```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('/content/drive/My Drive/kddcup99_csv.csv')
```

```
df.head()
```

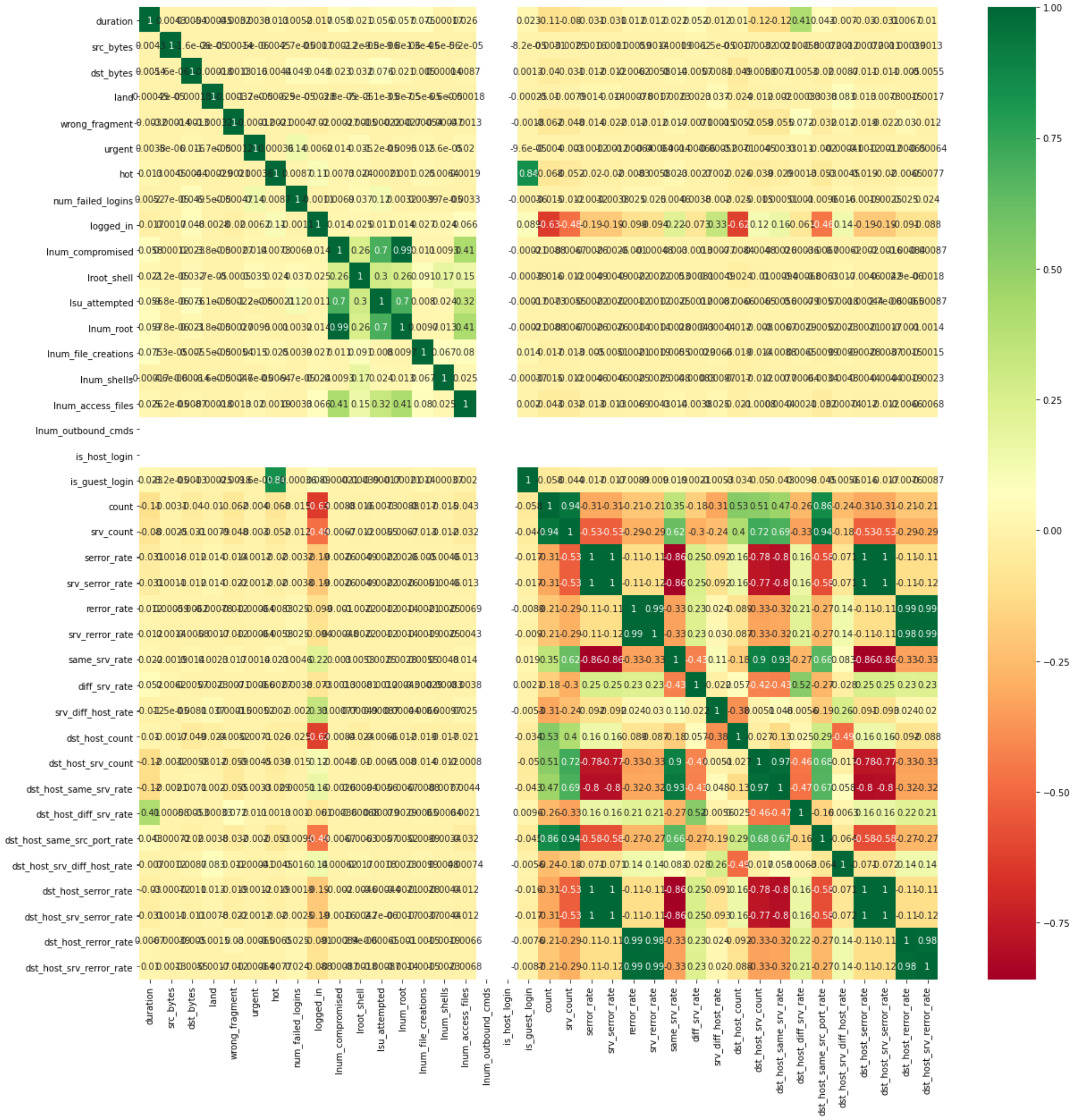
	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragmen
0	0	tcp	http	SF	181	5450	0	(
1	0	tcp	http	SF	239	486	0	(
2	0	tcp	http	SF	235	1337	0	(
3	0	tcp	http	SF	219	1337	0	(
4	0	tcp	http	SF	217	2032	0	(

```
df.describe()
```

