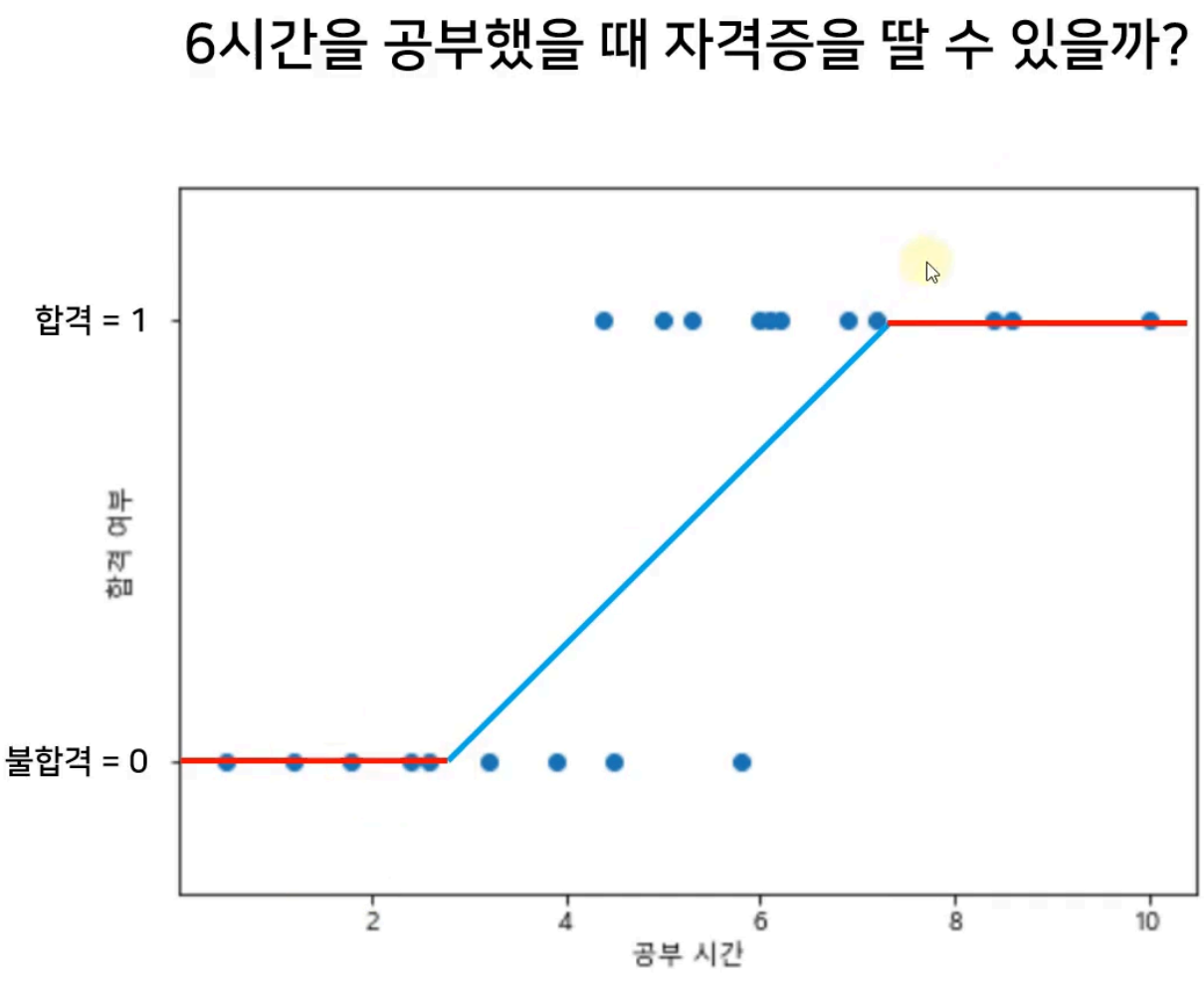
4. Logistic Regression - 분류 모델

선형 회귀방식을 분류에 적용한 알고리즘, 데이터가 어떤 범주에 속할 확률을 0~1 사이의 값으로 예측, 더 높은 범주에 속하는 쪽으로 분류해주는 지도 학습 알고리즘이다.

