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

# matplotlit 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']].copy()
y = df[['당뇨여부']].copy()
```

비당뇨 : 당뇨 = 10 : 4

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

rus = RandomUnderSampler(random_state=42, sampling_strategy=0.4)
X_rus, y_rus = rus.fit_resample(X, y)
```

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

```
163026
163026
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
...	...
163021	1.0
163022	1.0
163023	1.0
163024	1.0
163025	1.0

163026 rows × 1 columns

In [8]:
```
X_train, X_test, y_train, y_test = train_test_split(X_rus,y_rus.values,strati
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)
```

정확도

In [11]:
```
lgb.score(x_test, y_test)
```

Out[11]:
```
0.7279335091700914
```

예측 결과

In [12]:
```
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 : 4751.0
실제 당뇨 데이터 수 : 9316.0
전체 데이터 수 : 32606
```

TN, FP, FN, TP

In [13]:
```
from sklearn import metrics
metrics.confusion_matrix(y_test, y_pred)

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

Out[13]:
```
array([[21137,  2153],
       [ 6718,  2598]])
```

TN, FP, FN, TP 검토

In [14]:
```
y_test = y_test.reshape(-1)
```

In [15]:
```
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)
```

```
21137 2153 6718 2598
0.2788750536711035 0.09244310863031344
```

데이터 별 당뇨 분류 확률

In [16]:
```
y_pred_proba = lgb.predict_proba(x_test)
y_pred_proba
```

Out[16]:
```
array([[0.77852064, 0.22147936],
       [0.48426646, 0.51573354],
       [0.83286343, 0.16713657],
       ...,
       [0.49261155, 0.50738845],
```

```
       [0.46410203, 0.53589797],
       [0.8565563 , 0.1434437 ]])
```

[당뇨X 확률, 당뇨 확률]

In [17]:
```
y_pred_proba.shape
```

Out[17]:
```
(32606, 2)
```

데이터 별 당뇨로 예측할 확률

In [18]:
```
pos_proba = y_pred_proba[:,1]
# np.unique(pos_proba)
print(len(pos_proba[pos_proba > 0.8]))
```

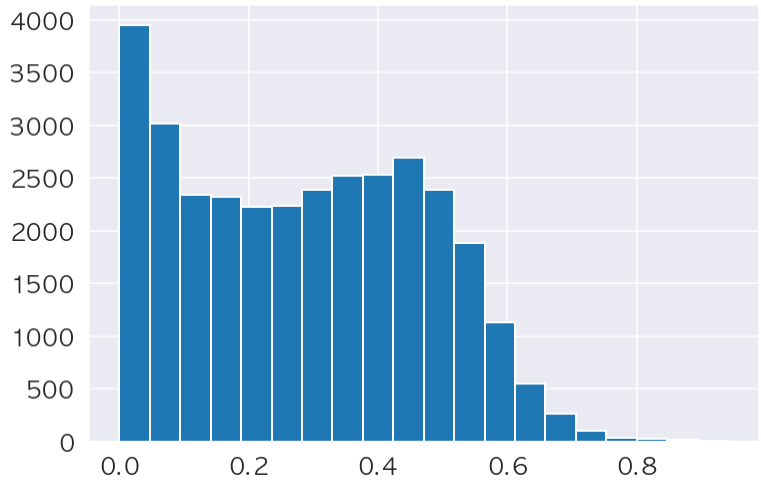
```
45
```

In [19]:
```
print("pos_proba 최솟값 :", min(pos_proba))
print("pos_proba 최댓값 :", max(pos_proba))
```

```
pos_proba 최솟값 : 6.387191488081119e-06
pos_proba 최댓값 : 0.9409455513526367
```

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

Out[20]:
```
(array([3944., 3018., 2337., 2316., 2229., 2237., 2383., 2522., 2530.,
        2695., 2388., 1881., 1133.,  547.,  260.,  102.,   39.,   24.,
          15.,    6.]),
 array([6.38719149e-06, 4.70533454e-02, 9.41003036e-02, 1.41147262e-01,
        1.88194220e-01, 2.35241178e-01, 2.82288136e-01, 3.29335095e-01,
        3.76382053e-01, 4.23429011e-01, 4.70475969e-01, 5.17522927e-01,
        5.64569886e-01, 6.11616844e-01, 6.58663802e-01, 7.05710760e-01,
        7.52757719e-01, 7.99804677e-01, 8.46851635e-01, 8.93898593e-01,
        9.40945551e-01]),
 <BarContainer object of 20 artists>)
```



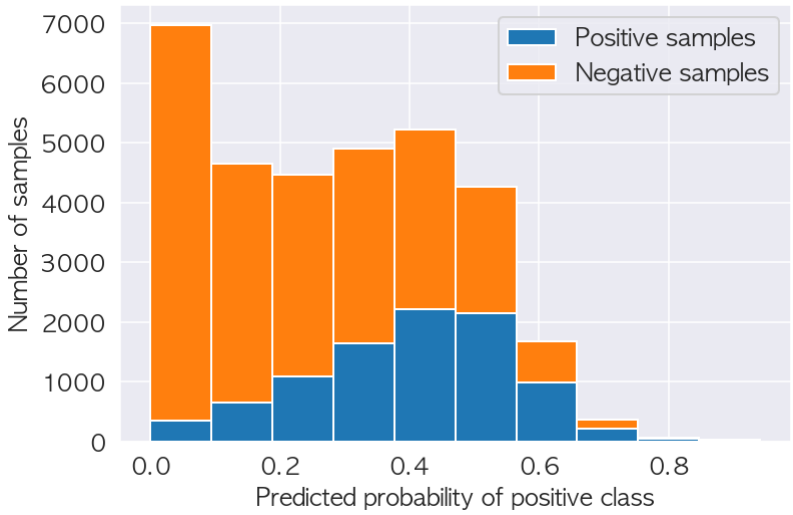
당뇨 예측 확률과 실제 데이터

In [21]:
```
pos_sample_pos_proba = pos_proba[y_test==1]
neg_sample_pos_proba = pos_proba[y_test==0]
```

In [22]:
```
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[22]: Text(0, 0.5, 'Number of samples')

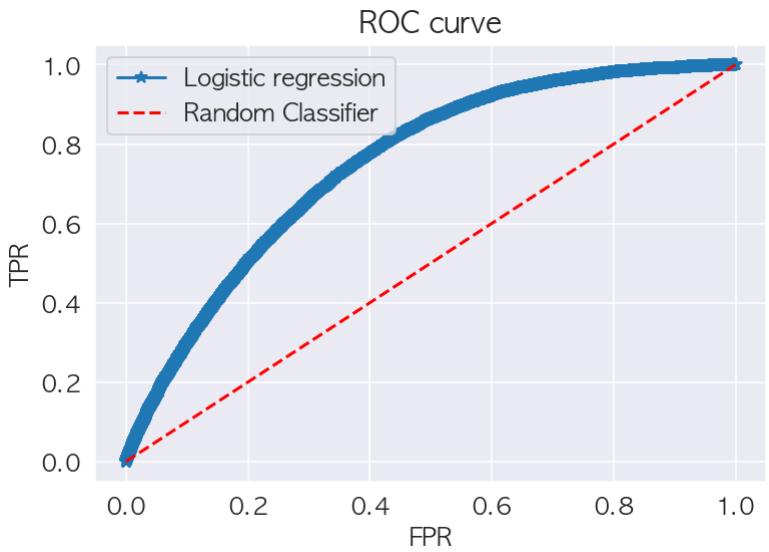


ROC Curve

In [23]:
```
fpr, tpr, thresholds = metrics.roc_curve(y_test, pos_proba)
```

In [24]:
```
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[24]: Text(0.5, 1.0, 'ROC curve')



AUC Score

In [25]:
```
metrics.roc_auc_score(y_test, pos_proba)
```

Out[25]: 0.7496368270694462

Precision, Recall, Accuracy

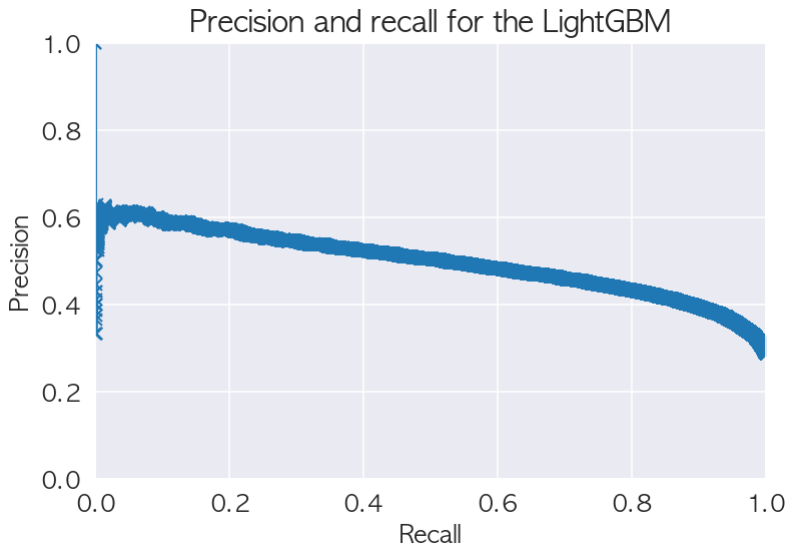
In [26]:
```
precision = TP/(TP+FP)
recall = TP/(TP+FN)
accuracy = (TP+TN)/(TP+FP+TN+FN)
print("precision :",precision)
print("recall :", recall)
print("accuracy :", accuracy)
```

```
precision : 0.5468322458429804
recall : 0.2788750536711035
accuracy : 0.7279335091700914
```

In [27]:
```
precision2, recall2, thresholds2 = metrics.precision_recall_curve(y_test, pos
```

In [28]:
```
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[28]: (0.0, 1.0)



In [29]:
```
metrics.auc(recall2, precision2)
```

Out[29]: 0.4959928902983133

In [ ]: