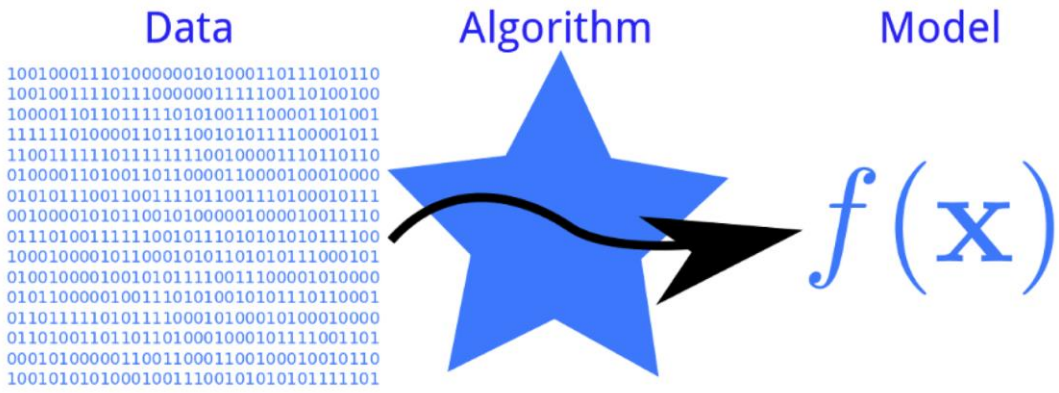
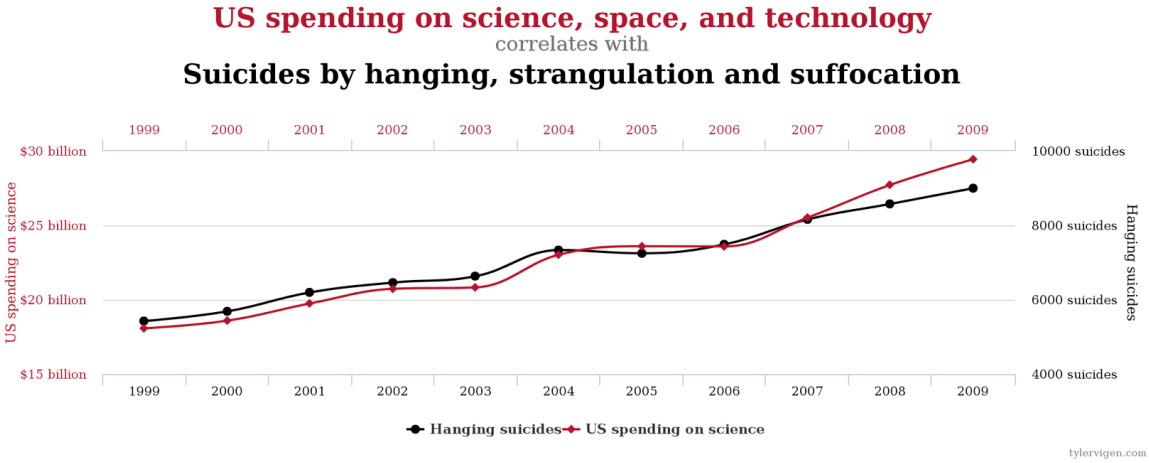
Scikit-learn Machine Learning in Python

1 What is Machine Learning

Firstly, let us learn the definition of Machine Learning: If you have some datasets and there are some algorithms by which you can extract the patterns form that leads you to a certain model, then you can make a useful prediction. This figure can show a machine learning workflow.



Machine Learning is a method of data analysis that automates analytical model building. Using algorithms that iteratively learn from data, machine learning allows computers to find the hidden insights, although we do not have the explicitly programmed before we work.



When we get a dataset, an important work of caution is: Is something learnable or not? Such as a topic like if we want to know the relationship from US spending on science and Ways of Suicides as the above figure shows, can we predict a result?
In this case, we find there is a correlation of two unrelated factors, "US spending on science, space, and technology" and "Suicides by hanging, strangulation and suffocation". However, both seem not related or without causation relation in our common sense. Thus, we should make sure if our data is learnable

or not and if the potential relationship makes sense before we use machine learning to deal with the data

2 General Learning Models

We need to recognize the data type when we get a dataset. A key distinction we need to pay attention to in machine learning is the concept of **supervised and unsupervised learning**. Supervised Learning provides explicit labels. A label means you give some data and then you say what's the output should be. While the Unsupervised Learning doesn't need labels and it aims to find the underlying structure of the data. Followings are the main differences between supervised and unsupervised learning.

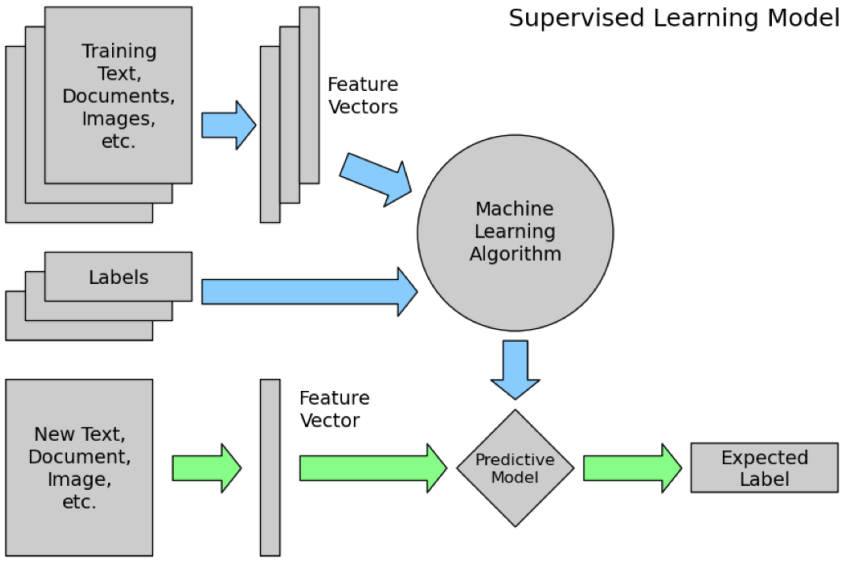
Supervised learning problems can be further grouped into regression and classification problems.

- **Classification:** A classification problem happens when the output variable is a category, such as "red" or "blue" or "disease" and "no disease".
- **Regression**: A regression problem happens when the output variable is a real value, such as "dollars" or "weight".

Unsupervised learning problems can be further grouped into clustering and association problems.

- **Clustering**: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- **Association:** An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

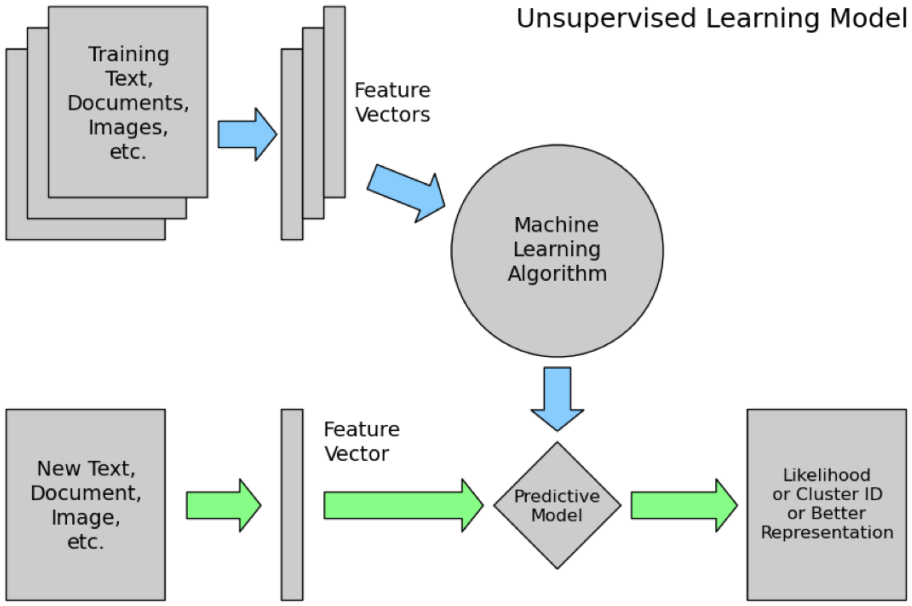
2.1 Supervised Learning Model



- For process supervised learning model, the main measures: the data should be labeled and with feature vectors. (Feature Vector + Labels = Predictive Model)
- We have some labeled data, for every example you have in a training dataset. Then you go through these examples and formalize them as feature vectors. Then use the machine learning algorithm to take your representation and the known correct labels, then develop a model that map the input data to the labels.

- It is called supervised learning because the process of an algorithm learning from the training dataset can lead the learning process with correct labels. The algorithm iteratively makes predictions on the training data and is corrected by the process. This learning process stops until the algorithm achieves an acceptable level of performance.

2.2 Unsupervised Learning Model



- For process unsupervised learning model, the data is unlabeled. And we try to find a relationship between feature vectors. (Feature Vectors + Intrinsic Structure = Predicted Clusters)
- In this case, we can there is no labels in the unsupervised learning models, then we are getting all training data same feature extraction process possibly.
- The goal for unsupervised learning is to model the underlying structure or distribution in the data, and we can explore more about the data.
- These are called unsupervised learning because unlike supervised learning above, there is no correct answers(labels). Algorithms are left to their own project to discover and present the interesting structure in the data.

3 Improve Model Performance

3.1 Feature Extraction

For increase the algorithm performance, Feature Extraction is a key step in machine leaning. The secret of success is to find the right way to represent your data so that your algorithm can find the potential pattern.

If the total prediction power is 100%:

- Effort on feature engineering contribute to 80%
- Effort on learning algorithm contribute to 20%

3.2 What are Features?

Features are information that is useful to make prediction. There are a lot of different Data Types: Numeric, Text, Audio, Image, Video etc. We will talk about those data types in the next part.

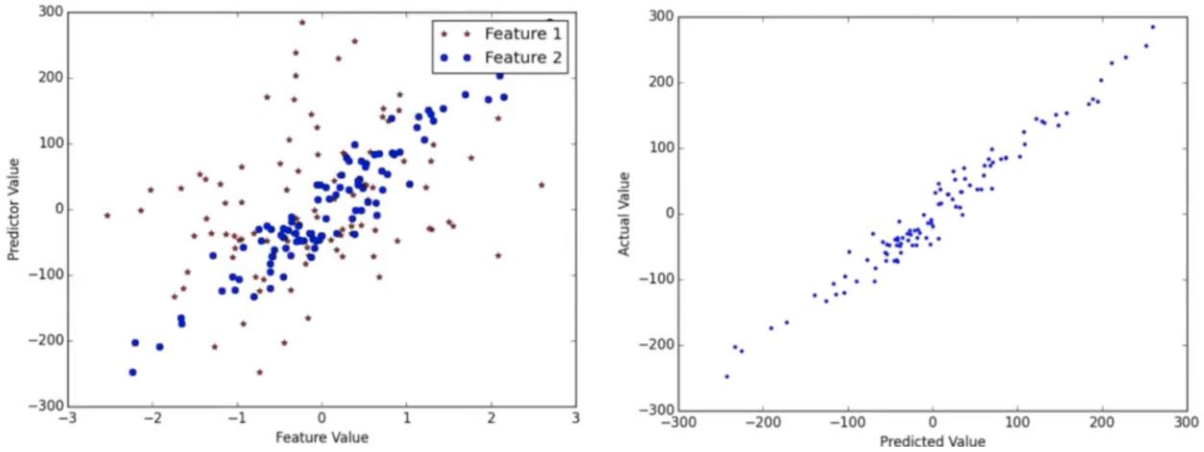
3.3 Learning Algorithms

For choosing algorithms to build our machine learning, we can talk with following two sides:

- Supervised learning (given x, y)
    - Regression: continuous variable
    - Classification: discrete variable
- Unsupervised learning (given x, no y)
    - Clustering
    - Dimension Reduction
    - Outlier / anomaly detection

Sometimes, we can find the intrinsic structure from supervised learning and unsupervised learning. Such as we use clustering to classify users of Amazon (unsupervised learning), and we can give label for each user group. Then we will use classification algorithms to classify new users (supervised learning)

3.5 Goal Today: Regression with Scikit-Learn



For example, we use Scikit-Learn to deal with the mussy data, and get a liner regression result.

4 Code Based Teaching: Regression with Scikit-Learn

Dataset

In [2]:
```
# import sklearn first
from sklearn import datasets
```

In [ ]:
```
# 3 ways for input data:
# load_, fetch_, make_
```

In [3]:
```
data = datasets.load_iris()
```

In [4]:
```
# check data (what the data for? what the data looks like?)
print(data.DESCR)
```

```
Iris Plants Database
====================

Notes
-----
Data Set Characteristics:
    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica
    :Summary Statistics:
```

```
    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20  0.76     0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

This is a copy of UCI ML iris datasets.
http://archive.ics.uci.edu/ml/datasets/Iris

The famous Iris database, first used by Sir R.A Fisher

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field
and
is referenced frequently to this day.  (See Duda & Hart, for example.)  T
he
data set contains 3 classes of 50 instances each, where each class refers
to a
type of iris plant.  One class is linearly separable from the other 2; th
e
latter are NOT linearly separable from each other.

References
----------
   - Fisher,R.A. "The use of multiple measurements in taxonomic problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions
to
```

```
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda,R.O., & Hart,P.E. (1973) Pattern Classification and Scene Analy
sis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
     Structure and Classification Rule for Recognition in Partially Expos
ed
     Environments".  IEEE Transactions on Pattern Analysis and Machine
     Intelligence, Vol. PAMI-2, No. 1, 67-71.
   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transa
ctions
     on Information Theory, May 1972, 431-433.
   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS
II
     conceptual clustering system finds 3 classes in the data.
   - Many, many more ...
```

In [6]:
```
data.data.shape
```

Out[6]:
```
(150, 4)
```

In [7]:
```
data.feature_names
```

Out[7]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [8]:
```
data.target
```

Out[8]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [9]:
```
data.target_names
```

Out[9]:
```
array(['setosa', 'versicolor', 'virginica'],
      dtype='<U10')
```

In [14]:
```
#fetch: loading when the data is really big
data = datasets.fetch_mldata('XXX')
```

In [ ]:
```
# mldata.org (data source)
data = datasets.fetch_mldata
```

In [16]:
```
# test algorithm
X, Y = datasets.make_regression(1000, n_features=2)
```

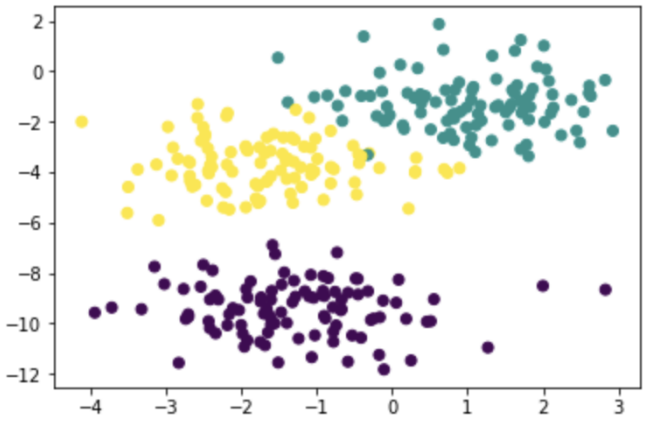
In [17]:
```
X.shape, Y.shape
```

Out[17]:
```
((1000, 2), (1000,))
```

In [ ]:
```
# Excercise: create visualize the above given data,
# see how the cluster is spread out.
```

In [31]:
```
X, Y = datasets.make_blobs(n_samples=300, n_features=2, centers=3,
                           random_state=2)
```

In [34]:
```
%matplotlib inline
plt.scatter(X[:,0], X[:,1], c=Y)
# When we test in work?
# 1. gain more understanding about one algorithm
# 2. fast test different algorithm process
# dimension
# non-linearity
# complexity
import matplotlib.pyplot as plt
# When I try running plt.scatter(X[:0], X[:,1], c=Y), I get the error name
```



- 3 ways for input data: load_, fetch_, make_
- Excercise: create visualize the above given data, see how the cluster is spread out.

Feature extraction

In [35]:
```
from sklearn import feature_extraction
```

In [ ]:
```
# DictVectorize, FeatureHasher (pro, con)
```

In [36]:
```
# Transforms lists of feature-value mappings to vectors.
feature_extraction.DictVectorizer()
```

In [39]:
```
data = [
    {'today': 1, 'is': 1, 'good': 1, 'day': 1},
    {'tomorrow': 1, 'is': 1, 'good': 1, 'day': 1},
    {'yesterday': 1, 'is': 1, 'good': 1, 'day': 1}
]
# json.read
```

In [37]:
```
fe_dv = feature_extraction.DictVectorizer()
```

In [40]:
```
fe_dv.fit(data)
```

Out[40]:
```
DictVectorizer(dtype=<class 'numpy.float64'>, separator='=', sort=True,
        sparse=True)
```

In [42]:
```
fe_dv.transform(data).toarray()
```

Out[42]:
```
array([[ 1.,  1.,  1.,  1.,  0.,  0.],
       [ 1.,  1.,  1.,  0.,  1.,  0.],
       [ 1.,  1.,  1.,  0.,  0.,  1.]])
```

In [43]:
```
fe_dv.vocabulary_
```

Out[43]:
```
{'day': 0, 'good': 1, 'is': 2, 'today': 3, 'tomorrow': 4, 'yesterday': 5}
```

In [ ]:
```
# CountVectorizer, TfidfVectorizer, HashingVectorizer
```

In [47]:
```
t1 = 'data scientist needs to learn statistics'
t2 = 'data scientist needs to learn computer science'
t3 = 'data scientist needs to learn business'
t4 = 'data scientist needs to know chemistry'
```

In [52]:
```
# Convert a collection of text documents to a matrix of token counts
fe_cv = feature_extraction.text.CountVectorizer(ngram_range=(1, 2))
```

In [53]:
```
fe_cv.fit([t1, t2, t3, t4])
```

Out[53]:
```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=1.0, max_features=None, min_df=1,
        ngram_range=(1, 2), preprocessor=None, stop_words=None,
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None, vocabulary=None)
```

In [54]:
```
fe_cv.transform([t1, t2, t3, t4]).toarray()
```

Out[54]:
```
array([[0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1],
       [0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1],
       [1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1],
       [0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0]])
```

In [55]:
```
fe_cv.vocabulary_
```

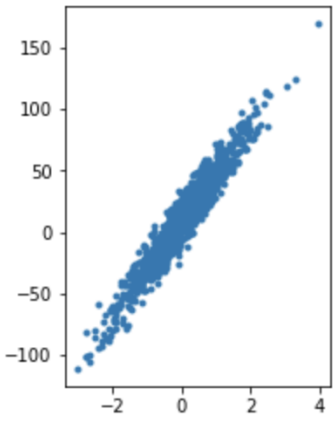
Out[55]:
```
{'business': 0,
 'chemistry': 1,
 'computer': 2,
 'computer science': 3,
 'data': 4,
 'data scientist': 5,
 'know': 6,
 'know chemistry': 7,
 'learn': 8,
 'learn business': 9,
 'learn computer': 10,
 'learn statistics': 11,
 'needs': 12,
 'needs to': 13,
 'science': 14,
 'scientist': 15,
 'scientist needs': 16,
 'statistics': 17,
 'to': 18,
 'to know': 19,
 'to learn': 20}
```

Feature selection

In [56]:
```
from sklearn import feature_selection
```

In [57]:
```
rgdata = datasets.make_regression(
    n_samples=1000, n_features=1, n_informative=1,
    n_targets=1, bias=10.0, effective_rank=None,
    tail_strength=0.5, noise=10.0, shuffle=True,
    coef=True, random_state=1)
fig = plt.figure()
ax1 = fig.add_subplot(1,2,1)
ax1.plot(rgdata[0][:,0], rgdata[1], '.')
fig.show()
```

```
/Users/peter/anaconda/lib/python3.6/site-packages/matplotlib/figure.py:403: UserWarning: matplotlib is currently using a non-GUI backend, so cannot show the figure
  "matplotlib is currently using a non-GUI backend, "
```



In [60]:
```
X = rgdata[0]
Y = rgdata[1]
```

In [61]:
```
X.shape, Y.shape
```

Out[61]:
```
((1000, 1), (1000,))
```

In [70]:
```
X1 = np.hstack([X, np.zeros([1000, 1])])
X1[1, 1] = 1
X1.shape
```

Out[70]:
```
(1000, 2)
```

In [75]:
```
fs_vt = feature_selection.VarianceThreshold(threshold=0.1)
```

In [76]:
```
fs_vt.fit(X1)
```

Out[76]:
```
VarianceThreshold(threshold=0.1)
```

In [77]:
```
fs_vt.variances_
```

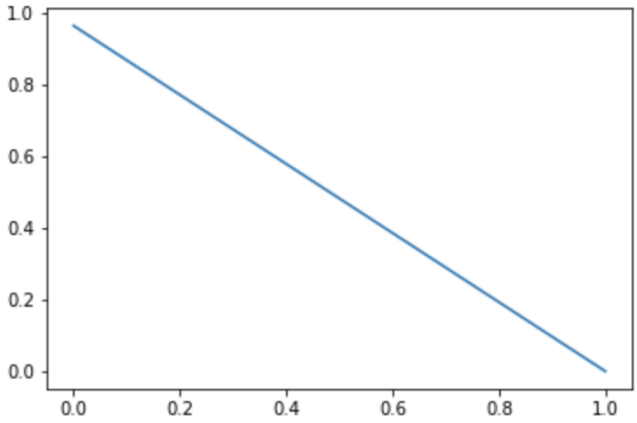
Out[77]:
```
array([ 0.96236911,  0.000999  ])
```

In [78]:
```
X2 = fs_vt.transform(X1)
X2.shape
```

Out[78]:
```
(1000, 1)
```

In [81]:
```
plt.plot(X1.var(axis=0))
```

Out[81]: [<matplotlib.lines.Line2D at 0x11bdc44e0>]



In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [ ]:
```
# GenericUnivariateSelect
```

In [83]:
```
# RFE
feature_selection.RFE?
```

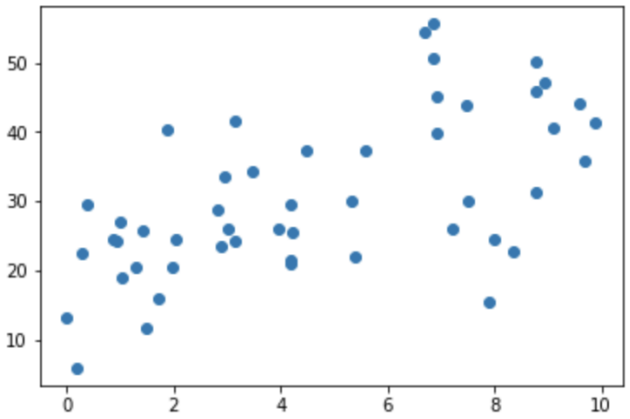
In [ ]:
```
np.random.seed(1)
X = np.random.random([1000, 1])
Y = X.ravel()
X1 = X + np.random.random([1000, 1]) * 0.01
X2 = np.random.random([1000, 3])
XF = np.hstack([X, X1, X2])
```

Learning algorithm

- Application levels
- Regression
- Classification
- Dimension reduction
- Clustering
- Outlier detection

In [84]:
```
np.random.seed(1)
X = np.random.random([50,1]).ravel() * 10
Y = X * 2 + 20 + 10 * np.random.randn(50)
plt.scatter(X,Y)
```

Out[84]:
```
<matplotlib.collections.PathCollection at 0x11ae68f60>
```



In [85]:
```
from sklearn import linear_model
```

In [86]:
```
lm_lr = linear_model.LinearRegression()
```

In [92]:
```
X.reshape(-1, 1).shape
```

Out[92]:
```
(50, 1)
```

In [91]:
```
lm_lr.fit(X.reshape(-1, 1), Y)
```

Out[91]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=Fal
se)
```

In [93]:
```
lm_lr.intercept_
```

Out[93]:
```
20.01422914446794
```

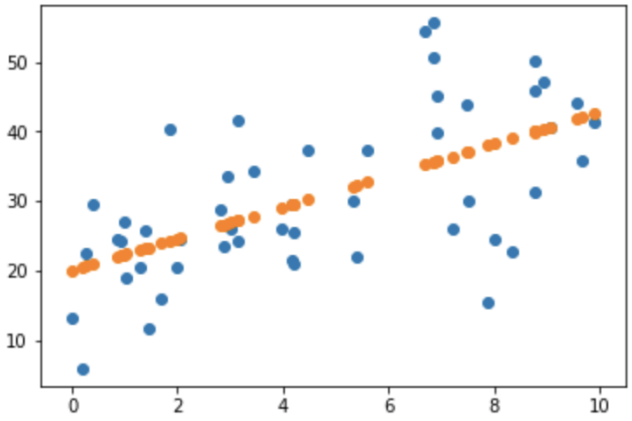
In [94]:
```
lm_lr.coef_
```

Out[94]:
```
array([ 2.2720881])
```

In [96]:
```
Ypred = lm_lr.predict(X.reshape(-1,1))
```

In [97]:
```
plt.scatter(X,Y)
plt.scatter(X,Ypred)
```

Out[97]:
```
<matplotlib.collections.PathCollection at 0x11ab15ef0>
```



Model evaluation

In [98]:
```
from sklearn import metrics
```

In [104]:
```
metrics.regression.r2_score(Y, Ypred)
```

Out[104]:
```
0.37024155257533542
```

In [ ]:
```
# regression
# rmse
```

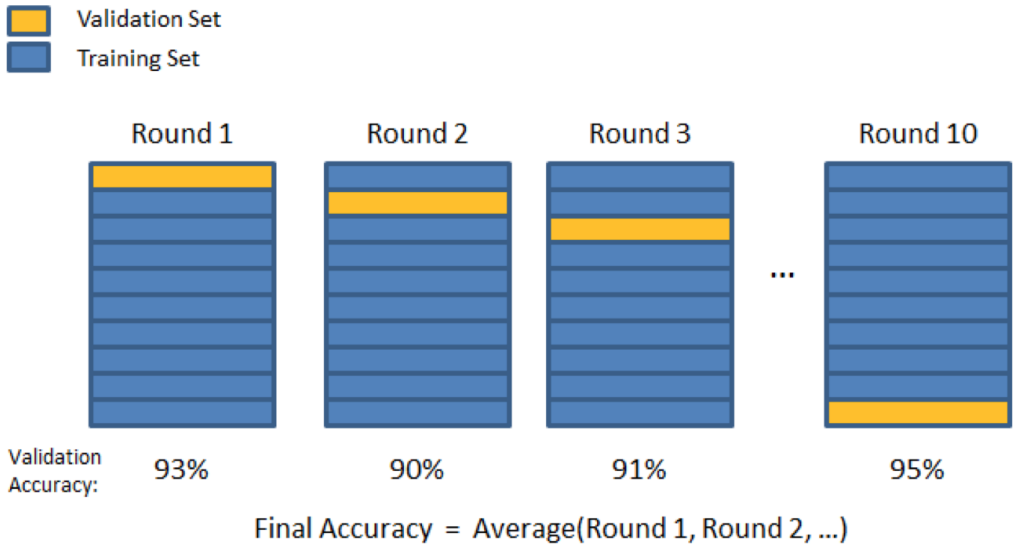
In [ ]:
```
# classification
# auc
```

In [ ]:
```
# r2_score vs. explained_varience_ratio
metrics.regression.r2_score
metrics.explained_variance_score
```

Model selection

Model selection is the task of selecting a statistical model from a set of candidate models, given data. In the simplest cases, a pre-existing set of data is considered. However, the task can also involve the design of experiments such that the data collected is well-suited to the problem of model selection. For avoid overfitting problem, we will split dataset to training & testing. (normally 70% for training, 30% test)

To better estimate the test error of a predictive model, we always use **cross validation**. The idea behind cross-validation is to create a number of partitions of sample observations, known as the validation sets, from the training data set. After fitting a model on to the training data, its performance is measured against each validation set and then averaged, gaining a better assessment of how the model will perform when asked to predict for new observations. The number of partitions to construct depends on the number of observations in the sample data set as well as the decision made regarding the bias-variance trade-off, with more partitions leading to a smaller bias but a higher variance.



In [106]:
```
from sklearn import model_selection
```

In [ ]:
```
lm_lr.fit(70%)
lm_lr.predict(15%)

lm_lr.predict(15%)

# cross validation
```

In [108]:
```
# model_selection.cross_val_predict?
model_selection.cross_val_score(lm_lr, X, y )
model_selection.cross_val_score(randomforest_reg, X, y )
model_selection.cross_val_score(reg1, X, y )
```

In [109]:
```
model_selection.train_test_split?
```

In [ ]:
```
# or manually specify the fold
# directly fold or split
# stratified fold or split
# label fold or split
```

Production pipeline

Pipeline of transforms with a final estimator.

In [106]:
```
from sklearn import model_selection
```

In [ ]:
```
lm_lr.fit(70%)
lm_lr.predict(15%)

lm_lr.predict(15%)

# cross validation
```

In [108]:
```
# model_selection.cross_val_predict?
model_selection.cross_val_score(lm_lr, X, y )
model_selection.cross_val_score(randomforest_reg, X, y )
model_selection.cross_val_score(reg1, X, y )
```

In [109]:
```
model_selection.train_test_split?
```

In [ ]:
```
# or manually specify the fold
# directly fold or split
# stratified fold or split
# label fold or split
```

Reference：

http://www.tylervigen.com/spurious-correlations
https://datascienceplus.com/cross-validation-estimating-prediction-error/