PCA

▼ Manual PCA

Imports

```
1  import numpy as np
2  import pandas as pd
3  import matplotlib.pyplot as plt
4  import seaborn as sns
```

✓ 0.2s

```
1  df = pd.read_csv('cancer_tumor_data_features.csv')
```

✓ 0.1s

```
1  df
```

✓ 0.2s

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883

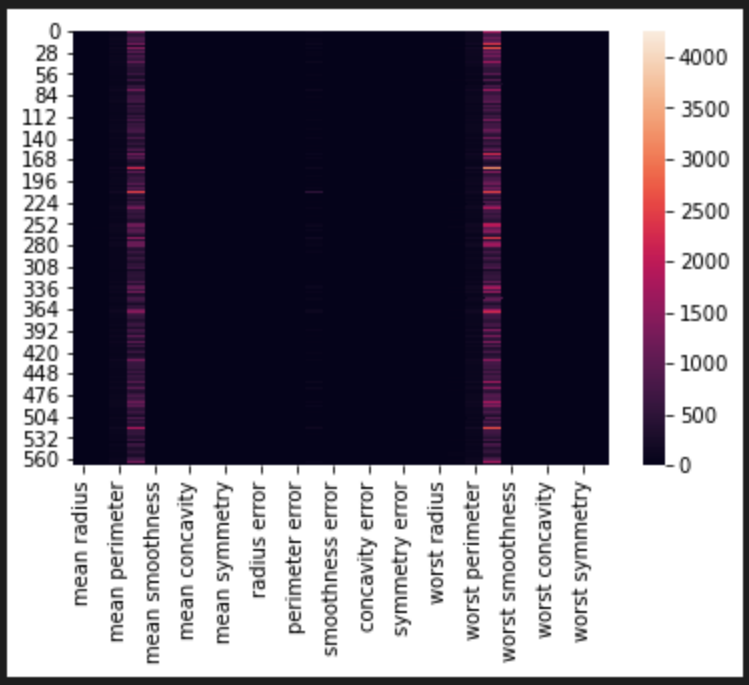
Manual Construction of PCA

Scaling Data

```
1  sns.heatmap(df)
2  # difficult to understand
```

✓ 1.2s

```
<AxesSubplot:>
```




```
1  from sklearn.preprocessing import StandardScaler
```

✓ 0.1s

```
1  scaler = StandardScaler()
```

✓ 0.1s

```
1  scaled_X = scaler.fit_transform(df)
```
✓ 0.1s

```
1  scaled_X.mean()
```
✓ 0.1s

```
-6.826538293184326e-17
```

```
1  # Grab Covariance Matrix
2  covariance_matrix = np.cov(scaled_X, rowvar=False)
```
✓ 0.1s

```
1  eigen_values, eigen_vectors = np.linalg.eig(covariance_matrix)
```
✓ 0.1s

```
1  # Choose some number of components
2  num_components=2
```
✓ 0.8s

```
1  # index [0,1,2]
2  np.argsort([2,1,3])
3  # sorted the index not the values
```
✓ 0.1s

```
array([1, 0, 2], dtype=int64)
```

```
1  sorted_key = np.argsort(eigen_values)[::-1][:num_components]
```
✓ 0.1s

```
1  eigen_values, eigen_vectors = eigen_vectors[sorted_key], eigen_vectors[:,sorted_key]
```
✓ 0.1s

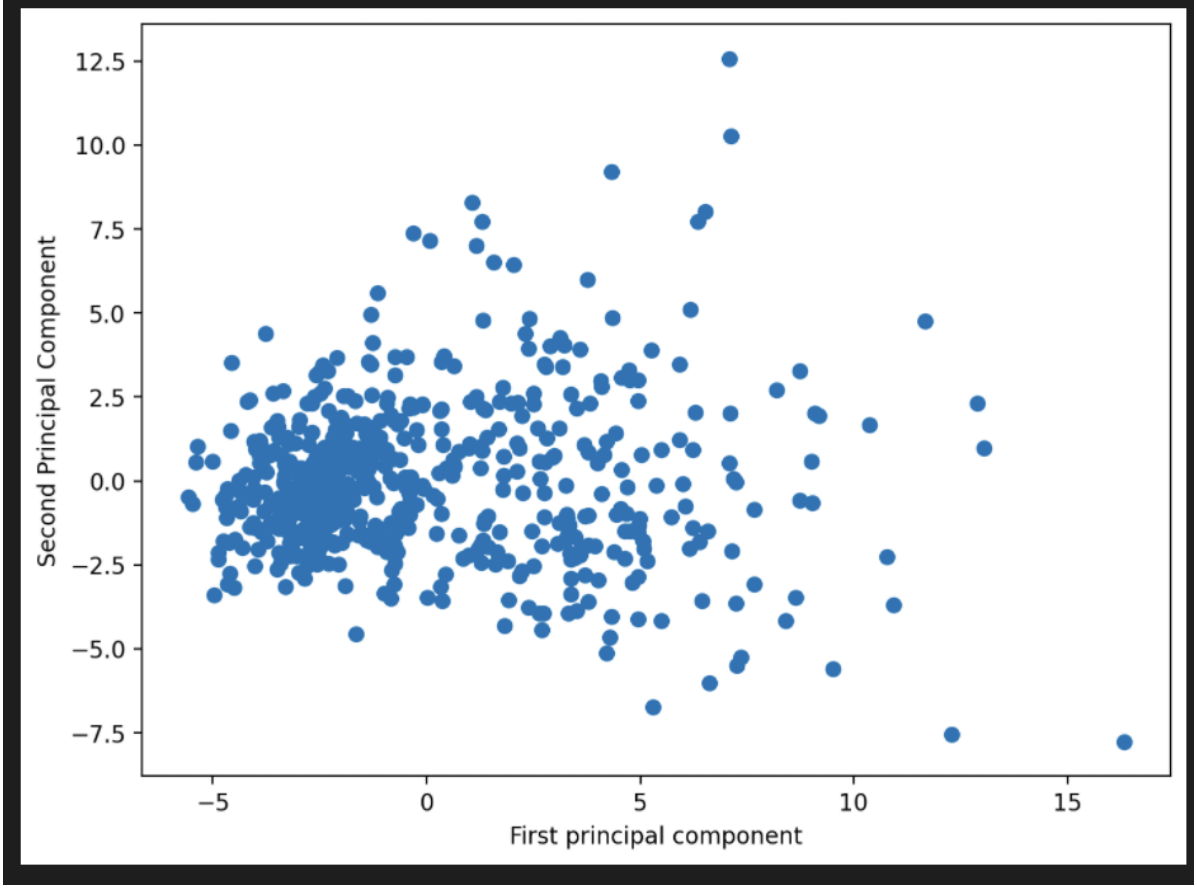
```
1  principal_components = np.dot(scaled_X, eigen_vectors)
2  principal_components
```
✓ 0.1s

```
array([[ 9.19283683,  1.94858307],
       [ 2.3878018 , -3.76817174],
       [ 5.73389628, -1.0751738 ],
       ...,
       [ 1.25617928, -1.90229671],
       [10.37479406,  1.67201011],
       [-5.4752433 , -0.67063679]])
```

```
1  plt.figure(figsize=(8,6), dpi=200)
2  plt.scatter(principal_components[:,0],principal_components[:,1])
3  plt.xlabel('First principal component')
4  plt.ylabel('Second Principal Component')
```

✓ 0.4s

```
Text(0, 0.5, 'Second Principal Component')
```




```
1  from sklearn.datasets import load_breast_cancer
```

✓ 0.8s

```
1  cancer_dictionary = load_breast_cancer()
```

✓ 0.1s

```
1  cancer_dictionary.keys()
```

✓ 0.1s

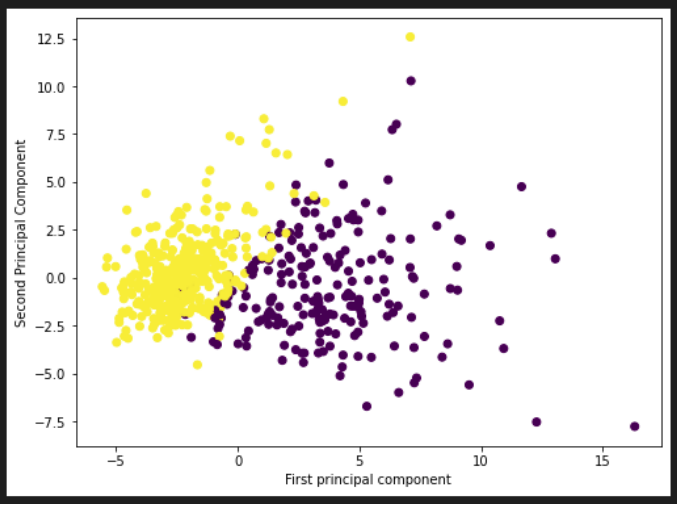
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename'])
```

```
1  print(cancer_dictionary["DESCR"])
```
✓ 0.1s

```
Output exceeds the size limit. Open the full output data in a text editor
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
...
     July-August 1995.
   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques
     to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994)
     163-171.
```

```
1  plt.figure(figsize=(8,6))
2  plt.scatter(principal_components[:,0],principal_components[:,1],c=cancer_dictionary['target'])
3  plt.xlabel('First principal component')
4  plt.ylabel('Second Principal Component')
```
✓ 0.7s

```
Text(0, 0.5, 'Second Principal Component')
```



▼ SKLearn PCA

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

✓ 7.1s

```python
df = pd.read_csv('cancer_tumor_data_features.csv')
```

✓ 0.1s

```python
from sklearn.preprocessing import StandardScaler
```

✓ 0.1s

```python
scaler = StandardScaler()
scaled_X = scaler.fit_transform(df)
scaled_X
```

✓ 0.2s

```
array([[ 1.09706398, -2.07333501,  1.26993369, ...,  2.29607613,
         2.75062224,  1.93701461],
       [ 1.82982061, -0.35363241,  1.68595471, ...,  1.0870843 ,
        -0.24388967,  0.28118999],
       [ 1.57988811,  0.45618695,  1.56650313, ...,  1.95500035,
         1.152255  ,  0.20139121],
```

```
1 from sklearn.decomposition import PCA
```
✓ 0.1s

```
1 pca_model = PCA(n_components=2)
```
✓ 0.1s

```
1 pca_model.fit(scaled_X)
```
✓ 0.2s

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

```
1 pca_model.transform(scaled_X)
```
✓ 0.1s

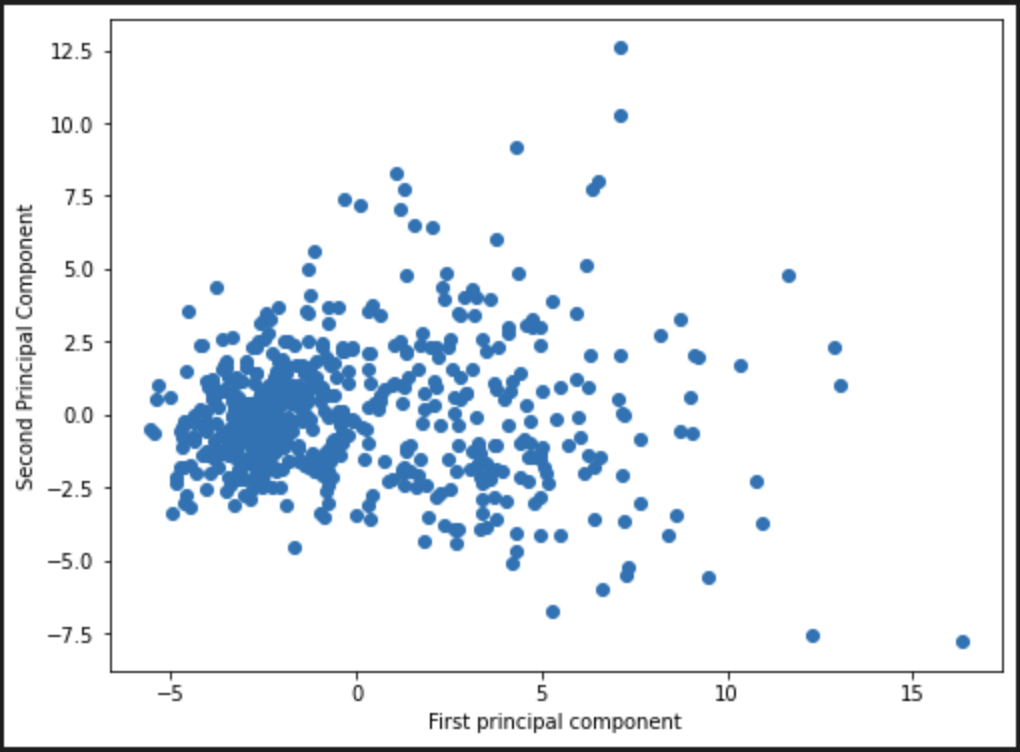
```
array([[ 9.19283683,  1.94858307],
       [ 2.3878018 , -3.76817174],
       [ 5.73389628, -1.0751738 ],
       ...,
       [ 1.25617928, -1.90229671],
       [10.37479406,  1.67201011],
       [-5.4752433 , -0.67063679]])
```

```
1 pc_results = pca_model.fit_transform(scaled_X)
```
✓ 0.1s

```
1  plt.figure(figsize=(8,6))
2  plt.scatter(pc_results[:,0],pc_results[:,1])
3  plt.xlabel('First principal component')
4  plt.ylabel('Second Principal Component')
```

✓ 0.3s

```
Text(0, 0.5, 'Second Principal Component')
```

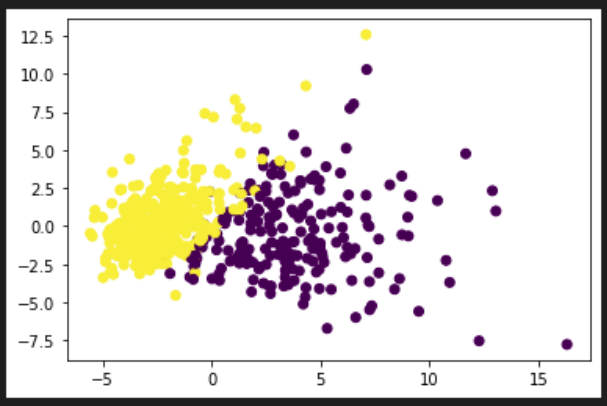


```
1  from sklearn.datasets import load_breast_cancer
```
✓ 0.1s

```
1  cancer_dictionary = load_breast_cancer()
```
✓ 0.8s

```
1  plt.scatter(pc_results[:,0],pc_results[:,1], c=cancer_dictionary["target"])
```
✓ 0.3s

```
<matplotlib.collections.PathCollection at 0x2ceb3b2aee0>
```



```
1  pca_model.n_components
```
✓ 0.9s

```
2
```

```
1  pca_model.components_
```
✓ 0.1s

```
array([[ 0.21890244,  0.10372458,  0.22753729,  0.22099499,  0.14258969,
         0.23928535,  0.25840048,  0.26085376,  0.13816696,  0.06436335,
         0.20597878,  0.01742803,  0.21132592,  0.20286964,  0.01453145,
         0.17039345,  0.15358979,  0.1834174 ,  0.04249842,  0.10256832,
         0.22799663,  0.10446933,  0.23663968,  0.22487053,  0.12795256,
         0.21009588,  0.22876753,  0.25088597,  0.12290456,  0.13178394],
       [-0.23385713, -0.05970609, -0.21518136, -0.23107671,  0.18611302,
         0.15189161,  0.06016536, -0.0347675 ,  0.19034877,  0.36657547,
        -0.10555215,  0.08997968, -0.08945723, -0.15229263,  0.20443045,
         0.2327159 ,  0.19720728,  0.13032156,  0.183848  ,  0.28009203,
        -0.21986638, -0.0454673 , -0.19987843, -0.21935186,  0.17230435,
         0.14359317,  0.09796411, -0.00825724,  0.14188335,  0.27533947]])
```

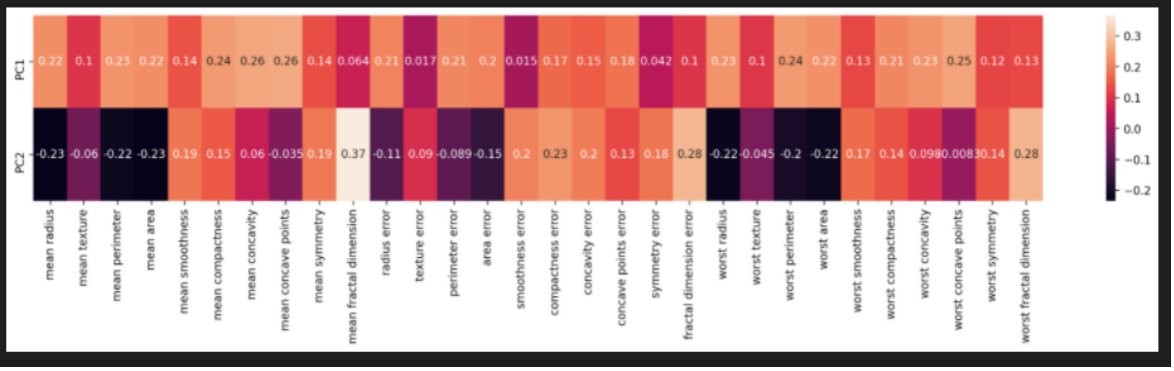
```
1  df_comp = pd.DataFrame(pca_model.components_, index=['PC1','PC2'],columns=df.columns)
```
✓ 0.8s

```
1  df_comp
```
✓ 0.1s

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
PC1	0.218902	0.103725	0.227537	0.220995	0.142590	0.239285	0.258400	0.260854	0.138167
PC2	-0.233857	-0.059706	-0.215181	-0.231077	0.186113	0.151892	0.060165	-0.034768	0.190349

```
1  plt.figure(figsize=(20,3),dpi=200)
2  sns.heatmap(df_comp, annot=True)
```
✓ 1.8s Python

```
<AxesSubplot:>
```



```
1  pca_model.explained_variance_ratio_
```

✓ 0.1s

```
array([0.44272026, 0.18971182])
```

```
1  np.sum(pca_model.explained_variance_ratio_)
```

✓ 0.9s

```
0.6324320765155946
```

```
1  pca_30 = PCA(n_components=30)
```

✓ 0.1s

```
1  pca_30.fit(scaled_X)
```

✓ 0.1s

```
PCA(n_components=30)
```

```
1  pca_30.explained_variance_ratio_
```

✓ 0.1s

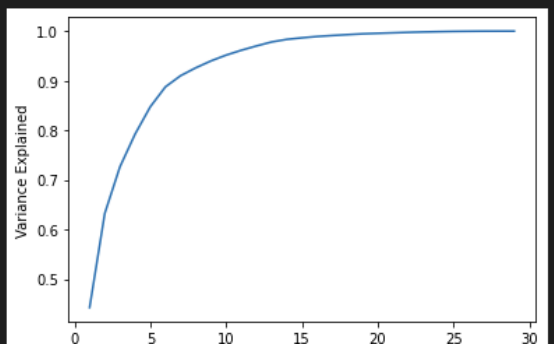
```
array([4.42720256e-01, 1.89711820e-01, 9.39316326e-02, 6.60213492e-02,
       5.49576849e-02, 4.02452204e-02, 2.25073371e-02, 1.58872380e-02,
       1.38964937e-02, 1.16897819e-02, 9.79718988e-03, 8.70537901e-03,
       8.04524987e-03, 5.23365745e-03, 3.13783217e-03, 2.66209337e-03,
       1.97996793e-03, 1.75395945e-03, 1.64925306e-03, 1.03864675e-03,
       9.99096464e-04, 9.14646751e-04, 8.11361259e-04, 6.01833567e-04,
       5.16042379e-04, 2.72587995e-04, 2.30015463e-04, 5.29779290e-05,
       2.49601032e-05, 4.43482743e-06])
```

```
1  np.sum(pca_30.explained_variance_ratio_)
```
✓ 0.1s

```
1.0
```

```
1  explained_variance = []
2
3  for n in range(1,30):
4      pca = PCA(n_components=n)
5      pca.fit(scaled_X)
6
7      explained_variance.append(np.sum(pca.explained_variance_ratio_))
```
✓ 0.4s

```
1  plt.plot(range(1,30),explained_variance)
2  plt.xlabel("Number of Components")
3  plt.ylabel("Variance Explained");
```
✓ 0.4s



▼ Handwritten Digits

Imports

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
```

✓ 7.4s

```
1 digits = pd.read_csv('digits.csv')
```

✓ 0.1s

```
1 digits
```

✓ 0.1s

	pixel_0_0	pixel_0_1	pixel_0_2	pixel_0_3	pixel_0_4	pixel_0_5	pixel_0_6	pixel_0_7	pixel_1_0	pixel_1_1
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0

```
1 pixels = digits.drop('number_label',axis=1)
2 pixels
```

✓ 0.1s

	pixel_0_0	pixel_0_1	pixel_0_2	pixel_0_3	pixel_0_4	pixel_0_5	pi
0	0.0	0.0	5.0	13.0	9.0	1.0	
1	0.0	0.0	0.0	12.0	13.0	5.0	
2	0.0	0.0	0.0	4.0	15.0	12.0	
3	0.0	0.0	7.0	15.0	13.0	1.0	
4	0.0	0.0	0.0	1.0	11.0	0.0	
...	...	...	...	...	...	...	...

```
1  single_image = pixels.iloc[0]
2  single_image
```

✓ 0.1s

```
pixel_0_0     0.0
pixel_0_1     0.0
pixel_0_2     5.0
pixel_0_3    13.0
pixel_0_4     9.0
             ...
pixel_7_3    13.0
pixel_7_4    10.0
pixel_7_5     0.0
pixel_7_6     0.0
pixel_7_7     0.0
Name: 0, Length: 64, dtype: float64
```

```
1  single_image.to_numpy()
```

✓ 0.1s

```
array([ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.,  0.,  0., 13., 15., 10.,
       15.,  5.,  0.,  0.,  3., 15.,  2.,  0., 11.,  8.,  0.,  0.,  4.,
       12.,  0.,  0.,  8.,  8.,  0.,  0.,  5.,  8.,  0.,  0.,  9.,  8.,
        0.,  0.,  4., 11.,  0.,  1., 12.,  7.,  0.,  0.,  2., 14.,  5.,
       10., 12.,  0.,  0.,  0.,  0.,  6., 13., 10.,  0.,  0.,  0.])
```

```
1  single_image.to_numpy().shape
```

✓ 0.1s

```
(64,)
```

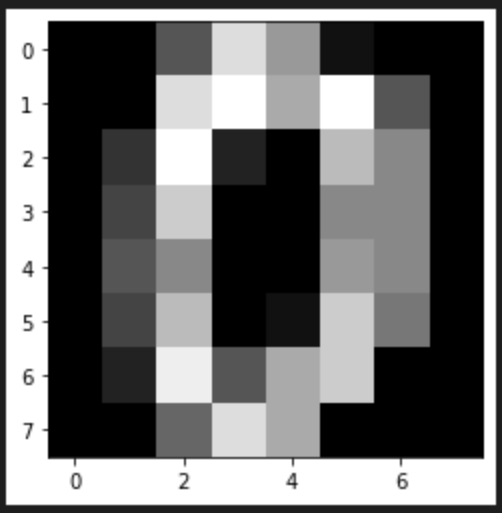
```
1  num = single_image.to_numpy().reshape(8,8)
```

✓ 0.1s

```
1  plt.imshow(num, cmap="gray")
```

✓ 0.5s

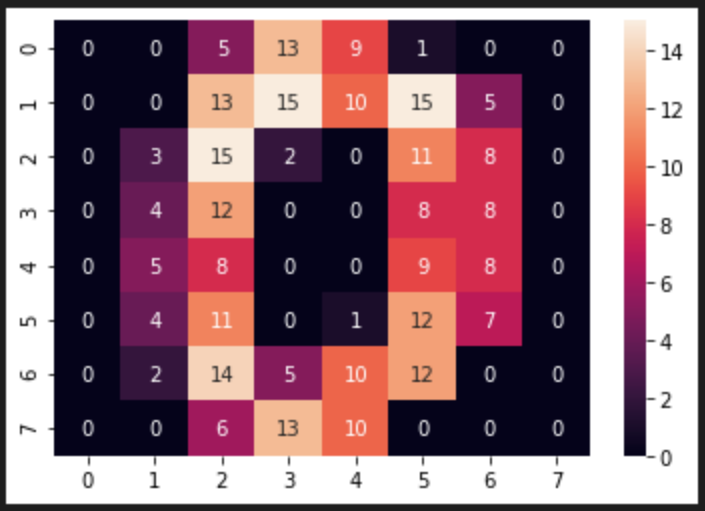
```
<matplotlib.image.AxesImage at 0x12a7769ac40>
```




```
1  sns.heatmap(num, annot=True)
```

✓ 1.1s

```
<AxesSubplot:>
```



Scaling Data

```
1 from sklearn.preprocessing import StandardScaler
```
✓ 0.5s

```
1 scaler = StandardScaler()
```
✓ 0.5s

```
1 scaled_pixels = scaler.fit_transform(pixels)
```
✓ 0.8s

```
1 pixels
```
✓ 0.2s

	pixel_0_0	pixel_0_1	pixel_0_2	pixel_0_3	pixel_0_4	pixel_0_5	pixel_0_6	pixel_0_7
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
1792	0.0	0.0	4.0	10.0	13.0	6.0	0.0	0.0
1793	0.0	0.0	6.0	16.0	13.0	11.0	1.0	0.0
1794	0.0	0.0	1.0	11.0	15.0	1.0	0.0	0.0
1795	0.0	0.0	2.0	10.0	7.0	0.0	0.0	0.0
1796	0.0	0.0	10.0	14.0	8.0	1.0	0.0	0.0

1797 rows × 64 columns

PCA

```
1 from sklearn.decomposition import PCA
```
✓ 0.1s

```
1 pca_model = PCA(n_components=2)
```
✓ 0.8s

```
1 pca_pixels = pca_model.fit_transform(scaled_pixels)
```
✓ 0.1s

```
1 pca_model.explained_variance_ratio_
```
✓ 0.1s

```
array([0.12033916, 0.09561054])
```

```
1 np.sum(pca_model.explained_variance_ratio_)
```
✓ 0.1s

```
0.2159497049303386
```

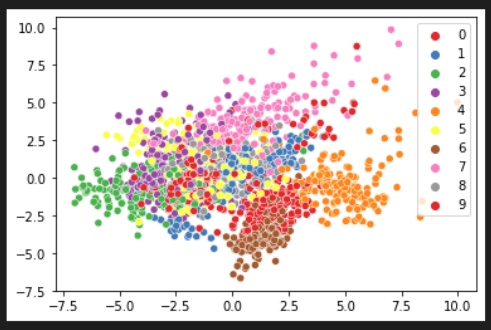
```
1 pca_pixels[:,1]
```
✓ 0.1s

```
array([-0.95450277,  0.92463286, -0.31734493, ..., -0.14782995,
       -0.38088582, -2.22747391])
```

```
1  s.scatterplot(pca_pixels[:,0] ,pca_pixels[:,1], hue=digits["number_label"].values, palette="Set1")
```

✓ 1.6s Python

```
C:\Users\mbatu\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following
variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(

<AxesSubplot:>
```




```
1  pca_model = PCA(n_components=3)
```

✓ 0.1s

```
1  pca_pixels = pca_model.fit_transform(scaled_pixels)
```

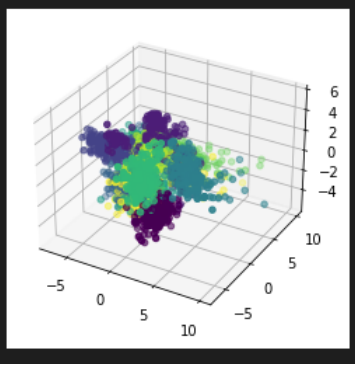
✓ 0.1s

```
1  from mpl_toolkits import mplot3d
```

✓ 0.1s

```
1
2  ax = plt.axes(projection='3d')
3  ax.scatter3D(pca_pixels[:,0],pca_pixels[:,1],pca_pixels[:,2],c=digits['number_label']);
```

✓ 0.4s Pytho




```
1  %matplotlib notebook
2  #adece jupyter notebookta çalılıyo
3  plt.figure(figsize=(8,8),dpi=150)
4  ax = plt.axes(projection='3d')
5  ax.scatter3D(pca_pixels[:,0],pca_pixels[:,1],pca_pixels[:,2],c=digits['number_label']);
```