Untitled

October 8, 2018

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns
%matplotlib inline

np.random.seed(2)

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import itertools

from keras.utils.np_utils import to_categorical # convert to one-hot-encoding
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv1D, MaxPool1D
from keras.optimizers import RMSprop
from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array,
from keras.callbacks import ReduceLROnPlateau


sns.set(style='white', context='notebook', palette='deep')
```

```
C:\Users\Varad\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of t
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

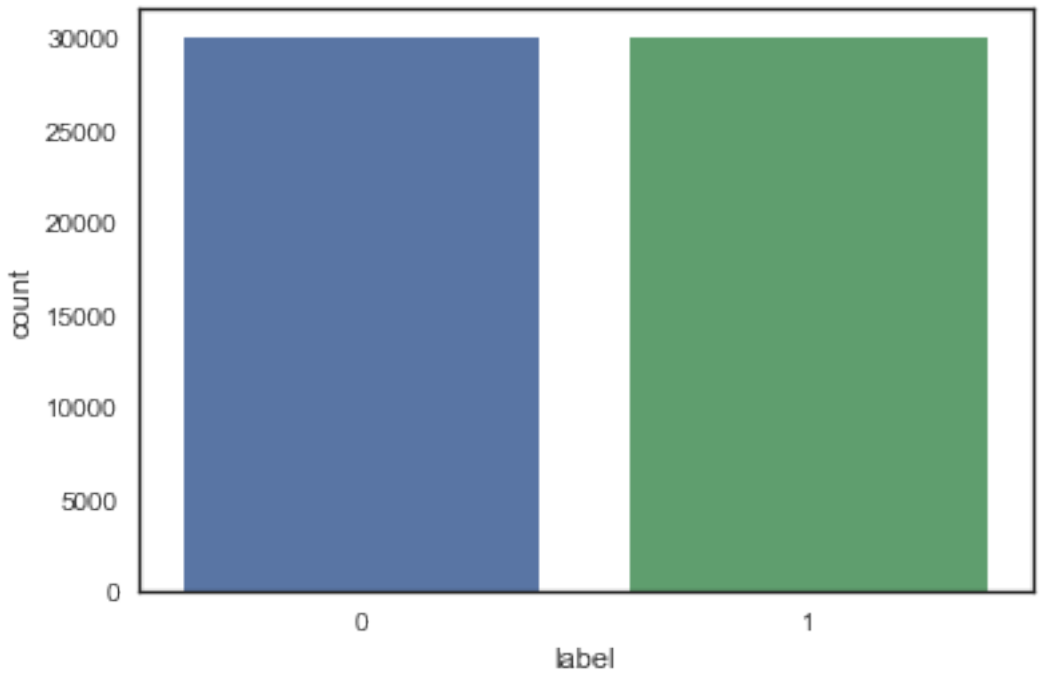
In [2]:
```
# Load the data
data_taken = pd.read_csv("Datasets/dataset_Copy.csv")
print(data_taken.head())
```

```
   label      cl1      cl2      cl3      cl4      cl5      cl6      cl7  \
0      0 -0.15256 -0.18260 -0.21046 -0.17843 -0.15295 -0.18675 -0.20970
1      0  0.81460  0.91668  0.99092  0.99092  0.99092  0.99092  0.99136
2      0  0.49070  0.45079  0.40279  0.29840  0.23562  0.23562  0.23688
3      0  0.21846  0.16398  0.11329  0.11329  0.11329  0.11329  0.11485
4      0  0.38019  0.34004  0.30267  0.30267  0.30267  0.30267  0.30267
```

```
       cl8      cl9    ...       cl491    cl492    cl493    cl494    cl495  \
0 -0.17432 -0.15406    ...     0.42617  0.42749  0.42749  0.47871  0.59030
1  1.01510  1.06120    ...    -3.34280 -4.15830 -4.57750 -5.00520 -5.29190
2  0.28875  0.32065    ...    -0.37281 -0.25422 -0.11002 -0.16828 -0.27202
3  0.17900  0.21661    ...    -0.47329  0.45015  0.93955  0.77506  0.44745
4  0.30267  0.30267    ...     0.29959  0.24987  0.15163  0.14762  0.14762

      cl496     cl497     cl498     cl499     cl500
0  0.659950  0.713680  0.743660  0.773480  0.849310
1 -4.891200 -3.965800 -3.050200 -2.104600 -1.283300
2 -0.204740 -0.105620 -0.104480 -0.104480 -0.073866
3  0.361440  0.323940  0.370540  0.428610  0.428790
4  0.095191 -0.001893 -0.046843 -0.083106 -0.119630

[5 rows x 501 columns]
```

In [3]:
```
y = data_taken.label
X = data_taken.drop('label', axis=1)
g = sns.countplot(y)
y.value_counts()
```

