Python + 머신러닝 예측

- jupyter/pyspark:python3.8.8에서 작성
- 데이터 출처 (캐글 링크 | https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud)

EDA

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, RobustScaler
from sklearn.manifold import TSNE
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import recall_score, precision_score, f1_score, accuracy_score
```

In [2]:

```
df = pd.read_csv('creditcard.csv')
```

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

In [4]:

```
df.describe()
```

Out[4]:

	Time	V1	V2	V3	V4	V5	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16	
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7

8 rows × 31 columns

In [5]:

```
df.head()
```

Out[5]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.3637
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.2554
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.5146
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.3870
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.8177

5 rows × 31 columns

In [6]:

```
# label 값 확인
df['Class'].value_counts()
```

Out[6]:

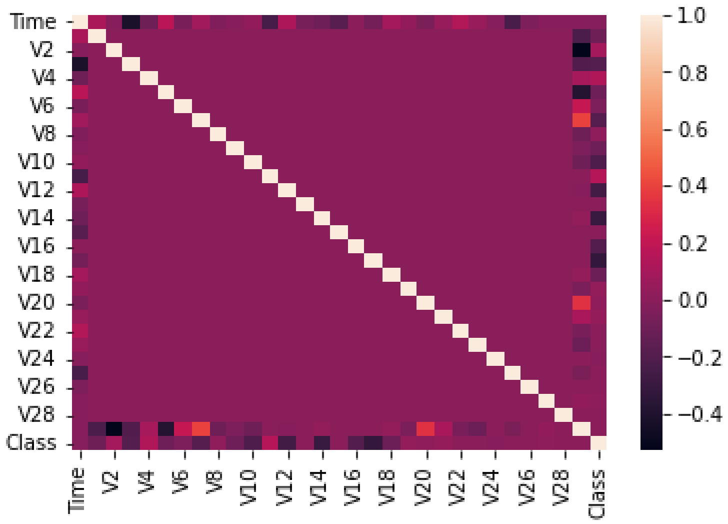
```
0    284315
1       492
Name: Class, dtype: int64
```

In [7]:

```
# 공산성이 낮은 데이터
sns.heatmap(df.corr())
```

Out[7]:

```
<AxesSubplot:>
```



In [8]:

```
# label값에 대한 상관관계 확인
df.corr()['Class'].sort_values()
```

Out[8]:

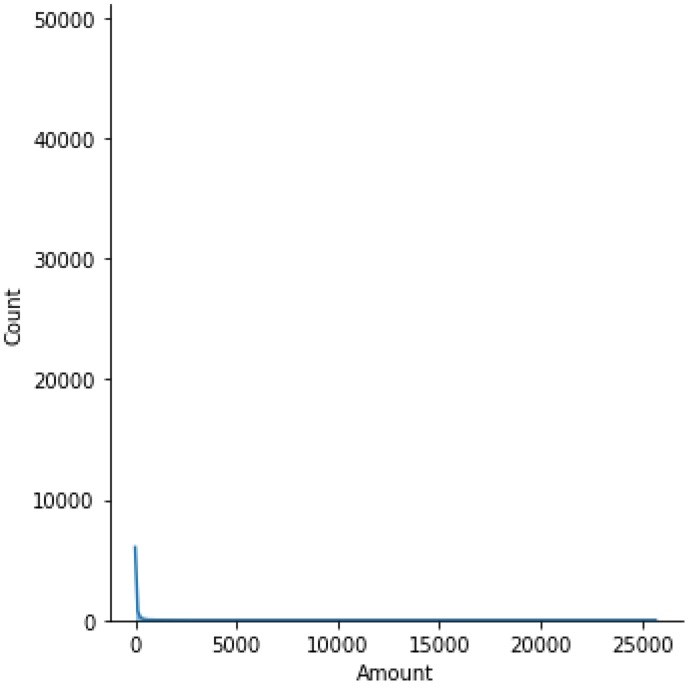
```
V17      -0.326481
V14      -0.302544
V12      -0.260593
V10      -0.216883
V16      -0.196539
V3       -0.192961
V7       -0.187257
V18      -0.111485
V1       -0.101347
V9       -0.097733
V5       -0.094974
V6       -0.043643
Time     -0.012323
V24      -0.007221
V13      -0.004570
V15      -0.004223
V23      -0.002685
V22       0.000805
V25       0.003308
V26       0.004455
Amount    0.005632
V28       0.009536
V27       0.017580
V8        0.019875
V20       0.020090
V19       0.034783
V21       0.040413
V2        0.091289
V4        0.133447
V11       0.154876
Class     1.000000
Name: Class, dtype: float64
```

In [9]:

```
# Amount 정규화 필요
sns.displot(data=df, x="Amount", kde=True)
```

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x7f282cdb6210>
```

