Exploring Gisette Data

Data: "Gisette" - highly confusible digits '4' and '9' data with dim = 5000 ¶

Author: Kuiyu Zhu

In [20]:
```
# import libraries
import numpy as np
import pandas as pd
import time
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.manifold.t_sne import _joint_probabilities
from scipy.spatial.distance import squareform
from sklearn.manifold import TSNE
import seaborn as sns
# set up seaborn plots
sns.set(rc={'figure.figsize':(12,8)})
palette = sns.color_palette('bright', 2)
```

In [21]:
```
# import gisette data
f=open('gisette_train.data',"r")
data=[]
for row in f.readlines():
    data.append((row.strip()).split(" "))
f.close()

# import labels
f= open("gisette_train.labels")
labels=[]
for row in f.readlines():
    labels.append((row.strip()).split(" "))
f.close()
```

In [22]:
```
# data and labels to arrays
data = np.array(data).astype(int)
labels = np.array(labels).astype(int)
labels = labels[:,0]
```

In [23]:
```
# scale the data
scaler = MinMaxScaler()
data = scaler.fit_transform(data)
```

In [24]:
```
# take a look of our data
data, data.shape
```

Out[24]:
```
(array([[0.55055055, 0.        , 0.4954955 , ..., 0.        , 0.        ,
         0.98398398],
        [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.        , 0.99199199, ..., 0.        , 0.        ,
         0.99199199]]),
 (6000, 5000))
```

In [25]:
```
# take a look of our labels
labels, labels.shape
```

Out[25]:
```
(array([ 1, -1,  1, ..., -1, -1, -1]), (6000,))
```

UMAP

In [47]:
```
import umap
```

In [48]:
```
umap_reducer = umap.UMAP(random_state=42)
umap_embedding = umap_reducer.fit_transform(data)
umap_embedding
```

Out[48]:
```
array([[7.341459 , 5.3794956],
       [3.324797 , 4.213805 ],
       [4.6126795, 3.7046814],
       ...,
       [3.0107596, 3.9624276],
       [1.8812418, 3.8654037],
       [2.8293433, 8.389717 ]], dtype=float32)
```

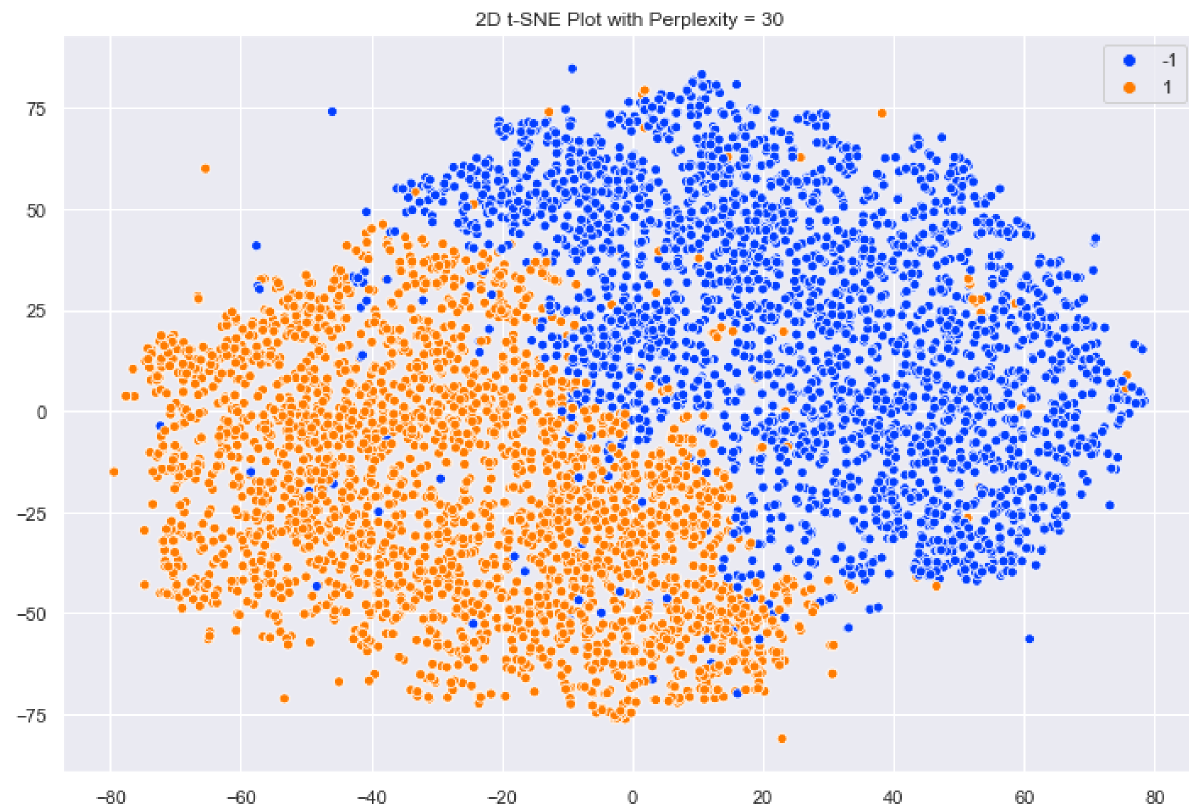
In [49]:
```
umap_embedding.shape
```

Out[49]:
```
(6000, 2)
```

In [50]:
```
sns.scatterplot(umap_embedding[:,0], umap_embedding[:,1], hue=labels, le
gend='full', palette=palette)
plt.title('2D UMAP Scatter Plot')
plt.show()
```

t-SNE

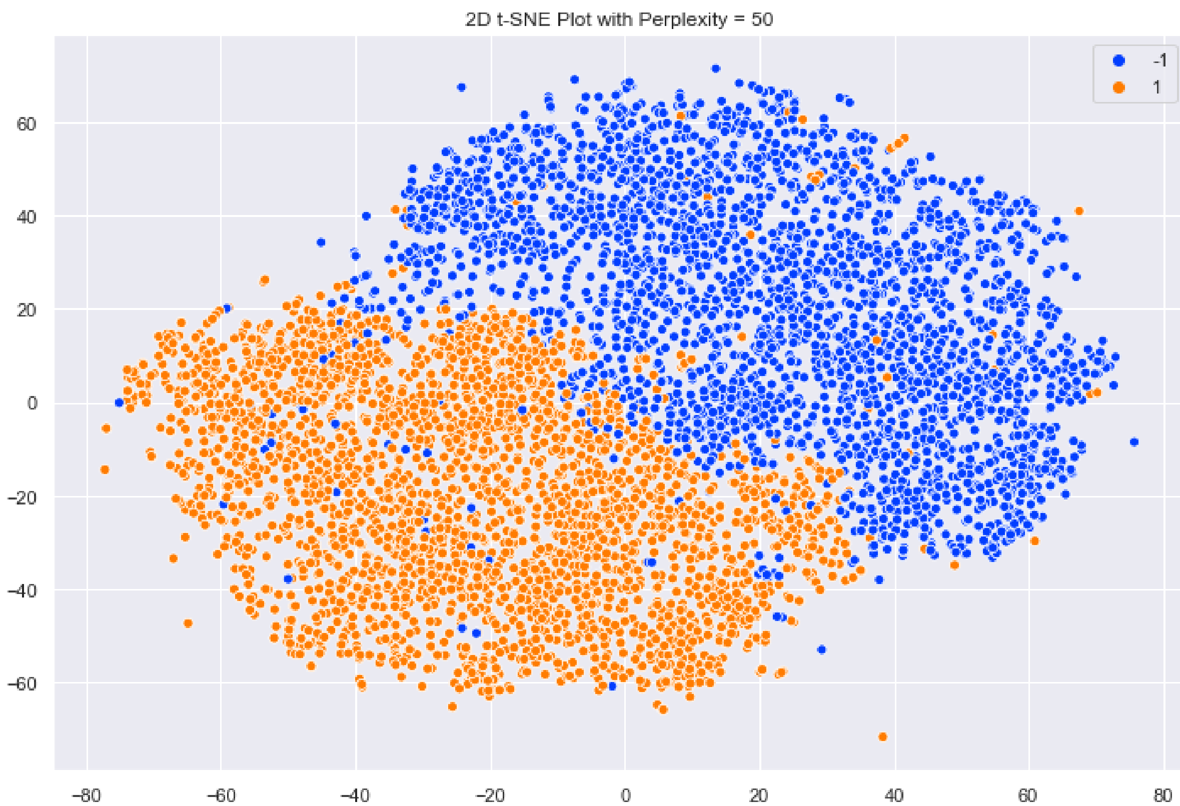
In [26]:
```
# tsne1 - n_components = 2 default, perplexity = 30 default
startTime1 = time.time()
tsne = TSNE()
data_embedded = tsne.fit_transform(data)
# visualization - 2D
sns.scatterplot(data_embedded[:,0], data_embedded[:,1], hue=labels, lege
nd='full', palette=palette)
plt.title('2D t-SNE Plot with Perplexity = 30')
plt.show()
endTime1 = time.time()
```



In [27]:
```
print('Time of execution is ', (endTime1 - startTime1),"second.")
```

```
Time of execution is  265.99127316474915 second.
```

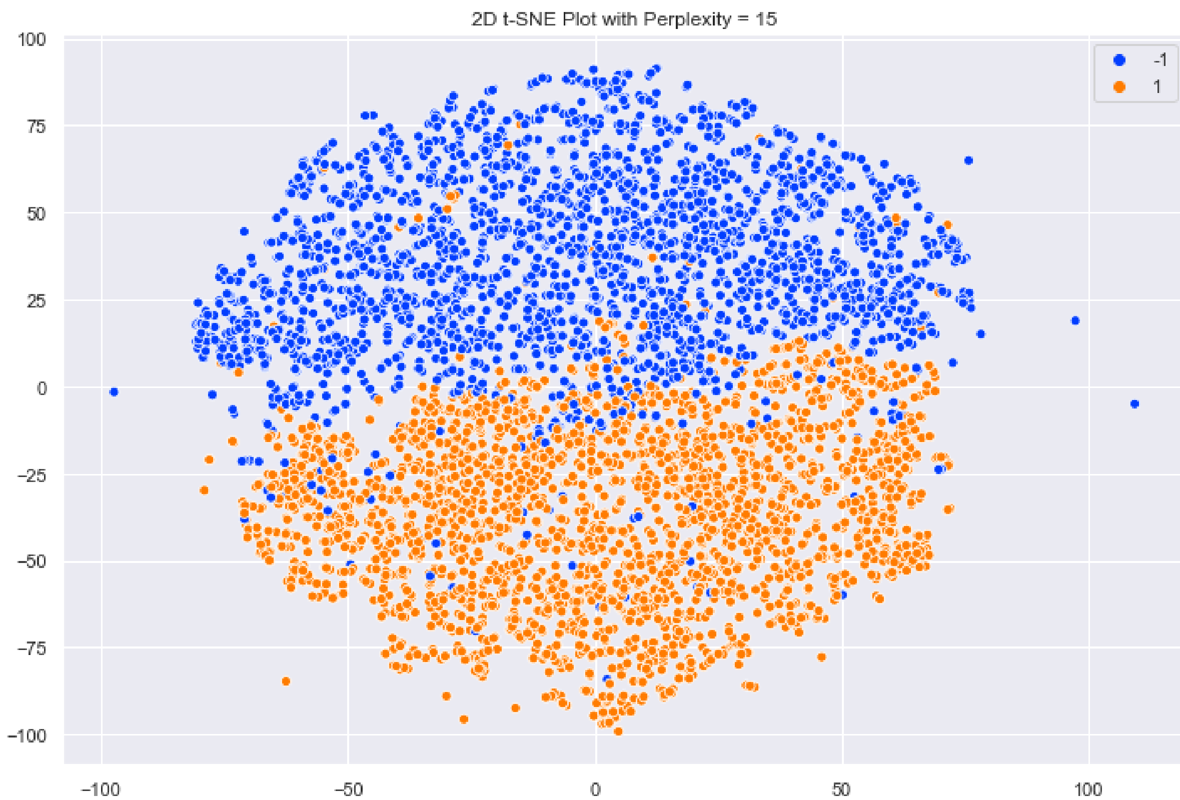
In [28]:
```
# tsne2 - n_components = 2 default, perplexity =  50
startTime2 = time.time()
tsne2 = TSNE(perplexity = 50)
data_embedded = tsne2.fit_transform(data)
# visualization - 2D
sns.scatterplot(data_embedded[:,0], data_embedded[:,1], hue=labels, lege
nd='full', palette=palette)
plt.title('2D t-SNE Plot with Perplexity = 50')
plt.show()
endTime2 = time.time()
```



In [29]:
```
print('Time of execution is ', (endTime2 - startTime2),"second.")
```

```
Time of execution is  269.4928369522095 second.
```

