In [1]:
```
import pandas as pd
import numpy as np
from datetime import datetime
from lightgbm import LGBMClassifier
import warnings
warnings.filterwarnings('ignore')
warnings.simplefilter(action='ignore', category=FutureWarning)

import matplotlib.pyplot as plt
import seaborn as sns
import time
from imblearn.over_sampling import SMOTE
from sklearn.metrics import confusion_matrix
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split, TimeSeriesSplit, RandomizedSearchC
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report,accuracy_score, plot_confusion_matrix,
```

In [2]:
```
df = pd.read_csv('smote.csv')
df.tail()
```

Out[2]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9
568625	1.084762	-0.063846	1.693838	-4.245310	2.295877	-1.212742	-2.054768	-1.860351	0.811518	-1.730054
568626	-0.637311	-1.711814	0.456246	-1.120049	0.609640	-0.859496	-0.304985	-1.335793	0.017532	-0.016575
568627	-0.480862	-0.528259	1.141664	-0.850414	2.055184	0.068474	-0.460563	-1.022289	0.539480	-1.365581
568628	1.201490	-0.711715	2.790795	-5.314459	4.956698	-0.224579	-1.956847	-2.172370	1.186167	-3.607043
568629	0.485418	-2.360354	1.863733	-4.112487	2.079790	-1.182344	-1.466585	-3.022281	-0.239394	-0.269545

5 rows × 31 columns

In [3]:
```
df = df.rename(columns={'Class':'label'})
df.tail()
```

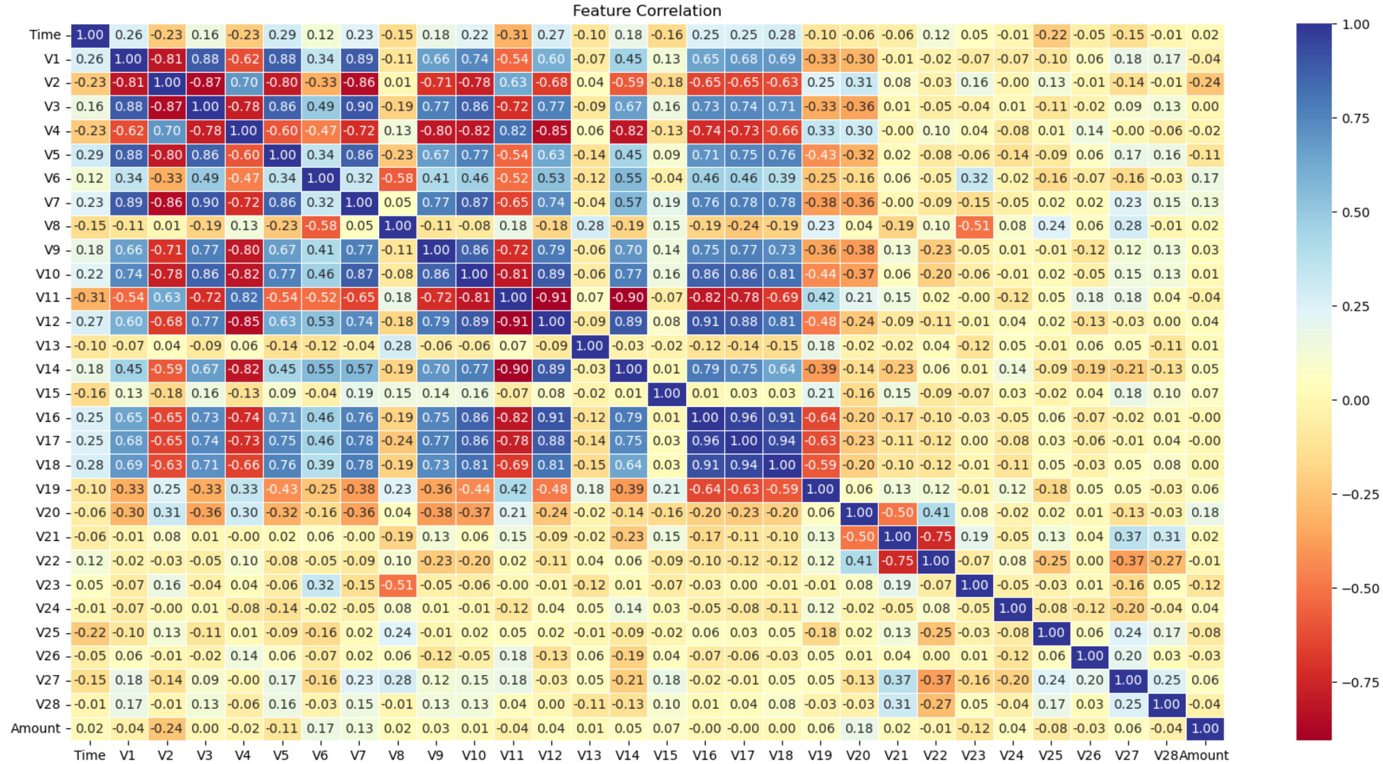
Out[3]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9
568625	1.084762	-0.063846	1.693838	-4.245310	2.295877	-1.212742	-2.054768	-1.860351	0.811518	-1.730054
568626	-0.637311	-1.711814	0.456246	-1.120049	0.609640	-0.859496	-0.304985	-1.335793	0.017532	-0.016575
568627	-0.480862	-0.528259	1.141664	-0.850414	2.055184	0.068474	-0.460563	-1.022289	0.539480	-1.365581
568628	1.201490	-0.711715	2.790795	-5.314459	4.956698	-0.224579	-1.956847	-2.172370	1.186167	-3.607043
568629	0.485418	-2.360354	1.863733	-4.112487	2.079790	-1.182344	-1.466585	-3.022281	-0.239394	-0.269545