	duration	src_bytes	dst_bytes	land	wrong_fragment	
count	494020.000000	4.940200e+05	4.940200e+05	494020.000000	494020.000000	49402
mean	47.979400	3.025616e+03	8.685308e+02	0.000045	0.006433	
std	707.747185	9.882191e+05	3.304003e+04	0.006673	0.134805	
min	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	
25%	0.000000	4.500000e+01	0.000000e+00	0.000000	0.000000	
50%	0.000000	5.200000e+02	0.000000e+00	0.000000	0.000000	
75%	0.000000	1.032000e+03	0.000000e+00	0.000000	0.000000	
max	58329.000000	6.933756e+08	5.155468e+06	1.000000	3.000000	

```
import seaborn as sns
import matplotlib.pyplot as plt
#get correlations of each features in dataset
corrmat = df.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarnir
  import pandas.util.testing as tm
```



Removal of redundant features¶

```
df['lnum_outbound_cmds'].value_counts()
df.drop('lnum_outbound_cmds', axis=1, inplace=True)
df['is_host_login'].value_counts()
df.drop('is_host_login', axis=1, inplace=True)
df['wrong_fragment'].value_counts()
df.drop('wrong_fragment', axis=1, inplace=True)
df['hot'].value_counts()
df.drop('hot', axis=1, inplace=True)
df['num_failed_logins'].value_counts()
df.drop('num_failed_logins', axis=1, inplace=True)
df['logged_in'].value_counts()
df.drop('logged_in', axis=1, inplace=True)
df['lroot_shell'].value_counts()
df.drop('lroot_shell', axis=1, inplace=True)
df['lnum_file_creations'].value_counts()
df.drop('lnum_file_creations', axis=1, inplace=True)
df['lnum_shells'].value_counts()
df.drop('lnum_shells', axis=1, inplace=True)
df['lnum_access_files'].value_counts()
df.drop('lnum_access_files', axis=1, inplace=True)
df['is_guest_login'].value_counts()
df.drop('is_guest_login', axis=1, inplace=True)
df['serror_rate'].value_counts()
df.drop('serror_rate', axis=1, inplace=True)
df['srv_serror_rate'].value_counts()
df.drop('srv_serror_rate', axis=1, inplace=True)
df['rerror_rate'].value_counts()
df.drop('rerror_rate', axis=1, inplace=True)
df['srv_rerror_rate'].value_counts()
df.drop('srv_rerror_rate', axis=1, inplace=True)
df['same_srv_rate'].value_counts()
df.drop('same_srv_rate', axis=1, inplace=True)
df['diff_srv_rate'].value_counts()
df.drop('diff_srv_rate', axis=1, inplace=True)
df['srv_diff_host_rate'].value_counts()
df.drop('srv_diff_host_rate', axis=1, inplace=True)
df['dst_host_same_srv_rate'].value_counts()
df.drop('dst_host_same_srv_rate', axis=1, inplace=True)
df['dst_host_diff_srv_rate'].value_counts()
df.drop('dst_host_diff_srv_rate', axis=1, inplace=True)
df['dst_host_same_src_port_rate'].value_counts()
df.drop('dst_host_same_src_port_rate', axis=1, inplace=True)
df['dst_host_srv_diff_host_rate'].value_counts()
df.drop('dst_host_srv_diff_host_rate', axis=1, inplace=True)
df['dst_host_serror_rate'].value_counts()
df.drop('dst_host_serror_rate', axis=1, inplace=True)
df['dst_host_srv_serror_rate'].value_counts()
df.drop('dst_host_srv_serror_rate', axis=1, inplace=True)
df['dst_host_rerror_rate'].value_counts()
df.drop('dst_host_rerror_rate', axis=1, inplace=True)
df.drop('dst_host_rerror_rate', axis=1, inplace=True)
df['dst_host_srv_rerror_rate'].value_counts()
df.drop('dst_host_srv_rerror_rate', axis=1, inplace=True)
df['lsu_attempted'].value_counts()
df.drop('lsu_attempted', axis=1, inplace=True)
df['urgent'].value_counts()
df.drop('urgent', axis=1, inplace=True)



#df['lnum_outbound_cmds'].value_counts()
#df.drop('lnum_outbound_cmds', axis=1, inplace=True)
#df['is_host_login'].value_counts()
#df.drop('is_host_login', axis=1, inplace=True)


df['protocol_type'] = df['protocol_type'].astype('category')
df['service'] = df['service'].astype('category')
df['flag'] = df['flag'].astype('category')
cat_columns = df.select_dtypes(['category']).columns
df[cat_columns] = df[cat_columns].apply(lambda x: x.cat.codes)
```

