Problem 01

In [1]:
```python
#Import basic library libraries

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```python
# Importing Dataset

swiss = pd.read_csv(r'C:\Users\ASUS\Downloads\swiss.csv')
```

In [3]:
```python
swiss.head()
```

Out[3]:

	Fertility	Agriculture	Examination	Education	Catholic	Infant.Mortality
0	80.2	17.0	15	12	9.96	22.2
1	83.1	45.1	6	9	84.84	22.2
2	92.5	39.7	5	5	93.40	20.2
3	85.8	36.5	12	7	33.77	20.3
4	76.9	43.5	17	15	5.16	20.6

In [4]:
```python
swiss.shape
```

Out[4]:
```
(47, 6)
```

In [5]:
```python
print(swiss.isnull().sum())
print(swiss.info())
```

```
Fertility           0
Agriculture         0
Examination         0
Education           0
Catholic            0
Infant.Mortality    0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47 entries, 0 to 46
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Fertility         47 non-null     float64
 1   Agriculture       47 non-null     float64
 2   Examination       47 non-null     int64
 3   Education         47 non-null     int64
 4   Catholic          47 non-null     float64
 5   Infant.Mortality  47 non-null     float64
dtypes: float64(4), int64(2)
memory usage: 2.3 KB
None
```

In [6]:
```python
# normalizing the data


from sklearn.preprocessing import StandardScaler
scale=StandardScaler()
```

In [7]:
```python
swiss_Scaled = scale.fit_transform(swiss)
```

In [8]:
```python
# Implementation of PCA

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

In [9]:
```python
pca.fit(swiss_Scaled)  #( explained variance)
```

Out[9]:
```
PCA()
```

In [10]:
```python
# extract the eigenvalues
eigValues = pca.explained_variance_
```

In [11]:
```python
eigValues
```

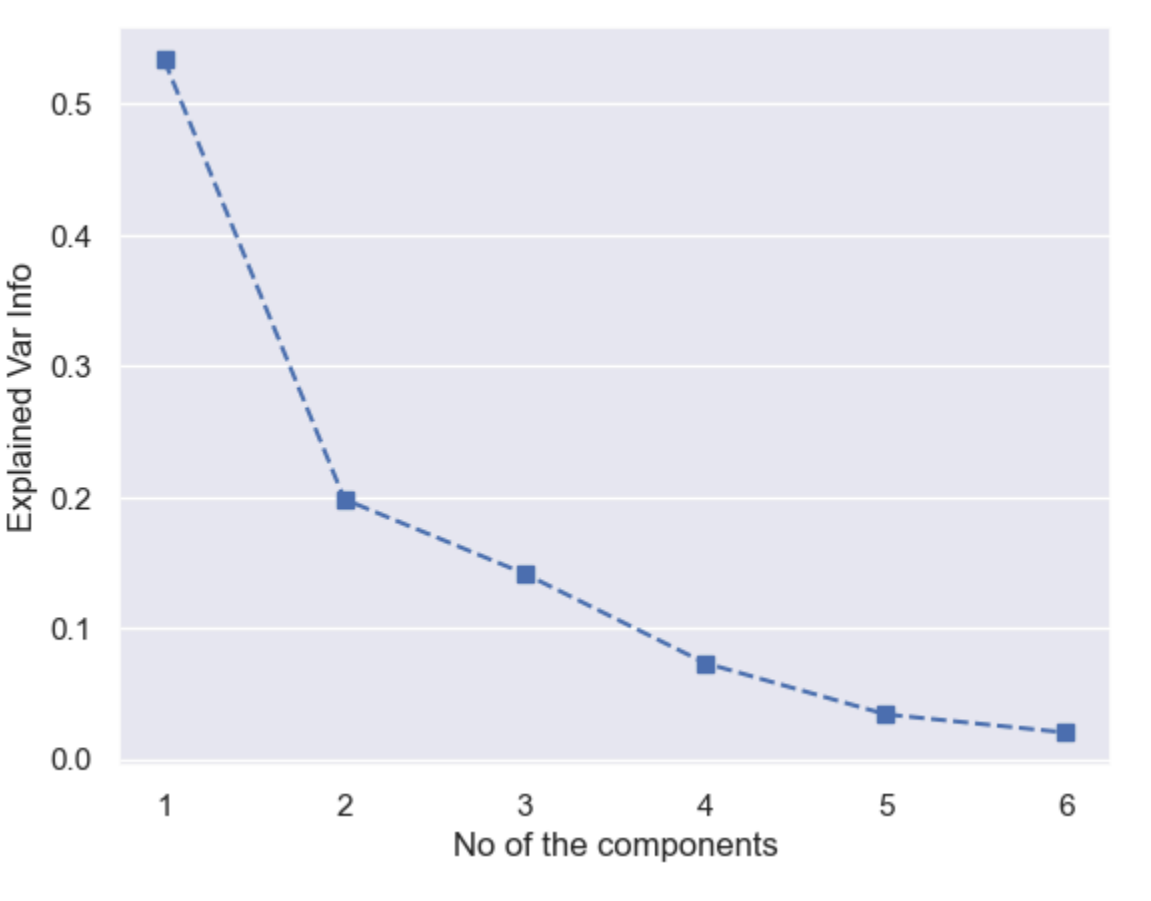
Out[11]:
```
array([3.26931693, 1.21414104, 0.86603611, 0.44847061, 0.20898009,
       0.12349   ])
```

In [12]:
```python
# the percentage of information
ratio = pca.explained_variance_ratio_
```

In [13]:
```python
ratio
```

Out[13]:
```
array([0.53329283, 0.19805137, 0.1412683 , 0.07315478, 0.03408895,
       0.02014376])
```

In [14]:
```python
plt.plot(ratio, 's--')
plt.xlabel("No of the components")
plt.ylabel("Explained Var Info")
plt.xticks(list(range(0,len(ratio))),list(range(1,len(ratio)+1)))
plt.grid(axis='x')
```



In [15]:
```python
# No of components required from the above graph = 2
```

In [ ]:

Problem 02

T-Distributed Stochastic Neighbourhood Embedding(t-SNE)

In [16]:
```python
# Import the basic libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [17]:
```python
# Importing dataset

opt = pd.read_csv(r'C:\Users\ASUS\Downloads\optdigits.csv')
```

In [18]:
```python
opt.head()
```

Out[18]:

	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	...	P55	P56	P57	P58	P59	P60	P61	P62	P63	y
0	0	0	5	14	4	0	0	0	0	0	...	0	0	0	4	12	14	7	0	0	6
1	0	0	0	3	16	11	1	0	0	0	...	0	0	0	0	2	14	14	1	0	1
2	0	0	7	11	11	6	0	0	0	9	...	0	0	0	14	16	12	10	1	0	3
3	0	0	9	13	1	0	0	0	0	0	...	5	0	0	4	15	16	16	16	16	1
4	0	0	0	10	12	0	0	0	0	0	...	0	0	0	1	11	14	12	1	0	6

5 rows × 65 columns

In [19]:
```python
opt.shape
```

Out[19]:
```
(1155, 65)
```

In [20]:
```python
print(opt.info())
print(opt.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1155 entries, 0 to 1154
Data columns (total 65 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   P0      1155 non-null   int64
 1   P1      1155 non-null   int64
 2   P2      1155 non-null   int64
 3   P3      1155 non-null   int64
 4   P4      1155 non-null   int64
 5   P5      1155 non-null   int64
 6   P6      1155 non-null   int64
 7   P7      1155 non-null   int64
 8   P8      1155 non-null   int64
 9   P9      1155 non-null   int64
 10  P10     1155 non-null   int64
 11  P11     1155 non-null   int64
 12  P12     1155 non-null   int64
 13  P13     1155 non-null   int64
 14  P14     1155 non-null   int64
 15  P15     1155 non-null   int64
 16  P16     1155 non-null   int64
 17  P17     1155 non-null   int64
 18  P18     1155 non-null   int64
 19  P19     1155 non-null   int64
 20  P20     1155 non-null   int64
 21  P21     1155 non-null   int64
 22  P22     1155 non-null   int64
 23  P23     1155 non-null   int64
 24  P24     1155 non-null   int64
 25  P25     1155 non-null   int64
 26  P26     1155 non-null   int64
 27  P27     1155 non-null   int64
 28  P28     1155 non-null   int64
 29  P29     1155 non-null   int64
 30  P30     1155 non-null   int64
 31  P31     1155 non-null   int64
 32  P32     1155 non-null   int64
 33  P33     1155 non-null   int64
 34  P34     1155 non-null   int64
 35  P35     1155 non-null   int64
 36  P36     1155 non-null   int64
 37  P37     1155 non-null   int64
 38  P38     1155 non-null   int64
 39  P39     1155 non-null   int64
 40  P40     1155 non-null   int64
 41  P41     1155 non-null   int64
 42  P42     1155 non-null   int64
 43  P43     1155 non-null   int64
 44  P44     1155 non-null   int64
 45  P45     1155 non-null   int64
 46  P46     1155 non-null   int64
 47  P47     1155 non-null   int64
 48  P48     1155 non-null   int64
 49  P49     1155 non-null   int64
 50  P50     1155 non-null   int64
 51  P51     1155 non-null   int64
 52  P52     1155 non-null   int64
 53  P53     1155 non-null   int64
 54  P54     1155 non-null   int64
 55  P55     1155 non-null   int64
 56  P56     1155 non-null   int64
 57  P57     1155 non-null   int64
 58  P58     1155 non-null   int64
 59  P59     1155 non-null   int64
 60  P60     1155 non-null   int64
 61  P61     1155 non-null   int64
 62  P62     1155 non-null   int64
 63  P63     1155 non-null   int64
 64  y       1155 non-null   int64
dtypes: int64(65)
memory usage: 586.6 KB
None
P0     0
P1     0
P2     0
P3     0
P4     0
      ..
P60    0
P61    0
P62    0
P63    0
y      0
Length: 65, dtype: int64
```

In [21]:
```python
final_opt = opt.drop(columns=['y'])
```

In [22]:
```python
final_opt.head()
```

Out[22]:

	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	...	P54	P55	P56	P57	P58	P59	P60	P61	P62	P63
0	0	0	5	14	4	0	0	0	0	0	...	12	0	0	0	4	12	14	7	0	0
1	0	0	0	3	16	11	1	0	0	0	...	0	0	0	0	0	2	14	14	1	0
2	0	0	7	11	11	6	0	0	0	9	...	12	0	0	0	14	16	12	10	1	0
3	0	0	9	13	1	0	0	0	0	0	...	8	5	0	0	4	15	16	16	16	16
4	0	0	0	10	12	0	0	0	0	0	...	14	0	0	0	1	11	14	12	1	0

5 rows × 64 columns

In [23]:
```python
# t-SNE

from sklearn.manifold import TSNE
```

In [24]:
```python
tsne =TSNE(n_components=3, learning_rate = 200, random_state=42, perplexity=1.0, n_iter=5000)
```

In [25]:
```python
x = tsne.fit_transform(final_opt)
```

In [26]:
```python
x
```

Out[26]:
```
array([[-21.209711, -14.911261,  24.415073],
       [ 23.91669 ,  34.352238,  18.247671],
       [-22.22433 , -23.099993, -31.063364],
       ...,
       [ -3.051235, -23.116709,   9.834617],
       [ 13.305171,  -5.816824, -50.663998],
       [  5.813324,  -8.33331 ,  48.568733]], dtype=float32)
```

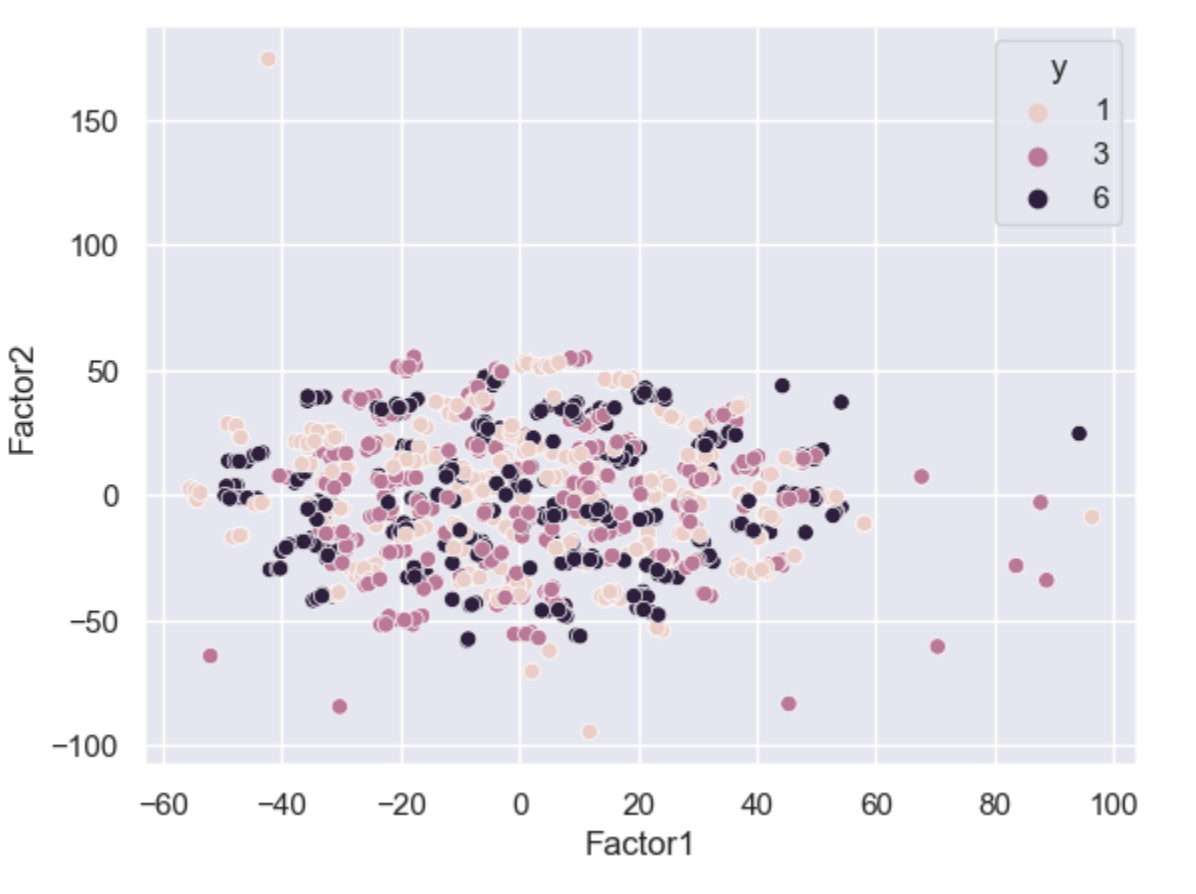
In [27]:
```python
opt_reduced = pd.DataFrame(x, columns=['Factor1','Factor2','Factor3'])
opt_reduced.head()
```

Out[27]:

	Factor1	Factor2	Factor3
0	-21.209711	-14.911261	24.415073
1	23.916691	34.352238	18.247671
2	-22.224331	-23.099993	-31.063364
3	-0.934462	-39.069431	44.554852
4	-17.098038	-31.905859	-5.987141

In [28]:
```python
sns.scatterplot(opt_reduced['Factor1'],opt_reduced['Factor2'], hue=opt['y'])
```

Out[28]:
```
<AxesSubplot:xlabel='Factor1', ylabel='Factor2'>
```



In [29]:
```python
tsne =TSNE(n_components=3, learning_rate = 200, random_state=42, perplexity=2.0, n_iter=5000)
```

In [30]:
```python
Y = tsne.fit_transform(final_opt)
```

In [31]:
```python
Y
```

Out[31]:
```
array([[ 34.50203  , -20.686102 ,  12.483117 ],
       [ 25.115973 ,   5.274    ,  30.226988 ],
       [-31.390554 ,  -6.659697 , -15.162277 ],
       ...,
       [-13.624261 ,   1.9471681, -26.341352 ],
       [ 13.814089 , -28.159151 , -22.414064 ],
       [ 33.90275  ,  -8.6862755,   4.0720162]], dtype=float32)
```