5 rows × 31 columns

In [4]:
```
corrmat = df.drop(['label' ],axis=1).corr()

# Visualize feature correlation
fig, ax = plt.subplots(figsize=(20,10))
sns.heatmap(corrmat, annot=True, annot_kws={"size": 10}, fmt="0.2f", linewidths=0.5, squ
ax.set_title('Feature Correlation', fontsize=12, color='black');
```



In [5]:
```python
X = df.drop(['label'],axis=1)
y = df['label']
```

In [6]:
```python
df_shuffled = df.sample(frac=1).reset_index(drop=True)
df_shuffled
```

Out[6]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	
0	-1.332606	-0.376344	0.610026	1.012892	-0.088310	-0.177241	-0.582397	0.372280	0.138438	-0.22
1	-0.920881	0.579338	0.133159	0.156001	0.783980	-0.260421	-0.384843	-0.115501	0.012350	0.50
2	-0.732534	-0.306165	1.540090	-1.769122	2.559741	-0.329996	-0.896657	-1.614333	0.639431	-2.33
3	-0.834059	0.505940	-0.221271	-0.273324	0.167876	-0.292997	-1.028146	0.512768	-0.350057	-0.09
4	-1.034425	0.650800	-0.234663	0.431018	-0.185936	-0.373417	0.456422	-0.659016	0.220902	0.81
...	...	...	...	...	...	...	...	...	...	
568625	0.726101	-0.052554	0.510153	-0.843696	0.905225	0.340690	-0.730501	0.816403	-0.004369	-0.42
568626	0.935391	-0.209749	-0.458019	-1.296263	-0.134144	1.370243	-1.069050	-0.057465	-0.827364	0.68
568627	-0.586544	0.662129	0.184216	-0.101091	0.210951	0.212353	-0.052634	-0.011032	-0.004482	-0.19
568628	-0.961523	0.629701	-0.053195	-0.060995	-0.075641	-0.149894	-0.463464	0.107150	-0.123758	0.02
568629	0.068355	-9.435073	8.128311	-13.554183	8.177360	-10.339398	0.005260	-20.259736	-5.865464	-7.86

568630 rows × 31 columns

In [7]:
```python
X = df_shuffled.iloc[:,:-1]; y = df_shuffled.iloc[:,-1]
```

In [8]:
```python
#train, test split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.35, shuffle=False)
```

In [9]:
```python
y_test.value_counts()
```

Out[9]:
```
1    99539
```

```
0    99482
Name: label, dtype: int64
```

In [10]:
```python
lgbm = LGBMClassifier()
```

In [11]:
```python
lgbm.get_params()
```

Out[11]:
```
{'boosting_type': 'gbdt',
 'class_weight': None,
 'colsample_bytree': 1.0,
 'importance_type': 'split',
 'learning_rate': 0.1,
 'max_depth': -1,
 'min_child_samples': 20,
 'min_child_weight': 0.001,
 'min_split_gain': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'num_leaves': 31,
 'objective': None,
 'random_state': None,
 'reg_alpha': 0.0,
 'reg_lambda': 0.0,
 'subsample': 1.0,
 'subsample_for_bin': 200000,
 'subsample_freq': 0}
```