Out[3]:
```
1    30000
0    30000
Name: label, dtype: int64
```



```
In [4]: y

Out[4]: 0        0
        1        0
        2        0
        3        0
        4        0
        5        0
        6        0
        7        0
        8        0
        9        0
        10       0
        11       0
        12       0
        13       0
        14       0
        15       0
        16       0
        17       0
        18       0
        19       0
        20       0
        21       0
        22       0
        23       0
        24       0
        25       0
        26       0
        27       0
        28       0
        29       0
                ..
        59970    1
        59971    1
        59972    1
        59973    1
        59974    1
        59975    1
        59976    1
        59977    1
        59978    1
        59979    1
        59980    1
        59981    1
        59982    1
        59983    1
        59984    1
```

```
59985    1
59986    1
59987    1
59988    1
59989    1
59990    1
59991    1
59992    1
59993    1
59994    1
59995    1
59996    1
59997    1
59998    1
59999    1
Name: label, Length: 60000, dtype: int64
```

In [5]: X

Out[5]:

	cl1	cl2	cl3	cl4	cl5	cl6	cl7	\
0	-0.152560	-0.182600	-0.210460	-0.178430	-0.152950	-0.186750	-0.209700	
1	0.814600	0.916680	0.990920	0.990920	0.990920	0.990920	0.991360	
2	0.490700	0.450790	0.402790	0.298400	0.235620	0.235620	0.236880	
3	0.218460	0.163980	0.113290	0.113290	0.113290	0.113290	0.114850	
4	0.380190	0.340040	0.302670	0.302670	0.302670	0.302670	0.302670	
5	-0.130700	-0.093764	-0.072983	-0.072983	-0.073755	-0.141040	-0.187930	
6	0.358290	0.493760	0.599210	0.632980	0.644790	0.616930	0.599980	
7	0.130370	0.115490	0.091294	0.066703	0.054090	0.054090	0.054090	
8	1.670300	1.644200	1.619900	1.619900	1.618500	1.491000	1.357800	
9	0.066110	-0.046800	-0.131120	-0.099990	-0.074840	-0.074840	-0.075685	
10	-0.407130	-0.463190	-0.515360	-0.515360	-0.513920	-0.387740	-0.299970	
11	0.471290	0.435220	0.401780	0.440230	0.471290	0.471290	0.472320	
12	0.260590	0.260590	0.260410	0.199500	0.150290	0.150290	0.149480	
13	-0.038984	-0.076784	-0.111890	-0.084507	-0.033775	0.057609	0.106960	
14	0.092442	0.134450	0.173540	0.173540	0.172460	0.077914	0.012151	
15	-0.233300	-0.216180	-0.182360	-0.121860	-0.086213	-0.156310	-0.242090	
16	0.623330	0.657350	0.689010	0.689010	0.687830	0.570360	0.406090	
17	-0.400630	-0.274470	-0.183090	-0.303420	-0.400140	-0.357800	-0.327990	
18	1.114200	1.182700	1.205300	0.832970	0.117530	-1.314500	-2.089600	
19	-0.108560	-0.126170	-0.154820	-0.183940	-0.198870	-0.198870	-0.198490	
20	-0.011685	0.043346	0.094372	0.035707	-0.010974	0.050949	0.093491	
21	-0.290050	-0.454180	-0.606630	-0.519150	-0.448470	-0.448470	-0.447300	
22	-0.469690	-0.407830	-0.320680	-0.181840	-0.022908	0.130430	0.269420	
23	0.261480	0.223520	0.154200	0.056349	0.002307	0.122660	0.278920	
24	0.532240	0.437720	0.318350	0.225570	0.183250	0.244400	0.286760	
25	0.461840	0.408910	0.359650	0.359650	0.360340	0.419900	0.461840	
26	0.335780	0.335780	0.335780	0.335780	0.334500	0.223200	0.147040	
27	-0.253290	-0.231040	-0.194840	-0.158060	-0.140330	-0.240100	-0.309200	