범주 : True/False, Yes/No, 합격/불합격,...
예)스팸메일, 은행 대출, 아ㅣㄱ성여부, 고객의 제품 구매 의사...
스펨 메일 분류기 같은 예시를 생각하면 쉬운데, 어떤 메일을 받았을 때 그 메일이 스팸일 확률이 0.5이상이면 스팸으로 분류하고, 메일이 스팸일 확률이 0.5보다 낮으면 일반 메일로 분류하는 것이다. 이렇게 데이터가 2개의 범주 중 하나에 속하도록 결정하는 것을 Binary Classification (2진 분류)라고 한다.

공부 시간에 따른 자격증 시험 합격 가능성

양끝에 범위를 벗어나는 부부은 빨간색으로 수정 즉 1보다 큰값들은 1로, 0보다 작은 값들은 0으로 보는 것이다.



이런식의 부류를 가능하게 하기 위해서 우리는 시그모이드(sigmoid)함수 또는 로지스틱(Logistic)함수 라는 것

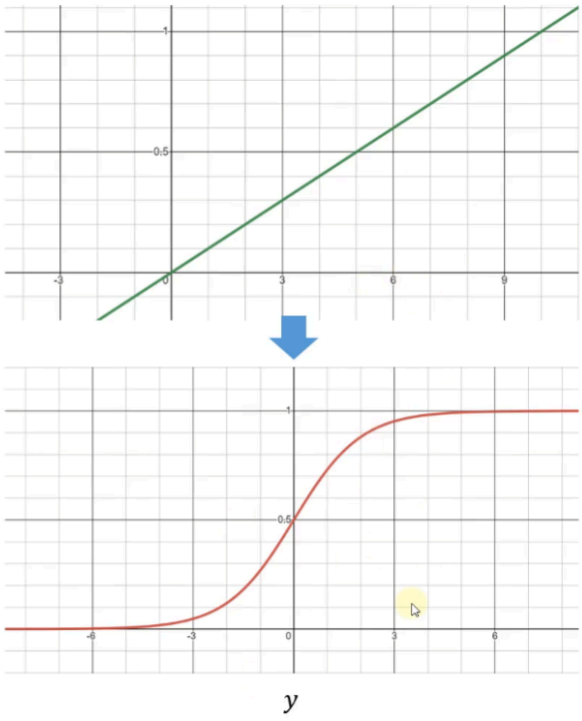
을 사용한다. 단순한 y = mx + b 의 함수를 시그모이—드 함수로 변형한 것이 아래 그림이다.

