Implementation Of PCA On Handwriting Digits Dataset

- We will make two CNN models to see the effect of before PCA data and after PCA data.

You can also load data from keras datasets

PART 1

Importing Libraries

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from keras.utils import to_categorical
```

Reading Dataset

In [2]:

```
df = pd.read_csv("C:/Users/Roshan Salunke/Downloads/Data Science Course/train.csv")
```

In [3]:

```
df.head()
```

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778
0	1	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
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
3	4	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

5 rows × 785 columns

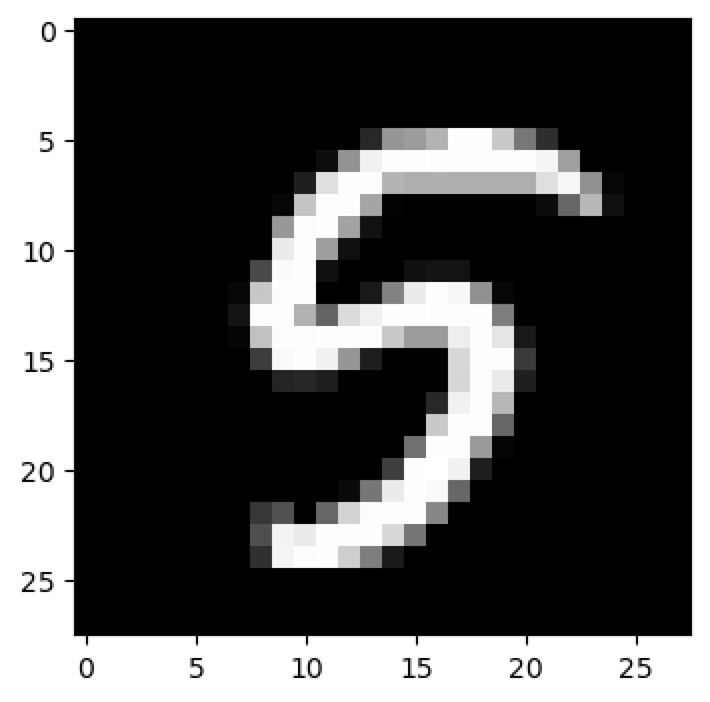
Separating x and y

In [4]:

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

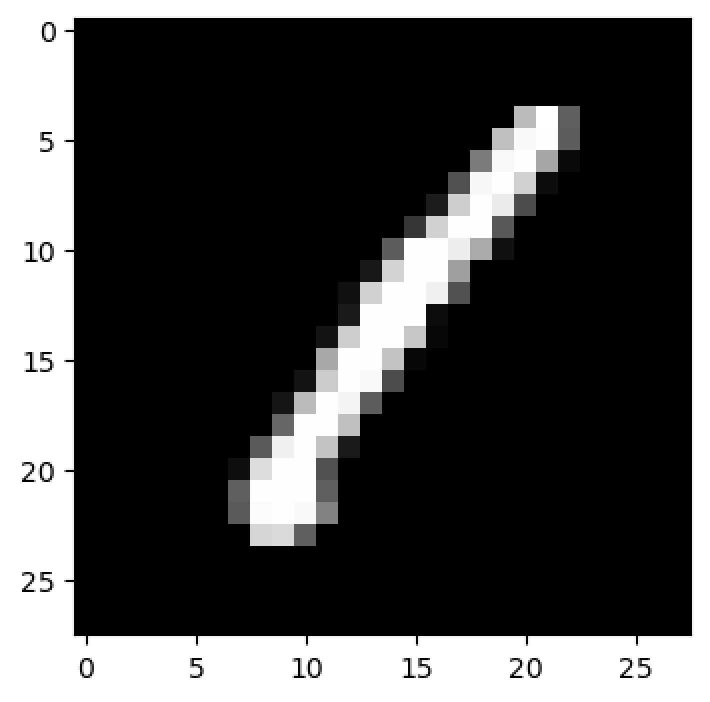
In [5]:

```python
plt.figure(figsize=(4,4))
grid_data = x.loc[8].values.reshape(28,28)
plt.imshow(grid_data, interpolation='none',cmap='gray')
plt.show()
```



In [6]:

```
plt.figure(figsize=(4,4))
grid_data = x.loc[0].values.reshape(28,28)
plt.imshow(grid_data, interpolation='none',cmap='gray')
plt.show()
```



In [7]:

```
x.shape
```

```
(42000, 784)
```

In [46]:
```
x = np.array(x) # converting x into an array
```

In [47]:
```
x = np.reshape(x,(x.shape[0],28,28)) # reshaping x
```

In [10]:
```
x = np.reshape(x, (x.shape[0],28,28,1))
x.shape
```

```
(42000, 28, 28, 1)
```

In [48]:
```
y = to_categorical(y) # converting y into categorical using one hot encoding
y.shape
```

```
(42000, 10, 2)
```

Bulding 1st Model

Splitting dataset into training and testing

In [12]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=.3,random_state=42)
```

In [13]:
```
x_train.shape, y_train.shape
```

```
((29400, 28, 28, 1), (29400, 10))
```

In [14]:
```
classes = np.unique(y)
len(classes)
```

```
2
```

In [15]:
```
import tensorflow as tf
from tensorflow import keras
from keras.layers import Conv2D, MaxPool2D, Dropout, Dense, BatchNormalization, Flatten,Conv1D
from keras.models import Sequential
```

In [16]:
```
model = Sequential()
model.add(Conv2D(32, 3, input_shape=(28,28,1),padding='same',activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(64,3, activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Flatten())
model.add(Dense(1024, activation='relu'))

model.add(Dense(10, activation='softmax'))
```

In [17]:
```
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

In [18]:

```
fit = model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test), batch_size=(128))
```

```
Epoch 1/5
230/230 [==============================] - 26s 107ms/step - loss: 3.3551 - accuracy: 0.8993 - val_loss: 0.0933 - val_accuracy: 0.9694
Epoch 2/5
230/230 [==============================] - 24s 104ms/step - loss: 0.0615 - accuracy: 0.9804 - val_loss: 0.0851 - val_accuracy: 0.9728
Epoch 3/5
230/230 [==============================] - 24s 106ms/step - loss: 0.0369 - accuracy: 0.9879 - val_loss: 0.0792 - val_accuracy: 0.9772
Epoch 4/5
230/230 [==============================] - 24s 106ms/step - loss: 0.0223 - accuracy: 0.9928 - val_loss: 0.0688 - val_accuracy: 0.9810
Epoch 5/5
230/230 [==============================] - 24s 105ms/step - loss: 0.0113 - accuracy: 0.9965 - val_loss: 0.0678 - val_accuracy: 0.9814
```

Our first model is giving an accuracy of around 98% which is really good.

In [ ]:

```

```

PART 2

In [19]:

```
df = pd.read_csv("C:/Users/Roshan Salunke/Downloads/Data Science Course/train.csv")
```

In [20]:

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

In [21]:

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

In [22]:
```
x = sc.fit_transform(x)
```

In [23]:
```
x.shape
```

```
(42000, 784)
```

In [24]:
```
sample_data = x
```

Implementation of PCA

In [25]:
```
from sklearn.decomposition import PCA
```

In [26]:
```
pca_data = PCA(n_components=2).fit_transform(sample_data)
```

In [27]:
```
pca_data = np.vstack((pca_data.T, y)).T
```

In [28]:
```
pca_data[0]
```

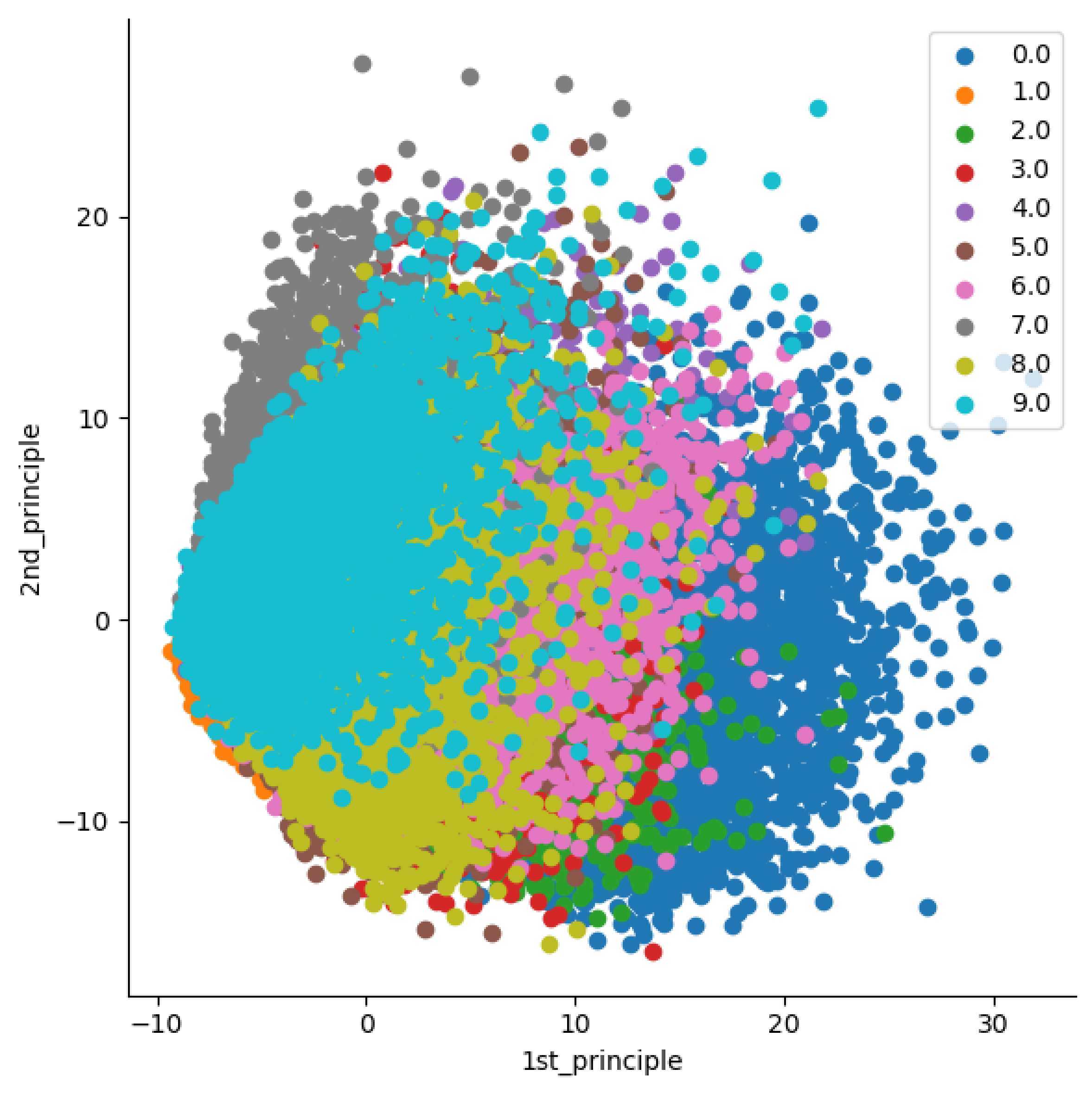
```
array([-5.14053799, -5.22693201,  1.        ])
```

In [29]:

```
pca_df = pd.DataFrame(data=pca_data, columns=('1st_principle', '2nd_principle', 'label'))

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

```
C:\Users\Roshan Salunke\anaconda3\lib\site-packages\seaborn\axisgrid.py:337: UserWarning: The `size` parameter has been renamed to `height
`; please update your code.
  warnings.warn(msg, UserWarning)
```


● 0.0
● 1.0
● 2.0
● 3.0
● 4.0
● 5.0
● 6.0
● 7.0
● 8.0
● 9.0
20
10
0
−10
2nd_principle
−10
0
10
20
30
1st_principle

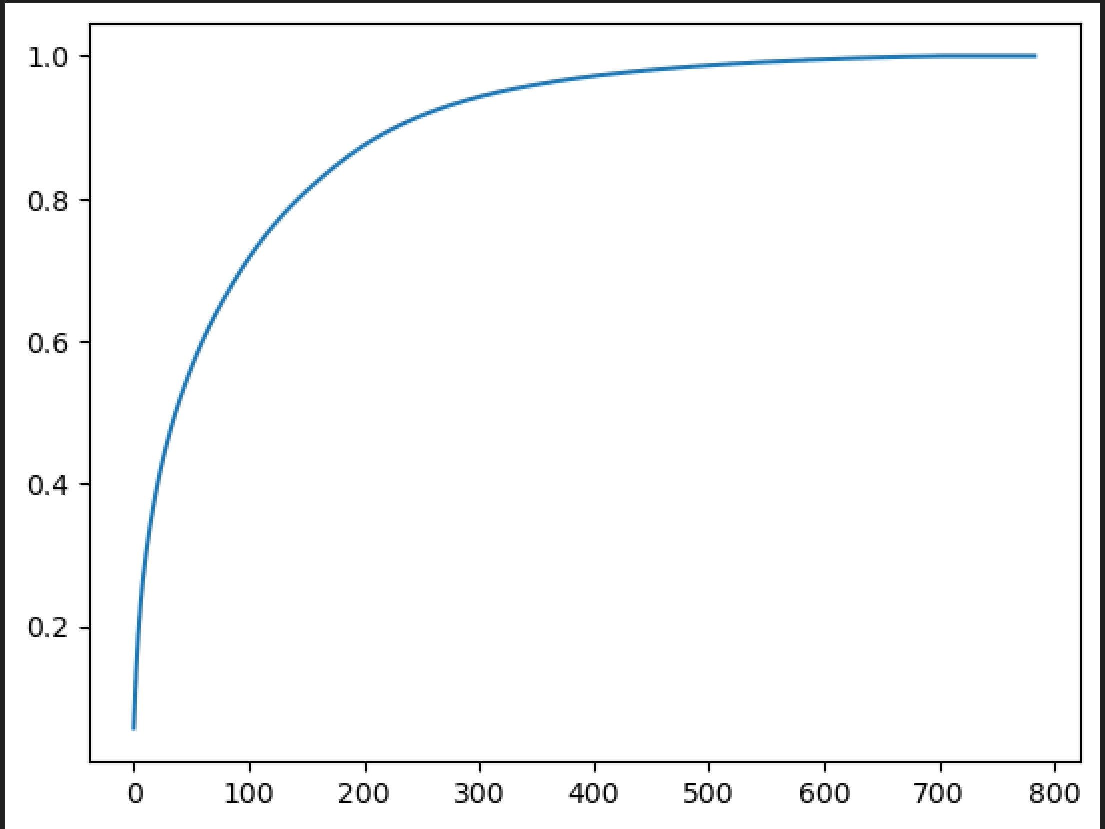
In [30]:

```
pca = PCA(n_components=2)
```

In [31]:

```
pca.n_components = 784
pca_data = pca.fit_transform(sample_data)
variance = pca.explained_variance_/sum(pca.explained_variance_)
cumsum = np.cumsum(variance)
plt.plot(cumsum)
```

```
[<matplotlib.lines.Line2D at 0x26ea0ba4460>]
```



In [32]:

```
pca_data.shape
```

```
(42000, 784)
```

In [33]:

```
x = pca_data[:,:2]
```

In [34]:

```
x.shape,y.shape
```

```
((42000, 2), (42000,))
```

In [35]:

```
x = np.repeat(x, 392,axis=1)
```

In [36]:

```
x.shape
```

```
(42000, 784)
```

In [37]:

```
x = np.reshape(x, (x.shape[0],28,28))
x.shape
```

```
(42000, 28, 28)
```

In [38]:

```
x = np.reshape(x, (x.shape[0],28,28,1))
```

In [39]:

```
x.shape
```

```
(42000, 28, 28, 1)
```

In [40]:

```
y = to_categorical(y)
y.shape
```

```
(42000, 10)
```

In [41]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=.3,random_state=42)
```

In [42]:

```
x_train.shape, y_train.shape
```

```
((29400, 28, 28, 1), (29400, 10))
```

Building 2nd Model

In [43]:

```
model = Sequential()
model.add(Conv2D(32, 3, input_shape=(28,28,1),padding='same',activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(64,3, activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Flatten())
model.add(Dense(1024, activation='relu'))

model.add(Dense(10, activation='softmax'))
```

In [44]:

```
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

In [45]:

```
fit = model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test), batch_size=(128))
```

```
Epoch 1/5
230/230 [==============================] - 26s 108ms/step - loss: 1.7340 - accuracy: 0.3306 - val_loss: 1.6503 - val_accuracy: 0.3568
Epoch 2/5
230/230 [==============================] - 25s 107ms/step - loss: 1.6468 - accuracy: 0.3617 - val_loss: 1.6381 - val_accuracy: 0.3608
Epoch 3/5
230/230 [==============================] - 24s 106ms/step - loss: 1.6394 - accuracy: 0.3680 - val_loss: 1.6247 - val_accuracy: 0.3683
Epoch 4/5
230/230 [==============================] - 24s 106ms/step - loss: 1.6330 - accuracy: 0.3688 - val_loss: 1.6261 - val_accuracy: 0.3581
Epoch 5/5
230/230 [==============================] - 24s 105ms/step - loss: 1.6263 - accuracy: 0.3692 - val_loss: 1.6187 - val_accuracy: 0.3694
```

Conclusion

I selected 2 pca components, you can increase this number. After pca we lost a lot of data and accuracy of the model also decreased.

Its important to select right number of pca components.

In [ ]:

```

```