28	0.847530	0.919380	0.959150	0.734990	0.502130	0.425030	0.401750
29	1.851800	1.743900	1.635900	1.513900	1.425300	1.393400	1.349200
...	...	...	...	...	...	...	...
59970	1.197800	1.219300	1.233600	1.269400	1.255100	1.176400	1.147700
59971	0.063810	0.050872	-0.020287	-0.052632	-0.065570	-0.078508	-0.117320
59972	0.167240	0.194920	0.236440	0.243350	0.250270	0.312550	0.326390
59973	-0.052431	-0.038648	-0.024865	-0.052431	-0.052431	-0.052431	-0.052431
59974	-4.888600	-5.174500	-4.693000	-3.737700	-2.789900	-2.105300	-1.691600
59975	0.035906	0.028158	0.004913	-0.018333	-0.041578	-0.033830	-0.026081
59976	0.044358	0.057201	0.025094	0.012252	-0.000591	0.005830	0.025094
59977	1.117000	1.177800	1.238500	1.276500	1.360000	1.428300	1.511800
59978	0.003536	0.029160	0.074003	0.093221	0.093221	0.150880	0.157280
59979	-0.299810	-0.195370	-0.083959	0.013522	0.117970	0.152780	0.166710
59980	0.181740	0.174600	0.181740	0.124630	0.160320	0.110350	0.103210
59981	0.026981	0.039916	0.072252	0.046383	0.046383	0.065784	0.065784
59982	0.476210	0.525580	0.560840	0.610210	0.652530	0.673690	0.716000
59983	-0.060835	-0.039313	-0.060835	-0.039313	-0.024965	-0.017791	-0.024965
59984	-0.115960	0.009565	-0.059474	-0.429770	-1.157800	-1.961200	-2.764500
59985	1.206400	1.227300	1.282900	1.310800	1.366500	1.352500	1.352500
59986	0.121390	0.121390	0.128560	0.121390	0.121390	0.135720	0.092743
59987	-0.105420	-0.005330	0.000926	-0.230530	-0.874840	-1.675500	-2.476200
59988	0.926690	1.014900	1.089500	1.137000	1.191200	1.218300	1.286200
59989	0.048005	0.075948	0.068962	0.068962	0.027048	0.041020	0.020063
59990	-0.036539	-0.042922	-0.036539	-0.068456	-0.087606	-0.074839	-0.049306
59991	0.554540	0.575910	0.590150	0.640010	0.668490	0.689860	0.711220
59992	0.141590	0.148760	0.134410	0.120060	0.141590	0.148760	0.120060
59993	0.123920	0.086204	0.042196	0.029623	0.017050	-0.008097	-0.026957
59994	-0.586800	-0.477570	-0.361500	-0.300050	-0.224950	-0.136190	-0.040610
59995	1.240900	1.157600	1.053500	0.935580	0.831500	0.748240	0.623340
59996	-0.055874	-0.042006	-0.083611	-0.083611	-0.062808	-0.083611	-0.076677
59997	-0.140850	-0.121840	-0.134520	-0.166200	-0.166200	-0.223220	-0.229560
59998	0.451730	0.492850	0.506550	0.513410	0.554530	0.609350	0.616210
59999	0.064428	0.050262	0.036096	0.064428	0.029013	0.029013	0.029013

	cl8	cl9	cl10	...	cl491	cl492	cl493	\
0	-0.174320	-0.154060	-0.190810	...	0.426170	0.427490	0.427490	
1	1.015100	1.061200	1.174200	...	-3.342800	-4.158300	-4.577500	
2	0.288750	0.320650	0.320650	...	-0.372810	-0.254220	-0.110020	
3	0.179000	0.216610	0.164340	...	-0.473290	0.450150	0.939550	
4	0.302670	0.302670	0.302670	...	0.299590	0.249870	0.151630	
5	-0.163270	-0.121670	-0.047385	...	-0.175680	-0.220610	-0.245350	
6	0.730010	0.919740	1.167000	...	-0.506420	-0.599840	-0.781570	
7	0.054090	0.053421	0.034465	...	-0.400560	-0.377610	-0.329280	
8	1.313300	1.245400	1.013400	...	-0.698110	-0.689650	-0.656910	
9	-0.115680	-0.168230	-0.223080	...	-0.162770	-0.169010	-0.186870	
10	-0.351160	-0.418420	-0.488620	...	0.125460	0.134060	0.134060	
11	0.514800	0.539700	0.505090	...	-0.225000	-0.201340	-0.109300	
12	0.115840	0.095145	0.095145	...	-0.454160	-0.432530	-0.360440	

13	0.106960	0.105880	0.072936	...	-0.398050	-0.403850	-0.403850
14	0.050507	0.100900	0.153500	...	-0.228740	-0.211930	-0.159320
15	-0.306820	-0.331840	-0.331840	...	-0.037545	-0.203990	-0.516180
16	0.235550	0.164680	0.197320	...	0.497180	0.491950	0.491950
17	-0.327990	-0.327510	-0.311930	...	-0.251000	-0.272990	-0.291040
18	-1.321200	-0.404710	0.373480	...	-0.244040	-0.140650	0.092540
19	-0.179200	-0.146650	-0.088531	...	4.547300	3.776900	1.545100
20	0.043247	-0.022769	-0.091673	...	-0.122150	-0.144170	-0.213080
21	-0.398980	-0.371310	-0.421500	...	0.343670	0.326560	0.274790
22	0.387370	0.485450	0.579460	...	-0.105080	-0.096219	-0.048621
23	0.415040	0.469810	0.469810	...	-0.521820	-0.519750	-0.519750
24	0.253680	0.210610	0.171610	...	-0.269220	-0.256090	-0.210520
25	0.461840	0.467130	0.596620	...	-0.049100	0.020494	0.149990
26	0.246070	0.348230	0.410630	...	0.240300	0.207790	0.147300
27	-0.255240	-0.184340	-0.110330	...	-6.249200	-4.921400	-2.354900
28	0.538730	0.622980	0.622980	...	-0.064740	-0.050685	-0.050685
29	1.182800	0.969170	0.725030	...	-0.189230	-0.186440	-0.186440
...	...	...	...	...	...	...	...
59970	1.047500	1.004600	0.933030	...	-0.154820	-0.212070	-0.212070
59971	-0.136730	-0.136730	-0.162600	...	-0.188480	-0.046163	0.070279
59972	0.360980	0.381740	0.444020	...	-0.109530	-0.116450	-0.095695
59973	-0.038648	-0.059323	-0.031757	...	-0.355660	-0.410800	-0.369450
59974	-1.315500	-1.022100	-0.834040	...	0.008470	0.023515	0.023515
59975	-0.026081	-0.026081	-0.049327	...	0.074649	0.059152	0.051403
59976	-0.000591	-0.000591	-0.000591	...	0.545220	0.538800	0.564480
59977	1.519400	1.618100	1.640800	...	0.016276	0.039051	0.039051
59978	0.170090	0.118840	0.138060	...	-2.994500	-2.180900	-1.367400
59979	0.264190	0.285080	0.292040	...	0.605370	0.487000	0.431300
59980	0.117490	0.096069	0.110350	...	0.053232	0.003256	-0.011023
59981	0.059317	0.052850	0.039916	...	0.402080	0.434410	0.473220
59982	0.751260	0.835890	0.920520	...	-0.116200	-0.123250	-0.116200
59983	-0.032139	-0.053661	-0.039313	...	-0.089531	-0.068009	-0.082357
59984	-3.567900	-4.283400	-4.484200	...	0.762720	0.781550	0.806650
59985	1.359500	1.296900	1.262100	...	0.023368	0.037286	0.023368
59986	0.121390	0.099906	0.107070	...	-0.107810	-0.107810	-0.150790
59987	-3.276900	-4.077600	-4.553000	...	0.595190	0.626470	0.632730
59988	1.313300	1.354000	1.394700	...	-0.009347	0.004219	-0.029695
59989	0.034034	0.054991	0.061976	...	-0.042808	-0.042808	-0.070750
59990	-0.055689	-0.081223	-0.068456	...	0.257100	0.359230	0.448600
59991	0.753950	0.775320	0.810930	...	0.412110	0.326650	0.269670
59992	0.127240	0.134410	0.105710	...	0.026791	0.033966	0.062665
59993	0.004476	0.010763	0.010763	...	-3.239500	-4.044200	-4.528300
59994	0.048147	0.171040	0.225660	...	1.167800	1.202000	1.242900
59995	0.540070	0.435990	0.352730	...	0.075175	0.082114	0.089052
59996	-0.083611	-0.062808	-0.083611	...	-0.104410	-0.083611	-0.062808
59997	-0.210550	-0.128180	-0.007794	...	0.340700	0.391380	0.423070
59998	0.698450	0.753280	0.780690	...	0.198150	0.150170	0.095345
59999	0.007763	0.021929	0.021929	...	-0.006403	0.029013	0.036096

	cl494	cl495	cl496	cl497	cl498	cl499	cl500
0	0.478710	0.590300	0.659950	0.713680	0.743660	0.773480	0.849310
1	-5.005200	-5.291900	-4.891200	-3.965800	-3.050200	-2.104600	-1.283300
2	-0.168280	-0.272020	-0.204740	-0.105620	-0.104480	-0.104480	-0.073866
3	0.775060	0.447450	0.361440	0.323940	0.370540	0.428610	0.428790
4	0.147620	0.147620	0.095191	-0.001893	-0.046843	-0.083106	-0.119630
5	-0.246120	-0.246120	-0.246120	-0.246120	-0.297620	-0.361360	-0.340770
6	-0.717250	-0.600610	-0.597780	-0.597780	-0.576460	-0.550060	-0.573020
7	-0.337760	-0.362520	-0.386370	-0.403420	-0.390980	-0.366390	-0.342190
8	-0.630640	-0.604800	-0.573480	-0.550910	-0.550680	-0.550680	-0.486930
9	-0.166450	-0.132050	-0.154360	-0.187220	-0.187600	-0.187600	-0.174010
10	0.093455	0.027429	0.025827	0.025827	0.122400	0.241930	0.137960
11	0.410270	1.301600	1.754700	2.000900	1.068700	-0.843000	-2.808200
12	-0.281260	-0.151840	0.464920	1.324900	1.651100	1.804400	0.956870
13	-0.448770	-0.541600	-0.593170	-0.622490	-0.557660	-0.477070	-0.476820
14	-0.108930	-0.070572	-0.069762	-0.069762	-0.042942	0.009348	0.045973
15	-0.528930	-0.528930	-0.448040	-0.333160	-0.331840	-0.331840	-0.364930
16	0.491950	0.491950	0.491950	0.491950	0.491950	0.491950	0.523610
17	-0.291670	-0.291670	-0.336390	-0.399900	-0.400630	-0.400630	-0.387550
18	0.102060	0.102060	0.026536	-0.080727	-0.028182	0.096142	0.310850
19	-0.765640	-2.980900	-4.222600	-4.888900	-4.589100	-3.889400	-2.880800
20	-0.279090	-0.329340	-0.330400	-0.330400	-0.283010	-0.224340	-0.224160
21	0.200700	0.114410	0.058940	0.027110	0.062149	0.105890	0.077503
22	0.170520	0.461260	0.387240	0.196560	-0.026377	-0.246130	-0.394500
23	-0.480670	-0.417130	-0.415580	-0.415580	-0.438820	-0.467580	-0.407600
24	-0.156160	-0.101620	-0.079414	-0.063884	-0.046017	-0.028113	-0.012767
25	0.116940	0.054601	0.080258	0.148670	0.264610	0.359380	0.359650
26	0.180650	0.238890	0.240300	0.240300	0.197710	0.144990	0.190850
27	-2.571200	-3.093200	-2.898500	-2.360800	-1.472800	-0.509380	0.045648
28	-0.112240	-0.257990	-0.398390	-0.498780	-0.469140	-0.391560	-0.203620
29	-0.160070	-0.117200	-0.173850	-0.255790	-0.225370	-0.186560	-0.186440
...	...	...	...	...	...	...	...
59970	-0.183450	-0.097563	-0.119030	-0.068935	-0.054622	-0.068935	-0.018837
59971	0.141440	0.193190	0.251410	0.264350	0.296690	0.322570	0.348450
59972	-0.081856	-0.109530	-0.123370	-0.095695	-0.102610	-0.109530	-0.130290
59973	-0.252290	-0.197160	-0.507280	-1.210200	-2.092400	-2.974500	-3.856600
59974	0.031037	0.053605	0.015993	-0.006575	-0.006575	-0.006575	-0.006575
59975	0.020409	-0.002836	-0.002836	0.028158	0.051403	0.082397	0.059152
59976	0.590170	0.686490	0.705750	0.718590	0.782810	0.821340	0.885550
59977	0.039051	0.039051	0.016276	0.077008	0.077008	0.077008	0.001093
59978	-0.899720	-0.630660	-0.476920	-0.316770	-0.227080	-0.156610	-0.073336
59979	0.305960	0.236340	0.166710	0.124930	0.076188	0.083151	0.013522
59980	-0.053861	-0.118120	-0.082419	-0.118120	-0.153810	-0.168090	-0.168090
59981	0.499090	0.492620	0.524960	0.531420	0.563760	0.563760	0.589630
59982	-0.123250	-0.151460	-0.109140	-0.151460	-0.144410	-0.123250	-0.116200
59983	-0.118230	-0.089531	-0.075183	-0.103880	-0.139750	-0.161270	-0.182790
59984	0.844310	0.907070	0.976110	1.001200	1.057700	1.101600	1.145600

```
59985  0.030327 -0.011427  0.030327  0.030327  0.016409  0.002491 -0.011427
59986 -0.165120 -0.157950 -0.157950 -0.136460 -0.186600 -0.236740 -0.258230
59987  0.670260  0.682770  0.720300  0.745330  0.795370  0.851670  0.857920
59988 -0.036478 -0.009347 -0.016130 -0.016130  0.011002  0.004219 -0.016130
59989 -0.112660 -0.119650 -0.070750 -0.063765 -0.035822 -0.056779 -0.056779
59990  0.454990  0.512440  0.544350  0.576270  0.595420  0.608190  0.633720
59991  0.191330  0.134360  0.091629  0.070263  0.041776  0.020411  0.006167
59992  0.134410  0.112890  0.127240  0.141590  0.098539  0.055490  0.026791
59993 -4.213900 -3.415500 -2.617100 -1.818700 -1.095700 -0.781360 -0.573900
59994  1.304400  1.304400  1.331700  1.338500  1.331700  1.297600  1.277100
59995  0.109870  0.068236  0.116810  0.082114  0.075175  0.095991  0.061297
59996 -0.069742 -0.062808 -0.055874 -0.062808 -0.111350 -0.125210 -0.083611
59997  0.461080  0.473750  0.480090  0.499100  0.505440  0.524440  0.549790
59998  0.047371  0.026811 -0.007457 -0.062284 -0.055431 -0.096552 -0.089698
59999  0.014846 -0.041819 -0.063068 -0.091401 -0.133900 -0.148070 -0.148070

[60000 rows x 500 columns]
```

In [6]:
```
# Check the data
X.isnull().any().describe
```

Out[6]:
```
<bound method NDFrame.describe of cl1      False
cl2      False
cl3      False
cl4      False
cl5      False
cl6      False
cl7      False
cl8      False
cl9      False
cl10     False
cl11     False
cl12     False
cl13     False
cl14     False
cl15     False
cl16     False
cl17     False
cl18     False
cl19     False
cl20     False
cl21     False
cl22     False
cl23     False
cl24     False
cl25     False
cl26     False
cl27     False
```

```
cl28     False
cl29     False
cl30     False
         ...
cl471    False
cl472    False
cl473    False
cl474    False
cl475    False
cl476    False
cl477    False
cl478    False
cl479    False
cl480    False
cl481    False
cl482    False
cl483    False
cl484    False
cl485    False
cl486    False
cl487    False
cl488    False
cl489    False
cl490    False
cl491    False
cl492    False
cl493    False
cl494    False
cl495    False
cl496    False
cl497    False
cl498    False
cl499    False
cl500    False
Length: 500, dtype: bool>
```

```
In [7]: X_train, X_test, Y_train, Y_test = train_test_split(X, y,test_size=0.5)
        print("\nX_train:\n")
        print(X_train.head())
        print(X_train.shape)
        print("\nX_test:\n")
        print(X_test.head())
        print(X_test.shape)
```

```
X_train:

          cl1      cl2       cl3       cl4       cl5       cl6      cl7  \
```

```
9996  -0.283840 -0.31571 -0.345320 -0.322230 -0.279620 -0.213700 -0.15523
57432  0.849660  0.91628  0.997720  1.005100  1.042100  1.034700  1.03470
6744   0.145620  0.17939  0.195160  0.167760  0.146360  0.216950  0.30302
55126  0.071549  0.13258  0.132580  0.181400  0.218010  0.291250  0.34007
41565  0.139720  0.00014  0.035035  0.035035 -0.034756  0.035035 -0.20923

           cl8       cl9      cl10    ...         cl491     cl492     cl493  \
9996  -0.12799 -0.085125  0.072314    ...    -0.224740 -0.243250 -0.282240
57432  1.04210  1.027300  1.056900    ...     0.227820  0.235220  0.235220
6744   0.35914  0.399790  0.425440    ...    -0.549080 -0.549080 -0.549080
55126  0.44992  0.449920  0.474330    ...    -0.001684  0.071549 -0.001684
41565  0.10483  0.000140 -0.034756    ...     0.523570  0.558470  0.523570

          cl494     cl495     cl496     cl497     cl498     cl499     cl500
9996  -0.237660 -0.162580 -0.211270 -0.283010 -0.238400 -0.162840 -0.120600
57432  0.220410  0.190800  0.213010  0.176000  0.213010  0.190800  0.205610
6744  -0.562690 -0.594890 -0.625920 -0.648100 -0.648330 -0.648330 -0.630460
55126 -0.001684 -0.001684 -0.026095  0.010521 -0.062712 -0.087123 -0.074917
41565  0.523570  0.523570  0.488680  0.418890  0.383990  0.314200  0.349090

[5 rows x 500 columns]
(30000, 500)

X_test:

           cl1       cl2       cl3       cl4       cl5       cl6       cl7  \
10327 -0.45956 -0.415210 -0.375500 -0.348750 -0.318130 -0.249250 -0.212870
11901  0.13053  0.352850  0.637110  0.773920  0.826120  0.476590  0.231950
13305  0.39119  0.408580  0.440310  0.480380  0.512120  0.524030  0.536780
27477  0.57629  0.719900  0.853530  0.853530  0.853530  0.853530  0.855580
55627 -0.16571 -0.079071 -0.050192 -0.035752 -0.079071 -0.079071 -0.079071

           cl8       cl9      cl10   ...         cl491     cl492     cl493  \
10327 -0.24642 -0.26705 -0.267050   ...      0.935900  0.954430  0.966530
11901  0.29292  0.32007  0.066769   ...      0.230470  0.332570  0.522540
13305  0.56059  0.58649  0.608750   ...      1.427600  1.415700  1.385000
27477  0.94014  0.99035  0.946440   ...     -0.047481 -0.047481 -0.047481
55627 -0.12239 -0.09351 -0.021313   ...      0.253040  0.224160  0.166400

          cl494     cl495     cl496     cl497     cl498    cl499    cl500
10327  1.014700  1.097600  1.168500  1.238900  1.307800  1.37940  1.46340
11901  0.448110  0.253500  0.066037 -0.068014 -0.003261  0.12562  0.25244
13305  1.370200  1.359100  1.348900  1.335500  1.264800  1.14730  1.04950
27477 -0.047481 -0.047481 -0.019034  0.021364 -0.040008 -0.11655 -0.11679
55627  0.151960  0.209720  0.224160  0.166400  0.195280  0.10864  0.15196

[5 rows x 500 columns]
(30000, 500)
```

```
In [8]: # Normalize the data
        X_train = X_train / 255.0
        X_test = X_test/255.0

In [9]: X_test.shape

Out[9]: (30000, 500)

In [10]: # Reshape image in 3 dimensions (height = 28px, width = 28px , canal = 1)
         X_train = X_train.values.reshape(-1,500,1)
         X_test = X_test.values.reshape(-1,500,1)

In [11]: X_train.shape

Out[11]: (30000, 500, 1)

In [12]: # Set the random seed
         random_seed = 2

In [13]: # Split the train and the validation set for the fitting
         #X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size = 0.1,

In [14]: X_train.shape

Out[14]: (30000, 500, 1)

In [15]: # Encode labels to one hot vectors (ex : 2 -> [0,0,1,0,0,0,0,0,0,0])
         Y_train = to_categorical(Y_train, num_classes = 2)

In [16]: # Encode labels to one hot vectors (ex : 2 -> [0,0,1,0,0,0,0,0,0,0])
         Y_test = to_categorical(Y_test, num_classes = 2)

In [17]: from keras.layers import LeakyReLU

         # Set the CNN model
         # my CNN architechture is In -> [[Conv2D->relu]*2 -> MaxPool2D -> Dropout]*2 -> Flatt

         model = Sequential()

         model.add(Conv1D(filters = 5, kernel_size = 5, strides = 1, input_shape = (500,1)))
         model.add(LeakyReLU(alpha=0.01))
         model.add(MaxPool1D(pool_size=2, strides=2))

         model.add(Conv1D(filters = 5, kernel_size = 5, strides = 1))
         model.add(LeakyReLU(alpha=0.01))
         model.add(MaxPool1D(pool_size=2, strides=2))

         model.add(Conv1D(filters = 10, kernel_size = 3, strides = 1))
```

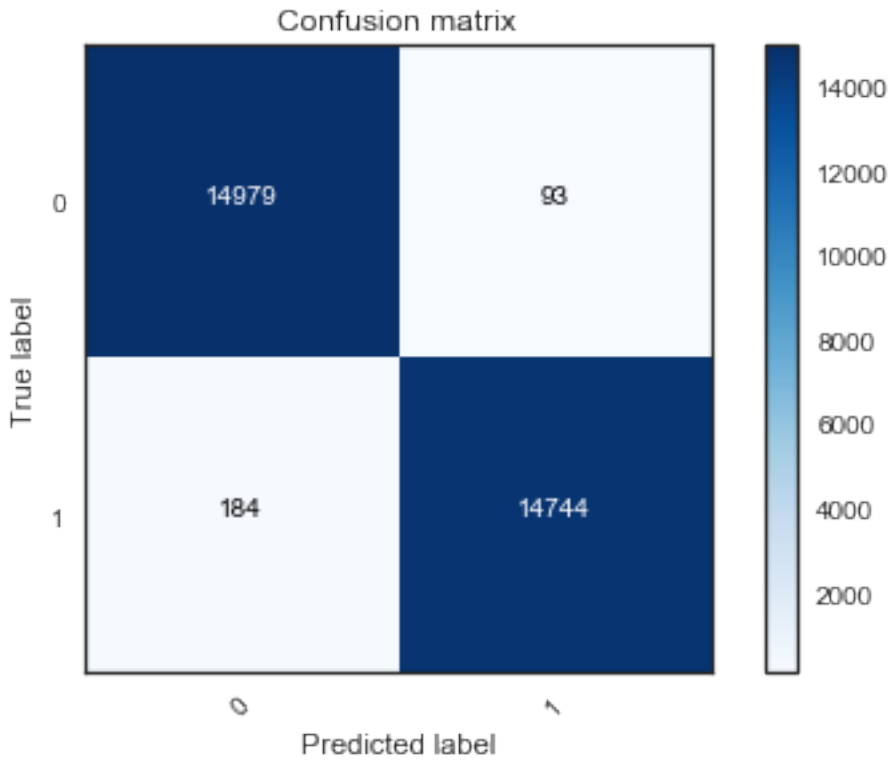
```
        model.add(LeakyReLU(alpha=0.01))
        model.add(MaxPool1D(pool_size=2, strides=2))

        model.add(Conv1D(filters = 10, kernel_size = 3, strides = 1))
        model.add(LeakyReLU(alpha=0.01))
        model.add(MaxPool1D(pool_size=2, strides=2))

        model.add(Flatten())
        model.add(Dense(40))
        model.add(LeakyReLU(alpha=0.01))
        model.add(Dense(20))
        model.add(LeakyReLU(alpha=0.01))
        model.add(Dense(2, activation = "softmax"))

        model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 496, 5)            30
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 496, 5)            0
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 248, 5)            0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 244, 5)            130
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 244, 5)            0
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 122, 5)            0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 120, 10)           160
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 120, 10)           0
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 60, 10)            0
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 58, 10)            310
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 58, 10)            0
_________________________________________________________________
max_pooling1d_4 (MaxPooling1 (None, 29, 10)            0
_________________________________________________________________
flatten_1 (Flatten)          (None, 290)               0
_________________________________________________________________
dense_1 (Dense)              (None, 40)                11640
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 40)                0
```