In [32]:
```python
opt_reduced = pd.DataFrame(Y, columns=['Factor1','Factor2','Factor3'])
opt_reduced.head()
```

Out[32]:

	Factor1	Factor2	Factor3
0	34.502029	-20.686102	12.483117
1	25.115973	5.274000	30.226988
2	-31.390554	-6.659697	-15.162277
3	-5.327826	8.335792	41.527443
4	29.136940	-29.469423	15.345924

In [33]:
```python
sns.scatterplot(opt_reduced['Factor1'],opt_reduced['Factor2'], hue=opt['y'])
```

Out[33]:
```
<AxesSubplot:xlabel='Factor1', ylabel='Factor2'>
```

In [34]:
```python
tsne =TSNE(n_components=3, learning_rate = 200, random_state=42, perplexity=5.0, n_iter=5000)
```

In [35]:
```python
A = tsne.fit_transform(final_opt)
```

In [36]:
```python
A
```

Out[36]:
```
array([[ -2.404165 ,   3.875909 , -16.27112  ],
       [ -5.2099986,  13.580372 ,  19.020296 ],
       [ 27.762115 ,   3.7767475,   5.763916 ],
       ...,
       [ 14.116489 ,  -3.5393512,  -3.2880824],
       [-18.01459  ,  -6.463248 , -25.732946 ],
       [ -3.1417108, -11.739696 , -18.920662 ]], dtype=float32)
```

In [37]:
```python
opt_reduced = pd.DataFrame(A, columns=['Factor1','Factor2','Factor3'])
opt_reduced.head()
```

Out[37]:

	Factor1	Factor2	Factor3
0	-2.404165	3.875909	-16.271120
1	-5.209999	13.580372	19.020296
2	27.762115	3.776747	5.763916
3	-2.255909	-11.382578	5.202106
4	-8.629186	5.720033	-14.395181

In [38]:
```python
sns.scatterplot(opt_reduced['Factor1'],opt_reduced['Factor2'], hue=opt['y'])
```

Out[38]:
```
<AxesSubplot:xlabel='Factor1', ylabel='Factor2'>
```

In [39]:
```python
tsne =TSNE(n_components=3, learning_rate = 200, random_state=42, perplexity=10.0, n_iter=5000)
```

In [40]:
```python
B = tsne.fit_transform(final_opt)
```

In [41]:
```python
B
```

Out[41]:
```
array([[-17.489122 ,   1.9962559,  -8.170795 ],
       [ 10.059246 ,  11.632093 ,  11.379742 ],
       [ 12.778338 ,  -6.1623335, -18.01434  ],
       ...,
       [ -3.7933543,  -6.090697 , -14.670245 ],
       [-18.135881 ,  -4.0833144,   4.079533 ],
       [-25.378677 ,  -6.383252 ,  -5.686799 ]], dtype=float32)
```