$$y = mx + b$$

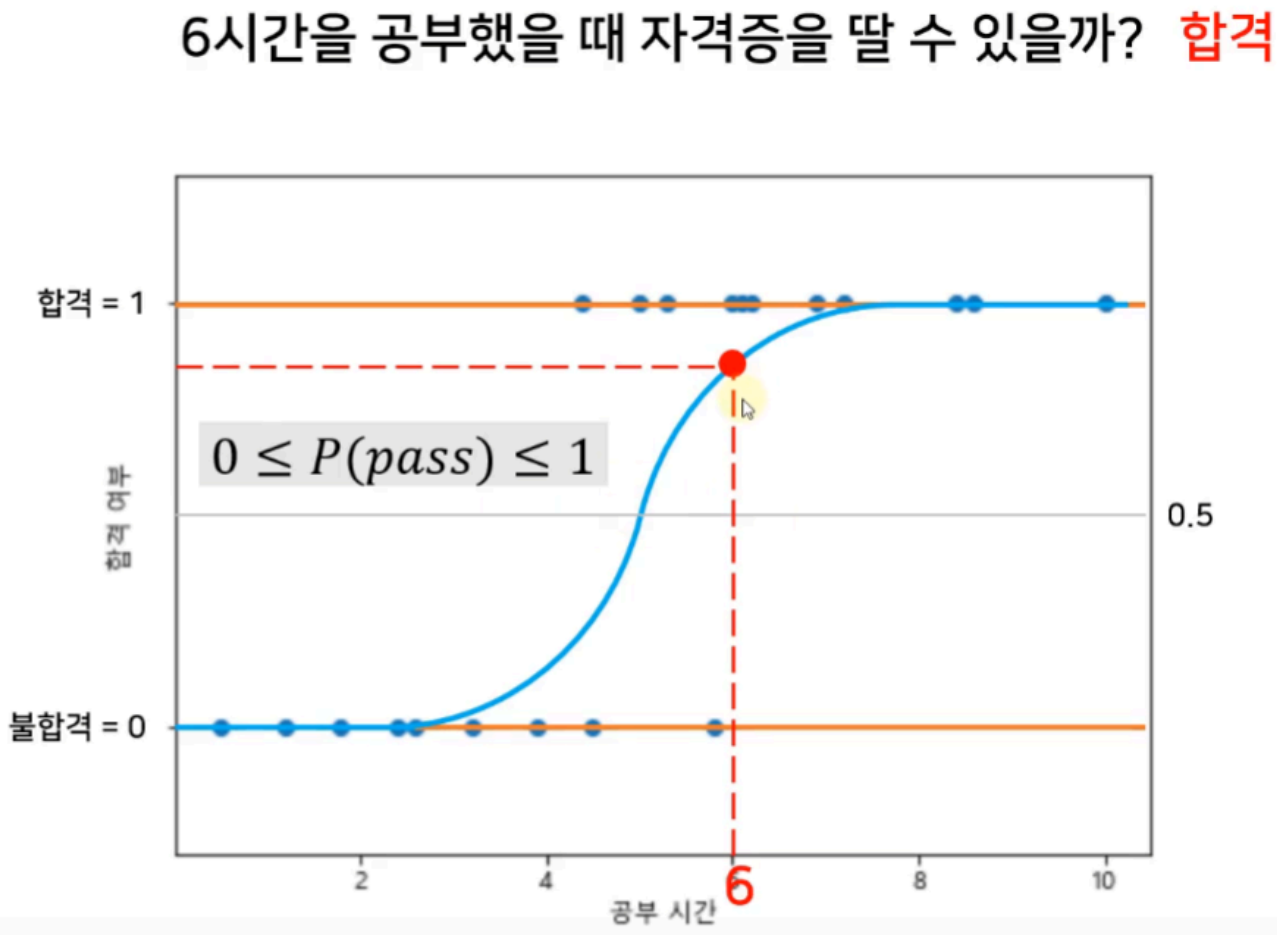
Sigmoid Function

$$P = \frac{1}{1 + e^{-y}}$$

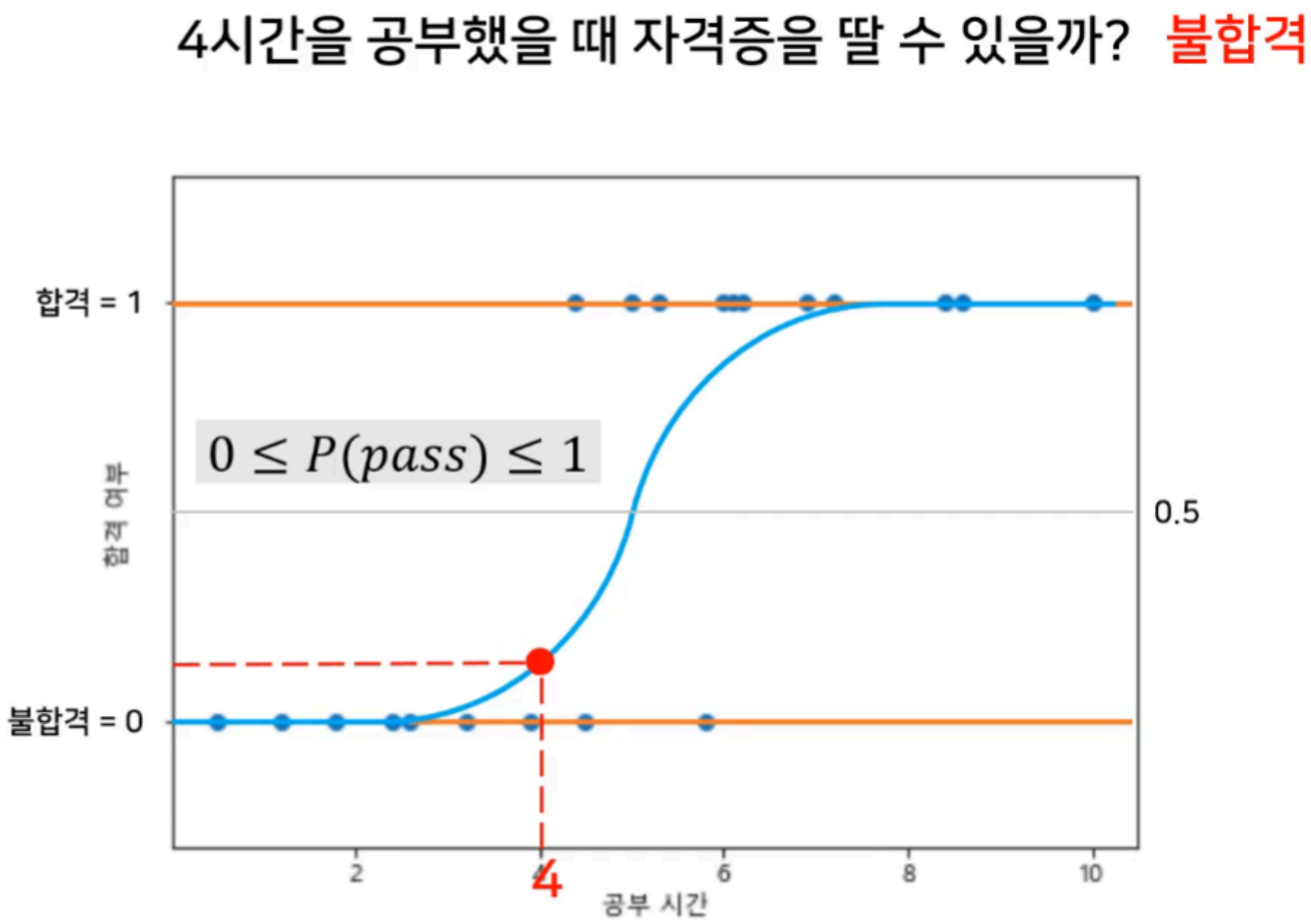
$$ln\left(\frac{P}{1-P}\right) = mx + b$$



여기서 보면 p가 항상 최소 0부터 최대 1 사이의 값만 가질수 있게 되고 y 값이 아무리 커져도 1에만 가까워지고, y값이 아무리 작아도 0으로만 수렴하는 것을 알수 있다. y 에 일차식을 대입해서 푼 식이 가장 아래 있는 식이다. 즉 정리를 하면 위 그래픽 형태에서 시그모이드 함수를 토과 시켜서 아래와 같은 형태의 모델로 만들수 있다. 우리에 식을 시그모이드 함수로 변형하면 아래와 같은 그래프가 나온다.



여기 중간에 회색 부분이 딱 0.5 값이 됩니다. 이렇게 가로로 연장선을 그어 보면은 가능성이 0.5 보다 큰 각도는 아무리 커져도 1에 가까워지고 0.5 보다 작은 각도는 아무리 작아도 0에 가까워져서 결국은 합격률이 0부터 1사이가 된다. 그래서 공부시간이 6시간이면 그림처럼 합격이 되고,



공부시간이 4시간이면 그림처럼 불합격이 된다.

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

```
from google.colab import drive # 구글드라이버에 있는 파일 사용하기
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
dataset = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/활용편7-머신러닝/ScikitLearn/Logisti
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

데이터 분리

```
from sklearn.model_selection import train_test_split # 4개의 데이터로 분리됨.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0) # 20% 분
```

학습 (로지스틱 회귀 모델)

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression() # 객체 생성
classifier.fit(X_train, y_train) # 학습
```

```
▾ LogisticRegression
LogisticRegression()
```

6시간 공부했을 때 예측?

```
classifier.predict([[6]]) # 6시간 공부했을 때의 예측 2차원 배열로
# 결과 1 : 합격할 것으로 예측
```

```
array([1])
```

```
classifier.predict_proba([[6]]) # 합격할 확률 출력
# 불합격 확률 14%, 합격 확률 86%
```

```
array([[0.14150735, 0.85849265]])
```

4시간 공부했을 때 예측?

```
classifier.predict([[4]]) # 4시간 공부했을 때의 예측 2차원 배열로
# 결과 0 : 불합격할 것으로 예측
```

```
array([0])
```

```
classifier.predict_proba([[4]]) # 합격할 확률 출력
# 불합격 확률 62%, 합격 확률 38%
```

```
array([[0.6249966, 0.3750034]])
```

