In [2]:

```
import pandas as pd
from sklearn import preprocessing
```

In [3]:

```
train_df = pd.read_csv("wat-all.csv")
```

In [4]:

```
train_df.head()
```

Out[4]:

	time	router	outport	inport	packet_address	packet_type	flit_id	flit_type	vnet	vc	src_ni
0	7	0	2	0	0x1dc0	0	0	3	2	8	0
1	7	1	1	0	0xecf40	0	0	3	2	8	1
2	7	0	2	0	0x1dc0	0	0	3	2	8	0
3	11	5	1	3	0xecf40	0	0	3	2	8	1
4	11	1	2	4	0x1dc0	0	0	3	2	8	0

In [5]:

```
train_X = train_df.drop(columns=['target'])
train_X = train_df.drop(columns=['time'])
train_X = train_df.drop(columns=['packet_address'])
```

```
In [6]:
```

```
train_X
```

```
Out[6]:

           time  router  outport  inport  packet_type  flit_id  flit_type  vnet  vc  src_ni  src_rou
     0        7       0        2       0            0        0          3     2   8       0
     1        7       1        1       0            0        0          3     2   8       1
     2        7       0        2       0            0        0          3     2   8       0
     3       11       5        1       3            0        0          3     2   8       1
     4       11       1        2       4            0        0          3     2   8       0
   ...      ...     ...      ...     ...          ...      ...        ...   ...  ...    ...
504035  3152966       8        3       2            1        2          1     4  16      27
504036  3152967       8        3       2            1        3          1     4  16      27
504037  3152967       9        4       2            1        4          2     4  16      27
504038  3152968       4        3       1            1        0          0     4  16      27
504039  3152969       4        3       1            1        1          1     4  16      27

504040 rows × 16 columns
```

In [7]:

```
x = train_X.values #returns a numpy array
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
train_X = pd.DataFrame(x_scaled)
train_X
```

Out[7]:

	0	1	2	3	4	5	6	7	8	9	
0	0.000000	0.000000	0.50	0.00	0.000000	0.00	1.000000	0.0	0.000000	0.000000	0.000
1	0.000000	0.066667	0.25	0.00	0.000000	0.00	1.000000	0.0	0.000000	0.032258	0.066
2	0.000000	0.000000	0.50	0.00	0.000000	0.00	1.000000	0.0	0.000000	0.000000	0.000
3	0.000001	0.333333	0.25	0.75	0.000000	0.00	1.000000	0.0	0.000000	0.032258	0.066
4	0.000001	0.066667	0.50	1.00	0.000000	0.00	1.000000	0.0	0.000000	0.000000	0.000
...	...	...	...	...	...	...	...	...	...	...	
504035	0.999999	0.533333	0.75	0.50	0.333333	0.50	0.333333	1.0	0.727273	0.870968	0.733
504036	0.999999	0.533333	0.75	0.50	0.333333	0.75	0.333333	1.0	0.727273	0.870968	0.733
504037	0.999999	0.600000	1.00	0.50	0.333333	1.00	0.666667	1.0	0.727273	0.870968	0.733
504038	1.000000	0.266667	0.75	0.25	0.333333	0.00	0.000000	1.0	0.727273	0.870968	0.733
504039	1.000000	0.266667	0.75	0.25	0.333333	0.25	0.333333	1.0	0.727273	0.870968	0.733

504040 rows × 16 columns

In [8]:

```
from keras.utils import to_categorical
```

```
Using TensorFlow backend.
```

In [9]:

```
train_Y = train_df['target']
```

In [10]:

```
train_Y
```

Out[10]:

```
0         1
1         0
2         0
3         0
4         0
         ..
504035    1
504036    1
504037    1
504038    1
504039    1
Name: target, Length: 504040, dtype: int64
```

In [11]:

```
from keras.models import Sequential
from keras.layers import Dense
```

In [12]:

```
model = Sequential()
```

In [13]:

```
n_cols = train_X.shape[1]
n_cols
```

Out[13]:

```
16
```

In [14]:

```
model.add(Dense(50, activation='relu', input_shape=(n_cols,)))
model.add(Dense(32, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

In [15]:

```
model.compile(optimizer='sgd', loss='mean_squared_error', metrics=['accuracy'])
```

In [16]:

```
from keras.callbacks import EarlyStopping
```

In [17]:

```
early_stopping_monitor = EarlyStopping(patience=5)
```

In [18]:

```
model.fit(train_X, train_Y, epochs=3, validation_split=0.2)
```

```
Train on 403232 samples, validate on 100808 samples
Epoch 1/3
403232/403232 [==============================] - 37s 91us/step - loss: 0.0252
- accuracy: 0.9780 - val_loss: 8.3938e-05 - val_accuracy: 1.0000
Epoch 2/3
403232/403232 [==============================] - 36s 88us/step - loss: 6.9663
e-05 - accuracy: 1.0000 - val_loss: 3.1461e-05 - val_accuracy: 1.0000
Epoch 3/3
403232/403232 [==============================] - 36s 90us/step - loss: 3.3759
e-05 - accuracy: 1.0000 - val_loss: 1.8446e-05 - val_accuracy: 1.0000
```

Out[18]:

```
<keras.callbacks.callbacks.History at 0x2099f7e22c8>
```

In [19]:

```
pred = model.predict(train_X)
```

In [20]:

```
for i in range(100):
    print("%s, %s" % (pred[i], train_Y[i]))
```

```
[0.9896243], 1
[0.00536102], 0
[0.00959565], 0
[0.00375869], 0
[0.00852191], 0
[0.9952206], 1
[0.00842151], 0
[0.00381809], 0
[0.9954352], 1
[0.00512045], 0
[0.00845471], 0
[0.9952852], 1
[0.994592], 1
[0.00934679], 0
[0.00293893], 0
[0.00206071], 0
[0.00179393], 0
[0.00155697], 0
[0.0027976], 0
[0.00538603], 0
[0.9973591], 1
[0.9976834], 1
[0.0018441], 0
[0.00332502], 0
[0.9976221], 1
[0.00302213], 0
[0.00157647], 0
[0.0010766], 0
[0.00135559], 0
[0.00268542], 0
[0.9974656], 1
[0.00472848], 0
[0.00318308], 0
[0.9976459], 1
[0.00278324], 0
[0.00177672], 0
[0.9978921], 1
[0.00152216], 0
[0.00243047], 0
[0.99764687], 1
[0.00129766], 0
[0.00216031], 0
[0.00092774], 0
[0.00195683], 0
[0.99719256], 1
[0.99666876], 1
[0.00463522], 0
[0.99763095], 1
[0.00489412], 0
[0.00268711], 0
[0.00308686], 0
[0.9978331], 1
[0.00268828], 0
[0.00236097], 0
[0.99758375], 1
[0.99583817], 1
[0.997042], 1
```

```
[0.0023073], 0
[0.00210311], 0
[0.00169372], 0
[0.00088654], 0
[0.9977908], 1
[0.00503273], 0
[0.00298104], 0
[0.9979488], 1
[0.00259602], 0
[0.99776804], 1
[0.00169087], 0
[0.00162031], 0
[0.0022527], 0
[0.9956599], 1
[0.99745303], 1
[0.99762696], 1
[0.00862185], 0
[0.00634011], 0
[0.9976592], 1
[0.9976192], 1
[0.0056463], 0
[0.99631315], 1
[0.99741995], 1
[0.00176722], 0
[0.00443075], 0
[0.9972958], 1
[0.00305697], 0
[0.00761852], 0
[0.00868523], 0
[0.00779837], 0
[0.0067108], 0
[0.9865936], 1
[0.00499666], 0
[0.99420184], 1
[0.008494], 0
[0.00557266], 0
[0.99428445], 1
[0.00672291], 0
[0.00447453], 0
[0.00378954], 0
[0.00324809], 0
[0.00563412], 0
[0.9962555], 1
```

In [21]:

```
from sklearn.decomposition import PCA
```

In [22]:

```
pca = PCA(n_components = 2)
```

In [23]:

```
pca.fit(train_X)
```

Out[23]:

```
PCA(copy=True, iterated_power='auto', n_components=2, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

In [24]:

```
principal_components = pca.transform(train_X)
principal_components
```

Out[24]:

```
array([[-0.70117465, -0.01660366],
       [-0.91498913,  0.92677719],
       [-0.69686615,  0.76900509],
       ...,
       [ 1.08271475, -0.7834302 ],
       [ 1.05687744, -0.76936859],
       [ 1.05343396, -0.7692734 ]])
```

In [25]:

```
principal_df = pd.DataFrame(data = principal_components , columns = ['pc 1', 'pc 2'])
principal_df
```

Out[25]:

	pc 1	pc 2
0	-0.701175	-0.016604
1	-0.914989	0.926777
2	-0.696866	0.769005
3	-1.061943	0.919297
4	-0.912609	0.767368
...	...	...
504035	1.021540	-0.777134
504036	1.026442	-0.777372
504037	1.082715	-0.783430
504038	1.056877	-0.769369
504039	1.053434	-0.769273

504040 rows × 2 columns

In [26]:

```
final_df = pd.concat([principal_df, train_df[['target']]], axis = 1)
final_df
```

Out[26]:

	pc 1	pc 2	target
0	-0.701175	-0.016604	1
1	-0.914989	0.926777	0
2	-0.696866	0.769005	0
3	-1.061943	0.919297	0
4	-0.912609	0.767368	0
...	...	...	...
504035	1.021540	-0.777134	1
504036	1.026442	-0.777372	1
504037	1.082715	-0.783430	1
504038	1.056877	-0.769369	1
504039	1.053434	-0.769273	1

504040 rows × 3 columns

In [27]:

```
final_df.corr()
```

Out[27]:

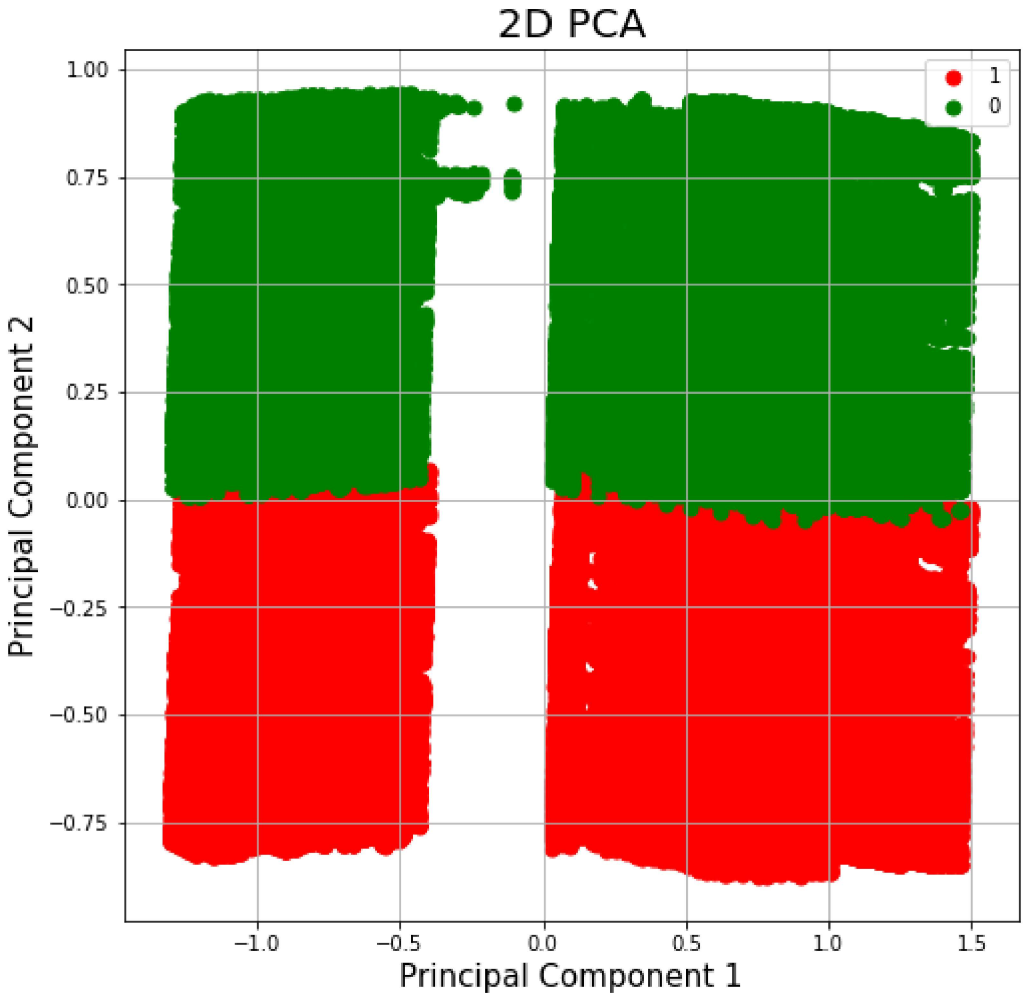
	pc 1	pc 2	target
pc 1	1.000000e+00	-7.640081e-17	-0.009797
pc 2	-7.640081e-17	1.000000e+00	-0.856061
target	-9.797429e-03	-8.560609e-01	1.000000

