PRINCIPAL COMPONENT ANALYSIS (PCA)

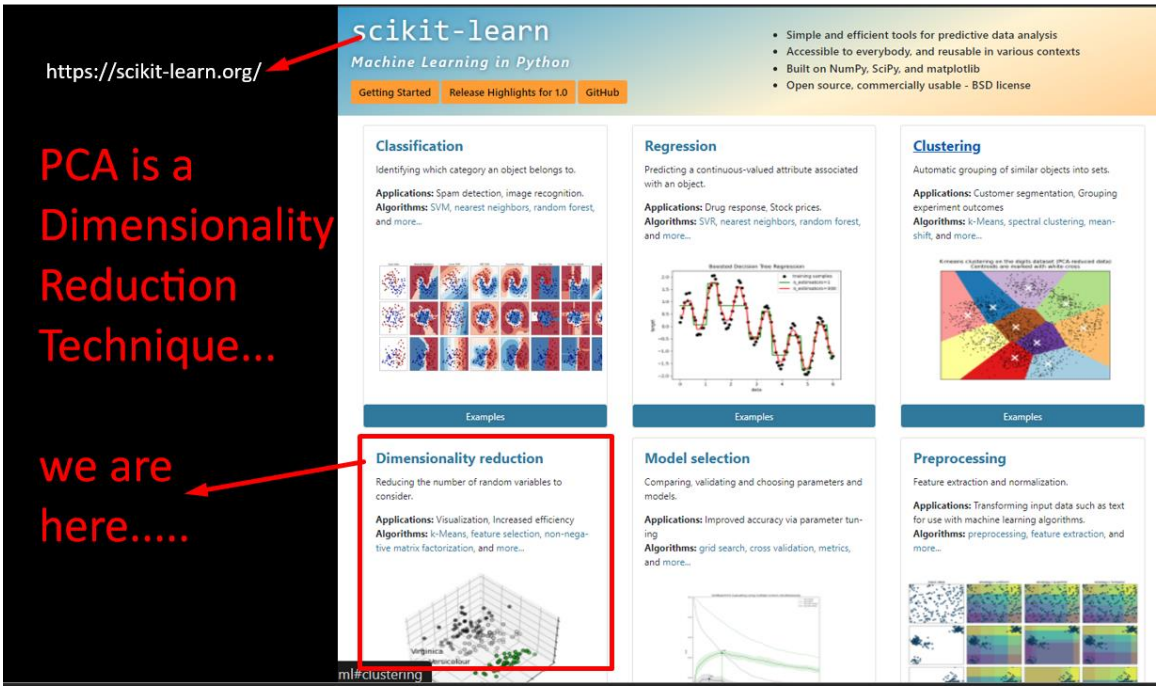
USING PYTHON
DR. ALVIN ANG

CONTENTS

I. UNDERSTANDING PRINCIPAL COMPONENT ANALYSIS (PCA)

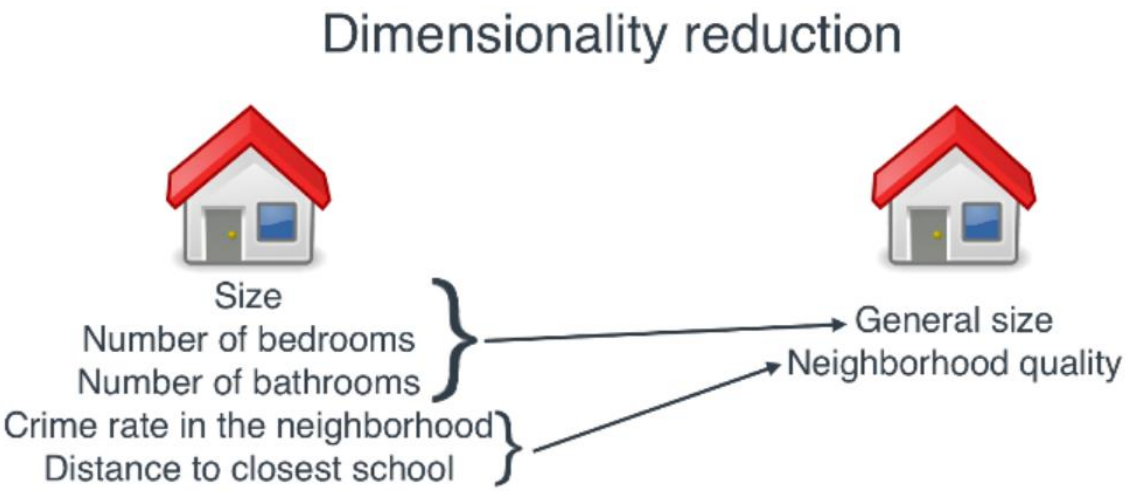
Most of the stuff here are abstracted frrom:

https://www.amazon.com/Grokking-Machine-Learning-Luis-Serrano/dp/1617295914

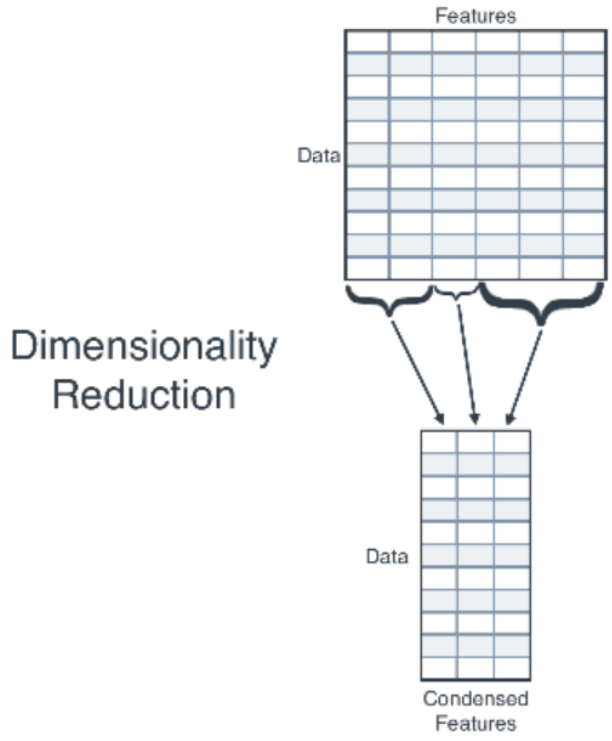


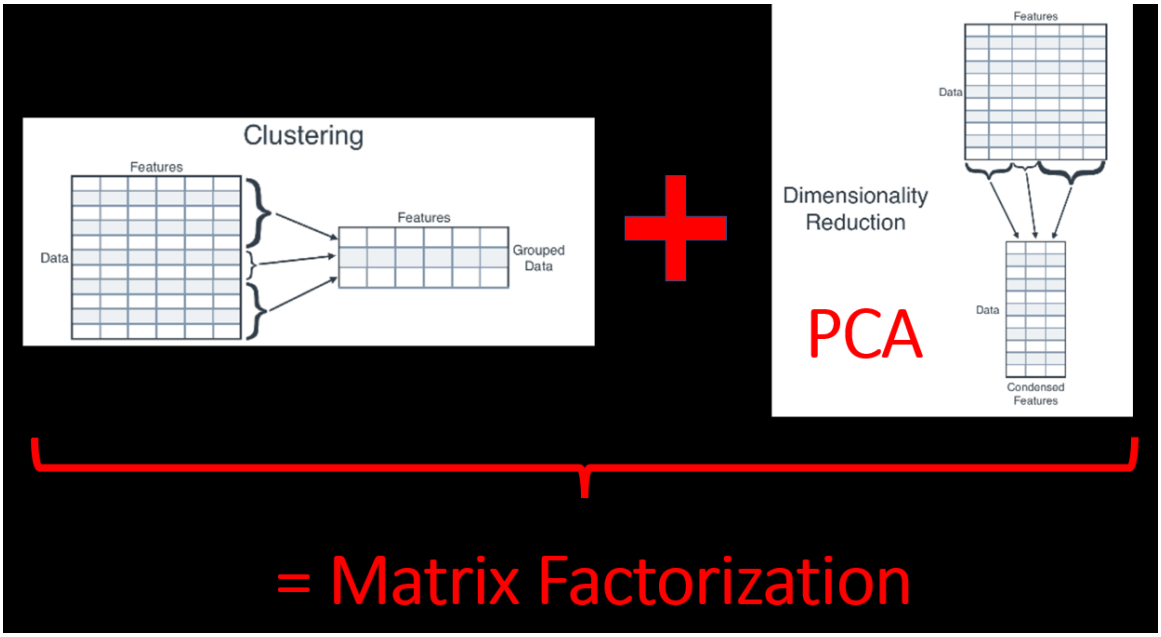
- There are various categories at https://scikit-learn.org/stable/
- Objective of PCA is to cut down the number of columns in your dataset (if you have too many) AND Combine them;
- Whilst trying not to lose too much valuable information.
- PCA is one of the many techniques to speed up Machine Learning.

