

```

In [1]: import pandas as pd
import numpy as np
from scipy import stats
import math

# matplotlib and seaborn for visualizations
import seaborn as sns
import matplotlib.pyplot as plt
plt.rcParams['font.size'] = 12

# Suppress warnings from pandas
import warnings
warnings.filterwarnings('ignore')

# modeling
import lightgbm as lgb

# utilities
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score

sns.set_style("darkgrid")
# memory management
import gc
import os
# 운영체제별 한글 폰트 설정
if os.name == 'posix': # Mac 환경 폰트 설정
    plt.rc('font', family='AppleGothic')
elif os.name == 'nt': # Windows 환경 폰트 설정
    plt.rc('font', family='Malgun Gothic')

plt.rc('axes', unicode_minus=False) # 마이너스 폰트 설정

# 글씨 선명하게 출력하는 설정
%config InlineBackend.figure_format = 'retina'

df = pd.read_csv("dataset2.csv")
# import mglearn
# from sklearn.model_selection import KFold
# from sklearn.preprocessing import LabelEncoder

```

```

In [2]: len(df[df['당뇨여부']==1])

```

Out[2]: 46579

```

In [3]: nodang = df[df['당뇨여부']==0].copy()
dang = df[df['당뇨여부']==1].copy()
print(len(nodang))
print(len(dang))

```

1133541
46579

```

In [4]: x = df[['허리둘레', '연령대코드 (5세단위)', '감마지티피', '트리글리세라이드', 'LDL콜레스테롤']].
y = df[['당뇨여부']].copy()

```

비당뇨 : 당뇨 = 10 : 2

```
In [5]: from imblearn.under_sampling import RandomUnderSampler

rus = RandomUnderSampler(random_state=42, sampling_strategy=0.2)
X_rus, y_rus = rus.fit_resample(X, y)
```

```
In [6]: print(len(X_rus))
print(len(y_rus))
print(np.sum(y_rus))
```

```
279474
279474
당뇨여부      46579.0
dtype: float64
```

```
In [7]: y_rus
```

```
Out[7]:
```

	당뇨여부
0	0.0
1	0.0
2	0.0
3	0.0
4	0.0
...	...
279469	1.0
279470	1.0
279471	1.0
279472	1.0
279473	1.0

279474 rows × 1 columns

```
In [8]: X_train, X_test, y_train, y_test = train_test_split(X_rus, y_rus.values, stratify=y_rus,
from lightgbm import LGBMClassifier, plot_importance
from sklearn.preprocessing import StandardScaler, RobustScaler
sc = RobustScaler()
x_train = sc.fit_transform(X_train)
x_test = sc.transform(X_test)
lgb = LGBMClassifier(n_estimators=400)
evals = [(x_test, y_test)]
```

```
In [9]: # lgb.fit(x_train, y_train, early_stopping_rounds=100, eval_metric="logloss",
lgb.fit(x_train, y_train, eval_metric="logloss", eval_set=evals, verbose=False)
```

```
Out[9]: LGBMClassifier(n_estimators=400)
```

```
In [10]: y_pred = lgb.predict(x_test)
```

```
In [11]: is_correct = y_pred == y_test
```

정확도

```
In [12]: lgb.score(x_test, y_test)
```

```
Out[12]: 0.8331156632972537
```

예측 결과

```
In [13]: print("당뇨로 예측한 데이터 수 :", np.sum(y_pred))
print("실제 당뇨 데이터 수 :", np.sum(y_test))
print("전체 데이터 수 :", len(y_test))
```

당뇨로 예측한 데이터 수 : 1306.0

실제 당뇨 데이터 수 : 9316.0

전체 데이터 수 : 55895

TN, FP, FN, TP

```
In [14]: from sklearn import metrics
metrics.confusion_matrix(y_test, y_pred)

# [[TN, FP],
#   [FN, TP]]
```

```
Out[14]: array([[45920, 659],
               [ 8669, 647]])
```

TN, FP, FN, TP 검토

```
In [15]: y_test = y_test.reshape(-1)
```

```
In [16]: P = sum(y_test)
TP = sum((y_test==1) & (y_pred==1))
TPR = TP/P
FN = sum((y_test==1) & (y_pred==0))
FNR = FN/P
N = sum(y_test==0)
TN = sum((y_test==0) & (y_pred==0))
TNR = TN/N
FP = sum((y_test==0) & (y_pred==1))
FPR = FP/N
print(TN, FP, FN, TP)
print(TPR, FPR)

45920 659 8669 647
0.06945040790038644 0.014148006612421906
```

데이터 별 당뇨 분류 확률

```
In [17]: y_pred_proba = lgb.predict_proba(x_test)
y_pred_proba
```

```
Out[17]: array([[0.90994922, 0.09005078],
               [0.72710705, 0.27289295],
```

```
[0.95273592, 0.04726408],
...,
[0.5836701 , 0.4163299 ],
[0.99330228, 0.00669772],
[0.92574684, 0.07425316]])
```

[당뇨X 확률, 당뇨 확률]

```
In [18]: y_pred_proba.shape
```

```
Out[18]: (55895, 2)
```

데이터 별 당뇨로 예측할 확률

```
In [19]: pos_proba = y_pred_proba[:,1]
# np.unique(pos_proba)
print(len(pos_proba[pos_proba > 0.8]))
```

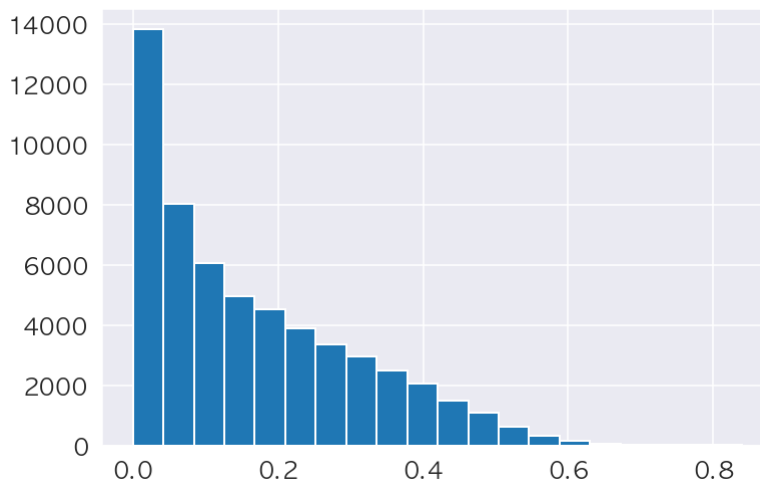
```
9
```

```
In [20]: print("pos_proba 최솟값 :", min(pos_proba))
print("pos_proba 최댓값 :", max(pos_proba))
```

```
pos_proba 최솟값 : 7.601913686444623e-05
pos_proba 최댓값 : 0.8400386366070816
```

```
In [21]: plt.hist(pos_proba, range=(min(pos_proba),max(pos_proba)), bins=20)
```

```
Out[21]: (array([1.3817e+04, 8.0320e+03, 6.0470e+03, 4.9670e+03, 4.5320e+03,
3.8890e+03, 3.3440e+03, 2.9480e+03, 2.4820e+03, 2.0640e+03,
1.4720e+03, 1.0950e+03, 6.1900e+02, 3.2500e+02, 1.4100e+02,
5.2000e+01, 2.8000e+01, 2.4000e+01, 8.0000e+00, 9.0000e+00]),
array([7.60191369e-05, 4.20741500e-02, 8.40722809e-02, 1.26070412e-01,
1.68068543e-01, 2.10066674e-01, 2.52064804e-01, 2.94062935e-01,
3.36061066e-01, 3.78059197e-01, 4.20057328e-01, 4.62055459e-01,
5.04053590e-01, 5.46051720e-01, 5.88049851e-01, 6.30047982e-01,
6.72046113e-01, 7.14044244e-01, 7.56042375e-01, 7.98040506e-01,
8.40038637e-01]),
<BarContainer object of 20 artists>)
```

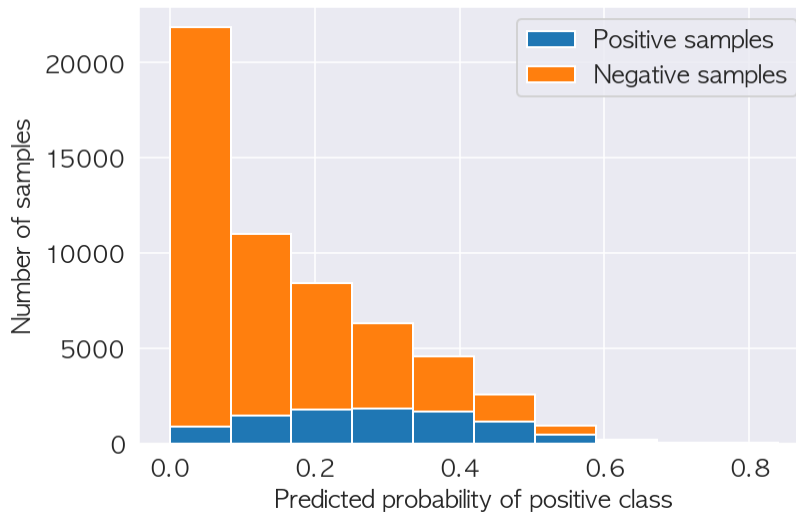


당뇨 예측 확률과 실제 데이터

```
In [22]: pos_sample_pos_proba = pos_proba[y_test==1]
neg_sample_pos_proba = pos_proba[y_test==0]
```

```
In [23]: plt.hist([pos_sample_pos_proba, neg_sample_pos_proba], histtype="barstacked")
plt.legend(["Positive samples", "Negative samples"])
plt.xlabel("Predicted probability of positive class")
plt.ylabel("Number of samples")
```

Out[23]: Text(0, 0.5, 'Number of samples')

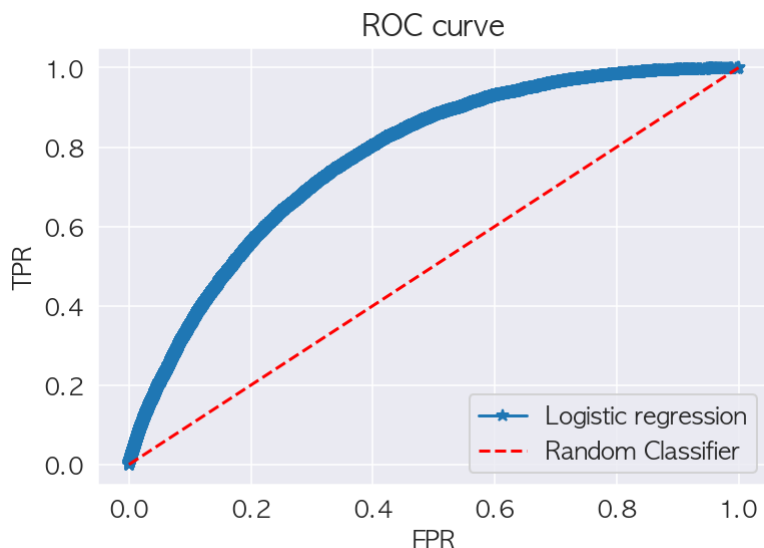


ROC Curve

```
In [24]: fpr, tpr, thresholds = metrics.roc_curve(y_test, pos_proba)
```

```
In [25]: plt.plot(fpr, tpr, '*-')
plt.plot([0,1], [0,1], 'r--')
plt.legend(['Logistic regression', 'Random Classifier'])
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.title('ROC curve')
```

Out[25]: Text(0.5, 1.0, 'ROC curve')



AUC Score

```
In [26]: metrics.roc_auc_score(y_test, pos_proba)
```

```
Out[26]: 0.7712831972119815
```

Precision, Recall, Accuracy

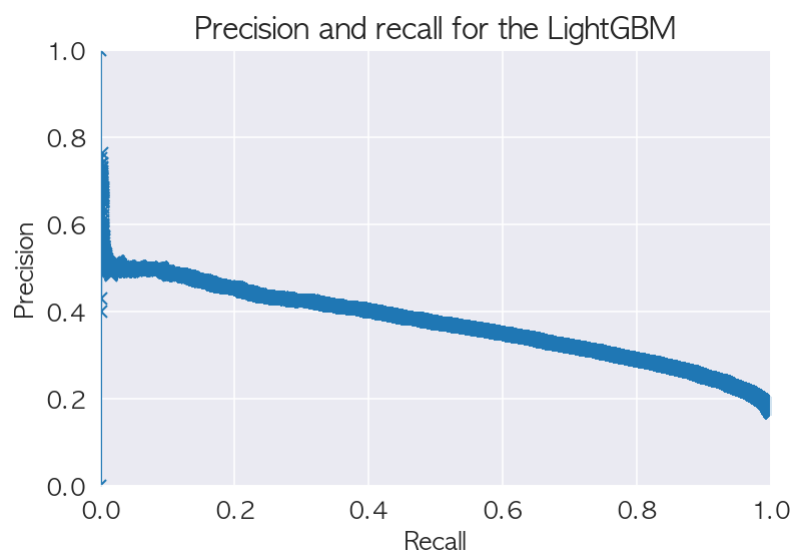
```
In [27]: precision = TP/(TP+FP)
recall = TP/(TP+FN)
accuracy = (TP+TN)/(TP+FP+TN+FN)
print("precision :",precision)
print("recall :", recall)
print("accuracy :", accuracy)
```

```
precision : 0.49540581929555894
recall : 0.06945040790038644
accuracy : 0.8331156632972537
```

```
In [28]: precision2, recall2, thresholds2 = metrics.precision_recall_curve(y_test, pos_
```

```
In [29]: plt.plot(recall2, precision2, '-x')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision and recall for the LightGBM')
plt.xlim([0,1])
plt.ylim([0,1])
```

```
Out[29]: (0.0, 1.0)
```



```
In [30]: metrics.auc(recall2, precision2)
```

```
Out[30]: 0.37042711586170246
```

```
In [ ]:
```