Data Visualization

Import

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from scipy import stats
import seaborn as sns
from scipy import linalg
from sklearn import decomposition
from sklearn.manifold import TSNE
import matplotlib.cm as cm
import matplotlib.animation as animation
```

```
!ls
```

```
drive  sample_data  __temp__.mp4
```

Data

```
data = pd.read_csv('/content/drive/MyDrive/AAIC/Datasets/mnist_train.csv')
```

```
print(data.shape)
print(data.columns)
data.head()
```

```
(42000, 785)
Index(['label', 'pixel0', 'pixel1', 'pixel2', 'pixel3', 'pixel4', 'pixel5',
       'pixel6', 'pixel7', 'pixel8',
```

```
X = data.drop('label', axis=1)
X.head()
```

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pi
0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	

5 rows × 784 columns

```
Y = data['label']
Y.head()
```

```
0    1
1    0
2    1
3    4
4    0
Name: label, dtype: int64
```

```
#Plotting

idx = 100
img_data = X.iloc[idx].values.reshape((28,28))

plt.figure(figsize=(7,7))
plt.imshow(img_data, cmap='gray')
plt.show()
```

2D visualization using PCA

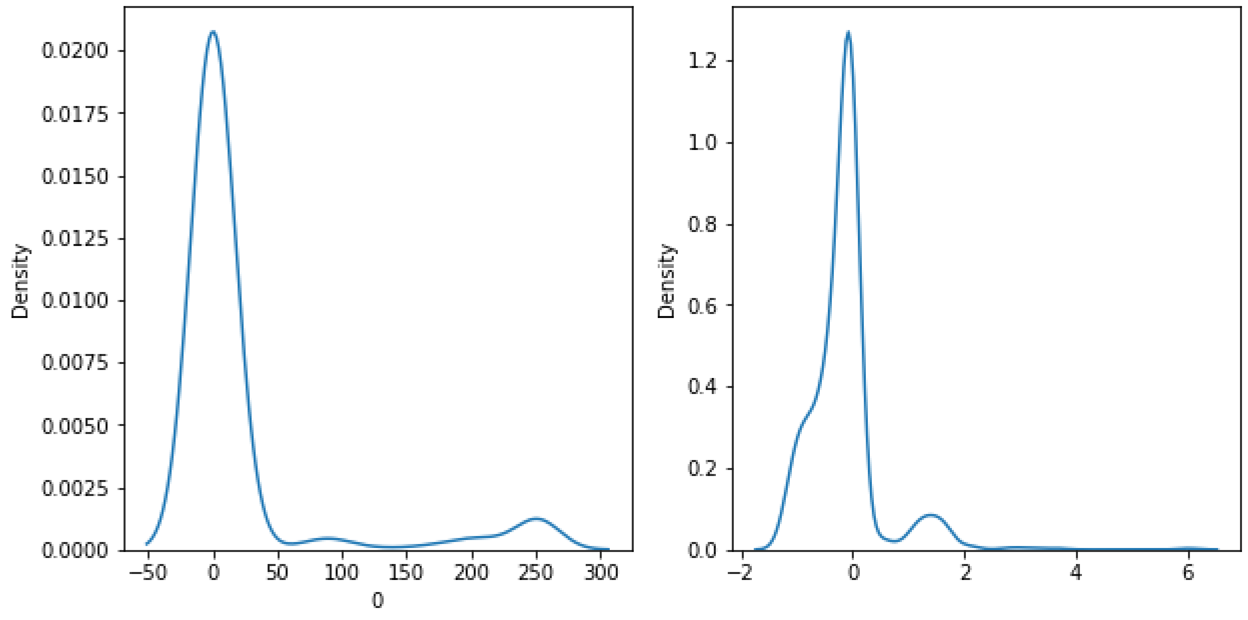
```
data = X.head(15000)
labels = Y.head(15000)
print(data.shape)
```

```
(15000, 784)
```

```
#Data standardization
s_data = StandardScaler().fit_transform(data)
```

```
#checking if the data is standarized or not
dist=getattr(stats, 'norm')
params = dist.fit(s_data[0])
print(params) #mean and std

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 5))
sns.kdeplot(data=data.iloc[0], ax=axes[0])
sns.kdeplot(data=s_data[0], ax=axes[1])
plt.show()
```

```
(-0.14151100239318568, 0.644844270835914)
```



```
cov_mat = np.matmul(s_data.T, s_data)
print(cov_mat.shape)
```

```
(784, 784)
```

```
#find eigen values and vector
values, vectors = linalg.eigh(cov_mat, eigvals=(782, 783)) #get the highest 2 eigen values & v
vectors = vectors.T
print(vectors.shape)

reduced_data = np.matmul(vectors, s_data.T)
print(reduced_data.shape)
```

```
(2, 784)
(2, 15000)
```

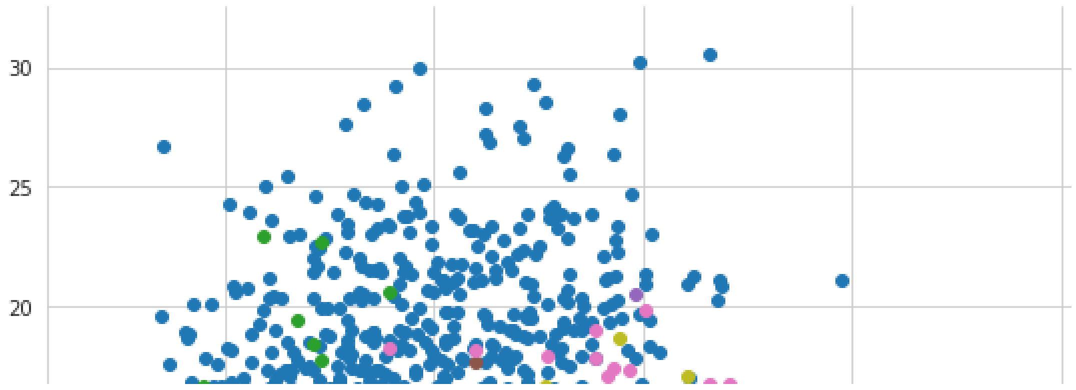
```
final_data = np.vstack((reduced_data, labels)).T
print(final_data.shape)
#convert to dataframe
df = pd.DataFrame(data=final_data, columns=["1st principal", "2nd principal", "Labels"])
df.head(5)
```

```
(15000, 3)
```

	1st principal	2nd principal	Labels
0	-5.558661	-5.043558	1.0
1	6.193635	19.305278	0.0
2	-1.909878	-7.678775	1.0
3	5.525748	-0.464845	4.0
4	6.366527	26.644289	0.0

```
sns.set_style("whitegrid")
sns.FacetGrid(df, hue='Labels', size=8).map(plt.scatter, '1st principal', '2nd principal').add
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:316: UserWarning
  warnings.warn(msg, UserWarning)
```



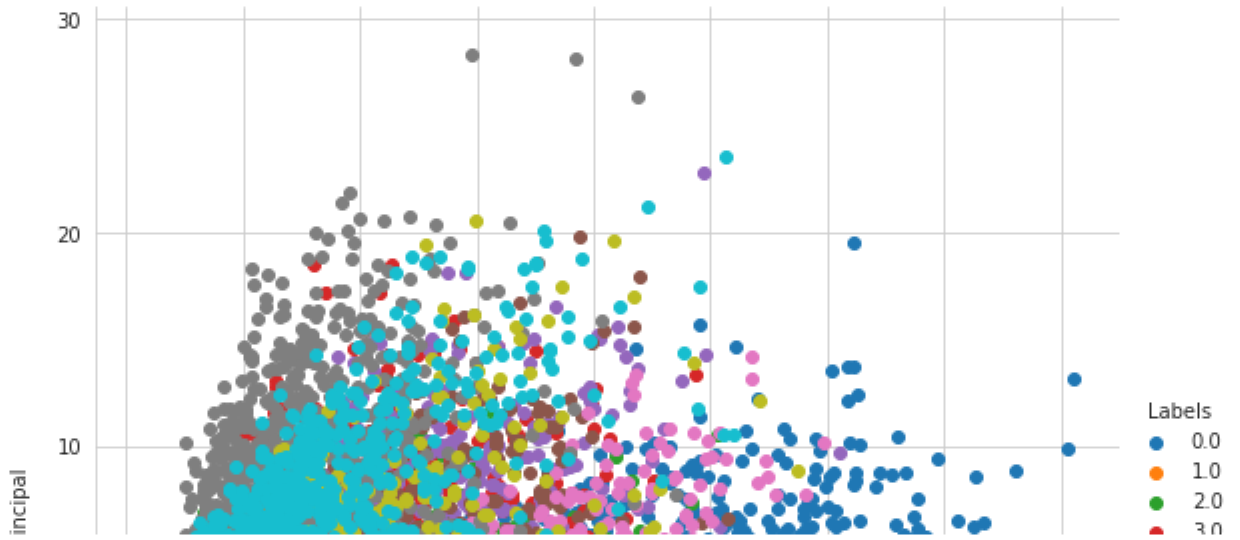
PCA Using sklearn.decomposition (visualization)

```
pca=decomposition.PCA(n_components=2)
pca_data = pca.fit_transform(s_data)
print(pca_data.shape)
print(pca.explained_variance_ratio_)
print(pca.explained_variance_)
print(pca.singular_values_)
```

```
(15000, 2)
[0.05912341 0.04251172]
[40.38397837 29.0374395 ]
[778.27970005 659.94890342]
```

```
print(labels.shape)
final_data = np.vstack((pca_data.T, labels)).T #since the shape of labels is (N,), vstack will
print(final_data.shape)
#convert to dataframe
df = pd.DataFrame(data=final_data, columns=["1st principal", "2nd principal", "Labels"])
sns.set_style("whitegrid")
sns.FacetGrid(df, hue='Labels', size=8).map(plt.scatter, '1st principal', '2nd principal').add
plt.show()
```

```
(15000,)
(15000, 3)
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:316: UserWarning
  warnings.warn(msg, UserWarning)
```



PCA Using sklearn.decomposition (dimensionality reduction)

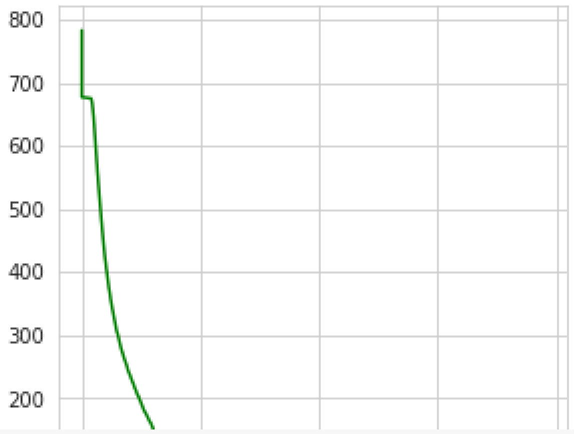
https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

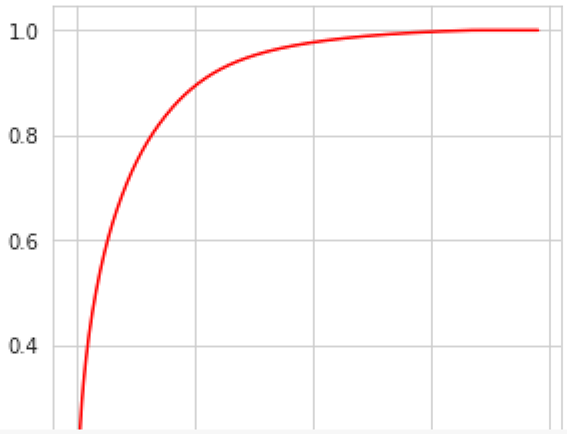
Below result shows that having around 400 components/features will have nearly 95% variance is explained

```
pca = decomposition.PCA()
pca_data = pca.fit_transform(s_data)
```

```
variance = pca.explained_variance_ #all the variance values
var_ratio = pca.explained_variance_ratio_
lambdas = pca.singular_values_
print(lambdas[0:10]) #higher to lower

#variance
print(var_ratio[0:10])
cum_variance = np.cumsum(variance/sum(variance))

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 5))
axes[0].plot(lambdas, np.arange(len(lambdas)) , color='green', label='Eigen values' )
axes[0].set_xlabel('784 features')
axes[1].plot(np.arange(len(var_ratio)), cum_variance , color='red', label='%variance explained
axes[1].set_xlabel('784 features')
plt.legend()

plt.show()
```

```
[778.2797002  659.94890549 637.69285381 561.46247874 520.54447241
 486.19379269 453.08238936 436.47978024 408.30944097 386.84406713]
[0.05912341 0.04251172 0.03969275 0.03077014 0.02644866 0.02307315
 0.02003745 0.01859586 0.01627297 0.01460696]
```





```
#code to get no of features to have 95% of variance explained
variance = pca.explained_variance_
cum_sum = np.cumsum(variance/np.sum(variance))
print(np.min(np.where(cum_sum>0.95)))
```

```
296
```

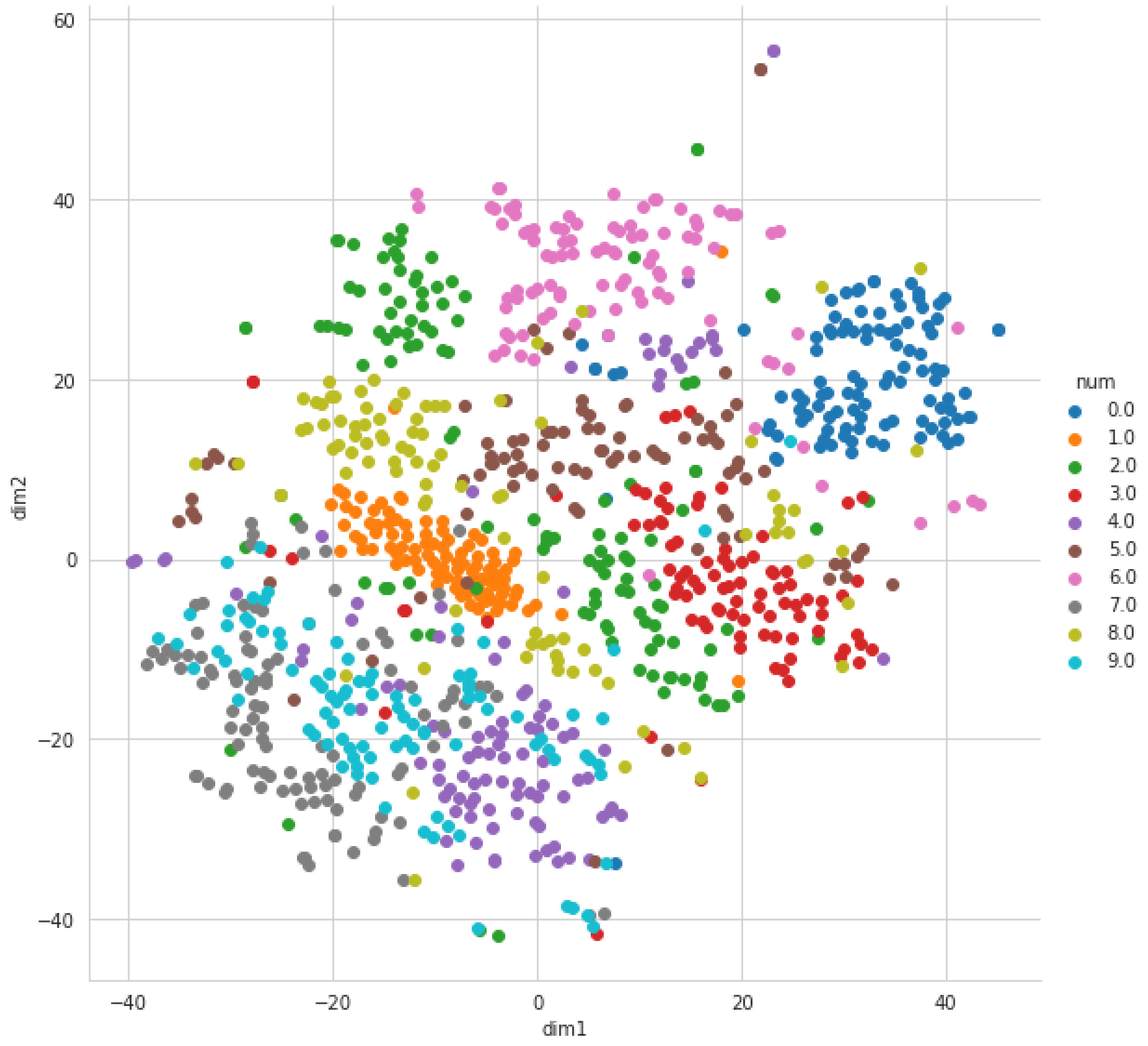
t-SNE

```
s1000 = s_data[0:1000,]
l1000 = labels[0:1000]
```

```
def do_tsne(s, l, perp=30, iter=1000):
  model = TSNE(n_components=2, perplexity=perp, random_state=0, n_iter=iter)
  tsne_data = model.fit_transform(s)
  tsne_data = np.vstack((tsne_data.T, l)).T
  df = pd.DataFrame(data = tsne_data, columns=["dim1", "dim2", "num"])
  sns.set_style('whitegrid')
  sns.FacetGrid(df, hue='num', size=8).map(plt.scatter, "dim1", "dim2").add_legend()
  plt.show()
```

```
do_tsne(s1000, l1000, 50, 5000)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:316: UserWarning: The `size` p
  warnings.warn(msg, UserWarning)
```



✓ 36s completed at 11:10 PM