Principal Component Analysis (PCA) in Python

Principal Component Analysis (PCA) is a linear dimensionality reduction technique that can be util dimensional space by projecting it into a lower-dimensional sub-space. It tries to preserve the esse data and remove the non-essential parts with fewer variation.

One important thing to note about PCA is that it is an Unsupervised dimensionality reduction techr

According to Wikipedia, PCA is a statistical procedure that uses an orthogonal transformation to c correlated variables (entities each of which takes on various numerical values) into a set of values principal components.

But where all you can apply PCA?

- Data Visualization: When working on any data related problem, the challenge in today's worlc variables/features that define that data. Considering that there are a large number of variable distributed, visualization can be a challenge and almost impossible.
- Speeding Machine Learning (ML) Algorithm: Since PCA's main idea is dimensionality reducti machine learning algorithm's training and testing time considering your data has a lot of feat slow.

What is a Principal Component?

- Principal components are the key to PCA; they represent what's underneath the hood of your projected into a lower dimension (assume three dimensions) from a higher space, the three Components that captures (or holds) most of the variance (information) of your data.

`Main : In this way, given a set of x correlated variables over y samples you achieve a components over the same y samples. The reason you achieve uncorrelated principal compor the correlated features contribute to the same principal component, thereby reducing the uncorrelated principal components`

`Note: Features, Dimensions, and Variables are all referring to the same thing. You will`

Let's load `breast-cancer` dataset

```
from sklearn.datasets import load_breast_cancer
breast = load_breast_cancer()
data = breast.data
data.shape
```

```
(569, 30)
```

```
breast_labels = breast.target
# breast_labels
```

```
features = breast.feature_names
features
```

```
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')
```

```
import numpy as np
labels = np.reshape(breast_labels,(569,1))
```

```
final_breast_data = np.concatenate([data,labels],axis=1)
final_breast_data.shape
```

```
(569, 31)
```

```
import pandas as pd
```

```
breast_dataset = pd.DataFrame(final_breast_data)
features = np.append(features,'label')
breast_dataset.columns = features
breast_dataset.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	sy
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	

```
breast_dataset['label'].replace(0, 'Benign',inplace=True)
breast_dataset['label'].replace(1, 'Malignant',inplace=True)
breast_dataset.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	sy
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	

Let's load `cifar10` dataset

4 cells hidden

Data Visualization using PCA

Visualizing the Breast Cancer data

- You start by Standardizing the data since PCA's output is influenced based on the scale of th
- It is a common practice to normalize your data before feeding it to any machine learning alg

While applying StandardScaler, each feature of your data should be normally distributed such that and a standard deviation of one.

```
from sklearn.preprocessing import StandardScaler

X = breast_dataset.drop(columns='label',axis=1)
y = breast_dataset.label

X_scaled = StandardScaler().fit_transform(X)
```

```
X_scaled.shape
```

```
(569, 30)
```

```
normalized_breast_dataset = pd.DataFrame(X_scaled)
normalized_breast_dataset.columns = [ 'feature' + str(i) for i in range(X_scaled.shape[1])]
normalized_breast_dataset.head()
```

	feature0	feature1	feature2	feature3	feature4	feature5	feature6	feature7
0	1.097064	-2.073335	1.269934	0.984375	1.568466	3.283515	2.652874	2.532475
1	1.829821	-0.353632	1.685955	1.908708	-0.826962	-0.487072	-0.023846	0.548144
2	1.579888	0.456187	1.566503	1.558884	0.942210	1.052926	1.363478	2.037231
3	-0.768909	0.253732	-0.592687	-0.764464	3.283553	3.402909	1.915897	1.451707
4	1.750297	-1.151816	1.776573	1.826229	0.280372	0.539340	1.371011	1.428493

```
# Now comes the critical part, the next few lines of code will be projecting the thirty-dimer
```

```
from sklearn.decomposition import PCA
pca_breast = PCA(n_components=2)

# let's fit the data...

principalComponents_breast = pca_breast.fit_transform(X_scaled)
```

```
principalComponents_breast.shape
```

```
(569, 2)
```

```
pca_breast_dataset = pd.DataFrame(principalComponents_breast,columns=['Principal_Component_1'
pca_breast_dataset.head()
```

	Principal_Component_1	Principal_Component_2
0	9.192837	1.948583
1	2.387802	-3.768172
2	5.733896	-1.075174
3	7.122953	10.275589
4	3.935302	-1.948072

```
# Once you have the principal components, you can find the explained_variance_ratio.
# It will provide you with the amount of information or variance each principal component hol
print('Explained variation per principal component: {}'.format(pca_breast.explained_variance_
```

```
Explained variation per principal component: [0.44272026 0.18971182]
```

From the above output, you can observe that the `principal component 1` holds `44.2%` of the infor holds only `19%` of the information.Also, the other point to note is that while projecting thirty-dimen information was `lost`.

```
breast_dataset.label == 'Benign'
```

```
0       True
1       True
2       True
3       True
4       True
       ...
564     True
565     True
566     True
567     True
568    False
Name: label, Length: 569, dtype: bool
```

```
import matplotlib.pyplot as plt
plt.figure()
plt.figure(figsize=(10,10))
plt.xticks(fontsize=12)
plt.yticks(fontsize=14)
plt.xlabel('Principal Component - 1',fontsize=20)
plt.ylabel('Principal Component - 2',fontsize=20)
plt.title("Principal Component Analysis of Breast Cancer Dataset",fontsize=20)
targets = ['Benign', 'Malignant']
colors = ['r', 'g']
for target, color in zip(targets,colors):
    indicesToKeep = breast_dataset['label'] == target
    plt.scatter(pca_breast_dataset.loc[indicesToKeep, 'Principal_Component_1']
                , pca_breast_dataset.loc[indicesToKeep, 'Principal_Component_1'], c = color, s

plt.legend(targets,prop={'size': 15})
```

```
<matplotlib.legend.Legend at 0x7fb9624d2be0>
<Figure size 432x288 with 0 Axes>
```