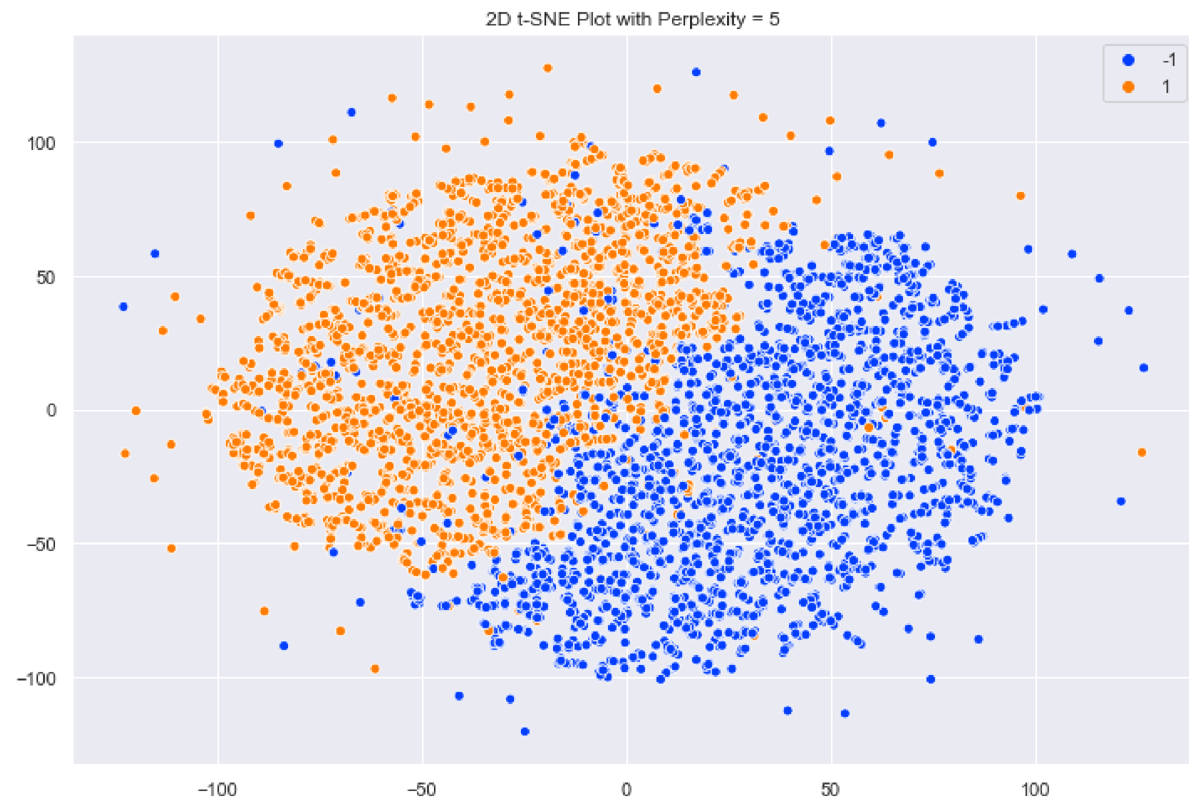
In [30]:
```
# tsne3 - n_components = 2 default, perplexity =  15
startTime3 = time.time()
tsne3 = TSNE(perplexity = 15)
data_embedded = tsne3.fit_transform(data)
# visualization - 2D
sns.scatterplot(data_embedded[:,0], data_embedded[:,1], hue=labels, lege
nd='full', palette=palette)
plt.title('2D t-SNE Plot with Perplexity = 15')
plt.show()
endTime3 = time.time()
```



In [31]:
```
print('Time of execution is ', (endTime3 - startTime3),"second.")
```

```
Time of execution is  267.2549409866333 second.
```

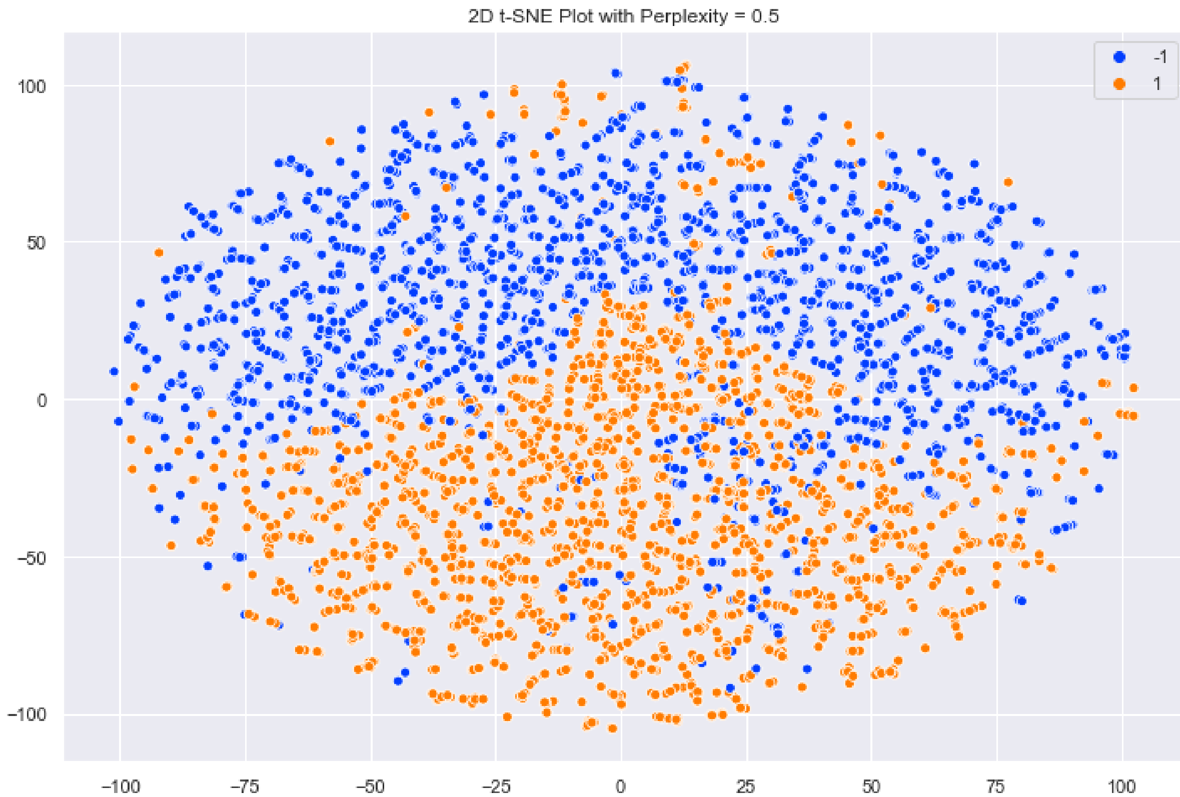
In [32]:
```
# tsne4 - n_components = 2 default, perplexity =  5
startTime4 = time.time()
tsne4 = TSNE(perplexity = 5)
data_embedded = tsne4.fit_transform(data)
# visualization - 2D
sns.scatterplot(data_embedded[:,0], data_embedded[:,1], hue=labels, lege
nd='full', palette=palette)
plt.title('2D t-SNE Plot with Perplexity = 5')
plt.show()
endTime4 = time.time()
```



In [33]:
```
print('Time of execution is ', (endTime4 - startTime4),"second.")
```

```
Time of execution is  265.8814489841461 second.
```

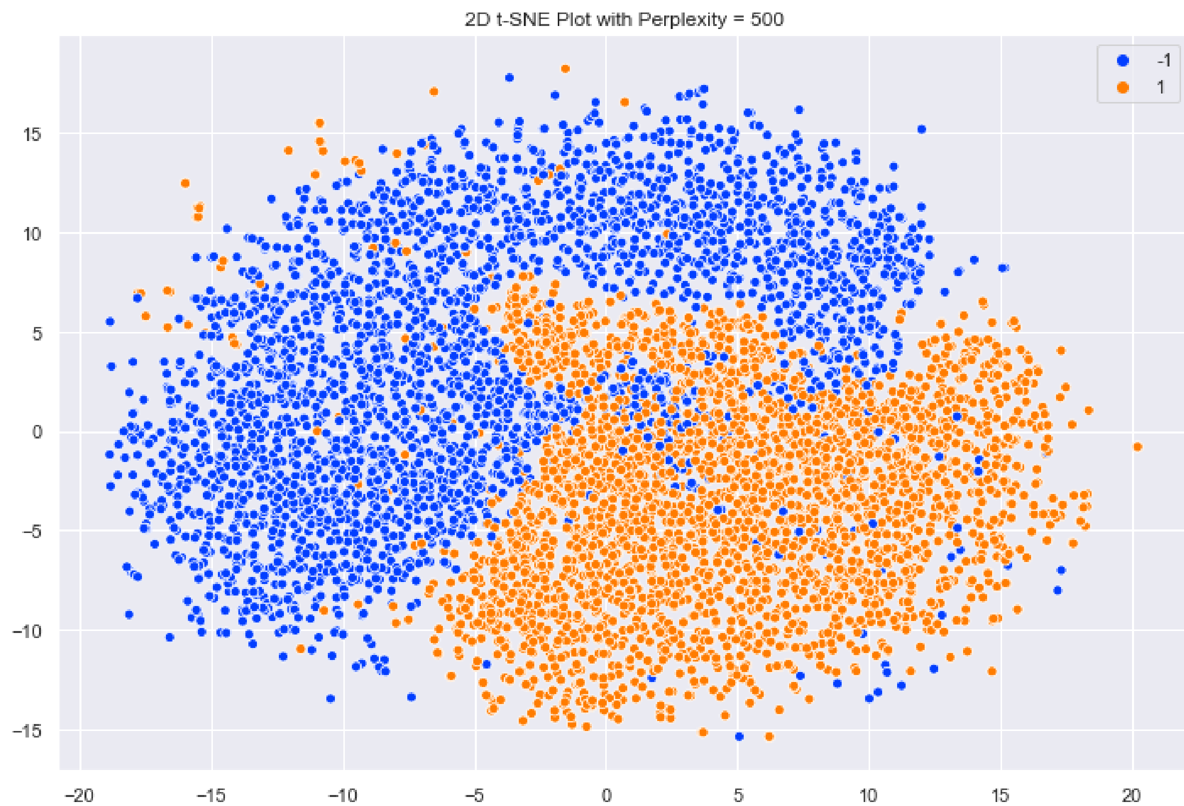
In [34]:
```
# tsne5 - n_components = 2 default, perplexity =  0.5
startTime5 = time.time()
tsne5 = TSNE(perplexity = 0.5)
data_embedded = tsne5.fit_transform(data)
# visualization - 2D
sns.scatterplot(data_embedded[:,0], data_embedded[:,1], hue=labels, lege
nd='full', palette=palette)
plt.title('2D t-SNE Plot with Perplexity = 0.5')
plt.show()
endTime5 = time.time()
```



In [35]:
```
print('Time of execution is ', (endTime5 - startTime5),"second.")
```

```
Time of execution is  263.7677948474884 second.
```

In [36]:
```
# tsne6 - n_components = 2 default, perplexity =  500
startTime6 = time.time()
tsne6 = TSNE(perplexity = 500)
data_embedded = tsne6.fit_transform(data)
# visualization - 2D
sns.scatterplot(data_embedded[:,0], data_embedded[:,1], hue=labels, lege
nd='full', palette=palette)
plt.title('2D t-SNE Plot with Perplexity = 500')
plt.show()
endTime6 = time.time()
```

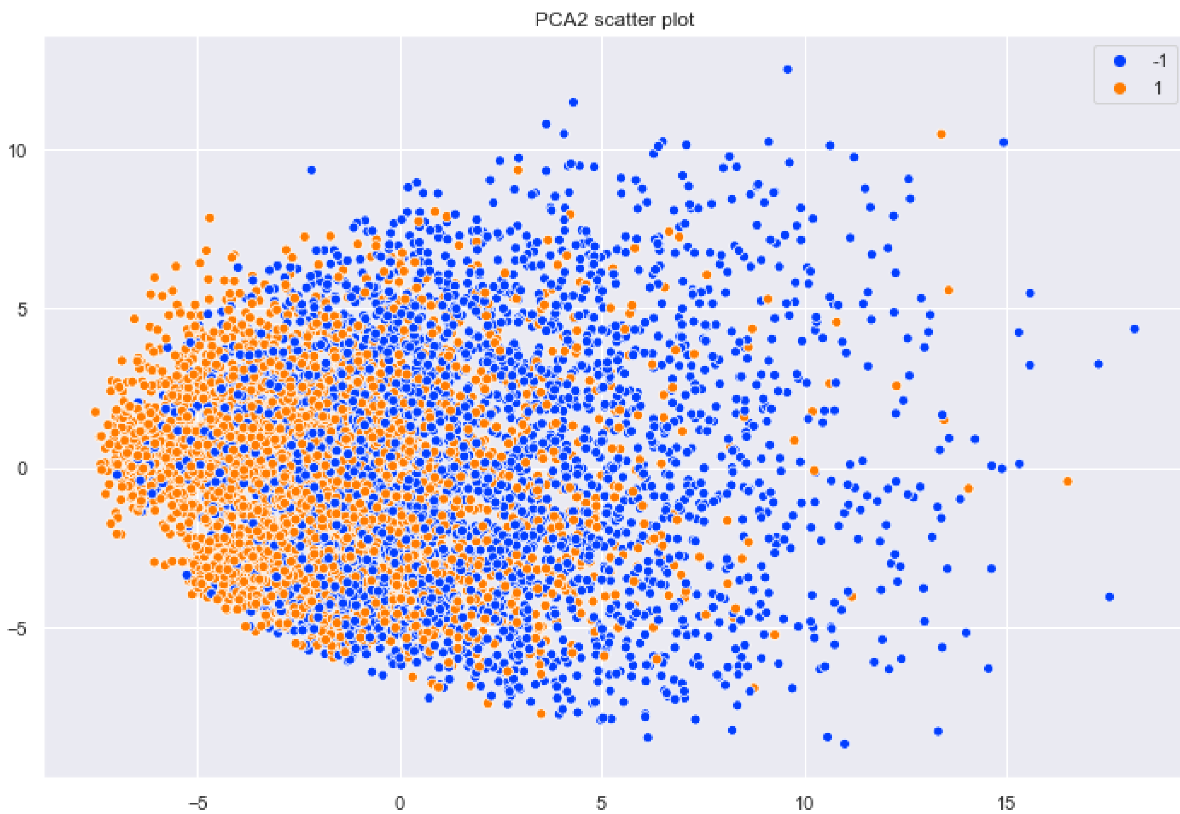


In [37]:
```
print('Time of execution is ', (endTime6 - startTime6),"second.")
```

```
Time of execution is  334.8708908557892 second.
```

PCA

In [41]:
```
from sklearn.decomposition import PCA
pcaTime1 = time.time()
pca2 = PCA(2)
pca2D = pca2.fit(data)
plot_PCA = pca2D.transform(data)
sns.scatterplot(x=plot_PCA[:,0], y=plot_PCA[:,1], hue= labels, legend='f
ull', palette=palette)
plt.title('PCA2 scatter plot')
plt.show()
pcaTime2 = time.time()
```



In [42]:
```
print('Time of PCA2 execution is ', (pcaTime2 - pcaTime1),"second.")
```

```
Time of PCA2 execution is  1.2220909595489502 second.
```

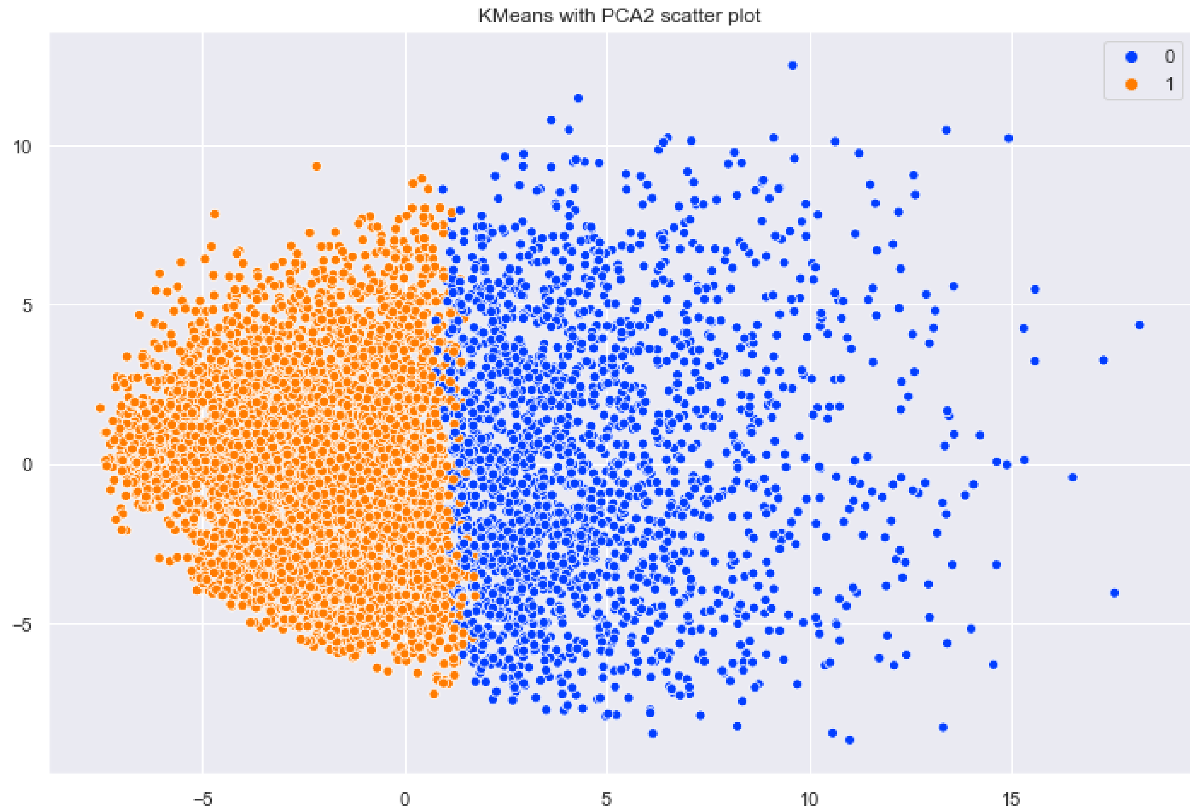
Clustering

KMeans with PCA2

In [44]:
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2)
kmeans_labels = kmeans.fit_predict(data)
kmeans_labels, kmeans_labels.shape
```

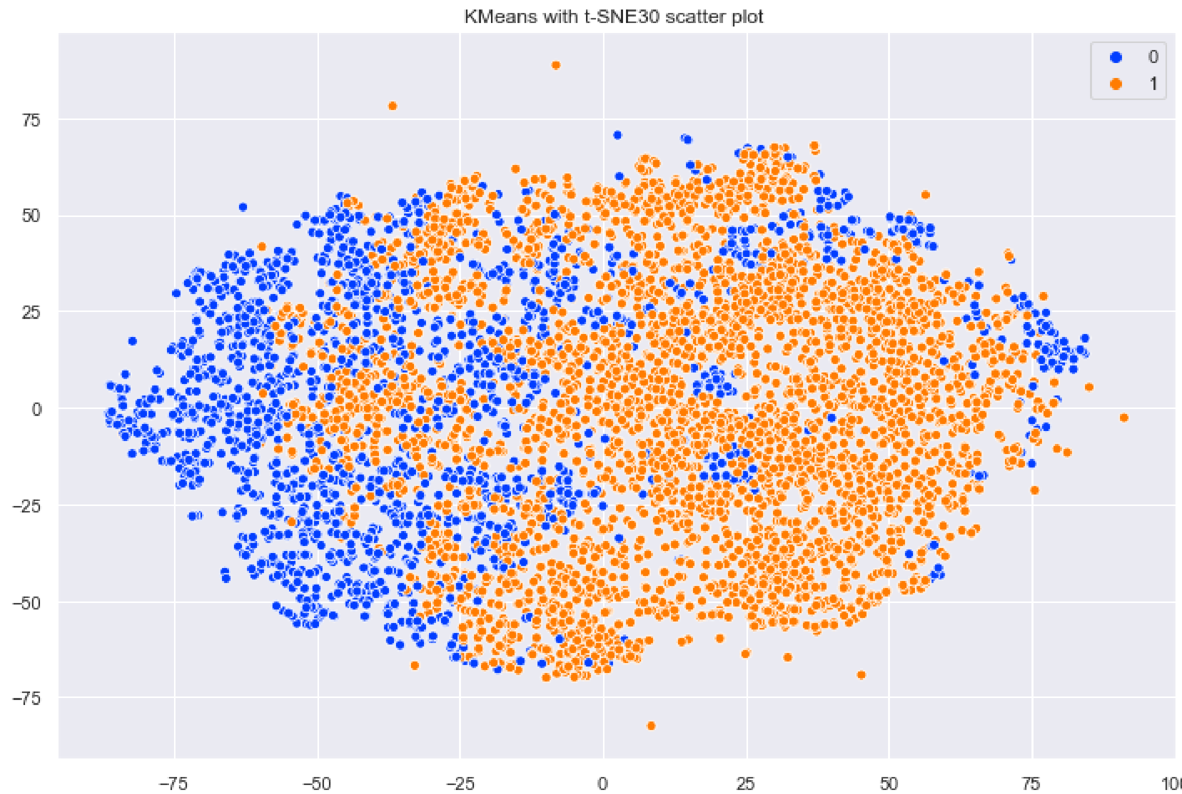
Out[44]:
```
(array([0, 0, 0, ..., 1, 1, 1], dtype=int32), (6000,))
```

In [45]:
```
sns.scatterplot(x=plot_PCA[:,0], y=plot_PCA[:,1], hue= kmeans_labels, le
gend='full', palette=palette)
plt.title('KMeans with PCA2 scatter plot')
plt.show()
```



KMeans with t-SNE (perplexity = 30, default)

In [46]:
```
tsne = TSNE()
data_embedded = tsne.fit_transform(data)
sns.scatterplot(data_embedded[:,0], data_embedded[:,1], hue=kmeans_label
s, legend='full', palette=palette)
plt.title('KMeans with t-SNE30 scatter plot ')
plt.show()
```



In [51]:
```
sns.scatterplot(umap_embedding[:,0], umap_embedding[:,1], hue=kmeans_lab
els, legend='full', palette=palette)
plt.title('KMeans with UMAP Scatter Plot')
plt.show()
```

In [ ]: