Week 4 Lecture

Chapter 3 and 4

February 6, 2018

Chapter 3 - Exploratory Data Analysis

- This material was covered in Week 2
- Skim Chapter 3 in the textbook
- Review Chapter 3 handouts

Jupyter Notebook Topic - Markdown

markdown cheatsheet (https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet)

Markdown Demonstration

Chapter 4 - Dimension Reduction Technique

We will only consider principal components analysis (PCA), not covering factor analysis

Need for Dimension Reducing in Data Mining

- Multicollinearity – a condition where some of the predictor variables are correlated with each other
  - leads to instability in the solution space
    - possibly resulting in incoherent results
- Inclusion of highly correlated variables overemphasizes particular components of the model
- Use of too many predictor variables can
  - unnecessarily complicate interpretation of analysis
  - violates *principle of parsimony*: that one should consider keeping the number of predictors to a size that could be easily interpreted
  - can lead to overfitting
    - generality of findings is hindered since new data does not behave the same way as training data

Need for Dimension Reducing in Data Mining *(cont'd)*

- Analysis solely based at variable level might miss underlying relationships among predictors
  - several predictors might fall into a single group (also called a *factor* or a *component)*
    - ex: savings account balance, checking account balance, home equity, stock portfolio balance, and 401K balance may all fall under a single component *assets*
- Some analysis may require retaining full dimensionality
- Goals of using correlation structure among predictor variables
  - reduce the number of predictor components
  - help ensure that these components are independent
  - provide a framework for interpretability

More on Multicollinearity

Source: Multicollinearity: Definition, Causes, Examples (http://www.statisticshowto.com/multicollinearity/)

- Multicollinearity occurs when there is high correlation among predictor variables
- "It's more common for multicollienarity to rear its ugly head in observational studies; it's less common with experimental data
- Problems:
  - Unstable and unreliabile regression estimates
  - The t-statistics will generally be very small and coefficient confidence intervals will be very wide. This means that it is harder to reject the null hypothesis
  - The partial regression coefficient may be an imprecise estimate; standard errors may be very large
  - Partial regression coefficiencts may have sign and/or magnituded changes as they pass from sample to sample
  - Multicollinearity makes it difficulty to gauge the effect of independent variables on dependent variables

What Causes Multicollinearity

- Two types of multicollinearity
  - Data-based multicollinearity
  - Structural multicollinearity
- Causes of multicollinearity can include:
  - Insufficient data
  - Dummy variables
  - Including a variable that is already a function of existing variables
  - Including two identical (or almost identical) variables

Principle Components Analysis (PCA)

- Explains correlation structure of predictor variables by using a smaller set of linear combinations of these variables
  - linear combinations are called *components*
- Total variability of the dataset of *m* variables can often almost entirely be explained by *k* components, where m>k
  - analyst can replace original m variables with the k<m components
    - modified dataset has n records with k components
- Suppose original variables $X_1, X_2, \ldots, X_m$ form a coordinate system in m-dimensional space
  - principle components represent a new coordinate system found by rotating the original system along the directions of maximum variability

Principle Components Analysis *(cont'd)*

- X_i represents an n x 1 vector where n = number of records
- Z_i represents the standardized n x 1 vector, $Z_i = (X_i - \mu_i)/\sigma_{ii}$
  - μ_i is the mean of X_i, σ_{ii} is the standard deviations of X_i
  - in matrix notation, $Z = (V^{1/2})^{-1}(X - \mu)$, where $^{-1}$ refers to the matrix inverse, and $V^{1/2}$ is a diagonal matrix (nonzero entries only on the diagonal)

m x m standard deviation matrix:

$$\mathbf{V}^{1/2} = \begin{bmatrix} \sigma_{11} & 0 & \cdots & 0 \\ 0 & \sigma_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{mm} \end{bmatrix}$$

symmetric covariance matrix:

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \cdots & \sigma_{1m}^2 \\ \sigma_{12}^2 & \sigma_{22}^2 & \cdots & \sigma_{2m}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{1m}^2 & \sigma_{2m}^2 & \cdots & \sigma_{mm}^2 \end{bmatrix}$$

where σ^2_{ij}, $i \neq j$ refers to the *covariance* between X_i and X_j

$$\sigma_{ij}^2 = \sum_{k=1}^{n} (x_{ki} - \mu_i)(x_{kj} - \mu_j)/n$$

Principle Components Analysis *(cont'd)*

- Covariance: a measure of the degree to which two variables vary together
  - positive covariance indicates that when one variable increases, the other also increases
  - negative covariance indicates that when one variable increases, the other *decreases*
  - σ^2_{ij} denotes the variance of X_{ij}
    - when X_i and X_j are independent, $\sigma^2_{ij}=0$
    - when $\sigma^2_{ij}=0$, X_i and X_j are NOT necessarily independent
  - covariance measure is not scaled, so changing units of measure would not change the value of the covariance
- Correlation Coefficient (r_{ij}): scales the covariance by each of the standard deviations $r_{ij} = \frac{\sigma^2_{ij}}{\sigma_{ii}\,\sigma_{jj}}$

Principle Components Analysis *(cont'd)*

- Correlation matrix is denoted as ρ (rho, the Greek letter for r)

$$\rho = \begin{bmatrix} \frac{\sigma^2_{11}}{\sigma_{11}\sigma_{11}} & \frac{\sigma^2_{12}}{\sigma_{11}\sigma_{22}} & \cdots & \frac{\sigma^2_{1m}}{\sigma_{11}\sigma_{mm}} \\ \frac{\sigma^2_{12}}{\sigma_{11}\sigma_{22}} & \frac{\sigma^2_{22}}{\sigma_{22}\sigma_{22}} & \cdots & \frac{\sigma^2_{2m}}{\sigma_{22}\sigma_{mm}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\sigma^2_{1m}}{\sigma_{11}\sigma_{mm}} & \frac{\sigma^2_{2m}}{\sigma_{22}\sigma_{mm}} & \cdots & \frac{\sigma^2_{mm}}{\sigma_{mm}\sigma_{mm}} \end{bmatrix}$$

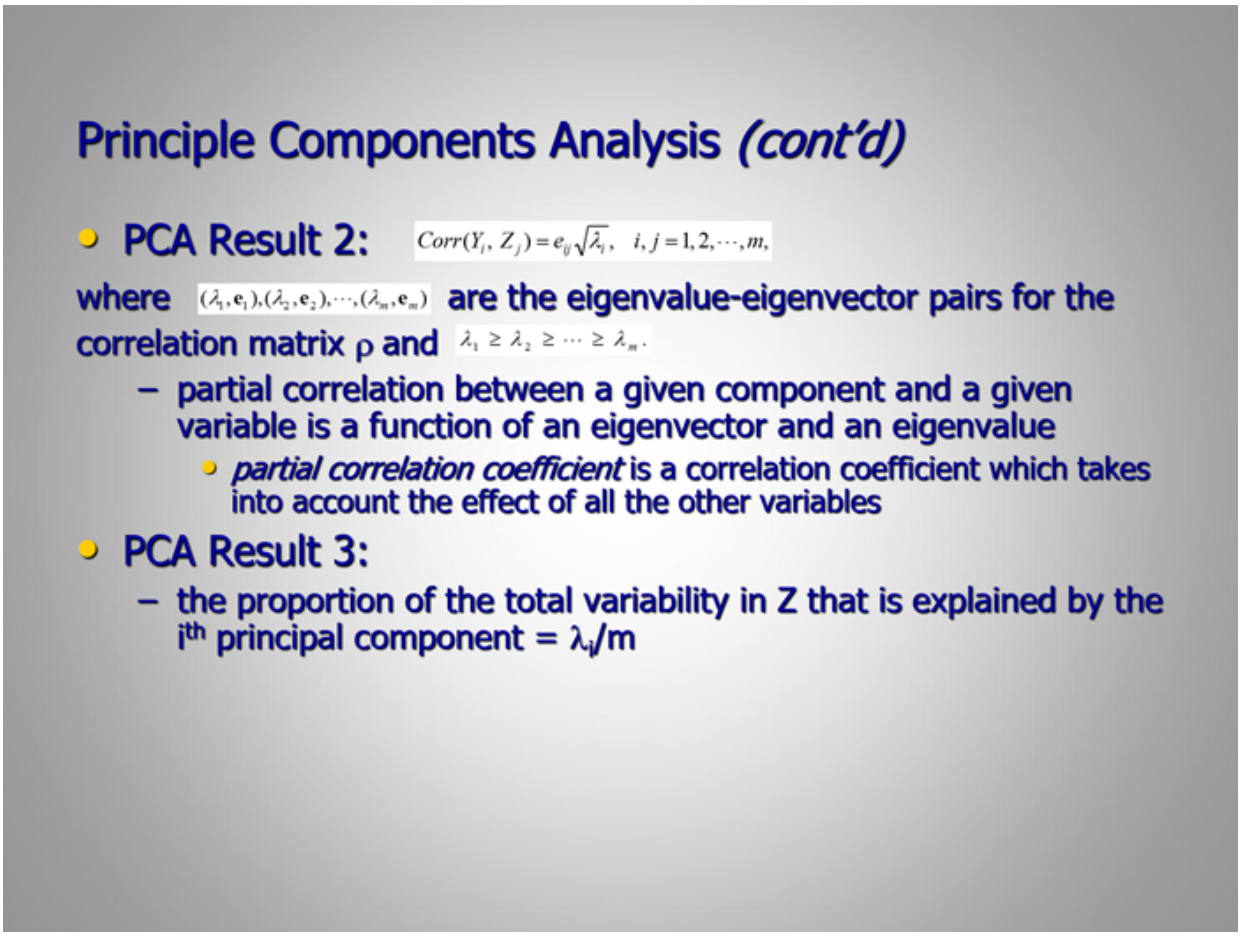
- Properties of the standardized data matrix
  - E(Z)=0, where 0 denotes and n x 1 vector of zeros
  - Z has covariance matrix $\mathbf{Z} = \left(\mathbf{V}^{1/2}\right)^{-1}(\mathbf{X}-\boldsymbol{\mu}).$
    - The two points above imply that for the standardized data set, covariance matrix = correlation matrix $Cov(\mathbf{Z}) = \left(\mathbf{V}^{1/2}\right)^{-1}\,\boldsymbol{\Sigma}\,\left(\mathbf{V}^{1/2}\right)^{-1} = \boldsymbol{\rho}$
- i^{th} principle component of standardized data matrix $\mathbf{Z} = [Z_1, Z_2, \cdots, Z_m]$ = $Y_i = \mathbf{e}_i'\mathbf{Z},$ where e_i refers to the i^{th} eigenvector and e_i' refers to the transpose of e_i

