自动化机器学习框架

笔记本： 自动化建模
创建时间： 2019/7/26 16:17 **更新时间：** 2019/8/5 14:03
作者： lizhigu1996@163.com

机器学习建模是一个流程化的过程。首先我们需要拿到数据，其次就是数据的预处理、特征工程，接着要做模型的构建，并通过调参的方式来寻找最好的模型参数。如果效果不佳，我们经常需要回到特征工程，重新再走整个的流程。显然，在实际工程上，我们需要花费大量的精力在这些每个流程上的优化（包括特征选取，调参等等）。如果一个工具或者框架能够帮助我们把所有流程优化好，那会极大地提升工作效率。在这个情况下，我们只需要把输入数据传递给一个框架或者平台，则可以拿到最后已经训练好的模型。其实这就是AutoML所做的事情。
自动化机器学习（AutoML）是一种流水线（也称管线），它能够让你自动执行机器学习（ML）问题中的重复步骤，从而节省时间，让你专注于使你的专业知识发挥更高价值。 基本概念非常简单，一旦我们收到原始数据，我们就开始使用标准的ML流水线。
在这条流水线中，我们有一些针对于给定数据集/问题的步骤，在这个过程中，我们有以下任务：
特征预处理
特征选择
模型选择
...
这些任务的共同之处在于，在每个方案中我们先使用一组方法，然后我们评估其性能（特征重要性，模型性能...）， 由于我们对各个步骤有明确的指标，所以我们可以自动化流程。

一、 现有自动化机器学习框架

Python中的自动机器学习目前主要存在以下包：
auto-sklearn, TPOT, auto_ml 和H2O等

Auto_ml

Auto_ml（github / 文档）是用于生产和分析的自动化机器学习模块，这个框架包括分析(传入数据显示每个变量与目标变量之间的关系)、特征工程(日期和NLP特征衍生变量)、自动缩放(将所有值转换为在0和1之间的缩放)、特性选择(特征重要度选择)、数据格式化(转换为稀疏矩阵、独热编码分类变量、回归问题取自然对数等)、模型选择、超参数优化等诸多方面。
该框架使用进化网格搜索算法来完成特征处理和模型优化的繁重工作，利用了其它成熟函数库（如XGBoost、TensorFlow、Keras、LightGBM和sklearn）来提高计算速度，还宣称只需最多1毫秒来实现预测，但实际测试时不如其他框架快。

Auto-sklearn

Auto-sklearn（github / 文档）是scikit-learn基础上搭建的自动化学习平台， 其核心是使用SMAC算法（贝叶斯优化的随机森林实现），整个流程由贝叶斯搜索来优化机器学习流程中使用的数据预处理器、特征预处理器和分类器，并把多个步骤经过训练后整合成一个完整模型（使用15个分类器，14个特征预处理方法和4个数据预处理方法，产生具有110个超参数的结构化假设空间）。除此之外，Auto-sklearn还具备了元学习（meta-learning）和自动模型集成（build-ensemble），自动推荐合适的模型以及将多个模型组合成一个更强更大的模型。

H20

H2O（github / 文档）是基于Java编写的框架，它和sklearn等机器学习库的使用体验相似，并与Hadoop和Spark等大数据技术无缝协作。自动机器学习只是H2O的其中一个模块，这个模块利用其内置算法来创建机器学习模型。H2O提供许多流行算法的实现，例如GBM，随机森林，深度神经网络，Word2Vec和Stacked Ensembles，该框架对内置于H2O系统的预处理器实施穷举搜索，并使用网格搜索或随机网格搜索来优化超参数，最后生成的模型是一个结合了多个模型的集成模型。

H2O的优势在于它能够形成大型计算机集群，这使得它在规模上有所增长。它还可在python、javascript、tableau、R和Flow（web UI）等环境中使用。

TPOT

TPOT（github / 文档）是基于树的流程优化工具，它的核心基于遗传算法（genetic algorithm），是一种用于查找和生成最佳数据科学流程代码的遗传编程框架。TPOP会考虑管道中的多种机器学习算法（随机森林，线性模型，SVM等）和多个预处理步骤（缩放，PCA，特征选择等）。