분류 결과 예측 (테스트 세트)

```
y_pred = classifier.predict(X_test)
y_pred # 예측 값
```

```
array([1, 0, 1, 1])
```

```
y_test # 실제 값 (테스트 세트)
```

```
array([1, 0, 1, 0])
```

```
X_test # 공부 시간 (테스트 세트)
```

```
array([[ 8.6],
       [ 1.2],
       [10. ],
       [ 4.5]])
```

```
classifier.score(X_test, y_test) # 모델 평가
# 전체 테스트 세트 4개 중에서 분류 예측을 올바로 맞힌 개수 3개 -> 3/4 = 0.75 맞힌 개수로 점수가 니
```

```
0.75
```

데이터 시각화 (훈련 세트)

```
X_range = np.arange(np.min(X), np.max(X), 0.1) # X 의 최소값에서 최대값까지를 0.1 단위로 잘라서 데
X_range
```

```
array([0.5, 0.6, 0.7, 0.8, 0.9, 1. , 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7,
       1.8, 1.9, 2. , 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 3. ,
       3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 4. , 4.1, 4.2, 4.3,
       4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 5. , 5.1, 5.2, 5.3, 5.4, 5.5, 5.6,
       5.7, 5.8, 5.9, 6. , 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, 6.8, 6.9,
       7. , 7.1, 7.2, 7.3, 7.4, 7.5, 7.6, 7.7, 7.8, 7.9, 8. , 8.1, 8.2,
       8.3, 8.4, 8.5, 8.6, 8.7, 8.8, 8.9, 9. , 9.1, 9.2, 9.3, 9.4, 9.5,
       9.6, 9.7, 9.8, 9.9])
```

```
p = 1 / (1 + np.exp(-(classifier.coef_ * X_range + classifier.intercept_))) # y = mx + b  를 대입
p
```

```
array([[0.01035705, 0.01161247, 0.01301807, 0.0145913 , 0.01635149,
        0.01832008, 0.02052073, 0.02297953, 0.02572521, 0.02878929,
        0.03220626, 0.03601375, 0.04025264, 0.04496719, 0.05020505,
        0.05601722, 0.06245802, 0.06958479, 0.07745757, 0.08613861,
        0.09569165, 0.10618106, 0.11767067, 0.13022241, 0.14389468,
        0.15874043, 0.17480509, 0.19212422, 0.2107211 , 0.23060425,
        0.25176509, 0.27417574, 0.29778732, 0.32252874, 0.34830616,
        0.3750034 , 0.40248315, 0.43058927, 0.45914989, 0.48798142,
        0.51689314, 0.54569221, 0.57418876, 0.60220088, 0.6295591 ,
        0.65611024, 0.68172044, 0.70627722, 0.72969059, 0.75189324,
        0.77283994, 0.79250621, 0.81088652, 0.82799203, 0.84384828,
        0.85849265, 0.871972  , 0.88434036, 0.89565683, 0.90598377,
        0.91538521, 0.92392546, 0.93166808, 0.93867499, 0.9450058 ,
        0.95071738, 0.95586346, 0.96049453, 0.96465764, 0.96839647,
        0.97175136, 0.97475939, 0.97745455, 0.97986786, 0.9820276 ,
        0.98395944, 0.98568665, 0.9872303 , 0.98860939, 0.98984107,
        0.9909408 , 0.99192244, 0.99279849, 0.99358014, 0.99427745,
        0.9948994 , 0.99545406, 0.99594865, 0.99638963, 0.99678276,
        0.99713321, 0.99744558, 0.997724  , 0.99797213, 0.99819325]])
```

```
p.shape # 로우 1개와 컬럼 95개인 2차원 배열
```

```
(1, 95)
```

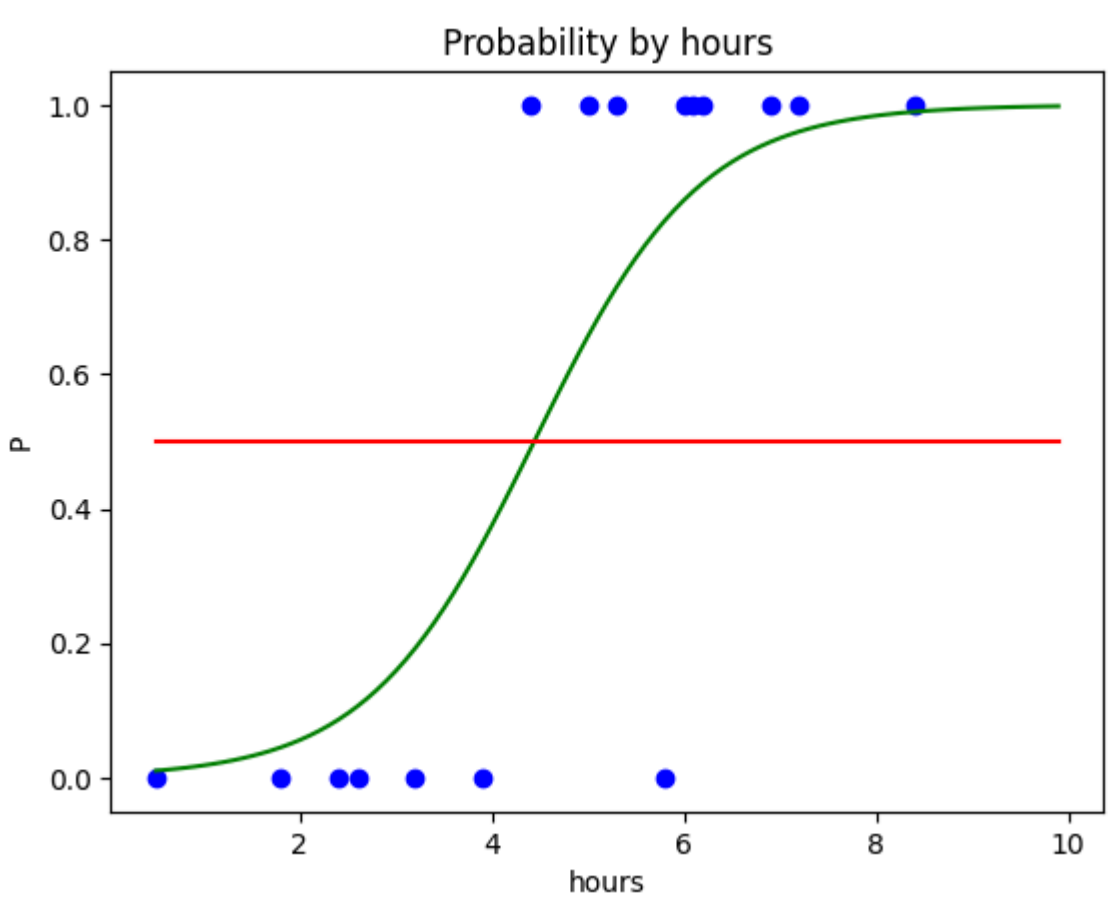
```
X_range.shape # 95개인 1차원 배열
```

```
(95,)
```

```
p = p.reshape(-1) # 2차원 배열을 1차원 배열 형태로 변경. -1인 자동으로 대입. 예) 4X4 = 2X(-1) 이면
p.shape
```

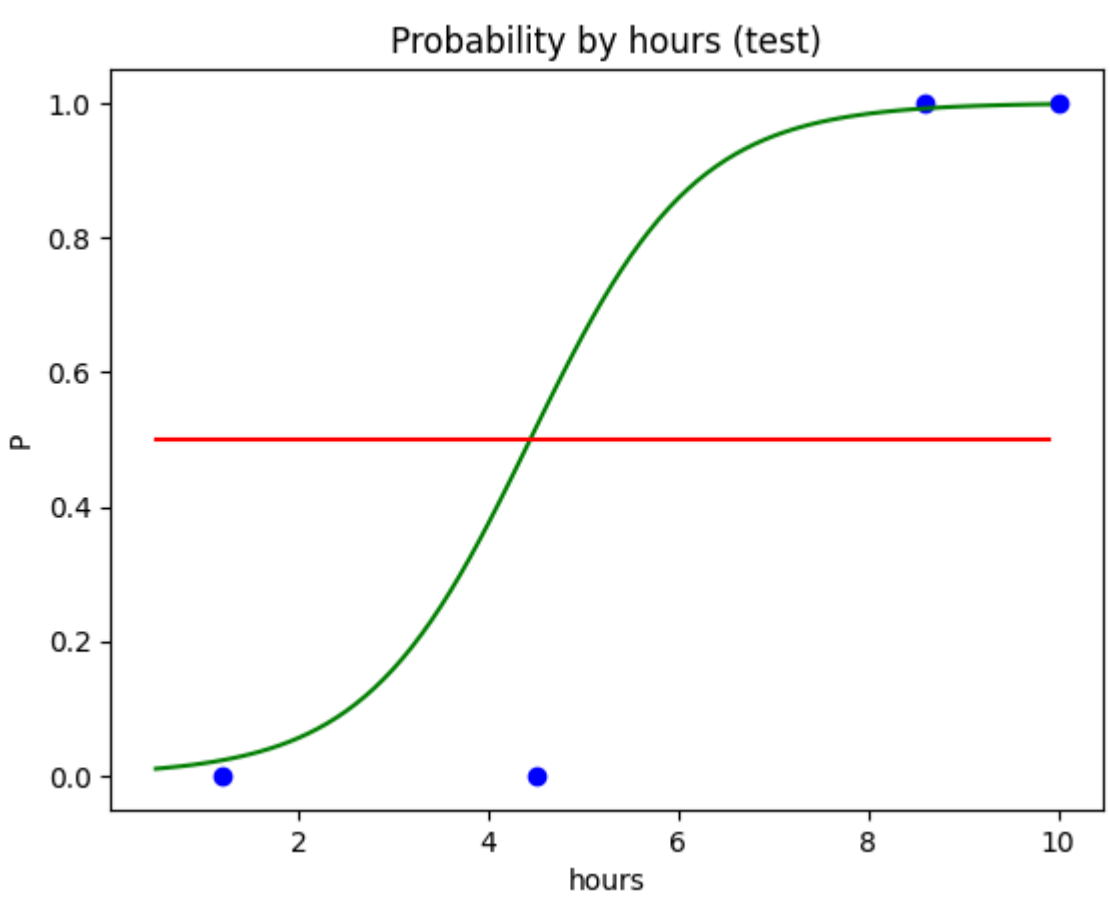
```
(95,)
```

```
plt.scatter(X_train, y_train, color='blue')
plt.plot(X_range, p, color='green')
plt.plot(X_range, np.full(len(X_range), 0.5), color='red') # X_range 개수만큼 0.5 로 가득찬 배열 생
plt.title('Probability by hours') # 합격 가능 시간
plt.xlabel('hours')
plt.ylabel('P')
plt.show()
```



데이터 시각화 (테스트 세트)

```
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_range, p, color='green')
plt.plot(X_range, np.full(len(X_range), 0.5), color='red') # X_range 개수만큼 0.5 로 가득찬 배열 생
plt.title('Probability by hours (test)')
plt.xlabel('hours')
plt.ylabel('P')
plt.show()
```




```
classifier.predict_proba([[4.5]]) # 4.5 시간 공부했을 때 확률 (모델에서는 51% 확률로 합격 예측, 실
```

```
array([[0.48310686, 0.51689314]])
```

혼동 행렬 (Confusion Matrix)

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm

# TRUE NEGATIVE (TN)=1개       FALSE POSITIVE (FP)=1개
# 불합격일거야 (예측)          합격일거야 (예측)
# 불합격 (실제)                불합격 (실제)

# FALSE NEGATIVE (FN)=0개      TRUE POSITIVE (TP)=2개
# 불합격일거야 (예측)          합격일거야 (예측)
# 합격 (실제)                  합격 (실제)

# 왼쪽위 오른쪽 아래는 올바로 예측한 거고 오른쪽위 왼쪽 아래는 잘못 예측을 한 것이다.
```

```
array([[1, 1],
       [0, 2]])
```


TRUE NEGATIVE
You're not pregnant
FALSE POSITIVE
You're pregnant
TYPE 1 ERROR
FALSE NEGATIVE
You're not pregnant
TYPE 2 ERROR
TRUE POSITIVE
You're pregnant