Principle Components Analysis *(cont'd)*

- Principle Components: linear combinations Y_1, Y_2,..., Y_n of standardized variables Z such that
  - variances of Y_i are as large as possible
  - the Y_i are uncorrelated
- First principle component: $Y_1 = \mathbf{e}_1'\mathbf{Z} = e_{11}Z_1 + e_{12}Z_2 + \cdots + e_{1m}Z_m$
  - greater variability than any other possible combination of the Z variables
  - $Y_1 = \mathbf{e}_1'\mathbf{Z}$ maximizes $Var(Y_1) = \mathbf{e}_1'\boldsymbol{\rho}\mathbf{e}_1$
- 2nd principle component: linear combination
  - independent of Y_1 $Y_2 = \mathbf{e}_2'\mathbf{Z}$
  - maximizes $Var(Y_2) = \mathbf{e}_2'\boldsymbol{\rho}\mathbf{e}_2$

Principle Components Analysis *(cont'd)*

- i^{th} principle component: linear combination $Y_i = \mathbf{e}_i'\mathbf{X}$
  - independent of all other principle components Y_j, where j < i
  - Maximizes $Var(Y_i) = \mathbf{e}_i'\boldsymbol{\rho}\mathbf{e}_i$
- Eigenvalues: scalars (λ_1, λ_2,.., λ_m), numbers of dimension 1x1, of **B** such that $|\mathbf{B} - \lambda\mathbf{I}| = 0$, where **B** is an m x m matrix and **I** is the identity matrix
- Eigenvectors: nonzero m x 1 vector **e** such that $\mathbf{Be} = \lambda\mathbf{e}$, where **B** is an m x m matrix, and λ is an eigenvalue of **B**
- PCA Result 1: $\sum_{i=1}^{m} Var(Y_i) = \sum_{i=1}^{m} Var(Z_i) = \sum_{i=1}^{m} \lambda_i = m.$
  - total variability in the standardized data set equals the sum of the variances for each Z-vector = the sum of the variances for each component = the sum of the eigenvalues = the number of variables



Python Support for PCA

In [17]:
```
import pandas as pd
from IPython.display import HTML, display
display(HTML("<h1>Loading the House Data Set</h1>"))

houses= pd.read_table('cadata_dataOnly.txt', delim_whitespace=True, names=('media
#display(HTML("<h2>Check for missing data"))
#houses.isnull().sum()
display(HTML("<h2>Descriptive Statistics"))
houses.describe()
```

Loading the House Data Set

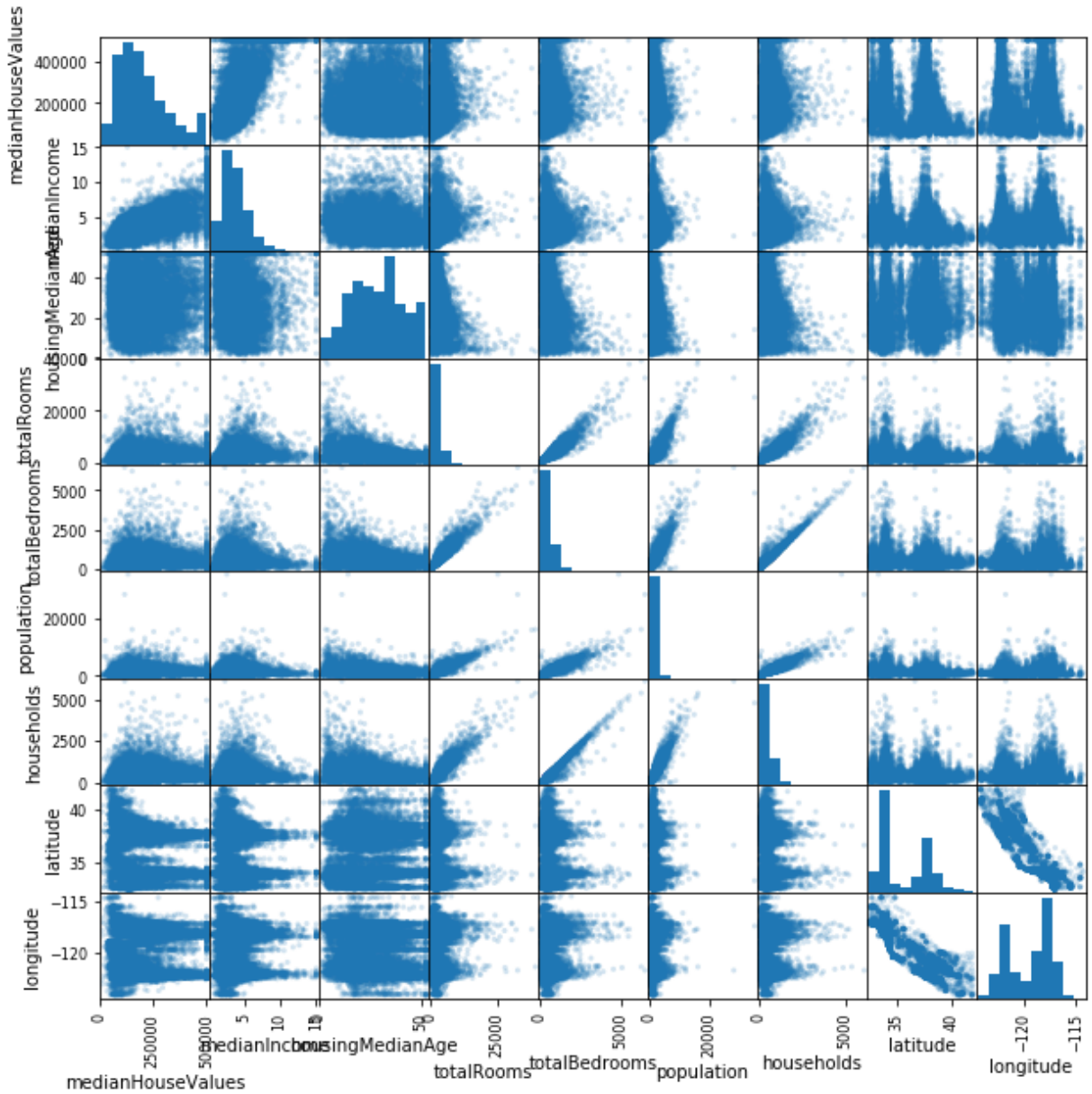
Descriptive Statistics

Out[17]:

	medianHouseValues	medianIncome	housingMedianAge	totalRooms	totalBedrooms	pop
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640
mean	206855.816909	3.870671	28.639486	2635.763081	537.898014	1425
std	115395.615874	1.899822	12.585558	2181.615252	421.247906	1132
min	14999.000000	0.499900	1.000000	2.000000	1.000000	3
25%	119600.000000	2.563400	18.000000	1447.750000	295.000000	787
50%	179700.000000	3.534800	29.000000	2127.000000	435.000000	1166
75%	264725.000000	4.743250	37.000000	3148.000000	647.000000	1725
max	500001.000000	15.000100	52.000000	39320.000000	6445.000000	35682

In [18]:
```
import matplotlib.pyplot as plt
display(HTML("<h1>Scatter Matrix</h1>"))
pd.plotting.scatter_matrix(houses,alpha=0.2, figsize=(10,10),diagonal='hist')
plt.show()
```

Scatter Matrix



In [19]:
```
display(HTML("<h1>Correlation Matrix</h1>"))
houses.corr()
```

Correlation Matrix

Out[19]:

	medianHouseValues	medianIncome	housingMedianAge	totalRooms	totalBedro
medianHouseValues	1.000000	0.688075	0.105623	0.134153	0.05
medianIncome	0.688075	1.000000	-0.119034	0.198050	-0.00
housingMedianAge	0.105623	-0.119034	1.000000	-0.361262	-0.32
totalRooms	0.134153	0.198050	-0.361262	1.000000	0.92
totalBedrooms	0.050594	-0.008093	-0.320485	0.929893	1.00
population	-0.024650	0.004834	-0.296244	0.857126	0.87
households	0.065843	0.013033	-0.302916	0.918484	0.97
latitude	-0.144160	-0.079809	0.011173	-0.036100	-0.06
longitude	-0.045967	-0.015176	-0.108197	0.044568	0.06

In [20]:
```
from sklearn import preprocessing
display(HTML("<h1>Standardization</h1>"))
std_scaler=preprocessing.StandardScaler().fit(houses.values)
houses_scaled=std_scaler.transform(houses)
houses_scaled_df=pd.DataFrame(houses_scaled,index=houses.index,columns=houses.col
houses_scaled_df.describe()
```

Standardization

Out[20]:

	medianHouseValues	medianIncome	housingMedianAge	totalRooms	totalBedrooms	p
count	2.064000e+04	2.064000e+04	2.064000e+04	2.064000e+04	2.064000e+04	2.06
mean	8.950635e-16	3.624093e-16	8.557001e-16	1.475181e-16	-6.450213e-17	-6.46
std	1.000024e+00	1.000024e+00	1.000024e+00	1.000024e+00	1.000024e+00	1.00
min	-1.662641e+00	-1.774299e+00	-2.196180e+00	-1.207283e+00	-1.274573e+00	-1.25
25%	-7.561633e-01	-6.881186e-01	-8.453931e-01	-5.445698e-01	-5.766293e-01	-5.63
50%	-2.353337e-01	-1.767951e-01	2.864572e-02	-2.332104e-01	-2.442754e-01	-2.29
75%	5.014973e-01	4.593063e-01	6.643103e-01	2.348028e-01	2.590034e-01	2.64
max	2.540411e+00	5.858286e+00	1.856182e+00	1.681558e+01	1.402320e+01	3.02

PCA in sklearn

sklearn.decomposition.PCA (http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html)

class `sklearn.decomposition.` **PCA** (*n_components=None, copy=True, whiten=False, svd_solver='auto', tol=0.0, iterated_power='auto', random_state=None*) [source]

Principal component analysis (PCA)

Linear dimensionality reduction using Singular Value Decomposition of the data to project it to a lower dimensional space.

It uses the LAPACK implementation of the full SVD or a randomized truncated SVD by the method of Halko et al. 2009, depending on the shape of the input data and the number of components to extract.

It can also use the scipy.sparse.linalg ARPACK implementation of the truncated SVD.

Notice that this class does not support sparse input. See `TruncatedSVD` for an alternative with sparse data.

Read more in the User Guide.

Methods for PCA

Methods

`fit` (X[, y])	Fit the model with X.
`fit_transform` (X[, y])	Fit the model with X and apply the dimensionality reduction on X.
`get_covariance` ()	Compute data covariance with the generative model.
`get_params` ([deep])	Get parameters for this estimator.
`get_precision` ()	Compute data precision matrix with the generative model.
`inverse_transform` (X)	Transform data back to its original space.
`score` (X[, y])	Return the average log-likelihood of all samples.
`score_samples` (X)	Return the log-likelihood of each sample.
`set_params` (**params)	Set the parameters of this estimator.
`transform` (X)	Apply dimensionality reduction to X.

Attributes of PCA

Attributes:

components_ : array, shape (n_components, n_features)

Principal axes in feature space, representing the directions of maximum variance in the data. The components are sorted by `explained_variance_`.

explained_variance_ : array, shape (n_components,)

The amount of variance explained by each of the selected components.

Equal to n_components largest eigenvalues of the covariance matrix of X.

New in version 0.18.

explained_variance_ratio_ : array, shape (n_components,)

Percentage of variance explained by each of the selected components.

If `n_components` is not set then all components are stored and the sum of explained variances is equal to 1.0.

singular_values_ : array, shape (n_components,)

The singular values corresponding to each of the selected components. The singular values are equal to the 2-norms of the `n_components` variables in the lower-dimensional space.

mean_ : array, shape (n_features,)

Per-feature empirical mean, estimated from the training set.

Equal to X.mean(axis=0).

n_components_ : int

The estimated number of components. When n_components is set to 'mle' or a number between 0 and 1 (with svd_solver == 'full') this number is estimated from input data. Otherwise it equals the parameter n_components, or n_features if n_components is None.

noise_variance_ : float

The estimated noise covariance following the Probabilistic PCA model from Tipping and Bishop 1999. See "Pattern Recognition and Machine Learning" by C. Bishop, 12.2.1 p. 574 or http://www.miketipping.com/papers/met-mppca.pdf. It is required to computed the estimated data covariance and score samples.

Equal to the average of (min(n_features, n_samples) - n_components) smallest eigenvalues of the covariance matrix of X.

In [21]:
```
import numpy as np
display(HTML("<h1>PCA</h1>"))
from sklearn import decomposition
#set up and execute PCA
pca=decomposition.PCA(n_components=9)
pca.fit(houses_scaled_df)
#display eigen values
display(HTML("<h2>Eigenvalues (sorted)</h2>"))
print(pca.explained_variance_)
#display explained variance ratio
display(HTML("<h2> Percent variability explained by each component"))
print(pca.explained_variance_ratio_)
#display cumulative sum of variability explained
display(HTML("<h2> Cumulative Percent variability explained by each component"))
np.cumsum(pca.explained_variance_ratio_)
```

PCA

Eigenvalues (sorted)

```
[ 3.91243965  1.92267742  1.69686478  0.91022813  0.29332659  0.14252168
  0.0626448   0.04454859  0.01474837]
```

Percent variability explained by each component

```
[ 0.43471552  0.21363082  0.18854053  0.10113646  0.03259184  0.01583574
  0.00696053  0.00494984  0.00163871]
```

Cumulative Percent variability explained by each component

Out[21]:
```
array([ 0.43471552,  0.64834634,  0.83688687,  0.93802333,  0.97061517,
        0.98645092,  0.99341145,  0.99836129,  1.        ])
```

In [22]:
```
display(HTML("<h1>Eigen Vectors</h1>"))
print(pca.components_)
```

Eigen Vectors

```
[[  4.58276345e-02   5.59051837e-02  -2.16566691e-01   4.84411104e-01
    4.89571749e-01   4.70228632e-01   4.91023110e-01  -7.52147930e-02
    7.52513481e-02]
 [ -1.90740261e-01  -1.77636178e-01   6.70198122e-04   6.02403977e-02
    7.17162828e-02   4.52647039e-02   7.13924241e-02   6.90791489e-01
   -6.62534760e-01]
 [  6.73804525e-01   6.71384881e-01   4.84249423e-02   8.46557088e-02
   -4.18212791e-02  -7.87592247e-02  -2.71062882e-02   1.26764630e-01
   -2.46905209e-01]
 [  1.67306361e-01  -2.09669988e-01   9.31336445e-01   2.85571966e-02
    1.24082805e-01   1.20326124e-01   1.44982844e-01  -8.55628067e-02
   -4.03491353e-02]
 [ -6.41689351e-01   6.34413078e-01   2.73255298e-01   1.62568208e-01
   -1.42930408e-01   1.99023340e-01  -1.03328251e-01   7.29999439e-02
    9.28053478e-02]
 [  1.19553166e-01  -3.60106165e-02  -8.49632076e-02  -4.04895038e-01
   -3.29460866e-01   8.04509339e-01  -7.83971155e-02  -1.29533082e-01
   -1.85867888e-01]
 [  2.04953497e-01  -2.42571959e-01   1.73736470e-02   6.01371338e-01
   -2.99055706e-01   2.09477268e-01  -4.80101519e-01   2.92726142e-01
    2.96146036e-01]
 [ -1.06360881e-01  -7.46920057e-02  -3.39158340e-02   4.21809661e-01
   -1.49384910e-01  -9.51219179e-02  -1.75783345e-01  -6.19382754e-01
   -5.98444636e-01]
 [ -1.17161618e-02   4.90312937e-02   5.15150091e-03  -1.54258173e-01
    7.05199961e-01   1.27985456e-01  -6.73665023e-01  -4.82512955e-02
   -6.16159801e-02]]
```

In [23]:
```
display(HTML("<h1>Creating Transformed Variables</h1>"))
houses_scaled_pca=pca.transform(houses_scaled_df)
houses_scaled_pca_df=pd.DataFrame(houses_scaled_pca,index=houses.index,columns=['
houses_scaled_pca_df.describe()
```

Creating Transformed Variables

Out[23]:

	pca1	pca2	pca3	pca4	pca5	pca6	
count	2.064000e+04	2.064000e+04	2.064000e+04	2.064000e+04	2.064000e+04	2.064000e+04	2
mean	7.351798e-16	4.146124e-16	7.355496e-16	1.173480e-16	1.169069e-16	-1.028462e-17	-
std	1.978037e+00	1.386640e+00	1.302669e+00	9.540819e-01	5.416095e-01	3.775296e-01	
min	-3.063244e+00	-3.238056e+00	-3.575548e+00	-3.547725e+00	-3.072455e+00	-5.492365e+00	-
25%	-1.142175e+00	-1.137869e+00	-9.230316e-01	-7.193740e-01	-2.785373e-01	-1.568887e-01	-
50%	-4.460699e-01	-5.598973e-01	-2.673424e-01	5.487480e-02	6.027322e-02	-1.604570e-02	
75%	5.765479e-01	1.324022e+00	6.828602e-01	6.451502e-01	3.421856e-01	1.405192e-01	
max	3.206799e+01	5.312076e+00	6.308914e+00	5.877087e+00	4.763907e+00	1.592432e+01	4

In [24]:
```
display(HTML("<h1>Correlation Matrix of PCA</h1>"))
houses_scaled_pca_df.corr()
```

Correlation Matrix of PCA

Out[24]:

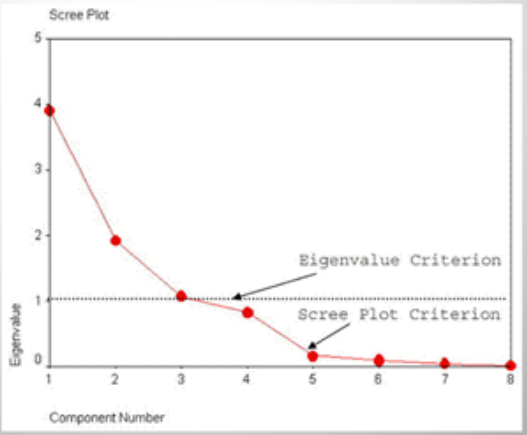
	pca1	pca2	pca3	pca4	pca5	pca6	
pca1	1.000000e+00	3.196660e-16	-8.674277e-17	2.028991e-16	-9.458680e-16	-5.055642e-16	1.871
pca2	3.196660e-16	1.000000e+00	8.580671e-16	-3.840298e-16	3.349758e-16	-1.524949e-16	-8.799
pca3	-8.674277e-17	8.580671e-16	1.000000e+00	3.678750e-16	-1.937610e-15	9.474956e-16	1.537
pca4	2.028991e-16	-3.840298e-16	3.678750e-16	1.000000e+00	-2.990074e-15	-6.295891e-16	-3.5
pca5	-9.458680e-16	3.349758e-16	-1.937610e-15	-2.990074e-15	1.000000e+00	-9.443345e-16	-1.0
pca6	-5.055642e-16	-1.524949e-16	9.474956e-16	-6.295891e-16	-9.443345e-16	1.000000e+00	6.200
pca7	1.871154e-16	-8.799115e-16	1.537216e-15	-3.525535e-16	-1.047168e-16	6.200372e-16	1.000(
pca8	2.029031e-15	1.266558e-15	2.191277e-16	-3.238712e-16	-5.176663e-17	6.514340e-18	9.938
pca9	9.106650e-16	4.388415e-18	-4.598238e-16	6.198808e-16	1.902365e-16	-7.243045e-16	1.875

How many components should we extract?

- Four criteria are:
  - The Eigenvalue Criterion
  - The Proportion of Variance Explained Criterion
  - The Minimum Communality Criterion
  - The Scree Plot Criterion
- The Eigenvalue Criterion
  - PCA Result 1: sum of eigenvalues represents the number of input variables into the model
    - An eigenvalue of '1' represents one variable's worth of information
  - Keep components only if eigenvalues > 1
  - However, problem when n < 20 or n > 50 as this criteria keeps to few or too many components, respectively
- The Proportion of Variance Explained Criterion
  - Step 1: specify how much variability to keep
  - Step 2: keep the number of components reflecting this variability

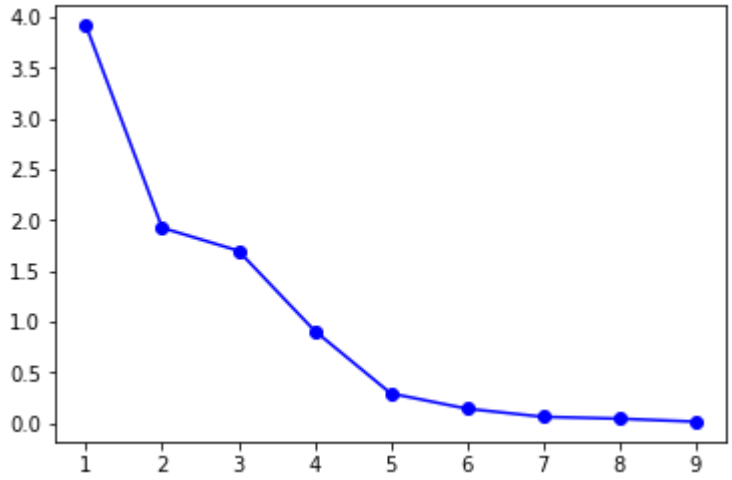
How many components should we extract? *(cont'd)*

- The Scree Plot Criterion
  - Scree plot: a graphical plot of the eigenvalues against the component number
    - Useful for finding an upper bound for the # of components to keep
  - Keep variables before the bend in the "elbow" of the graph



In [25]:
```
display(HTML("<h1>Scree Plot using Python</h1>"))
plt.plot(np.arange(1,10,1),pca.explained_variance_,'-bo')
plt.show()
```

Scree Plot using Python



Profiling the Principle Components

- Picture:
  - component matrices for extracting 3 & 4 components
  - weights smaller than 0.15 are suppressed to ease component interpretation

Component Matrix[a]

	Component		
	1	2	3
MINC_Z			.922
HAGE_Z	-.429		-.407
ROOMS_Z	.956		
BEDRMS_Z	.970		
POPN_Z	.933		
HHLDS_Z	.972		
LAT_Z		.970	
LONG_Z		-.969	

Extraction Method: PCA

a. 3 components extracted.

Component Matrix[a]

	Component			
	1	2	3	4
MINC_Z			.922	.370
HAGE_Z	-.429		-.407	.806
ROOMS_Z	.956			
BEDRMS_Z	.970			
POPN_Z	.933			
HHLDS_Z	.972			
LAT_Z		.970		
LONG_Z		-.969		

Extraction Method: PCA

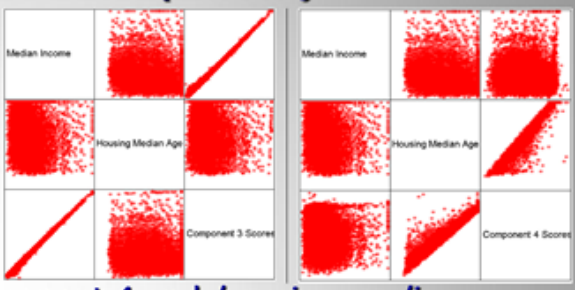
a. 4 components extracted.

- Principal Component 1: "block group size" type variables
  - *total rooms*, *total bedrooms*, *population*, *households* are either large or small together
  - large block groups - large values for all four variables
  - small block groups - small values for all four variables
  - *Median housing age* smaller, counterweight to these four variables
- Principal Component 2: "geographical" component
  - *latitude* and *longitude* only - strongly negatively correlated (opposite sign of the weights)

Profiling the Principle Components *(cont'd)*

- Principal Component 3: *median income* of the block group
  - smaller effect for *housing median age* of block group
- Principal Component 4:
  - *housing median age* and *median income* again
    - negative correlation been accounted for (component 3)
    - remaining variability is positive
- Consider *factor scores* to understand the relationship between component 3 & 4
  - estimated values of the factors for each observation
  - Based on Factor Analysis
- Look at a matrix plot of these components against the individual variables that make up the component

Profiling the Principle Components *(cont'd)*



- Figure on left:
  - Strong correlation between component 3 and *median income*
- Figure on right:
  - Strong correlation between component 4 and *housing median age*
- Component weight represents the correlation between the component and the variable
  - squared component weight = amount of the variable's total variability explained
  - Component weight should be > .50 (or < -.50) to be considered significant
  - threshold value of +/- .50 requires that at least 25% of the variable's variance be explained by a particular component

Profiling the Principle Components *(cont'd)*

- Removing all variables whose weight is < +/- .50
  - Principal Component 1: "block group size" component
    - four variables: *total rooms*, *total bedrooms*, *population*, and *households*
  - Principal Component 2: "geographical" component
    - two variables: *latitude* and *longitude*
  - Principal Component 3: "income" component
    - one variable: *median income*
  - Principal Component 4: "housing age" component
    - one variable: *housing median age*
- Note: partition of variables among the four components
  - mutually exclusive: no variable is shared (after suppression) between any two components
  - exhaustive: all variables are contained in the four components
- Note: weight loading (positive versus negative) can be reversed and interpretation remains the same

Communalities

- Communality: proportion of variance of a particular variable that is shared with other variables
  - represents the overall importance of each of the variables in the PCA as a whole
    - large communality values indicate that PCA covers a large proportion of the variability of the variables
    - small communality values indicate that there is still a lot of variance not accounted for by the principle components
  - Calculation: sum of squares of the component weights across all components
    - calculated for individual variables
      - Communality for *housing median age*, three components: $(-0.429)^2 + (0.025)^2 + (-0.407)^2 = 0.350315$
    - communalities less than 0.5 are considered too low
- Minimum Communality Criterion
  - Step 1: identify the set of variables are to be retained
  - Step 2: calculate the communalities of these variables
    - should exceed a certain threshold (such as 50%)

Communalities *(cont'd)*

- Evaluation of PCA component selection for *Houses* data set
  - The Eigenvalue Criterion: recommends 3 components
    - rejection not absolute on the fourth component
    - for small numbers of variables, this criterion can underestimate the best number of components to extract
  - The Proportion of Variance Explained Criterion: 4 components
    - accounts for 96% of the variability
  - The Scree Plot Criterion: 4 components
  - The Minimum Communality Criterion: 4 components if *housing median age* is to remain in the analysis
    - Intention: substitute the components for the original data
      - need to keep this variable

Decision: Retain first four components

Validation of Principle Components *(cont'd)*

- Original data set was split into a training and testing data set
- Standardize and perform PCA on testing data set
- Components should extract a one-to-one correspondence with the variables expressed in the prior analysis
  - Component weights need not match perfectly
- If validation not proved by above method
  - Results are not gerneralizable and should not be reported as valid
  - If lack of validation stems from the subset of variables, omit these variables and try PCA again