Removal of duplicates

```
df.drop_duplicates(subset=None, keep='first', inplace=True)
```

```
df.shape
```

```
(115856, 14)
```

```
df['label'].value_counts()
```

```
normal              83691
neptune             27618
```

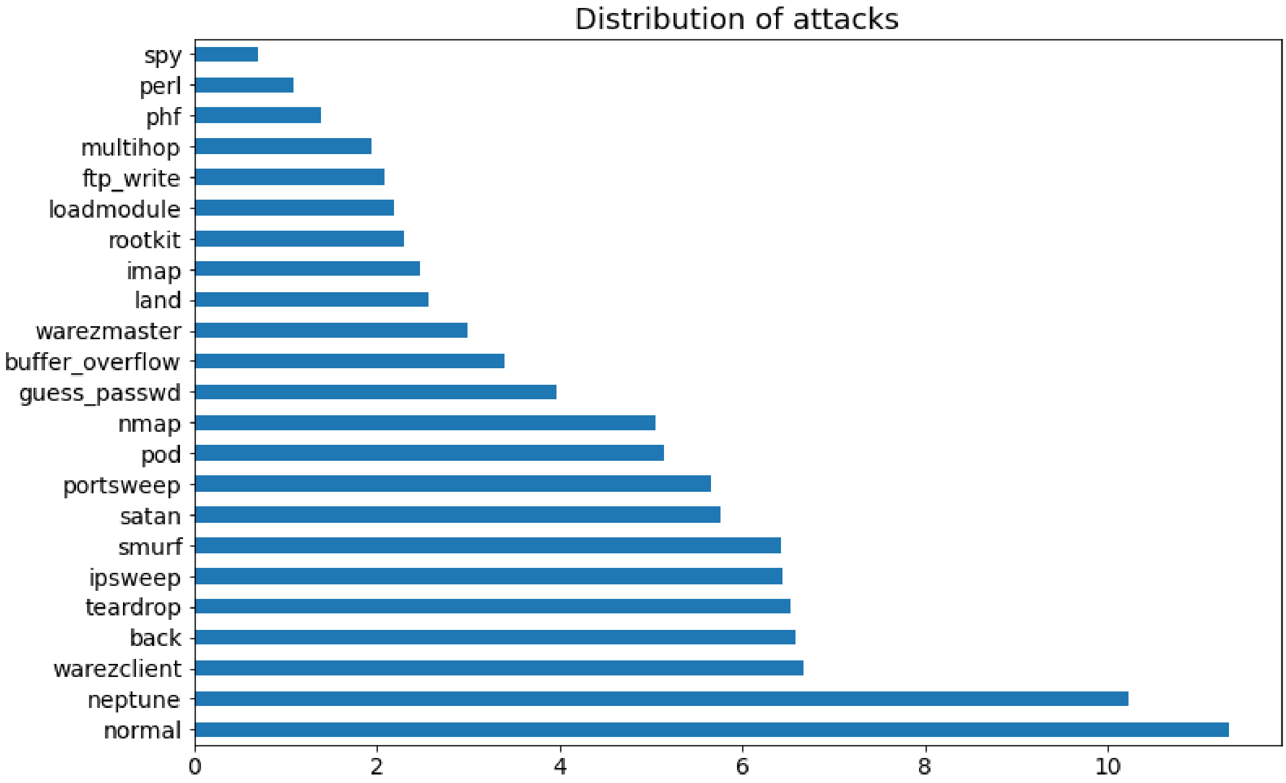
Log-scaled distribution of attacks

```
teardrop              686
```

```
plt.clf()
plt.figure(figsize=(12,8))
params = {'axes.titlesize':'18',
          'xtick.labelsize':'14',
          'ytick.labelsize':'14'}
matplotlib.rcParams.update(params)
plt.title('Distribution of attacks')
#df.plot(kind='barh')
df['label'].value_counts().apply(np.log).plot(kind='barh')

plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



KDD skewness and kurtosis

```
df.skew()
```

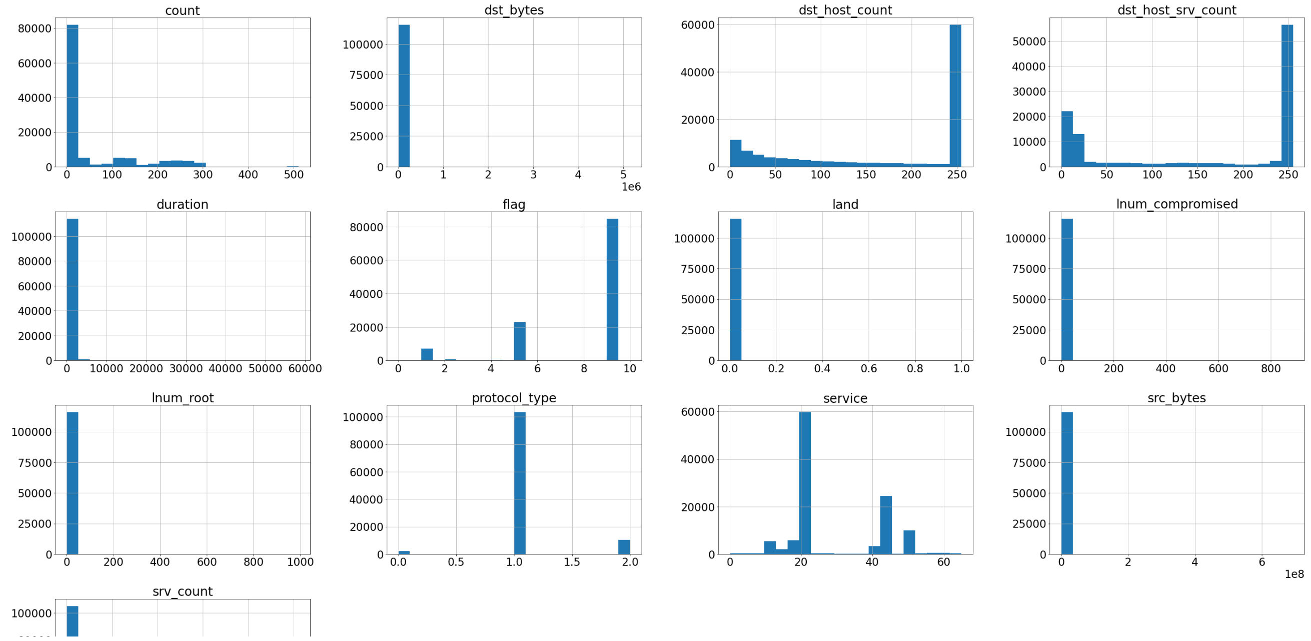
```
duration               14.985219
protocol_type           1.424176
service                 0.538835
flag                   -1.593586
src_bytes             338.612541
dst_bytes              66.342305
land                   90.954070
lnum_compromised      202.472917
lnum_root             201.989612
count                   1.856803
srv_count               9.890390
```

```
df.kurtosis()
```

```
duration                 306.909769
protocol_type              5.583877
service                   -1.085504
flag                       1.398477
src_bytes             115052.482121
dst_bytes               4779.294916
land                    8270.785682
lnum_compromised       44196.852559
lnum_root              44312.253465
count                      2.751708
srv_count                121.865683
dst_host_count            -1.490660
dst_host_srv_count        -1.719594
dtype: float64
```

Univariate histogramms

```
import matplotlib.pyplot as plt
import matplotlib
params = {'axes.titlesize':'28',
          'xtick.labelsize':'24',
          'ytick.labelsize':'24'}
matplotlib.rcParams.update(params)
df.hist(figsize=(50, 30), bins=20)
plt.show()
```



KDD standardization

```
df.shape
```

```
(115856, 14)
```

```
data = df.values
```

```
X = data[:, 0:13]
```

```
X
```

```
array([[0, 1, 22, ..., 8, 9, 9],
       [0, 1, 22, ..., 8, 19, 19],
       [0, 1, 22, ..., 8, 29, 29],
       ...,
       [0, 1, 22, ..., 18, 16, 255],
       [0, 1, 22, ..., 12, 26, 255],
       [0, 1, 22, ..., 35, 6, 255]], dtype=object)
```

```
from sklearn.preprocessing import StandardScaler
sScaler = StandardScaler()
rescaleX = sScaler.fit_transform(X)
```

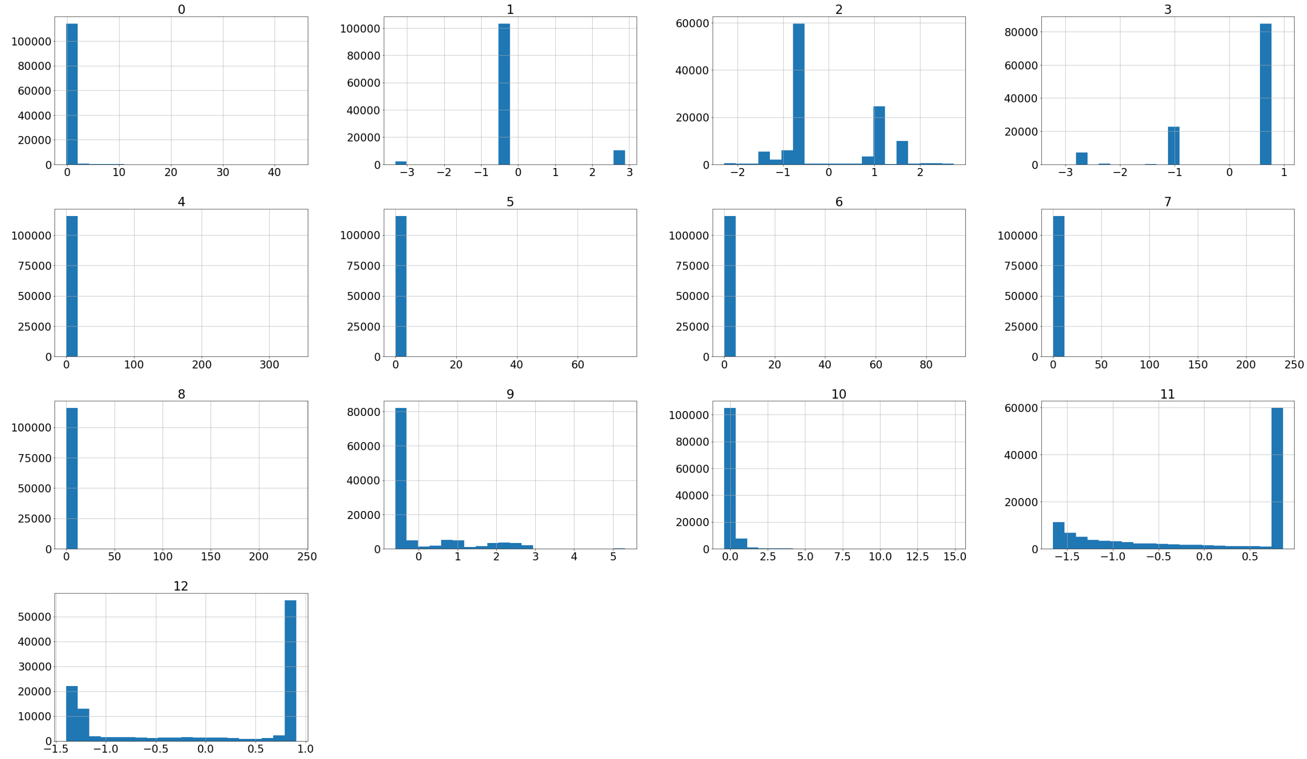
```
rescaleX
```

```
array([[-0.10990358, -0.2158199 , -0.58706037, ..., -0.17459515,
        -1.56558403, -1.31409047],
       [-0.10990358, -0.2158199 , -0.58706037, ..., -0.17459515,
        -1.46695844, -1.22375248],
       [-0.10990358, -0.2158199 , -0.58706037, ..., -0.17459515,
        -1.36833284, -1.1334145 ],
```

