TSNE

T Distributed Stochastic Neighborhood Embedding

In [1]:

```
# neighbor hood idea preserved
# in LLE the neighbor Wij is deterministically caculated -- no probability
# in TSNE the nighbor weight calculated probabistically -- probob considered
# Xi is determined such that the neighbor Probab Density is Normal
# the data poihnts near Xi--closer to mean--more closer to neighbor
# data points far from Xi---far from mean---not so neighbor

# more probab--near
#lesser probab -far

# neighbors determined not by distance but by conditional probability

# the high dimension to lower dimension thus focus to preserve the Conditional Probab (stochastic neighborhood remains
# probab distribution distance to be kept minmum for determining the neighbor

# preserve the conditionl probab--> during transformation from higher to lower dimension


# when transformation done --problem in Normal Distribution is Compactness/Crowding of data points and difficult to an

# hence T distributed to remove this crowding .....

# stochastic means analysis of probability
```

In [2]:

```
# Perplexity - no of neigbors identified based on conditional probability

# experimentally proved perplexity value btw 5-50 gives good resulted

# TSNE - non linear,manifold learning , Local and Global characterisitc Presrved

```

In [3]:

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

from sklearn.datasets import fetch_openml
```

Accessing the dataset

In [4]:

```
X,y=fetch_openml('mnist_784',version=1,return_X_y=True)
```

In [5]:

```
X.shape
```

Out[5]:

```
(70000, 784)
```

In [6]:

```
y.shape
```

Out[6]:

```
(70000,)
```

In [7]:

```
X.head()
```

Out[7]:

	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	...	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	p
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 784 columns

In [8]:

```
X.iloc[1]
```

Out[8]:

```
pixel1      0.0
pixel2      0.0
pixel3      0.0
pixel4      0.0
pixel5      0.0
           ...
pixel780    0.0
pixel781    0.0
pixel782    0.0
pixel783    0.0
pixel784    0.0
Name: 1, Length: 784, dtype: float64
```

In [9]:

```
y.value_counts()
```

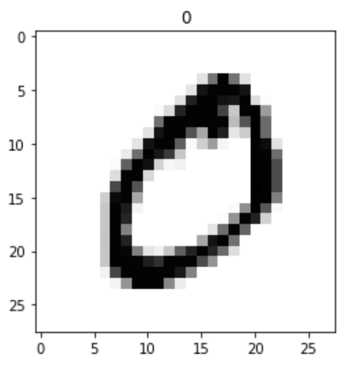
Out[9]:

```
1    7877
7    7293
3    7141
2    6990
9    6958
0    6903
6    6876
8    6825
4    6824
5    6313
Name: class, dtype: int64
```

Plotting the images

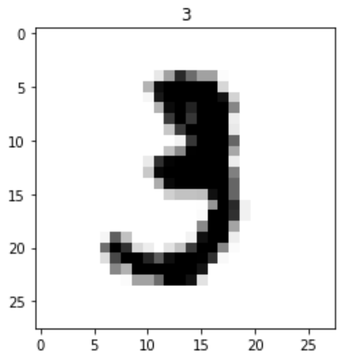
In [11]:

```
plt.imshow(X.iloc[1].to_numpy().reshape(28,28),'Greys')

plt.title(y[1]);
```



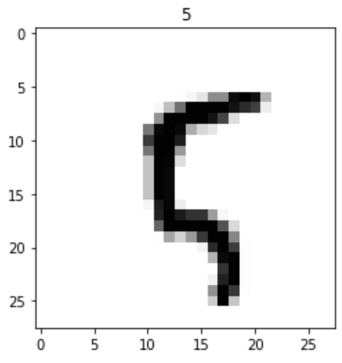
In [12]:

```
plt.imshow(X.iloc[10].to_numpy().reshape(28,28),'Greys')

plt.title(y[10]);
```



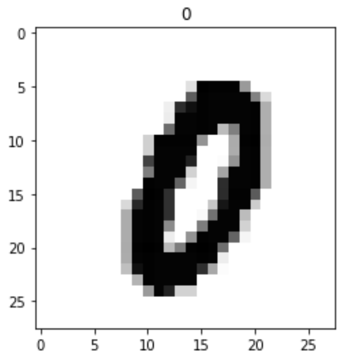
In [13]:

```
plt.imshow(X.iloc[100].to_numpy().reshape(28,28),'Greys')

plt.title(y[100]);
```



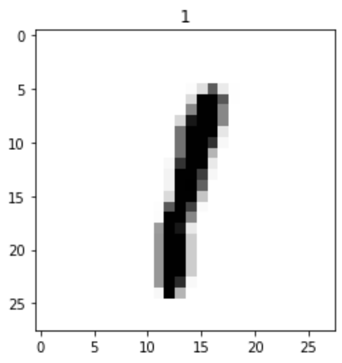
In [14]:

```
plt.imshow(X.iloc[34].to_numpy().reshape(28,28),'Greys')

plt.title(y[34]);
```



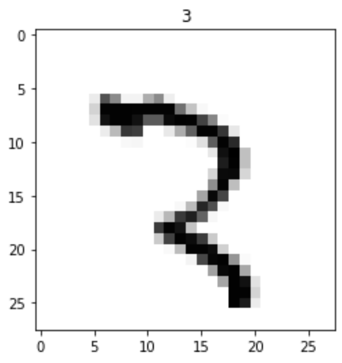
In [15]:

```
plt.imshow(X.iloc[1012].to_numpy().reshape(28,28),'Greys')

plt.title(y[1012]);
```



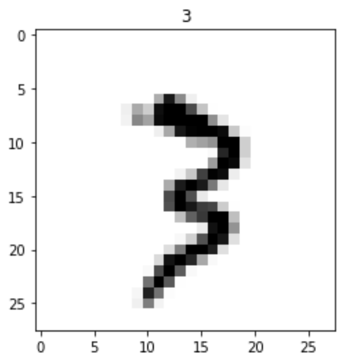
In [16]:

```
plt.imshow(X.iloc[500].to_numpy().reshape(28,28),'Greys')

plt.title(y[500]);
```



In [17]:

```
plt.imshow(X.iloc[25000].to_numpy().reshape(28,28),'Greys')

plt.title(y[25000]);
```



Create a Random Sample of 1000 from 70K

In [19]:

```
np.random.seed(100)
sample=np.random.choice(X.shape[0],1000) # can directly write (70000,1000)
print(sample)
```

```
[38408 56088 65615 63370 14260 20533 13890 49954 59243 17724  7738 42491
 38537  6230 30596 27039 21377 35981  6901 54995 10843 44463 21243 52560
 49154 66425 35294 26003 12571 61240 23790 62833 20382 68528 11567 32423
   806 10412 18624 23480 53685  1738 40977   488 19803 31195 30675 47950
 48128 30156 36995  1667 25812 57738 46295 28150 53743 32700 52073 46256
 46891  6182  5290  6239 14650 14438 63894  7424 42679 41186  4914 63907
 29079 32265 36336 50965 24985 52662 34469 62650 18142  5939  2626  4535
 34905   429 66576 11730 14957 24556 44255 41748 39050 38302 67468 19445
 66909 23415   556 58425  6527 25344  9960 35804 30530 16016 42373 49498
 50014 69844 30579 65684 36376 18874 45852 56018 44679 10815 56450 46295
 28817 11994 25053 26698 28085 34590 68675 42890 49805 60277 17039 39525
 14374 14196 32855 63485 33274 61612  4095  8336 40755  6327    83 34346
 39672 35574 22309  1648 11984 19774 12131 29137 17478 62434 30432 12207
 60297 23407 53028  7881 39539 44414 51524 67283 64583 25405 38386 52347
 11021 24880 63954 31014 10737 64181 34190 45270 13477  2176 29544 23227
 52581 54057 43149 14328 39094 34387 15081  4929  6359 60620 13046  9507
 35866 14922 11740 20276 66445 30040 68247 16618 26171 51704 62472 26189
 37167 37293 22431 40458 55760 24791 49509 63548 56977 27027 49812 18789
 56763 61451 21808 18310 32730 53704 59131 58112 28830 25810 21787 28394
 33205 51188 11877  7335 40883 58663 62382 47420  8825 43903  8567 27353
 51906  3136 60461 34747  7164 66883  2174 35062 69202 18763 58712 34533
 40785 56615  8060 52006 34213 22419 47104 14153 40833 45348 17587  9249
 26976 54102 35577 41370 68182  7388 26078 61315 24092 18335 35603 51442
 14249 49018 53042 38702 23623 24113 45746 25832 69709 40634 51020 19299
 40962  1597 43666 44930 22583 19007 13721 53118 68319 22123 41658 13857
 12482 51179 49170 23014 48194 35492 62437  3953 37349  1214 12417 38655
 64981  1209 50078 18780 26256 55761 50528 10659 66474  5606  5475 52999
 60749 56412 12801 12779 66345 13718 27737 54127 18313 25650 49214 18375
  4110 67585  7815 32063 56367 33137 37736 45850 53842 43408 44484 26095
  5508   826 39849 57381 52517 20987 16805 62915 16303 13288 39830 67560
 20764 43094 45734  3457 62641 40539 14535  7163 21944   782  1120 25311
 52152 33307 37345 60842  1269 62289 44978 54725 41747 37970 26251 69925
 59121 33291  1310 53701 62401 66867 23706 65207 16112 64292 55634 36227
 12585 49670 49264 66430 23826  4991   915 69065 34831 57580 46669 27929
  2469 50745 67610 18875 62773 69550 53551  4182 54385 48685 62908 26717
  4231 29365 61159 43170  3212 49064 17186 56079 27602  3223 67907 33647
 10633 24127 21320 24720 27290 22774 26775 11515 59131  9388 23734 58759
 46948 44467 62546 62790 26591 25678 48790 26523 19511 39042 52079 69603
 21934 49159 21441 51476  4942 10632 62563 52620 16421 15954 24903 64958
 14182 35803 19494  2177 14306 22364 53133 43504 45718 41705 19175 44759
 49235 37618 23462 63793 22907 26147 38804 60640 34257 49983 40963 37830
 34236 25854 34806 36696 14425 19498 16962  6203 46288 28540 31771 52789
 65860 67745 46677  1892   884 37340  9115 42562 41287  5633 48745 32755
 23151 23107 55002  5983 57622 42972 36655 25269 60849 25713 28668 30006
 60793 67491 68801 12540 25370 61804 37355 39158 52718  2847 45910 61747
 42068 35330 37228 27856 30272 28977 67481 24272 18967 55329 24809 48419
 39291 59958 51920 20261 18976  4716 21328 39914 15291  1635 11226 57668
 22167 35500 18328 36851 40735 26223 56410 32066 62828 20302 45729  3439
 33809 34616 65241  2828 69903 44518 19945 37270 62134 59852 24969 13042
 20929 42573 61350 31844 62055 15169 18511 53529  1015  3166 23764 41992
  9640 59164 43374  6328  6324   793 21896 56394 32359 29084 69282 17737
 64152 56919 13546 66049 24954 15243 58047  9099   422 66376 14423 38054
 40732 24034 11020 65984 28275 10439 35531  8292 49476 54653 45568 34205
 43936 40327 65053  4878 20538 51164 23638 67040 24854 60778 26959  3689
 40107 11774 22822  8994 19477 23749 20116 58033 19294 43724 31709 67759
 46197 23129 55099 50784 67943 35069 43561 26821 45108 61479 64075  3307
 48391 17689 31476  9949  7722 12528 53282 59072  7892 53942 48407 26309
 69044 16484 37731 44730 24611 64730 20194 22268 43106 68547 53882 35616
 63828 28917  2121 54092 59972 19325 24399 36924 66113 18216 35726 68316
 57523  9968 55915 50436 42280 31000 19314 16806 26791  2045  1972 42786
  5987 59837 51532  2818 35961 66196  7625 60602  2802 69067 20542 52813
 51776 47319 31776 53096 34884 36945 34736  7799 25996 34952 47270 28076
 34614 33624  2465  4026 58557 52090 15540 31544 57280 10149 21249 19435
 66899 24532 52928 50331 30999 41010 22566 64520 16196  1837 34706 60433
 14937 32828 54986  3855 67024 53652 25271 30546 28734 38213 58595 10140
 39038 53454 22095 27684 44032 24169  9913 31498 58426 33155 18204 61379
 65465 30247 25857 41800  9955 19462 55220 12680 40849 35320 26703 11709
 37621 69910 49184 32461 14787 55870 19693 11463 55255 12413 68111 15593
 54656 13631 52330 69269  2502 22391 28053  3544 51320 43510  1170 33747
 39486 10989 17348 48435 15371 15983 34475  5720 40473 30870 60633 46194
 30038 39379 28126 40403  5094 22752 48006 57536 37103  5950 35538 55802
 41871 22330 54468 15611 17586 54248 55005   136 47996   224 47571 57864
 43033 31577 50728 62712 36986 25372 48481 56478 31895 65977 21831 24218
 62378 25567 58057 54201 33463 44351 53709 52290 63741 17570  3083 25821
 15668 29674 15450 63820 28336  5816 65232 50515 65966 33128 63068  5635
  9174 31412 49846 51413 29090 58533 19053 61388  2724  2343 32731 11318
 17412 13052 59192 64267 15158 34678 50815 34608 46564 64573 37237   714
  4923 55115 12245 24334  4631 43474 65169 37222 37905  4408 32383 66270
 48200 20092 38786 63084 58249 28085 40498 59078  3365 36633 38660 44761
 51668 69087 60622 56750 29951 37697 41042 46333  2915 59163 65502 13588
 51005  3299 53043  5436 60522 15698 30530 36997 29733 48009 66318 49843
   136 26787 68177  9681 18097 36508  8137 39439 11755  2552 17452 21514
 51331  2235 66132 53756 15891 47043 16897 60587 24320  1970  1758 43545
  6051 22851 65851 65658]
```

Creating A subset of 1k samples of X,y

In [20]:

```
X1=X.iloc[sample,:] # iloc for rows
X1.shape
```

Out[20]:

```
(1000, 784)
```

In [21]:

```
X1.head()
```

Out[21]:

	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	...	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel78
38408	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0
56088	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0
65615	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0
63370	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0
14260	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0

5 rows × 784 columns

In [22]:

```
y1=y[sample]
y1.shape
```

Out[22]:

```
(1000,)
```

In [23]:

```
y1.head()
```

Out[23]:

```
38408    5
56088    2
65615    2
63370    4
14260    4
Name: class, dtype: category
Categories (10, object): ['0', '1', '2', '3', ..., '6', '7', '8', '9']
```

Building TSNE model

In [24]:

```
from sklearn.manifold import TSNE

tsne=TSNE(n_components=2,perplexity=30)
```

In [25]:

```
X_tsne=tsne.fit_transform(X1)
X_tsne.shape
```

```
C:\Users\Vishrut\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:780: FutureWarning: The default initi
alization in TSNE will change from 'random' to 'pca' in 1.2.
  warnings.warn(
C:\Users\Vishrut\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:790: FutureWarning: The default learn
ing rate in TSNE will change from 200.0 to 'auto' in 1.2.
  warnings.warn(
```

Out[25]:

```
(1000, 2)
```

In [27]:

```
X_tsne[0] # dimension reduced to 2
```

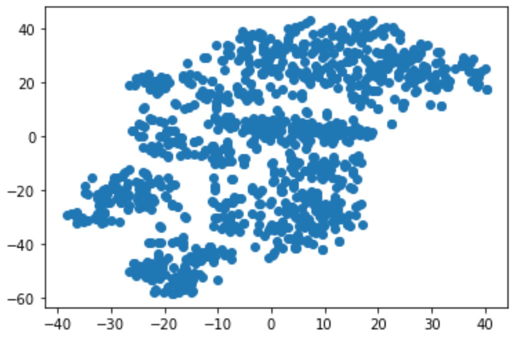
Out[27]:

```
array([22.939573, 13.993068], dtype=float32)
```

Plotting transformed data point

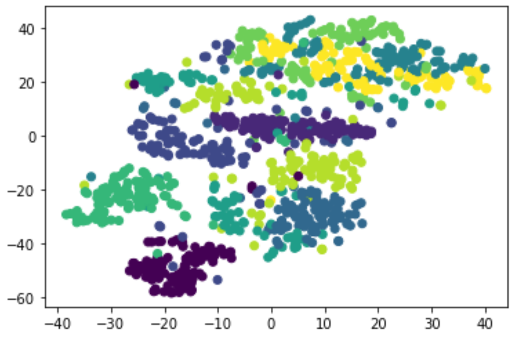
In [28]:

```
plt.scatter(X_tsne[:,0],X_tsne[:,1]);
```



In [29]:

```
plt.scatter(X_tsne[:,0],X_tsne[:,1],c=y1.astype(float));
```



Creating A DF

In [30]:

```
X_df=pd.DataFrame({'X0':X_tsne[:,0],
                   'X1':X_tsne[:,1],
                   'Label':y1})

X_df
```

Out[30]:

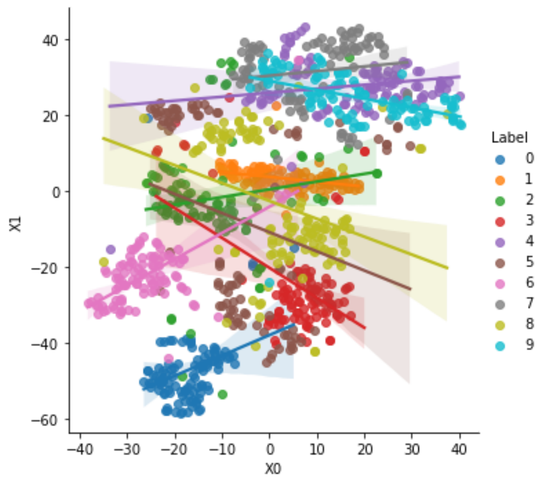
	X0	X1	Label
38408	22.939573	13.993068	5
56088	-13.561443	-1.145843	2
65615	-9.424026	-8.405927	2
63370	34.267212	25.340717	4
14260	29.056919	34.054592	4
...	...	...	...
43545	-24.426233	-17.090849	6
6051	1.756038	30.691936	9
22851	-29.010527	-18.734632	6
65851	20.353048	20.824131	4
65658	-20.211996	20.025196	5

1000 rows × 3 columns

In [32]:

```python
plt.figure(figsize=(15,12))
sns.lmplot(data=X_df,x='X0',y='X1',hue='Label');
```

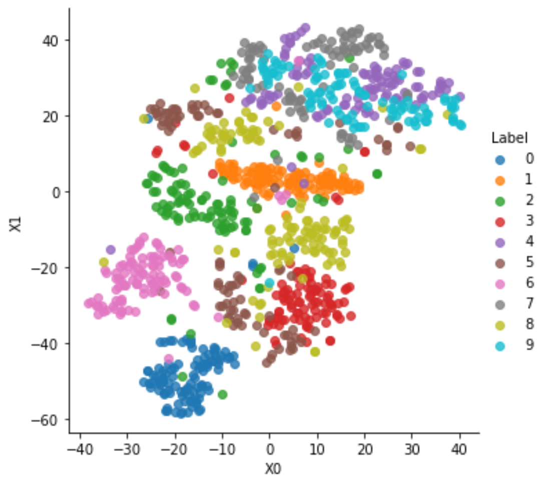
```
<Figure size 1080x864 with 0 Axes>
```



In [33]:

```python
plt.figure(figsize=(15,12))
sns.lmplot(data=X_df,x='X0',y='X1',hue='Label',fit_reg=False);
```

```
<Figure size 1080x864 with 0 Axes>
```



exercise :

perform TSNE on wine dataset

In [34]:

```python
wine=pd.read_csv('https://gist.githubusercontent.com/tijptjik/9408623/raw/b237fa5848349a14a14e5d4107dc7897c21951f5/win
```

In [35]:

```
wine
```

Out[35]:

	Wine	Alcohol	Malic.acid	Ash	Acl	Mg	Phenols	Flavanoids	Nonflavanoid.phenols	Proanth	Color.int	Hue	OD	Proline
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
173	3	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	1.06	7.70	0.64	1.74	740
174	3	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	1.41	7.30	0.70	1.56	750
175	3	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	1.35	10.20	0.59	1.56	835
176	3	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	1.46	9.30	0.60	1.62	840
177	3	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	1.35	9.20	0.61	1.60	560

178 rows × 14 columns

In [39]:

```
X=wine.drop(['Wine'],axis=1)
X
```

Out[39]:

	Alcohol	Malic.acid	Ash	Acl	Mg	Phenols	Flavanoids	Nonflavanoid.phenols	Proanth	Color.int	Hue	OD	Proline
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735
...	...	...	...	...	...	...	...	...	...	...	...	...	...
173	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	1.06	7.70	0.64	1.74	740
174	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	1.41	7.30	0.70	1.56	750
175	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	1.35	10.20	0.59	1.56	835
176	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	1.46	9.30	0.60	1.62	840
177	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	1.35	9.20	0.61	1.60	560

178 rows × 13 columns

In [37]:

```
y=wine['Wine']
y

```

Out[37]:

```
0      1
1      1
2      1
3      1
4      1
      ..
173    3
174    3
175    3
176    3
177    3
Name: Wine, Length: 178, dtype: int64
```

In [40]:

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_scaled=sc.fit_transform(X)
X_scaled
```

Out[40]:

```
array([[ 1.51861254, -0.5622498 ,  0.23205254, ...,  0.36217728,
         1.84791957,  1.01300893],
       [ 0.24628963, -0.49941338, -0.82799632, ...,  0.40605066,
         1.1134493 ,  0.96524152],
       [ 0.19687903,  0.02123125,  1.10933436, ...,  0.31830389,
         0.78858745,  1.39514818],
       ...,
       [ 0.33275817,  1.74474449, -0.38935541, ..., -1.61212515,
        -1.48544548,  0.28057537],
       [ 0.20923168,  0.22769377,  0.01273209, ..., -1.56825176,
        -1.40069891,  0.29649784],
       [ 1.39508604,  1.58316512,  1.36520822, ..., -1.52437837,
        -1.42894777, -0.59516041]])
```

In [41]:

```
X_tsne1=tsne.fit_transform(X_scaled)
X_tsne1.shape
```

```
C:\Users\Vishrut\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:780: FutureWarning: The default initi
alization in TSNE will change from 'random' to 'pca' in 1.2.
  warnings.warn(
C:\Users\Vishrut\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:790: FutureWarning: The default learn
ing rate in TSNE will change from 200.0 to 'auto' in 1.2.
  warnings.warn(
```

Out[41]:

```
(178, 2)
```

In [43]:

```
X1_df=pd.DataFrame({'X0':X_tsne1[:,0],
                    'X1':X_tsne1[:,1],
                    'Label':y})

X1_df
```

Out[43]:

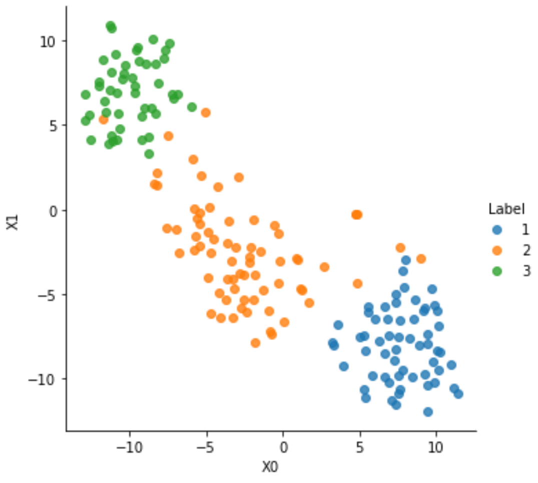
	X0	X1	Label
0	7.347120	-11.491316	1
1	5.356689	-8.376481	1
2	9.458805	-7.371906	1
3	9.938459	-10.236125	1
4	7.817649	-3.619836	1
...	...	...	...
173	-12.022795	7.351344	3
174	-10.405829	8.078941	3
175	-8.957156	8.595711	3
176	-9.391712	8.767690	3
177	-11.709583	8.868699	3

178 rows × 3 columns

In [47]:

```
plt.figure(figsize=(15,12))
sns.lmplot(data=X1_df,x='X0',y='X1',hue='Label',fit_reg=False);
```

```
<Figure size 1080x864 with 0 Axes>
```



In [ ]:

```

```