**ML Practice**

**Performance measure:**

Use coef in Logistic Regression, p-value in Linear Regression or feature importance in Random Forest to check the contribution of variables.

Think about the following performance measures that can give more insight into the accuracy of the model than traditional classification accuracy:

Confusion Matrix: A breakdown of predictions into a table showing correct predictions (the diagonal) and the types of incorrect predictions made (what classes incorrect predictions were assigned).

Precision: A measure of classifier exactness.

Recall: A measure of classifier completeness

F1 Score (or F-score): A weighted average of precision and recall.

Especially,

Kappa (or Cohen’s kappa): Classification accuracy normalized by the imbalance of the classes in the data.

ROC Curves: Like precision and recall, accuracy is divided into sensitivity and specificity and models can be chosen based on the balance thresholds of these values.

用于分类：

Accuracy：准确率

Average\_precision：This score corresponds to the area under the precision-recall curve.（Note: this implementation is restricted to the binary classification task or multi-label classification task.）

F1：The F1 score can be interpreted as a weighted average of the precision and recall

F1 = 2 \* (precision \* recall) / (precision + recall)

Neg\_log\_loss：也就是交叉验证熵 -log P(yt|yp) = -(yt log(yp) + (1 - yt) log(1 - yp))

This is the loss function used in (multinomial) logistic regression and extensions of it such as neural networks

Precision recall 召回率

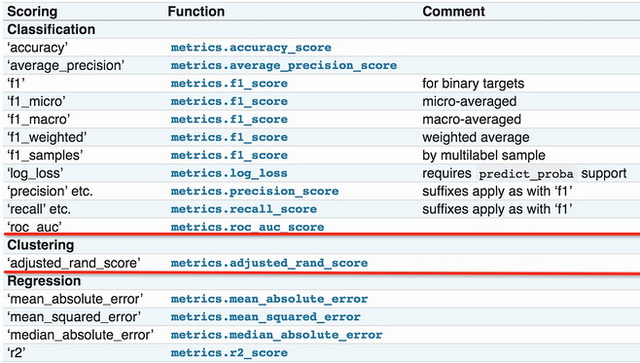
roc\_auc：Compute Area Under the Curve (AUC) from prediction scores

用于回归：

neg\_mean\_absolute\_error 绝对差

neg\_mean\_squared\_error 方差

R2\_score R^2 (coefficient of determination) regression score function.



机器学习性能评估指标：http://charleshm.github.io/2016/03/Model-Performance/

**Steps of Data Analysis:**

0）选择合适的机器学习算法All models are wrong, but some models are useful. — George Box (Box and Draper 1987)根据No free lunch theorem，在机器学习中，不存在一个在各方面都最好的模型/算法，因为每一个模型都或多或少地对数据分布有先验的统计假设。取所有可能的数据分布的平均，每个模型的表现都一样好（或者一样糟糕）。因此，我们需要针对具体的问题，找到最好的机器学习算法。

1）数据分析（Exploratory Data Analysis）在选择具体的算法之前，最好对数据中每一个特征的模式和产生原理有一定的了解：特征是连续的（real-valued）还是离散的（discrete）？如果特征是连续的，它的直方图（histogram）长什么样？它的mean和variance是如何分布的？如果特征是离散的，不同的特征值之间是否存在某种顺序关系？例如，豆瓣上从1星到5星的打分，虽然是离散数据，但有一个从低到高的顺序。如果某个特征是“地址”，则不太可能存在一个明确的顺序。特征数据是如何被采集的？

2）特征工程（Feature Engineering）特征工程（根据现有的特征，制造出新的、有价值的特征）决定了机器学习能力的上限，各种算法不过是在逼近这个上限而已。不同的机器学习算法一般会有其对应的不同的特征工程。在实践中，特征工程、调整算法参数这两个步骤常常往复进行。

3）由简至繁：具体算法的选择sklearn包括了众多机器学习算法。为了简化问题，在此只讨论几大类常见的分类器、回归器。至于算法的原理，sklearn的文档中往往有每个算法的参考文献，机器学习的课本也都有所涉及。

3.1）General Linear Models最开始建立模型时，我个人一般会选择high bias, low variance的线性模型。线性模型的优点包括计算量小、速度快、不太占内存、不容易过拟合。常用线性回归器的有Ridge（含有L2正则化的线性回归）和Lasso（含有L1正则化的线性回归，自带特征选择，可以获得sparse coefficients）。同时，如果对于超参数没有什么特别细致的要求，那么可以使用sklearn提供的RidgeCV和LassoCV，自动通过高效的交叉验证来确定超参数的值。假如针对同一个数据集X（m samples \* n features），需要预测的y值不止一个（m samples \* n targets），则可以使用multitask的模型。线性分类器中，最好用的是LogisticRegression和相应的LogisticRegressionCV。SGDClassifier和SGDRegressor可以用于极大的数据集。然而，如果数据集过大的话，最好从数据中取样，然后和小数据一样分析建模，未必一开始就要在整个数据集上跑算法。

3.2）Ensemble Methodsensemble能够极大提升各种算法，尤其是决策树的表现。在实际应用中，单独决策树几乎不会被使用。Bagging（如RandomForest）通过在数据的不同部分训练一群high variance算法来降低算法们整体的variance；boosting通过依次建立high bias算法来提升整体的variance。最常用的ensemble算法是RandomForest和GradientBoosting。不过，在sklearn之外还有更优秀的gradient boosting算法库：XGBoost和LightGBM。BaggingClassifier和VotingClassifier可以作为第二层的meta classifier/regressor，将第一层的算法（如xgboost）作为base estimator，进一步做成bagging或者stacking。我个人比较倾向于使用这一类模型。

3.3）支持向量机（SVM）SVM相关的知识可以参考Andrew Ng教授在Coursera上的CS229（有能力的可以去看youtube或者网易公开课上的原版CS229）。svm的API文档很完善，当一个调包侠也没有太大困难。不过在大多数的数据挖掘竞赛（如kaggle）中，SVM的表现往往不如xgboost。

3.4）神经网络（Neural Network）相比业内顶尖的神经网络库（如TensorFlow和Theano），sklearn的神经网络显得比较简单。个人而言，如果要使用神经网络进行分类/回归，我一般会使用keras或者pytorch。

机器学习算法中GBDT和XGBOOST的区别有哪些？

https://www.zhihu.com/question/41354392

SVM Para: <http://blog.csdn.net/bryan__/article/details/51506801>

Random Forest Para: <http://www.cnblogs.com/gczr/p/7141712.html>

|  |  |
| --- | --- |
| True0  Pred0  TP | True0  Pred1  FN |
| True1  Pred0  FP | True1  Pred1  TN |

<https://github.com/HanXiaoyang/Kaggle_Titanic/blob/master/Titanic.ipynb>

<http://blog.csdn.net/han_xiaoyang/article/details/49797143>