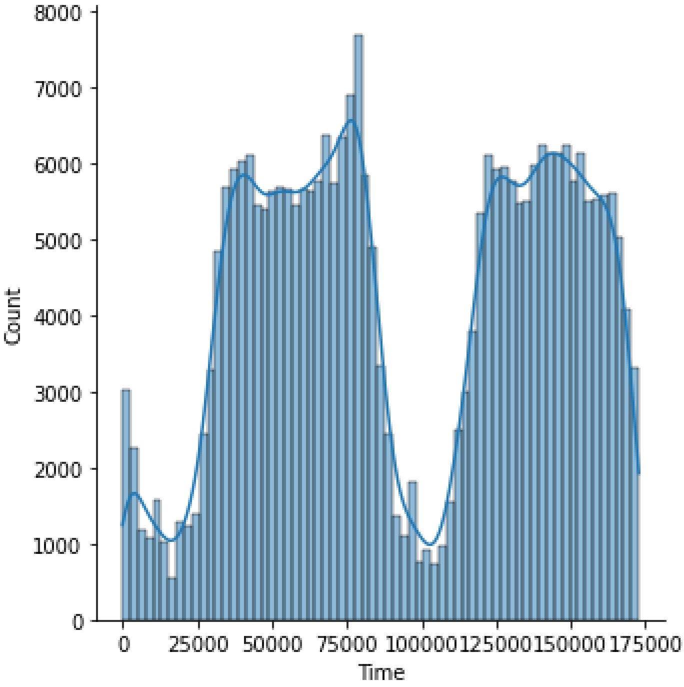


In [10]:

```
# Time 정규화 필요
sns.displot(data=df, x="Time", kde=True)
```

Out[10]:

```
<seaborn.axisgrid.FacetGrid at 0x7f2830d3f0d0>
```

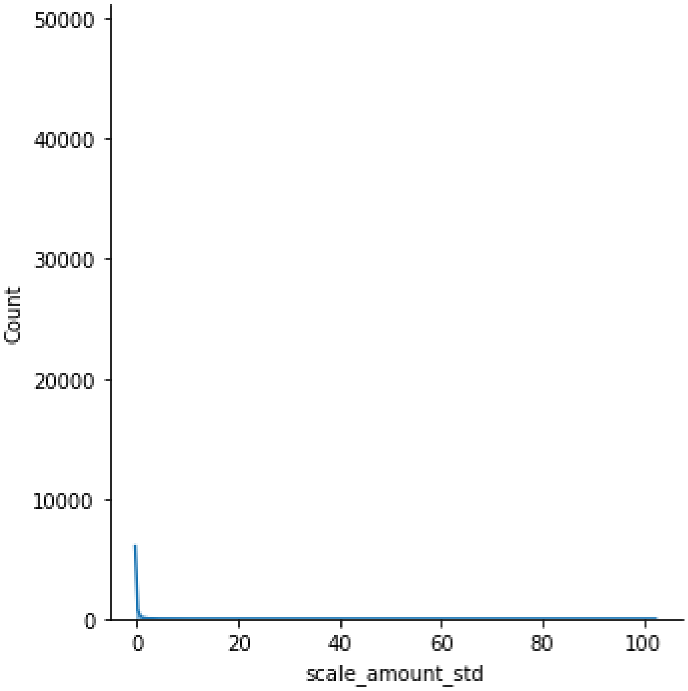


In [11]:

```
# standard scaler
std_scaler = StandardScaler()
df['scale_amount_std'] = std_scaler.fit_transform(df['Amount'].values.reshape(-1, 1))
sns.displot(data=df, x="scale_amount_std", kde=True)
```

Out[11]:

```
<seaborn.axisgrid.FacetGrid at 0x7f282cd6f450>
```



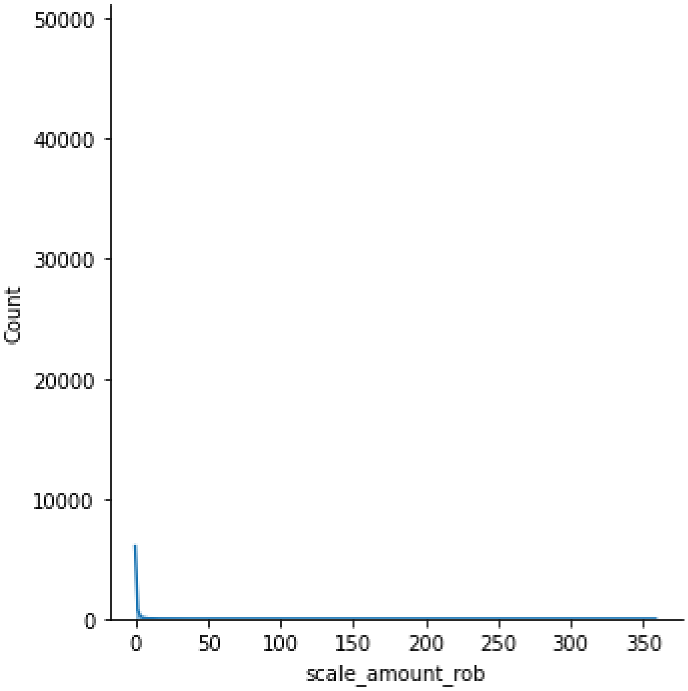
In [12]:

```
# robust scaler
rob_scaler = RobustScaler()
```

```
df['scale_amount_rob'] = rob_scaler.fit_transform(df['Amount'].values.reshape(-1, 1))
sns.displot(data=df, x="scale_amount_rob", kde=True)
```

Out[12]:

```
<seaborn.axisgrid.FacetGrid at 0x7f281d24c550>
```

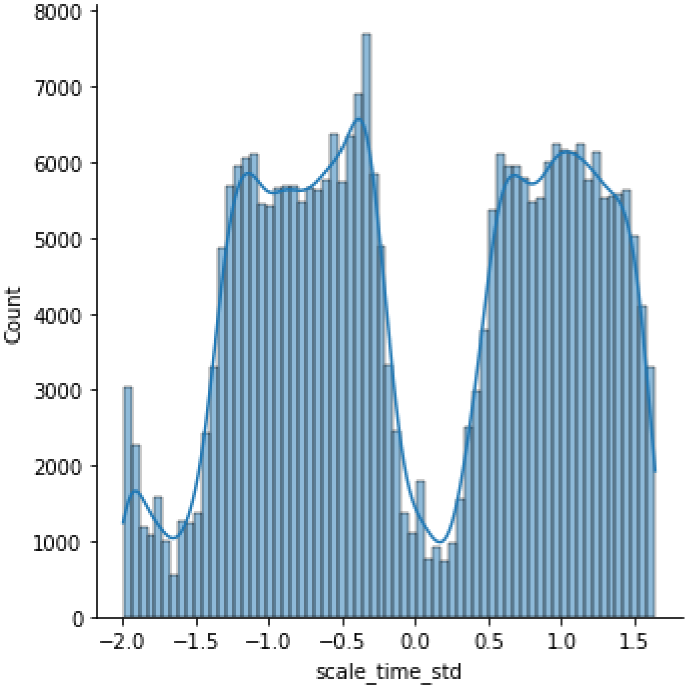


In [13]:

```
# Time 정규화
df['scale_time_std'] = std_scaler.fit_transform(df['Time'].values.reshape(-1, 1))
sns.displot(data=df, x="scale_time_std", kde=True)
```

Out[13]:

```
<seaborn.axisgrid.FacetGrid at 0x7f282cc086d0>
```



In [14]:

```
# 안 쓰는 column 없애기
df = df.drop(['Time', 'Amount', 'scale_amount_std'], axis=1)
scale_amount = df['scale_amount_rob']
scale_time = df['scale_time_std']
```

In [15]:

```
df.insert(0, 'scale_amount', scale_amount)
df.insert(1, 'scale_time', scale_time)
df.head()
```

Out[15]:

	scale_amount	scale_time	V1	V2	V3	V4	V5	V6	V7
0	1.783274	-1.996583	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.23959
1	-0.269825	-1.996583	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.07880
2	4.983721	-1.996562	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.79146
3	1.418291	-1.996562	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.23760
4	0.670579	-1.996541	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.59294

5 rows × 33 columns

In [16]:

```
# dateset shuffle
df = df.sample(frac=1)
df.head()
```

Out[16]:

	scale_amount	scale_time	V1	V2	V3	V4	V5	V6	
198746	3.884580	0.796372	-0.887797	-0.920315	1.505975	0.054601	-1.032067	0.129266	0
78813	7.621603	-0.781162	0.256881	-2.213319	-0.404803	0.029907	-1.370190	-0.569455	0
236461	-0.194508	1.137299	0.086933	0.995813	-0.329656	-0.626455	0.912197	-0.574713	0
47431	1.229651	-1.086670	-0.931375	0.322760	-0.566347	-0.480548	1.687959	3.933089	-0
283565	-0.148536	1.618978	-1.315703	1.308200	1.583054	0.843591	-0.016332	0.665115	0

5 rows × 33 columns

In [17]:

```
fraud_df = df[df['Class'] == 1]
non_fraud_df = df[df['Class'] == 0][:len(fraud_df)]
```

In [18]:

```
# undersampling
under_sampled = pd.concat([fraud_df, non_fraud_df])
new_df = under_sampled.sample(frac=1)
new_df.head()
```

Out[18]:

	scale_amount	scale_time	V1	V2	V3	V4	V5	V6	
40336	-0.146720	-1.150118	1.213232	0.019081	0.653910	0.890635	-0.624259	-0.566418	-0
233035	-0.056033	1.107692	1.988135	0.116951	-1.605889	0.351343	0.434941	-0.639378	0
52466	-0.293440	-1.039227	-1.476893	2.122314	-1.229470	1.201849	-0.343264	-1.317704	-1
239214	-0.044435	1.162274	0.143476	0.583583	-0.083261	-0.469798	0.445817	-1.184308	0
95534	0.138476	-0.620279	1.193916	-0.571085	0.742522	-0.014588	-0.624561	0.832162	-0

5 rows × 33 columns

In [19]:

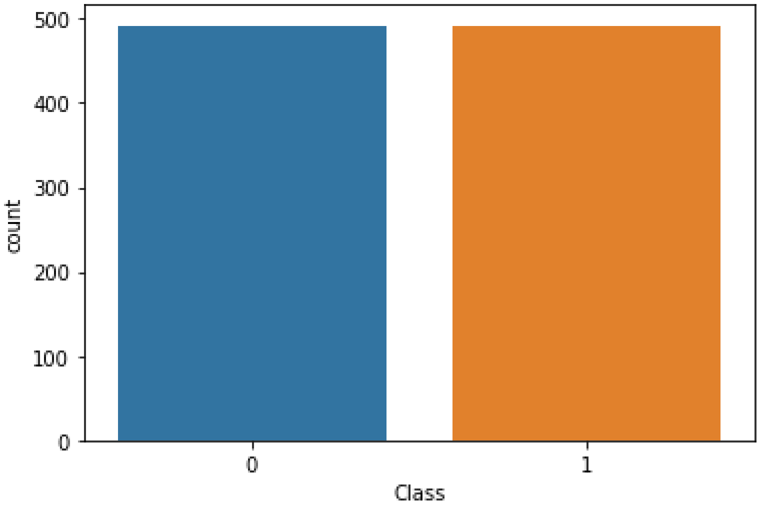
```
print("오리지널 라벨 분포:")
print(df['Class'].value_counts() / len(df))

print("언더 샘플 후 라벨 분포:")
print(new_df['Class'].value_counts() / len(new_df))

sns.countplot('Class', data=new_df)
```

```
오리지널 라벨 분포:
0    0.998273
1    0.001727
Name: Class, dtype: float64
언더 샘플 후 라벨 분포:
0    0.5
1    0.5
Name: Class, dtype: float64
```

```
/opt/conda/envs/py37/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, the only va
lid positional argument will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[19]:

```
<AxesSubplot:xlabel='Class', ylabel='count'>
```

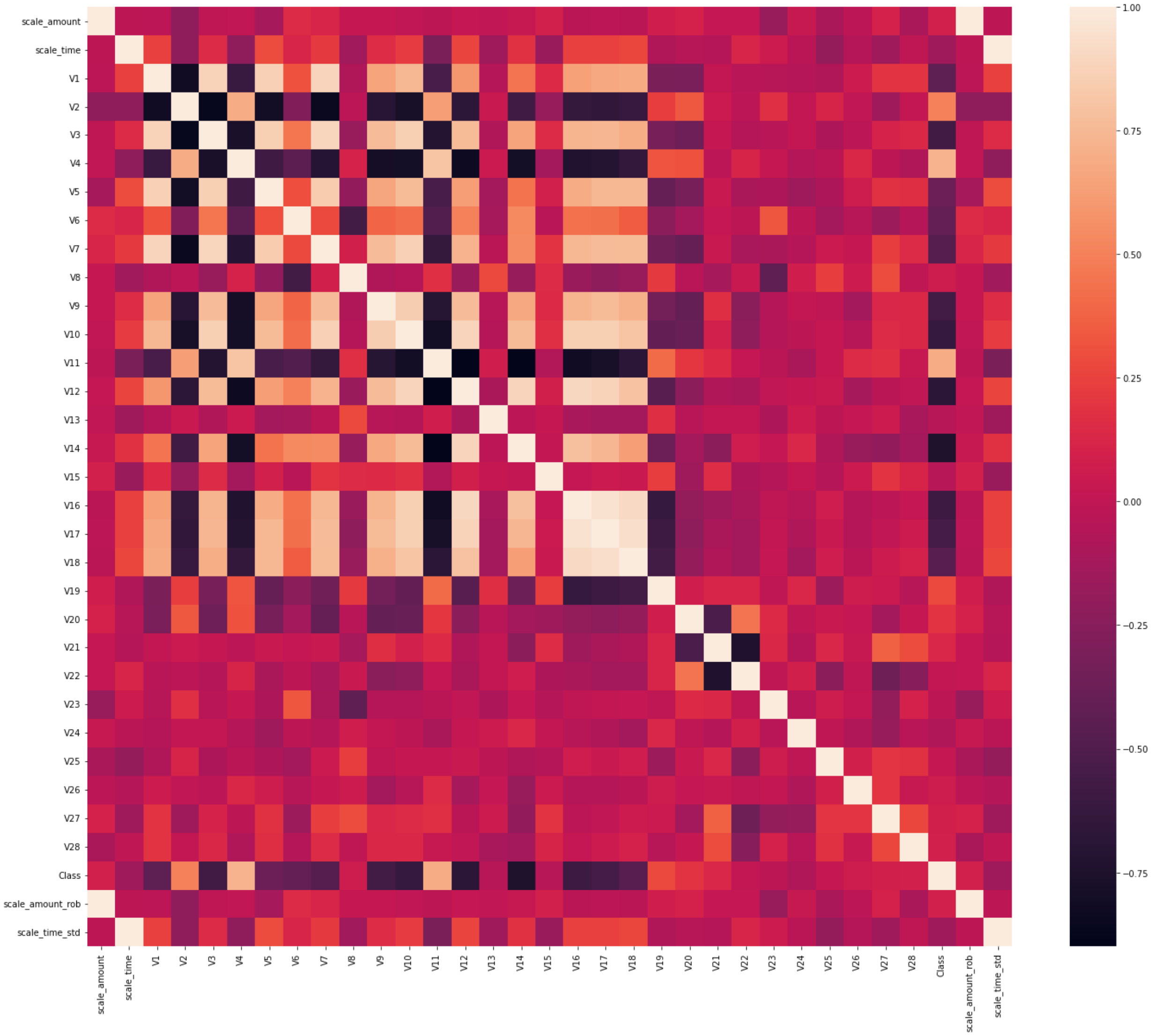


In [20]:

```
# 상관관계 확인, 이전과 다르게 상관성을 띠는 피쳐들이 생기기 시작.
under_sample_corr = new_df.corr()
f, ax = plt.subplots(1, figsize=(24,20))
sns.heatmap(under_sample_corr, ax=ax)
```

Out[20]:

```
<AxesSubplot:>
```

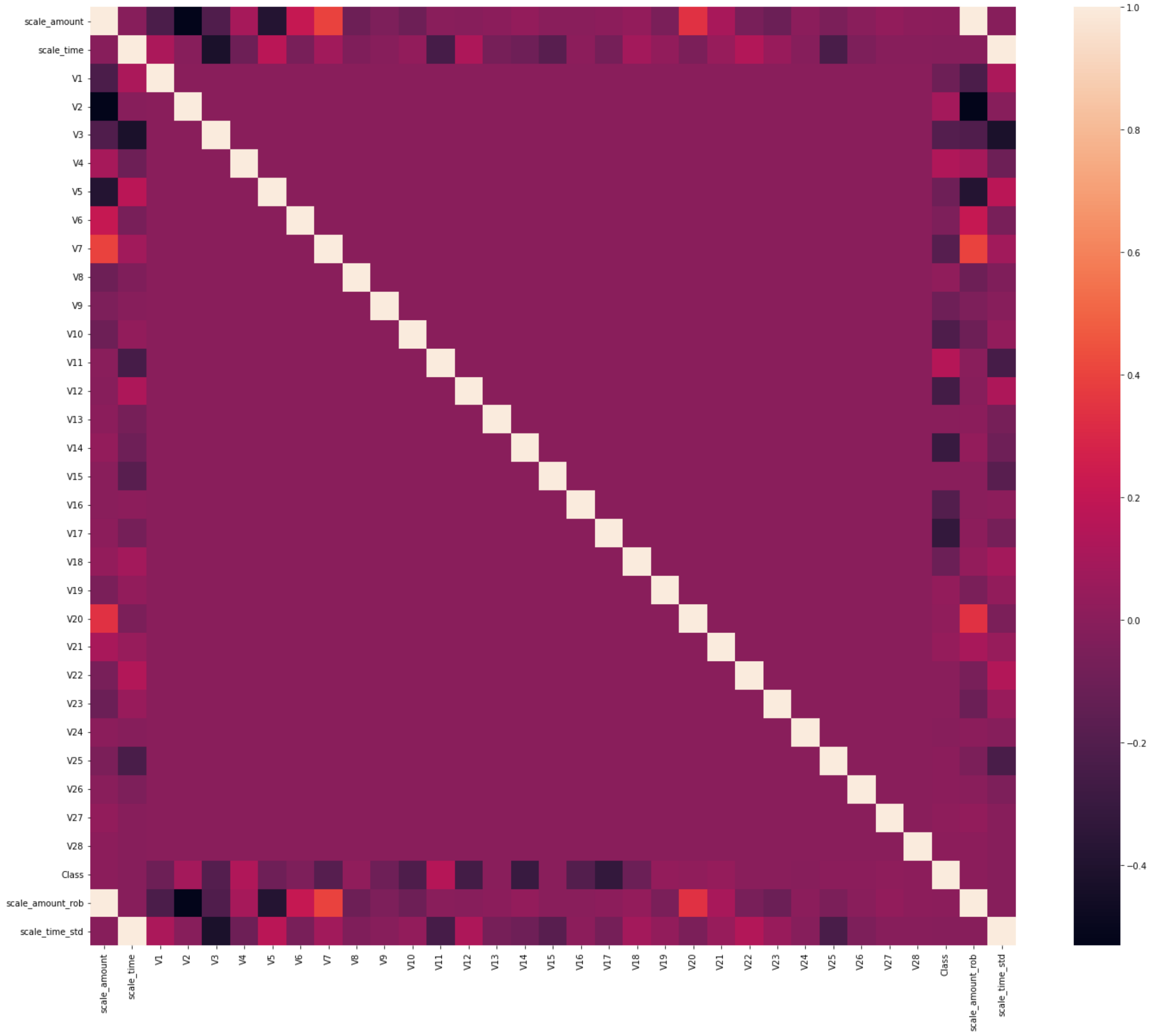


In [21]:

```python
# 비교
before_corr = df.corr()
f, ax = plt.subplots(1, figsize=(24,20))
sns.heatmap(before_corr, ax=ax)
```

Out[21]:

```
<AxesSubplot:>
```

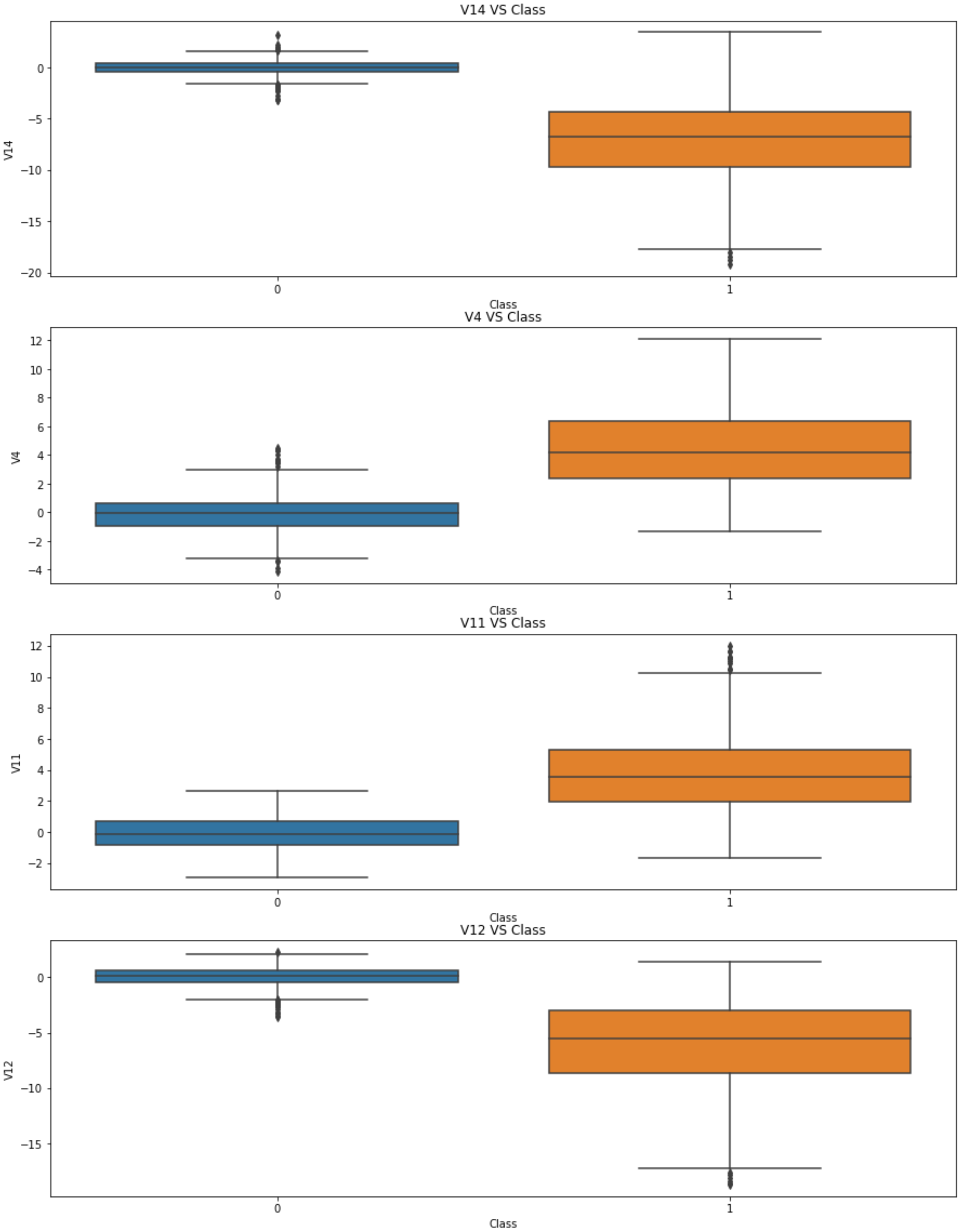


In [22]:

```
# class와 가장 상관 관계가 높은 4개 feature 확인
top_4_corr = abs(under_sample_corr['Class']).sort_values(ascending=False)[1:5]
```

In [23]:

```
f, ax = plt.subplots(4, figsize=(15, 20))

for i, col in enumerate(top_4_corr.index):
    sns.boxplot(x="Class", y=col, data=new_df, ax=ax[i])
    ax[i].set_title(f"{col} VS Class")

plt.show()
```



In [24]:

```
# outlier들이 존재. outlier들을 제거
v14_fraud = new_df['V14'].loc[new_df['Class'] == 1].values
q25 = np.percentile(v14_fraud, 25)
q75 = np.percentile(v14_fraud, 75)
v14_iqr = q75 - q25
v14_cutoff = v14_iqr * 1.5
v14_lower, v14_upper = q25 - v14_cutoff, q75 + v14_cutoff
print('q25: {}, q75: {}'.format(q25, q75))
print('v14_iqr: {}'.format(v14_iqr))
print('v14_cutoff: {}'.format(v14_cutoff))
print('v14_lower: {} v14_upper: {}'.format(v14_lower, v14_upper))
```

```
q25: -9.692722964972386, q75: -4.282820849486865
v14_iqr: 5.409902115485521
v14_cutoff: 8.114853173228282
v14_lower: -17.807576138200666 v14_upper: 3.8320323237414167
```

In [25]:

```
# outliers
outliers = [i for i in v14_fraud if i < v14_lower or i > v14_upper]
outliers
```

Out[25]:

```
[-18.8220867423816, -18.0499976898594, -19.2143254902614, -18.4937733551053]
```

In [26]:

```
new_df[(new_df['V14']> v14_upper) | (new_df['V14'] < v14_lower)].index
```

Out[26]:

```
Int64Index([8615, 9252, 8296, 9035], dtype='int64')
```

In [27]:

```
# v14 outlier 제거
new_df = new_df.drop(new_df[(new_df['V14']> v14_upper) | (new_df['V14'] < v14_lower)].
```

In [28]:

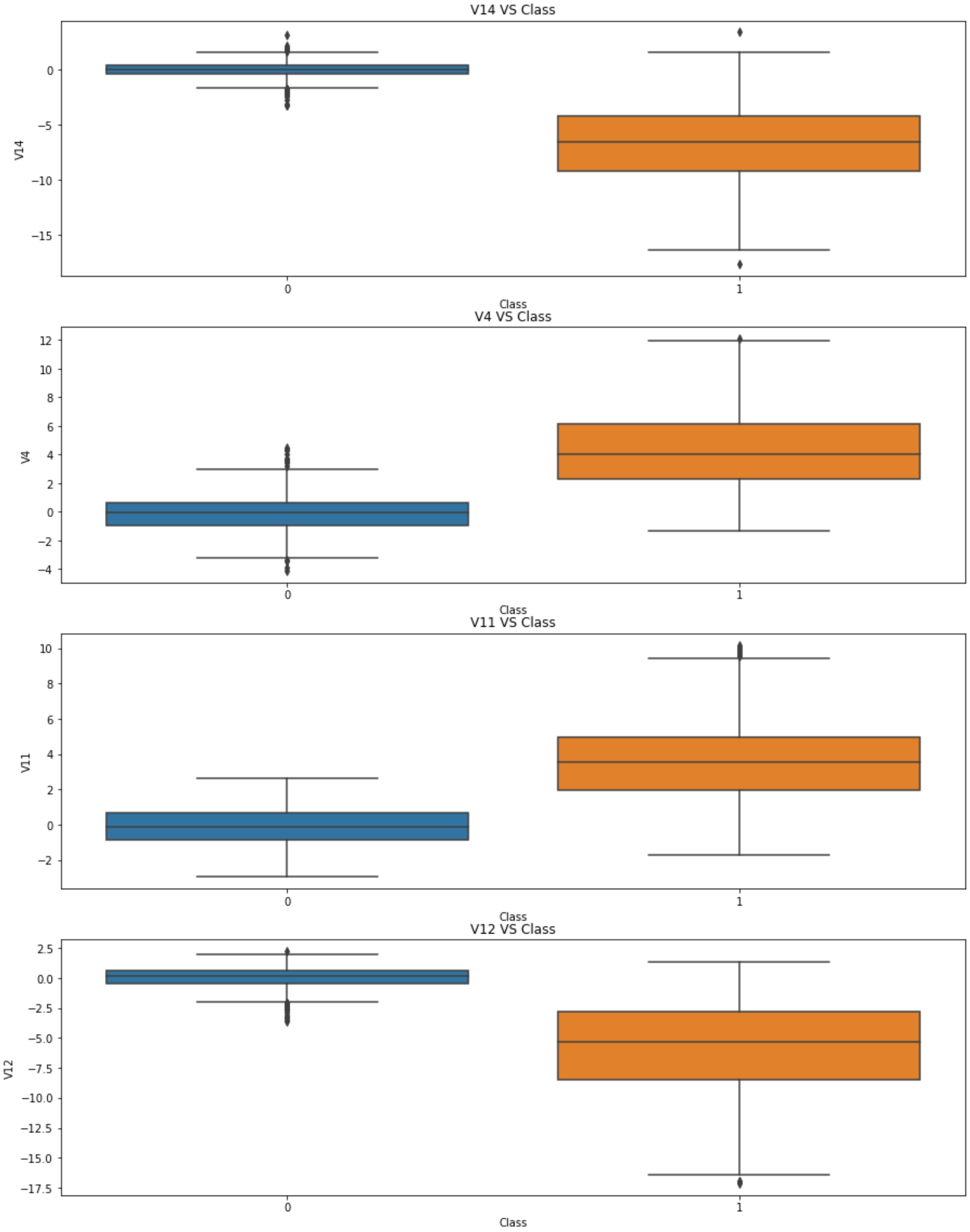
```
# v11 outlier 제거
v11_fraud = new_df['V11'].loc[new_df['Class'] == 1].values
q25 = np.percentile(v11_fraud, 25)
q75 = np.percentile(v11_fraud, 75)

v11_iqr = q75 - q25

v11_cutoff = v11_iqr * 1.5

v11_lower, v11_upper = q25 - v11_cutoff, q75 + v11_cutoff

new_df = new_df.drop(new_df[(new_df['V11']> v11_upper) | (new_df['V11'] < v11_lower)].
```

In [29]:

```
# v12 outlier 제거
v12_fraud = new_df['V12'].loc[new_df['Class'] == 1].values
q25 = np.percentile(v12_fraud, 25)
q75 = np.percentile(v12_fraud, 75)

v12_iqr = q75 - q25

v12_cutoff = v12_iqr * 1.5

v12_lower, v12_upper = q25 - v12_cutoff, q75 + v12_cutoff

new_df = new_df.drop(new_df[(new_df['V12']> v12_upper) | (new_df['V12'] < v12_lower)].
```

In [30]:

```
# outlier 제거 후 시각화
f, ax = plt.subplots(4, figsize=(15, 20))

for i, col in enumerate(top_4_corr.index):
    sns.boxplot(x="Class", y=col, data=new_df, ax=ax[i])
    ax[i].set_title(f"{col} VS Class")

plt.show()
```



In [31]:
```
X = new_df.drop('Class', axis=1)
y = new_df['Class']
```

In [32]:
```
x_reduced_tsne = TSNE(n_components=2).fit_transform(X.values)
```

```
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/manifold/_t_sne.py:783: Futu
reWarning: The default initialization in TSNE will change from 'random' to 'pca' in
1.2.
  FutureWarning,
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/manifold/_t_sne.py:793: Futu
reWarning: The default learning rate in TSNE will change from 200.0 to 'auto' in 1.2.
  FutureWarning,
```

In [33]:

```
tsne_df = new_df.copy()
tsne_df['tsne_1'] = x_reduced_tsne[:,0]
tsne_df['tsne_2'] = x_reduced_tsne[:,1]
```

In [34]:

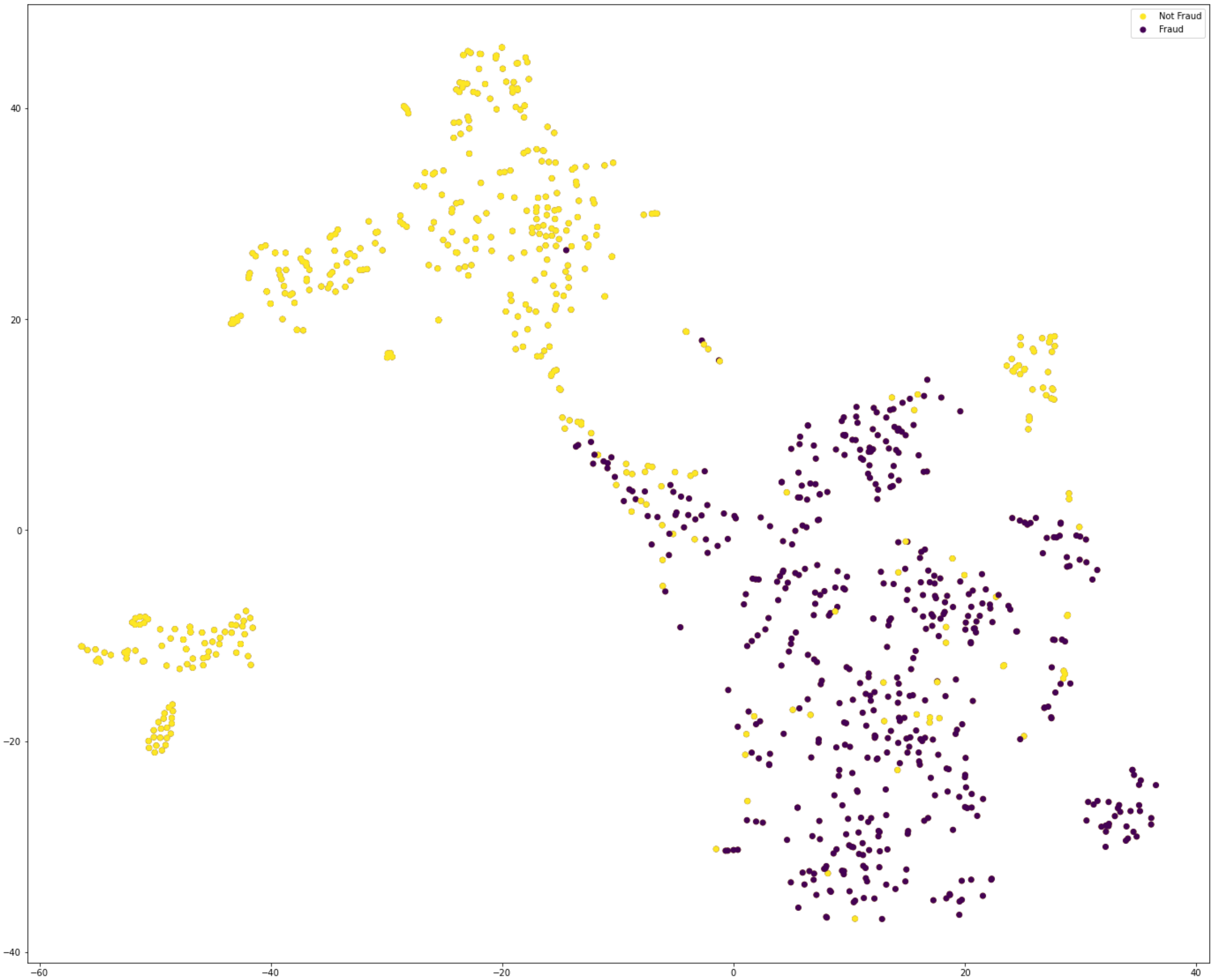
```
# TSNE value 확인
tsne_df.corr()['tsne_2'].sort_values()[:10]
```

Out[34]:

```
V14      -0.545031
tsne_1   -0.436242
V12      -0.363767
V6       -0.275328
V16      -0.212727
V10      -0.209848
V9       -0.189780
V24      -0.163188
V17      -0.147153
V3       -0.124907
Name: tsne_2, dtype: float64
```

In [35]:

```
# TSNE 시각화
f, ax = plt.subplots(1, figsize=(24,20))

ax.scatter(x_reduced_tsne[:,0], x_reduced_tsne[:,1], c=(y==0), label='Not Fraud')
ax.scatter(x_reduced_tsne[:,0], x_reduced_tsne[:,1], c=(y==1), label='Fraud')

ax.legend()

plt.show()
```



Model Baseline

In [43]:

```
X = df.drop('Class', axis=1)
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

In [ ]:

```
classifiers = {
    'LR': LogisticRegression(),
    'KM': KNeighborsClassifier(),
    'SVC': SVC(),
    'Decision Tree': DecisionTreeClassifier(),
    'RandomForestClassifier': RandomForestClassifier()
}

X_train = X_train.values
X_test = X_test.values
y_train = y_train.values
y_test = y_test.values

# 모델 정확도 예측
for key, classifier in classifiers.items():
    classifier.fit(X_train, y_train)
    y_pred = cross_val_predict(classifier, X_train, y_train, cv=5)
    precision, recall, threshold = precision_recall_curve(y_train, y_pred)
    print(key)
    print('---' * 20)
    print('Recall Score: {:.2f}'.format(recall_score(y_train, y_pred)))
    print('Precision Score: {:.2f}'.format(precision_score(y_train, y_pred)))
    print('F1 Score: {:.2f}'.format(f1_score(y_train, y_pred)))
    print('Accuracy Score: {:.2f}'.format(accuracy_score(y_train, y_pred)))
    print('---' * 20)
#     score = cross_val_score(classifier, X_train, y_train, cv=5)
#     print(key, score.mean())
```

```
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:81
8: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:81
8: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:81
8: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:81
8: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:81
8: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:81
8: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
LR
------------------------------------------------------------
Recall Score: 0.62
Precision Score: 0.86
F1 Score: 0.72
Accuracy Score: 1.00
------------------------------------------------------------
```

Model

In [36]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

In [37]:

```
X_train = X_train.values
X_test = X_test.values
y_train = y_train.values
y_test = y_test.values
```

In [38]:

```
classifiers = {
    'LR': LogisticRegression(),
    'KM': KNeighborsClassifier(),
    'SVC': SVC(),
    'Decision Tree': DecisionTreeClassifier(),
    'RandomForestClassifier': RandomForestClassifier()
}
```

In [39]:

```
# 모델 정확도 예측
for key, classifier in classifiers.items():
    classifier.fit(X_train, y_train)
    score = cross_val_score(classifier, X_train, y_train, cv=5)
    print(key, score.mean())
```

```
LR 0.9404692082111439
KM 0.941767909509845
SVC 0.9404692082111437
Decision Tree 0.9223795559279431
RandomForestClassifier 0.9495266024298281
```

