# 정기후원자의 이탈 예측모형 구현

작성일: 2021-08-05

작성자: 공학전문대학원 응용공학과 | 김민창 (2021-23150) | minchang.kim@snu.ac.kr

## Install packages

기본 통계 패키지를 설치합니다.

In [1]:
```
import pandas as pd
import numpy as np
import missingno as msno
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import warnings
warnings.filterwarnings('ignore')
```

기계학습 패키지를 설치합니다.

In [2]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import recall_score, confusion_matrix, precision_score, f1_scor
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
```

## Gather Data

CSV 형식으로 저장된 정기후원자의 데이터를 불러옵니다.

In [3]:
```
df = pd.read_csv('./donus_churn_data_v3.csv')
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45469 entries, 0 to 45468
Data columns (total 26 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
```

```
 0   member_code                            45469 non-null  int64
 1   is_payment_2021-06                     45469 non-null  object
 2   full_payment_rate_11months             45469 non-null  float64
 3   is_payment_2021-05                     45469 non-null  object
 4   is_payment_2021-04                     44088 non-null  object
 5   is_payment_2021-03                     42598 non-null  object
 6   is_payment_2021-02                     41073 non-null  object
 7   is_payment_2021-01                     39545 non-null  object
 8   is_payment_2020-12                     37669 non-null  object
 9   is_payment_2020-11                     36144 non-null  object
 10  is_payment_2020-10                     35028 non-null  object
 11  is_payment_2020-09                     34037 non-null  object
 12  is_payment_2020-08                     32810 non-null  object
 13  is_payment_2020-07                     31584 non-null  object
 14  duration_month_until_2021-05           45469 non-null  int64
 15  total_payments                         45469 non-null  int64
 16  monthly_amount                         45469 non-null  int64
 17  total_amount                           45469 non-null  int64
 18  payment_method                         45469 non-null  object
 19  total_promises                         45469 non-null  int64
 20  total_engagements                      45469 non-null  int64
 21  total_logins                           45469 non-null  int64
 22  general_interactions_2020-07_2021-06   45469 non-null  int64
 23  is_receipt                             45469 non-null  object
 24  sex                                    45469 non-null  object
 25  age                                    44508 non-null  float64
dtypes: float64(2), int64(9), object(15)
memory usage: 9.0+ MB
```

# Preprocessing

Null 값이 존재하는 레코드를 제거합니다.

In [5]:
```
df.dropna(subset=['age'], inplace=True)
```

학습에 이용하지 않는 불필요한 Feature를 제거합니다.

In [6]:
```
df = df.drop(['member_code'], axis=1)
```

In [7]:
```
df = df.drop(['is_payment_2021-05', 'is_payment_2021-04', 'is_payment_2021-03',
              'is_payment_2021-02', 'is_payment_2021-01', 'is_payment_2020-12', 'is_pa
              'is_payment_2020-10', 'is_payment_2020-09', 'is_payment_2020-08', 'is_pa
```

In [8]:
```
df = df.drop(['general_interactions_2020-07_2021-06'], axis = 1)
```

In [9]:
```
df = df.drop(['is_receipt'], axis=1)
```

Data type을 int로 변경합니다.

In [10]:
```
df = df.astype({"age": int})
```

Preprocessing 결과를 확인합니다.

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 44508 entries, 0 to 45468
Data columns (total 12 columns):
```

```
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   is_payment_2021-06            44508 non-null  object
 1   full_payment_rate_11months    44508 non-null  float64
 2   duration_month_until_2021-05  44508 non-null  int64
 3   total_payments                44508 non-null  int64
 4   monthly_amount                44508 non-null  int64
 5   total_amount                  44508 non-null  int64
 6   payment_method                44508 non-null  object
 7   total_promises                44508 non-null  int64
 8   total_engagements             44508 non-null  int64
 9   total_logins                  44508 non-null  int64
 10  sex                           44508 non-null  object
 11  age                           44508 non-null  int32
dtypes: float64(1), int32(1), int64(7), object(3)
memory usage: 4.2+ MB
```

In [12]:
```
df.isnull().sum()
```

Out[12]:
```
is_payment_2021-06              0
full_payment_rate_11months      0
duration_month_until_2021-05    0
total_payments                  0
monthly_amount                  0
total_amount                    0
payment_method                  0
total_promises                  0
total_engagements               0
total_logins                    0
sex                             0
age                             0
dtype: int64
```

# EDA

Feature Engineering을 진행하기 주요 데이터의 분포와 특징을 파악합니다.

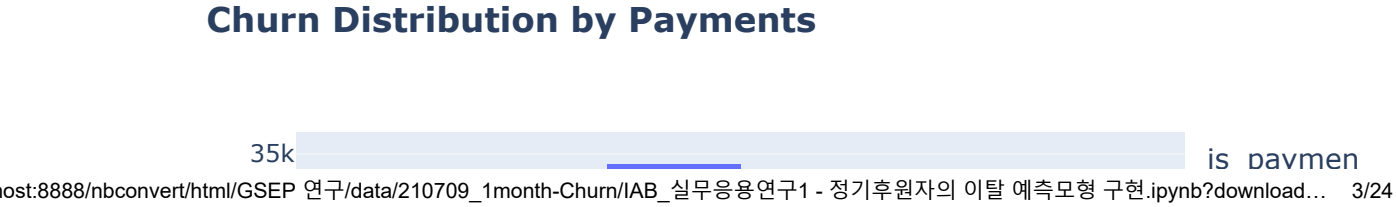
In [13]:
```
df['is_payment_2021-06'].value_counts()
```

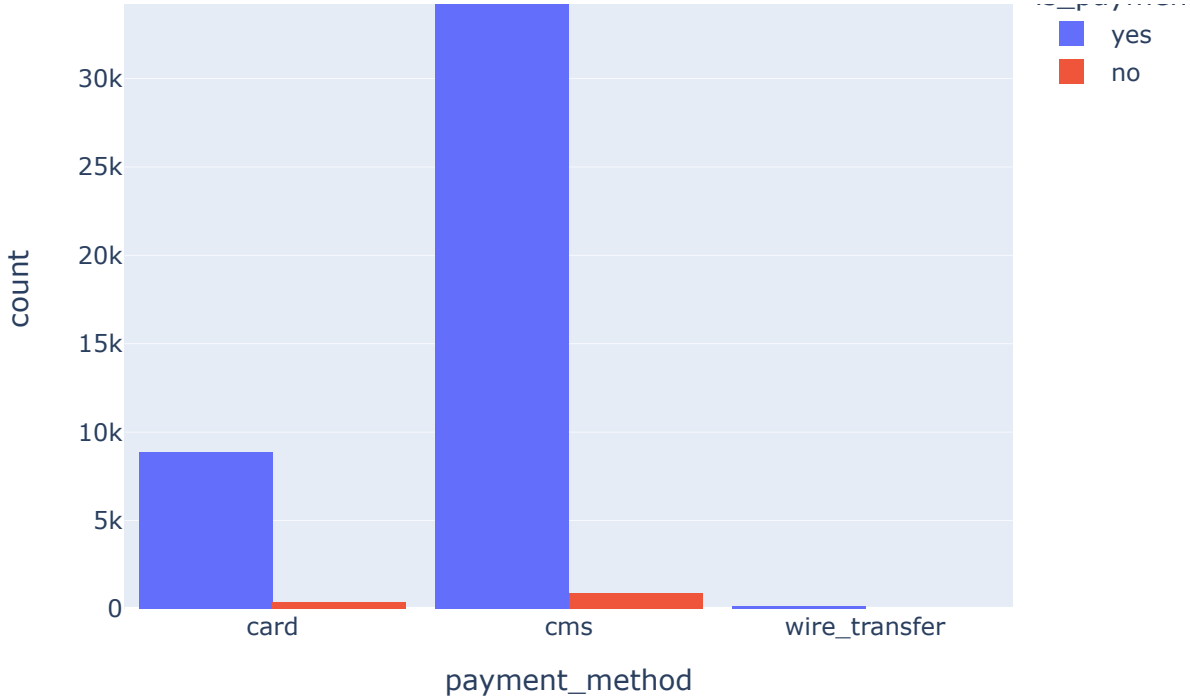
Out[13]:
```
yes    43260
no      1248
Name: is_payment_2021-06, dtype: int64
```

In [14]:
```
df['payment_method'].value_counts()
```

Out[14]:
```
cms              35317
card              9183
wire_transfer        8
Name: payment_method, dtype: int64
```

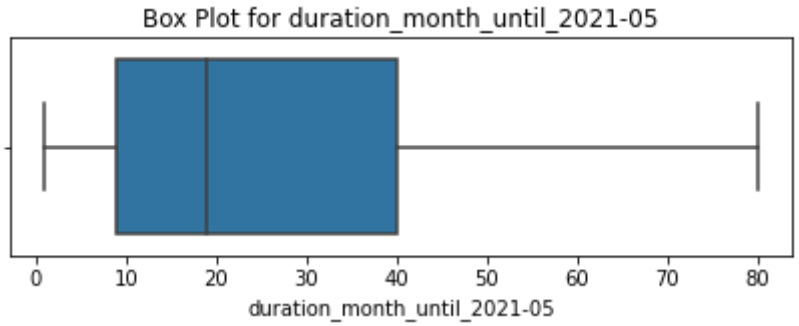
In [15]:
```
fig = px.histogram(df, x="payment_method", color="is_payment_2021-06", barmode="g    p
                   title="<b>Churn Distribution by Payments<b>")
fig.update_layout(width=700, height=500, bargap=0.1)
fig.show()
```

