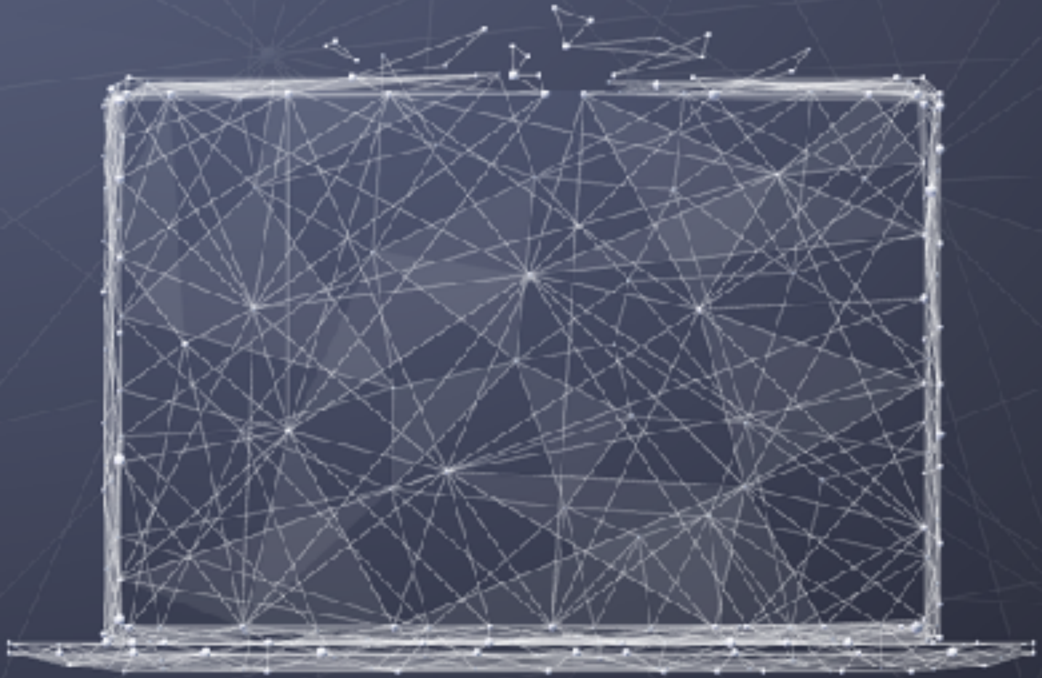
Data Science
Data Engineering I

Analysis and visualization



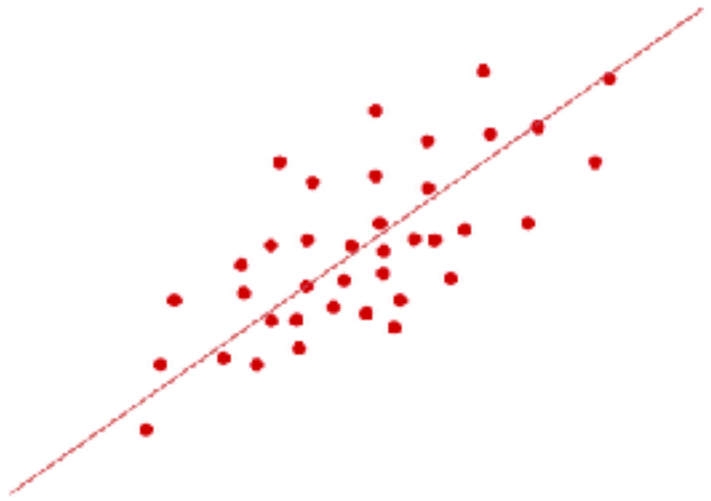
PURDUE UNIVERSITY | College of Science

Dimensionality reduction

- High-dimensional data (i.e., examples represented by many, many attributes) will often have simpler structure, e.g.
  - Topics in documents/products
  - Product characteristics (reliability, cost etc.)
  - User preferences/demographics
- This simpler structure can often be approximated using a lower-dimensional representation, i.e., projecting objects into different space w/fewer attributes

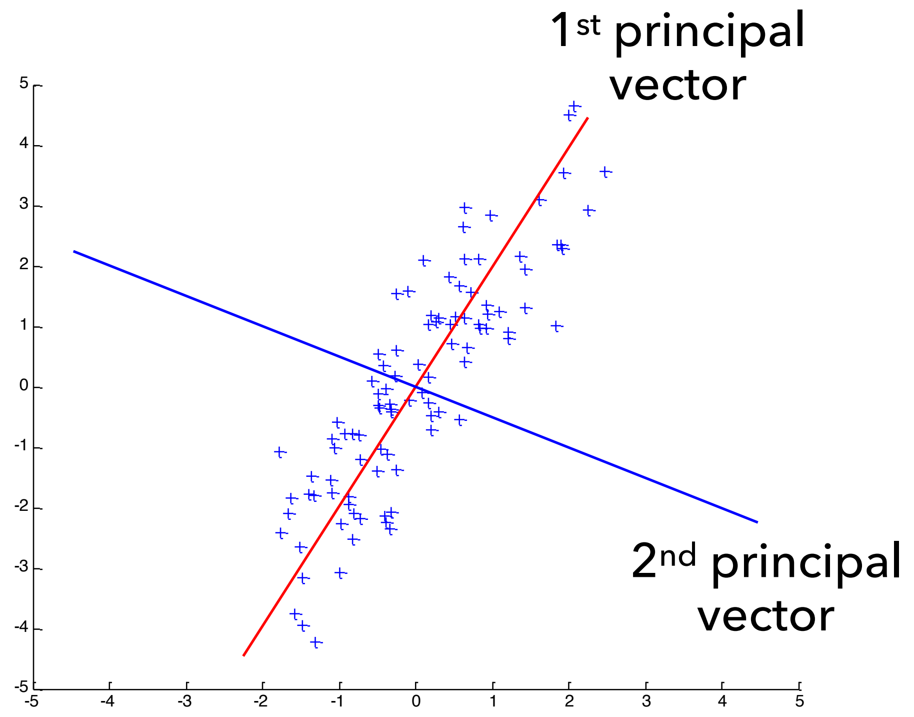
Dimensionality reduction (cont)

- Usually, the data vary more strongly in some dimensions than others
- Fitting a linear model to the strongest directions gives a good approximation to the data



Principal component analysis (PCA)

- PCA is a linear dimensionality reduction method dating back to Pearson in 1901
- One of the most useful techniques in exploratory data analysis
- PCA finds the directions (principal components) that maximize the variance of the data
- Principal vectors are orthogonal, gives best axes to project data onto



PCA: High level approach

- Given data matrix X with p dimensions (attributes)
  - Mean center each column of X
    (ie. subtract from each attribute value the mean of that attribute)
  - Compute the p x p covariance matrix: $\Sigma = X^T X$
  - Find eigenvectors and eigenvalues of the symmetric covariance matrix Σ
- The eigenvectors correspond to the principle components of X
- Select the m eigenvectors with largest eigenvalues to reduce dimensionality

Applying PCA

- New data vectors are formed by projecting the data onto the first few principal components (i.e., top m eigenvectors)

$$\mathbf{x} = [x_1, x_2, \ldots, x_p] \quad \text{(original instance)}$$

$$\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_p] \quad \text{(principal components)}$$

$$x'_1 = \mathbf{a}_1 \mathbf{x} = \sum_{j=1}^{p} a_{1j} x_j$$

$$\ldots$$

$$x'_m = \mathbf{a}_m \mathbf{x} = \sum_{j=1}^{p} a_{mj} x_j \quad for\ m < p$$

If m=p then data is transformed to new basis
If m<p transformation is lossy, dimensionality is reduced

$$\mathbf{x}' = [x'_1, x'_2, \ldots, x'_m] \quad \text{(transformed instance)}$$

PURDUE UNIVERSITY College of Science

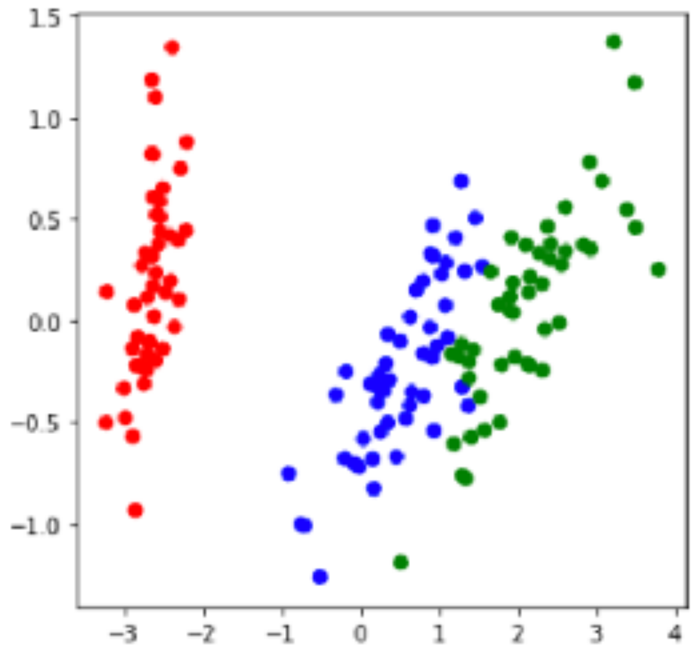
How to apply PCA in python

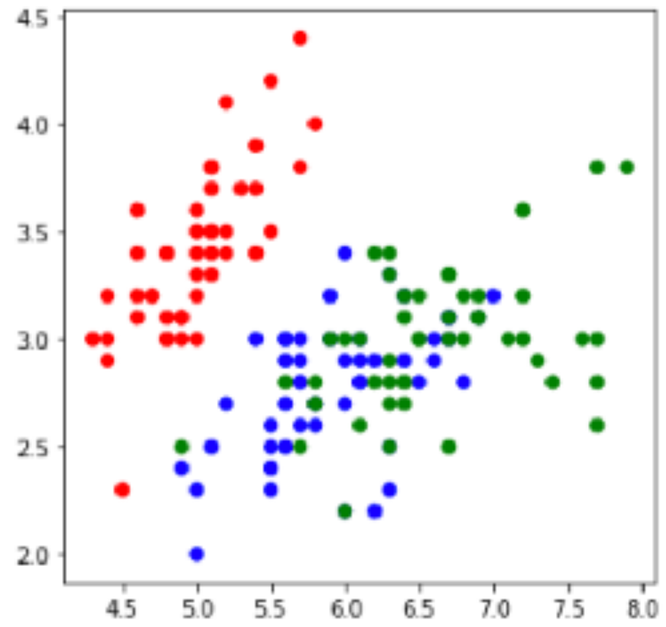
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import decomposition; from sklearn import preprocessing

# read in data and mean center
data = pd.read_csv('iris.dat')
X = data.iloc[:,0:4].values
X_scaled = preprocessing.scale(X, with_std=False)