In [40]:

```
# 모델 튜닝
# Logistic Regression
lr_param = {"penalty": ['l1', 'l2'], 'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]}

lr = GridSearchCV(LogisticRegression(), lr_param)
lr.fit(X_train, y_train)
lr_bestparam = lr.best_estimator_

# KNeighbors Classifier
kn_param = {"n_neighbors": list(range(2,5,1)), 'algorithm': ['auto', 'ball_tree', 'kd_

knn = GridSearchCV(KNeighborsClassifier(), kn_param)
knn.fit(X_train, y_train)
knn_bestparam = knn.best_estimator_

# Support Vector Classifier
svc_param = {'C': [0.5, 0.7, 0.9, 1], 'kernel': ['rbf', 'poly', 'sigmoid', 'linear']}
svc = GridSearchCV(SVC(), svc_param)
svc.fit(X_train, y_train)
svc_bestparam = svc.best_estimator_
```

```
# DecisionTree Classifier
tree_param = {"criterion": ["gini", "entropy"], "max_depth": list(range(2,4,1)),
              "min_samples_leaf": list(range(5,7,1))}
dt = GridSearchCV(DecisionTreeClassifier(), tree_param)
dt.fit(X_train, y_train)
dt_bestparam = dt.best_estimator_
```

```
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:81
8: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:81
8: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:81
8: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:81
8: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:81
8: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:81
8: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:81
8: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
```

```
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/model_selection/_validation.
py:372: FitFailedWarning:
35 fits failed out of a total of 70.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score
='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
35 fits failed with the following error:
Traceback (most recent call last):
  File "/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/model_selection/_val
idation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logist
ic.py", line 1461, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logist
ic.py", line 449, in _check_solver
    % (solver, penalty)
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/model_selection/_search.py:9
72: UserWarning: One or more of the test scores are non-finite: [       nan 0.9146208
6        nan 0.94046921        nan 0.94176791
        nan 0.94046921        nan 0.94306661        nan 0.94436531
        nan 0.94306661]
  category=UserWarning,
```

In [41]:

```
# Model accuracy 출력

lr_score = cross_val_score(lr_bestparam, X_train, y_train, cv=5)
print('Logistic Regression Cross Validation Score: ', round(lr_score.mean() * 100, 2).


knn_score = cross_val_score(knn_bestparam, X_train, y_train, cv=5)
print('Knears Neighbors Cross Validation Score', round(knn_score.mean() * 100, 2).asty

svc_score = cross_val_score(svc_bestparam, X_train, y_train, cv=5)
print('Support Vector Classifier Cross Validation Score', round(svc_score.mean() * 100

dt_score = cross_val_score(dt_bestparam, X_train, y_train, cv=5)
print('DecisionTree Classifier Cross Validation Score', round(dt_score.mean() * 100, 2
```

```
Logistic Regression Cross Validation Score:  94.44%
Knears Neighbors Cross Validation Score 94.44%
Support Vector Classifier Cross Validation Score 94.83%
DecisionTree Classifier Cross Validation Score 92.5%
```

```
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:81
8: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:81
8: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:81
8: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

In [42]:

```
lr_pred = cross_val_predict(lr_bestparam, X_train, y_train, cv=5)

precision, recall, threshold = precision_recall_curve(y_train, lr_pred)
y_pred = lr_pred

print('Logistic Regression')
print('---' * 20)
print('Recall Score: {:.2f}'.format(recall_score(y_train, y_pred)))
print('Precision Score: {:.2f}'.format(precision_score(y_train, y_pred)))
print('F1 Score: {:.2f}'.format(f1_score(y_train, y_pred)))
print('Accuracy Score: {:.2f}'.format(accuracy_score(y_train, y_pred)))
print('---' * 20)

knn_pred = cross_val_predict(knn_bestparam, X_train, y_train, cv=5)

precision, recall, threshold = precision_recall_curve(y_train, knn_pred)
y_pred = knn_pred

print('KNeighbors Classifier')
print('---' * 20)
print('Recall Score: {:.2f}'.format(recall_score(y_train, y_pred)))
print('Precision Score: {:.2f}'.format(precision_score(y_train, y_pred)))
print('F1 Score: {:.2f}'.format(f1_score(y_train, y_pred)))
print('Accuracy Score: {:.2f}'.format(accuracy_score(y_train, y_pred)))
print('---' * 20)

svc_pred = cross_val_predict(svc_bestparam, X_train, y_train, cv=5)

precision, recall, threshold = precision_recall_curve(y_train, svc_pred)
y_pred = svc_pred
```

```
print('Support Vector Machine')
print('---' * 20)
print('Recall Score: {:.2f}'.format(recall_score(y_train, y_pred)))
print('Precision Score: {:.2f}'.format(precision_score(y_train, y_pred)))
print('F1 Score: {:.2f}'.format(f1_score(y_train, y_pred)))
print('Accuracy Score: {:.2f}'.format(accuracy_score(y_train, y_pred)))
print('---' * 20)

dt_pred = cross_val_predict(dt_bestparam, X_train, y_train, cv=5)

precision, recall, threshold = precision_recall_curve(y_train, dt_pred)
y_pred = dt_pred

print('Decision Tree')
print('---' * 20)
print('Recall Score: {:.2f}'.format(recall_score(y_train, y_pred)))
print('Precision Score: {:.2f}'.format(precision_score(y_train, y_pred)))
print('F1 Score: {:.2f}'.format(f1_score(y_train, y_pred)))
print('Accuracy Score: {:.2f}'.format(accuracy_score(y_train, y_pred)))
print('---' * 20)
```

```
Logistic Regression
------------------------------------------------------------
Recall Score: 0.92
Precision Score: 0.96
F1 Score: 0.94
Accuracy Score: 0.94
------------------------------------------------------------
KNeighbors Classifier
------------------------------------------------------------
Recall Score: 0.91
Precision Score: 0.97
F1 Score: 0.94
Accuracy Score: 0.94
------------------------------------------------------------
Support Vector Machine
------------------------------------------------------------
Recall Score: 0.92
Precision Score: 0.97
F1 Score: 0.94
Accuracy Score: 0.95
------------------------------------------------------------
Decision Tree
------------------------------------------------------------
Recall Score: 0.89
Precision Score: 0.95
F1 Score: 0.92
Accuracy Score: 0.92
------------------------------------------------------------
```

```
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:81
8: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:81
8: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
/opt/conda/envs/py37/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:81
8: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

```
In [ ]:
```