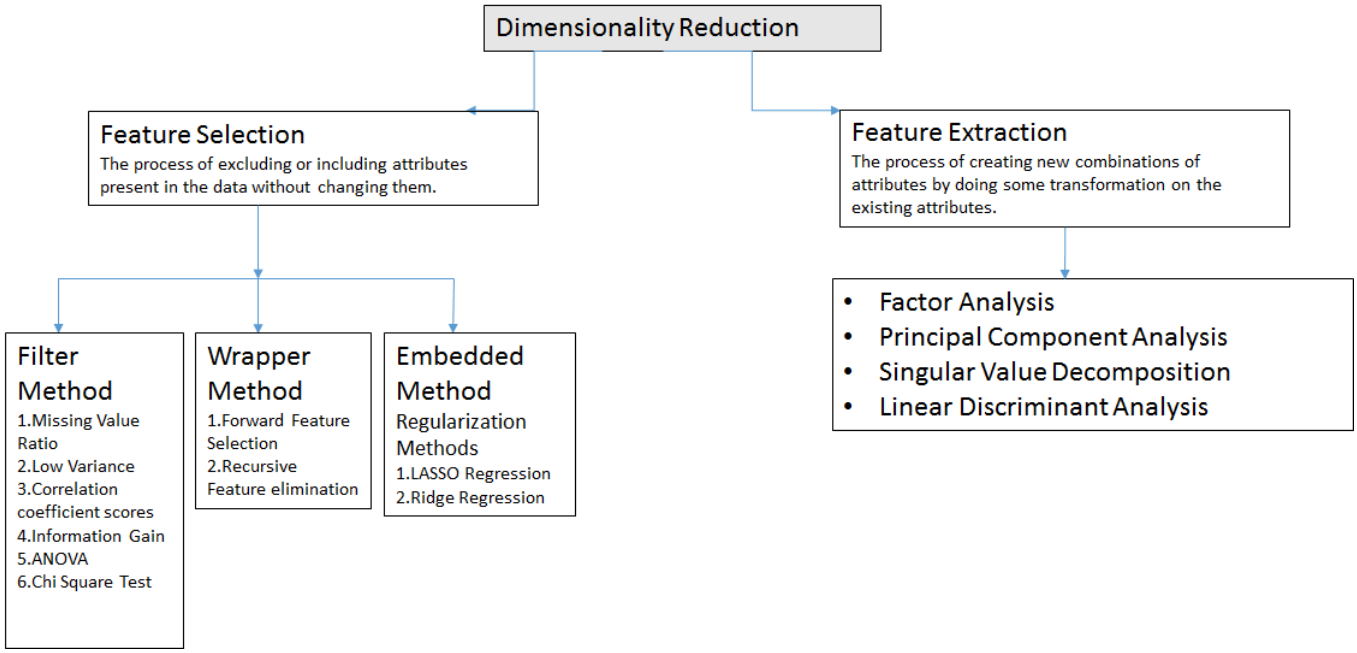
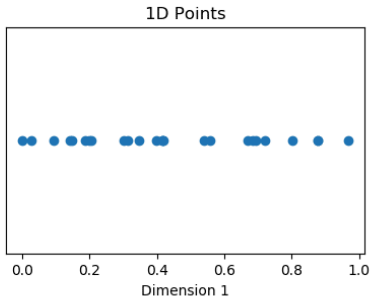
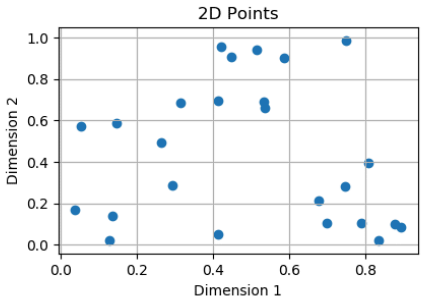
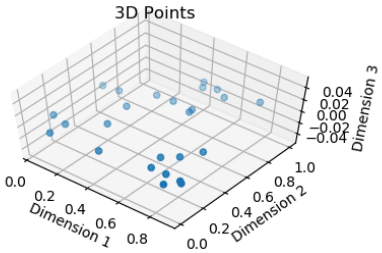
Dimension reduction:









