Load MNIST data

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

df = pd.read_csv('MNIST_train.csv')

print(df.head())
```

```
   label  pixel0  pixel1  pixel2  pixel3  pixel4  pixel5  pixel6  pixel7  \
0      1       0       0       0       0       0       0       0       0
1      0       0       0       0       0       0       0       0       0
2      1       0       0       0       0       0       0       0       0
3      4       0       0       0       0       0       0       0       0
4      0       0       0       0       0       0       0       0       0

   pixel8  ...  pixel774  pixel775  pixel776  pixel777  pixel778  pixel779  \
0       0  ...         0         0         0         0         0         0
1       0  ...         0         0         0         0         0         0
2       0  ...         0         0         0         0         0         0
3       0  ...         0         0         0         0         0         0
4       0  ...         0         0         0         0         0         0

   pixel780  pixel781  pixel782  pixel783
0         0         0         0         0
1         0         0         0         0
2         0         0         0         0
3         0         0         0         0
4         0         0         0         0

[5 rows x 785 columns]
```

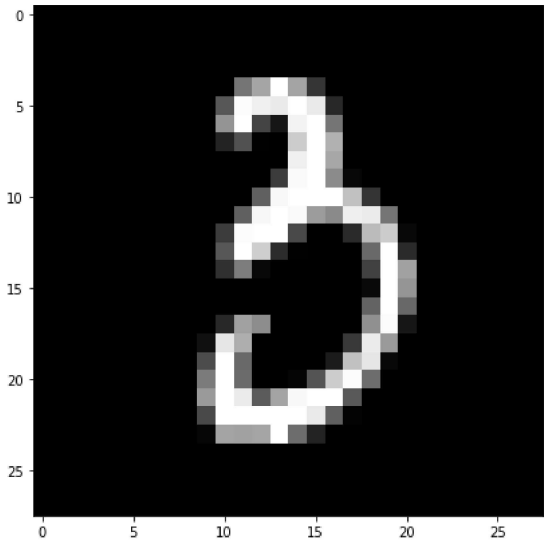
```
label = df['label']
data = df.drop('label', axis=1)
```

```
data.shape,label.shape
```

```
((42000, 784), (42000,))
```

```
### Sanity Check ###
plt.figure(figsize=(7,7))
idx= 150

image_150 = data.iloc[idx].values.reshape(28,28)
plt.imshow(image_150, interpolation ='none', cmap='gray')
plt.show()
label[150]
```




```
3
```

Visualization using PCA

```
labels = label.head(15000)
data = data.head(15000)
print("the shape of sample data = ", data.shape)
```

```
the shape of sample data =  (15000, 784)
```

```
from sklearn.preprocessing import StandardScaler
standardized_data = StandardScaler().fit_transform(data)
print(standardized_data.shape)
```

```
(15000, 784)
```

Find out why this method is not working

mean = data.mean() std = data.std() df_standardized = data.apply(lambda x: (x -mean)/ std) df_standardized.head()

```
#finding covariance mar=triz

sample_data = standardized_data
cov_matrix = np.matmul(sample_data.T, sample_data)
cov_matrix.shape
```

```
(784, 784)
```

```
 from scipy.linalg import eigh
 #TAKING TOP TWO EIGH ONLY
 values, vectors = eigh(cov_matrix, eigvals = (782, 783))  ### top two eigvalues only calculated
 vectors.shape, vectors.T.shape
```

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

```
vectors = vectors.T
```

```
new_coordinates = np.matmul(vectors, sample_data.T)
vectors.shape, sample_data.shape
new_coordinates.shape  ## Represents projected 15,000 input data points on top two eigen vectors
```

```
(2, 15000)
```

```
new_coordinates.shape, labels.shape
```

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

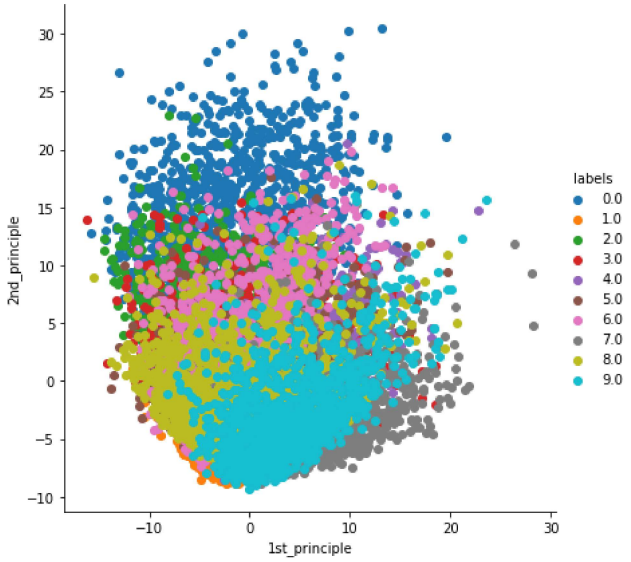
```
 # Data massaging

 new_coordinates = np.vstack((new_coordinates, labels)).T    #Note do not execute this cell again and again will cause error unnecessarily.
 dataframe = pd.DataFrame(data = new_coordinates, columns=('1st_principle', '2nd_principle', 'labels'))
 dataframe.head() ##Transformed data i.e projected data points
```

	1st_principle	2nd_principle	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

```
import seaborn as sns
sns.FacetGrid(dataframe, hue= 'labels', size=6).map(plt.scatter,
                                                    '1st_principle',
                                                    '2nd_principle').add_legend()
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/axisgrid.py:337: UserWarning: The `size` parameter h
  warnings.warn(msg, UserWarning)
```



PCA using Scikit-learn

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

```
pca.n_components = 2
pca_data = pca.fit_transform(sample_data)

pca_data.shape
```

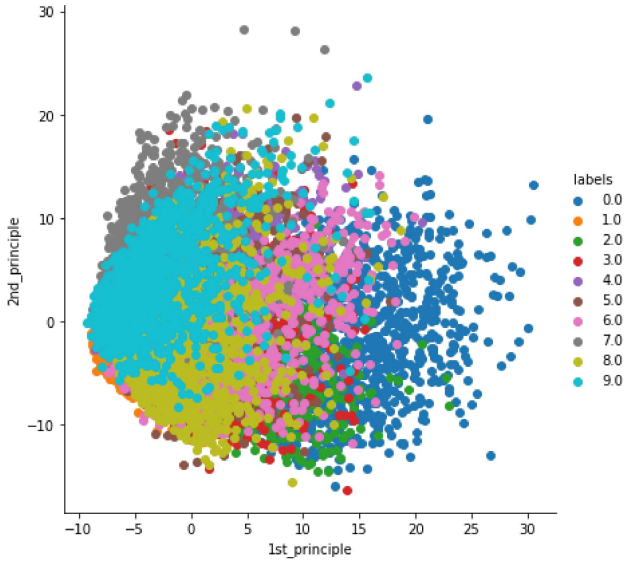
```
(15000, 2)
```

```
pca_data = np.vstack((pca_data.T, labels)).T
pca_df = pd.DataFrame(data=pca_data, columns=('1st_principle', '2nd_principle', 'labels'))
pca_df.head()
```

	1st_principle	2nd_principle	labels
0	-5.043562	-5.558364	1.0
1	19.305334	6.192572	0.0
2	-7.678767	-1.910249	1.0
3	-0.464817	5.525317	4.0
4	26.644314	6.366089	0.0

```
sns.FacetGrid(pca_df, hue='labels', size=6).map(plt.scatter,
                                                '1st_principle',
                                                '2nd_principle').add_legend()
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/axisgrid.py:337: UserWarning: The `size` parameter h
  warnings.warn(msg, UserWarning)
```



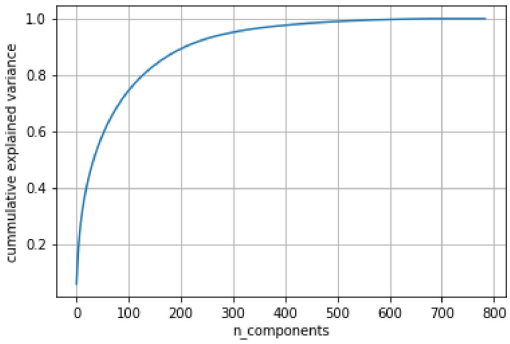
PCA FOR DIMENTIONALITY REDUCTION ---> Checking what numbers of eigen vectors(Principle components) are required w.r.t data variance(Percentage of information to keep while performing dimentionality reduction)

```
pca.n_components = 784
pca_data = pca.fit_transform(sample_data)

percentage_var_explained = pca.explained_variance_/np.sum(pca.explained_variance_)

cum_var_explained = np.cumsum(percentage_var_explained)   #cummulative sum of variance ratios

plt.plot(cum_var_explained)
plt.grid()
plt.ylabel('cummulative explained variance')
plt.xlabel('n_components')
plt.show()
percentage_var_explained.shape, cum_var_explained.shape
```




```
((784,), (784,))
```