In [12]:
```python
start_time = time.time()
lgbm.fit(X_train, y_train)
end_time = time.time()
print('the training time is:',end_time-start_time)
```

```
  File "C:\Users\sjw2000824\anaconda3\lib\site-packages\joblib\externals\loky\backend\co
ntext.py", line 227, in _count_physical_cores
    cpu_info = subprocess.run(
  File "C:\Users\sjw2000824\anaconda3\lib\subprocess.py", line 505, in run
    with Popen(*popenargs, **kwargs) as process:
  File "C:\Users\sjw2000824\anaconda3\lib\subprocess.py", line 951, in __init__
    self._execute_child(args, executable, preexec_fn, close_fds,
  File "C:\Users\sjw2000824\anaconda3\lib\subprocess.py", line 1420, in _execute_child
    hp, ht, pid, tid = _winapi.CreateProcess(executable, args,
[LightGBM] [Info] Number of positive: 184776, number of negative: 184833
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.
032746 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7650
[LightGBM] [Info] Number of data points in the train set: 369609, number of used feature
s: 30
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.499923 -> initscore=-0.000308
[LightGBM] [Info] Start training from score -0.000308
the training time is: 1.603391408920288
```

In [13]:
```python
y_pred = lgbm.predict(X_test)
```

In [14]:
```python
for i in range(X_test.shape[0]):
    if y_pred[i]>=0.5:
        y_pred[i] = 1
    else:
        y_pred[i] = 0
```

In [15]:
```python
pd.DataFrame(y_pred).value_counts()
```

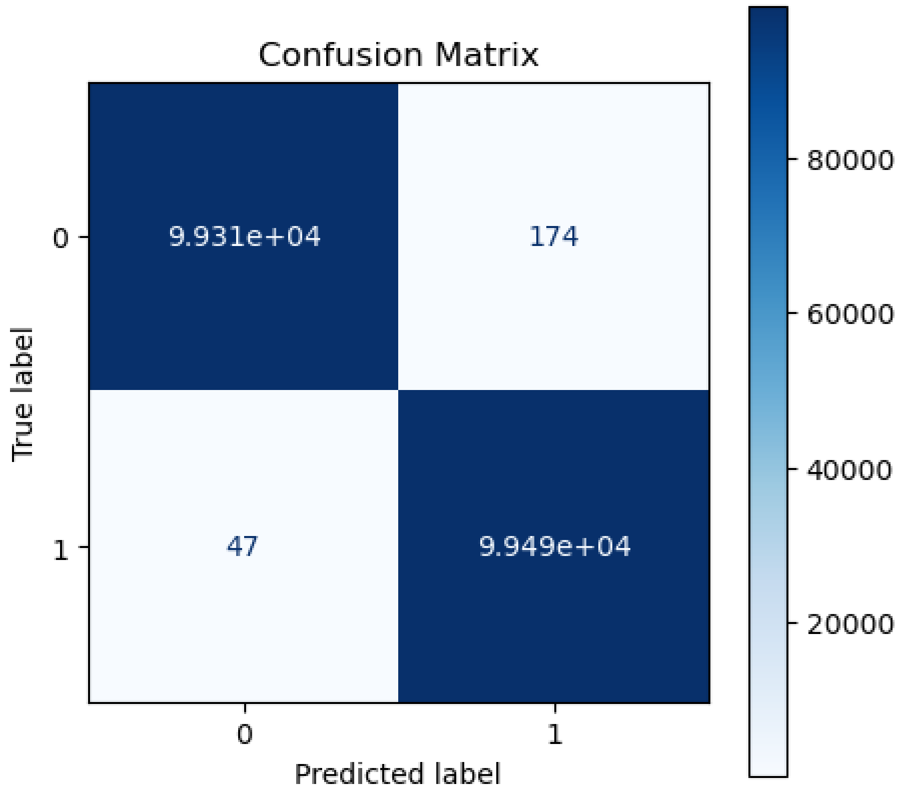
Out[15]:
```
1    99666
0    99355
dtype: int64
```

