creditCardFraud

May 21, 2023

[ ]:
```
# Dataset : https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud?
↪datasetId=310&searchQuery=anomaly
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
```

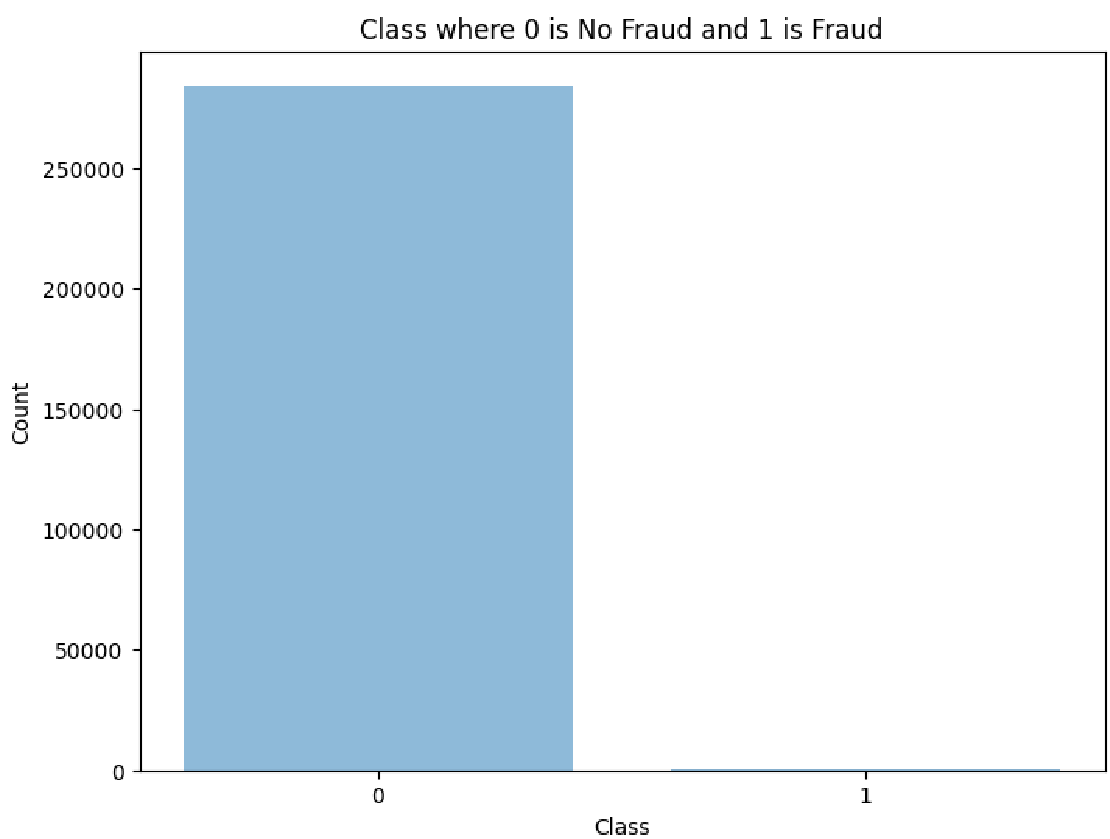
[ ]:
```
df = pd.read_csv("notebooks/creditcard.csv")
df.head()
```

[ ]:
```
   Time        V1        V2        V3        V4        V5        V6        V7  \
0   0.0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599
1   0.0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803
2   1.0 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461
3   1.0 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609
4   2.0 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941

         V8        V9  …       V21       V22       V23       V24       V25  \
0  0.098698  0.363787  … -0.018307  0.277838 -0.110474  0.066928  0.128539
1  0.085102 -0.255425  … -0.225775 -0.638672  0.101288 -0.339846  0.167170
2  0.247676 -1.514654  …  0.247998  0.771679  0.909412 -0.689281 -0.327642
3  0.377436 -1.387024  … -0.108300  0.005274 -0.190321 -1.175575  0.647376
4 -0.270533  0.817739  … -0.009431  0.798278 -0.137458  0.141267 -0.206010

        V26       V27       V28  Amount  Class
0 -0.189115  0.133558 -0.021053  149.62      0
1  0.125895 -0.008983  0.014724    2.69      0
2 -0.139097 -0.055353 -0.059752  378.66      0
3 -0.221929  0.062723  0.061458  123.50      0
4  0.502292  0.219422  0.215153   69.99      0

[5 rows x 31 columns]
```

[ ]:
```
df.columns
```

```
[ ]: Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
            'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
            'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
            'Class'],
           dtype='object')
```

```
[ ]: count = df['Class'].value_counts()
x_pos = count.index.astype(str)
fig, ax = plt.subplots(figsize=(8, 6))
ax.bar(x_pos, count, align='center', alpha=0.5)
plt.ylabel('Count')
plt.xlabel('Class')
plt.title('Class where 0 is No Fraud and 1 is Fraud')

plt.show()
```




```
[ ]: # Feature Scaling
# Scaling the time and amount coulmns as other columns are already scaled.
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

```
df['updated_amount'] = sc.fit_transform(df['Amount'].values.reshape(-1,1))
df['updated_time'] = sc.fit_transform(df['Time'].values.reshape(-1,1))
df.head()
```

[ ]:

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

         V8        V9  …       V23       V24       V25       V26       V27  \
0  0.098698  0.363787  … -0.110474  0.066928  0.128539 -0.189115  0.133558
1  0.085102 -0.255425  …  0.101288 -0.339846  0.167170  0.125895 -0.008983
2  0.247676 -1.514654  …  0.909412 -0.689281 -0.327642 -0.139097 -0.055353
3  0.377436 -1.387024  … -0.190321 -1.175575  0.647376 -0.221929  0.062723
4 -0.270533  0.817739  … -0.137458  0.141267 -0.206010  0.502292  0.219422

        V28  Amount  Class  updated_amount  updated_time
0 -0.021053  149.62      0        0.244964     -1.996583
1  0.014724    2.69      0       -0.342475     -1.996583
2 -0.059752  378.66      0        1.160686     -1.996562
3  0.061458  123.50      0        0.140534     -1.996562
4  0.215153   69.99      0       -0.073403     -1.996541

[5 rows x 33 columns]
```

[ ]:

```
# Splitting and Sampling

from sklearn.model_selection import train_test_split
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,␣
 ↪shuffle=True, random_state=0)

#count_fraud = df['Class'].value_counts()[1]
#print(count_fraud)
fraud = df.loc[df['Class'] == 1]
noFraud = df.loc[df['Class'] == 0][:len(fraud)]
balanced_df = pd.concat([fraud, noFraud])
balanced_df = balanced_df.sample(frac=1, random_state=42).reset_index(drop=True)
balanced_df = balanced_df.drop('Time' , axis = 1)
balanced_df = balanced_df.drop('Amount' , axis = 1)
balanced_df.head()
```

[ ]:

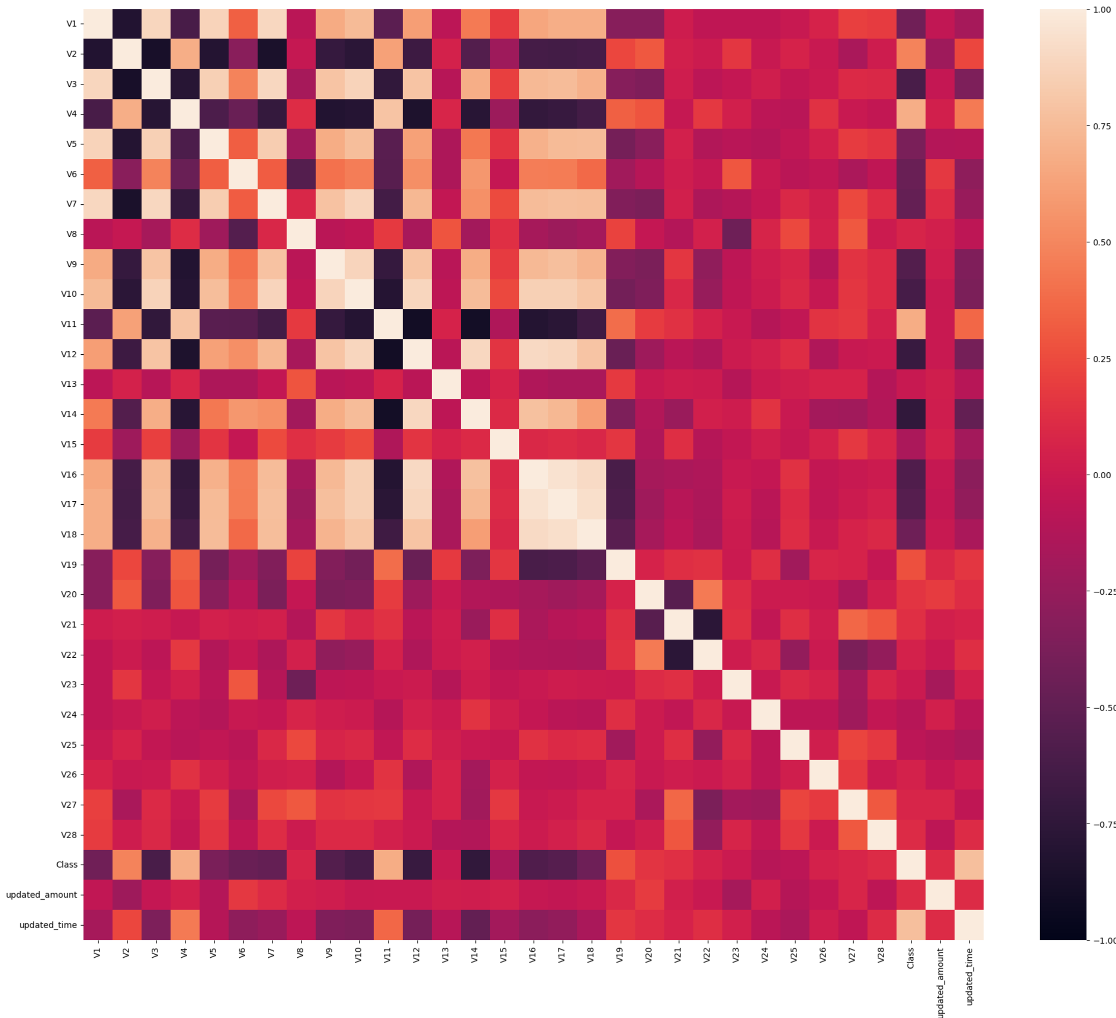
```
         V1        V2        V3        V4        V5        V6        V7  \
0 -0.427191  0.745708  1.761811 -0.165130  0.058298 -0.213413  0.647323
```

```
1 -0.613696  3.698772 -5.534941  5.620486  1.649263 -2.335145 -0.907188
2  1.171439  0.474974  0.011761  1.264303  0.116234 -0.865986  0.554393
3 -6.682832 -2.714268 -5.774530  1.449792 -0.661836 -1.148650  0.849686
4 -6.713407  3.921104 -9.746678  5.148263 -5.151563 -2.099389 -5.937767

         V8        V9       V10  …       V22       V23       V24       V25  \
0  0.073464 -0.291864  0.064800  … -0.432070  0.013164  0.161606 -0.401310
1  0.706362 -3.747646 -4.230984  … -0.471379 -0.075890 -0.667909 -0.642848
2 -0.276375 -0.471302  0.029104  …  0.278843 -0.097491  0.426278  0.744938
3  0.433427 -1.315646 -2.796332  …  1.187013  0.335821  0.215683  0.803110
4  3.578780 -4.684952 -8.537758  … -0.451086  0.127214 -0.339450  0.394096

        V26       V27       V28  Class  updated_amount  updated_time
0  0.047423  0.102549 -0.116571      0       -0.316767     -1.994962
1  0.070600  0.488410  0.292345      1       -0.353229      1.243705
2 -0.274728  0.008472  0.015492      0       -0.273268     -1.993214
3  0.044033 -0.054988  0.082337      1        0.595357      1.165033
4  1.075295  1.649906 -0.394905      1        0.657967     -0.200658

[5 rows x 31 columns]
```

[ ]:
```python
# Printing the balanced classes
count = balanced_df['Class'].value_counts()
x_pos = count.index.astype(str)
fig, ax = plt.subplots(figsize=(8, 6))
ax.bar(x_pos, count, align='center', alpha=0.5)
plt.ylabel('Count')
plt.xlabel('Class')
plt.title('Class where 0 is No Fraud and 1 is Fraud')

plt.show()
```

[ ]:
```
#%pip install seaborn
import seaborn as sns
f, (ax1) = plt.subplots(1, figsize=(24,20))
sns.heatmap(balanced_df.corr(),annot_kws={'size':20}, ax=ax1, vmin = -1, vmax =␣
 ↪1)
```

[ ]: <AxesSubplot:>



[ ]:
```python
# Using the balanced dataset
Xb = balanced_df.drop('Class', axis = 1)
yb = balanced_df['Class']
X_btrain, X_btest, y_btrain, y_btest = train_test_split(Xb, yb, test_size=0.2,␣
↪random_state=42)
```

[ ]:
```python
print(X_btrain)
```

```
            V1        V2         V3        V4         V5        V6         V7  \
962 -12.833631  7.508790 -20.491952  7.465780 -11.575304 -5.140999 -14.020564
762  -0.646513  1.004199   1.616224 -0.099628  -0.122477 -0.671327   0.656183
334  -7.427924  2.948209  -8.678550  5.185303  -4.761090 -0.957095  -7.773380
889   1.243848  0.524526  -0.538884  1.209196   0.479538 -0.197429   0.049166
529  -1.309441  1.786495  -1.371070  1.214335  -0.336642 -1.390120  -1.709109
..         …        …          …       …          …         …          …
106  -7.901421  2.720472  -7.885936  6.348334  -5.480119 -0.333059  -8.682376
```

```
270  -2.207631  3.259076  -5.436365  3.684737  -3.066401 -0.671323  -3.696178
860  -0.471796  0.523169   1.948967  0.995503   0.379069 -0.577466   0.521413
435  -1.554216  1.694229  -0.903334  2.425436  -2.899787  0.133028  -0.286226
102  -1.298359  1.079671  -0.180678  1.287839   1.858273 -2.223695   0.525167

           V8        V9        V10  …       V21       V22       V23  \
962  8.332120 -4.337713 -15.563791  …  2.966842  0.615344 -0.766495
762  0.009755 -0.635963  -0.047364  … -0.147934 -0.420046  0.061424
334  0.717309 -3.682359  -8.403150  … -0.299847  0.610479  0.789023
889  0.037792  0.128119  -0.552903  … -0.051660 -0.084089 -0.192846
529  0.667748 -1.699809  -3.843911  …  0.533521 -0.022180 -0.299556
..        …         …          …  …       …         …         …
106  1.164431 -4.542447  -7.748480  …  0.077739  1.092437  0.320133
270  1.822272 -3.049653  -6.353887  …  0.920899  0.037675  0.026754
860 -0.128940 -0.704962   0.186559  … -0.099422 -0.287139  0.151288
435  0.555945 -1.394918  -2.892612  …  0.493436  0.733393  0.202350
102 -0.096874 -0.168893  -2.544410  … -0.332983 -0.851270 -0.370800

          V24       V25       V26       V27       V28  updated_amount  \
962  0.431261 -0.104975 -0.010091 -2.400811 -0.720557        0.062692
762  0.520997 -0.238845  0.030135  0.140481  0.101163       -0.293338
334 -0.564512  0.201196 -0.111225  1.144599  0.102280        0.168281
889 -0.917392  0.681953 -0.194419  0.045917  0.040136       -0.349231
529 -0.226416  0.364360 -0.475102  0.571426  0.293426       -0.349231
..        …         …         …         …         …               …
106 -0.434643 -0.380687  0.213630  0.423620 -0.105169        0.260317
270 -0.791489  0.176493 -0.136312  1.087585  0.373834        0.609390
860  0.490367 -0.725252 -0.741834  0.004784 -0.045977       -0.279465
435  0.492054 -0.183791 -0.199917  0.395201  0.027693        1.086082
102  0.298242  0.442930 -0.522832  0.000105  0.135698       -0.349231

     updated_time
962     -0.019686
762     -1.994519
334     -0.825383
889     -0.974579
529     -1.026718
..            …
106     -0.804199
270     -0.194530
860     -1.991192
435     -0.944003
102     -0.377734

[787 rows x 30 columns]
```

[ ]:
```
print(y_btrain)
```

```
962    1
762    0
334    1
889    1
529    1
      ..
106    1
270    1
860    0
435    1
102    1
Name: Class, Length: 787, dtype: int64
```

[ ]:
```
print(X_btest)
```

```
            V1         V2         V3        V4        V5        V6         V7  \
613  -0.887287   1.390002   1.219686  1.661425  1.009228 -0.733908   0.855829
451  -3.613850  -0.922136  -4.749887  3.373001 -0.545207 -1.171301  -4.172315
731  -0.913600   0.162262   0.541429 -1.931799  0.235402 -0.209263   0.770523
436  -5.140723   3.568751  -5.896245  4.164720 -4.091193 -1.989960  -5.472436
275 -13.192671  12.785971  -9.906650  3.320337 -4.801176  5.760059 -18.750889
..         …         …         …        …         …        …         …
292  -0.948896   0.248414   2.956914  2.813750  0.145539 -0.027353   0.133702
209  -0.264869   3.386140  -3.454997  4.367629  3.336060 -2.053918   0.256890
506   1.954852   1.630056  -4.337200  2.378367  2.113348 -1.583851   0.653745
49   -0.935732   0.170416   2.746261 -1.077965 -0.305594  0.011577  -0.296178
717 -10.281784   6.302385 -13.271718  8.925115 -9.975578 -2.832513 -12.703253

            V8        V9        V10  …        V21       V22       V23  \
613   0.000077 -1.275631  -0.433394  …  -0.083734 -0.346930 -0.050619
451   1.517016 -1.775833  -3.754054  …   0.786787  0.893065  1.034907
731  -0.407195 -1.374754   0.311188  …  -0.382552 -0.546739 -0.320022
436   2.422821 -2.909735  -6.287803  …   1.131130  0.118022 -0.332704
275 -37.353443 -0.391540  -5.052502  …  27.202839 -8.887017  5.303607
..         …         …         …  …        …         …        …
292  -0.307535 -0.125244   1.034940  …  -0.083647  0.416090  0.207537
209  -2.957235 -2.855797  -2.808456  …  -1.394504 -0.166029 -1.452081
506  -0.192892  1.217608  -2.829098  …  -0.474437 -0.974625 -0.048155
49    0.402776 -0.040472  -0.852046  …   0.401212  1.064864 -0.158325
717   6.706846 -7.078424 -12.805683  …   2.479414  0.366933  0.042805

          V24       V25       V26       V27       V28  updated_amount  \
613  0.231044 -0.450760 -0.376205  0.034504  0.157775       -0.322924
451  0.097671 -1.345551 -0.788329  1.055442  0.099971        0.225693
731 -0.928385 -0.080009  0.908687 -0.286881  0.140450        0.046379
436  0.139941  0.324758 -0.180769  0.177810  0.661555        0.046179
275 -0.639435  0.263203 -0.108877  1.269566  0.939407       -0.349231
..        …         …        …        …        …              …
```

```
292  0.716064 -0.602311 -0.064230 -0.315058 -0.272463       -0.350231
209 -0.251815  1.243461  0.452787  0.132218  0.424599       -0.349231
506 -0.023524  0.362192 -0.570709  0.025619  0.081880       -0.349231
49   0.295505 -0.259370  0.754195  0.046664  0.093948       -0.316847
717  0.478279  0.157771  0.329901  0.163504 -0.485552        0.119744

     updated_time
613     -0.989467
451      1.011331
731     -1.992582
436     -0.396644
275     -0.560285
..            …
292     -1.995656
209     -0.773813
506     -0.069278
49      -1.995888
717     -1.128217

[197 rows x 30 columns]
```

[ ]:
```
print(y_btest)
```

```
613    1
451    1
731    0
436    1
275    1
      ..
292    0
209    1
506    1
49     0
717    1
Name: Class, Length: 197, dtype: int64
```

[ ]:
```
X_btrain = X_btrain.values
X_btest = X_btest.values
y_btrain = y_btrain.values
y_btest = y_btest.values
```

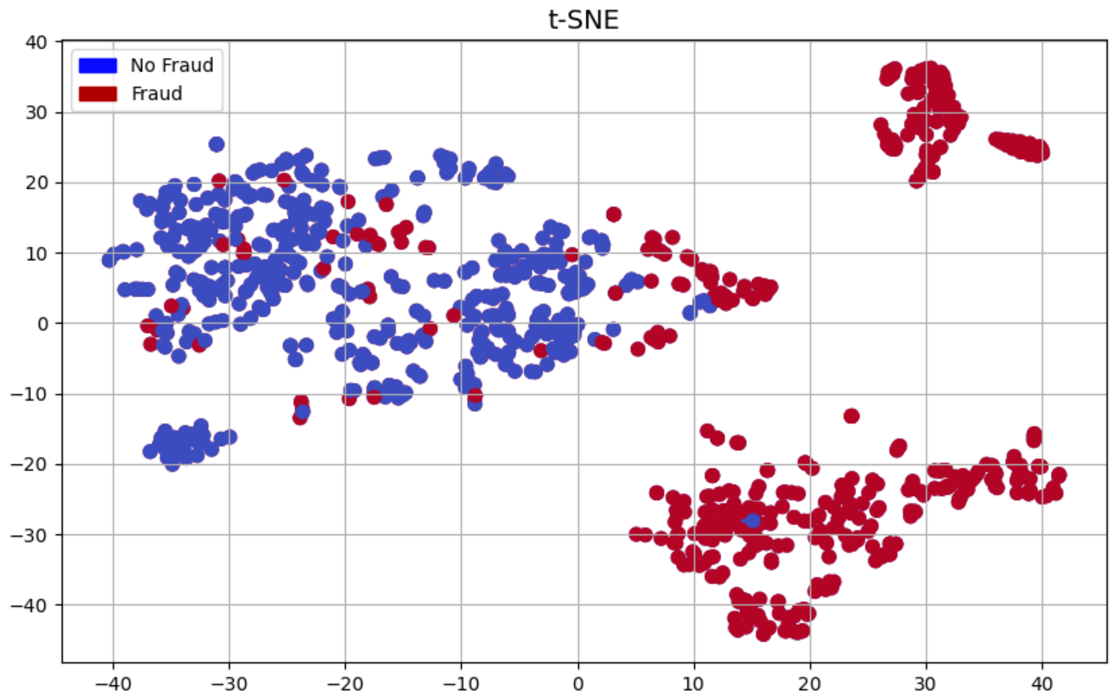
[ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
import numpy as np
from sklearn.metrics import confusion_matrix, accuracy_score

classifier_lr = LogisticRegression()
```

```
classifier_lr.fit(X_btrain, y_btrain)
training_score = cross_val_score(classifier_lr, X_btrain, y_btrain, cv=5)
print("LogisticRegression training accuracy score = " ,(training_score.mean()))

y_bpred = classifier_lr.predict(X_btest)
test_score = accuracy_score(y_btest, y_bpred)
print("LogisticRegression test accuracy score = ", test_score)
```

```
LogisticRegression training accuracy score =  0.9923728130291058
LogisticRegression test accuracy score =  0.9898477157360406
```

[ ]:
```
import matplotlib.patches as mpatches
import time

t0 = time.time()
X_reduced_tsne = TSNE(n_components=2, random_state=42).fit_transform(X.values)
t1 = time.time()
print("T-SNE took {:.2} s".format(t1 - t0))

blue_patch = mpatches.Patch(color='#0A0AFF', label='No Fraud')
red_patch = mpatches.Patch(color='#AF0000', label='Fraud')

f, (ax1) = plt.subplots(1,figsize=(10,6))

# t-SNE scatter plot
ax1.scatter(X_reduced_tsne[:,0], X_reduced_tsne[:,1], c=(y == 0),␣
 ↪cmap='coolwarm', label='No Fraud', linewidths=2)
ax1.scatter(X_reduced_tsne[:,0], X_reduced_tsne[:,1], c=(y == 1),␣
 ↪cmap='coolwarm', label='Fraud', linewidths=2)
ax1.set_title('t-SNE', fontsize=14)

ax1.grid(True)

ax1.legend(handles=[blue_patch, red_patch])
plt.show()
```

```
T-SNE took 9.1 s
```



[ ]:
```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC

classifier_kn = KNeighborsClassifier()
classifier_kn.fit(X_btrain, y_btrain)
training_score = cross_val_score(classifier_kn, X_btrain, y_btrain, cv=5)
print("KNeighborsClassifier training accuracy score = " , (training_score.
    ↪mean()))

y_bpred = classifier_kn.predict(X_btest)
test_score = accuracy_score(y_btest, y_bpred)
print("KNeighborsClassifier test accuracy score = ", test_score)

classifier_svc = SVC()
classifier_svc.fit(X_btrain, y_btrain)
training_score = cross_val_score(classifier_svc, X_btrain, y_btrain, cv=5)
print("SVC training accuracy score = " , (training_score.mean()))

y_bpred = classifier_svc.predict(X_btest)
test_score = accuracy_score(y_btest, y_bpred)
print("SVC test accuracy score = ", test_score)
```

```
KNeighborsClassifier training accuracy score =  0.9542852535676852
KNeighborsClassifier test accuracy score =  0.9441624365482234
SVC training accuracy score =  0.9606466177537692
```

```
SVC test accuracy score =  0.9593908629441624
```

[ ]:

```
```

[ ]:

```
```

[ ]:

```
```