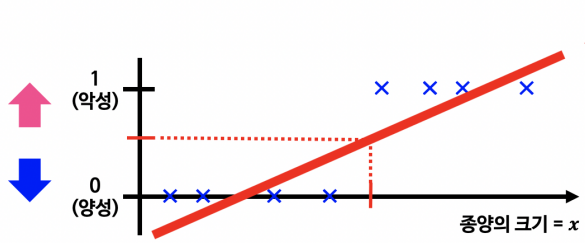
Logistic Regression

PinkLAB Edu

4 November, 2025

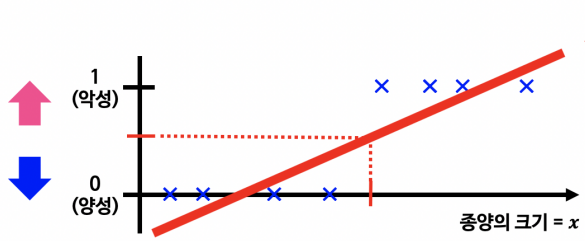
Table of Contents

1 분류?? 회귀?? 악성 종양을 찾는 문제



- $h_\theta(x)$가 0.5보다 크거나 같으면 1(악성)으로 예측
- $h_\theta(x)$가 0.5보다 작으면 0(양성)으로 예측

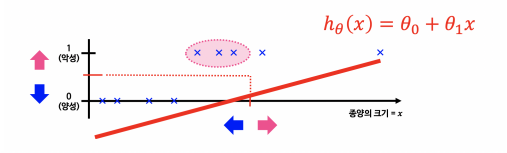
2 Linear Regression을 분류 문제에 적용할 수 있을까?

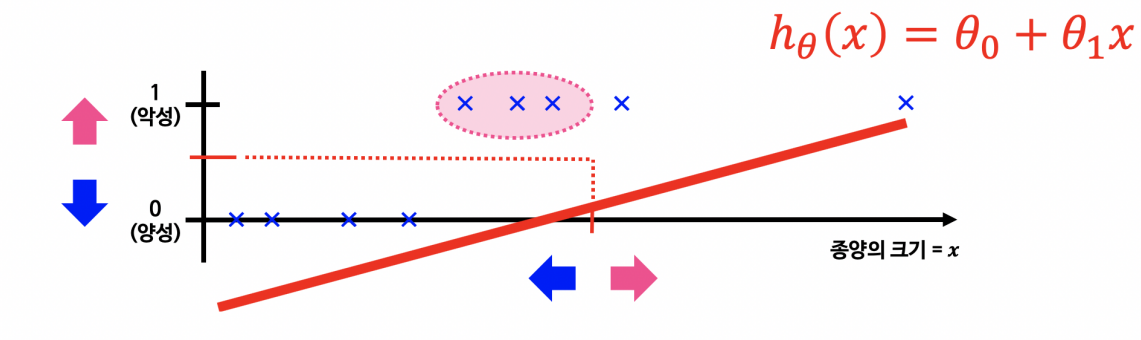


$$h_\theta(x) = \theta_0 + \theta_1 x$$

Linear Regression 문제 해결 방법을 그대로 적용해도 잘 동작할 듯 합니다 ...
맞나요?

3 Linear Regression으로는 분류하는 게 적용하기 힘들듯 하다





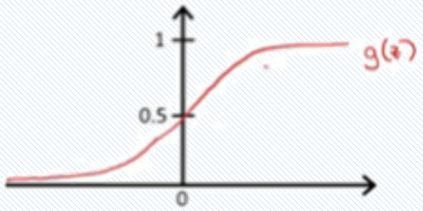
4 모델 재설정

분류 문제는 0 또는 1로 예측해야 하나 Linear Regression을 그대로 적용하면
예측값 $h_\theta(x)$는 **0보다 작거나 1보다 큰 값을 가질 수 있음**

$h_\theta(x)$가 항상 **0에서 1 사이의 값을 갖도록** Hypothesis 함수를 수정!!

$$h_\theta(x) = g(\theta_0 + \theta_1 x)$$

$$g(z) = \frac{1}{1+e^{-z}}$$



5 어떻게 생긴 걸까? Logistic Function - 간단히 확인하러 가기

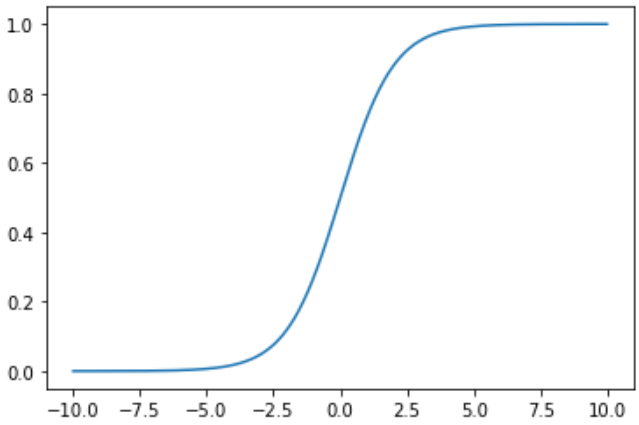
```python
import numpy as np

z = np.arange(-10 , 10 , 0.01)
g = 1 / (1 + np.exp(-z))
```

Python

6 Python 유저 답게 확인

```python
import matplotlib.pyplot as plt
%matplotlib inline


plt.plot(z, g);
```

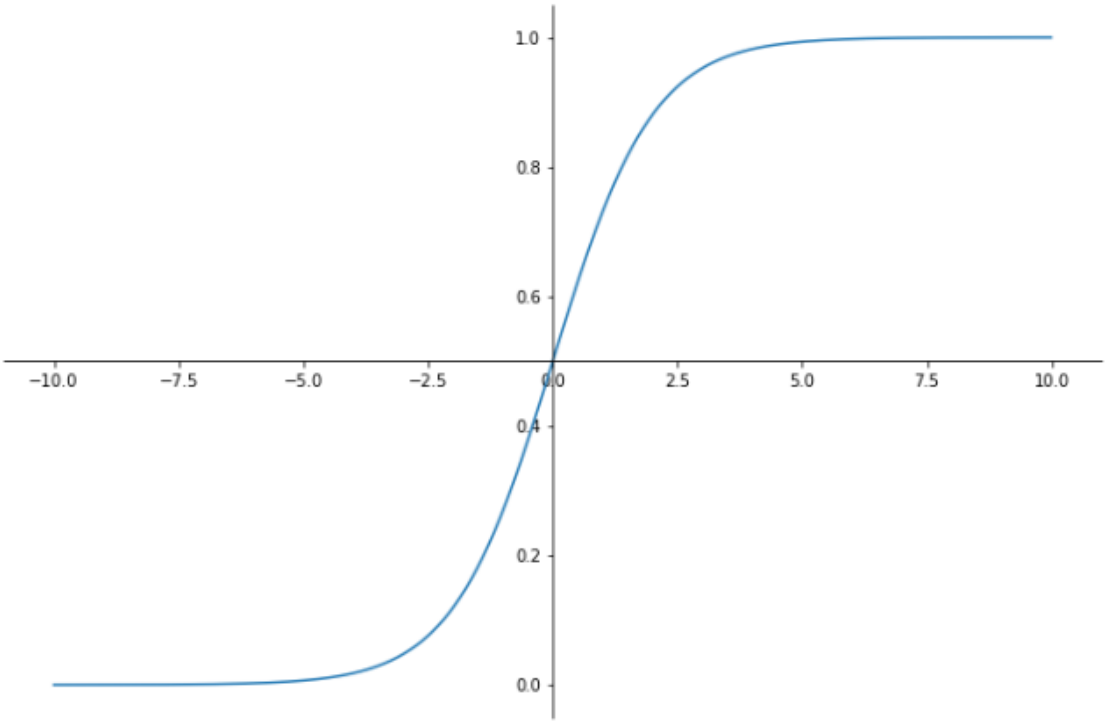
Python



7 그래프를 조금 다듬어 볼까요?

```python
plt.figure(figsize = (12, 8))
ax = plt.gca()

ax.plot(z, g)
ax.spines['left'].set_position('zero')
ax.spines['right'].set_color('none')
ax.spines['bottom'].set_position('center')
ax.spines['top'].set_color('none')


plt.show()
```

Python



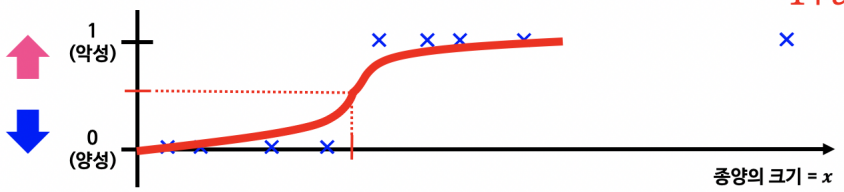
8 Hypothesis 함수의 결과에 따른 분류

$h_\theta(x)$는 주어진 입력 x(종양의 크기)에서의 **예측 결과가 1(악성)이 될 확률**을 의미함

→ $h_\theta(x)$가 0.7이라면... 환자의 종양이 악성일 확률이 70%임

9 분류 문제용 hypothesis

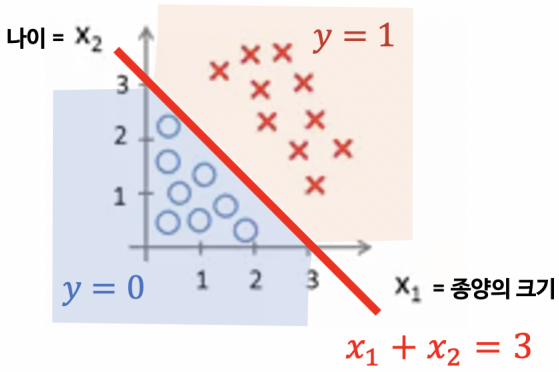
$$h_\theta(x) = g(\theta_0 + \theta_1 x) = \frac{1}{1+e^{-(\theta_0+\theta_1 x)}}$$



- $h_\theta(x)$가 0.5보다 크거나 같으면 1(악성)으로 예측
- $h_\theta(x)$가 0.5보다 작으면 0(양성)으로 예측

10 Decision Boundary

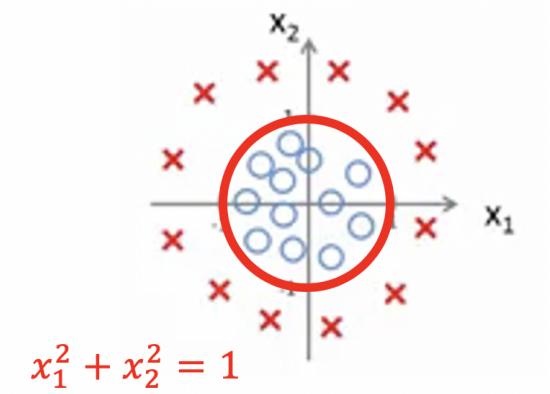
$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$



$-3 + x_1 + x_2 \geq 0$ 이면
$y = 1$로 예측 (악성)

$-3 + x_1 + x_2 < 0$ 이면
$y = 0$으로 예측 (양성)

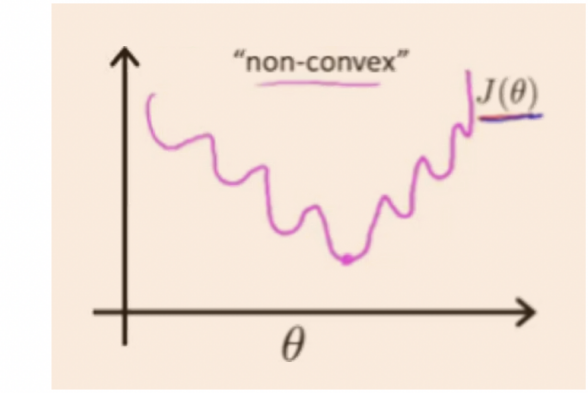
11 또다른 Decision Boundary

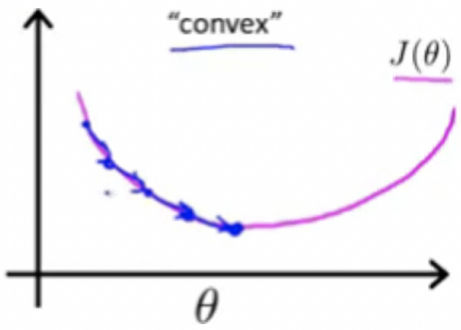


$-1 + x_1^2 + x_2^2 \geq 0$ 이면
$y = 1$로 예측 (악성)

$-1 + x_1^2 + x_2^2 < 0$ 이면
$y = 0$으로 예측 (양성)

12 Cost Function은 어떻게?





13 Logistic Regression에서 Cost Function을 재정의

$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} Cost\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right), y^{(i)}\right)$$

$$Cost(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) & y = 0 \end{cases}$$

$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} Cost\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right), y^{(i)}\right)$$

$$Cost(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) & y = 0 \end{cases}$$

14 Learning 알고리즘은 동일

수렴할 때까지 반복 {

$$\theta := \theta - \alpha \frac{d}{d\theta} J(\theta)$$

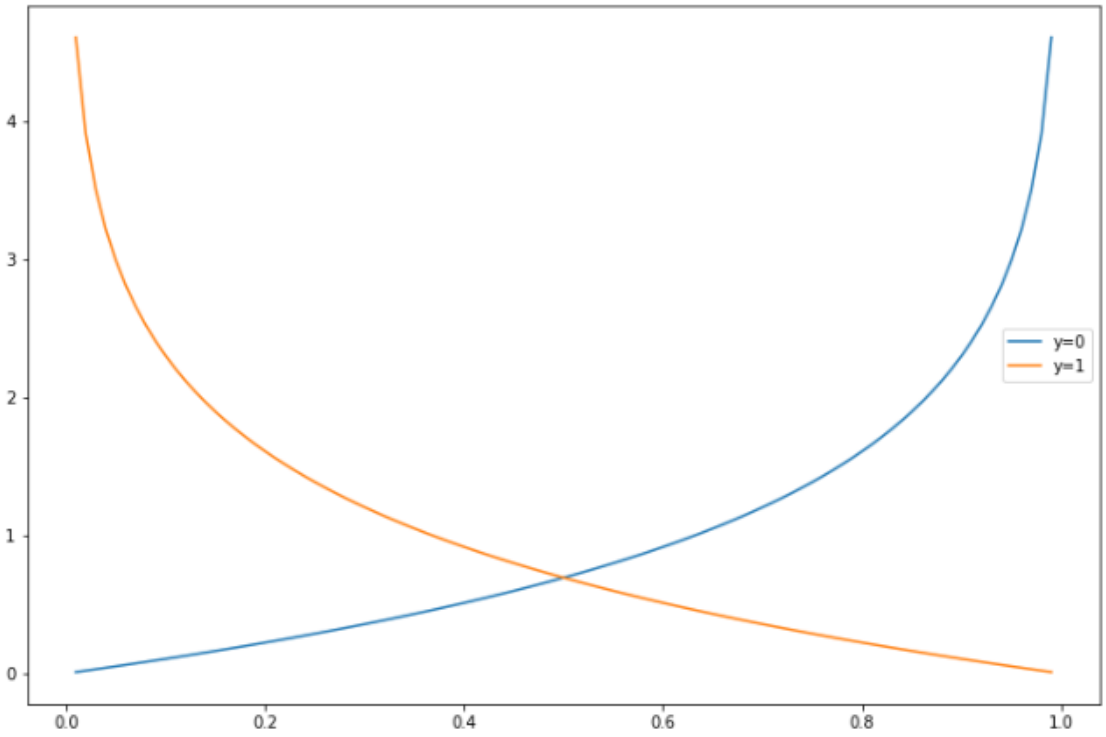
}

학습률
(Learning Rate)

15 Logistic Reg. Cost Fcn의 그래프

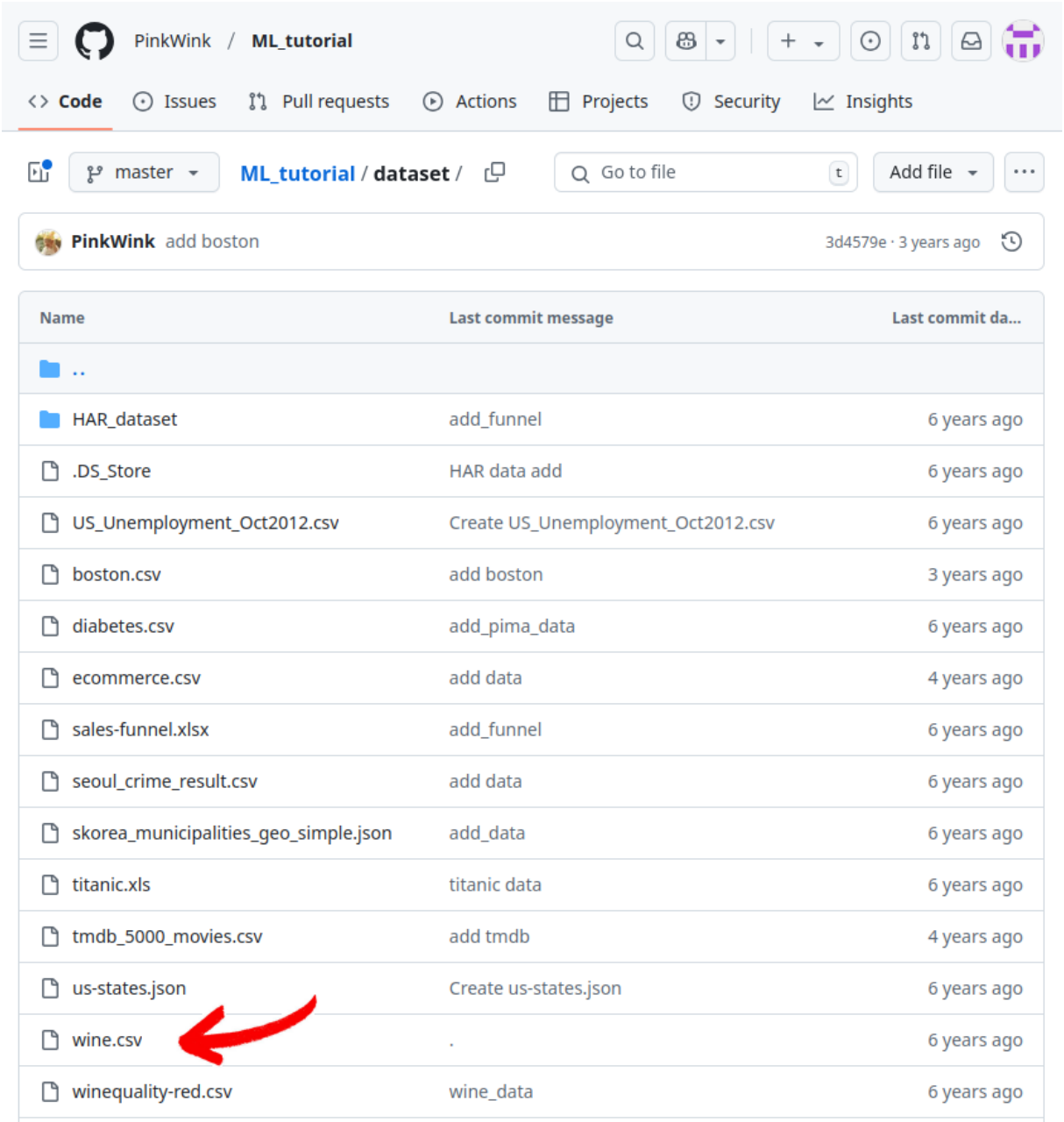
```python
h = np.arange(0.01, 1 , 0.01)

c0 = -np.log(1 - h)
c1 = -np.log(h)

plt.figure(figsize=(12, 8))
plt.plot(h, c0 , label = 'y=0')
plt.plot(h, c1, label = 'y=1')
plt.legend()

plt.show()
```

Python



16 직접 실습하기

16.1 일단 통합 와인데이터 찾기



16.2 데이터 받기

```
import pandas as pd

wine_url = 'https://raw.githubusercontent.com/PinkWink/ML_tutorial/'+\
                                                    'master/dataset/wine.csv'

wine = pd.read_csv(wine_url, index_col = 0)
wine.head()
```

✓ 0.6s Python

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	a
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	

16.3 맛 등급 만들어 넣기

```
wine['taste'] = [1. if grade > 5 else 0 for grade in wine['quality']]

X = wine.drop(['taste', 'quality'], axis = 1)
y = wine['taste']
```

✓ 0.0s Python

16.4 데이터 분리

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=13
)
```

✓ 0.3s Python

16.5 초 간단 로지스틱 회귀 테스트

```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

lr = LogisticRegression(solver="liblinear", random_state=13)
lr.fit(X_train, y_train)

y_pred_tr = lr.predict(X_train)
y_pred_test = lr.predict(X_test)

print("Train Acc : ", accuracy_score(y_train, y_pred_tr))
print("Test Acc : ", accuracy_score(y_test, y_pred_test))
```

✓ 0.0s Python

```
Train Acc :  0.7427361939580527
Test Acc :  0.7438461538461538
```

16.6 스케일러까지 적용해서 파이프라인 구축

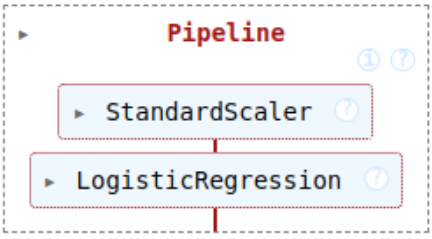
```python
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

estimators = [
    ("scaler", StandardScaler()),
    ("clf", LogisticRegression(solver="liblinear", random_state=13)),
]
pipe = Pipeline(estimators)
```

✓ 0.0s Python

16.7 fit~~~

```python
pipe.fit(X_train, y_train)
```

✓ 0.0s Python



16.8 상승효과가 있긴 하다

```python
y_pred_tr = pipe.predict(X_train)
y_pred_test = pipe.predict(X_test)

print("Train Acc : ", accuracy_score(y_train, y_pred_tr))
print("Test Acc : ", accuracy_score(y_test, y_pred_test))
```

✓ 0.0s Python

```
Train Acc :  0.7444679622859341
Test Acc :  0.7469230769230769
```

16.9 Decision Tree와의 비교를 위한 작업

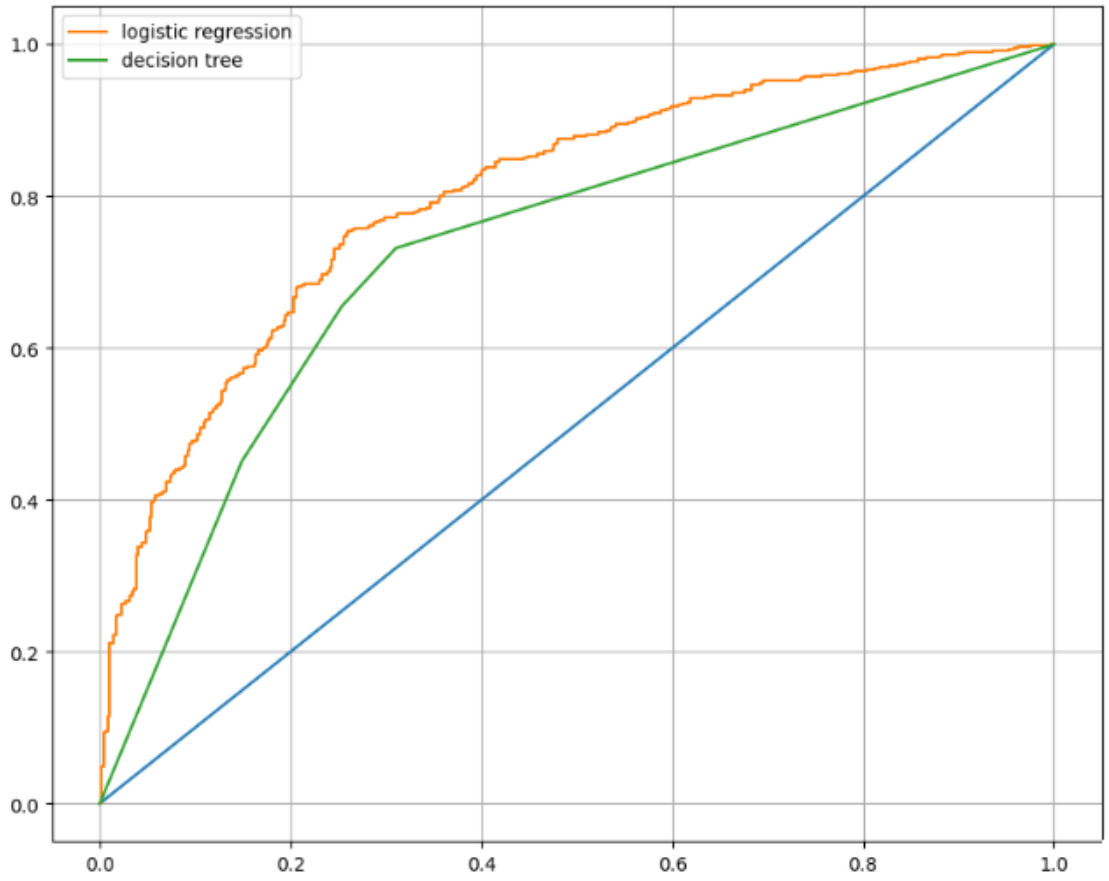
```python
from sklearn.tree import DecisionTreeClassifier

wine_tree = DecisionTreeClassifier(max_depth=2, random_state=13)
wine_tree.fit(X_train, y_train)

models = {"logistic regression": pipe, "decision tree": wine_tree}
```

✓ 0.0s Python

16.10 AUC 그래프를 이용한 모델간 비교

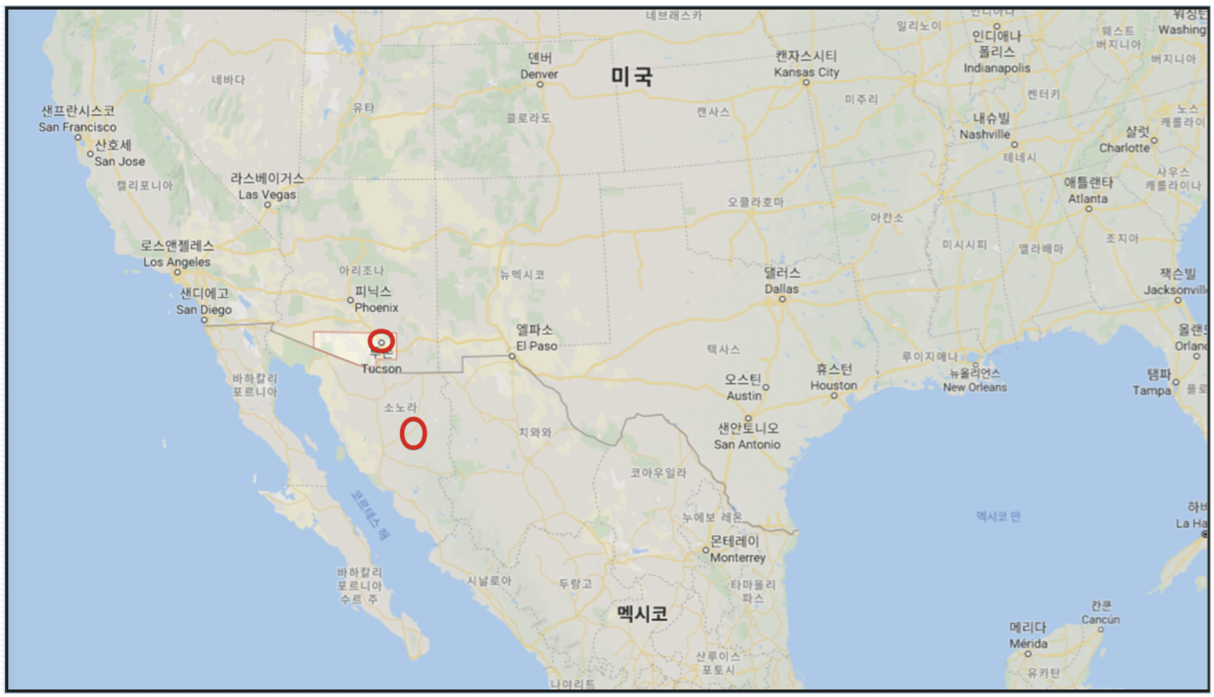
```python
from sklearn.metrics import roc_curve

plt.figure(figsize=(10, 8))
plt.plot([0, 1], [0, 1])
for model_name, model in models.items():
    pred = model.predict_proba(X_test)[:, 1]
    fpr, tpr, thresholds = roc_curve(y_test, pred)
    plt.plot(fpr, tpr, label=model_name)

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

✓ 0.0s Python

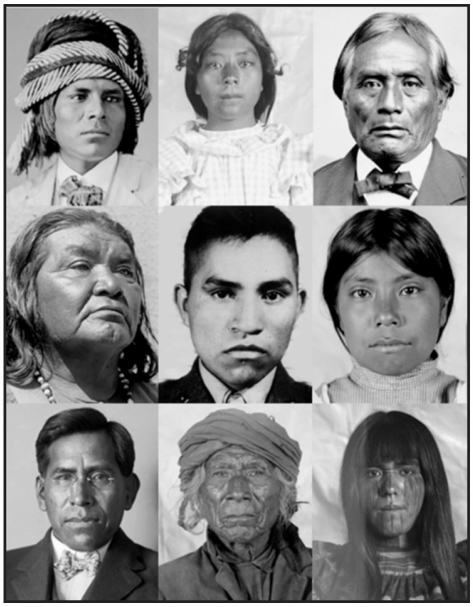


17 PIMA 인디언 당뇨병 예측

17.1 PIMA 인디언 문제?



17.2 PIMA 인디언 당뇨병 문제

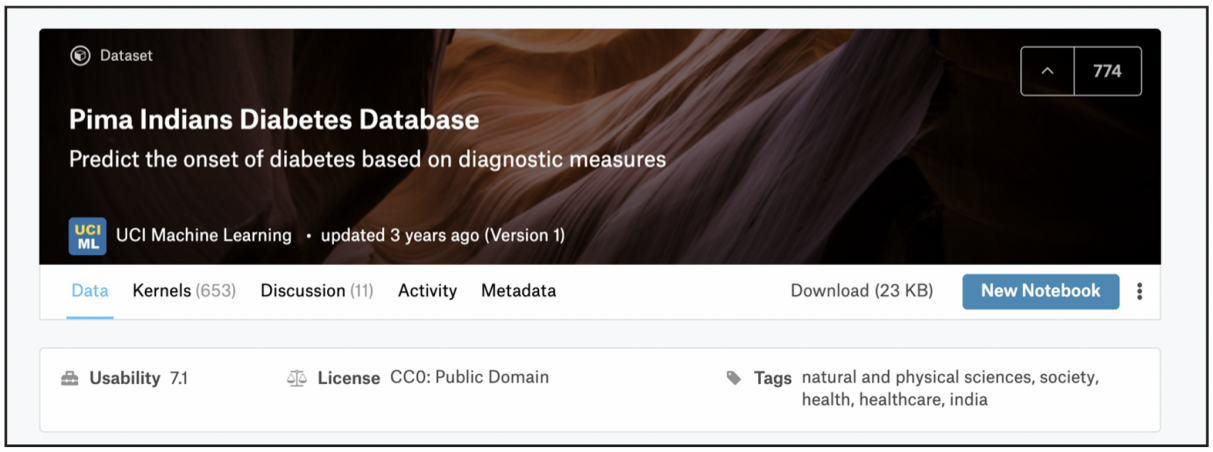


- 50년대까지 PIMA 인디언은 당뇨가 없었음
- 20세기 말, 50%가 당뇨에 걸림
- 50년만에 50%의 인구가 당뇨에 걸림

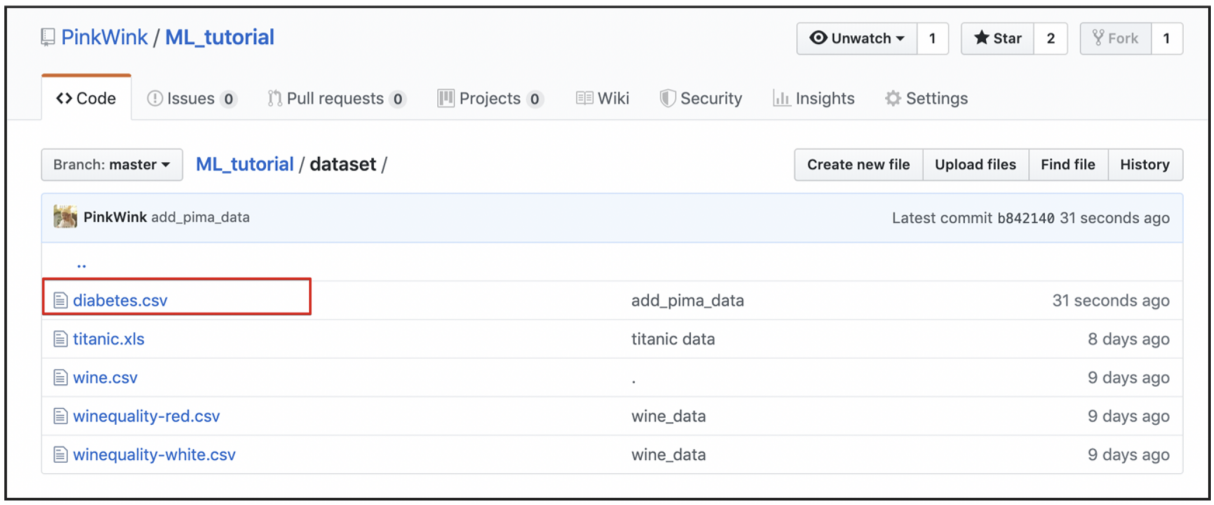
17.3 본래 강가에서 수렵하던 가난한 소수 인디언

17.4 이 중, 미국 쪽 PIMA 인디언은 미국 정부에 의해 강제 이주 후 식량을 배급 받음

17.5 원래 데이터는 Kaggle



17.6 이 데이터를 PinkWink's GitHub에 보관 중



17.7 Data의 컬럼의 의미

Pregnancies	임신 횟수
Glucose	포도당 부하 검사 수치
BloodPressure	혈압
Skin Thickness	팔 삼두근 뒤쪽의 피하지방 측정값
Insulin	혈청 인슐린
BMI	체질량지수
Diabetes Pedigree Funtion	당뇨 내력 가중치 값
Age	나이
Outcome	클래스 결정. 당뇨 유무

17.8 데이터 읽기

```python
import pandas as pd

PIMA_url = (
    "https://raw.githubusercontent.com/PinkWink/ML_tutorial/master/"+\
                                        "dataset/diabetes.csv"
)

PIMA = pd.read_csv(PIMA_url)
PIMA.head()
```

✓ 0.5s Python

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
0	6	148	72	35	0	33.6	0.6
1	1	85	66	29	0	26.6	0.3
2	8	183	64	0	0	23.3	0.6
3	1	89	66	23	94	28.1	0.1
4	0	137	40	35	168	43.1	2.2

17.9 데이터 확인

```
PIMA.info()
```

✓ 0.0s Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

17.10 float으로 데이터 변환

```
PIMA = PIMA.astype("float")
PIMA.info()
```

✓ 0.0s Python

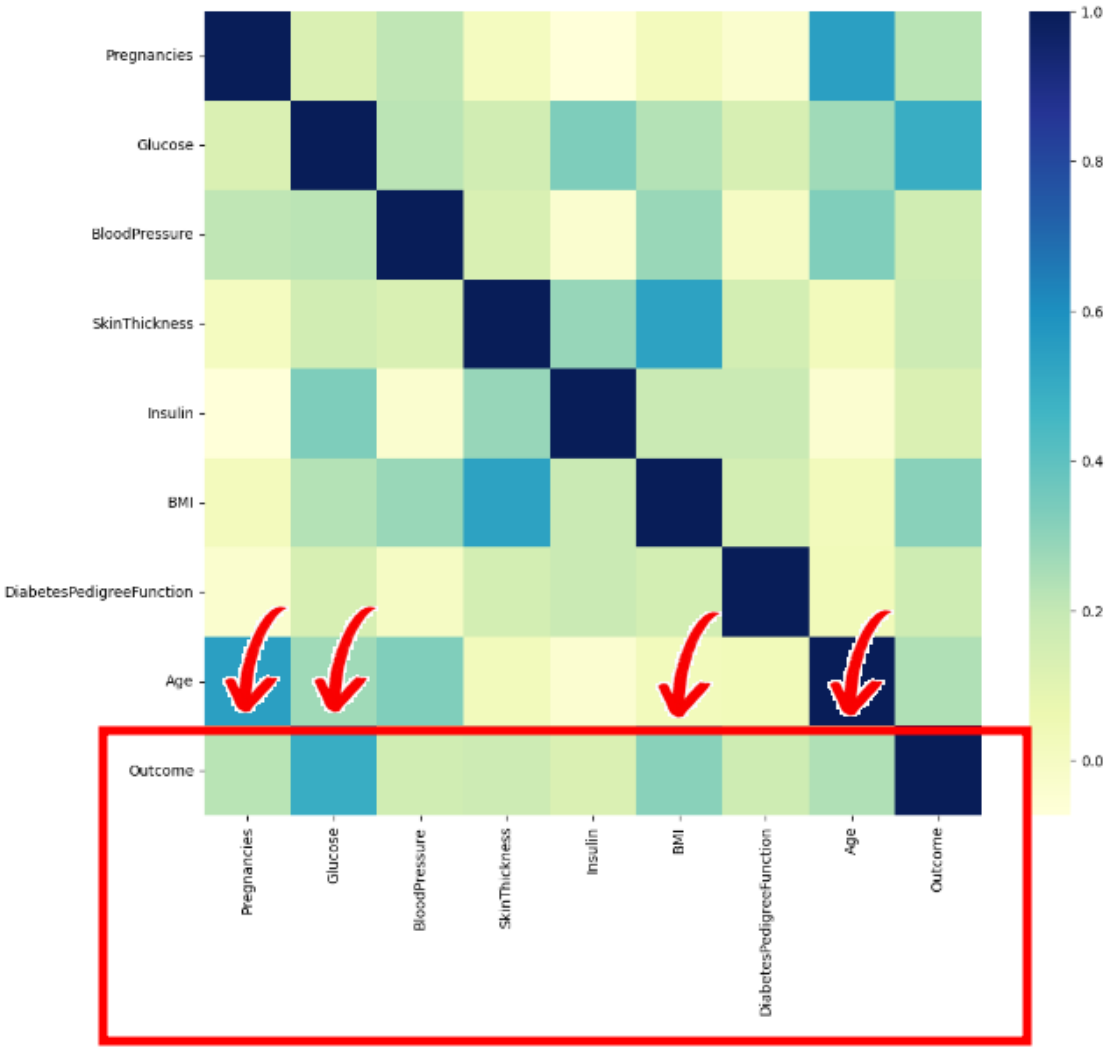
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    float64
 1   Glucose                   768 non-null    float64
 2   BloodPressure             768 non-null    float64
 3   SkinThickness             768 non-null    float64
 4   Insulin                   768 non-null    float64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    float64
 8   Outcome                   768 non-null    float64
dtypes: float64(9)
memory usage: 54.1 KB
```

17.11 일단 상관관계 확인

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 10))
sns.heatmap(PIMA.corr(), cmap = 'YlGnBu')
plt.show()
```

✓ 0.1s Python

17.12 Outcome과 다른 특성과의 관계



17.13 그런데 0 이 있다

```
(PIMA == 0).astype(int).sum()
```

✓ 0.0s Python

```
Pregnancies                 111
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                     500
dtype: int64
```

- 결측치는 데이터에 따라 그 정의가 다르다. 지금은 0이라는 숫자가 혈압에 있다는 것은 확실히 문제겠지

17.14 의학적 지식과 PIMA 인디언에 대한 정보가 없으므로 일단 평균값으로 대체

```
zero_features = ["Glucose", "BloodPressure", "SkinThickness", "BMI"]
PIMA[zero_features] = PIMA[zero_features].replace(0, PIMA[zero_features].mean())
(PIMA == 0).astype(int).sum()
```

✓ 0.0s Python

```
Pregnancies                 111
Glucose                       0
BloodPressure                 0
SkinThickness                 0
Insulin                     374
BMI                           0
DiabetesPedigreeFunction      0
Age                           0
Outcome                     500
dtype: int64
```

17.15 데이터를 나누고

```python
from sklearn.model_selection import train_test_split

X = PIMA.drop(["Outcome"], axis=1)
y = PIMA["Outcome"]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=13, stratify=y
)
```

✓ 0.0s Python

17.16 Pipeline을 만들고

```python
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

estimators = [
    ("scaler", StandardScaler()),
    ("clf", LogisticRegression(solver="liblinear", random_state=13)),
]

pipe_lr = Pipeline(estimators)
pipe_lr.fit(X_train, y_train)
pred = pipe_lr.predict(X_test)
```

✓ 0.0s Python

17.17 몇몇 수치를 확인

```python
from sklearn.metrics import (
    accuracy_score,
    recall_score,
    precision_score,
    roc_auc_score,
    f1_score,
)

print("Accuracy : ", accuracy_score(y_test, pred))
print("Recall : ", recall_score(y_test, pred))
print("Precision : ", precision_score(y_test, pred))
print("AUC score : ", roc_auc_score(y_test, pred))
print("F1 Score : ", f1_score(y_test, pred))
```

✓ 0.0s Python

```
Accuracy :  0.7727272727272727
Recall :  0.6111111111111112
Precision :  0.7021276595744681
AUC score :  0.7355555555555556
F1 Score :  0.6534653465346535
```

- 그러나 상대적 의미를 가질 수 없어서 이 수치 자체를 평가할 수는 없다.

17.18 다변수 방정식의 각 계수값을 확인할 수 있다

```python
coeff = list(pipe_lr["clf"].coef_[0])
labels = list(X_train.columns)
```

✓ 0.0s Python

```python
coeff
```

✓ 0.0s Python

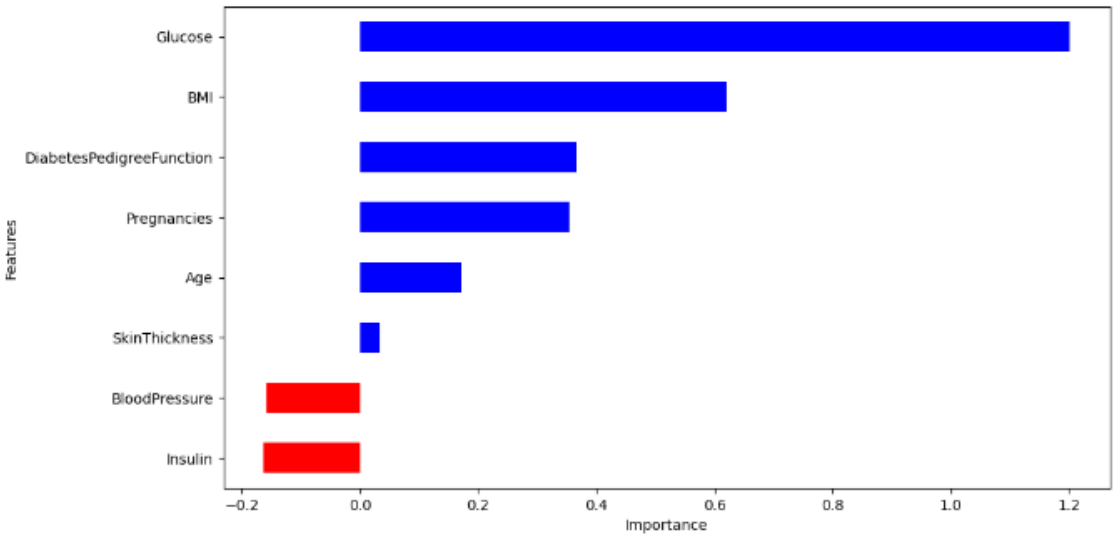
```
[np.float64(0.3542658884412649),
 np.float64(1.201424442503758),
 np.float64(-0.1584013553628671),
 np.float64(0.033946577129299535),
 np.float64(-0.1628647195398812),
 np.float64(0.620404521989511),
 np.float64(0.36669355795578734),
 np.float64(0.17195965447035097)]
```

17.19 중요한 feature에 대해 그려볼까

```python
features = pd.DataFrame({"Features": labels, "importance": coeff})
features.sort_values(by=["importance"], ascending=True, inplace=True)
features["positive"] = features["importance"] > 0
features.set_index("Features", inplace=True)
features["importance"].plot(
    kind="barh",
    figsize=(11, 6),
    color=features["positive"].map({True: "blue", False: "red"}),
)
plt.xlabel("Importance")
plt.show()
```

✓ 0.0s Python

17.20 해석을 해볼수 있을까



- 포도당, BMI 등은 당뇨에 영향을 미치는 정도가 높다
- 혈압은 예측에 부정적 영향을 준다
- 연령이 BMI보다 출력 변수와 더 관련되어 있었지만, 모델은 BMI와 Glucose에 더 의존함