Import Dependencies

In [0]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
```

Dataset

In [0]:

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

# load dataset into Pandas DataFrame
df = pd.read_csv(url, names=['sepal length','sepal width','petal length','petal width','targe
```

In [4]:

```
df.head()
```

Out[4]:

	sepal length	sepal width	petal length	petal width	target
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Standardize The Data

In [0]:

```
feature = ['sepal length', 'sepal width', 'petal length', 'petal width']

# separating features
x = df.loc[:,feature]

# separating target
y = df.loc[:,'target']

#Standardising features
x = StandardScaler().fit_transform(x)
```

PCA (Principal Component Analysis)

In [0]:

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)

pct = pca.fit_transform(x)

principal_df = pd.DataFrame(pct,columns=['pc1','pc2'])

finaldf= pd.concat([principal_df,df[['target']]],axis=1)
```

In [16]:

```
finaldf.head()
```

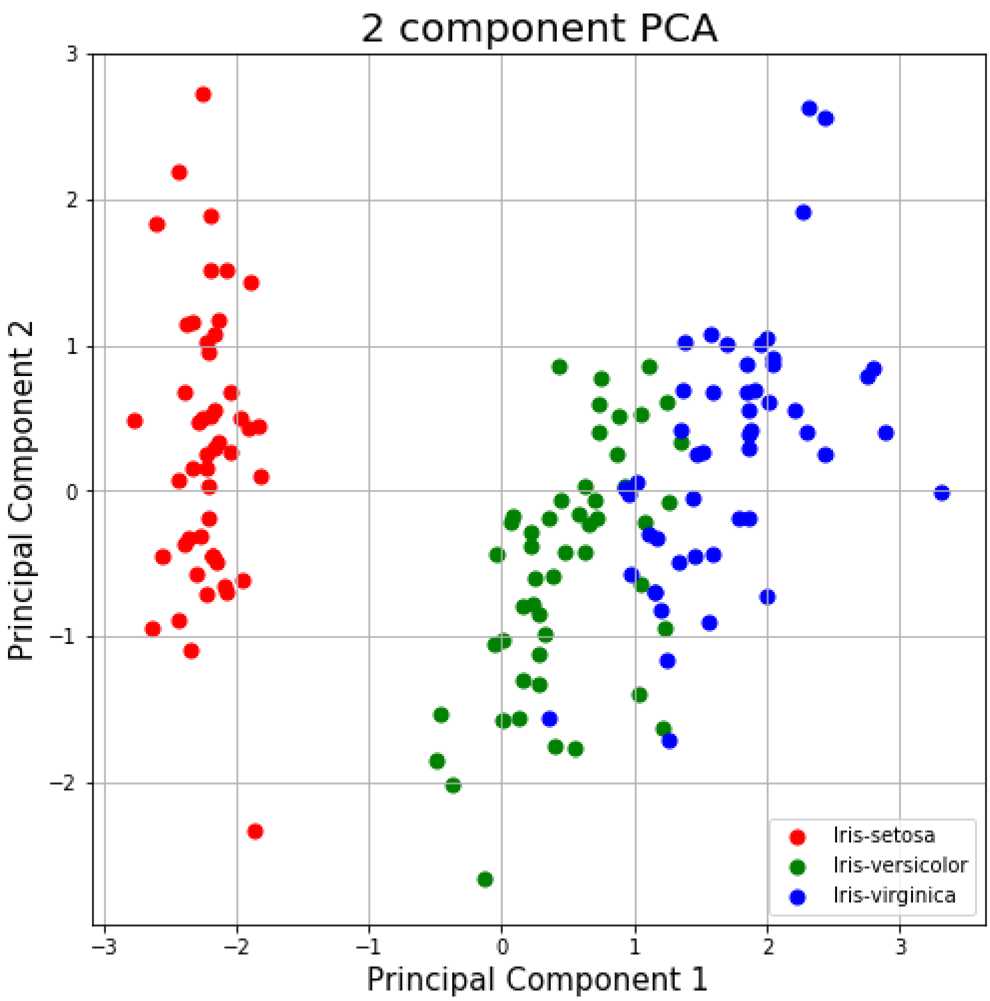
Out[16]:

	pc1	pc2	target
0	-2.264542	0.505704	Iris-setosa
1	-2.086426	-0.655405	Iris-setosa
2	-2.367950	-0.318477	Iris-setosa
3	-2.304197	-0.575368	Iris-setosa
4	-2.388777	0.674767	Iris-setosa

Component Projection (2D)

In [20]:

```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)
targets = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
colors = ['r', 'g', 'b']
for target, color in zip(targets,colors):
    indicesToKeep = finaldf['target'] == target
    ax.scatter(finaldf.loc[indicesToKeep, 'pc1']
               , finaldf.loc[indicesToKeep, 'pc2']
               , c = color
               , s = 50)
ax.legend(targets)
ax.grid()
```



In [21]:

```
pca.explained_variance_ratio_
```

Out[21]:

```
array([0.72770452, 0.23030523])
```

Conclusion:

- The explained variance tells you how much information (variance) can be attributed to each of the principal components.
- This is important as while you can convert 4 dimensional space to 2 dimensional space, you lose some of the variance (information) when you do this.
- By using the attribute explained_variance_ratio_, you can see that the first principal component contains 72.77% of the variance and the second principal component contains 23.03% of the variance.
- Together, the two components contain 95.80% of the information.

Variance Threshold

In [1]:

```
from sklearn.feature_selection import VarianceThreshold
from sklearn import datasets
```

Load Data

In [2]:

```
# Load iris data
iris = datasets.load_iris()

# Create features and target
X = iris.data
y = iris.target
```

Conduct Variance Thresholding

In [3]:

```
# Create VarianceThreshold object with a variance with a threshold of 0.5
thresholder = VarianceThreshold(threshold=.5)

# Conduct variance thresholding
X_high_variance = thresholder.fit_transform(X)
```

View high variance features

In [4]:

```
# View first five rows with features with variances above threshold
X_high_variance[0:5]
```

Out[4]:

```
array([[5.1, 1.4, 0.2],
       [4.9, 1.4, 0.2],
       [4.7, 1.3, 0.2],
       [4.6, 1.5, 0.2],
       [5. , 1.4, 0.2]])
```