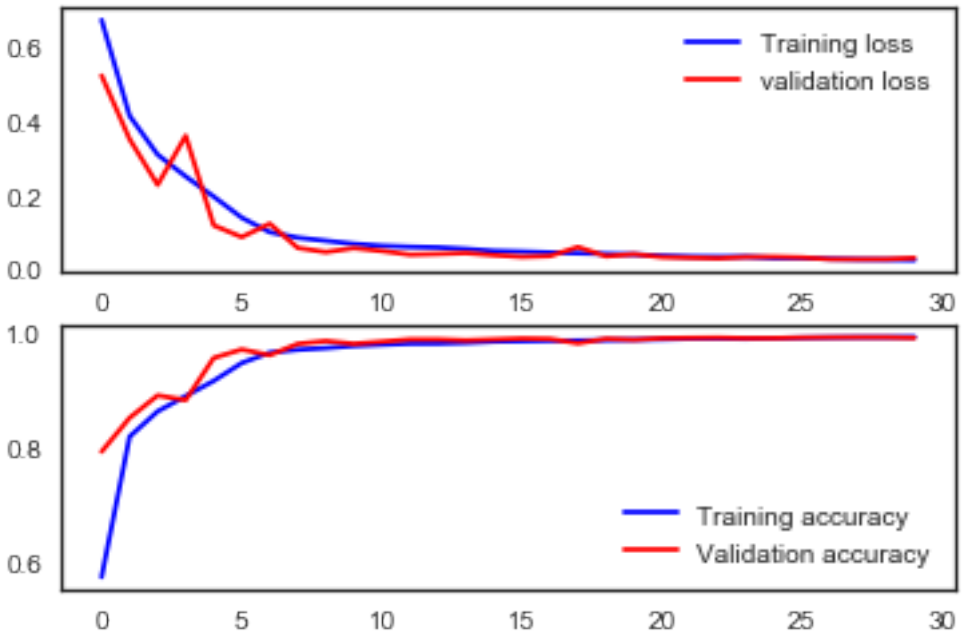
```
_________________________________________________________________
dense_2 (Dense)              (None, 20)                820
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 20)                0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 42
=================================================================
Total params: 13,132
Trainable params: 13,132
Non-trainable params: 0
_________________________________________________________________
```

```
In [18]: X_train.shape

Out[18]: (30000, 500, 1)

In [19]: # Define the optimizer
         optimizer = RMSprop(lr=0.001, rho=0.9, epsilon=1e-08, decay=0.0)

In [20]: # Compile the model
         model.compile(optimizer = optimizer , loss = "categorical_crossentropy", metrics=["ac

In [21]: # Set a learning rate annealer
         learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc', patience=3, verbose=0,

In [22]: epochs = 30 # Turn epochs to 30 to get 0.9967 accuracy
         batch_size = 86

In [23]: X_train.shape

Out[23]: (30000, 500, 1)

In [24]: # Without data augmentation i obtained an accuracy of 0.98114
         history = model.fit(X_train, Y_train, batch_size = batch_size, epochs = epochs,
                   validation_data = (X_test, Y_test), verbose = 0)


In [25]: # Look at confusion matrix

         def plot_confusion_matrix(cm, classes, normalize=False, title='Confusion matrix', cma
             """
             This function prints and plots the confusion matrix.
             Normalization can be applied by setting `normalize=True`.
             """
             plt.imshow(cm, interpolation='nearest', cmap=cmap)
             plt.title(title)
             plt.colorbar()
             tick_marks = np.arange(len(classes))
```

```
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

# Predict the values from the validation dataset
Y_pred = model.predict(X_test)
# Convert predictions classes to one hot vectors
Y_pred_classes = np.argmax(Y_pred,axis = 1)
# Convert validation observations to one hot vectors
Y_true = np.argmax(Y_test,axis = 1)
# compute the confusion matrix
confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
# plot the confusion matrix
plot_confusion_matrix(confusion_mtx, classes = range(2))
```



In [26]:
```
# Plot the loss and accuracy curves for training and validation
fig, ax = plt.subplots(2,1)
ax[0].plot(history.history['loss'], color='b', label="Training loss")
ax[0].plot(history.history['val_loss'], color='r', label="validation loss",axes =ax[0]
legend = ax[0].legend(loc='best', shadow=True)

ax[1].plot(history.history['acc'], color='b', label="Training accuracy")
ax[1].plot(history.history['val_acc'], color='r',label="Validation accuracy")
legend = ax[1].legend(loc='best', shadow=True)
```


Training loss
validation loss
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
1.0
0.8
0.6
Training accuracy
Validation accuracy
0
5
10
15
20
25
30