# initialize PCA model with four components
pca = decomposition.PCA(n_components=4)
# fit PCA model with scaled data and output transformed data
X_trans = pca.fit_transform(X_scaled)
```

PURDUE UNIVERSITY | College of Science

Visualizing PCA results

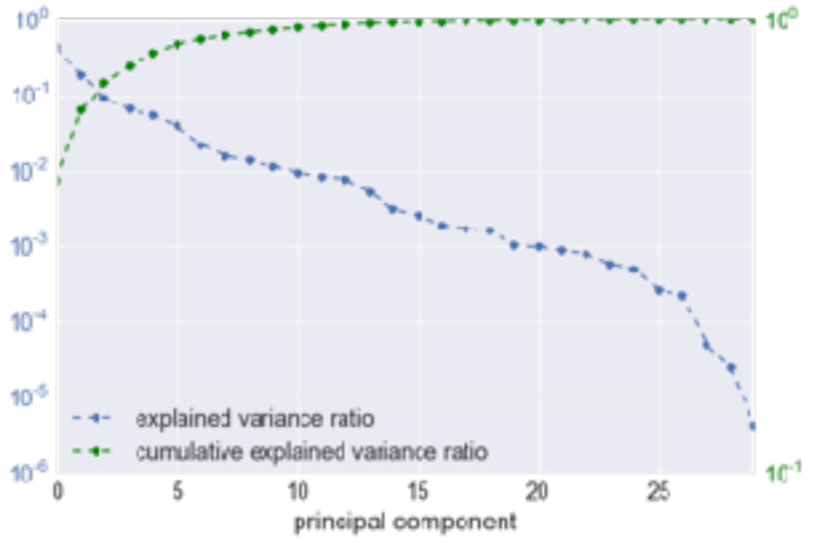
```
# map each category to a different color
data['colors']=data.category.map({'Iris-setosa':'r', 'Iris-versicolor':'b', 'Iris-virginica':'g'})
# visualize results for 1st two dimensions
plt.scatter(X_trans[:,0],X_trans[:,1],c=data[‘colors'])
# compare to 1st two dimensions in original data
plt.scatter(X[:,0],X[:,1],c=data['colors'])
```





How to choose number of dimensions

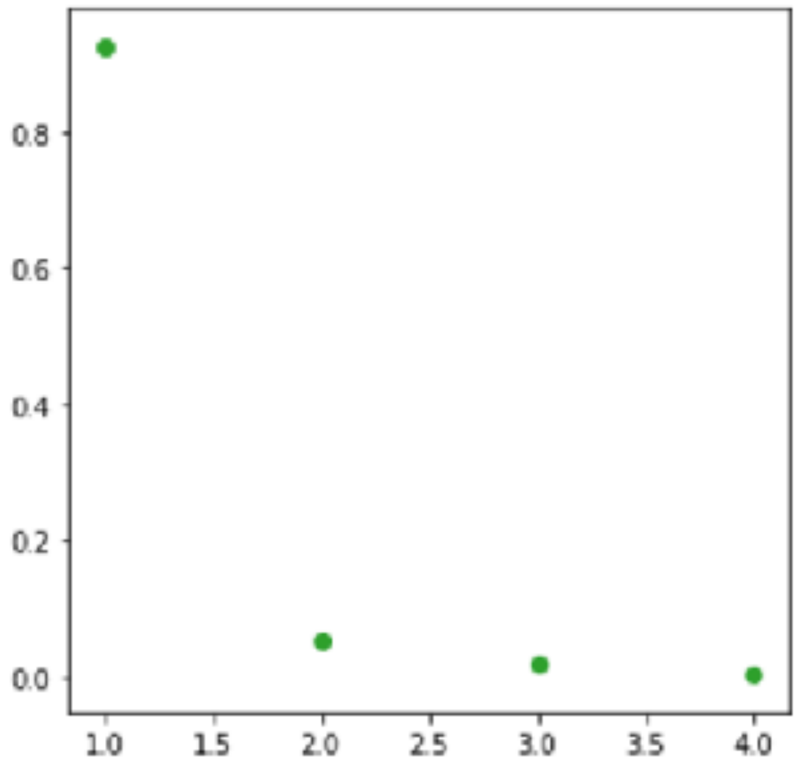
- In PCA, each transformed dimension explains a proportion of the variance of the data
- As m (the number of components) increases the total explained variance cannot increase
- Look at the total proportion of variance explained by each dimension to choose number of dimensions
- Pick number that provides a good tradeoff between accuracy (large proportion of variance explained) and complexity (small number of dimensions)



PURDUE UNIVERSITY. College of Science

How to apply PCA (cont)

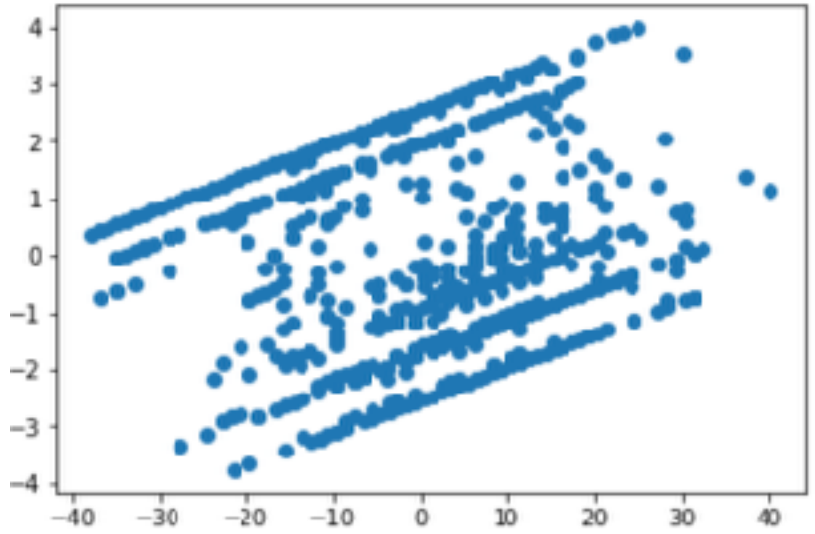
```
# plot explained variance vs. dimension number
# to choose number of dimensions
plt.scatter(range(1,5),pca.explained_variance_ratio_)
```

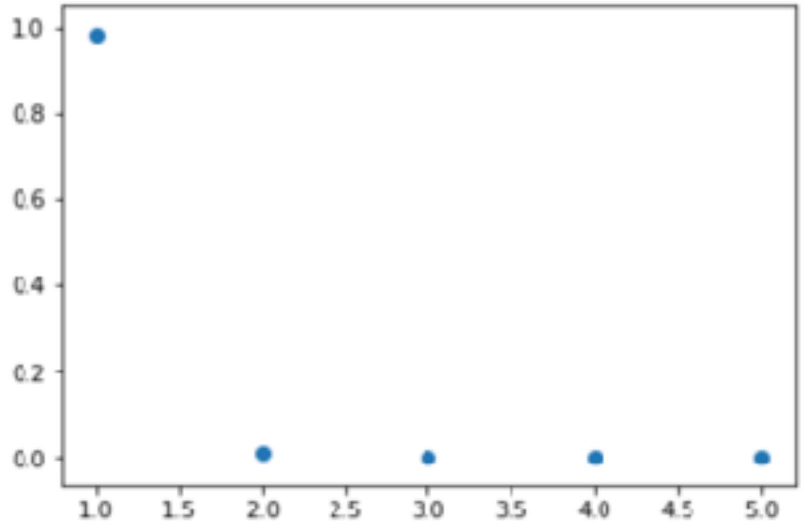


PCA component

Another PCA example

```
data = pd.read_csv('mammographic_masses.data')
data2 = data.dropna()

X = data2.iloc[:,1:7].values
X_scaled = preprocessing.scale(X, with_std=False)

pca = decomposition.PCA(n_components=5)
X_trans = pca.fit_transform(X_scaled)

plt.scatter(X_trans[:,0],X_trans[:,1])
plt.scatter(range(1,6),pca.explained_variance_ratio_)
```



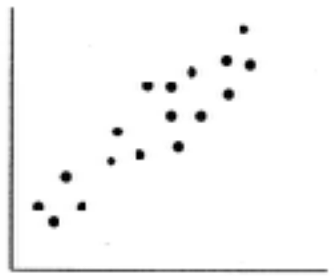


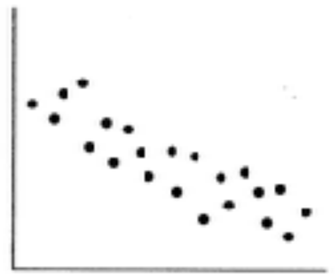
Covariance

- Measures how variables X and Y vary together

$$COV(x,y) = \frac{1}{n}\sum_{i=1}^{n}(x(i)-\bar{x})(y(i)-\bar{y})$$

- Positive if large values of X are associated with large values of Y
  Negative if large values of X are associated with small values of Y


(a)


(b)

Measures linear relationship

- Covariance matrix (∑): Symmetric matrix of covariances for p variables
  - Note: Diagonal values record the variance of each variable

Covariance example

```
# covariance for mammography data
data2.cov()
```

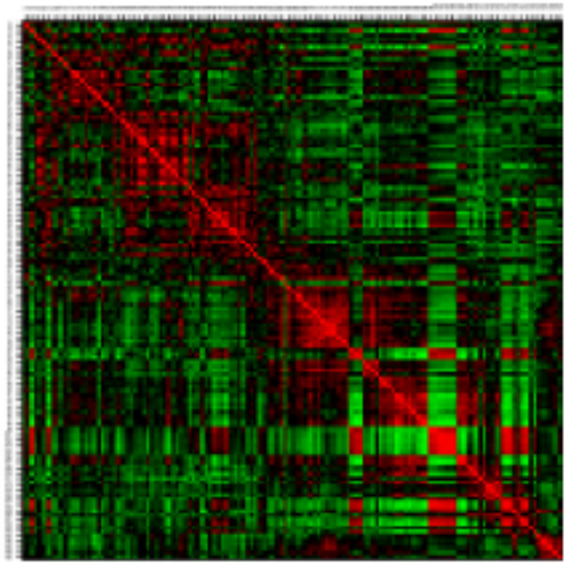
	BIRADS	Age	Shape	Margin	Density	Severity
BIRADS	3.569809	2.616550	0.422919	0.465869	0.018753	0.211251
Age	2.616550	215.369000	6.933671	9.660191	0.269067	3.337083
Shape	0.422919	6.933671	1.545267	1.438069	0.032266	0.351172
Margin	0.465869	9.660191	1.438069	2.453221	0.068530	0.449436
Density	0.018753	0.269067	0.032266	0.068530	0.123296	0.012010
Severity	0.211251	3.337083	0.351172	0.449436	0.012010	0.250074

Correlation

- Covariance measures the direction of the relationship between two variables, but it depends on the units of X and Y (i.e., ranges)
- Correlation standardizes covariance by dividing through standard deviations

$$\rho(x, y) = \frac{\frac{1}{n}\sum_{i=1}^{n}(x(i)-\bar{x})(y(i)-\bar{y})}{\sigma_x \sigma_y}$$

- Correlation values range from +1 (max positive association) to -1 (max negative association)
- Correlation matrix is similar to covariance matrix
  - Symmetric matrix of correlations for p variables
  - Diagonal values in this case are equal to 1



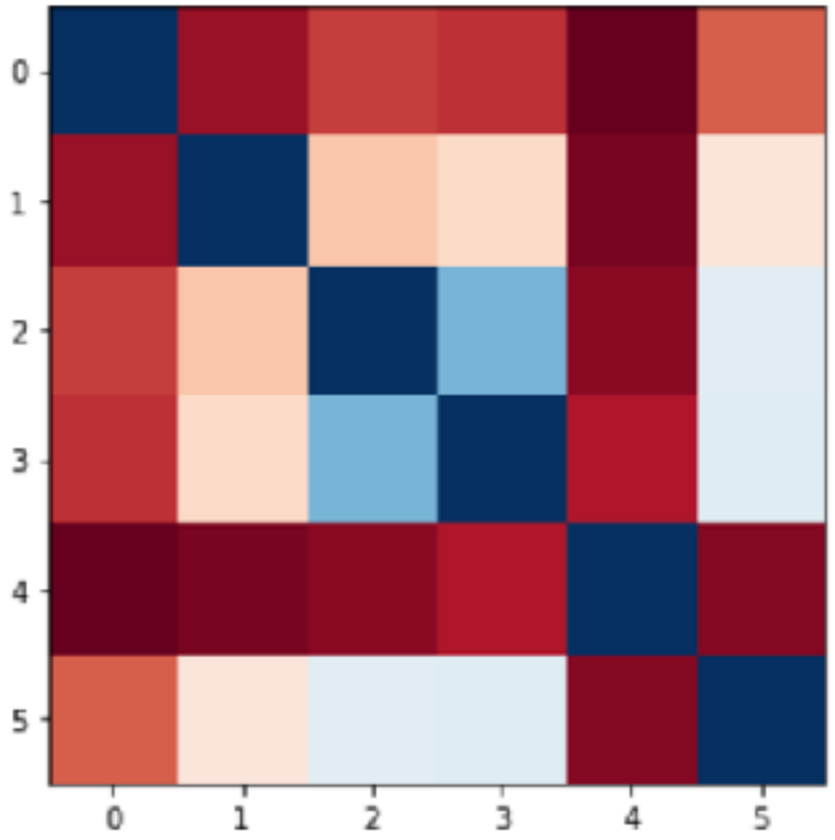
Correlation example

```
# correlation for mammography data
data2.corr()
```

	BIRADS	Age	Shape	Margin	Density	Severity
BIRADS	1.000000	0.094366	0.180067	0.157425	0.028266	0.223584
Age	0.094366	1.000000	0.380075	0.420268	0.052215	0.454717
Shape	0.180067	0.380075	1.000000	0.738601	0.073922	0.564916
Margin	0.157425	0.420268	0.738601	1.000000	0.124606	0.573805
Density	0.028266	0.052215	0.073922	0.124606	1.000000	0.068398
Severity	0.223584	0.454717	0.564916	0.573805	0.068398	1.000000

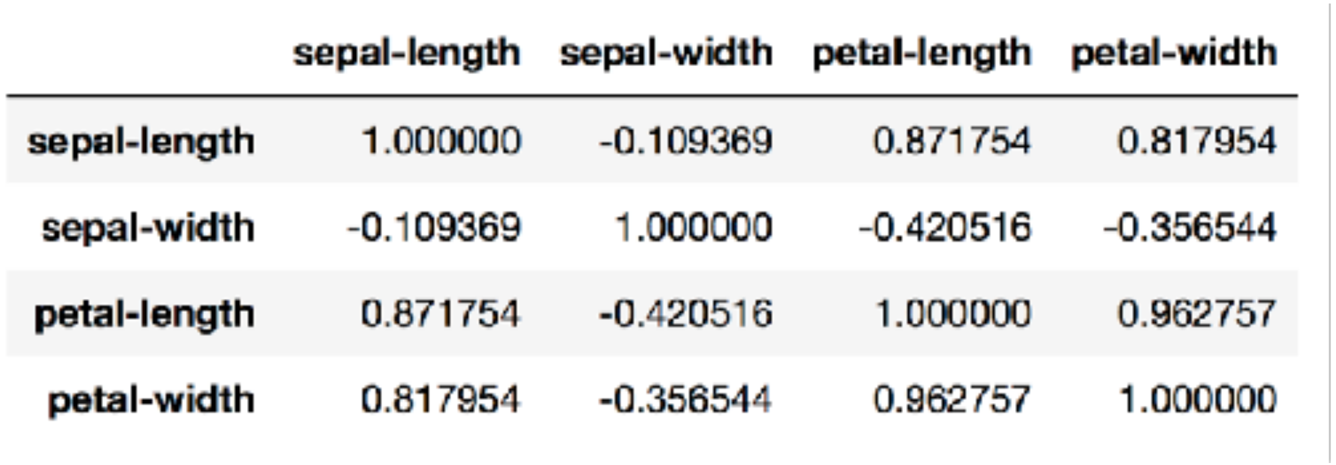
Correlation example

```
# visualizing correlation matrix using a divergent color map
plt.imshow(data2.corr(),cmap='RdBu')
```



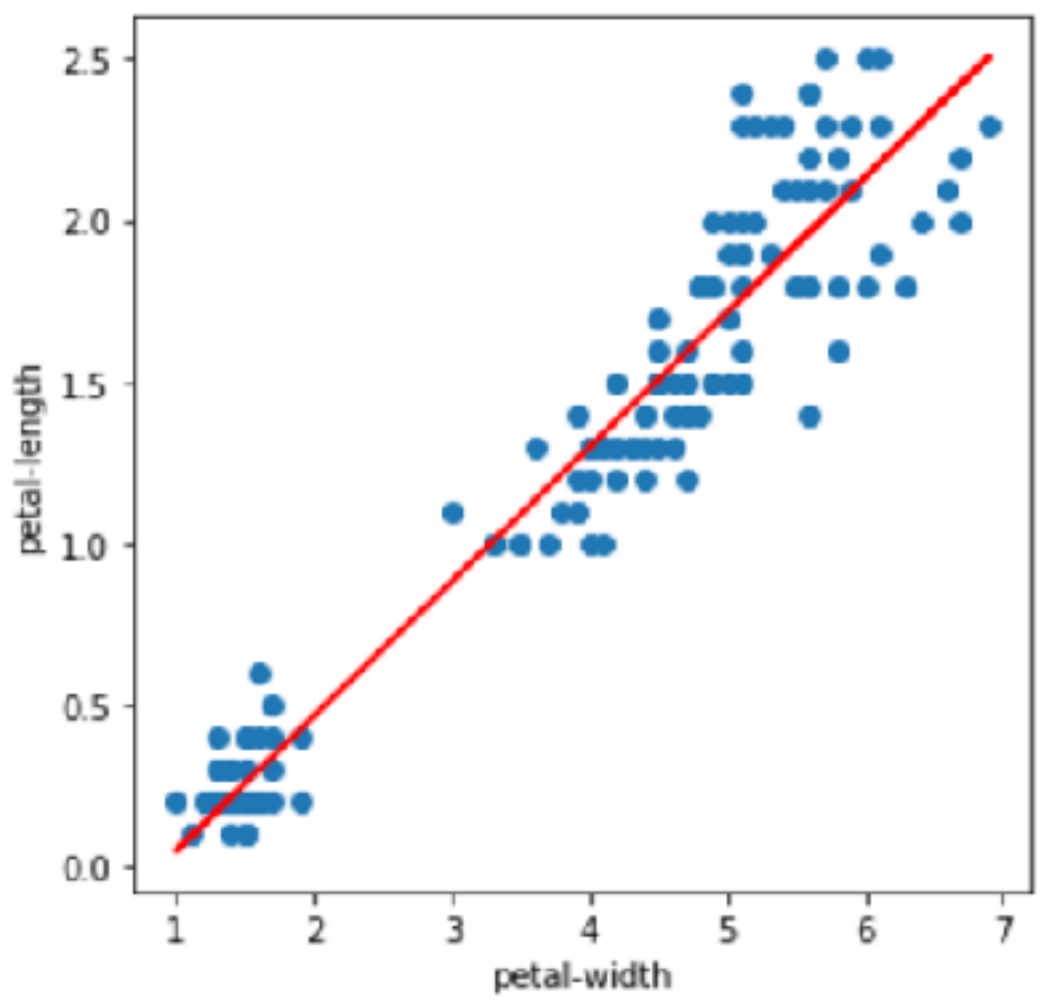
Correlation example

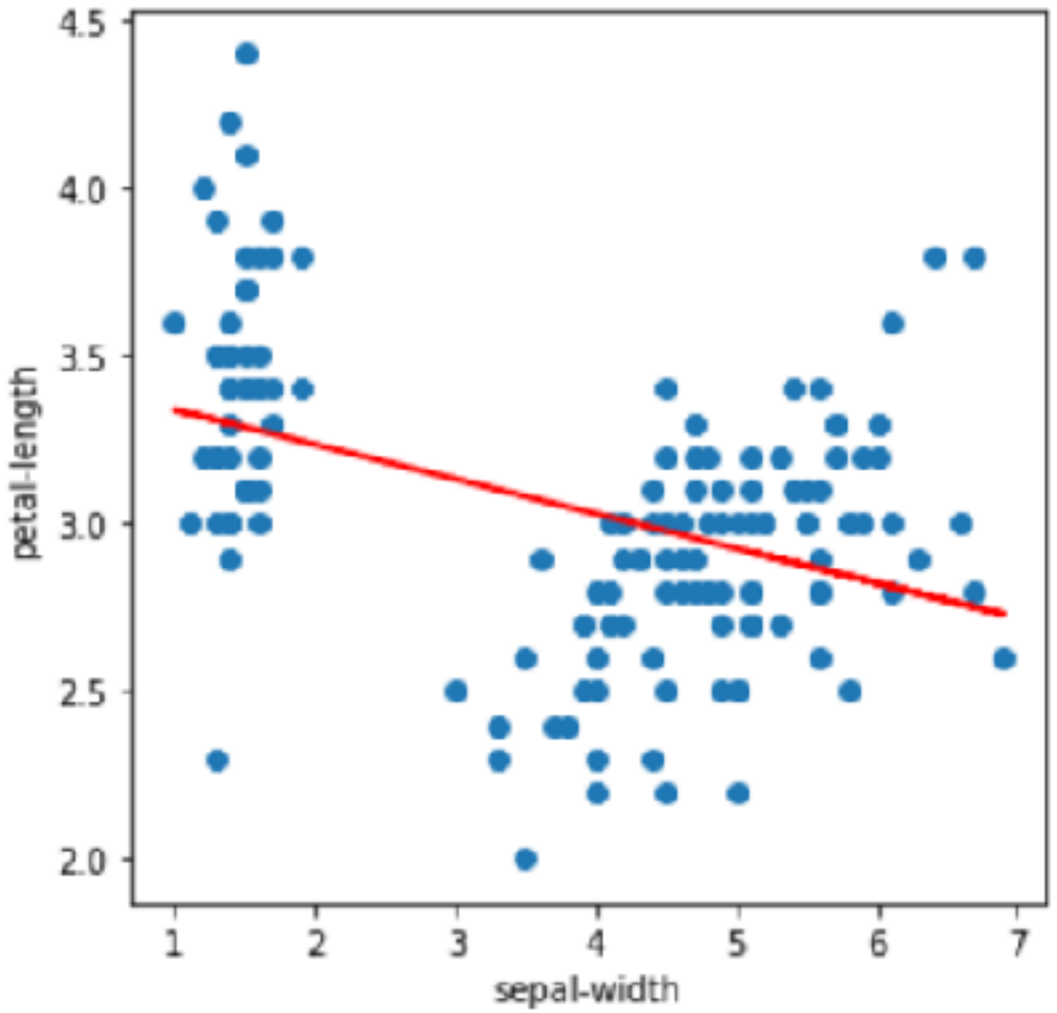
```
# correlation of iris data
data.corr()
```



	sepal-length	sepal-width	petal-length	petal-width
sepal-length	1.000000	-0.109369	0.871754	0.817954
sepal-width	-0.109369	1.000000	-0.420516	-0.356544
petal-length	0.871754	-0.420516	1.000000	0.962757
petal-width	0.817954	-0.356544	0.962757	1.000000

Scatterplots of correlated variables





Adding a fit line to scatter plots

```
from sklearn import linear_model

xvar = 'petal-length'
yvar = 'petal-width'
plt.scatter(data[xvar],data[yvar])
plt.ylabel(xvar)
plt.xlabel(yvar)

# Create linear regression object
regr = linear_model.LinearRegression()
# Fit the regression model using the data
regr.fit(data[[xvar]].values, data[yvar].values)
# Plot the predicted y values as a line
plt.plot(data[xvar], regr.predict(t), color='red', linewidth=2)
```