```
df_rescaled = pd.DataFrame(data=rescaleX)
```

```
       [-0.10990358, -0.2158199 , -0.58706037, ..., -0.05443954
```

```
df_rescaled.hist(figsize=(50, 30), bins=20)
plt.show()
```



KDD normalization¶

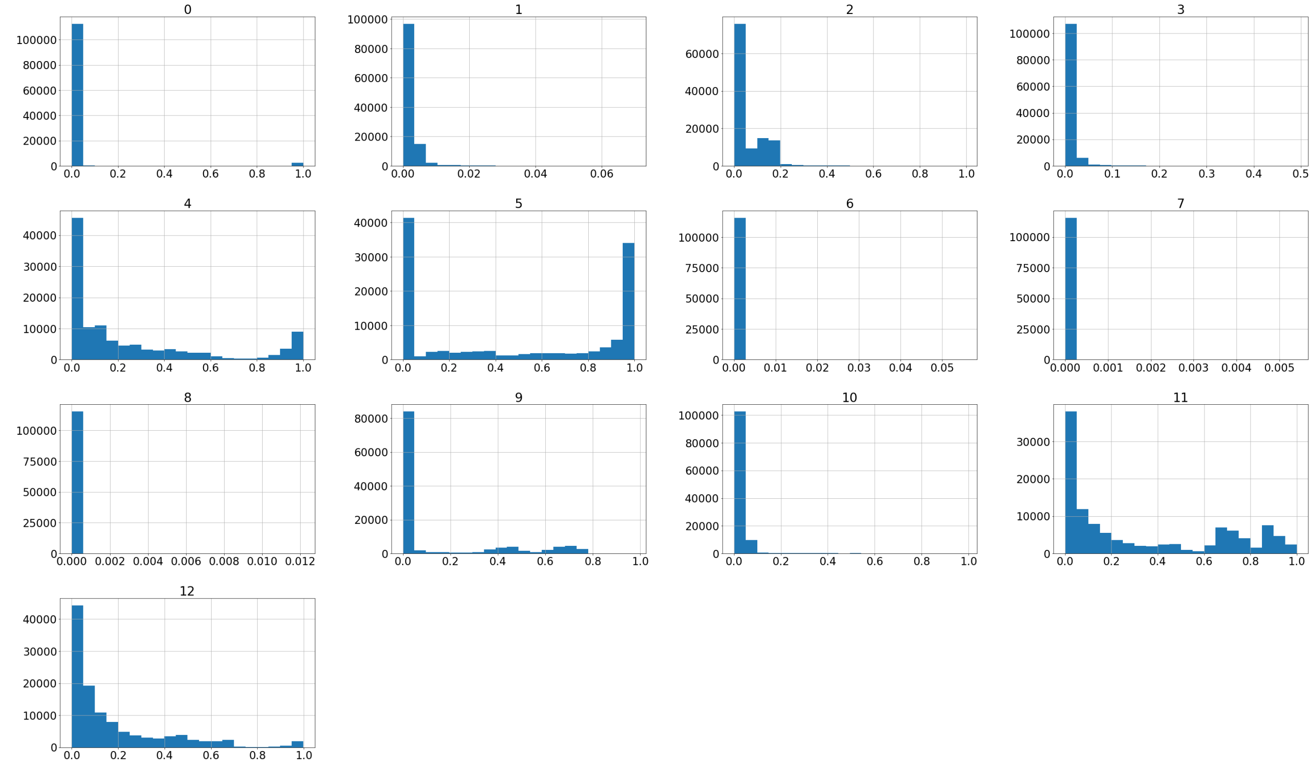
```
from sklearn.preprocessing import Normalizer
norm = Normalizer()
xNormalize = norm.fit_transform(X)
```

```
xNormalize
```

```
array([[0.00000000e+00, 1.83382493e-04, 4.03441484e-03, ...,
        1.46705994e-03, 1.65044243e-03, 1.65044243e-03],
       [0.00000000e+00, 1.84198300e-03, 4.05236260e-02, ...,
        1.47358640e-02, 3.49976770e-02, 3.49976770e-02],
       [0.00000000e+00, 7.36176259e-04, 1.61958777e-02, ...,
        5.88941007e-03, 2.13491115e-02, 2.13491115e-02],
       ...,
       [0.00000000e+00, 8.03889689e-04, 1.76855732e-02, ...,
        1.44700144e-02, 1.28622350e-02, 2.04991871e-01],
       [0.00000000e+00, 7.92770426e-04, 1.74409494e-02, ...,
        9.51324512e-03, 2.06120311e-02, 2.02156459e-01],
       [0.00000000e+00, 7.81439612e-04, 1.71916715e-02, ...,
        2.73503864e-02, 4.68863767e-03, 1.99267101e-01]])
```

```
df_Normalized = pd.DataFrame(data=xNormalize)
```

```
df_Normalized.hist(figsize=(50, 30), bins=20)
plt.show()
```



Encoding

```
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
```

```
df['label'] = df['label'].astype('category')
cat_columns = df.select_dtypes(['category']).columns
df[cat_columns] = df[cat_columns].apply(lambda x: x.cat.codes)
```

```
data = df.values
```

```
Y = data[:,13]
```

```
X = data[:,0:14]
```

```
Y
```

```
array([11, 11, 11, ..., 11, 11, 11])
```

```
X
```

```
array([[  0,   1,  22, ...,   9,   9,  11],
       [  0,   1,  22, ...,  19,  19,  11],
       [  0,   1,  22, ...,  29,  29,  11],
       ...,
       [  0,   1,  22, ...,  16, 255,  11],
       [  0,   1,  22, ...,  26, 255,  11],
       [  0,   1,  22, ...,   6, 255,  11]])
```

```
X = np.transpose(X)
X
```

```
array([[  0,   0,   0, ...,   0,   0,   0],
       [  1,   1,   1, ...,   1,   1,   1],
       [ 22,  22,  22, ...,  22,  22,  22],
       ...,
       [  9,  19,  29, ...,  16,  26,   6],
       [  9,  19,  29, ..., 255, 255, 255],
       [ 11,  11,  11, ...,  11,  11,  11]])
```

```
df.shape
```

```
(115856, 14)
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components=3)
```

```
pca.fit(X,Y)
```

```
    PCA(copy=True, iterated_power='auto', n_components=3, random_state=None,
```

```
pca.components_
```

```
array([[-3.52173642e-07,  2.79466075e-07,  1.77143379e-07, ...,
         1.21788939e-07,  2.48040098e-07,  1.40279534e-07],
       [ 2.34367963e-04,  2.05770794e-05,  5.71442093e-05, ...,
         5.04536665e-05,  5.04396671e-05,  5.18923964e-05],
       [-1.65785111e-05, -1.92922353e-05, -2.32806533e-05, ...,
        -7.07292499e-05, -7.11720485e-05, -7.25587675e-05]])
```

```
pca.explained_variance_
```

```
array([3.44607340e+16, 3.83057958e+13, 1.43144455e+10])
```

```
pca.transform(X)
```

```
array([[-4.96067932e+07, -1.78470363e+06,  4.12964297e+05],
       [-4.96089793e+07, -1.78750338e+06, -3.80981995e+04],
       [-4.96089514e+07, -1.78705313e+06, -3.69984891e+04],
       [-4.96089735e+07, -1.78736297e+06, -3.79208693e+04],
       [ 6.44973089e+08,  1.82711188e+03, -1.28239185e+00],
       [-4.96679542e+07,  2.14397646e+07, -4.08055851e+01],
       [-4.96089809e+07, -1.78752093e+06, -3.81544438e+04],
       [-4.96089810e+07, -1.78749197e+06, -3.80738178e+04],
       [-4.96089810e+07, -1.78749154e+06, -3.80663269e+04],
       [-4.96089231e+07, -1.78744445e+06, -3.87414537e+04],
       [-4.96089765e+07, -1.78740344e+06, -3.83026862e+04],
       [-4.96086917e+07, -1.78566113e+06, -3.12102674e+04],
       [-4.96089478e+07, -1.78466182e+06, -3.95372808e+04],
       [-4.96089556e+07, -1.78729335e+06, -3.78183750e+04]])
```