简单易用，同时支持多种推荐算法：

[基础算法/baseline algorithms](http://surprise.readthedocs.io/en/stable/basic_algorithms.html)

[基于近邻方法(协同过滤)/neighborhood methods](http://surprise.readthedocs.io/en/stable/knn_inspired.html)

[矩阵分解方法/matrix factorization-based (SVD, PMF, SVD++, NMF)](http://surprise.readthedocs.io/en/stable/matrix_factorization.html#surprise.prediction_algorithms.matrix_factorization.SVD)

| **算法类名** | **说明** |
| --- | --- |
| [**random\_pred.NormalPredictor**](http://surprise.readthedocs.io/en/stable/basic_algorithms.html#surprise.prediction_algorithms.random_pred.NormalPredictor) | **基于训练集的分布预测随机评分的算法，假定为正常。** |
| [**baseline\_only.BaselineOnly**](http://surprise.readthedocs.io/en/stable/basic_algorithms.html#surprise.prediction_algorithms.baseline_only.BaselineOnly) | **算法预测给定用户和项目的基线估计。** |
| [**knns.KNNBasic**](http://surprise.readthedocs.io/en/stable/knn_inspired.html#surprise.prediction_algorithms.knns.KNNBasic) | **基本的协作过滤算法。** |
| [**knns.KNNWithMeans**](http://surprise.readthedocs.io/en/stable/knn_inspired.html#surprise.prediction_algorithms.knns.KNNWithMeans) | **基本的协作过滤算法，考虑到每个用户的平均评分。** |
| [**knns.KNNBaseline**](http://surprise.readthedocs.io/en/stable/knn_inspired.html#surprise.prediction_algorithms.knns.KNNBaseline) | **考虑基线评级的基本协作过滤算法。** |
| [**matrix\_factorization.SVD**](http://surprise.readthedocs.io/en/stable/matrix_factorization.html#surprise.prediction_algorithms.matrix_factorization.SVD) | **着名的SVD算法，在Netflix奖期间由Simon Funk推广。** |
| [**matrix\_factorization.SVDpp**](http://surprise.readthedocs.io/en/stable/matrix_factorization.html#surprise.prediction_algorithms.matrix_factorization.SVDpp) | **SVD ++算法是SVD的扩展，考虑了隐式评级。** |
| [**matrix\_factorization.NMF**](http://surprise.readthedocs.io/en/stable/matrix_factorization.html#surprise.prediction_algorithms.matrix_factorization.NMF) | **基于非负矩阵分解的协同过滤算法。** |
| [**slope\_one.SlopeOne**](http://surprise.readthedocs.io/en/stable/slope_one.html#surprise.prediction_algorithms.slope_one.SlopeOne) | **一个简单但精确的协作过滤算法。** |
| [**co\_clustering.CoClustering**](http://surprise.readthedocs.io/en/stable/co_clustering.html#surprise.prediction_algorithms.co_clustering.CoClustering) | **一种基于协同聚类的协同过滤算法。** |

其中基于近邻的方法(协同过滤)可以设定不同的度量准则。

| **相似度度量标准** | **度量标准说明** |
| --- | --- |
| [**cosine**](http://surprise.readthedocs.io/en/stable/similarities.html#surprise.similarities.cosine) | **计算所有用户（或项目）对之间的余弦相似度。** |
| [**msd**](http://surprise.readthedocs.io/en/stable/similarities.html#surprise.similarities.msd) | **计算所有用户（或项目）对之间的均方差差异相似度。** |
| [**pearson**](http://surprise.readthedocs.io/en/stable/similarities.html#surprise.similarities.pearson) | **计算所有用户（或项目）对之间的Pearson相关系数。** |
| [**pearson\_baseline**](http://surprise.readthedocs.io/en/stable/similarities.html#surprise.similarities.pearson_baseline) | **计算所有用户（或项目）对之间的（收缩）皮尔逊相关系数，**  **使用基准进行居中而不是平均值。** |

使用方法：作为算法的参数

sim\_options = {'name': 'pearson\_baseline',

'shrinkage': 0 # no shrinkage

}

algo = KNNBasic(sim\_options=sim\_options)

支持不同的评估准则

| **评估准则** | **准则说明** |
| --- | --- |
| [**rmse**](http://surprise.readthedocs.io/en/stable/accuracy.html#surprise.accuracy.rmse) | **计算均方根误差（均方根误差）。** |
| [**mae**](http://surprise.readthedocs.io/en/stable/accuracy.html#surprise.accuracy.mae) | **计算MAE（平均绝对误差）。** |
| [**fcp**](http://surprise.readthedocs.io/en/stable/accuracy.html#surprise.accuracy.fcp) | **计算FCP（一致性对的分数）。** |

平均绝对误差( MAE) 是常用的评价协同过滤算法的指标。使用 MAE 作为评价标准，通过计算预测评分和实际评分值之间的偏差来度量预测的准确度。MAE 越小说明推荐系统的推荐质量越高

## 使用要点：

直接导入算法类名即是一个模型，模型参数先不管。

from surprise import SVD

algo = SVD()

## 使用方法： from surprise.model\_selection 的三种解法

1. 使用交叉检验

from surprise import SVD

from surprise.model\_selection import cross\_validate #**交叉检验**

algo = SVD()

data = 假装有个data

cross\_validate(algo, data, **measures**=['RMSE', 'MAE'], cv=5, verbose=True)

1. 使用网格搜索

from surprise.model\_selection import GridSearchCV

param\_grid = {'n\_epochs': [5, 10], 'lr\_all': [0.002, 0.005],

'reg\_all': [0.4, 0.6]}

gs = GridSearchCV(SVD, param\_grid, measures=['rmse', 'mae'], cv=3) # 构建网格

gs.fit(data) # 拟合

# best RMSE score

print(gs.best\_score['rmse']) # 评估

# combination of parameters that gave the best RMSE score

print(gs.best\_params['rmse'])

1. 逐步分解

from surprise.model\_selection import train\_test\_split

trainset, testset = train\_test\_split(data, test\_size=.25) # 注意这个与sklearn不同，只有两个返回值

algo.fit(trainset) # 拟合

predictions = algo.test(testset) # 测试

# Then compute RMSE

accuracy.rmse(predictions) # 评估