Tpot的涉及到的建模过程主要有3块：Classifiers、Preprocesssors、Selectors，tpot的遗传算法优化是以pipeline为基础的，首先构建初始pipeline，根据父母的“特征/参数”创建子代的思想，在每次迭代结束时进行拟合测试创建新的种群，也就是说在每次迭代中我们将创建新的后代pipeline，如果后代表现更好，就可以用它们取代现有的pipeline，并且最适合的pipeline从原始的种群取出，这使得总体性能增加或者至少在每次迭代保持相同。经过遗传算法中的交叉、变异最终生成符合条件的模型效果更好的pipeline，再从最优的那一代中选取其中建模效果最好的pipeline即可。tpot是基于sklearn框架的，它本身不去实现我们常用的分类回归等算法，而是通过遗传算法优化pipeline，也就是从中选出最优的数据处理、特征选择、算法的组合。

auto_modeling

自己编写的自动化学习脚本，基于成熟的Python库，可以实现特征选择、超参数优化、模型选择等功能，在模型的支持上远不如上面几种成熟的自动化学习框架多，更多的是与实际生产环境的连接，比如支持pmml文件导出、报表输出。

总结

	编译语言	运行速度	流程框架	模型支持	元学习	自动模型集成
Auto_ml	Python，主要基于sklearn库	最慢，慢到没有一次成功调完参（默认为不进行超参数优化，选择超参数优化后根据参数组合数目选择网格搜索或进化算法，就算是选择了进化算法依旧很慢）	按规则进行数据预处理和特征预处理，然后用网格搜索或进化算法搜索模型超参数选择模型	支持向量机、决策树、逻辑回归、AdaBoost、梯度提升、随机森林、LightGBM、XGBoost、深度学习等		
Auto-sklearn	Python，主要基于sklearn库	较快，可按照设定时间运行	使用贝叶斯优化对数据预处理、特征预处理和模型的组合进行搜索	支持向量机、决策树、AdaBoost、梯度提升、随机森林、XGBoost等	支持	支持，将所有效果较好的单个模型使用ensemble selection方法进行集成
H2O	JAVA，自己用	快，可按照设定时间运行，	源码使用JAVA写的，不清楚内部具体流程框架	广义线性模型、随机森林、梯度提		支持，将训练的所有模型或每个算

	JAVA编写各模块	并且在hadoop环境上运行		升、XGBoost、深度学习等		法中的最佳模型进行简单的线性集合
TPOT	Python，主要基于sklearn库	慢，按照管道进行搜索，TPOT默认是评估10,000个管道配置并加上交叉验证	先按照population_size生成随机数据处理和模型算法的管道，再根据generations数目通过遗传算法生成新的管道， 从中选出最优的数据处理、特征选择、模型的组合。	支持向量机、决策树、逻辑回归、随机森林、梯度提升、XGBoost等		
auto_modeling	Python，主要基于sklearn库	一般	选择特征后按默认值填充缺失，贝叶斯优化或随机搜索优化参数选择模型	随机森林、梯度提升、LightGBM、XGBoost		

二、训练代码

Auto_ml

注意：该库会导入catboost，但支持的catboost版本较低，如果是最新版本，就算不使用catboost也会报错，并且该库仅支持lightgbm<2.1,>=2.0.11，版本较低，是2017年的版本。

1.导入库

```
from auto_ml import Predictor
from auto_ml.utils_models import load_ml_model
```

2.声明变量类型，有这几种类型可以声明："output", "categorical", "ignore", "nlp", or "date"

```
column_descriptions = {
    'TARGET': 'output',
    'NAME_CONTRACT_TYPE': 'categorical',
    'CODE_GENDER': 'categorical',
    ......
    'DATE': 'ignore'}
```

3.训练模型，其中type_of_estimator选择'classifier'和'regressor'，model_names为选择的模型，如果选择了多个模型将默认不进行网格搜索，需将optimize_final_model设置为True，并且需安装scikit-learn<=0.20.3，否则报错。这里的输出有问题，就算设置了verbose=False，选择网格搜索后也会打印大量的信息导致电脑notebook卡死

```
ml_predictor = Predictor(type_of_estimator='classifier',
column_descriptions=column_descriptions)
ml_predictor.train(train, optimize_final_model=True, model_names=
['GradientBoostingClassifier', 'RandomForestClassifier',
'DeepLearningClassifier', 'LGBMClassifier', 'XGBClassifier'], cv=5, verbose=False)
```

4.模型保存， 这个pickle / dill文件只能在安装了相同模块且运行相同Python版本的环境中加载，也就是说auto_ml虽然是直接与生产连接，但不支持跨平台部署，这样子在生产上较为局限。

```
file_name = ml_predictor.save(file_name='auto_ml_saved_pipeline.dill')
```

5.模型加载及预测，由于模型是对整个数据集训练，预测的数据集的变量应跟训练时相同

```
trained_model = load_ml_model(file_name)
```

predictions = trained_model.predict(df_test)

Auto-sklearn
auto-sklearn具有以下系统要求：
* Linux操作系统，auto-sklearn在很大程度上依赖于Python模块resource，resource 是Python的Unix特定服务的一部分， 在Windows机器上不可用。
* Python（> = 3.5）。
* C ++编译器（支持C ++ 11）
* SWIG（3.0或更高版本）。
实际使用时最好用Ubuntu16

1.导入库
import autosklearn.classification
2.训练模型，X数据集中不能含有非数值型变量，如果有需要先进行LabelEncoder转换，然后用feat_type声明类型。
automl = autosklearn.classification.AutoSklearnClassifier(seed=1234)
automl.fit(X_train, Y_train, feat_type=feat_type, metric=autosklearn.metrics.roc_auc)
3.模型预测
automl.predict_proba(X_test)
4.模型保存与导入
from sklearn.externals import joblib
joblib.dump(automl, 'automl_train.m')
automl_new = joblib.load('automl_train.m')

H20
linux环境才支持xgboost
1.导入库
import h2o
from h2o.automl import H2OAutoML
2.启动hadoop环境，连接到H2O服务器
h2o.init()
3.转换数据集，需将数据集转换成H2OFrame格式，如果是分类模型，需将目标变量进行转换asfactor()。
train = h2o.H2OFrame(train)
train[-1] = train[-1].asfactor()
4.训练模型
auto_ml_estimator = H2OAutoML(seed=1234)
auto_ml_estimator.train(x=train.names[:-1], y=train.names[-1], training_frame=train)
5.模型预测，auto_ml_estimator.leader为最好的模型，就算将auto_ml_estimator直接预测，也是用leader去预测。预测出的结果依旧为H2OFrame格式的数据集。
auto_ml_estimator.predict(test)
6.保存模型文件
file_name = auto_ml_estimator.download_mojo(r'model.mojo')
7.结束 h2o
h2o.shutdown()

TPOT
1.导入库
from tpot import TPOTClassifier
2.训练模型，训练集不能有类别变量，测试时对类别性变量手动进行独热编码，并用默认值填充缺失。
pipeline_optimizer = TPOTClassifier(cv=5, scoring='roc_auc', random_state=42, verbosity=2, n_jobs=-1)
pipeline_optimizer.fit(X_train, Y_train)
3.评估最终管道在测试集的效果
pipeline_optimizer.score(X_test, y_test)
4.导出模型文件，支持把TPOT相应的Python代码导出到文本文件中，这个管道并不是一个可以直接部署的文件，而是有模型参数的python代码，让你导入数据按照里面的代码去训练生成管道，

管道由sklearn.pipeline的make_pipeline封装，所以TPOT支持的预处理方法和机器学习算法也仅是sklearn.pipeline支持的而已。作者认为生成这样可以准备运行的、独立的Python代码可以让我们修改与审查这份代码，这份代码并不会是最终的模型，而是可以当做是我们寻找最优模型的有效起点。
pipeline_optimizer.export('tpot_exported_pipeline.py')

auto_modeling
1.导入库
import auto_modeling as am
2.训练模型并导出模型文件
am.full_automatic_complete(data, test_size=0.3, time_test_type=True, model_list=['xgb', 'lgbm', 'gbdt'], report_file='./result/result_report.html', pkl_file='./result/model.pkl', pmml_file='./result/model.pmml')

	类别型变量	数据预处理	特征预处理	超参数优化	模型文件
Auto_ml	支持	数值型变量缩放，类别型变量独热编码	日期和NLP变量特征工程，特征选择	网格搜索或进化算法（调用deep库）	pickle/dill文件，支持安装了相同模块且运行相同Python版本的环境中加载
Auto-sklearn	支持，但值不能为字符，需用LabelEncoder转换，fit时再用feat_type声明	数值型变量缩放，类别型变量独热编码，缺失值填充（与模型参数一同搜索）	特征选择、PCA、ICA、SVD、多项式特征生成等14个特征预处理方法（与模型参数一同搜索）	贝叶斯优化（随机森林方法实现，调用SMAC库）	m文件，也就是joblib可以导出的文件
H2O	支持			网格搜索或随机搜索	pojo/mojo文件，支持在JAVA中加载
TPOT	不支持，需自己进行独热编码转换（源码中找到OneHotEncoder的代码，但实测不支持非数值型变量入参，且不能声明为字符型）	中位数填充缺失（填充缺失的过程不在管道中，按照生成的py文件训练的管道不包括缺失值填充过程）	标准化、PCA、ICA、特征聚合、多项式特征生成、特征选择等（与模型参数一同搜索）	遗传算法（属于进化算法，同样调用deep库）	py文件，只是有数据处理参数和模型参数的python代码，需按照里面的代码去训练生成管道，不支持直接部署
auto_modeling	支持	类别型	特征选择	贝叶斯优	pmml文件，支

	变量独热编码，默认值填充缺失（部署后需先填充缺失再过模型文件）		化或随机搜索	持在JAVA中加载	

三、预测结果

H2O预测结果为H2OFrame格式的数据集，需使用 as_data_frame转换为pd.DataFrame数据集后再计算auc和ks。

Auto_ml默认为不调参，当传入optimize_final_model=True时，调参时间过长，未成功过，所以Auto_ml为不调参的情况。

auto_modeling相比与Auto-sklearn、H2O、TPOT偏差，主要有两个原因，一个是因为考虑了上线，变量选择比较严格，入模型的变量会尽量控制的较少，而其他自动建模方法则是全部变量入模型，变量选择上也比较粗暴（比如直接用方差过滤变量），并且 H2O和Auto-sklearn用了模型集成。

总体来说虽然auto_modeling并不优于Auto-sklearn、H2O、TPOT，但是选择变量较少，易于上线，其次加入了惩罚，可以在减小训练集与测试集差距的同时，尽量不让测试集降低太多。

	训练集AUC	测试集AUC	跨时间AUC	训练集KS	测试集KS	跨时间测试KS	最优模型
Auto_ml	0.8193	0.7288	0.7215	0.4814	0.3544	0.3395	单个XGBoost
Auto-sklearn	0.8781	0.7459	0.7419	0.6038	0.3786	0.3601	10个模型（包括adaboost、gradient_boosting、extra_trees、liblinear_svc、passive_aggressive）的线性组合
H2O	0.9424	0.7332	0.7399	0.7329	0.3601	0.363	每个算法最优模型的组合（包括GLM（广义线性模型）、XGBoost、DRF（分布式随机森林）、GBM（梯度提升）DeepLearning、XRT（极随机森林））的线性组合（其中梯度提升的系数为0）
TPOT	0.8	0.7374	0.7394	0.4404	0.3613	0.3582	单个XGBoost
auto_modeling	0.8707	0.7321	0.7318	0.573	0.3646	0.3504	单个LightGBM
auto_modeling（调参时加入惩罚）	0.7653	0.7319	0.7306	0.4102	0.3759	0.349	单个XGBoost