

# scikit-learn的使用

---

scikit-learn的使用

Scikit-Learn 4 通用学习模式 (机器学习 sklearn 教学教程tutorial)

Scikit-Learn 5 sklearn 的 datasets 数据库 (机器学习 sklearn 教学教程tutorial)

Scikit-Learn 6 model 常用属性和功能 (机器学习 sklearn 教学教程tutorial)

Scikit-Learn 7 normalization 标准化数据 (机器学习 sklearn 教学教程tutorial)

怎样检验神经网络 (深度学习)? How to evaluate neural networks (deep learning)?

Scikit-Learn 8 cross validation 交叉验证1 (机器学习 sklearn 教学教程tutorial)

Scikit-Learn 9 cross validation 交叉验证2 (机器学习 sklearn 教学教程tutorial)

Scikit-Learn 10 cross validation 交叉验证3 (机器学习 sklearn 教学教程tutorial)

Scikit-Learn 11 Save (机器学习 sklearn 教学教程tutorial)

感悟总结

## Scikit-Learn 4 通用学习模式 (机器学习 sklearn 教学教程tutorial)

---

模型的选择，参考这个图片

[http://scikit-learn.org/stable/tutorial/machine\\_learning\\_map/index.html](http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html)


有个错误

```
: import numpy as np
from sklearn import datasets
from sklearn.cross_validation import train_test_split
```

-----

```
ImportError                                Traceback (most recent call last)
<ipython-input-1-31026ec4543f> in <module>()
      1 import numpy as np
      2 from sklearn import datasets
----> 3 from sklearn.cross_validation import train_test_split

ImportError: No module named 'sklearn.cross_validation'
```



我把它注释掉，因为现在用的python3

```

from sklearn.cross_validation import train_test_split
from sklearn.neighbors import KNeighborsClassifier

iris = datasets.load_iris()
iris_X = iris.data
iris_y = iris.target

##print(iris_X[:2, :])
##print(iris_y)

X_train, X_test, y_train, y_test = train_test_split(
    iris_X, iris_y, test_size=0.3)

print(y_train)

```

## Scikit-Learn 5 sklearn 的 datasets 数据库 (机器学习 sklearn 教学教程tutorial)

波士顿房价的例子

### Scikit-Learn 5 sklearn 的 datasets 数据库 (机器学习 sklearn 教学教程tutorial)

```

[23]: from sklearn import datasets
      from sklearn.linear_model import LinearRegression
      import matplotlib.pyplot as plt

[15]: loaded_data = datasets.load_boston()
      data_X = loaded_data.data
      data_y = loaded_data.target

      model = LinearRegression()
      model.fit(data_X, data_y)

[16]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
      normalize=False)

[18]: print(model.predict(data_X[:4, :]))
      print(data_y[:4])

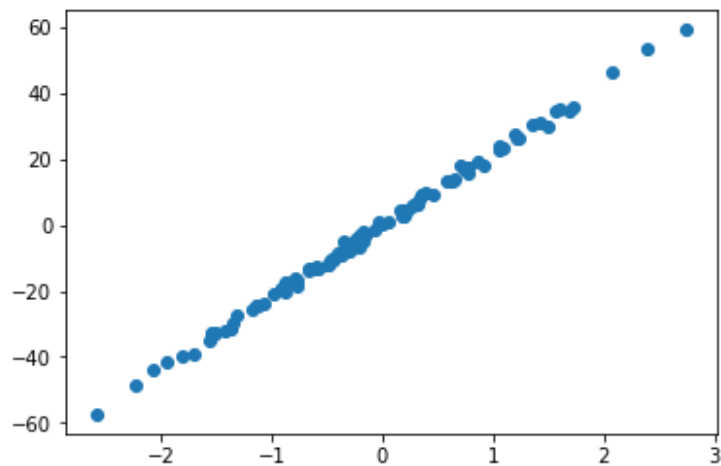
[30.00384338 25.02556238 30.56759672 28.60703649]
[24.  21.6 34.7 33.4]

[24]: # 增加模块plt
      X, y = datasets.make_regression(n_samples=100, n_features=1, n_targets=1, noise=1)
      plt.scatter(X, y)
      plt.show()

```

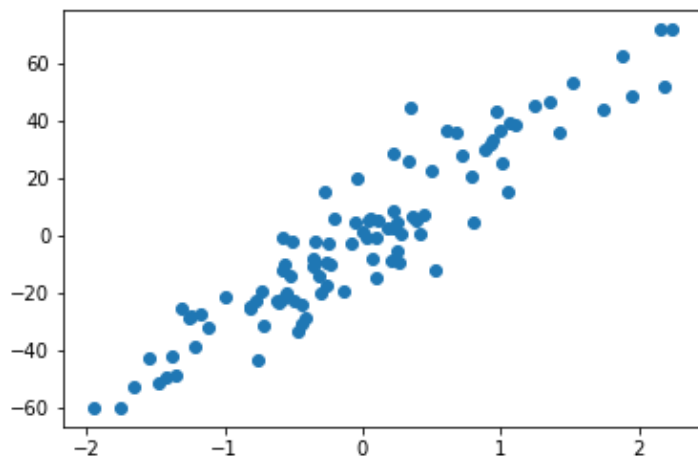


```
[24]: # 增加模块plt
X,y = datasets.make_regression(n_samples=100,n_features=1,n_targets=1,noise=1)
plt.scatter(X,y)
plt.show()
```



调整模型的参数：噪声改为10

```
]： #将噪声加到10
X,y = datasets.make_regression(n_samples=100,n_features=1,n_targets=1,noise=10)
plt.scatter(X,y)
plt.show()
```



---

## Scikit-Learn 6 model 常用属性和功能 (机器学习 sklearn 教学教程tutorial)

---

```
##print(model.predict(data_X[:4, :]))
##print(model.coef_)
##print(model.intercept_)
|
```

```
[ -1.07170557e-01  4.63952195e-02  2.08602395e-02  2.68856140e+00
 -1.77957587e+01  3.80475246e+00  7.51061703e-04 -1.47575880e+00
 3.05655038e-01 -1.23293463e-02 -9.53463555e-01  9.39251272e-03
 -5.25466633e-01]
36.4911032804
```

前面定义的参数

```
print(model.get_params())
print(model.score(data_X, data_y))
```

打分，

```
0.740607742865
>>>
```

常用的属性

```
##print(model.predict(data_X[:4, :]))
##print(model.coef_)
##print(model.intercept_)
print(model.score(data_X, data_y)) # R^2 coefficient of
```

## Scikit-Learn 7 normalization 标准化数据 (机器学习 sklearn 教学教程tutorial)

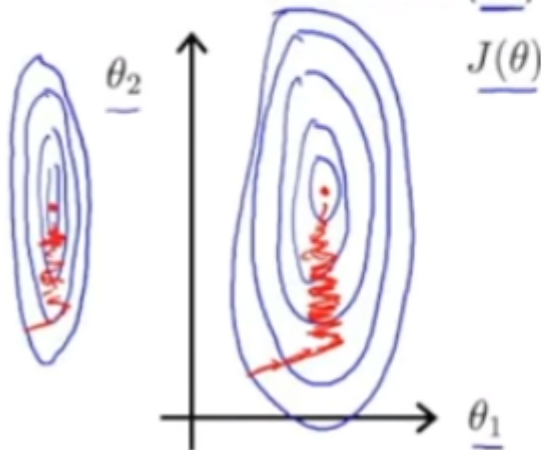
数据的归一化

## Feature Scaling

Idea: Make sure features are on a similar scale.

E.g.  $x_1 = \text{size (0-2000 feet}^2\text{)}$  ←

$x_2 = \text{number of bedrooms (1-5)}$  ←



$$\rightarrow x_1 = \frac{\text{size (feet}^2\text{)}}{2000}$$

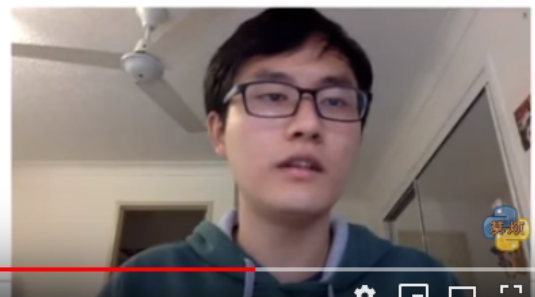
$$\rightarrow x_2 = \frac{\text{number of bedrooms}}{5}$$

$$0 \leq x_1 \leq 1 \quad 0 \leq x_2 \leq 1$$



归一化之后，在进入机器学习的步骤

```
print(a)
##print(preprocessing.scale(a))
X, y = make_classification(n_samples=300, n_features=2, n_redundant=0, n_informative=2,
                          random_state=22, n_clusters_per_class=1, scale=100)
##plt.scatter(X[:, 0], X[:, 1], c=y)
##plt.show()
X = preprocessing.scale(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3)
clf = SVC()
clf.fit(X_train, y_train)
```



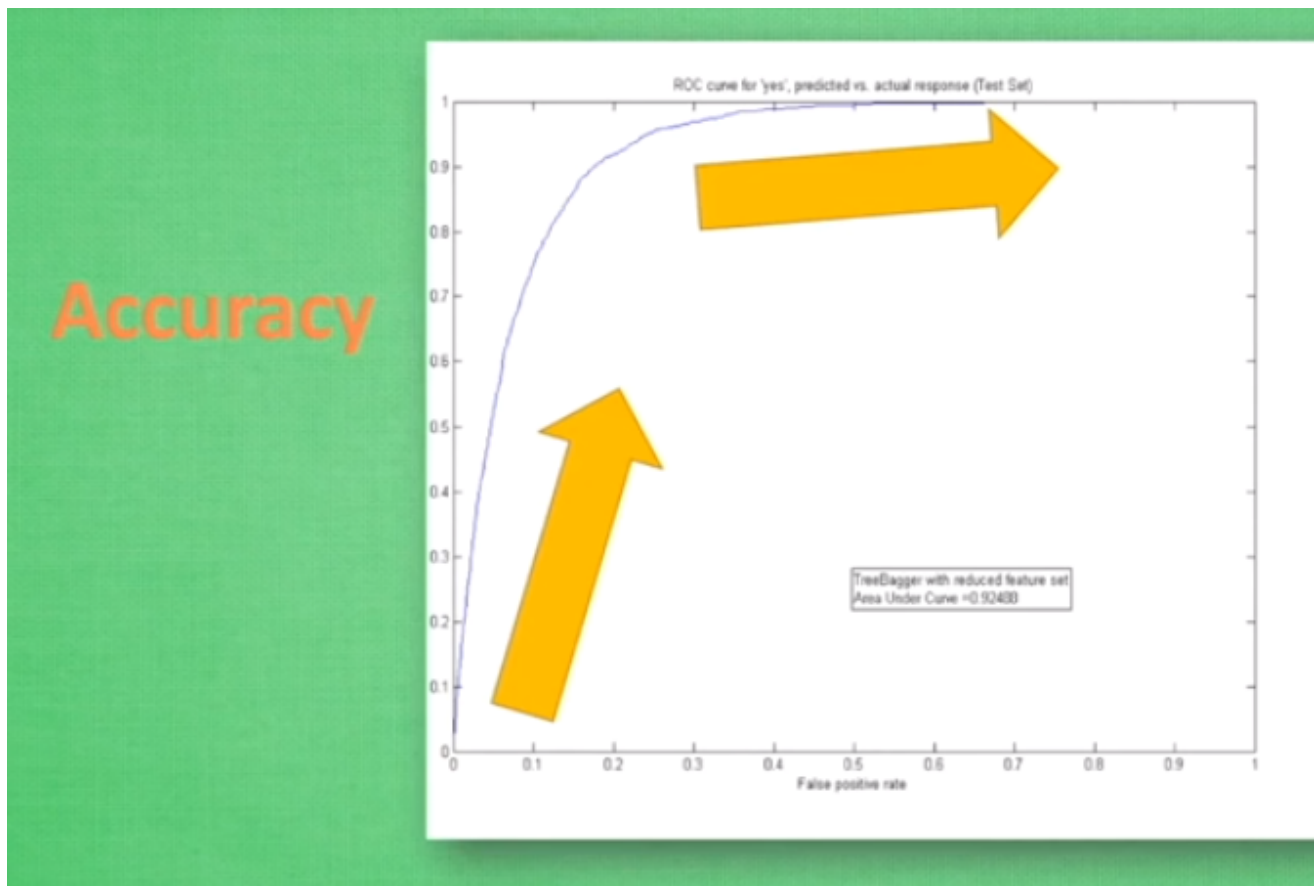
使用归一化，会使得机器学习模型的精度更高。

# 怎样检验神经网络 (深度学习)? How to evaluate neural networks (deep learning)?

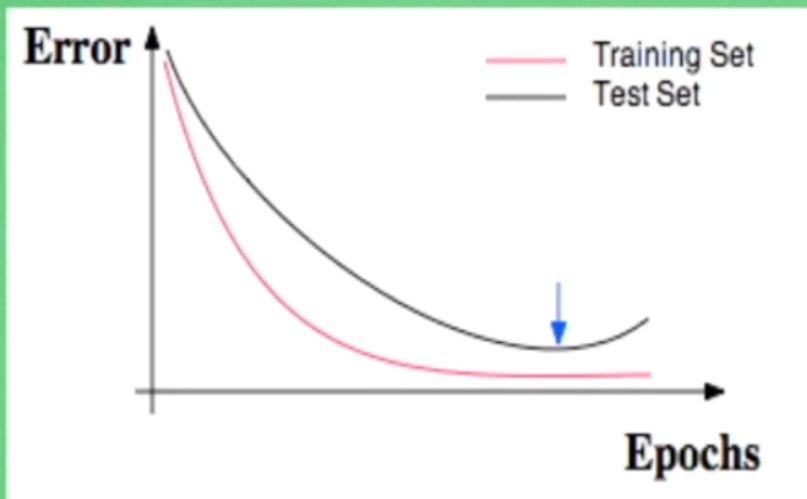
数据集分为训练集和测试集

像平时考试和期末考试

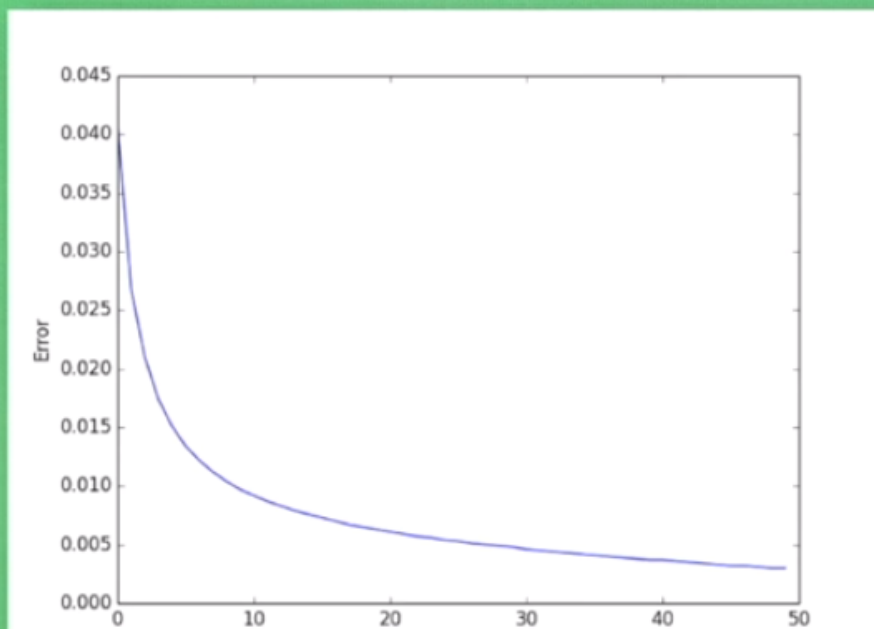
对于分类和回归







过拟合，没有把



Learning step

交叉验证，不进可以用于参数的调整。

## Scikit-Learn 8 cross validation 交叉验证1 (机器学习 sklearn 教学教程tutorial)

---



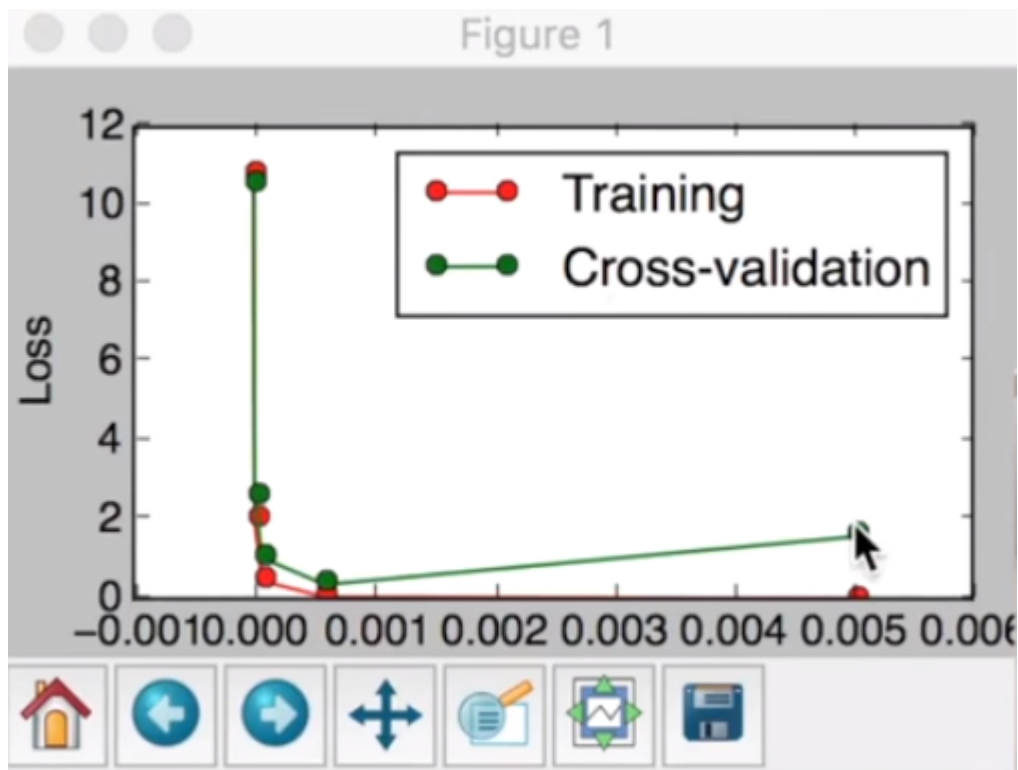
## Scikit-Learn 9 cross validation 交叉验证2 (机器学习 sklearn 教学教程tutorial)

---



# Scikit-Learn 10 cross validation 交叉验证3 (机器学习 sklearn 教学教程tutorial)

这次继续，



```

from sklearn.learning_curve import validation_curve
from sklearn.datasets import load_digits
from sklearn.svm import SVC
import matplotlib.pyplot as plt
import numpy as np

digits = load_digits()
X = digits.data
y = digits.target
train_sizes, train_loss, test_loss = learning_curve(
    SVC(gamma=0.01), X, y, cv=10, scoring='mean_squared_error',
    train_sizes=[0.1, 0.25, 0.5, 0.75, 1])
train_loss_mean = -np.mean(train_loss, axis=1)
test_loss_mean = -np.mean(test_loss, axis=1)

plt.plot(train_sizes, train_loss_mean, 'o-', color="r",
         label="Training")
plt.plot(train_sizes, test_loss_mean, 'o-', color="g",
         label="Cross-validation")

```

```

python1.py - /Users/MorvanZhou/Desktop/python1.py (3.5.1)
gamma = 0.01
param_range = np.logspace(-6, -2.3, 5)
train_loss, test_loss = validation_curve(
    SVC(), X, y, param_name='gamma', param_range=param_range,
    scoring='mean_squared_error')
train_loss_mean = -np.mean(train_loss, axis=1)
test_loss_mean = -np.mean(test_loss, axis=1)

plt.plot(param_range, train_loss_mean, 'o-', color="r",
         label="Training")
plt.plot(param_range, test_loss_mean, 'o-', color="g",
         label="Cross-validation")

plt.xlabel("gamma")
plt.ylabel("Loss")
plt.legend(loc="best")
plt.show()

```

SciKit-Learn  
Python 3 #11  
保存 model



这部分代码需要去他的官网上下载下来，然后在做一次实验。

## Scikit-Learn 11 Save (机器学习 sklearn 教学教程tutorial)

---

根据那张图表，选择合适的模型，训练模型，然后保存模型。

```
from sklearn import svm
from sklearn import datasets

##clf = svm.SVC()
##iris = datasets.load_iris()
##X, y = iris.data, iris.target
##clf.fit(X, y)

# method 1: pickle
import pickle
##with open('save/clf.pickle', 'wb') as f:
##    pickle.dump(clf, f)
with open('save/clf.pickle', 'rb') as f:
    pickle.dump(clf, f)
```

这里是列出保存模型的三种方法吗？

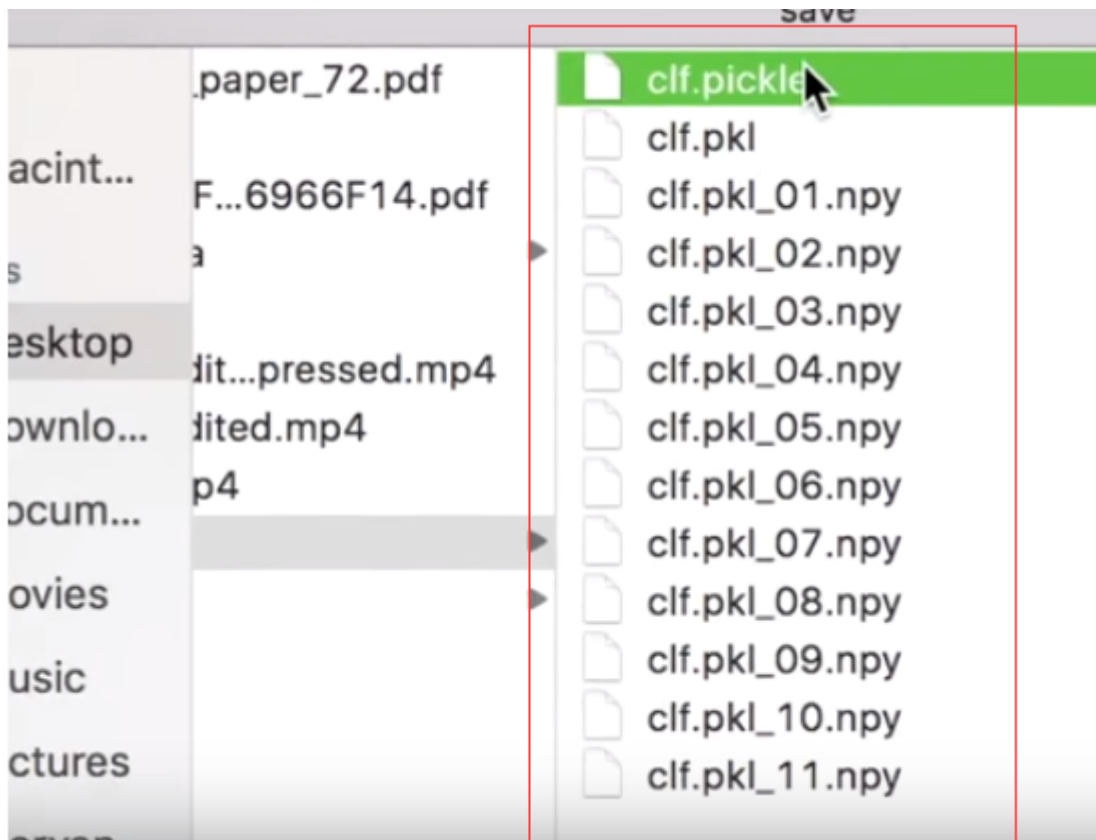
另一种方法，更简单；前面需要先运行。

```
# method 2: joblib
from sklearn.externals import joblib
# Save
joblib.dump(clf, 'save/clf.pkl')
# restore
clf3 = joblib.load('save/clf.pkl')
print(clf3.predict(X[0:1]))
```

打印出预测值

```
===== RESTART: /User
[0]
```

储存的位置



存储模型的好处：训练每个模型就要花一段相当长的时间，如果不保存，下次要用，又要重新训练。

# 感悟总结

---

看完这个视频，感觉理解有加深，但是对于scikit-learn的使用还是需要去阅读官网的帮众文档。我的选择会是跟着那张模型选择地图，在我复习机器学习算法的时候去看，同时实现上面的代码。把上面的代码调试通。

至少就是要熟悉一个通用的模式，然后，日积月累，去使用其它的模型，知道所有的模型都是用完。

在看这个教程的过程中，我先粗略的做了这份笔记，就是为了至少能回忆起大部分的知识。同时我收藏了几个链接到机器学习的收藏夹，后面就可以查看了。

我要去把那张地图给打印出来。

回顾老师讲得内容，针对我的情况，我比较看重通用模型的使用，还有交叉验证部分，最后新收获的知识的模型的保存这部分。

在看代码的时候，我也发现我对Python的语言还是很生疏，因为我使用太少了。