A. IN A NUTSHELL….



- You have too many columns / features and you need to merge them for faster processing.





- If you combine both Clustering + PCA together, you are doing Matrix Factorization.
- But Matrix Factorization is out of scope for now.

B. SIMPLICITY VS INTERPRETABILITY

- PCA is a trade-off between SIMPLICITY vs INTERPRETABILITY.
- PCA increases SIMPLICITY of Machine Learning models.
- But PCA increases the difficulty of INTERPRETING the meaning of each variable.
- Because every Principal Component is a linear combination of all the other variables.

C. PCA IN PRACTICE

- PCA is used with Machine Learning (Classification models like logistic regression or k nearest neighbours) to make predictions.

II. PCA WITH PYTHON

References here:

https://www.datacamp.com/community/tutorials/principal-component-analysis-in-python

https://nickmccullum.com/python-machine-learning/principal-component-analysis-python/

IPYNB here: https://www.alvinang.sg/s/PCA_with_Python.ipynb

A. IMPORTING LIBRARIES

```
[1]  import pandas as pd
     import numpy as np

     import matplotlib.pyplot as plt
     import seaborn
     %matplotlib inline
```

B. IMPORT DATASET AND VIEW

```
from sklearn.datasets import load_breast_cancer
raw_data = load_breast_cancer()
raw_data
```

```
{'DESCR': '.. _breast_cancer_dataset:\n\nBreast cancer wisconsin (diagnostic) dataset\n-
 'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
         1.189e-01],
        [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
         8.902e-02],
        [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
         8.758e-02],
        ...,
        [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
         7.820e-02],
        [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
         1.240e-01],
        [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
         7.039e-02]]),
 'data_module': 'sklearn.datasets.data',
```

```
 'feature_names': array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
        'mean smoothness', 'mean compactness', 'mean concavity',
        'mean concave points', 'mean symmetry', 'mean fractal dimension',
        'radius error', 'texture error', 'perimeter error', 'area error',
        'smoothness error', 'compactness error', 'concavity error',
        'concave points error', 'symmetry error',
        'fractal dimension error', 'worst radius', 'worst texture',
        'worst perimeter', 'worst area', 'worst smoothness',
        'worst compactness', 'worst concavity', 'worst concave points',
        'worst symmetry', 'worst fractal dimension'], dtype='<U23'),
 'filename': 'breast_cancer.csv',
 'frame': None,
```

```
 'frame': None,
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
        1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
        1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
        0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
        1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
        0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
        1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
        0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
        1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
        1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
        1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
```

- We see that the Breast Cancer dataset is feature rich → it is stored as a Dictionary → Key Value pairs.
    - 'feature_names': columns of the dataset
    - 'data': values inside the dataset
    - 'target': Malignant (1) or Benign (0)
- The Breast Cancer data set consists of two classes:
    - Malignant = harmful / has cancer
    - Benign = not harmful / no cancer
- Malignant class has 212 samples.
- Benign class has 357 samples.

worst oothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension	label
4100	0.21130	0.4107	0.2216	0.2060	0.07115	Benign
1660	0.19220	0.3215	0.1628	0.2572	0.06637	Benign
1390	0.30940	0.3403	0.1418	0.2218	0.07820	Benign
6500	0.86810	0.9387	0.2650	0.4087	0.12400	Benign
8996	0.06444	0.0000	0.0000	0.2871	0.07039	Malignant

- You can see that the dataset uses 30 features / columns: radius, texture, perimeter, area, smoothness, fractal dimension, etc. in order to predict Benign or Malignant.
- Later on, we will use PCA to shrink these 30 columns to 2 columns and plot them.
- Thereafter, the 2 columns (known as Principal Component 1 and 2) plot can be used to see whether "Malignant" or "Benign".

C. EXTRACT ROWS AND COLUMNS FROM DATASET

```
raw_data_frame = pd.DataFrame(raw_data['data'], columns = raw_data['feature_names'])
raw_data_frame
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	25.380	17.33	184.60	2019.0	0.16220	0.66560	0.7119	0.2654
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	24.990	23.41	158.80	1956.0	0.12380	0.18660	0.2416	0.1860
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	23.570	25.53	152.50	1709.0	0.14440	0.42450	0.4504	0.2430
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	14.910	26.50	98.87	567.7	0.20980	0.86630	0.6869	0.2575
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	22.540	16.67	152.20	1575.0	0.13740	0.20500	0.4000	0.1625
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	25.450	26.40	166.10	2027.0	0.14100	0.21130	0.4107	0.2216
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	23.690	38.25	155.00	1731.0	0.11660	0.19220	0.3215	0.1628
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	18.980	34.12	126.70	1124.0	0.11390	0.30940	0.3403	0.1418
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	25.740	39.42	184.60	1821.0	0.16500	0.86810	0.9387	0.2650
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	9.456	30.37	59.16	268.6	0.08996	0.06444	0.0000	0.0000

569 rows × 30 columns

'Features' are extracted as column names

'Data' is extracted as values in the rows

- From the Breast Cancer dictionary, we extract out:
    - 'feature_names' as the column headers
    - 'data' as the values in the rows

```
raw_data_frame.columns
```

```
Index(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error', 'fractal dimension error',
       'worst radius', 'worst texture', 'worst perimeter', 'worst area',
       'worst smoothness', 'worst compactness', 'worst concavity',
       'worst concave points', 'worst symmetry', 'worst fractal dimension'],
      dtype='object')
```

D. STANDARDIZING THE DATA

```
[6] #Standardize the data
    from sklearn.preprocessing import StandardScaler
    data_scaler = StandardScaler()
    data_scaler.fit(raw_data_frame)
    scaled_data_frame = data_scaler.transform(raw_data_frame)
```

```
scaled_data_frame
```

```
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],
       ...,
       [ 0.70228425,  2.0455738 ,  0.67267578, ...,  0.41406869,
        -1.10454895, -0.31840916],
       [ 1.83834103,  2.33645719,  1.98252415, ...,  2.28998549,
         1.91908301,  2.21963528],
       [-1.80840125,  1.22179204, -1.81438851, ..., -1.74506282,
        -0.04813821, -0.75120669]])
```

- Since its quite hard to see in array format, we use pandas to reformat it as below….

```
[13] a = pd.DataFrame(scaled_data_frame)
     a
```

	0	1	2	3	4	5	6	7	8	9	...	20	21	22
0	1.097064	-2.073335	1.269934	0.984375	1.568466	3.283515	2.652874	2.532475	2.217515	2.255747	...	1.886690	-1.359293	2.303601
1	1.829821	-0.353632	1.685955	1.908708	-0.826962	-0.487072	-0.023846	0.548144	0.001392	-0.868652	...	1.805927	-0.369203	1.535126
2	1.579888	0.456187	1.566503	1.558884	0.942210	1.052926	1.363478	2.037231	0.939685	-0.398008	...	1.511870	-0.023974	1.347475
3	-0.768909	0.253732	-0.592687	-0.764464	3.283553	3.402909	1.915897	1.451707	2.867383	4.910919	...	-0.281464	0.133984	-0.249939
4	1.750297	-1.151816	1.776573	1.826229	0.280372	0.539340	1.371011	1.428493	-0.009560	-0.562450	...	1.298575	-1.466770	1.338539
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
564	2.110995	0.721473	2.060786	2.343856	1.041842	0.219060	1.947285	2.320965	-0.312589	-0.931027	...	1.901185	0.117700	1.752563
565	1.704854	2.085134	1.615931	1.723842	0.102458	-0.017833	0.693043	1.263669	-0.217664	-1.058611	...	1.536720	2.047399	1.421940

E. PCA THE STANDARDIZED DATA

```
#Perform the principal component analysis transformation
from sklearn.decomposition import PCA

pca = PCA(n_components = 2)
pca.fit(scaled_data_frame)
```

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

we decide to shrink the number of columns from 30 to 2.
n = 2 means that we are going to use PCA to compress those 30 columns (from the breast cancer dataset) down to 2 columns.

```
[37] #Peeking at the Principal Component 1
     pca.components_[0]
```

```
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])
```

```
#Peeking at the Principal Component 2
pca.components_[1]
```

```
array([-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])
```

```
x_pca = pca.transform(scaled_data_frame)
x_pca
```

```
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]])
```

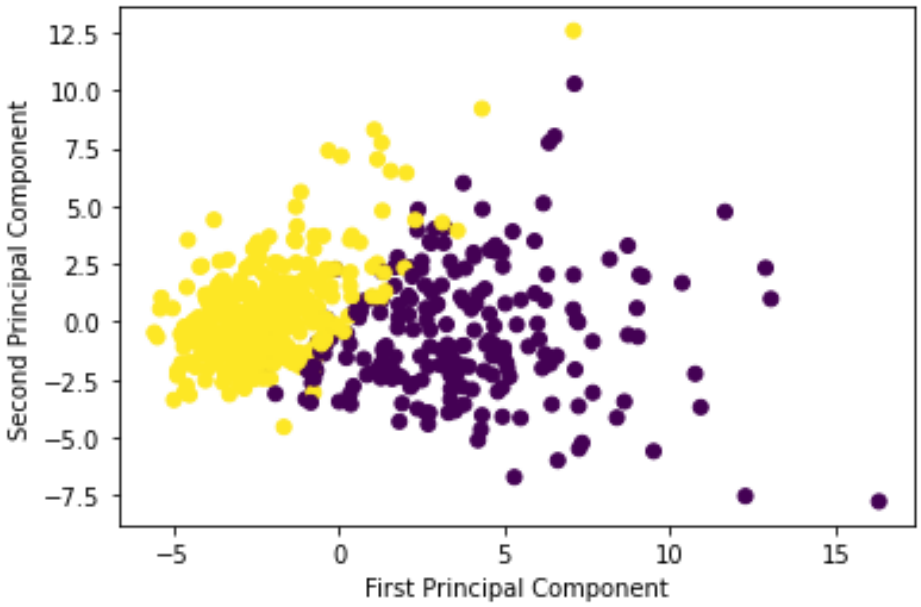
```
[15] x_pca.shape
```

```
(569, 2)
```

the new PCA dataframe still consists of 569 rows (as per previous) but now shrunk down to 2 columns

F. PLOTTING

```
#Visualize the principal components with a color scheme
plt.scatter(x_pca[:,0],x_pca[:,1], c=raw_data['target'])
plt.xlabel('First Principal Component')
plt.ylabel('Second Principal Component')
```

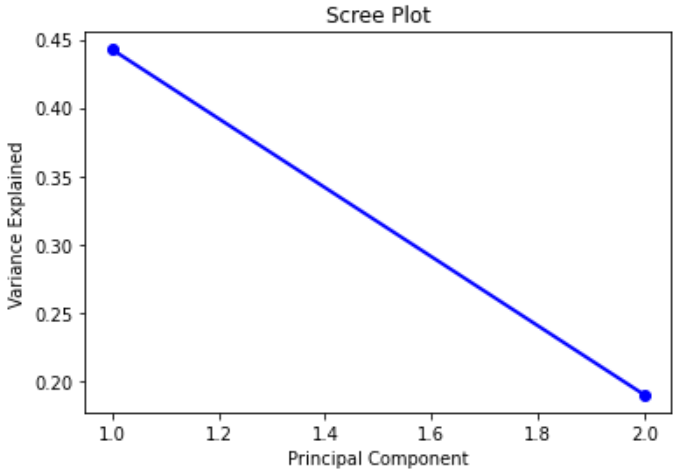


- From the graph, you can observe the two classes: Benign and Malignant

G. SCREE PLOT

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

PC_values = np.arange(pca.n_components_) + 1
plt.plot(PC_values, pca.explained_variance_ratio_, 'o-', linewidth=2, color='blue')
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Variance Explained')
plt.show()
```



H. VARIANCE RATIO

```
print(pca.explained_variance_ratio_)
```

```
[0.44272026 0.18971182]
```

- The Scree Plot shows the Variance Ratio for Principal Component (PC) 1 and 2.
- Meaning, PC 1 holds 44.2% of the information
- PC 2 holds 19% of the information
- The remaining 36.8% information was lost due to PCA (i.e. compressing 30 columns to 2 columns).

ABOUT DR. ALVIN ANG

Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He is a scientist, entrepreneur, as well as a personal/business advisor. More about him at www.AlvinAng.sg.