In [16]:
```
fig,ax = plt.subplots(figsize=(5,5))
plot_confusion_matrix(lgbm,X_test,y_test, ax=ax, cmap='Blues', values_format='.4g')
plt.title('Confusion Matrix')
plt.grid(False)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     99482
           1       1.00      1.00      1.00     99539

    accuracy                           1.00    199021
   macro avg       1.00      1.00      1.00    199021
weighted avg       1.00      1.00      1.00    199021
```

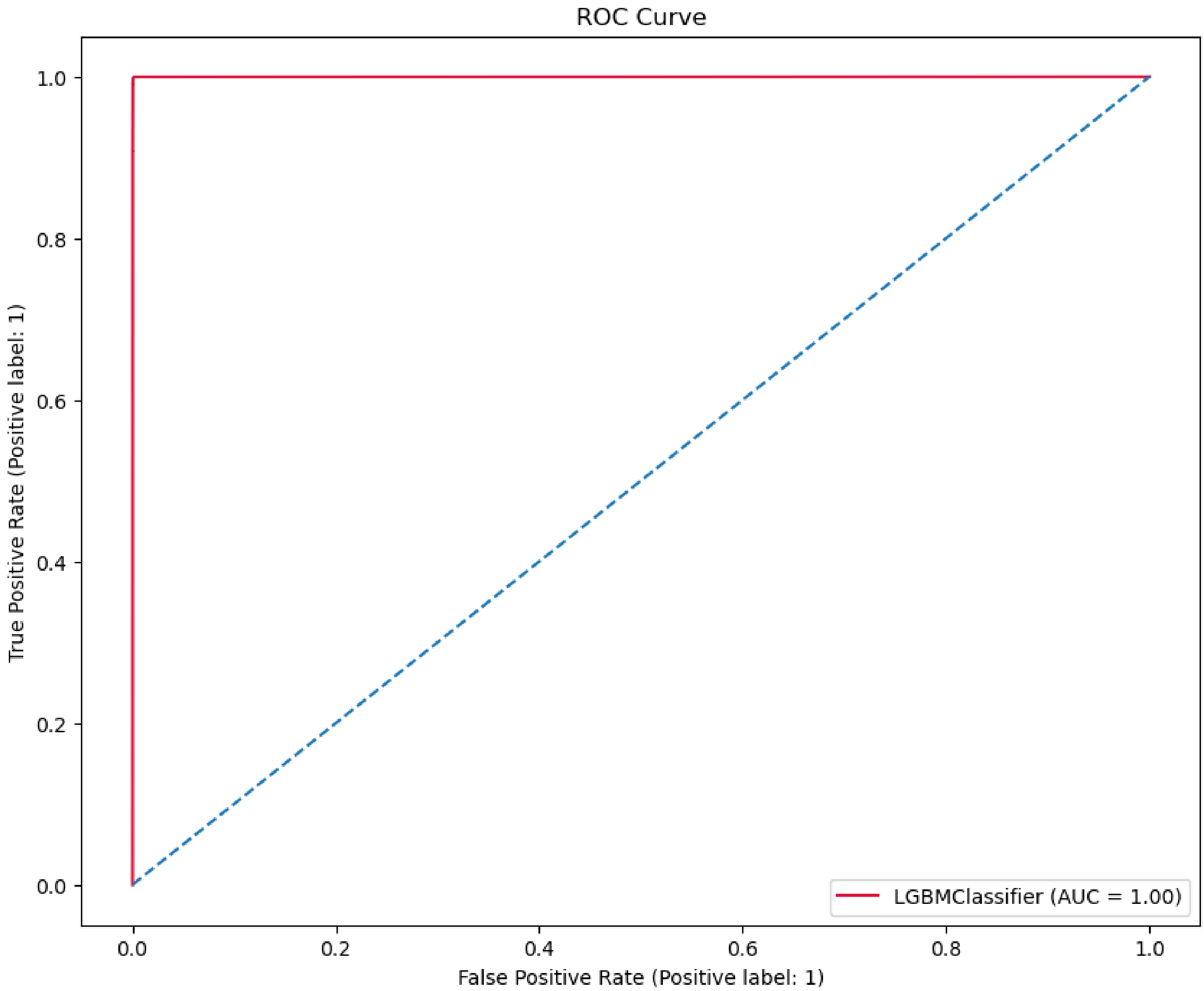


In [17]:
```
print(f'Train accuracy:{lgbm.score(X_train,y_train):0.4}')
print(f'Test accuracy:{lgbm.score(X_test,y_test):0.4}')
```

```
Train accuracy:0.9993
Test accuracy:0.9989
```

In [18]:
```
fig,ax = plt.subplots(figsize=(10,8))
plot_roc_curve(lgbm, X_test,y_test,ax=ax, color='crimson')
ax.plot([0,1],[0,1],linestyle='--')
ax.set_title('ROC Curve')
```

Out[18]:
```
Text(0.5, 1.0, 'ROC Curve')
```



In [19]:
```
lgbm.get_params()
```

Out[19]:
```
{'boosting_type': 'gbdt',
 'class_weight': None,
 'colsample_bytree': 1.0,
 'importance_type': 'split',
 'learning_rate': 0.1,
 'max_depth': -1,
 'min_child_samples': 20,
 'min_child_weight': 0.001,
 'min_split_gain': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'num_leaves': 31,
 'objective': None,
 'random_state': None,
 'reg_alpha': 0.0,
 'reg_lambda': 0.0,
 'subsample': 1.0,
 'subsample_for_bin': 200000,
 'subsample_freq': 0}
```

In [20]:
```
param_space = {
    'num_leaves': [31,32,33,34,35],
    'learning_rate': [0.05,0.07,0.1,0.15,0.2],
    'reg_alpha' : [0,0.001,0.005,0.01,0.05],
    'reg_lambda': [0.5,1,1.5,2,3]
    }
```

In [21]:
```
start_time = time.time()
tscv = TimeSeriesSplit(n_splits=5, gap=1)
rs = RandomizedSearchCV(lgbm, param_space, n_iter=100, scoring='f1', cv = tscv, verbose=
rs.fit(X_train, y_train)
end_time = time.time()
print('the tuning time is:',end_time-start_time)
```

```
[LightGBM] [Info] Number of positive: 30723, number of negative: 30880
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.
003763 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7650
[LightGBM] [Info] Number of data points in the train set: 61603, number of used feature
s: 30
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.498726 -> initscore=-0.005097
[LightGBM] [Info] Start training from score -0.005097
[LightGBM] [Info] Number of positive: 61440, number of negative: 61764
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.
010449 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7650
[LightGBM] [Info] Number of data points in the train set: 123204, number of used feature
s: 30
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.498685 -> initscore=-0.005260
[LightGBM] [Info] Start training from score -0.005260
[LightGBM] [Info] Number of positive: 92316, number of negative: 92489
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.
010338 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7650
[LightGBM] [Info] Number of data points in the train set: 184805, number of used feature
s: 30
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.499532 -> initscore=-0.001872
[LightGBM] [Info] Start training from score -0.001872
[LightGBM] [Info] Number of positive: 123086, number of negative: 123320
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.
014267 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7650
[LightGBM] [Info] Number of data points in the train set: 246406, number of used feature
s: 30
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.499525 -> initscore=-0.001899
[LightGBM] [Info] Start training from score -0.001899
[LightGBM] [Info] Number of positive: 153828, number of negative: 154179
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.
017784 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7650
[LightGBM] [Info] Number of data points in the train set: 308007, number of used feature
s: 30
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.499430 -> initscore=-0.002279
[LightGBM] [Info] Start training from score -0.002279
[LightGBM] [Info] Number of positive: 30723, number of negative: 30880
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.
004924 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7650
[LightGBM] [Info] Number of data points in the train set: 61603, number of used feature
s: 30
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.498726 -> initscore=-0.005097
[LightGBM] [Info] Start training from score -0.005097
[LightGBM] [Info] Number of positive: 61440, number of negative: 61764
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.
007225 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7650
```

```
[LightGBM] [Info] Start training from score -0.001872
[LightGBM] [Info] Number of positive: 123086, number of negative: 123320
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.
019646 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7650
[LightGBM] [Info] Number of data points in the train set: 246406, number of used feature
s: 30
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.499525 -> initscore=-0.001899
[LightGBM] [Info] Start training from score -0.001899
[LightGBM] [Info] Number of positive: 153828, number of negative: 154179
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.
019942 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7650
[LightGBM] [Info] Number of data points in the train set: 308007, number of used feature
s: 30
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.499430 -> initscore=-0.002279
[LightGBM] [Info] Start training from score -0.002279
[LightGBM] [Info] Number of positive: 184776, number of negative: 184833
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.
024590 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7650
[LightGBM] [Info] Number of data points in the train set: 369609, number of used feature
s: 30
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.499923 -> initscore=-0.000308
[LightGBM] [Info] Start training from score -0.000308
the tuning time is: 498.78461241722107
```

In [22]:

```
rs.best_params_
```

Out[22]:

```
{'reg_lambda': 1, 'reg_alpha': 0, 'num_leaves': 34, 'learning_rate': 0.2}
```

In [23]:

```
cls = LGBMClassifier(**rs.best_params_)
cls.fit(X_train,y_train)
```

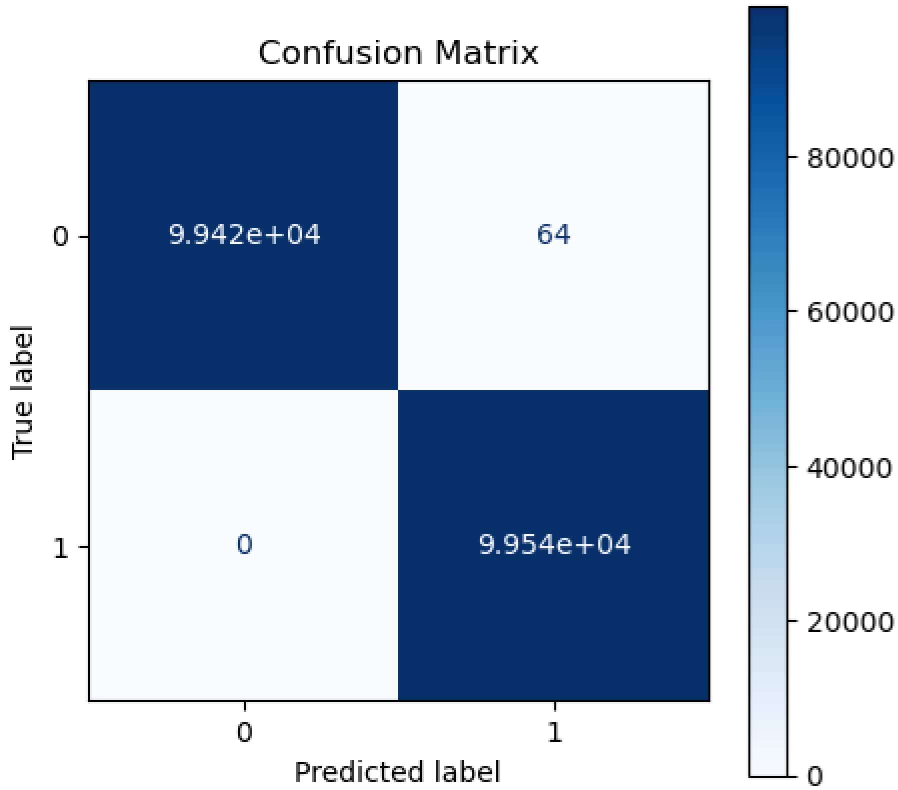
```
[LightGBM] [Info] Number of positive: 184776, number of negative: 184833
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.
036650 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7650
[LightGBM] [Info] Number of data points in the train set: 369609, number of used feature
s: 30
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.499923 -> initscore=-0.000308
[LightGBM] [Info] Start training from score -0.000308
```

Out[23]:

```
LGBMClassifier(learning_rate=0.2, num_leaves=34, reg_alpha=0, reg_lambda=1)
```

In [29]:

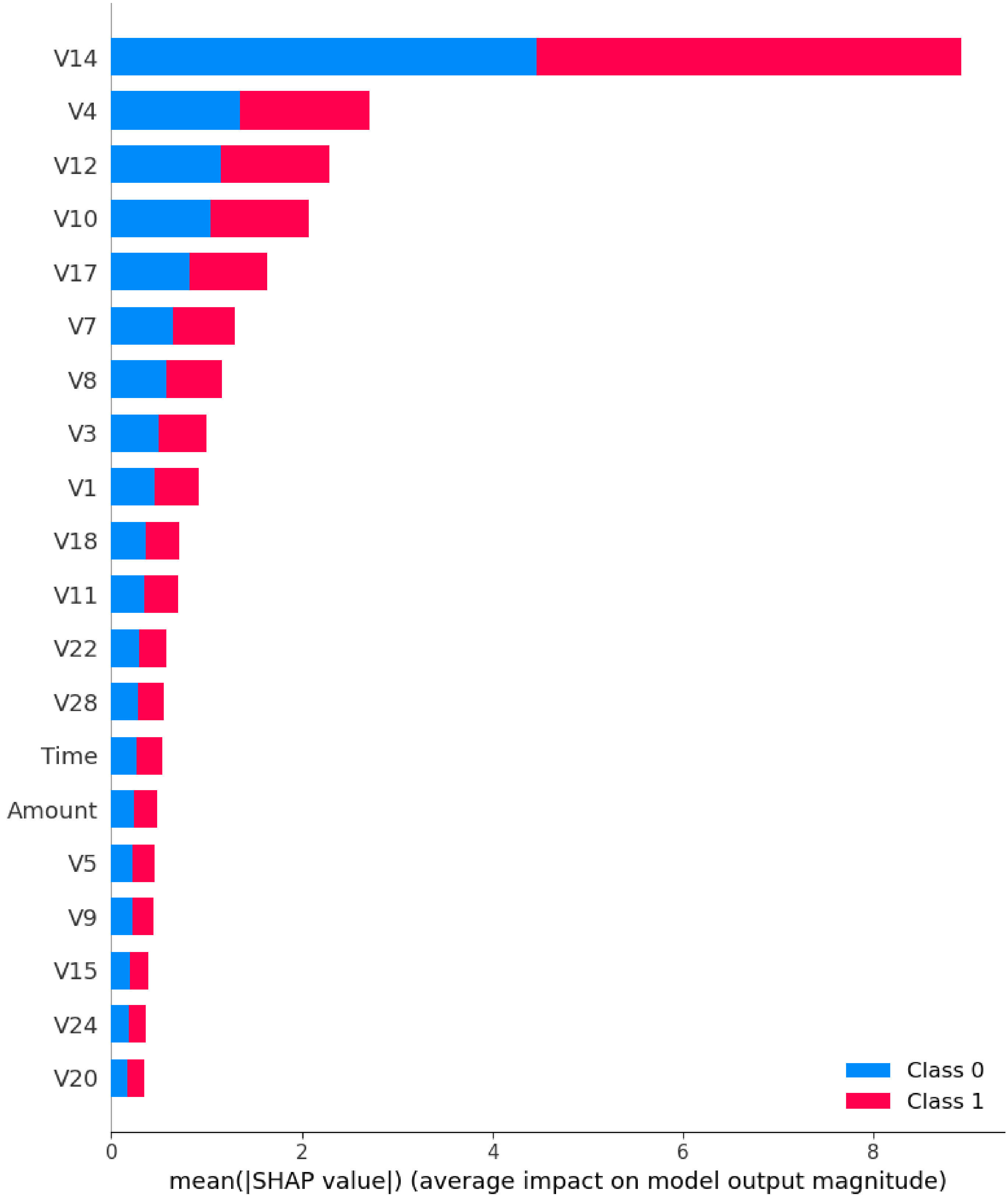
```
fig,ax = plt.subplots(figsize=(5,5))
plot_confusion_matrix(cls,X_test,y_test, ax=ax, cmap='Blues', values_format='.4g')
plt.title('Confusion Matrix')
plt.grid(False)
```



In [25]:
```
print(f'Train accuracy:{cls.score(X_train,y_train):0.4}')
print(f'Test accuracy:{cls.score(X_test,y_test):0.4}')
```

```
Train accuracy:1.0
Test accuracy:0.9997
```

In [30]:
```
import shap
explainer = shap.TreeExplainer(cls)
shap_values = explainer.shap_values(X_test)
shap.summary_plot(shap_values, X_test)
```



summary

In this project, we focus on Gradient Boosting Decision Tree (GBDT) methodology, built XGBoost and LightGBM model,and compared their perfomance on Credit card Fraud dataset.

In practice, It's validated that lightGBM is 15 times faster than XGBoost, and achieved better accuracy based upon large-scale dataset.

Contribution

YUAN, Shuoqi: Part 1 Credit card fraud background

ZHENG, Zezhou: Part 2 Algorithm methodology

Yang, Ziyi: Part 3 XGBoost establish and optimization

SHEN, Jiawei: Part 4 LightGBM establish, optimization and comparision