How to Handle High Dimensional Data

There are two common ways to deal with high dimensional data:

1. Choose to include fewer features.
    - Drop features with many missing values
    - Drop features with low variance
    - Drop features with low correlation with the response variable
2. Use a regularization method.
    - Principal Components Analysis
    - Factor analysis
    - Principal Components Regression
    - Ridge Regression
    - Lasso Regression

Each of these techniques can be used to effectively deal with high dimensional data

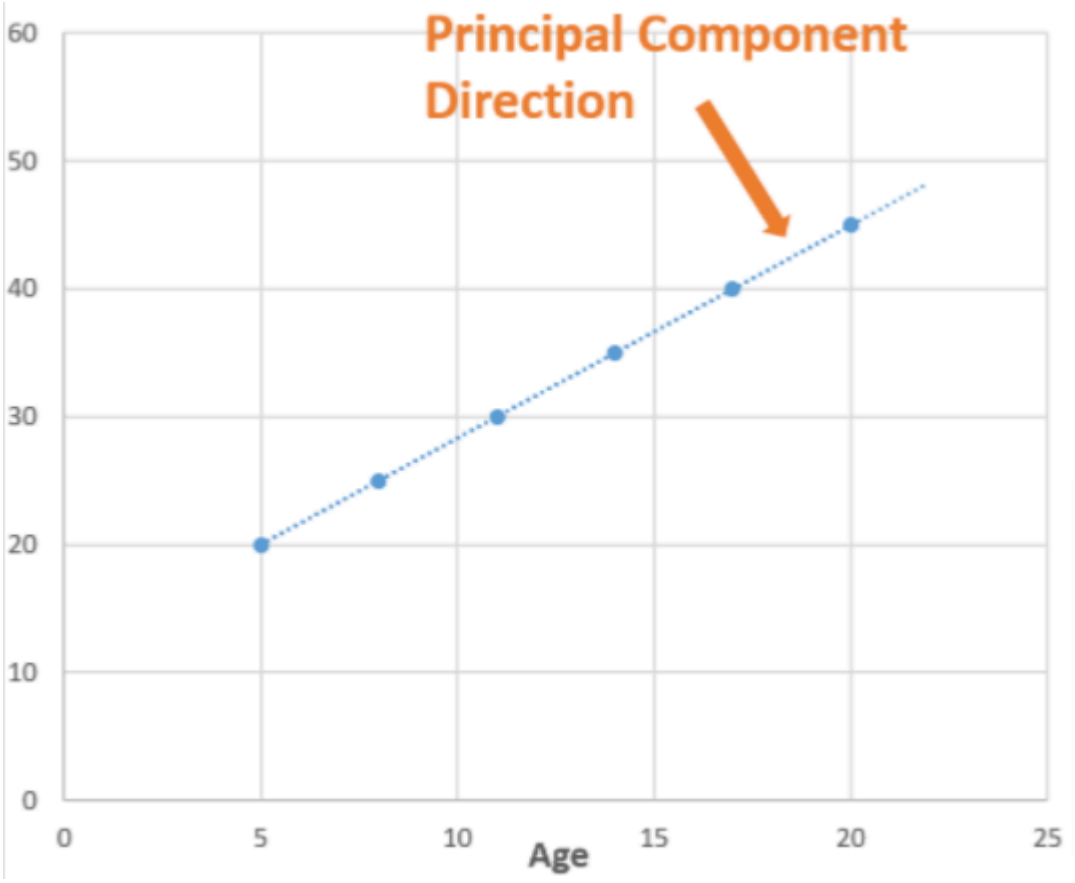
Principal Components Analysis (PCA):

1. How does Dimensionality Reduction works (using PCA)?
    - Once again remember the difference between DATA and INFORMATION.

Age	Weight	Height
5	20	4
8	25	4
11	30	4
14	35	4
17	40	4
20	45	4

The information lies in the variance!

As the Height variance is 0 thus it’s not adding any information.


Principal Component
Direction
60
50
40
30
20
10
0
0
5
10
15
20
25
Age

Definition:

Principal Component Analysis is basically a statistical procedure to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables. Each of the principal components is chosen in such a way so that it would describe most of them still available variance and all these principal components are orthogonal to each other. In all principal components first principal component has a maximum variance.

Uses of PCA:

- It is used to find inter-relation between variables in the data.
- It is used to interpret and visualize data.
- The number of variables is decreasing it makes further analysis simpler.
- It's often used to visualize genetic distance and relatedness between populations.

These are basically performed on a square symmetric matrix. It can be a pure sums of squares and cross-products matrix or Covariance matrix or Correlation matrix. A correlation matrix is used if the individual variance differs much.

Objectives of PCA:

- It is basically a non-dependent procedure in which it reduces attribute space from a large number of variables to a smaller number of factors.
- PCA is basically a dimension reduction process but there is no guarantee that the dimension is interpretable.
- The main task in this PCA is to select a subset of variables from a larger set, based on which original variables have the highest correlation with the principal amount.

Principal Axis Method:

PCA basically searches a linear combination of variables so that we can extract maximum variance from the variables. Once this process completes it removes it and searches for another linear combination that gives an explanation about the maximum proportion of remaining variance which basically leads to orthogonal factors. In this method, we analyze total variance.

Eigenvector:

It is a non-zero vector that stays parallel after matrix multiplication. Let's suppose x is an eigenvector of dimension r of matrix M with dimension r*r if Mx and x are parallel. Then we need to solve Mx=Ax where both x and A are unknown to get eigenvector and eigenvalues. Under Eigen-Vectors we can say that Principal components show both common and unique variance of the variable. Basically, it is variance focused approach seeking to reproduce total variance and correlation with all components. The principal components are basically the linear combinations of the original variables weighted by their contribution to explain the variance in a particular orthogonal dimension.

Eigen Values:

It is basically known as characteristic roots. It basically measures the variance in all variables which is accounted for by that factor. The ratio of eigenvalues is the ratio of explanatory importance of the factors with respect to the variables. If the factor is low then it is contributing less to the explanation of variables. In simple

words, it measures the amount of variance in the total given database accounted by the factor. We can calculate the factor's eigenvalue as the sum of its squared factor loading for all the variables.

Covariance

- Variance and Covariance are a measure of the "spread" of a set of points around their center of mass (mean)
- Variance – measure of the deviation from the mean for points in one dimension e.g. heights
- Covariance as a measure of how much each of the dimensions vary from the mean with respect to each other.
- Covariance is measured between 2 dimensions to see if there is a relationship between the 2 dimensions e.g. number of hours studied & marks obtained.
- The covariance between one dimension and itself is the variance

$$\text{covariance}(X,Y) = \frac{\sum_{i=1}^{n} (\bar{X}_i - X)(\bar{Y}_i - Y)}{(n-1)}$$

Covariance Matrix

- Representing Covariance between dimensions as a matrix e.g. for 3 dimensions:

$$C = \begin{bmatrix} \text{cov}(x,x) & \text{cov}(x,y) & \text{cov}(x,z) \\ \text{cov}(y,x) & \text{cov}(y,y) & \text{cov}(y,z) \\ \text{cov}(z,x) & \text{cov}(z,y) & \text{cov}(z,z) \end{bmatrix}$$

Variances

- Diagonal is the variances of x, y and z
- cov(x,y) = cov(y,x) hence matrix is symmetrical about the diagonal
- N-dimensional data will result in NxN covariance matrix

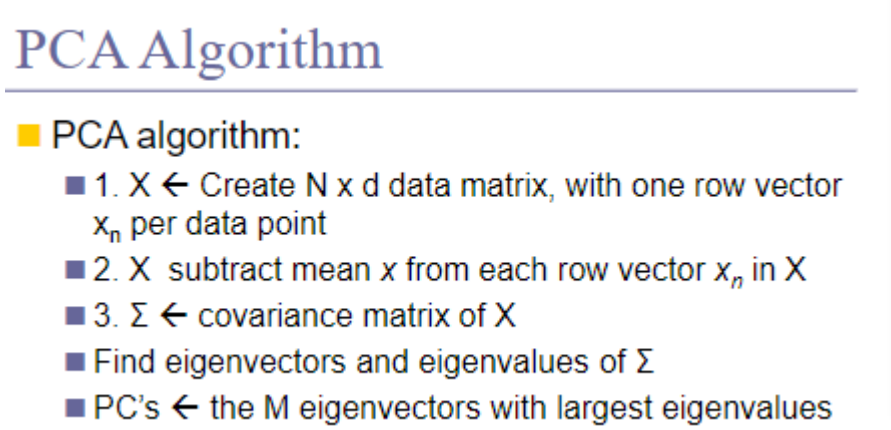
Algorithm 1 The PCA algorithm

1: **Input**: a D-dimensional training set $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ and the new (lower) dimensionality d (with $d \leq D$)
2: Compute the mean $\bar{\boldsymbol{x}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}_i$
3: Compute the covariance matrix $\text{Cov}(\boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}_i)(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_i)^T$
4: Find the spectral decomposition of $\text{Cov}(\boldsymbol{x})$, obtaining the eigenvectors $\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_D$ and their corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_D$. Note that the eigenvalues are sorted, such that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D \geq 0$
5: For any $\boldsymbol{x} \in \mathbb{R}^D$, its new lower dimensional representation is:

$$\boldsymbol{y} = \left(\boldsymbol{\xi}_1^T(\boldsymbol{x} - \bar{\boldsymbol{x}}), \boldsymbol{\xi}_2^T(\boldsymbol{x} - \bar{\boldsymbol{x}}), \ldots, \boldsymbol{\xi}_d^T(\boldsymbol{x} - \bar{\boldsymbol{x}})\right)^T \in \mathbb{R}^d, \tag{29}$$

and the original $\boldsymbol{x}$ can be approximated as

$$\boldsymbol{x} \approx \bar{\boldsymbol{x}} + (\boldsymbol{\xi}_1^T(\boldsymbol{x} - \bar{\boldsymbol{x}}))\boldsymbol{\xi}_1 + (\boldsymbol{\xi}_2^T(\boldsymbol{x} - \bar{\boldsymbol{x}}))\boldsymbol{\xi}_2 + \cdots + (\boldsymbol{\xi}_d^T(\boldsymbol{x} - \bar{\boldsymbol{x}}))\boldsymbol{\xi}_d \tag{30}$$



First principal component is a linear combination of original predictor variables which captures the maximum variance in the data set. It determines the direction of highest variability in the data. The second principal component lies perpendicular to the first principal component.

In [1]:
```
# importing required libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# importing or loading the dataset
dataset = pd.read_csv('wine.csv')

# distributing the dataset into two components X and Y
X = dataset.iloc[:, 0:13].values
y = dataset.iloc[:, 13].values
```

In [2]:
```
# Splitting the X and Y into the
# Training set and Testing set
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 0)
```

In [3]:
```
# performing preprocessing part
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [4]:
```
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
df = pd.read_csv('winequality-red.csv')
df.head()
```

Out[4]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alco
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	

In [5]:
```
y = df.pop('quality')
```

In [6]:
```
# AS DATA STANDARDIZATION/NORMALIZATION IS MUST WITH PCA
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
df_scaled = pd.DataFrame(scalar.fit_transform(df), columns=df.columns)
df_scaled
```

Out[6]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	
0	-0.528360	0.961877	-1.391472	-0.453218	-0.243707	-0.466193	-0.379133	0.558274	1
1	-0.298547	1.967442	-1.391472	0.043416	0.223875	0.872638	0.624363	0.028261	-0
2	-0.298547	1.297065	-1.186070	-0.169427	0.096353	-0.083669	0.229047	0.134264	-0
3	1.654856	-1.384443	1.484154	-0.453218	-0.264960	0.107592	0.411500	0.664277	-0
4	-0.528360	0.961877	-1.391472	-0.453218	-0.243707	-0.466193	-0.379133	0.558274	1
...	...	...	...	...	...	...	...	...	
1594	-1.217796	0.403229	-0.980669	-0.382271	0.053845	1.542054	-0.075043	-0.978765	0
1595	-1.390155	0.123905	-0.877968	-0.240375	-0.541259	2.211469	0.137820	-0.862162	1
1596	-1.160343	-0.099554	-0.723916	-0.169427	-0.243707	1.255161	-0.196679	-0.533554	0
1597	-1.390155	0.654620	-0.775267	-0.382271	-0.264960	1.542054	-0.075043	-0.676657	1
1598	-1.332702	-1.216849	1.021999	0.752894	-0.434990	0.203223	-0.135861	-0.666057	(

1599 rows × 11 columns

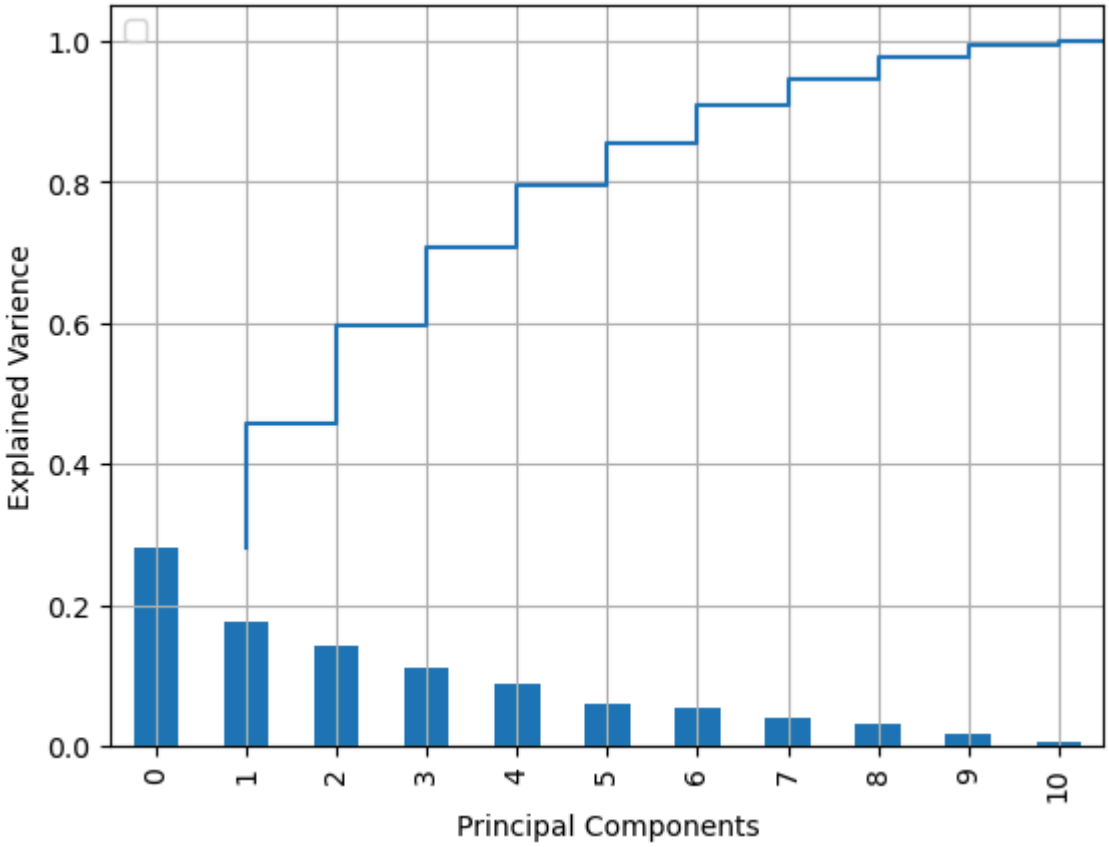
In [7]:
```
pca = PCA()
df_pca = pd.DataFrame(pca.fit_transform(df_scaled))
df_pca
```

Out[7]:

	0	1	2	3	4	5	6	7	
0	-1.619530	0.450950	-1.774454	0.043740	0.067014	-0.913921	-0.161043	-0.282258	0
1	-0.799170	1.856553	-0.911690	0.548066	-0.018392	0.929714	-1.009829	0.762587	-0
2	-0.748479	0.882039	-1.171394	0.411021	-0.043531	0.401473	-0.539553	0.597946	-0
3	2.357673	-0.269976	0.243489	-0.928450	-1.499149	-0.131017	0.344290	-0.455375	0
4	-1.619530	0.450950	-1.774454	0.043740	0.067014	-0.913921	-0.161043	-0.282258	0
...	...	...	...	...	...	...	...	...	
1594	-2.150500	0.814286	0.617063	0.407687	-0.240936	0.054835	0.170812	-0.355866	-0
1595	-2.214496	0.893101	1.807402	0.414003	0.119592	-0.674711	-0.607970	-0.247640	-1
1596	-1.456129	0.311746	1.124239	0.491877	0.193716	-0.506410	-0.231082	0.079382	-0
1597	-2.270518	0.979791	0.627965	0.639770	0.067735	-0.860408	-0.321487	-0.468876	-0
1598	-0.426975	-0.536690	1.628955	-0.391716	0.450482	-0.496154	1.189132	0.042176	0

1599 rows × 11 columns

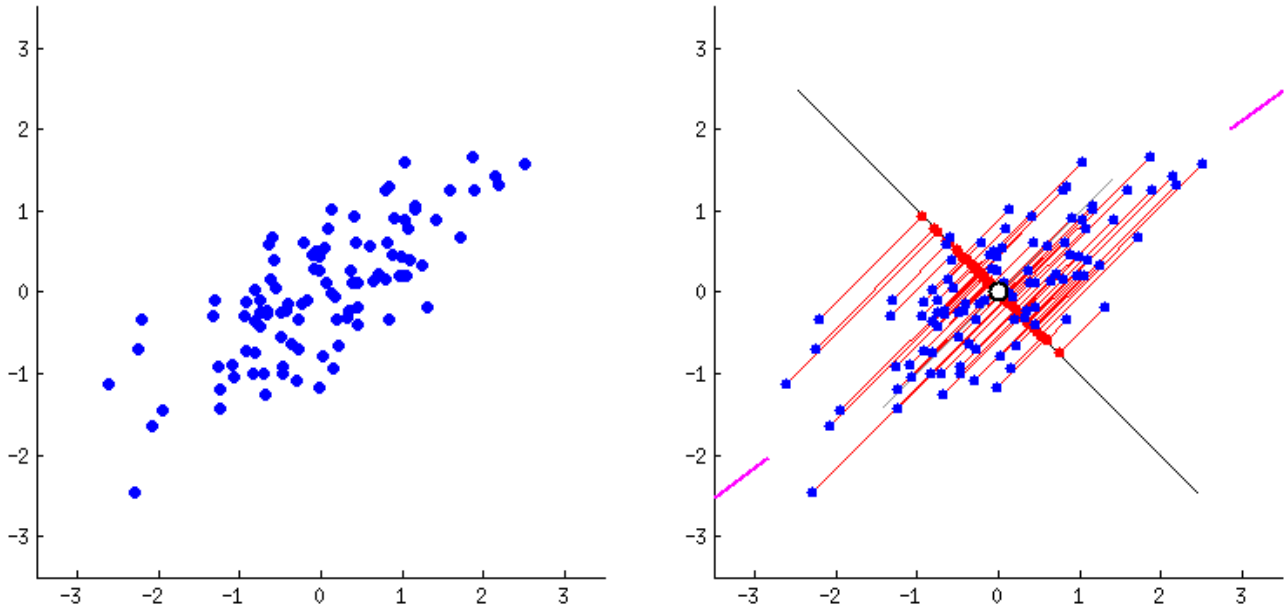
In [32]:
```
import matplotlib.pyplot as plt
pd.DataFrame(pca.explained_variance_ratio_).plot.bar()
plt.step(range(1,len(pca.explained_variance_ratio_)+1),np.cumsum(pca.explai
ned_variance_ratio_))
plt.legend('')
plt.grid('on')
plt.xlabel('Principal Components')
plt.ylabel('Explained Varience');
```



In [15]:
```
len(pca.explained_variance_ratio_)
```

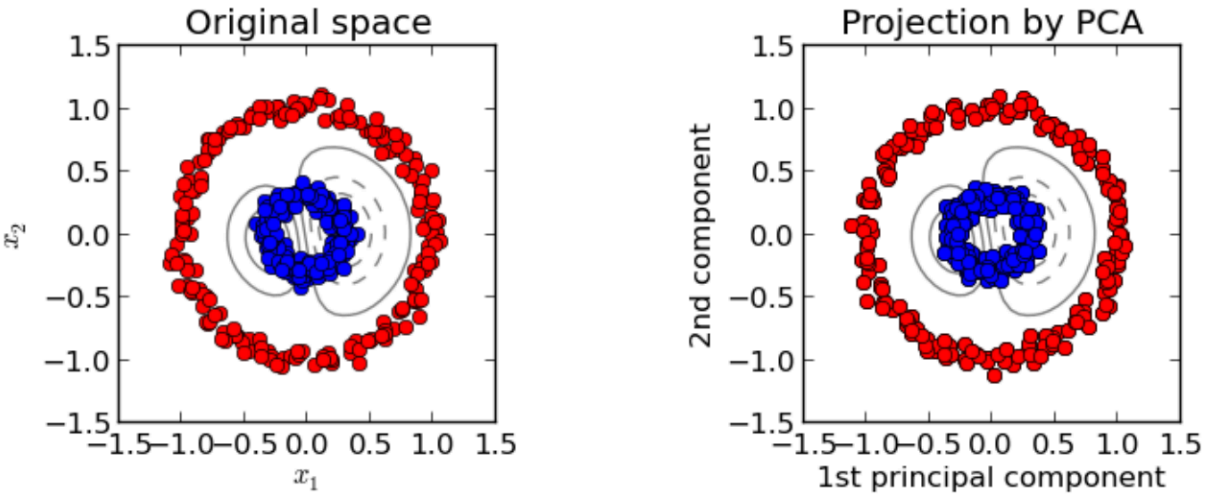
Out[15]: 11

The first 5 Principal Components are capturing around 80% of the variance so we can replace the 11 original features (acidity, residual sugar, chlorides, etc.) with the new 5 features having 80% of the information. So, we have reduced the 11 dimensions to only 5 dimensions while retaining most of the information.



[PCA GIF] https://i.ibb.co/f42F86J/nagesh-pca-5.gif (https://i.ibb.co/f42F86J/nagesh-pca-5.gif)

What if the data is non-linear data?



In [ ]: