```
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', '트리글리세라이드', 'LDL콜레스테롤']].
           y = df[['당뇨여부']].copy()
```

비당뇨 : 당뇨 = 10 : 10

```python
from imblearn.under_sampling import RandomUnderSampler

rus = RandomUnderSampler(random_state=42, sampling_strategy=1)
X_rus, y_rus = rus.fit_resample(X, y)
```

```python
print(len(X_rus))
print(len(y_rus))
print(np.sum(y_rus))
```

```
93158
93158
당뇨여부    46579.0
dtype: float64
```

```python
y_rus
```

	당뇨여부
0	0.0
1	0.0
2	0.0
3	0.0
4	0.0
...	...
93153	1.0
93154	1.0
93155	1.0
93156	1.0
93157	1.0

93158 rows × 1 columns

```python
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)]
```

```python
# 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
```

```
LGBMClassifier(n_estimators=400)
```

```python
y_pred = lgb.predict(x_test)
```

```python
is_correct = y_pred == y_test
```

정확도

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

Out[12]: 0.7055066552168313

예측 결과

In [13]:
```python
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 : 10347.0
실제 당뇨 데이터 수 : 9316.0
전체 데이터 수 : 18632

TN, FP, FN, TP

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

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

Out[14]: array([[6057, 3259],
       [2228, 7088]])

TN, FP, FN, TP 검토

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

In [16]:
```python
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)
```

6057 3259 2228 7088
0.7608415629025332 0.3498282524688708

데이터 별 당뇨 분류 확률

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

Out[17]: array([[0.37866912, 0.62133088],
       [0.28232093, 0.71767907],

```
    [0.09449995, 0.90550005],
    ...,
    [0.39446942, 0.60553058],
    [0.85803203, 0.14196797],
    [0.32277   , 0.67723   ]])
```

[당뇨X 확률, 당뇨 확률]

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

Out[18]: (18632, 2)

데이터 별 당뇨로 예측할 확률

In [19]:
```python
pos_proba = y_pred_proba[:,1]
# np.unique(pos_proba)
print(len(pos_proba[pos_proba > 0.8]))
```
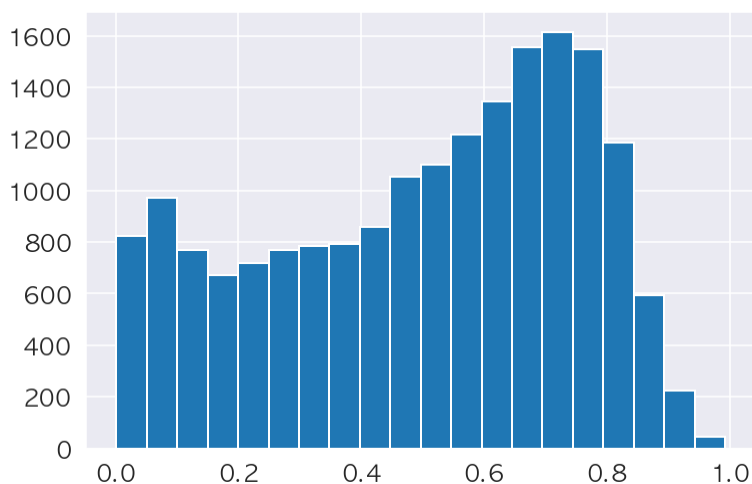
1869

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

pos_proba 최솟값 : 0.0011448151467392579
pos_proba 최댓값 : 0.9924261805883251

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

Out[21]: (array([822., 973., 769., 670., 720., 768., 783., 792., 860.,
        1053., 1098., 1218., 1344., 1555., 1612., 1548., 1186., 593.,
         223.,   45.]),
 array([0.00114482, 0.05070888, 0.10027295, 0.14983702, 0.19940109,
        0.24896516, 0.29852922, 0.34809329, 0.39765736, 0.44722143,
        0.4967855 , 0.54634957, 0.59591363, 0.6454777 , 0.69504177,
        0.74460584, 0.79416991, 0.84373398, 0.89329804, 0.94286211,
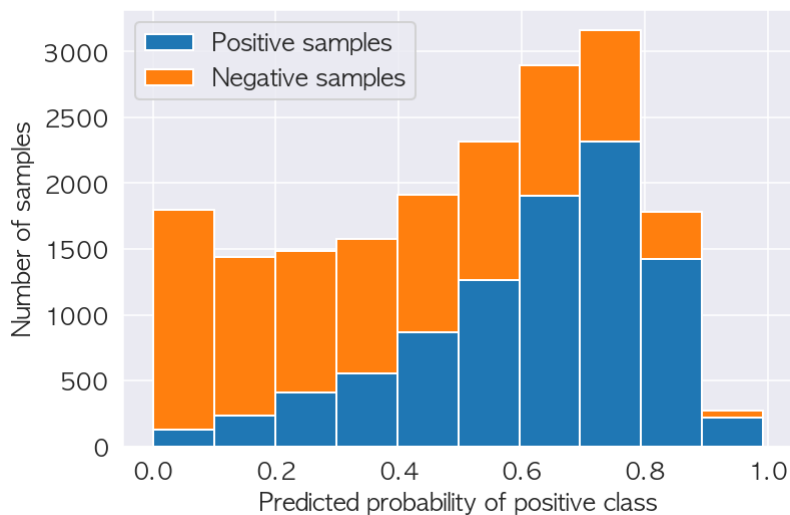   0.99242618]),
 <BarContainer object of 20 artists>)
```



## 당뇨 예측 확률과 실제 데이터

In [22]:
```python
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

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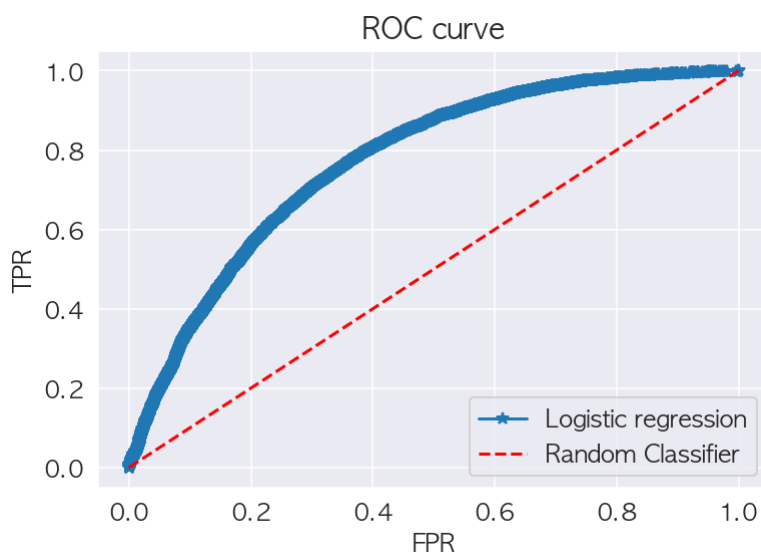In [24]:  fpr, tpr, thresholds = metrics.roc_curve(y_test, pos_proba)
```

```
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)
```

0.7705706141651892

## Precision, Recall, Accuracy

In [27]:
```python
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 : 0.6850294771431332
recall : 0.7608415629025332
accuracy : 0.7055066552168313
```

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

In [29]:
```python
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)
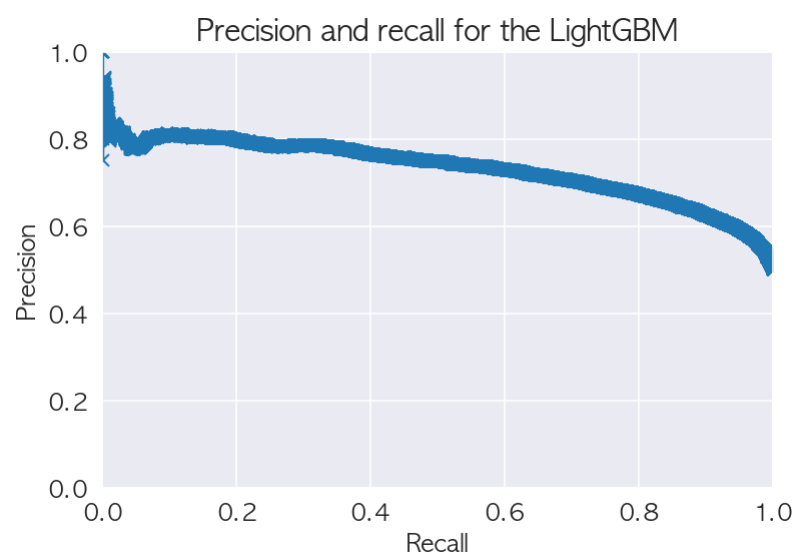
In [30]:
```python
metrics.auc(recall2, precision2)
```

Out[30]: 0.7315713319391657

In [ ]: