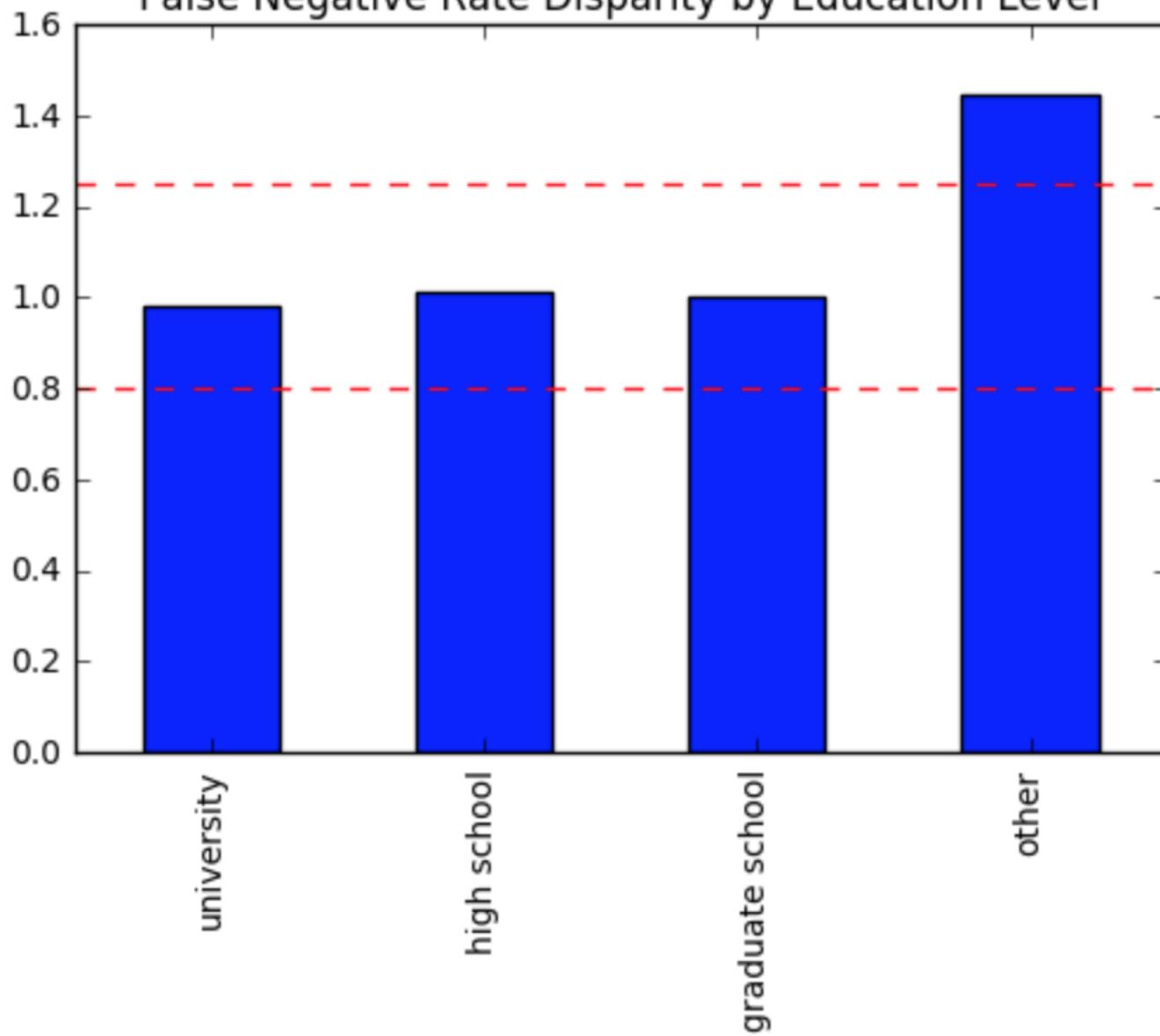
머신러닝 결과 분석(Decision Tree)



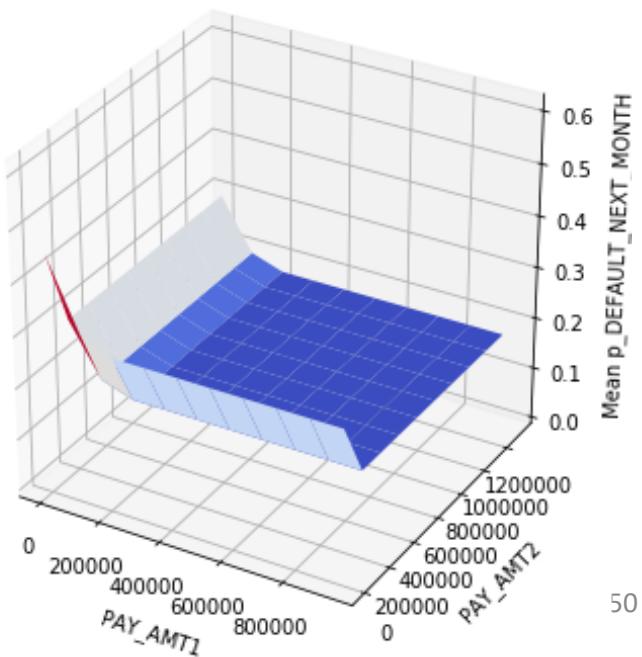
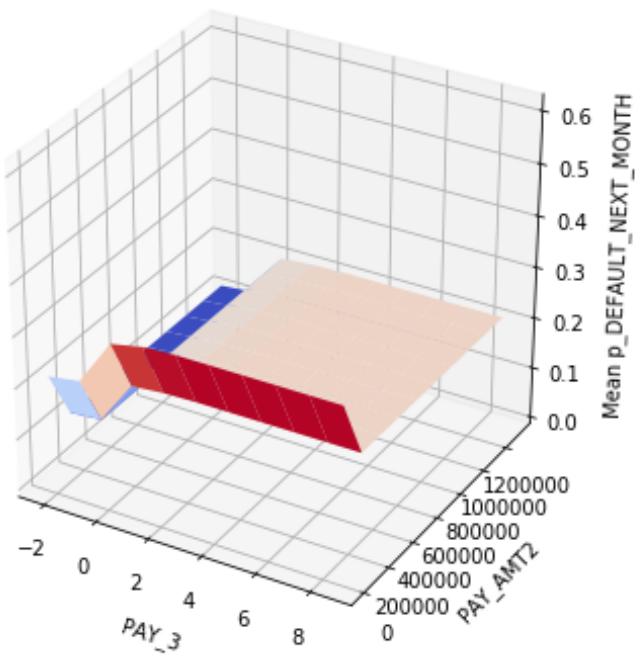
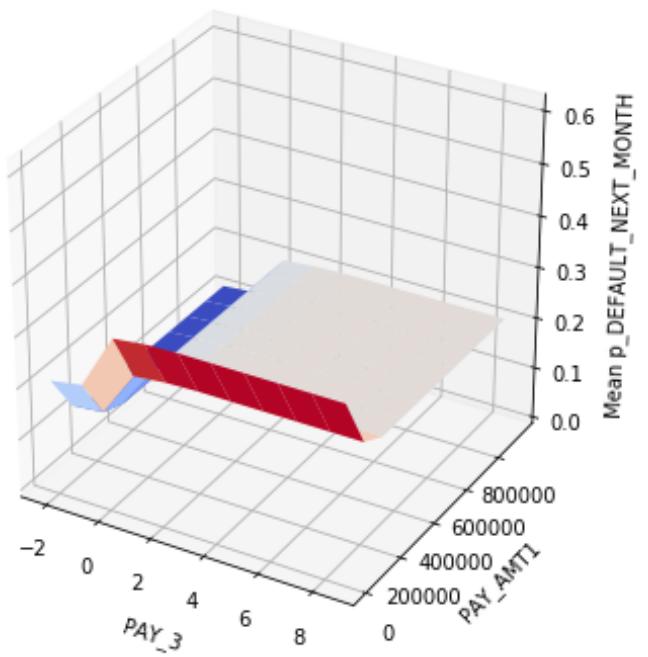
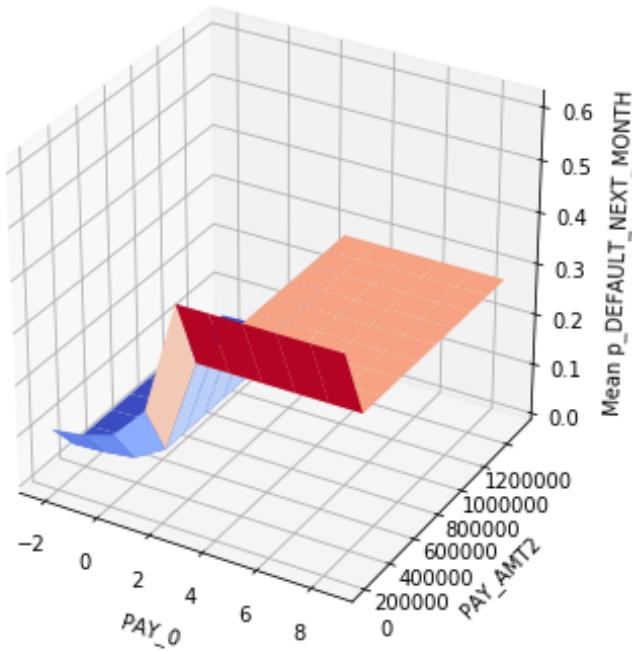
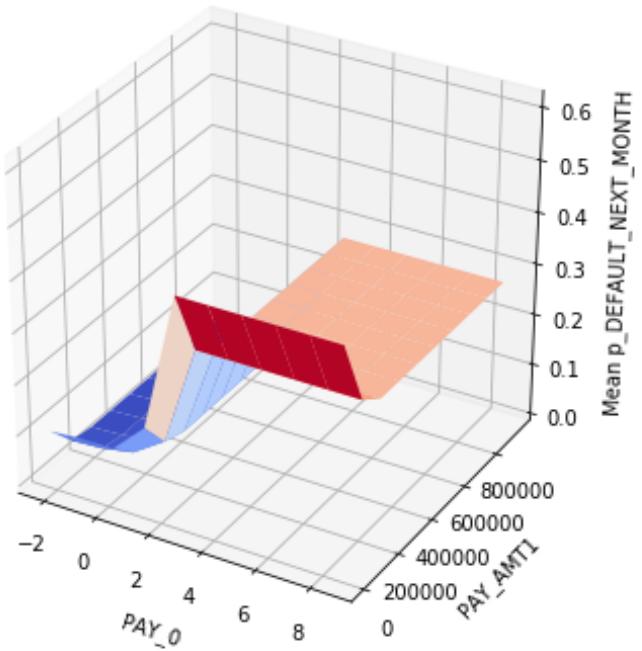
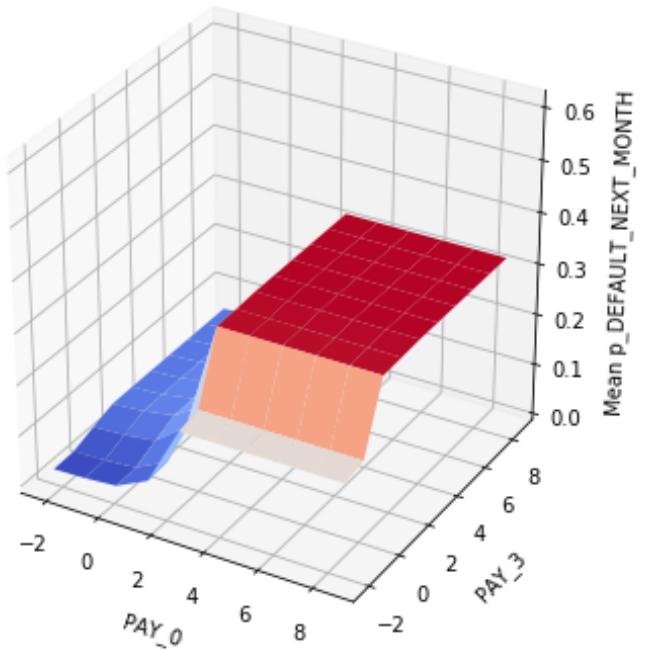
Error Metrics for PAY_0

	Prevalence	Accuracy	True Positive Rate	Precision	Specificity	Negative Predicted Value	False Positive Rate	False Discovery Rate	False Negative Rate	False Omissions Rate
PAY_0										
-2	0.049	0.857	0.3	0.119	0.885	0.961	0.115	0.881	0.7	0.039
-1	0.117	0.805	0.383	0.267	0.861	0.913	0.139	0.733	0.617	0.087
0	0.05	0.864	0.345	0.143	0.891	0.963	0.109	0.857	0.655	0.037
1	0.822	0.457	0.368	0.93	0.871	0.229	0.129	0.07	0.632	0.771
2	1	0.709	0.709	1	0.5	0	0.5	0	0.291	1
3	1	0.748	0.748	1	0.5	0	0.5	0	0.252	1
4	1	0.571	0.571	1	0.5	0	0.5	0	0.429	1
5	1	0.444	0.444	1	0.5	0	0.5	0	0.556	1
6	1	0.25	0.25	1	0.5	0	0.5	0	0.75	1
7	1	0.5	0.5	1	0.5	0	0.5	0	0.5	1
8	1	0.75	0.75	1	0.5	0	0.5	0	0.25	1

False Negative Rate Disparity by Education Level

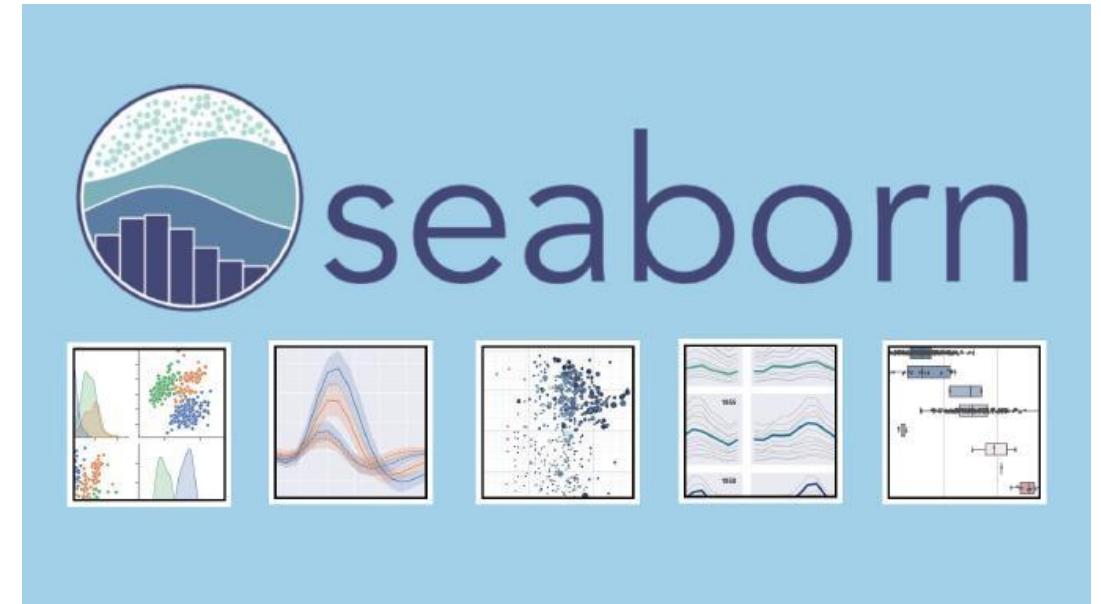


Mean p_DEFAULT_NEXT_MONTH for ['PAY_0', 'PAY_3', 'PAY_AMT1', 'PAY_AMT2']





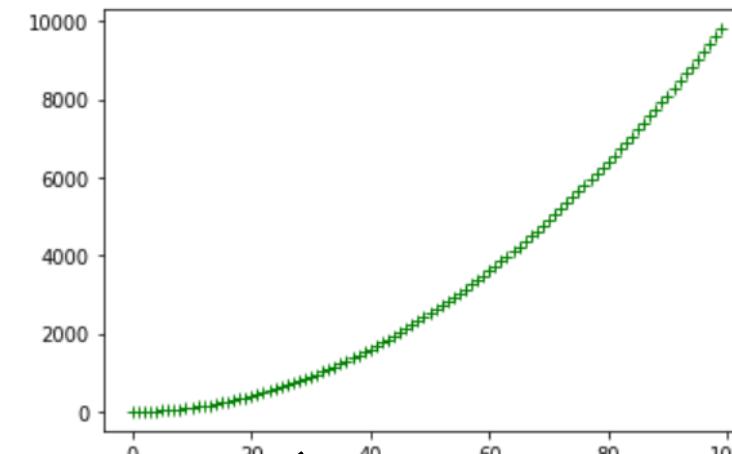
Version 3.2.1



데이터 시각화 - 간단한 그래프 그리기

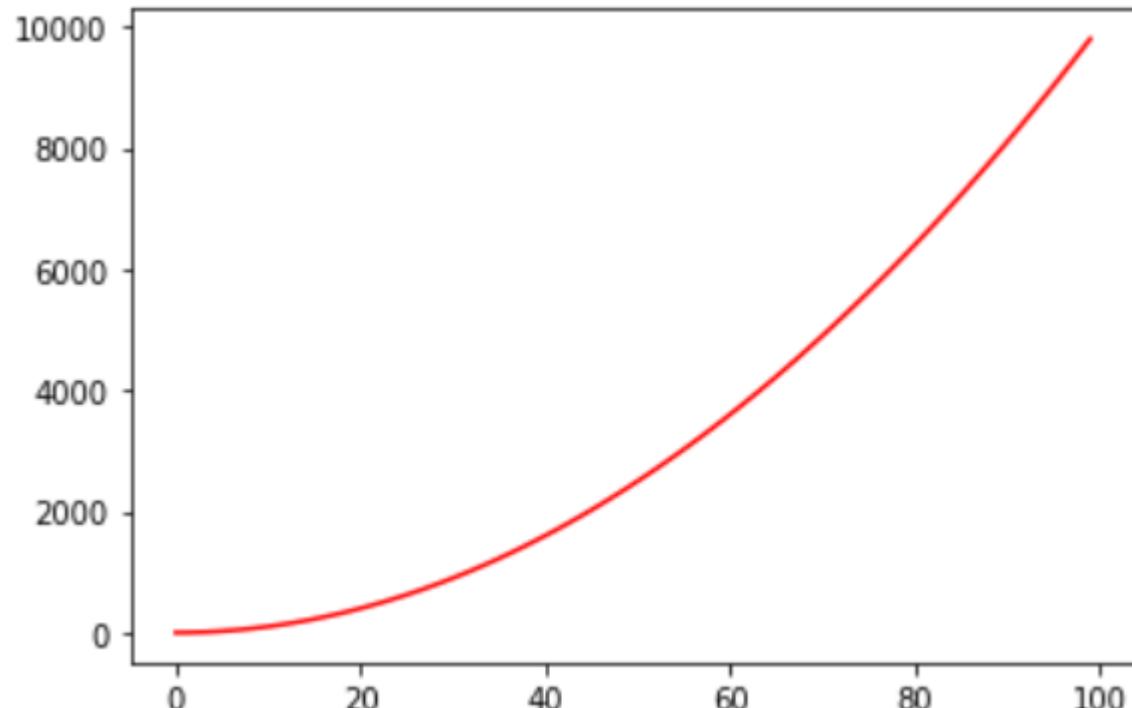
- matplotlib.pyplot 모듈
- plot() 함수 : 값을 서로 연결하여 라인 형태의 그래프 그림
 - 기본 포맷 : 파란색 선
 - 'r' : 빨간색 선, plt.plot(x, y, 'r')
 - 'ro' : 빨간색 원 마커 모양, plt.plot(x, y, 'ro')
 - 'r--' : 빨간색 대쉬라인, plt.plot(x, y, 'r--')
 - 'bs' : 파란색 사각형, plt.plot(x, y, 'bs')
 - 'g+' : 녹색 십자모양, plt.plot(x, y, 'g+')

```
1 import matplotlib.pyplot as plt
2 %matplotlib inline
3
4 x = [i for i in range(100)]
5 y = [i**2 for i in x]
6
7 plt.plot(x, y, 'g+')
```

