```
import numpy as np
import pandas as pd
import math
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import seaborn as sns
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
import warnings
from sklearn import linear_model, preprocessing
from sklearn.metrics import mean_squared_error, classification_report, precision_recall_fscore_support
from collections import Counter
from sklearn.linear_model import LogisticRegression
from imblearn.over_sampling import SMOTE
from sklearn.neural_network import MLPClassifier
import itertools
warnings.filterwarnings('ignore')
%matplotlib inline
df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Datasets/creditcard.csv')
df.head(n=10)
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278
5	2.0	-0.425966	0.960523	1.141109	-0.168252	0.420987	-0.029728	0.476201	0.260314	-0.568671	...	-0.208254	-0.559825
6	4.0	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.005159	0.081213	0.464960	...	-0.167716	-0.270710
7	7.0	-0.644269	1.417964	1.074380	-0.492199	0.948934	0.428118	1.120631	-3.807864	0.615375	...	1.943465	-1.015455
8	7.0	-0.894286	0.286157	-0.113192	-0.271526	2.669599	3.721818	0.370145	0.851084	-0.392048	...	-0.073425	-0.268092
9	9.0	-0.338262	1.119593	1.044367	-0.222187	0.499361	-0.246761	0.651583	0.069539	-0.736727	...	-0.246914	-0.633753

10 rows × 31 columns

```
df.shape
```

```
(284807, 31)
```

```
df.columns.tolist()
```

```
['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']
```

```
new_df = df.sample(frac=1,random_state=42).reset_index(drop=True)
new_df.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	
0	41505.0	-16.526507	8.584972	-18.649853	9.505594	-13.793819	-2.832404	-16.701694	7.517344	-8.507059	...	1.190739	-1.127670	-2.358579	0.673461	-1.413700	-0.462762	-2.018575	-1.04
1	44261.0	0.339812	-2.743745	-0.134070	-1.385729	-1.451413	1.015887	-0.524379	0.224060	0.899746	...	-0.213436	-0.942525	-0.526819	-1.156992	0.311211	-0.746647	0.040996	0.10
2	35484.0	1.399590	-0.590701	0.168619	-1.029950	-0.539806	0.040444	-0.712567	0.002299	-0.971747	...	0.102398	0.168269	-0.166639	-0.810250	0.505083	-0.232340	0.011409	0.00
3	167123.0	-0.432071	1.647895	-1.669361	-0.349504	0.785785	-0.630647	0.276990	0.586025	-0.484715	...	0.358932	0.873663	-0.178642	-0.017171	-0.207392	-0.157756	-0.237386	0.00
4	168473.0	2.014160	-0.137394	-1.015839	0.327269	-0.182179	-0.956571	0.043241	-0.160746	0.363241	...	-0.238644	-0.616400	0.347045	0.061561	-0.360196	0.174730	-0.078043	-0.07

5 rows × 31 columns

```
df_1 = new_df.iloc[:227845,:].reset_index(drop=True)
test = new_df.iloc[227845:,:].reset_index(drop=True)
train, valid = train_test_split(df_1,test_size=0.2)
```

```
train.info()
print('*'*100)
valid.info()
print('*'*100)
test.info()
```

```
 14  V14     45569 non-null  float64
 15  V15     45569 non-null  float64
```

```
 15  V15     45569 non-null  float64
 16  V16     45569 non-null  float64
 17  V17     45569 non-null  float64
 18  V18     45569 non-null  float64
 19  V19     45569 non-null  float64
 20  V20     45569 non-null  float64
 21  V21     45569 non-null  float64
 22  V22     45569 non-null  float64
 23  V23     45569 non-null  float64
 24  V24     45569 non-null  float64
 25  V25     45569 non-null  float64
 26  V26     45569 non-null  float64
 27  V27     45569 non-null  float64
 28  V28     45569 non-null  float64
 29  Amount  45569 non-null  float64
 30  Class   45569 non-null  int64
dtypes: float64(30), int64(1)
memory usage: 11.1 MB
****************************************************************************************************
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56962 entries, 0 to 56961
Data columns (total 31 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Time    56962 non-null  float64
 1   V1      56962 non-null  float64
 2   V2      56962 non-null  float64
 3   V3      56962 non-null  float64
 4   V4      56962 non-null  float64
 5   V5      56962 non-null  float64
 6   V6      56962 non-null  float64
 7   V7      56962 non-null  float64
 8   V8      56962 non-null  float64
 9   V9      56962 non-null  float64
 10  V10     56962 non-null  float64
 11  V11     56962 non-null  float64
 12  V12     56962 non-null  float64
 13  V13     56962 non-null  float64
 14  V14     56962 non-null  float64
 15  V15     56962 non-null  float64
 16  V16     56962 non-null  float64
 17  V17     56962 non-null  float64
 18  V18     56962 non-null  float64
 19  V19     56962 non-null  float64
 20  V20     56962 non-null  float64
 21  V21     56962 non-null  float64
 22  V22     56962 non-null  float64
 23  V23     56962 non-null  float64
 24  V24     56962 non-null  float64
 25  V25     56962 non-null  float64
 26  V26     56962 non-null  float64
 27  V27     56962 non-null  float64
 28  V28     56962 non-null  float64
 29  Amount  56962 non-null  float64
 30  Class   56962 non-null  int64
dtypes: float64(30), int64(1)
memory usage: 13.5 MB
```

```
print(pd.isnull(train).sum())
```

```
print('*'*100)
print(pd.isnull(valid).sum())
print('*'*100)
print(pd.isnull(test).sum())
```

```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
****************************************************************************************************
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
```

```
V21       0
V22       0
V23       0
```

```
print(train['Class'].value_counts())
print('*'*100)
print(valid['Class'].value_counts())
print('*'*100)
print(test['Class'].value_counts())
```

```
0    181968
1       308
Name: Class, dtype: int64
****************************************************************************************************
0    45490
1       79
Name: Class, dtype: int64
****************************************************************************************************
0    56857
1      105
Name: Class, dtype: int64
```

```
train.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')
```

```
plt.figure(figsize=(10,6))
labels=['Not Fraud' , 'Frauds']
explode = [.01,.01]
color=['LightGreen' , 'Blue']
sizes=df.Class.value_counts().values

plt.pie(sizes,explode,labels,autopct="%1.1f%%", colors = color)
plt.show()
```

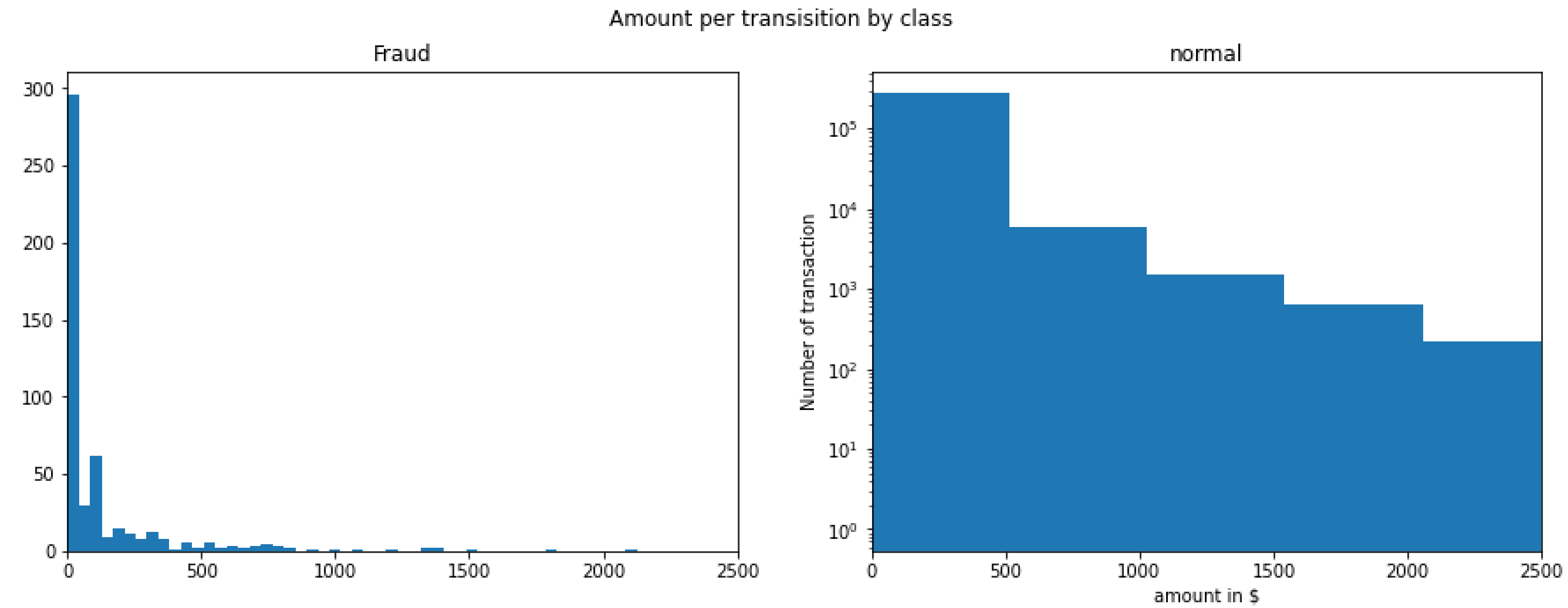
```
fraud=df[df['Class']==1]
normal=df[df['Class']==0]

f,(ax1,ax2)=plt.subplots(1,2,figsize=(15,5),sharex=True)
f.suptitle('Amount per transisition by class')
bins =50
ax1.hist(fraud.Amount , bins=bins)
ax1.set_title('Fraud')

ax2.hist(normal.Amount,bins=bins)
ax2.set_title('normal')

plt.xlabel('amount in $')
plt.ylabel('Number of transaction')

plt.xlim(0,2500)
plt.yscale('log')
plt.show()
```




```
df[df['Class']==0].describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
Time	284315.0	94838.202258	47484.015786	0.000000	54230.000000	84711.000000	139333.000000	172792.000000
V1	284315.0	0.008258	1.929814	-56.407510	-0.917544	0.020023	1.316218	2.454930
V2	284315.0	-0.006271	1.636146	-72.715728	-0.599473	0.064070	0.800446	18.902453
V3	284315.0	0.012171	1.459429	-48.325589	-0.884541	0.182158	1.028372	9.382558
V4	284315.0	-0.007860	1.399333	-5.683171	-0.850077	-0.022405	0.737624	16.875344
V5	284315.0	0.005453	1.356952	-113.743307	-0.689398	-0.053457	0.612181	34.801666
V6	284315.0	0.002419	1.329913	-26.160506	-0.766847	-0.273123	0.399619	73.301626
V7	284315.0	0.009637	1.178812	-31.764946	-0.551442	0.041138	0.571019	120.589494
V8	284315.0	-0.000987	1.161283	-73.216718	-0.208633	0.022041	0.326200	18.709255
V9	284315.0	0.004467	1.089372	-6.290730	-0.640412	-0.049964	0.598230	15.594995
V10	284315.0	0.009824	1.044204	-14.741096	-0.532880	-0.091872	0.455135	23.745136
V11	284315.0	-0.006576	1.003112	-4.797473	-0.763447	-0.034923	0.736362	10.002190
V12	284315.0	0.010832	0.945939	-15.144988	-0.402102	0.141679	0.619207	7.848392
V13	284315.0	0.000189	0.995067	-5.791881	-0.648067	-0.013547	0.662492	7.126883
V14	284315.0	0.012064	0.897007	-18.392091	-0.422453	0.051947	0.494104	10.526766
V15	284315.0	0.000161	0.915060	-4.391307	-0.582812	0.048294	0.648842	8.877742
V16	284315.0	0.007164	0.844772	-10.115560	-0.465543	0.067377	0.523738	17.315112
V17	284315.0	0.011535	0.749457	-17.098444	-0.482644	-0.064833	0.399922	9.253526
V18	284315.0	0.003887	0.824919	-5.366660	-0.497414	-0.002787	0.501103	5.041069
V19	284315.0	-0.001178	0.811733	-7.213527	-0.456366	0.003117	0.457499	5.591971
V20	284315.0	-0.000644	0.769404	-54.497720	-0.211764	-0.062646	0.132401	39.420904
V21	284315.0	-0.001235	0.716743	-34.830382	-0.228509	-0.029821	0.185626	22.614889
V22	284315.0	-0.000024	0.723668	-10.933144	-0.542403	0.006736	0.528407	10.503090
V23	284315.0	0.000070	0.621541	-44.807735	-0.161702	-0.011147	0.147522	22.528412
V24	284315.0	0.000182	0.605776	-2.836627	-0.354425	0.041082	0.439869	4.584549
V25	284315.0	-0.000072	0.520673	-10.295397	-0.317145	0.016417	0.350594	7.519589

```
df[df['Class']==1].describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
Time	492.0	80746.806911	47835.365138	406.000000	41241.500000	75568.500000	128483.000000	170348.000000
V1	492.0	-4.771948	6.783687	-30.552380	-6.036063	-2.342497	-0.419200	2.132386
V2	492.0	3.623778	4.291216	-8.402154	1.188226	2.717869	4.971257	22.057729
V3	492.0	-7.033281	7.110937	-31.103685	-8.643489	-5.075257	-2.276185	2.250210
V4	492.0	4.542029	2.873318	-1.313275	2.373050	4.177147	6.348729	12.114672
V5	492.0	-3.151225	5.372468	-22.105532	-4.792835	-1.522962	0.214562	11.095089
V6	492.0	-1.397737	1.858124	-6.406267	-2.501511	-1.424616	-0.413216	6.474115
V7	492.0	-5.568731	7.206773	-43.557242	-7.965295	-3.034402	-0.945954	5.802537
V8	492.0	0.570636	6.797831	-41.044261	-0.195336	0.621508	1.764879	20.007208
V9	492.0	-2.581123	2.500896	-13.434066	-3.872383	-2.208768	-0.787850	3.353525
V10	492.0	-5.676883	4.897341	-24.588262	-7.756698	-4.578825	-2.614184	4.031435
V11	492.0	3.800173	2.678605	-1.702228	1.973397	3.586218	5.307078	12.018913
V12	492.0	-6.259393	4.654458	-18.683715	-8.688177	-5.502530	-2.974088	1.375941
V13	492.0	-0.109334	1.104518	-3.127795	-0.979117	-0.065566	0.672964	2.815440
V14	492.0	-6.971723	4.278940	-19.214325	-9.692723	-6.729720	-4.282821	3.442422
V15	492.0	-0.092929	1.049915	-4.498945	-0.643539	-0.057227	0.609189	2.471358
V16	492.0	-4.139946	3.865035	-14.129855	-6.562915	-3.549795	-1.226043	3.139656
V17	492.0	-6.665836	6.970618	-25.162799	-11.945057	-5.302949	-1.341940	6.739384
V18	492.0	-2.246308	2.899366	-9.498746	-4.664576	-1.664346	0.091772	3.790316
V19	492.0	0.680659	1.539853	-3.681904	-0.299423	0.646807	1.649318	5.228342
V20	492.0	0.372319	1.346635	-4.128186	-0.171760	0.284693	0.822445	11.059004
V21	492.0	0.713588	3.869304	-22.797604	0.041787	0.592146	1.244611	27.202839
V22	492.0	0.014049	1.494602	-8.887017	-0.533764	0.048434	0.617474	8.361985
V23	492.0	-0.040308	1.579642	-19.254328	-0.342175	-0.073135	0.308378	5.466230
V24	492.0	-0.105130	0.515577	-2.028024	-0.436809	-0.060795	0.285328	1.091435
V25	492.0	0.041449	0.797205	-4.781606	-0.314348	0.088371	0.456515	2.208209
V26	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

```
scaler = preprocessing.MinMaxScaler(feature_range=(0, 1))
names = train.columns
d = scaler.fit_transform(train)
scaled_df = pd.DataFrame(d, columns=names)
scaled_df.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount
0	0.357297	0.977844	0.735707	0.752364	0.196624	0.578518	0.560240	0.478895	0.793422	0.465269	...	0.556878	0.426547	0.671011	0.566537	0.596512	0.566346	0.647533	0.341033	0.000053
1	0.739479	0.984386	0.733790	0.734158	0.299318	0.569061	0.515456	0.490651	0.786724	0.460816	...	0.564843	0.506589	0.668041	0.178131	0.593288	0.361145	0.650135	0.339143	0.006744
2	0.244166	0.976593	0.741837	0.778640	0.265097	0.556790	0.494146	0.485056	0.785245	0.454831	...	0.556532	0.430576	0.669361	0.288011	0.598165	0.449564	0.649259	0.341404	0.000047
3	0.428405	0.975364	0.738307	0.789678	0.280893	0.550457	0.492521	0.484703	0.784267	0.478000	...	0.556181	0.435893	0.668979	0.353918	0.604416	0.462570	0.649551	0.341545	0.001586
4	0.746834	0.944087	0.746032	0.777418	0.244498	0.570822	0.463398	0.497785	0.780291	0.463420	...	0.565804	0.528292	0.666980	0.388514	0.555949	0.398285	0.658452	0.343861	0.001027

```
#Oversampling
X_train, y_train = train.drop(['Class'], axis = 1), train['Class']
sm = SMOTE(random_state = 42)
X_train, y_train = sm.fit_resample(X_train, y_train)#.ravel())
```

```
lg=LogisticRegression()
lg.fit(X_train,y_train)
lg_pred=lg.predict(valid.drop(['Class'],axis=1))
```

```
print(classification_report(valid['Class'],lg_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     45490
           1       0.08      0.89      0.14        79

    accuracy                           0.98     45569
   macro avg       0.54      0.93      0.57     45569
weighted avg       1.00      0.98      0.99     45569
```

```
mlp_clf = MLPClassifier(hidden_layer_sizes=(10,2),
                        max_iter = 10,activation = 'relu',
                        solver = 'adam')
```

```
mlp_clf.fit(X_train, y_train)
```

```
MLPClassifier(hidden_layer_sizes=(10, 2), max_iter=10)
```

```
mlp_pred = mlp_clf.predict(valid.drop(['Class'],axis=1))
print(classification_report(valid['Class'],mlp_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     45490
           1       0.00      0.00      0.00        79

    accuracy                           1.00     45569
   macro avg       0.50      0.50      0.50     45569
weighted avg       1.00      1.00      1.00     45569
```

```
mlp_clf_iter25 = MLPClassifier(hidden_layer_sizes=(15,2),
                        max_iter = 25,activation = 'relu',
                        solver = 'adam')

mlp_clf_iter25.fit(X_train, y_train)

y_pred_25 = mlp_clf_iter25.predict(test.drop(['Class'],axis=1))

print('Accuracy: {:.2f}'.format(accuracy_score(test['Class'], y_pred_25)))
```

```
Accuracy: 1.00
```

```
mlp_clf_iter50 = MLPClassifier(hidden_layer_sizes=(15,2),
                        max_iter = 50,activation = 'relu',
                        solver = 'adam')

mlp_clf_iter50.fit(X_train, y_train)

y_pred_50 = mlp_clf_iter50.predict(test.drop(['Class'],axis=1))

print('Accuracy: {:.2f}'.format(accuracy_score(test['Class'], y_pred_50)))
```

```
Accuracy: 0.98
```

```
mlp_clf_iter75 = MLPClassifier(hidden_layer_sizes=(15,2),
                        max_iter = 75,activation = 'relu',
                        solver = 'adam')

mlp_clf_iter75.fit(X_train, y_train)

y_pred_75 = mlp_clf_iter75.predict(test.drop(['Class'],axis=1))

print('Accuracy: {:.2f}'.format(accuracy_score(test['Class'], y_pred_75)))
```

```
Accuracy: 0.00
```