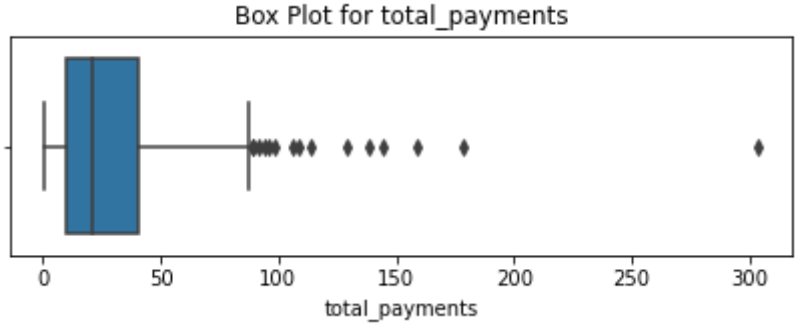


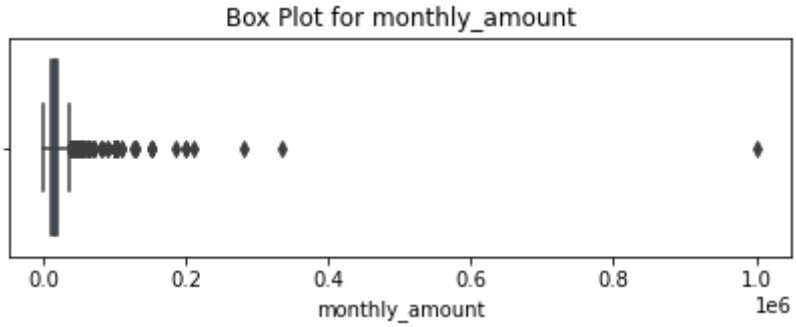


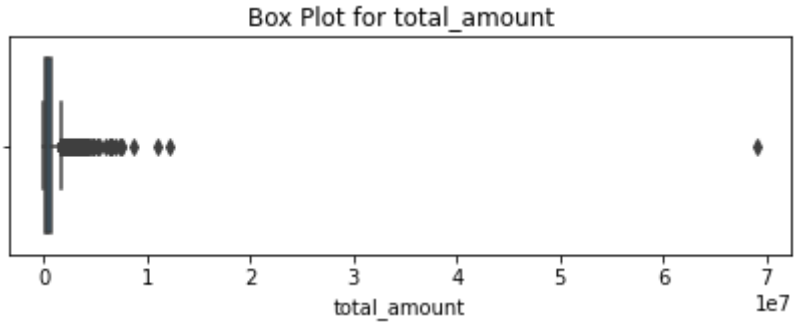

In [16]:
```
num_cols = ['duration_month_until_2021-05', 'total_payments', 'monthly_amount',
            'total_amount', 'total_promises', 'total_engagements', 'total_logins']
```

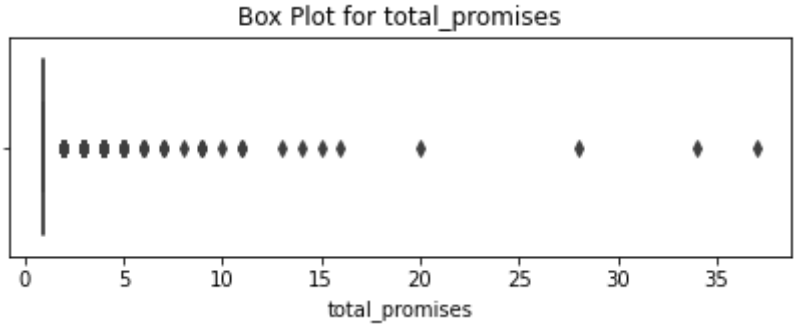
In [17]:
```
def box_plots(feature, df):
    plt.figure(figsize=(7, 2))
    plt.title("Box Plot for {}".format(feature))
    sns.boxplot(df[feature])
    plt.show()

for feat in num_cols:
    box_plots(feat, df)
```

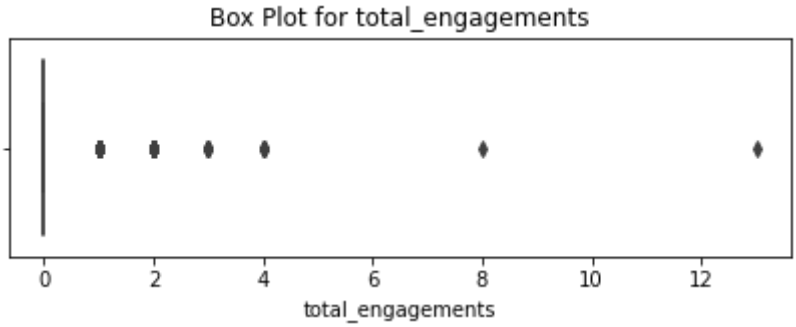


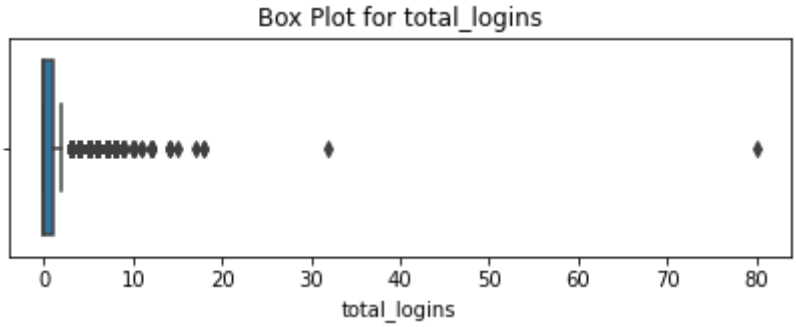





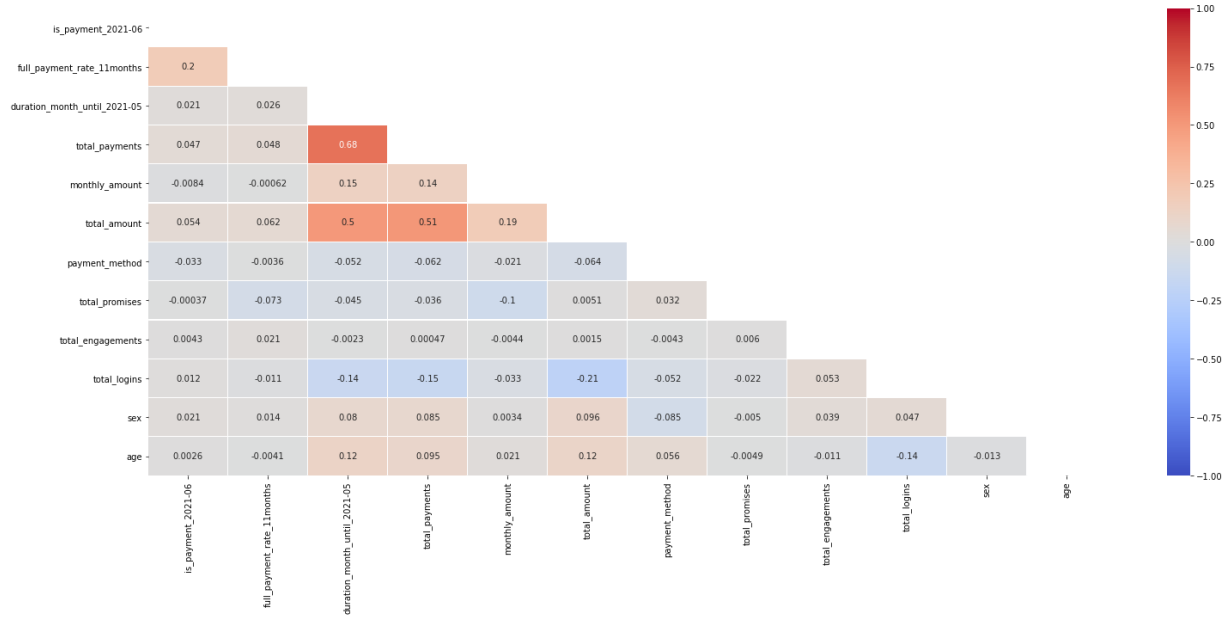




In [18]:

```
plt.figure(figsize=(25, 10))
corr = df.apply(lambda x: pd.factorize(x)[0]).corr()
mask = np.triu(np.ones_like(corr, dtype=bool))
ax = sns.heatmap(corr, mask=mask, xticklabels=corr.columns, yticklabels=corr.columns,
                 annot=True, linewidths=.2, cmap='coolwarm', vmin=-1, vmax=1)
```




# Feature Engineering

Churn은 1, Retetion은 0으로 변환합니다.

In [19]:
```
def object_to_int(dataframe_series):
    if dataframe_series.dtype == 'object':
        dataframe_series = dataframe_series.eq('no').mul(1)
    return dataframe_series
```

In [20]:
```
df = df.apply(lambda x: object_to_int(x))
```

범주형 데이터는 One Hot Encoding 형식으로 변환합니다.

In [21]:
```
df_dummy_payment_method = pd.get_dummies(df['payment_method'])
df = df.join(df_dummy_payment_method.add_prefix('p_method_'))
df = df.drop(['payment_method'], axis = 1)
```

In [22]:
```
df_dummy_sex = pd.get_dummies(df['sex'])
df = df.join(df_dummy_sex.add_prefix('sex_'))
df = df.drop(['sex'], axis = 1)
```

In [23]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 44508 entries, 0 to 45468
Data columns (total 12 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   is_payment_2021-06              44508 non-null  int32
 1   full_payment_rate_11months      44508 non-null  float64
 2   duration_month_until_2021-05    44508 non-null  int64
 3   total_payments                  44508 non-null  int64
 4   monthly_amount                  44508 non-null  int64
 5   total_amount                    44508 non-null  int64
 6   total_promises                  44508 non-null  int64
 7   total_engagements               44508 non-null  int64
 8   total_logins                    44508 non-null  int64
 9   age                             44508 non-null  int32
```

```
 10  p_method_0                  44508 non-null  uint8
 11  sex_0                       44508 non-null  uint8
dtypes: float64(1), int32(2), int64(7), uint8(2)
memory usage: 4.5 MB
```

연속형 범주를 가진 데이터의 Outliers를 제거합니다.

In [24]:
```
def modify_payments(x):
    if x >= 82: return 82
    else: return x

df['total_payments'] = df['total_payments'].apply(modify_payments)
```

In [25]:
```
def modify_monthly_amount(x):
    if x >= 100000: return 100000
    else: return x

df['monthly_amount'] = df['monthly_amount'].apply(modify_monthly_amount)
```

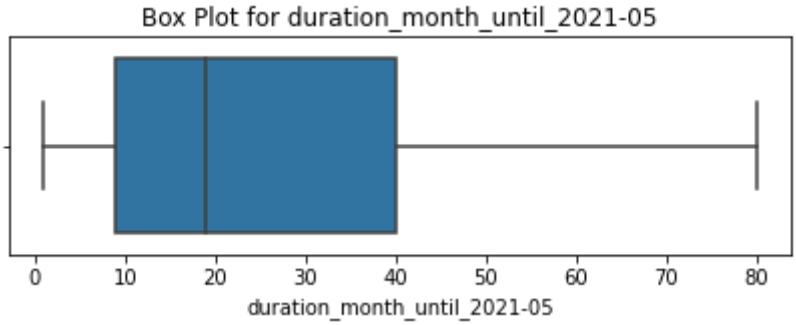
In [26]:
```
def modify_total_amount(x):
    if x >= 2500000: return 2500000
    else: return x

df['total_amount'] = df['total_amount'].apply(modify_total_amount)
```

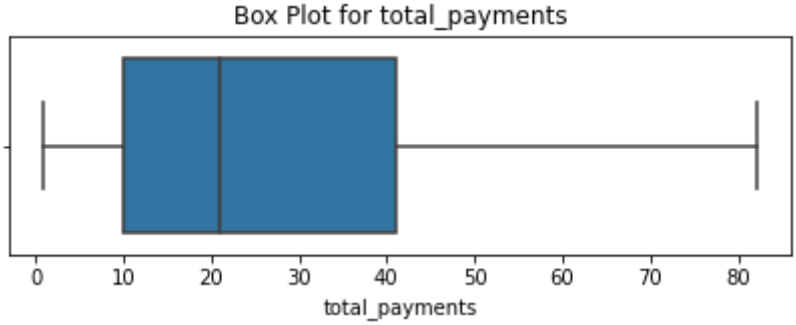
In [27]:
```
def modify_promises(x):
    if x >= 6: return 6
    else: return x

df['total_promises'] = df['total_promises'].apply(modify_promises)
```

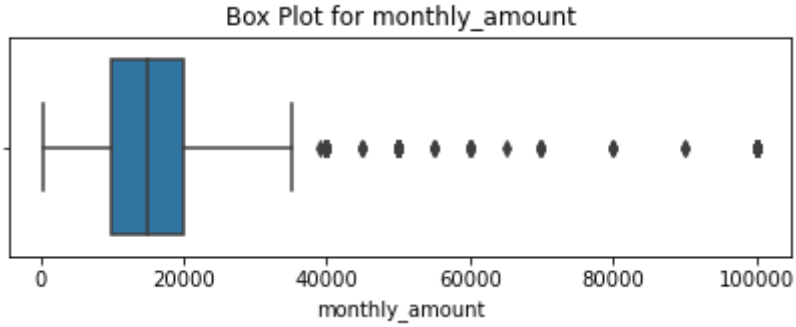
In [28]:
```
def modify_engagements(x):
    if x >= 3: return 3
    else: return x

df['total_engagements'] = df['total_engagements'].apply(modify_engagements)
```

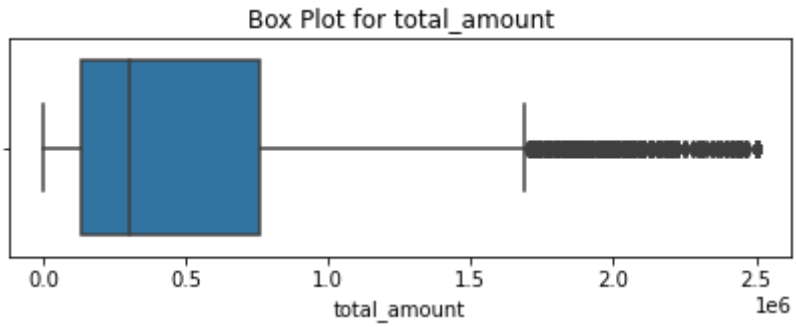
In [29]:
```
def modify_logins(x):
    if x >= 10: return 10
    else: return x

df['total_logins'] = df['total_logins'].apply(modify_logins)
```

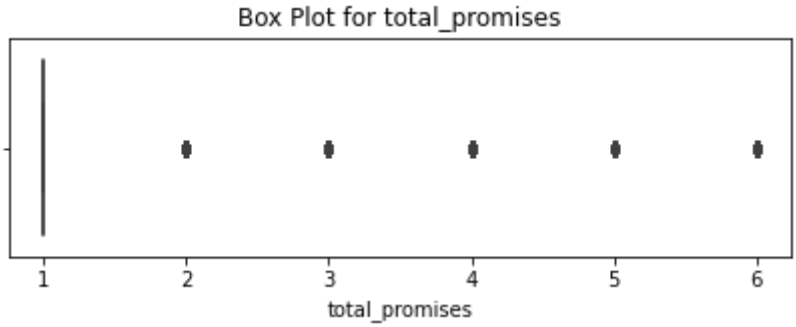
In [30]:
```
def box_plots(feature, df):
    plt.figure(figsize=(7, 2))
    plt.title("Box Plot for {}".format(feature))
    sns.boxplot(df[feature])
    plt.show()

for feat in num_cols:
    box_plots(feat, df)
```

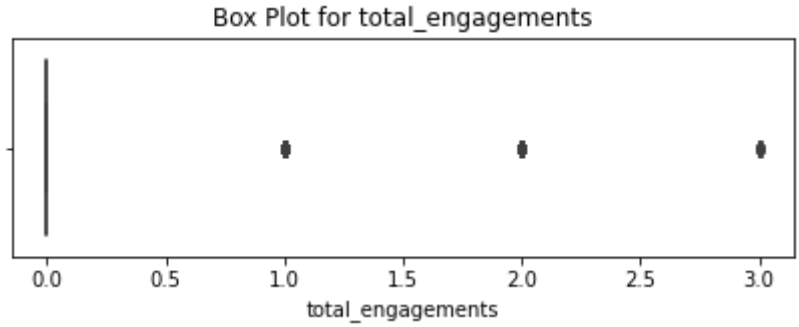

Box Plot for duration_month_until_2021-05
0
10
20
30
40
50
60
70
80
duration_month_until_2021-05

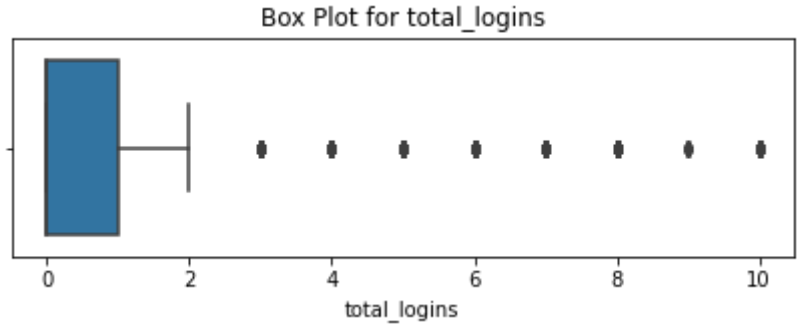

Box Plot for total_payments
0
10
20
30
40
50
60
70
80
total_payments


Box Plot for monthly_amount
0
20000
40000
60000
80000
100000
monthly_amount


Box Plot for total_amount
0.0
0.5
1.0
1.5
2.0
2.5
1e6
total_amount


Box Plot for total_promises
1
2
3
4
5
6
total_promises

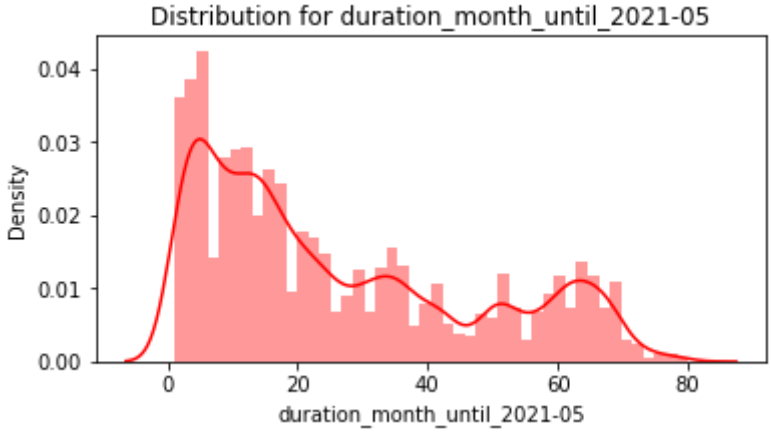


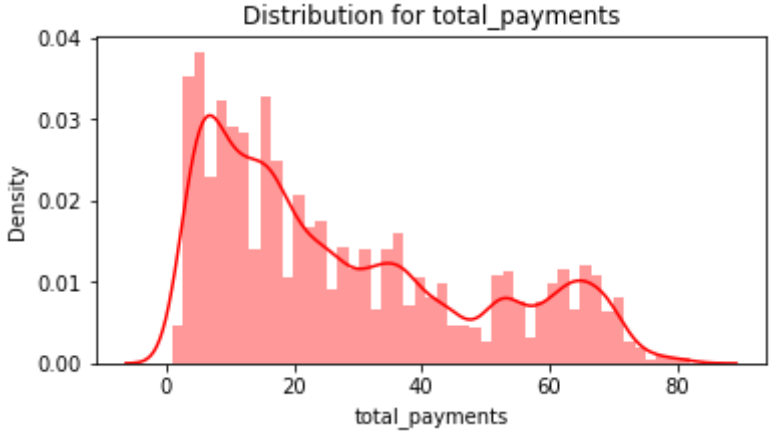



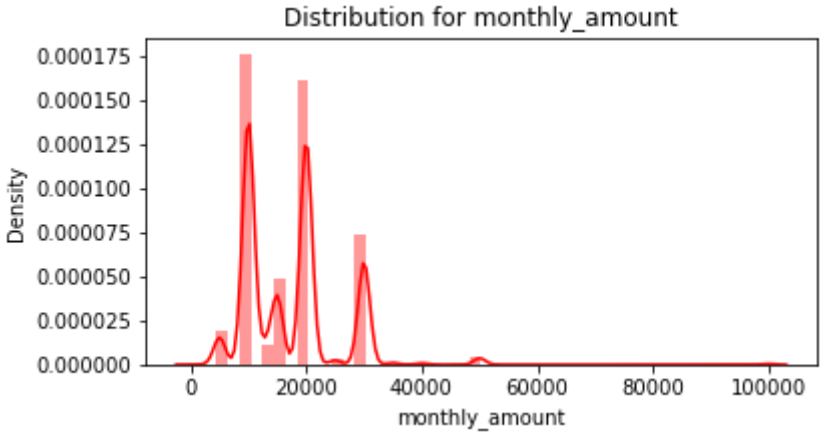

In [31]:
```
def distplot(feature, frame, color='r'):
    plt.figure(figsize=(6,3))
    plt.title("Distribution for {}".format(feature))
    ax = sns.distplot(frame[feature], color=color)
```

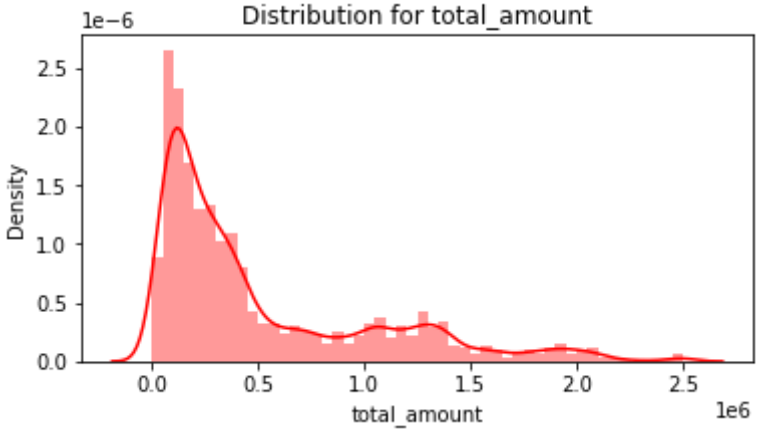
In [32]:
```
num_cols = ['duration_month_until_2021-05', 'total_payments', 'monthly_amount',
            'total_amount', 'total_promises', 'total_engagements', 'total_logins']
```

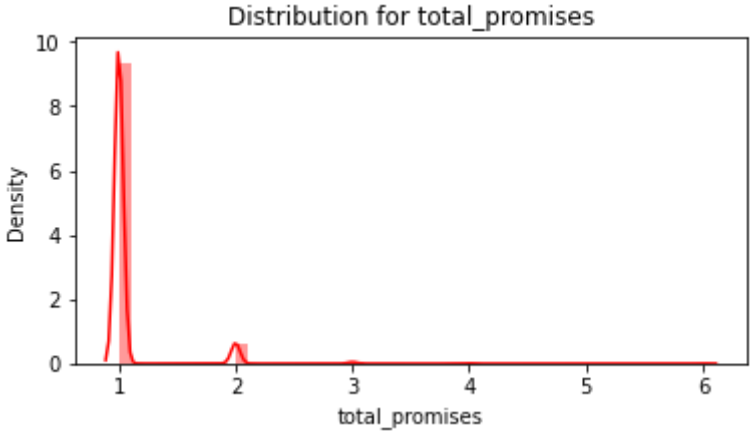
In [33]:
```
for feat in num_cols: distplot(feat, df)
```

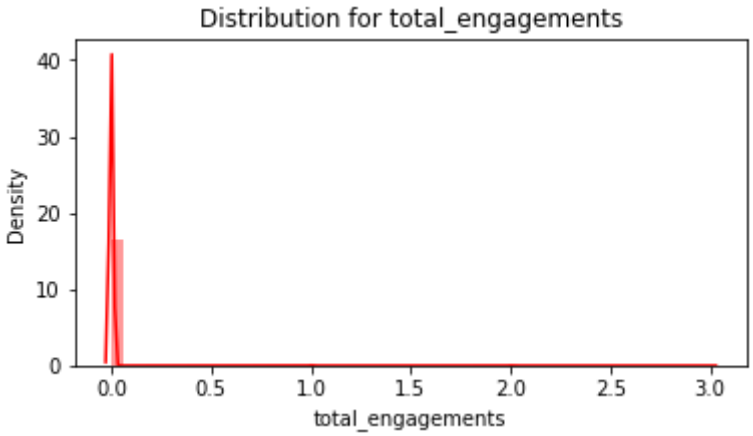


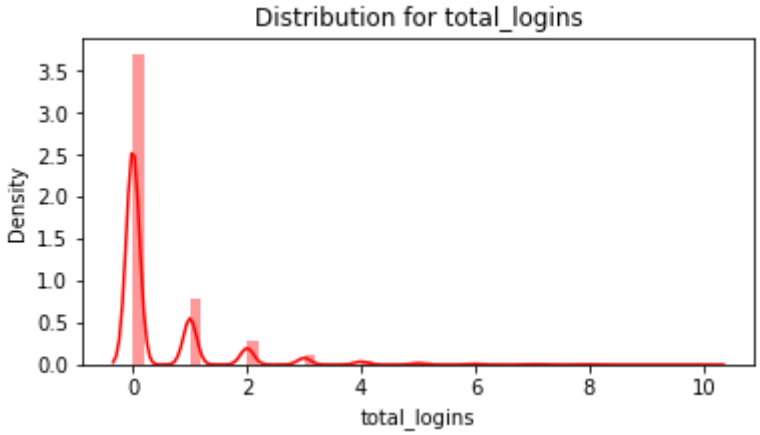




Distribution for monthly_amount
Density
0.000175
0.000150
0.000125
0.000100
0.000075
0.000050
0.000025
0.000000
0
20000
40000
60000
80000
100000
monthly_amount


Distribution for total_amount
1e−6
Density
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0
2.5
total_amount
1e6


Distribution for total_promises
Density
10
8
6
4
2
0
1
2
3
4
5
6
total_promises


Distribution for total_engagements
Density
40
30
20
10
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
total_engagements




# Split train and test set

학습을 진행하기 전 Train set과 Test set을 구분합니다.

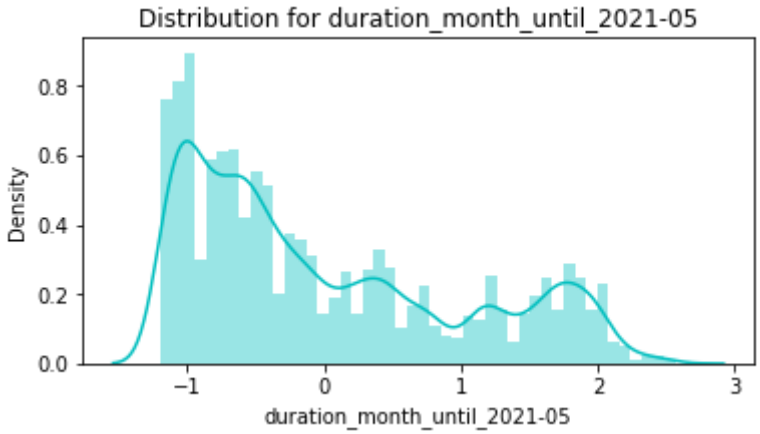
In [34]:
```
X = df.drop(columns = ['is_payment_2021-06'])
y = df['is_payment_2021-06'].values
```

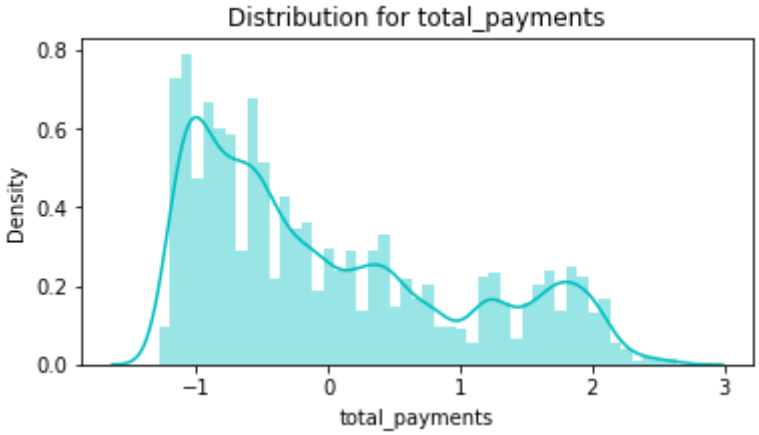
In [35]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.30, random_sta
```

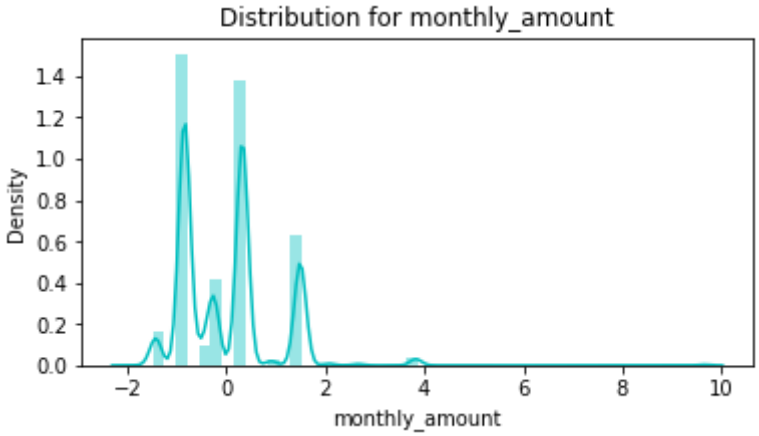
연속형 범주를 가진 데이터에 대한 표준화를 수행합니다.

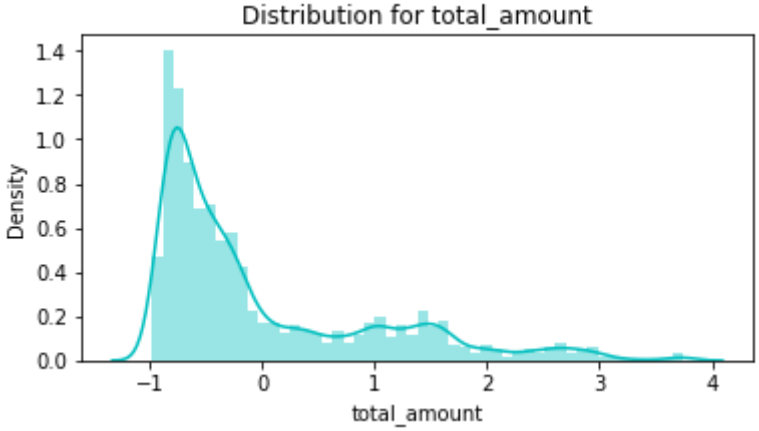
In [36]:
```
scaler = StandardScaler()
X_train[num_cols] = scaler.fit_transform(X_train[num_cols])
X_test[num_cols] = scaler.transform(X_test[num_cols])
```

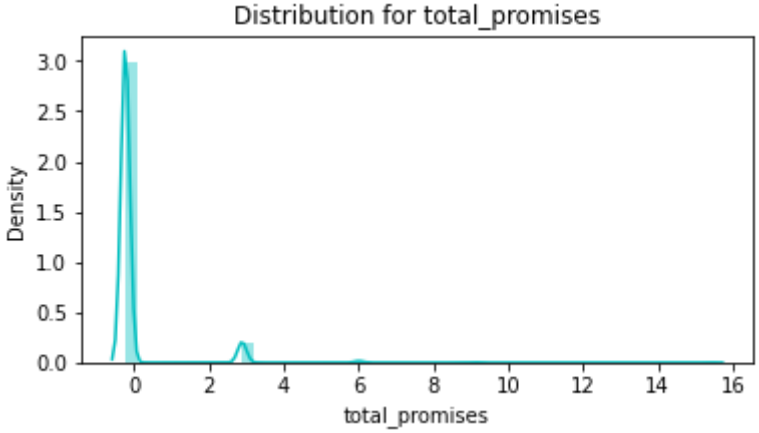
In [37]:
```
df_std = pd.DataFrame(StandardScaler().fit_transform(df[num_cols].astype('float64
for feat in num_cols: distplot(feat, df_std, color='c')
```

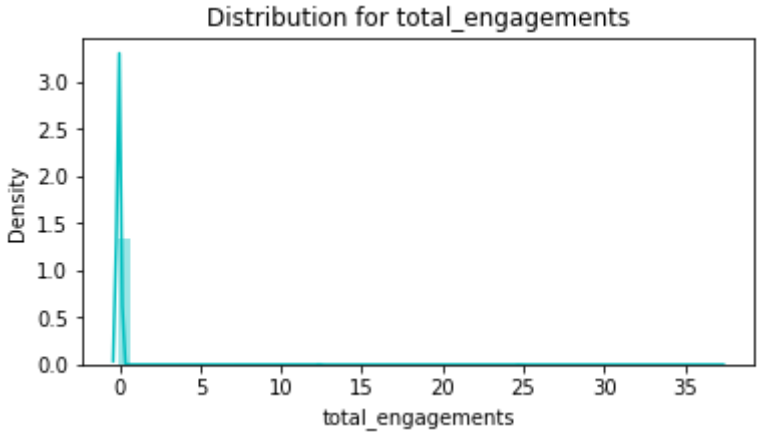


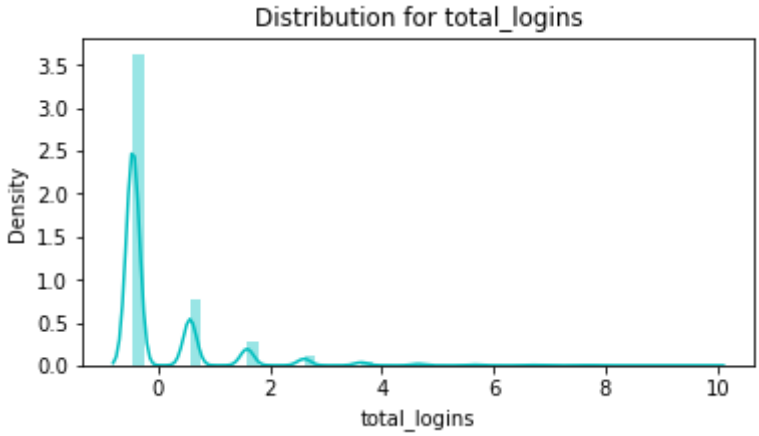

Distribution for total_payments
Density
total_payments


Distribution for monthly_amount
Density
monthly_amount


Distribution for total_amount
Density
total_amount


Distribution for total_promises
Density
total_promises




# Prediction

기계학습 Classifier로 학습을 진행한 후 Chrun 예측결과를 비교합니다. Chrun을 1, Retention은 0으로 출력됩니다. Churn의 비중이 낮은 데이터셋의 특징을 고려하여 Churn(1)의 f1-score 값을 성능을 평가하기 위한 비교 지표로 설정하였습니다.

## KNN

In [38]:
```
knn_model = KNeighborsClassifier(n_neighbors = 5)
knn_model.fit(X_train,y_train)
predicted_y = knn_model.predict(X_test)
accuracy_knn = knn_model.score(X_test,y_test)
print("KNN accuracy:",accuracy_knn)
```

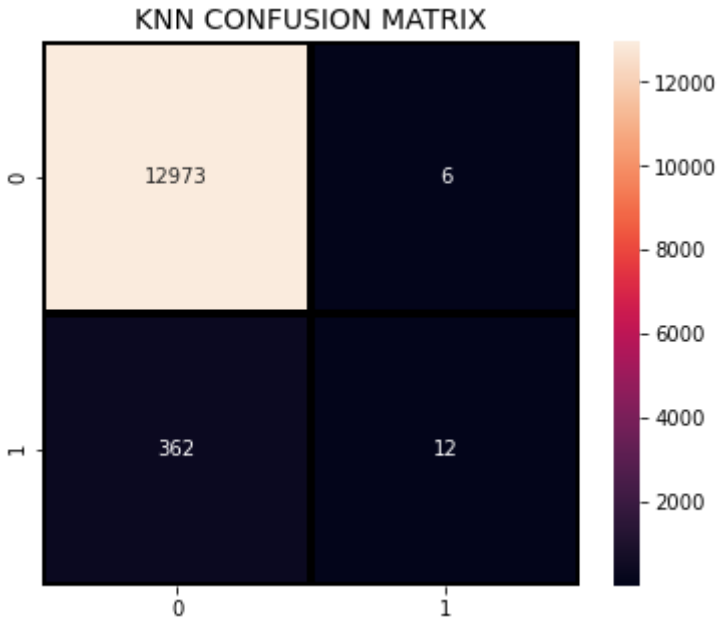
```
KNN accuracy: 0.9724406500411893
```

In [39]:
```
print(classification_report(y_test, predicted_y))
```

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.99     12979
           1       0.67      0.03      0.06       374

    accuracy                           0.97     13353
   macro avg       0.82      0.52      0.52     13353
weighted avg       0.96      0.97      0.96     13353
```

In [40]:
```
plt.figure(figsize=(6,5))
sns.heatmap(confusion_matrix(y_test, predicted_y),
                annot=True,fmt = "d",linecolor="k",linewidths=3)
```

```
plt.title("KNN CONFUSION MATRIX",fontsize=14)
plt.show()
```




## SVM

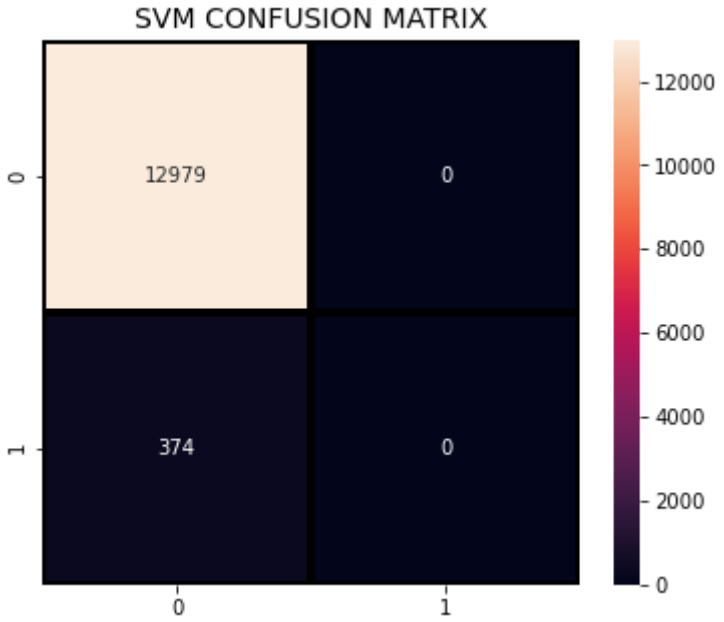
In [41]:
```
svc_model = SVC(random_state = 1)
svc_model.fit(X_train,y_train)
predict_y = svc_model.predict(X_test)
accuracy_svc = svc_model.score(X_test,y_test)
print("SVM accuracy is :",accuracy_svc)
```

```
SVM accuracy is : 0.9719913128136
```

In [42]:
```
print(classification_report(y_test, predict_y))
```

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.99     12979
           1       0.00      0.00      0.00       374

    accuracy                           0.97     13353
   macro avg       0.49      0.50      0.49     13353
weighted avg       0.94      0.97      0.96     13353
```

In [43]:
```
plt.figure(figsize=(6,5))
sns.heatmap(confusion_matrix(y_test, predict_y),
                annot=True,fmt = "d",linecolor="k",linewidths=3)
plt.title("SVM CONFUSION MATRIX",fontsize=14)
plt.show()
```




## Random Forest

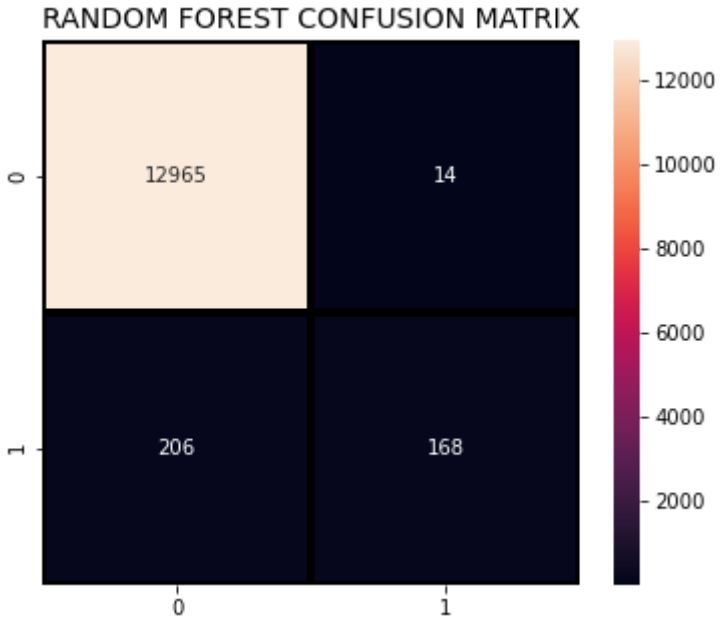
In [44]:
```
model_rf = RandomForestClassifier(n_estimators = 500 , oob_score = True, n_jobs =
                                  random_state = 50, max_features = 'auto',
                                  class_weight = 'balanced')
model_rf.fit(X_train, y_train)
prediction_test = model_rf.predict(X_test)
print (metrics.accuracy_score(y_test, prediction_test))
```

```
0.9835243016550588
```

In [45]:
```
print(classification_report(y_test, prediction_test))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99     12979
           1       0.92      0.45      0.60       374

    accuracy                           0.98     13353
   macro avg       0.95      0.72      0.80     13353
weighted avg       0.98      0.98      0.98     13353
```

In [46]:
```
plt.figure(figsize=(6,5))
sns.heatmap(confusion_matrix(y_test, prediction_test),
                annot=True,fmt = "d",linecolor="k",linewidths=3)
plt.title("RANDOM FOREST CONFUSION MATRIX",fontsize=14)
plt.show()
```




## Logistic Regression

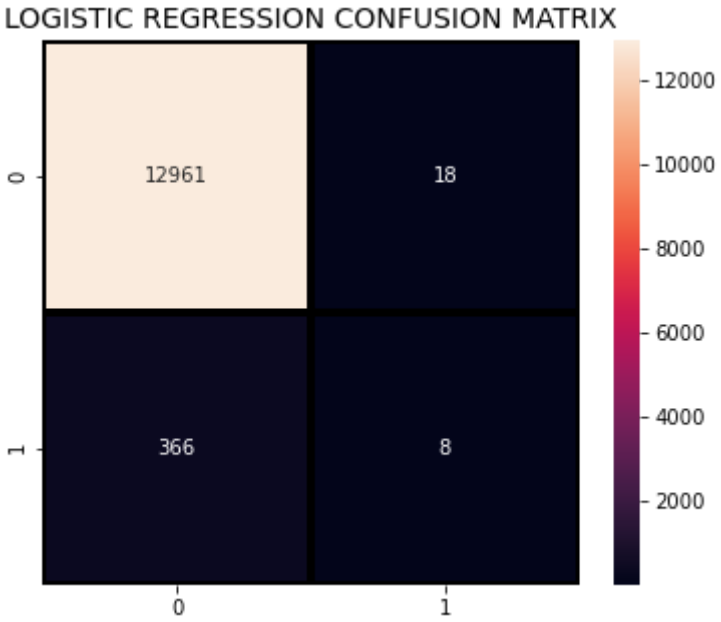
In [47]:
```
lr_model = LogisticRegression()
lr_model.fit(X_train,y_train)
accuracy_lr = lr_model.score(X_test,y_test)
print("Logistic Regression accuracy is :",accuracy_lr)
```

```
Logistic Regression accuracy is : 0.9712424174342844
```

In [48]:
```
lr_pred= lr_model.predict(X_test)
report = classification_report(y_test,lr_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.99     12979
           1       0.31      0.02      0.04       374

    accuracy                           0.97     13353
   macro avg       0.64      0.51      0.51     13353
weighted avg       0.95      0.97      0.96     13353
```

In [49]:
```
plt.figure(figsize=(6,5))
sns.heatmap(confusion_matrix(y_test, lr_pred),
                annot=True,fmt = "d",linecolor="k",linewidths=3)
plt.title("LOGISTIC REGRESSION CONFUSION MATRIX",fontsize=14)
plt.show()
```




## Decision Tree Classifier

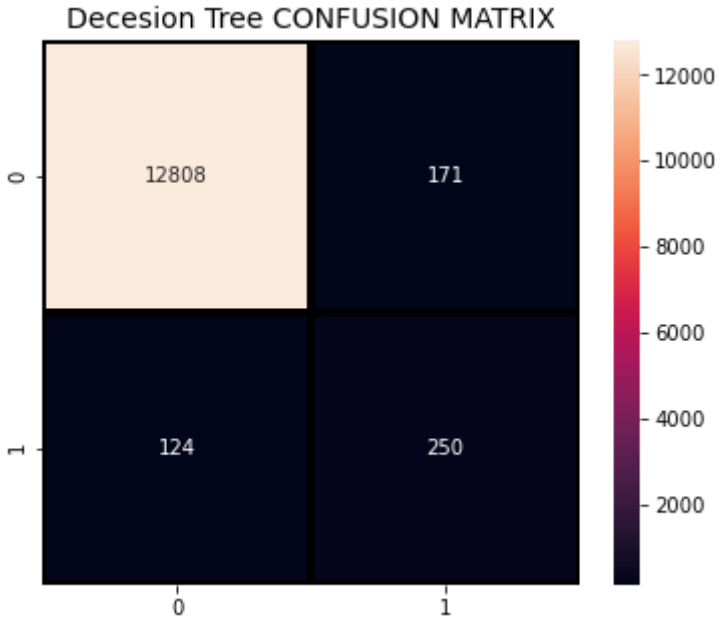
In [50]:
```
dt_model = DecisionTreeClassifier()
dt_model.fit(X_train,y_train)
predictdt_y = dt_model.predict(X_test)
accuracy_dt = dt_model.score(X_test,y_test)
print("Decision Tree accuracy is :",accuracy_dt)
```

```
Decision Tree accuracy is : 0.9779075863101925
```

In [51]:
```
print(classification_report(y_test, predictdt_y))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     12979
           1       0.59      0.67      0.63       374

    accuracy                           0.98     13353
   macro avg       0.79      0.83      0.81     13353
weighted avg       0.98      0.98      0.98     13353
```

In [52]:
```
plt.figure(figsize=(6,5))
sns.heatmap(confusion_matrix(y_test, predictdt_y),
                annot=True, fmt = "d",linecolor="k",linewidths=3)
plt.title("Decesion Tree CONFUSION MATRIX",fontsize=14)
plt.show()
```




## AdaBoost Classifier

In [53]:
```
a_model = AdaBoostClassifier()
a_model.fit(X_train,y_train)
a_preds = a_model.predict(X_test)
print("AdaBoost Classifier accuracy")
metrics.accuracy_score(y_test, a_preds)
```

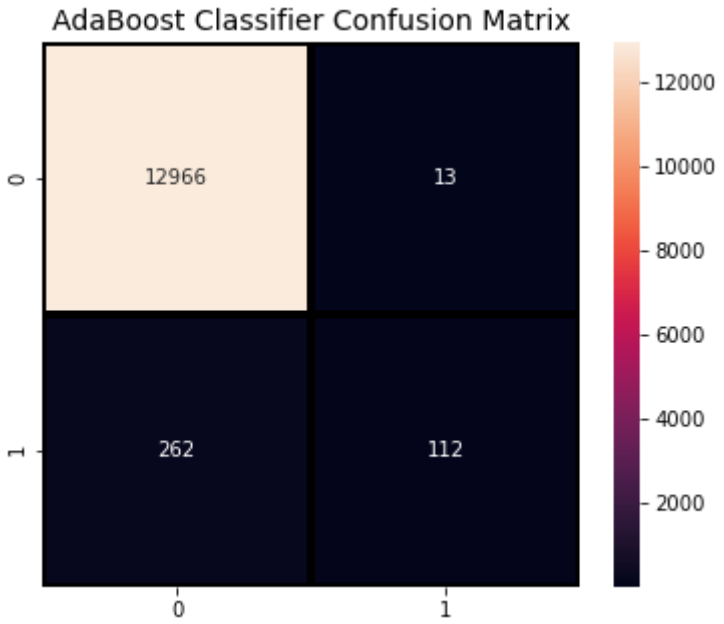
```
AdaBoost Classifier accuracy
```

Out[53]:
```
0.9794053770688235
```

In [54]:
```
print(classification_report(y_test, a_preds))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99     12979
           1       0.90      0.30      0.45       374

    accuracy                           0.98     13353
   macro avg       0.94      0.65      0.72     13353
weighted avg       0.98      0.98      0.97     13353
```

In [55]:
```
plt.figure(figsize=(6,5))
sns.heatmap(confusion_matrix(y_test, a_preds),
                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title("AdaBoost Classifier Confusion Matrix",fontsize=14)
plt.show()
```




## Gradient Boosting Classifier

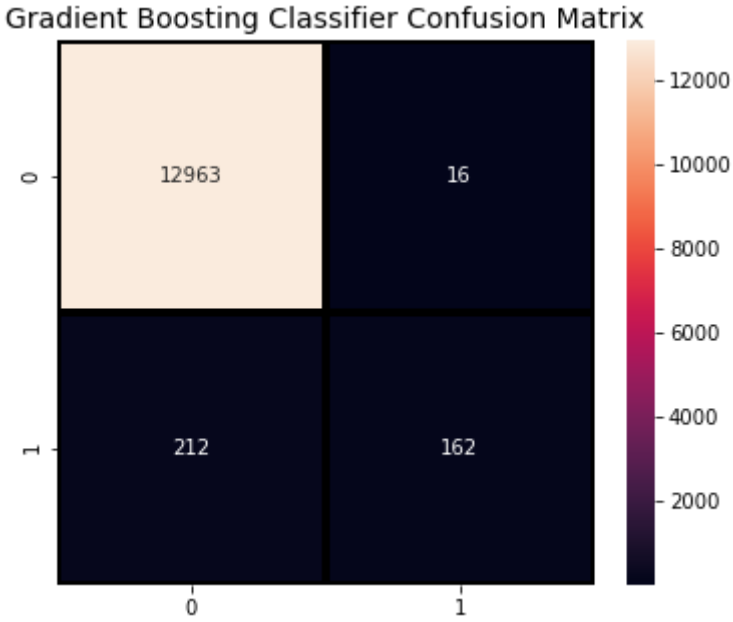
In [56]:
```
gb = GradientBoostingClassifier()
gb.fit(X_train, y_train)
gb_pred = gb.predict(X_test)
print("Gradient Boosting Classifier", accuracy_score(y_test, gb_pred))
```

```
Gradient Boosting Classifier 0.9829251853516063
```

In [57]:
```
print(classification_report(y_test, gb_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99     12979
           1       0.91      0.43      0.59       374

    accuracy                           0.98     13353
   macro avg       0.95      0.72      0.79     13353
weighted avg       0.98      0.98      0.98     13353
```

In [58]:
```
plt.figure(figsize=(6,5))
sns.heatmap(confusion_matrix(y_test, gb_pred),
                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title("Gradient Boosting Classifier Confusion Matrix",fontsize=14)
plt.show()
```




# Prediction (Additional)

IAB 최종발표 후의 피드백을 반영하여 LightGBM Classifier로 추가 예측을 진행하였습니다. 적용 결과 LightGBM Classifier을 통해 가장 높은 f1-score를 얻을 수 있었습니다.

## LightGBM Classifier

Scikit Learn 라이브러리의 GridSearchCV 함수를 이용하여 AUC를 기준으로 Hyperparameters 최적화를 진행하였습니다.

In [ ]:
```
lgbm = LGBMClassifier(n_estimators=200)
evals = [(X_train, y_train), (X_test, y_test)]
params = {'num_leaves': [100, 200],
          'learning_rate': [0.05, 0.04],
          'subsample': [0.6, 0.8],
          'max_depth': [100, 200],
          'min_child_samples': [50, 100]}

gridcv = GridSearchCV(lgbm, param_grid=params, cv=3)
gridcv.fit(X_train, y_train, early_stopping_rounds=30, eval_metric="auc", eval_set=ev
lgbm_roc_score = roc_auc_score(y_test, gridcv.predict_proba(X_test)[:, 1], average='m

print('GridSearchCV Best Parameters:', gridcv.best_params_)
print('ROC AUC: {0:.4f}'.format(lgbm_roc_score))
```

최적화를 통해 도출된 Hyperparameters를 적용하여 예측을 수행하였습니다.

In [59]:
```
lgbm = LGBMClassifier(n_estimators=1000, learning_rate=0.04, num_leaves=100,
                      max_depth=100, min_child_samples=50, subsample=0.6)
evals = [(X_test, y_test)]
lgbm.fit(X_train, y_train, early_stopping_rounds=100, eval_metric="auc", eval_set=eva
lgbm_pred = lgbm.predict(X_test)
lgbm_pred_proba = lgbm.predict_proba(X_test)[:, 1]
```

```
[1]     valid_0's auc: 0.946765 valid_0's binary_logloss: 0.112062
Training until validation scores don't improve for 100 rounds
[2]     valid_0's auc: 0.958685 valid_0's binary_logloss: 0.104944
[3]     valid_0's auc: 0.966483 valid_0's binary_logloss: 0.0996811
[4]     valid_0's auc: 0.96735  valid_0's binary_logloss: 0.0953809
[5]     valid_0's auc: 0.969175 valid_0's binary_logloss: 0.0911006
```

```
[6]	valid_0's auc: 0.969865 valid_0's binary_logloss: 0.0876702
[7]	valid_0's auc: 0.970762 valid_0's binary_logloss: 0.084705
[8]	valid_0's auc: 0.971157 valid_0's binary_logloss: 0.0820162
[9]	valid_0's auc: 0.971786 valid_0's binary_logloss: 0.0796898
[10]	valid_0's auc: 0.972507 valid_0's binary_logloss: 0.077409
[11]	valid_0's auc: 0.972436 valid_0's binary_logloss: 0.0755977
[12]	valid_0's auc: 0.97282  valid_0's binary_logloss: 0.0738125
[13]	valid_0's auc: 0.972748 valid_0's binary_logloss: 0.0722742
[14]	valid_0's auc: 0.972882 valid_0's binary_logloss: 0.0708043
[15]	valid_0's auc: 0.973466 valid_0's binary_logloss: 0.069444
[16]	valid_0's auc: 0.973822 valid_0's binary_logloss: 0.0681008
[17]	valid_0's auc: 0.974    valid_0's binary_logloss: 0.0668693
[18]	valid_0's auc: 0.974903 valid_0's binary_logloss: 0.0657054
[19]	valid_0's auc: 0.975024 valid_0's binary_logloss: 0.0645712
[20]	valid_0's auc: 0.975484 valid_0's binary_logloss: 0.0634405
[21]	valid_0's auc: 0.97557  valid_0's binary_logloss: 0.0624404
[22]	valid_0's auc: 0.975964 valid_0's binary_logloss: 0.0614657
[23]	valid_0's auc: 0.976061 valid_0's binary_logloss: 0.0606262
[24]	valid_0's auc: 0.975861 valid_0's binary_logloss: 0.0599034
[25]	valid_0's auc: 0.975791 valid_0's binary_logloss: 0.0591636
[26]	valid_0's auc: 0.975834 valid_0's binary_logloss: 0.0584376
[27]	valid_0's auc: 0.975829 valid_0's binary_logloss: 0.0577528
[28]	valid_0's auc: 0.975941 valid_0's binary_logloss: 0.057052
[29]	valid_0's auc: 0.976054 valid_0's binary_logloss: 0.0564124
[30]	valid_0's auc: 0.976347 valid_0's binary_logloss: 0.0557345
[31]	valid_0's auc: 0.976935 valid_0's binary_logloss: 0.0551118
[32]	valid_0's auc: 0.977034 valid_0's binary_logloss: 0.0545569
[33]	valid_0's auc: 0.977156 valid_0's binary_logloss: 0.0540221
[34]	valid_0's auc: 0.977424 valid_0's binary_logloss: 0.0534644
[35]	valid_0's auc: 0.977495 valid_0's binary_logloss: 0.0530155
[36]	valid_0's auc: 0.977565 valid_0's binary_logloss: 0.0525443
[37]	valid_0's auc: 0.977666 valid_0's binary_logloss: 0.0521732
[38]	valid_0's auc: 0.977834 valid_0's binary_logloss: 0.0518066
[39]	valid_0's auc: 0.977652 valid_0's binary_logloss: 0.0514505
[40]	valid_0's auc: 0.977585 valid_0's binary_logloss: 0.0511397
[41]	valid_0's auc: 0.977707 valid_0's binary_logloss: 0.0507459
[42]	valid_0's auc: 0.977856 valid_0's binary_logloss: 0.0504009
[43]	valid_0's auc: 0.977876 valid_0's binary_logloss: 0.0500838
[44]	valid_0's auc: 0.977758 valid_0's binary_logloss: 0.0498079
[45]	valid_0's auc: 0.977891 valid_0's binary_logloss: 0.0495081
[46]	valid_0's auc: 0.977847 valid_0's binary_logloss: 0.0492145
[47]	valid_0's auc: 0.978054 valid_0's binary_logloss: 0.0488923
[48]	valid_0's auc: 0.97817  valid_0's binary_logloss: 0.0486038
[49]	valid_0's auc: 0.978313 valid_0's binary_logloss: 0.0483056
[50]	valid_0's auc: 0.978238 valid_0's binary_logloss: 0.0480424
[51]	valid_0's auc: 0.978362 valid_0's binary_logloss: 0.0477929
[52]	valid_0's auc: 0.978537 valid_0's binary_logloss: 0.0475288
[53]	valid_0's auc: 0.978577 valid_0's binary_logloss: 0.0473291
[54]	valid_0's auc: 0.97865  valid_0's binary_logloss: 0.0471022
[55]	valid_0's auc: 0.978636 valid_0's binary_logloss: 0.0468816
[56]	valid_0's auc: 0.978788 valid_0's binary_logloss: 0.0466654
[57]	valid_0's auc: 0.978789 valid_0's binary_logloss: 0.0464856
[58]	valid_0's auc: 0.978745 valid_0's binary_logloss: 0.0463094
[59]	valid_0's auc: 0.978753 valid_0's binary_logloss: 0.046155
[60]	valid_0's auc: 0.978693 valid_0's binary_logloss: 0.0460178
[61]	valid_0's auc: 0.978817 valid_0's binary_logloss: 0.0458658
[62]	valid_0's auc: 0.978751 valid_0's binary_logloss: 0.0457303
[63]	valid_0's auc: 0.978885 valid_0's binary_logloss: 0.0455978
[64]	valid_0's auc: 0.978834 valid_0's binary_logloss: 0.0455054
[65]	valid_0's auc: 0.978835 valid_0's binary_logloss: 0.0453964
[66]	valid_0's auc: 0.978757 valid_0's binary_logloss: 0.0453126
[67]	valid_0's auc: 0.978829 valid_0's binary_logloss: 0.0452162
[68]	valid_0's auc: 0.978729 valid_0's binary_logloss: 0.0451166
[69]	valid_0's auc: 0.978677 valid_0's binary_logloss: 0.0450185
[70]	valid_0's auc: 0.978528 valid_0's binary_logloss: 0.0449543
[71]	valid_0's auc: 0.978487 valid_0's binary_logloss: 0.0448698
[72]	valid_0's auc: 0.978234 valid_0's binary_logloss: 0.0448287
[73]	valid_0's auc: 0.978007 valid_0's binary_logloss: 0.0448154
[74]	valid_0's auc: 0.978065 valid_0's binary_logloss: 0.0447227
```

```
[75]    valid_0's auc: 0.978165 valid_0's binary_logloss: 0.0446544
[76]    valid_0's auc: 0.977985 valid_0's binary_logloss: 0.0446138
[77]    valid_0's auc: 0.978089 valid_0's binary_logloss: 0.0445496
[78]    valid_0's auc: 0.977996 valid_0's binary_logloss: 0.04451
[79]    valid_0's auc: 0.977945 valid_0's binary_logloss: 0.0444694
[80]    valid_0's auc: 0.97801  valid_0's binary_logloss: 0.0444639
[81]    valid_0's auc: 0.978023 valid_0's binary_logloss: 0.0444305
[82]    valid_0's auc: 0.978085 valid_0's binary_logloss: 0.0443515
[83]    valid_0's auc: 0.977966 valid_0's binary_logloss: 0.0443411
[84]    valid_0's auc: 0.978007 valid_0's binary_logloss: 0.0442513
[85]    valid_0's auc: 0.97794  valid_0's binary_logloss: 0.0442536
[86]    valid_0's auc: 0.97798  valid_0's binary_logloss: 0.0441817
[87]    valid_0's auc: 0.977994 valid_0's binary_logloss: 0.0441324
[88]    valid_0's auc: 0.977915 valid_0's binary_logloss: 0.0441441
[89]    valid_0's auc: 0.977769 valid_0's binary_logloss: 0.0441311
[90]    valid_0's auc: 0.977816 valid_0's binary_logloss: 0.0440661
[91]    valid_0's auc: 0.977713 valid_0's binary_logloss: 0.0439931
[92]    valid_0's auc: 0.977624 valid_0's binary_logloss: 0.0439364
[93]    valid_0's auc: 0.977587 valid_0's binary_logloss: 0.0438968
[94]    valid_0's auc: 0.977603 valid_0's binary_logloss: 0.0438647
[95]    valid_0's auc: 0.977735 valid_0's binary_logloss: 0.0437841
[96]    valid_0's auc: 0.977617 valid_0's binary_logloss: 0.04372
[97]    valid_0's auc: 0.977551 valid_0's binary_logloss: 0.0437043
[98]    valid_0's auc: 0.977545 valid_0's binary_logloss: 0.0436638
[99]    valid_0's auc: 0.977465 valid_0's binary_logloss: 0.0436786
[100]   valid_0's auc: 0.977578 valid_0's binary_logloss: 0.0435953
[101]   valid_0's auc: 0.977521 valid_0's binary_logloss: 0.0435874
[102]   valid_0's auc: 0.977369 valid_0's binary_logloss: 0.0436275
[103]   valid_0's auc: 0.977221 valid_0's binary_logloss: 0.0436663
[104]   valid_0's auc: 0.977232 valid_0's binary_logloss: 0.0436519
[105]   valid_0's auc: 0.977365 valid_0's binary_logloss: 0.0435725
[106]   valid_0's auc: 0.977247 valid_0's binary_logloss: 0.0436048
[107]   valid_0's auc: 0.977288 valid_0's binary_logloss: 0.0435726
[108]   valid_0's auc: 0.977257 valid_0's binary_logloss: 0.0435563
[109]   valid_0's auc: 0.977308 valid_0's binary_logloss: 0.0435755
[110]   valid_0's auc: 0.977362 valid_0's binary_logloss: 0.043583
[111]   valid_0's auc: 0.977315 valid_0's binary_logloss: 0.0435517
[112]   valid_0's auc: 0.977352 valid_0's binary_logloss: 0.0435421
[113]   valid_0's auc: 0.977259 valid_0's binary_logloss: 0.0435662
[114]   valid_0's auc: 0.97727  valid_0's binary_logloss: 0.0435196
[115]   valid_0's auc: 0.977086 valid_0's binary_logloss: 0.0435644
[116]   valid_0's auc: 0.976924 valid_0's binary_logloss: 0.0435805
[117]   valid_0's auc: 0.976735 valid_0's binary_logloss: 0.043617
[118]   valid_0's auc: 0.976719 valid_0's binary_logloss: 0.0435845
[119]   valid_0's auc: 0.976692 valid_0's binary_logloss: 0.0435777
[120]   valid_0's auc: 0.976673 valid_0's binary_logloss: 0.0435385
[121]   valid_0's auc: 0.976625 valid_0's binary_logloss: 0.043546
[122]   valid_0's auc: 0.97663  valid_0's binary_logloss: 0.043526
[123]   valid_0's auc: 0.976633 valid_0's binary_logloss: 0.0434965
[124]   valid_0's auc: 0.976647 valid_0's binary_logloss: 0.0435075
[125]   valid_0's auc: 0.976527 valid_0's binary_logloss: 0.0435361
[126]   valid_0's auc: 0.976551 valid_0's binary_logloss: 0.0435512
[127]   valid_0's auc: 0.976448 valid_0's binary_logloss: 0.0435627
[128]   valid_0's auc: 0.976323 valid_0's binary_logloss: 0.0435707
[129]   valid_0's auc: 0.976254 valid_0's binary_logloss: 0.0436128
[130]   valid_0's auc: 0.976316 valid_0's binary_logloss: 0.0435713
[131]   valid_0's auc: 0.976405 valid_0's binary_logloss: 0.043531
[132]   valid_0's auc: 0.976358 valid_0's binary_logloss: 0.0435221
[133]   valid_0's auc: 0.976296 valid_0's binary_logloss: 0.043542
[134]   valid_0's auc: 0.976343 valid_0's binary_logloss: 0.0435125
[135]   valid_0's auc: 0.976101 valid_0's binary_logloss: 0.0435709
[136]   valid_0's auc: 0.97607  valid_0's binary_logloss: 0.0435884
[137]   valid_0's auc: 0.976196 valid_0's binary_logloss: 0.0435681
[138]   valid_0's auc: 0.976111 valid_0's binary_logloss: 0.0435799
[139]   valid_0's auc: 0.975969 valid_0's binary_logloss: 0.0435988
[140]   valid_0's auc: 0.975967 valid_0's binary_logloss: 0.0436158
[141]   valid_0's auc: 0.975829 valid_0's binary_logloss: 0.0436589
[142]   valid_0's auc: 0.975863 valid_0's binary_logloss: 0.0436868
[143]   valid_0's auc: 0.975986 valid_0's binary_logloss: 0.0436338
```

```
[144]   valid_0's auc: 0.975921 valid_0's binary_logloss: 0.0436482
[145]   valid_0's auc: 0.975966 valid_0's binary_logloss: 0.04366
[146]   valid_0's auc: 0.976045 valid_0's binary_logloss: 0.0436673
[147]   valid_0's auc: 0.975909 valid_0's binary_logloss: 0.0436919
[148]   valid_0's auc: 0.975853 valid_0's binary_logloss: 0.043697
[149]   valid_0's auc: 0.975777 valid_0's binary_logloss: 0.0437129
[150]   valid_0's auc: 0.975734 valid_0's binary_logloss: 0.0437103
[151]   valid_0's auc: 0.9756   valid_0's binary_logloss: 0.0437328
[152]   valid_0's auc: 0.975521 valid_0's binary_logloss: 0.0437783
[153]   valid_0's auc: 0.97558  valid_0's binary_logloss: 0.0437751
[154]   valid_0's auc: 0.975474 valid_0's binary_logloss: 0.0438187
[155]   valid_0's auc: 0.97544  valid_0's binary_logloss: 0.0438365
[156]   valid_0's auc: 0.975441 valid_0's binary_logloss: 0.0438548
[157]   valid_0's auc: 0.975391 valid_0's binary_logloss: 0.0438921
[158]   valid_0's auc: 0.975342 valid_0's binary_logloss: 0.043939
[159]   valid_0's auc: 0.975197 valid_0's binary_logloss: 0.0439776
[160]   valid_0's auc: 0.975085 valid_0's binary_logloss: 0.0440365
[161]   valid_0's auc: 0.975012 valid_0's binary_logloss: 0.0440623
[162]   valid_0's auc: 0.974951 valid_0's binary_logloss: 0.0441027
[163]   valid_0's auc: 0.974878 valid_0's binary_logloss: 0.0441477
Early stopping, best iteration is:
[63]    valid_0's auc: 0.978885 valid_0's binary_logloss: 0.0455978
```

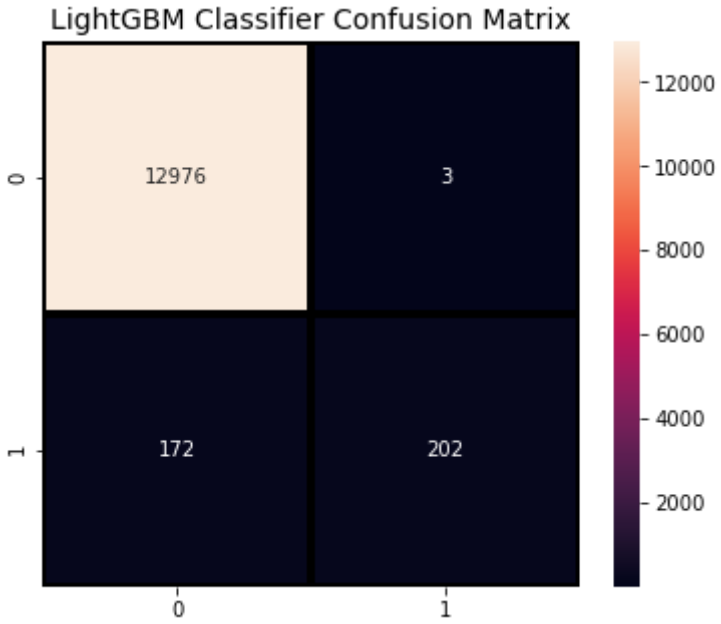
In [60]:

```
print(classification_report(y_test, lgbm_pred))
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99     12979
           1       0.99      0.54      0.70       374

    accuracy                           0.99     13353
   macro avg       0.99      0.77      0.85     13353
weighted avg       0.99      0.99      0.99     13353
```

In [61]:

```
plt.figure(figsize=(6,5))
sns.heatmap(confusion_matrix(y_test, lgbm_pred),
                annot=True,fmt = "d",linecolor="k",linewidths=3)
plt.title("LightGBM Classifier Confusion Matrix",fontsize=14)
plt.show()
```




In [62]:

```
fpr, tpr, thresholds = roc_curve(y_test, lgbm_pred_proba)
plt.plot([0, 1], [0, 1], 'k--' )
plt.plot(fpr, tpr, label='LightBGM Classifier',color = "r")
```

```
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('LightBGM Classifier ROC Curve',fontsize=16)
plt.show();
```