In [28]:

```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [29]:

```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2D PCA', fontsize = 20)
targets = [1, 0]
colors = ['r', 'g']
for target, color in zip(targets,colors):
    indicesToKeep = final_df['target'] == target
    ax.scatter(final_df.loc[indicesToKeep, 'pc 1']
               , final_df.loc[indicesToKeep, 'pc 2']
               , c = color
               , s = 50)
ax.legend(targets)
ax.grid()
```

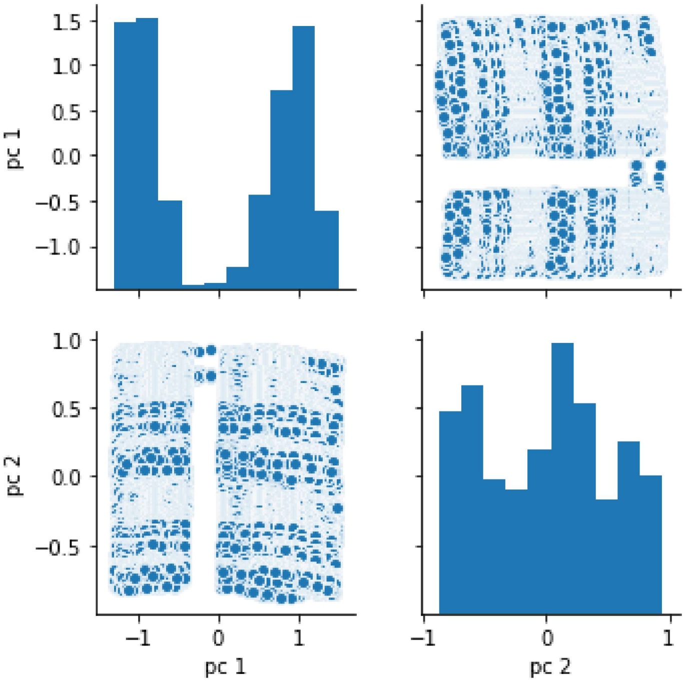


In [32]:

```
sns.pairplot(final_df.loc[:,final_df.dtypes == 'float64'])
```

Out[32]:

```
<seaborn.axisgrid.PairGrid at 0x209a1ed6a08>
```



In [33]:

```
pca.explained_variance_ratio_
```

Out[33]:

```
array([0.43219671, 0.12277561])
```