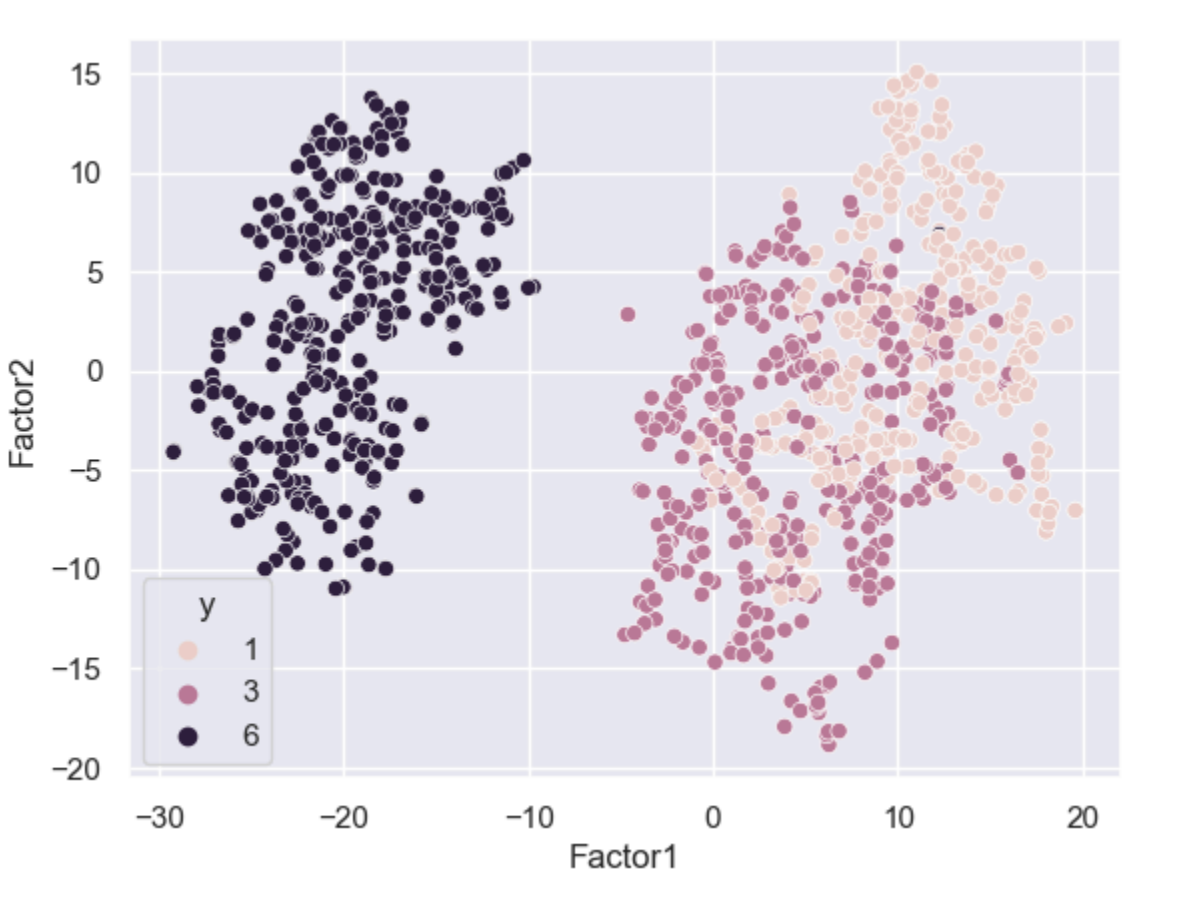
In [42]:
```python
opt_reduced = pd.DataFrame(B, columns=['Factor1','Factor2','Factor3'])
opt_reduced.head()
```

Out[42]:

	Factor1	Factor2	Factor3
0	-17.489122	1.996256	-8.170795
1	10.059246	11.632093	11.379742
2	12.778338	-6.162333	-18.014339
3	16.565680	-6.067654	7.387315
4	-18.559565	3.526744	-3.404772

In [43]:
```python
sns.scatterplot(opt_reduced['Factor1'],opt_reduced['Factor2'], hue=opt['y'])
```

Out[43]:
```
<AxesSubplot:xlabel='Factor1', ylabel='Factor2'>
```



In [44]:
```python
tsne =TSNE(n_components=3, learning_rate = 200, random_state=42, perplexity=50.0, n_iter=5000)
```

In [45]:
```python
C = tsne.fit_transform(final_opt)
```

In [46]:
```python
C
```

Out[46]:
```
array([[ -1.9105376 ,  11.974509  ,  13.755121  ],
       [  7.320098  , -13.108042  ,  -1.2420027 ],
       [ -1.1750499 ,  -0.44684643, -12.649712  ],
       ...,
       [  3.4799898 ,   3.094847  ,  -6.7340794 ],
       [ -0.711605  ,  11.20505   ,   6.6383643 ],
       [ -5.0895486 ,  12.412542  ,   9.0917845 ]], dtype=float32)
```

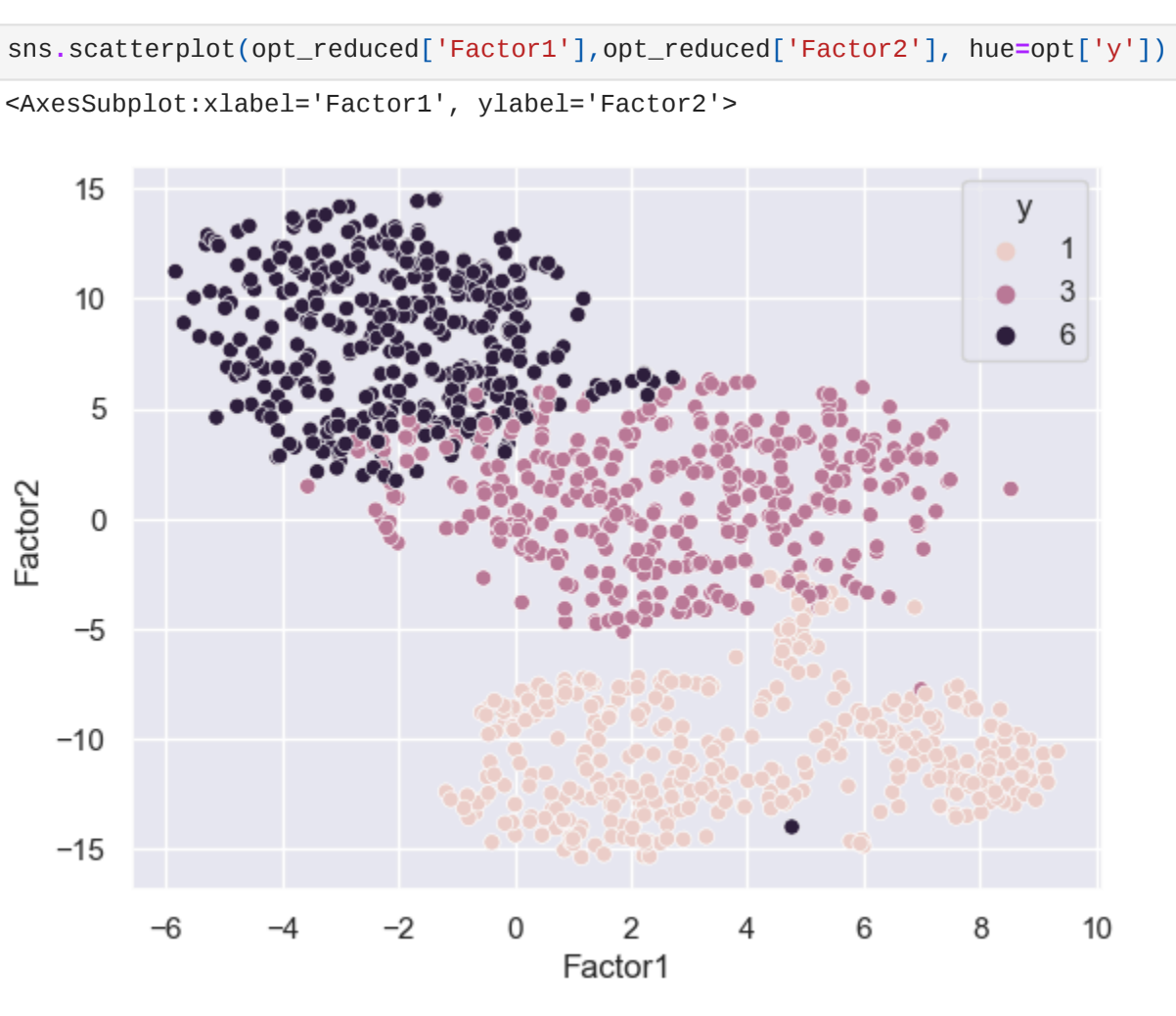
In [47]:
```python
opt_reduced = pd.DataFrame(C, columns=['Factor1','Factor2','Factor3'])
opt_reduced.head()
```

Out[47]:

	Factor1	Factor2	Factor3
0	-1.910538	11.974509	13.755121
1	7.320098	-13.108042	-1.242003
2	-1.175050	-0.446846	-12.649712
3	5.338614	-3.659347	1.834520
4	-1.736414	9.808995	12.130286

In [48]:
```python
sns.scatterplot(opt_reduced['Factor1'],opt_reduced['Factor2'], hue=opt['y'])
```

Out[48]:
```
<AxesSubplot:xlabel='Factor1', ylabel='Factor2'>
```



In [ ]:
```python
##perplexity clusters (50) are perfectly separated and more concrete.
```