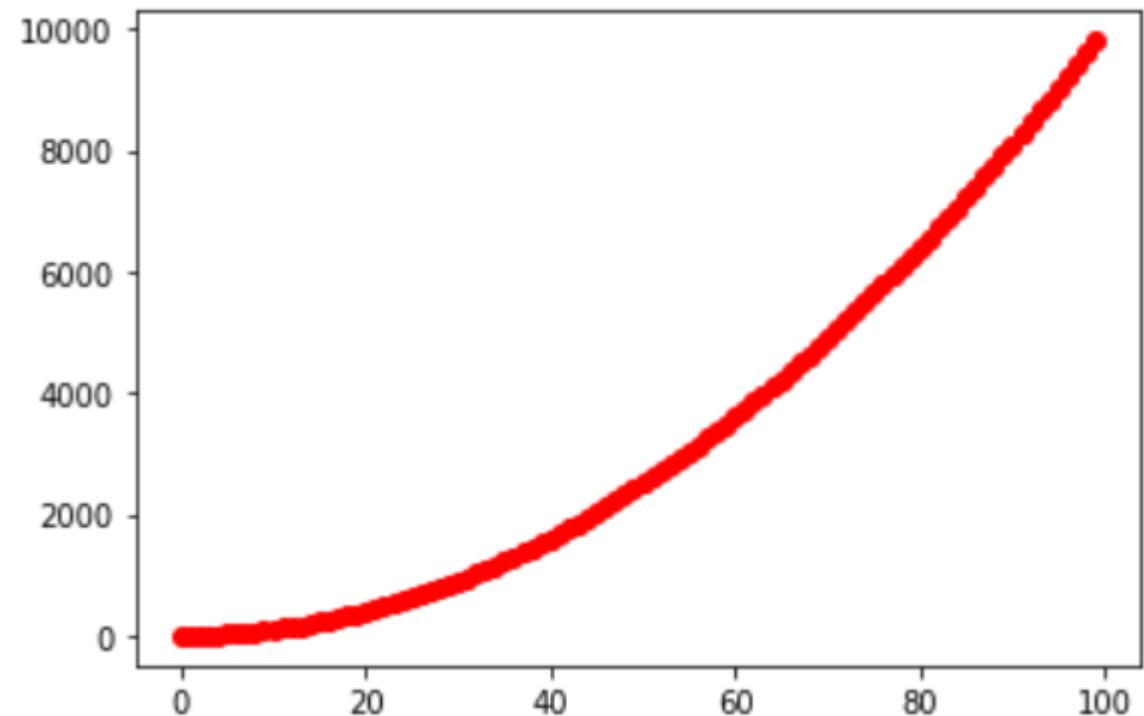


데이터 시각화 - 간단한 그래프 그리기

```
1 plt.plot(x, y, 'r')
```



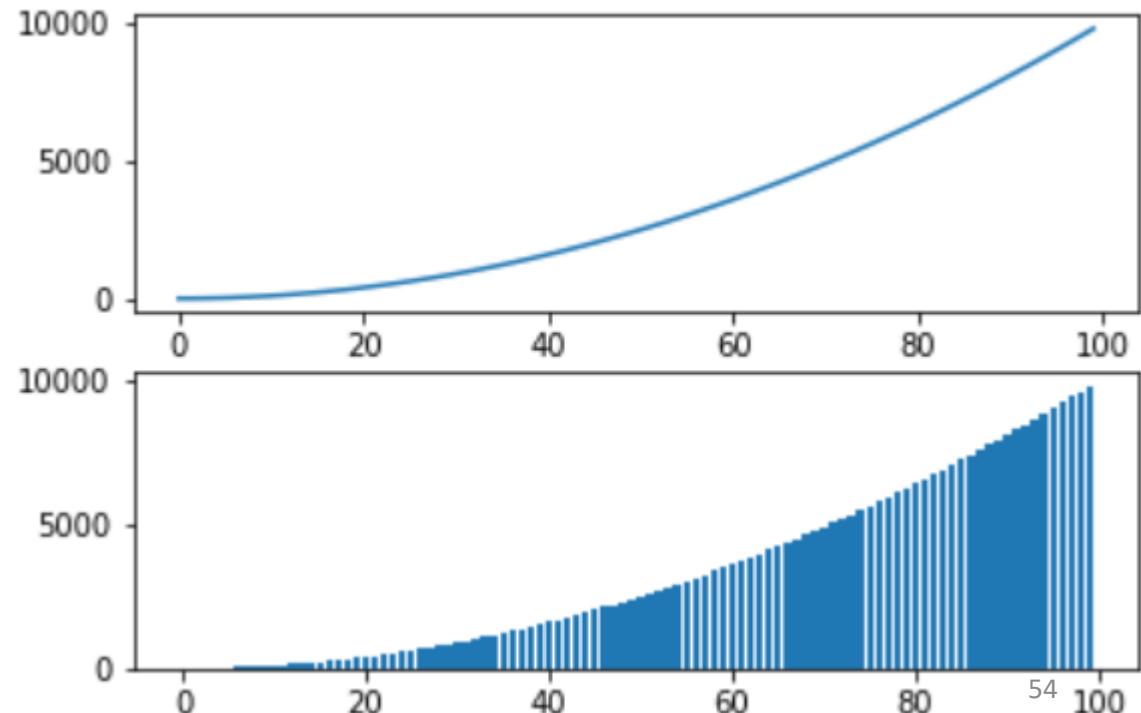
```
1 plt.plot(x, y, 'ro')
```



데이터 시각화 - 한 화면에 여러 개 그래프

- figure() : figure 객체 생성
- add_subplot(위치 및 개수) : 생성된 figure 객체에 subplot 추가
 - add_subplot(2, 1, 1) : 2x1 형태의 subplot, 두 개의 subplot 중 첫 번째
 - add_subplot(2, 1, 2) : 2x1 형태의 subplot, 두 개의 subplot 중 두 번째
- add_subplot() 함수 호출 시 AxesSubplot 객체 생성됨
- 생성된 subplot 객체를 그래프 drawing 함수와 연결

```
1 import matplotlib.pyplot as plt
2 %matplotlib inline
3
4 x = [i for i in range(100)]
5 y = [i**2 for i in x]
6
7 fig = plt.figure() # instance a figure object
8 axe_01 = fig.add_subplot(2,1,1) # 1st subplot in 2x1
9 axe_02 = fig.add_subplot(2,1,2) # 2nd subplot in 2x2
10
11 axe_01.plot(x, y)           # plot a line graph
12 axe_02.bar(x, y)           # plot a bar graph
```

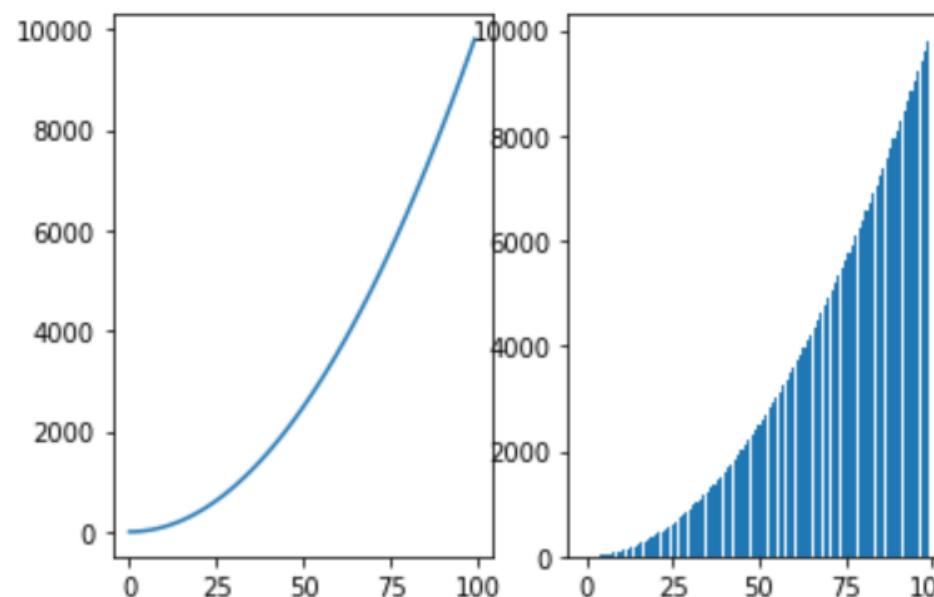


데이터 시각화 - 한 화면에 여러 개 그래프

- 1x2 형태 그래프들
 - add_subplot(1, 2, 1) : 1x2 형태의 subplot, 두 개의 subplot 중 첫 번째
 - add_subplot(1, 2, 2) : 1x2 형태의 subplot, 두 개의 subplot 중 두 번째

```
1 fig = plt.figure() # instance a figure object
2 axe_01 = fig.add_subplot(1,2,1) # 1st subplot in 2x1
3 axe_02 = fig.add_subplot(1,2,2) # 2nd subplot in 2x2
4
5 axe_01.plot(x, y)           # plot a line graph
6 axe_02.bar(x, y)           # plot a bar graph
```

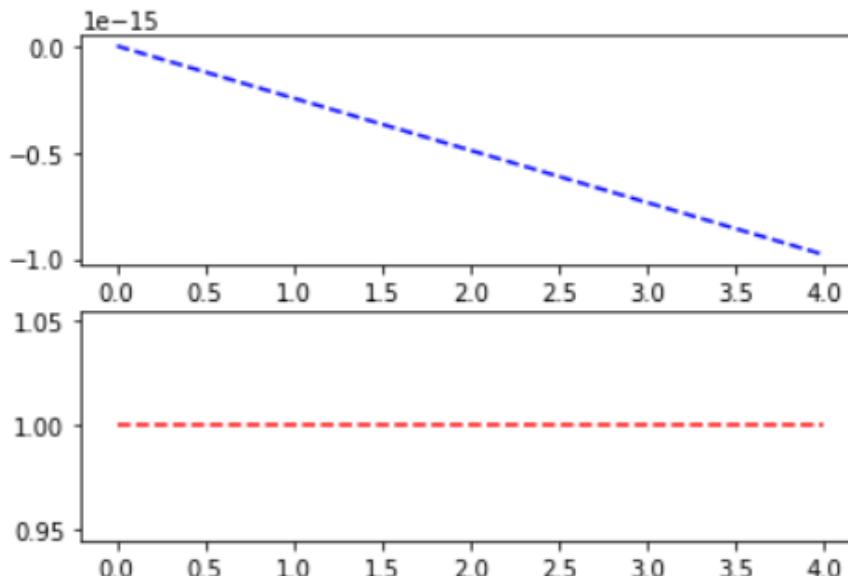
<BarContainer object of 100 artists>



데이터 시각화 - sine, cosine 그래프

```
1 import math  
2  
3 x = [i for i in range(5)]  
4 sin_y = [math.sin(2*math.pi*i) for i in x]  
5 cos_y = [math.cos(2*math.pi*i) for i in x]  
6  
7 fig = plt.figure() # instance a figure object  
8 axe_01 = fig.add_subplot(2,1,1) # 1st subplot in 2x1  
9 axe_02 = fig.add_subplot(2,1,2) # 2nd subplot in 2x2  
10  
11 axe_01.plot(x, sin_y, 'b--') # plot a sin graph  
12 axe_02.plot(x, cos_y, 'r--') # plot a cos graph
```

```
[<matplotlib.lines.Line2D at 0x1531b92aa88>]
```



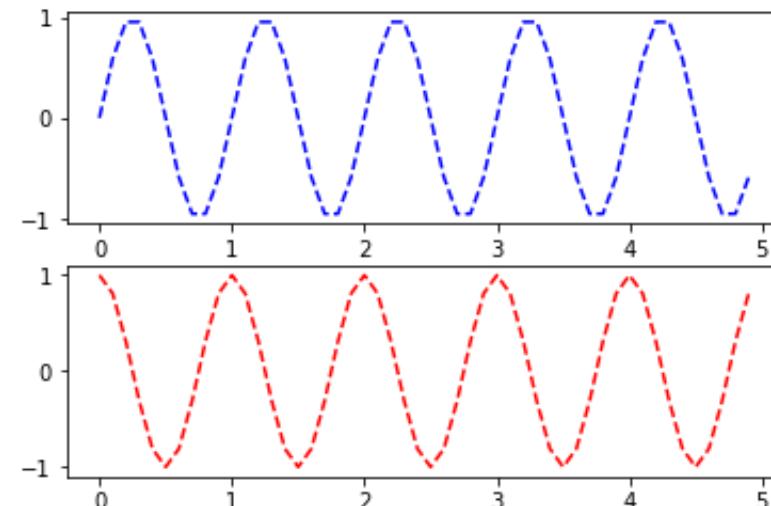
- 파이썬의 기본 math 모듈 사용

정수 0~5 사이 범위의 값에서 range(0,5)로는 시간의 값을 촘촘하게 만들 수 없음

데이터 시각화 - numpy 모듈

- numpy : Numerical Python의 약자. 행렬 연산이나 수치 계산에 자주 사용
- numpy.arange() 함수 : 실수 단위로도 step 값 가능
 - arange(0.0, 5.0, 0.1) : 0.0 ~ 5.0 사이에서 0.1 간격으로 값 생성
- numpy.pi : π 값

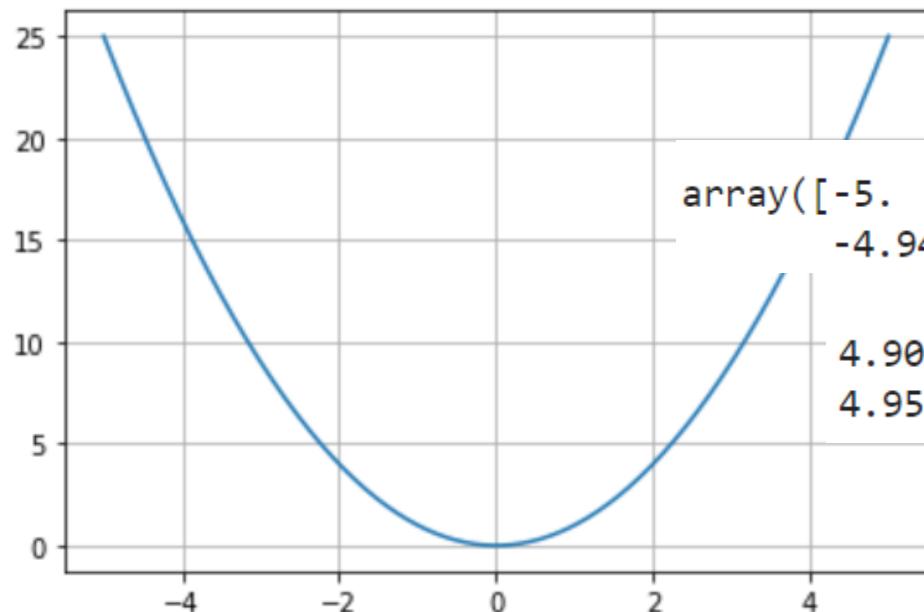
```
1 import numpy as np
2
3 x = np.arange(0., 5., 0.1)
4 sin_y = [np.sin(2*np.pi*i) for i in x]
5 cos_y = [np.cos(2*np.pi*i) for i in x]
6
7 fig = plt.figure() # instance a figure object
8 axe_01 = fig.add_subplot(2,1,1) # 1st subplot in 2x1
9 axe_02 = fig.add_subplot(2,1,2) # 2nd subplot in 2x2
10
11 axe_01.plot(x, sin_y, 'b--')    # plot a sin graph
12 axe_02.plot(x, cos_y, 'r--')    # plot a cos graph
```



데이터 시각화 - numpy 기반 값 생성

- `numpy.linspace(start, end, num-points)` : 시작점과 끝점을 균일 간격으로 나눈 값 생성
- `numpy.power(x, y)` : x의 y제곱 값

```
1 x = np.linspace(-5, 5, 1_000) # ndarray
2 y = np.power(x, 2)             # ndarray
3
4 plt.plot(x, y)   # plot a line with x, y
5 plt.grid()       # show a grid
6 plt.show()        # show a whole graph
```



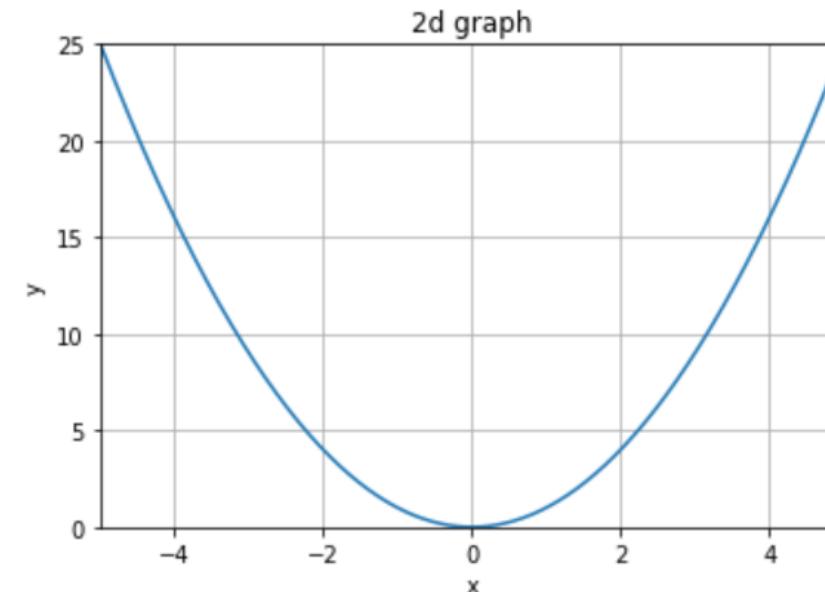
array([-5.0, -4.98998999, -4.97997998, -4.96996997, -4.95995996,
 -4.94994995, -4.93993994, -4.92992993, -4.91991992, -4.90990991,
 4.90990991, 4.91991992, 4.92992993, 4.93993994, 4.94994995,
 4.95995996, 4.96996997, 4.97997998, 4.98998999, 5.0])



데이터 시각화 - 그래프 라벨 및 범례 표시

- title(문자열) : 그래프 제목
- xlabel(문자열) : x축 라벨
- ylabel(문자열) : y축 라벨
- xlim((시작값, 끝값)) : x축 값 범위
- ylim((시작값, 끝값)) : y축 값 범위

```
1 x = np.linspace(-5, 5, 1_000) # ndarray
2 y = np.power(x, 2)             # ndarray
3
4 plt.plot(x, y)    # plot a Line with x, y
5 plt.title("2d graph")
6 plt.xlabel("x")   # x axe Label
7 plt.ylabel('y')   # y axe Label
8 plt.xlim((-5,5)) # range tuple
9 plt.ylim((0,25)) # range tuple
10 plt.grid()       # show a grid
11 plt.show()       # show a whole graph
```

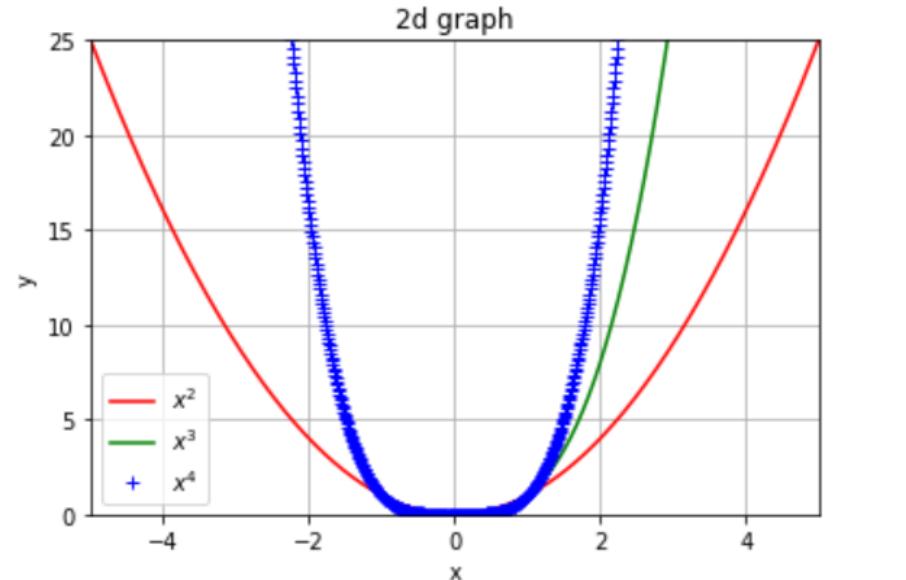


데이터 시각화 - multiple lines

- legend() : 범례 추가, loc 파라미터 = 표시 위치

```
1 x = np.linspace(-5, 5, 1_000) # ndarray
2
3 plt.plot(x, np.power(x, 2), "r-", label="$x^2$")
4 plt.plot(x, np.power(x, 3), "g-", label="$x^3$")
5 plt.plot(x, np.power(x, 4), "b+", label="$x^4$")
6
7 plt.title("2d graph")
8 plt.xlabel("x") # x axe label
9 plt.ylabel('y') # y axe label
10 plt.xlim((-5,5)) # range tuple
11 plt.ylim((0,25)) # range tuple
12 plt.grid() # show a grid
13 plt.legend(loc="best")
14 plt.show() # show a whole graph
```

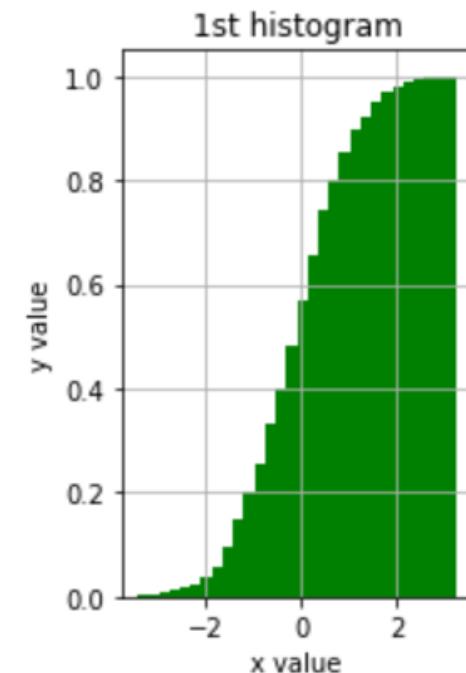
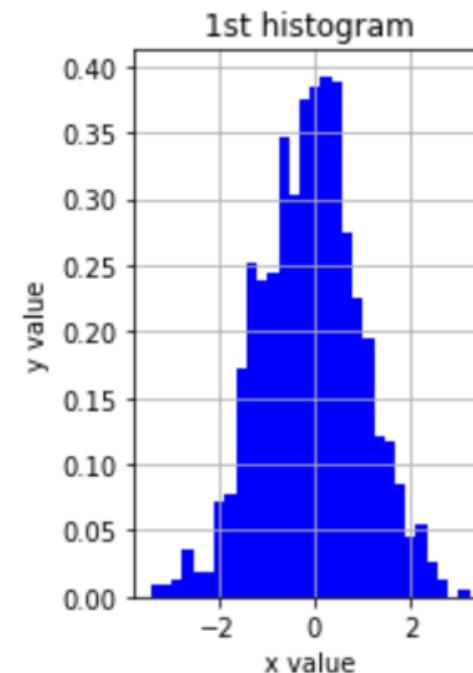
Location String	Location Code
'best'	0
'upper right'	1
'upper left'	2
'lower left'	3
'lower right'	4
'right'	5
'center left'	6
'center right'	7
'lower center'	8
'upper center'	9



데이터 시각화 - histogram

- numpy.random 모듈 randn() 함수 : 임의의 표준정규분포 데이터 생성
- matplotlib 모듈 hist() 함수 : 히스토그램 그리기
 - bin=나누는 구간
 - cumulative=누적 옵션
- grid(True) : 그리드 배경
- savefig("파일명") 파일로 저장

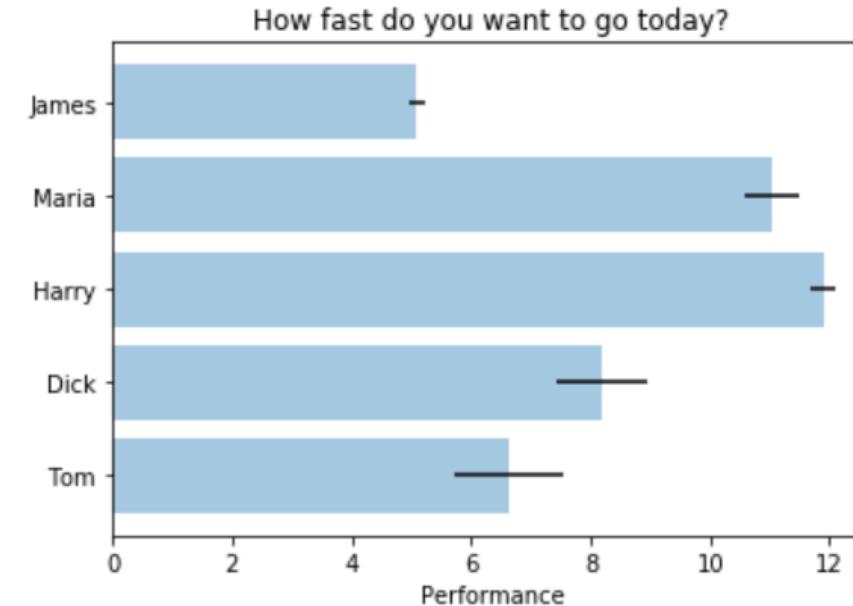
```
1 data = np.random.randn(1_000)
2
3 fig    = plt.figure()          # instance a figure object
4 axe_01 = fig.add_subplot(121) # 1st subplot in 2x1
5 axe_02 = fig.add_subplot(122) # 2nd subplot in 2x2
6
7 axe_01.set_title("1st histogram")
8 axe_01.set_xlabel("x value")
9 axe_01.set_ylabel("y value")
10 axe_01.grid(True)
11 axe_01.hist(data, bins=30, density=True, color='b')
12
13 axe_02.set_title("1st histogram")
14 axe_02.set_xlabel("x value")
15 axe_02.set_ylabel("y value")
16 axe_02.grid(True)
17 axe_02.hist(data, bins=30, density=True, color='g', cumulative=True)
18
19 plt.subplots_adjust(wspace=.4)
20 plt.show()
```



데이터 시각화 - Horizontal bar

- barh() 함수 : 수평 차트 그리기
- yticks() 함수 : ticker 위치별 각 위치에서의 라벨 설정

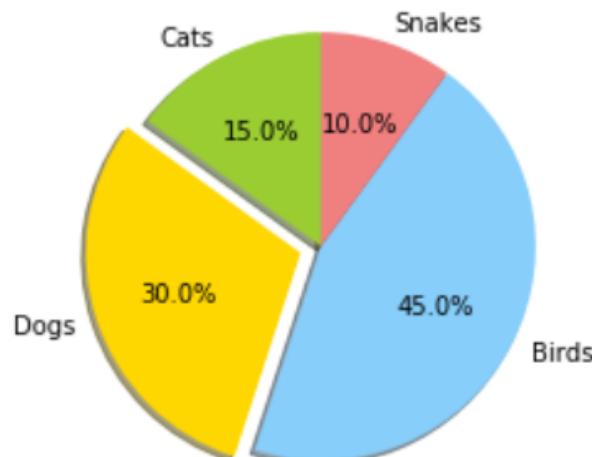
```
1 people = ["Tom", "Dick", "Harry", "Maria", "James"]
2
3 y_position = np.arange(len(people))
4 performance = 3 + 10 * np.random.rand(len(people))
5 error = np.random.rand(len(people))
6
7 plt.barh(y_position, performance,
8           xerr=error, align='center', alpha=0.4)
9 plt.yticks(y_position, people)
10 plt.xlabel("Performance")
11 plt.title("How fast do you want to go today?")
12
13 plt.show()
```



데이터 시각화 - 원 그래프

- pie () 함수 : 원 그래프 그리기

```
1 # The slices will be ordered and plotted counter-clockwise.
2 labels = ["Cats", "Dogs", "Birds", "Snakes"]
3 sizes = [15, 30, 45, 10]
4 colors = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral']
5 explode = [0, 0.1, 0, 0] # only explode : 2nd slice(i.e. Dogs)
6
7 plt.pie(sizes, explode=explode, labels=labels, colors=colors,
8         autopct="%3.1f%%", shadow=True, startangle=90)
9
10 plt.show()
```



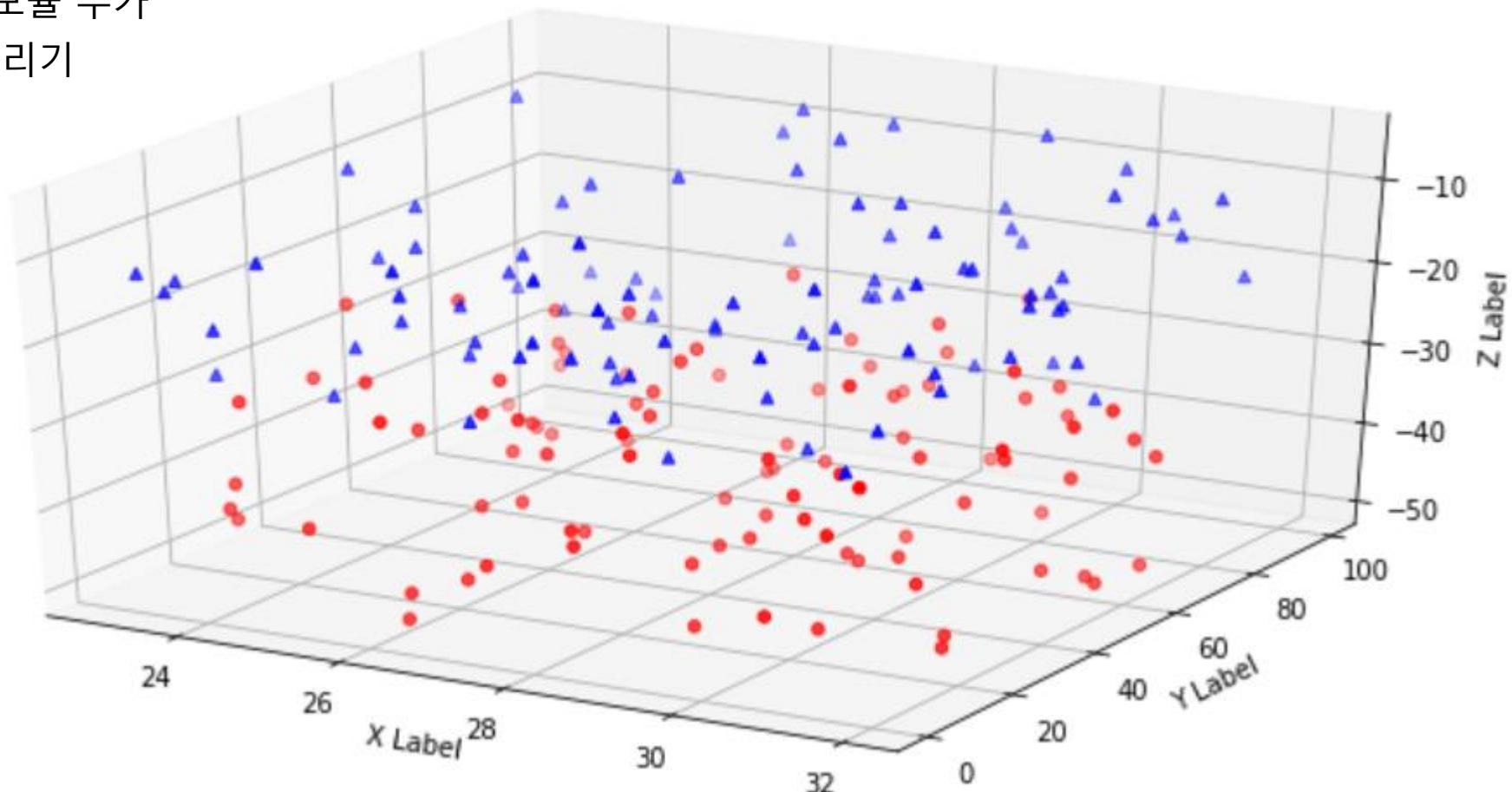
데이터 시각화 - 3D 그래프(2/1)

- mpl_toolkits.mplot3d 모듈 추가
- scatter() : 점 그래프 그리기

```
1 from mpl_toolkits.mplot3d import Axes3D
2
3 def randrange(n, min, max):
4     return (max-min)*np.random.rand(n)+min
5
6 n = 100
7 fig = plt.figure()
8 axe = fig.add_subplot(111, projection='3d')
9
10 for c, m, zl, zh in [('r', 'o', -50, -25), ('b', '^', -30, -5)]:
11     xs = randrange(n, 23, 32)
12     ys = randrange(n, 0, 100)
13     zs = randrange(n, zl, zh)
14     axe.scatter(xs, ys, zs, c=c, marker=m)
15
16 axe.set_xlabel('X Label')
17 axe.set_ylabel('Y Label')
18 axe.set_zlabel('Z Label')
19
20 plt.show()
```

데이터 시각화 - 3D 그래프(2/2)

- ◆ mpl_toolkits.mplot3d 모듈 추가
- ◆ scatter() : 점 그래프 그리기

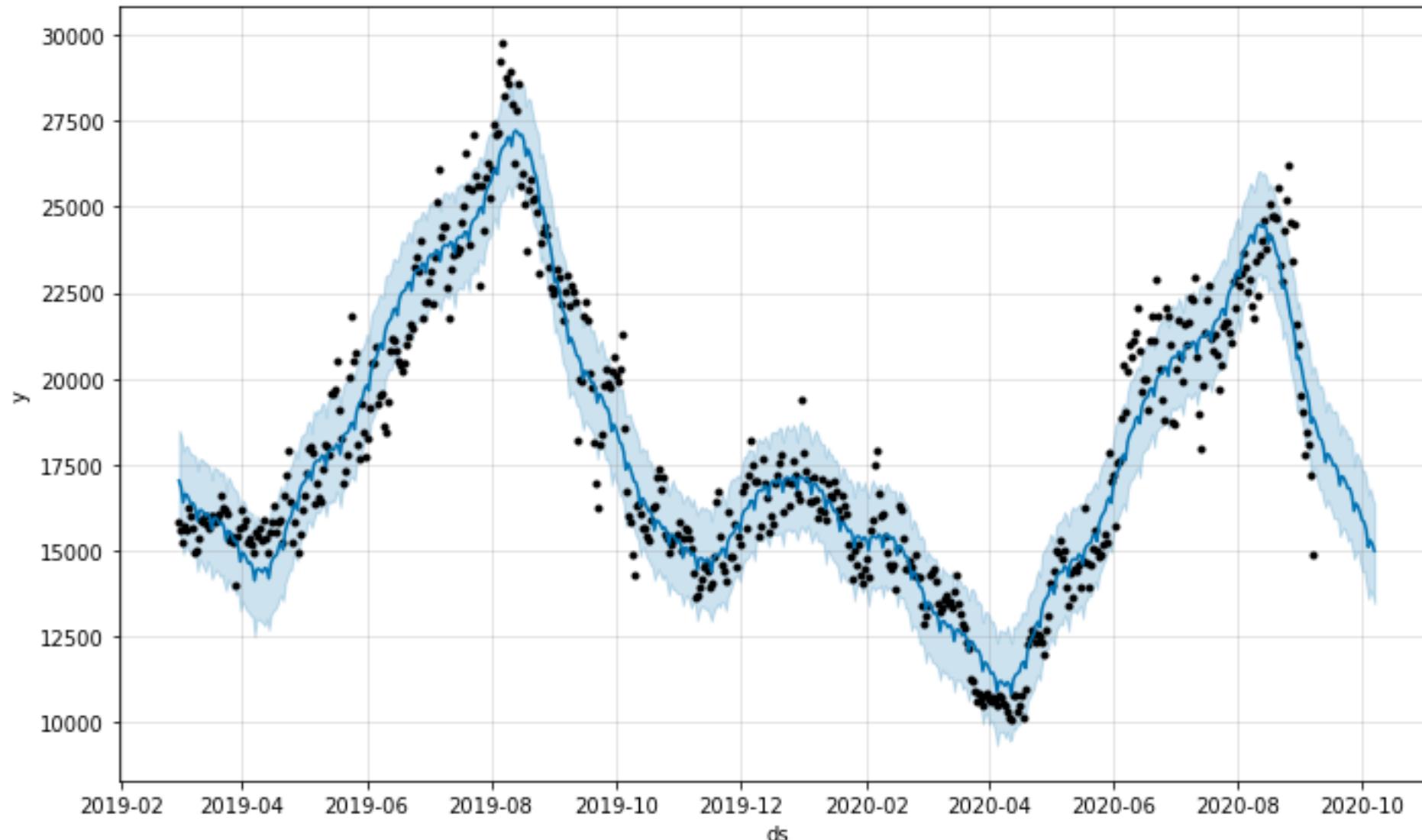


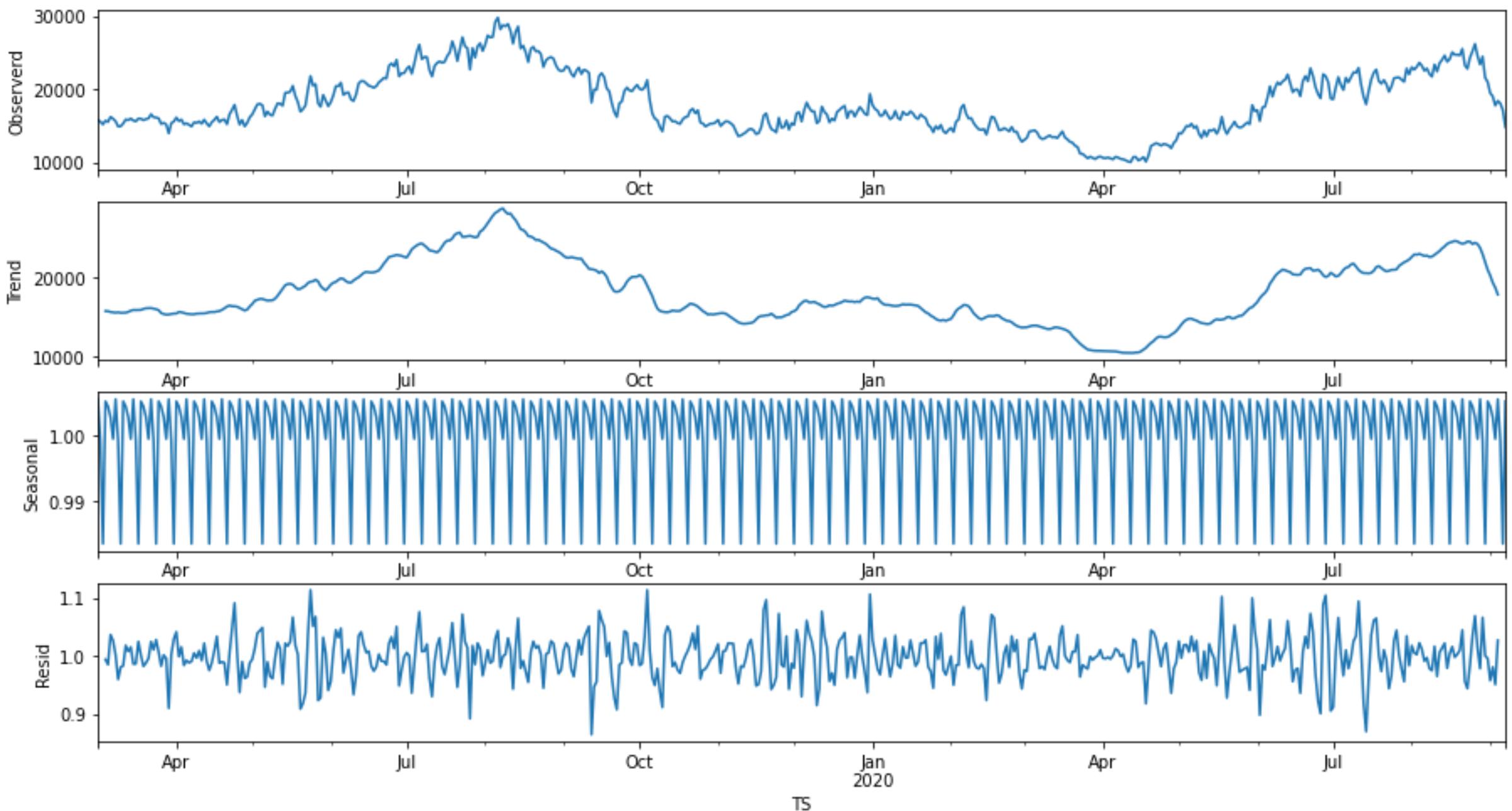


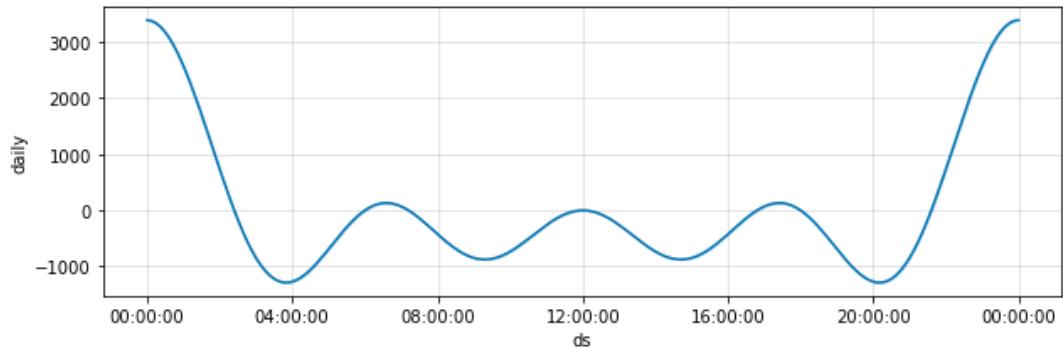
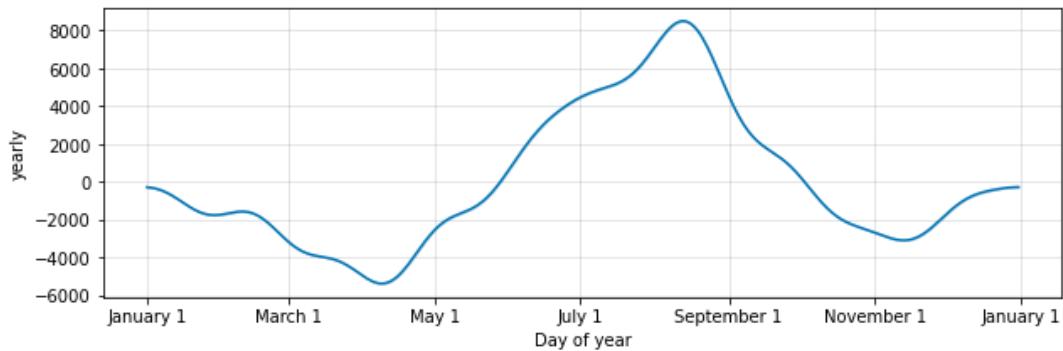
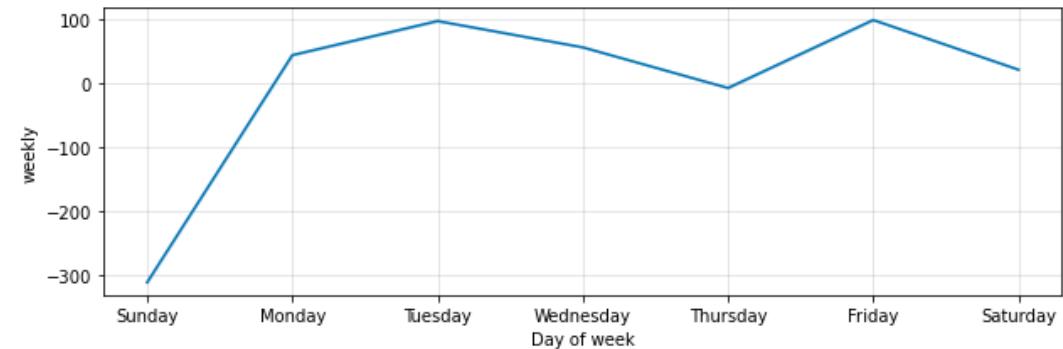
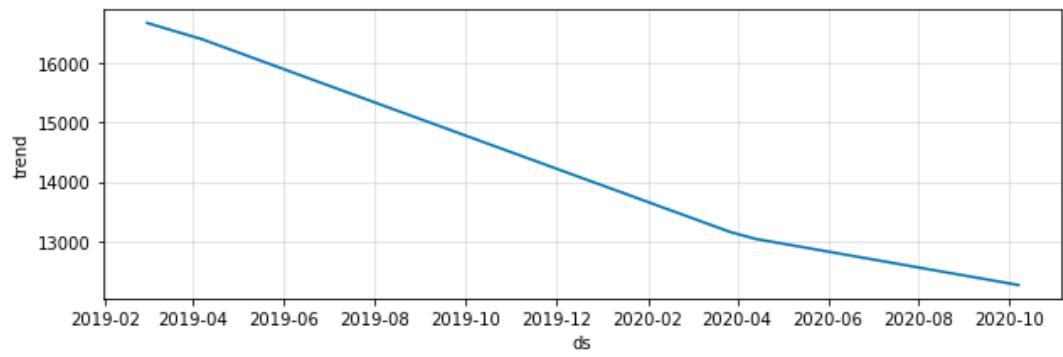
시계열 예측 분석

<https://github.com/JSJeong-me/KOSA-ML-BigData-Analysis/tree/main/prophet>

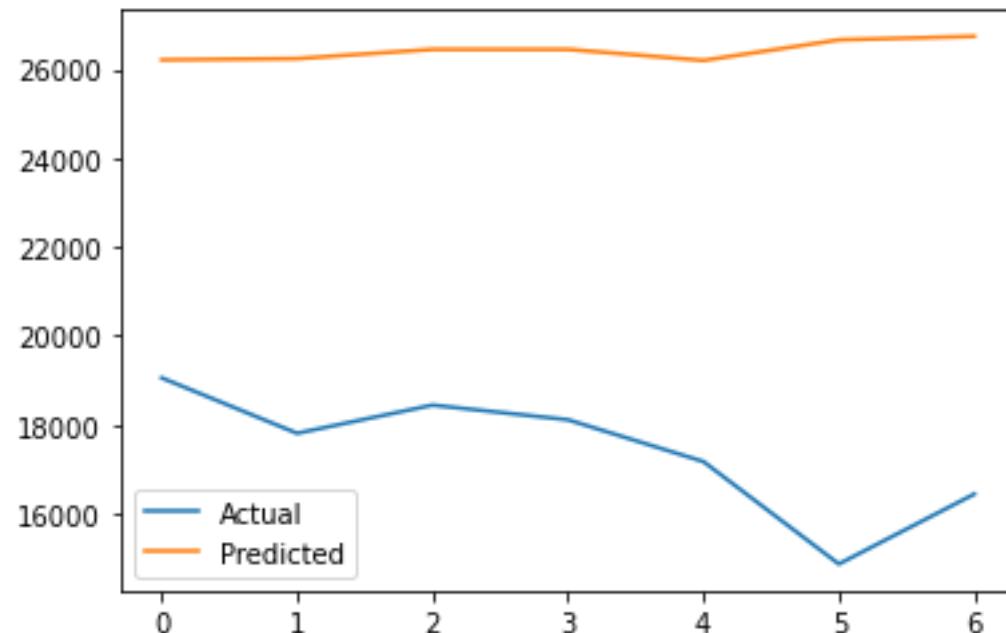
M 호텔 전력 수요 예측 분석



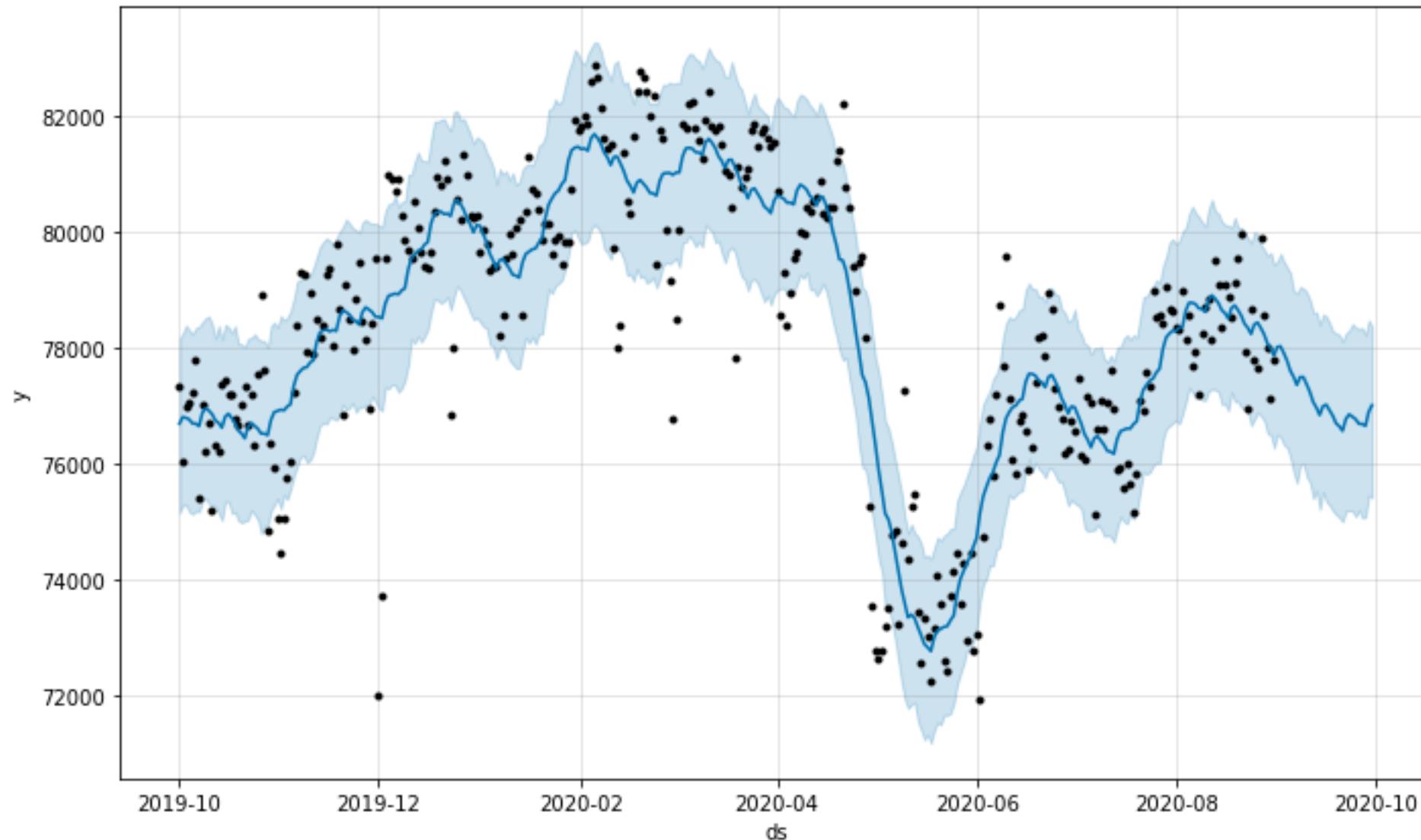


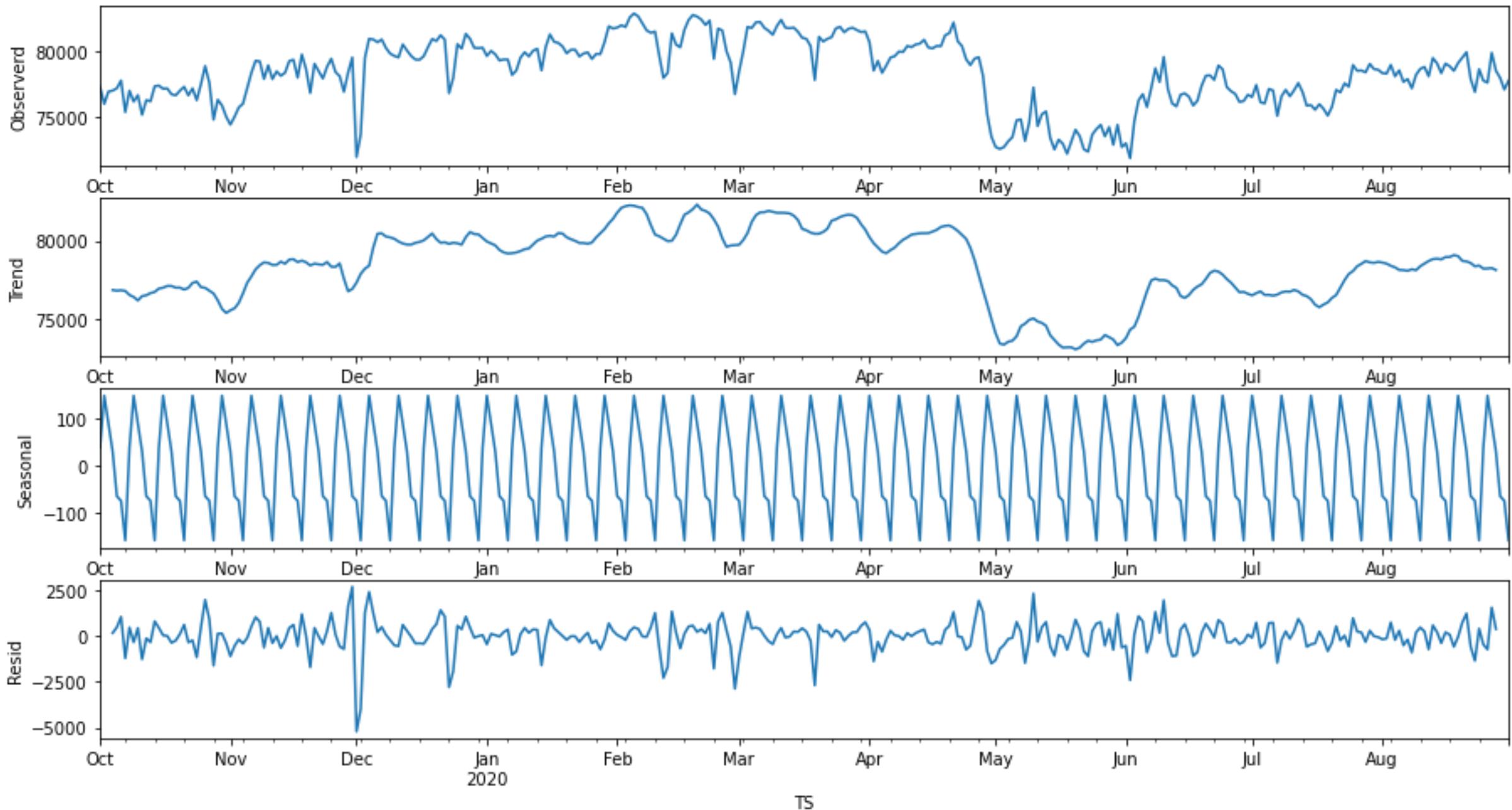


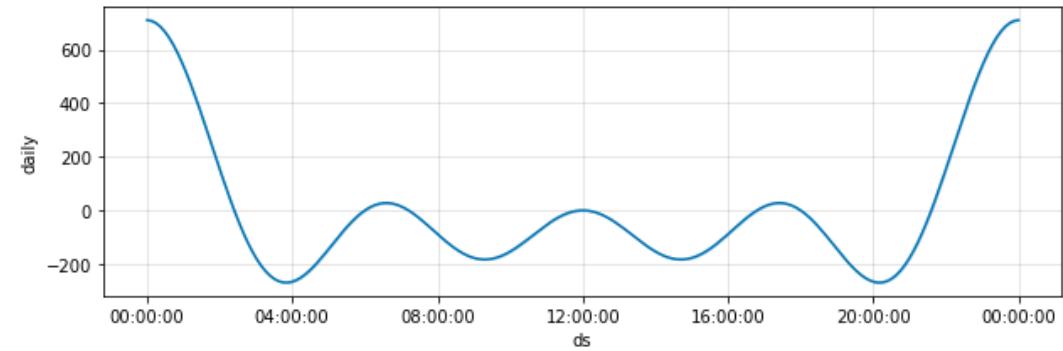
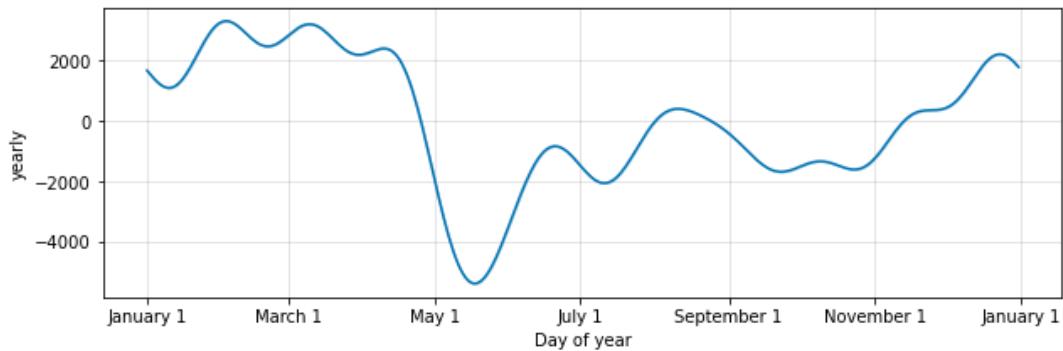
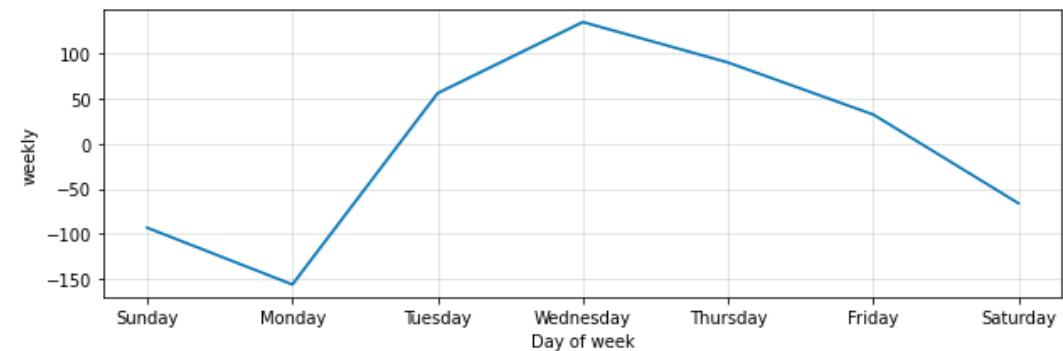
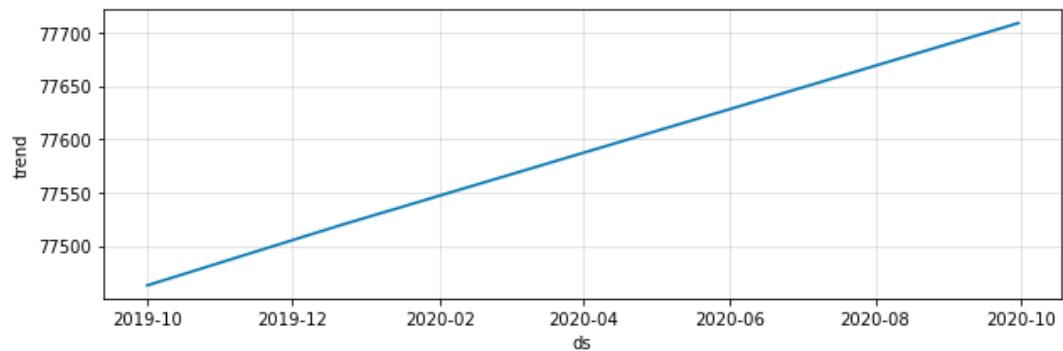
주(week) 전력 수요 예측 분석



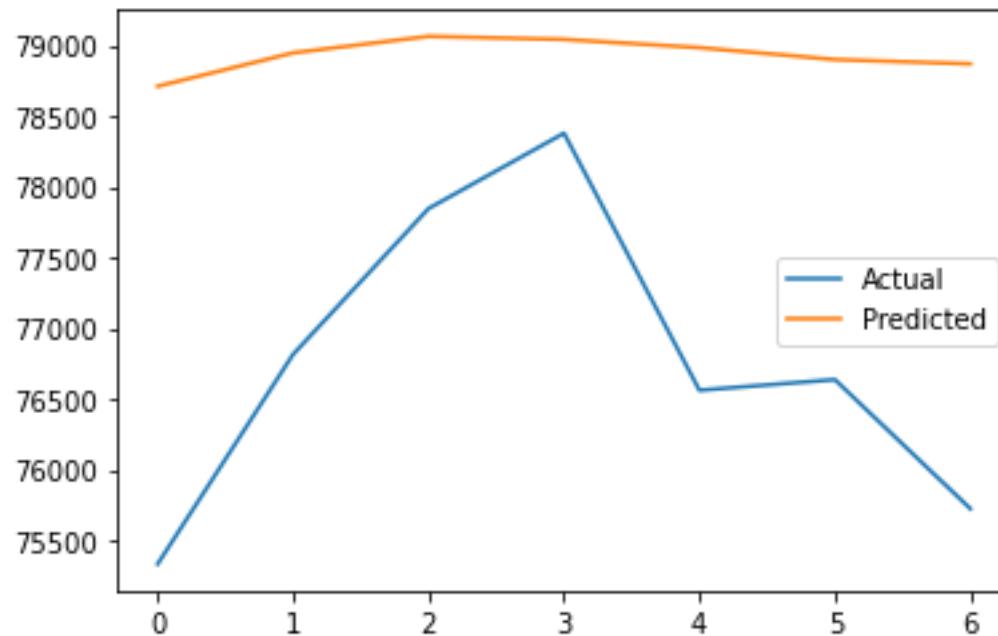
K 공장 전력 수요 예측 분석







주(week) 전력 수요 예측 분석



강사 소개



정 준 수 / Ph.D (heinem@naver.com)

- 前) 삼성전자 연구원
- 前) 삼성의료원 (삼성생명과학연구소)
- 前) 삼성SDS (정보기술연구소)
- 現) (사)한국인공지능협회, AI, 머신러닝 강의
- 現) 한국소프트웨어산업협회, AI, 머신러닝 강의
- 現) 서울디지털재단, AI 자문위원
- 現) 한성대학교 교수(겸)
- 전문분야: 시각 모델링, 머신러닝(ML), RPA
- <https://github.com/JSeong-me/>