

# 目录

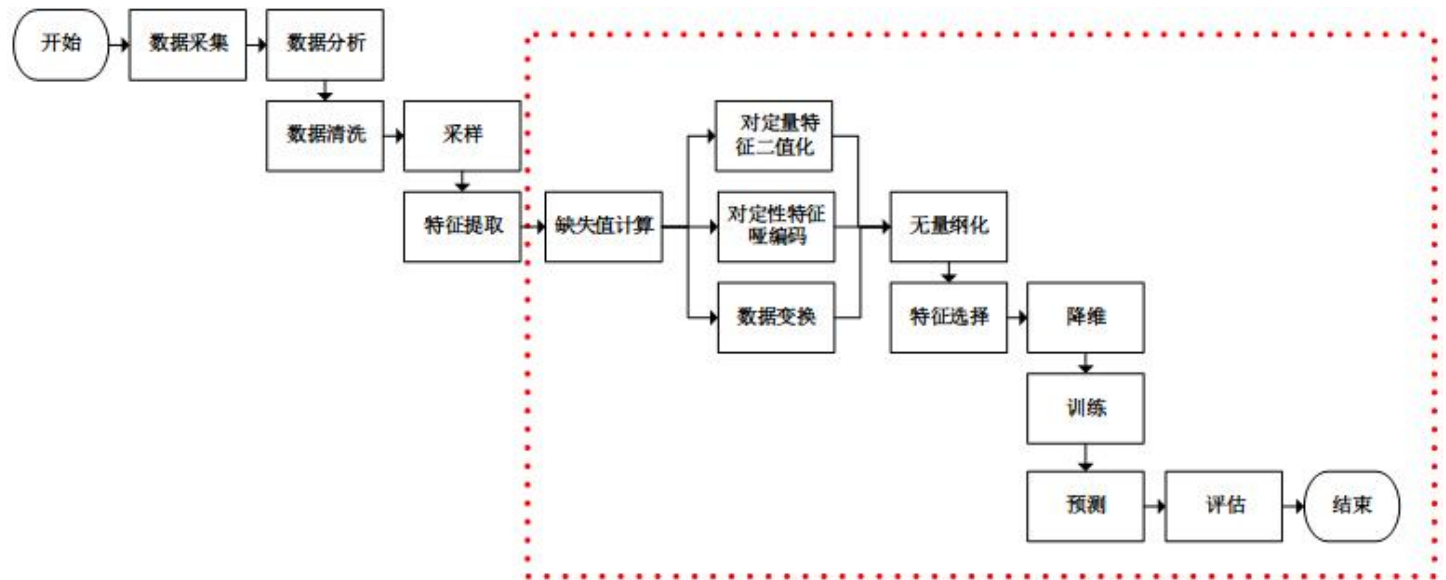
- 1 使用sklearn进行数据挖掘
  - 1.1 数据挖掘的步骤
  - 1.2 数据初貌
  - 1.3 关键技术
- 2 并行处理
  - 2.1 整体并行处理
  - 2.2 部分并行处理
- 3 流水线处理
- 4 自动化调参
- 5 持久化
- 6 回顾
- 7 总结
- 8 参考资料

## 1 使用sklearn进行数据挖掘

### 1.1 数据挖掘的步骤

数据挖掘通常包括数据采集，数据分析，特征工程，训练模型，模型评估等步骤。使用sklearn工具可以方便地进行特征工程和模型训练工作，在《使用sklearn做单机特征工程》中，我们最后留下了一些疑问：特征处理类都有三个方法fit、transform和fit\_transform，fit方法居然和模型训练方法fit同名（不光同名，参数列表都一样），这难道都是巧合？

显然，这不是巧合，这正是sklearn的设计风格。我们能够更加优雅地使用sklearn进行特征工程和模型训练工作。此时，不妨从一个基本的数据挖掘场景入手：




我们使用sklearn进行虚线框内的工作（sklearn也可以进行文本特征提取）。通过分析sklearn源码，我们可以看到除训练，预测和评估以外，处理其他工作的类都实现了3个方法：fit、transform和fit\_transform。从命名中可以看到，fit\_transform方法是先调用fit然后调用transform，我们只需要关注fit方法和transform方法即可。

transform方法主要用来对特征进行转换。从可利用信息的角度来说，转换分为无信息转换和有信息转换。无信息转换是指不利用任何其他信息进行转换，比如指数、对数函数转换等。有信息转换从是否利用目标值向量又可分为无监督转换和有监督转换。无监督转换指只利用特征的统计信息的转换，统计信息包括均值、标准差、边界等等，比如标准化、PCA法降维等。有监督转换指既利用了特征信息又利用了目标值信息的转换，比如通过模型选择特征、LDA法降维等。通过总结常用的转换类，我们得到下表：

包	类	参数列表	类别	fit方法有用	说明
sklearn.preprocessing	StandardScaler	特征	无监督	Y	标准化
sklearn.preprocessing	MinMaxScaler	特征	无监督	Y	区间缩放
sklearn.preprocessing	Normalizer	特征	无信息	N	归一化
sklearn.preprocessing	Binarizer	特征	无信息	N	定量特征二值化
sklearn.preprocessing	OneHotEncoder	特征	无监督	Y	定性特征编码
sklearn.preprocessing	Imputer	特征	无监督	Y	缺失值计算
sklearn.preprocessing	PolynomialFeatures	特征	无信息	N	多项式变换（fit方法仅仅生成了多项式的表达式）
sklearn.preprocessing	FunctionTransformer	特征	无信息	N	自定义函数变换（自定义函数在transform方法中调用）
sklearn.feature_selection	VarianceThreshold	特征	无监督	Y	方差选择法
sklearn.feature_selection	SelectKBest	特征/特征+目标值	无监督/有监督	Y	自定义特征评分选择法
sklearn.feature_selection	SelectKBest+chi2	特征+目标值	有监督	Y	卡方检验选择法
sklearn.feature_selection	RFE	特征+目标值	有监督	Y	递归特征消除法
sklearn.feature_selection	SelectFromModel	特征+目标值	有监督	Y	自定义模型训练选择法
sklearn.decomposition	PCA	特征	无监督	Y	PCA降维
sklearn.lda	LDA	特征+目标值	有监督	Y	LDA降维

不难看出，只有有信息的转换类的fit方法才实际有用，显然fit方法的主要工作是获取特征信息和目标值信息，在这点上，fit方法和模型训练时的fit方法就能够联系在一起了：都是通过分析特征和目标值，提取有价值的信息，对于转换类来说是某些统计量，对于模型来说可能是特征的权值系数等。另外，只有有监督的转换类的fit和transform方法才需要特征和目标值两个参数。fit方法无用不代表其没实现，而是除合法性校验以外，其并没有对特征和目标值进行任何处理，Normalizer的fit方法实现如下：



```
1 def fit(self, X, y=None):
2     """Do nothing and return the estimator unchanged
3     This method is just there to implement the usual API and hence
```

```
4         work in pipelines.  
5         """"  
6         X = check_array(X, accept_sparse='csr')  
7         return self
```



基于这些特征处理工作都有共同的方法，那么试想可不可以将他们组合在一起？在本文假设的场景中，我们可以看到这些工作的组合形式有两种：流水线式和并行式。基于流水线组合的工作需要依次进行，前一个工作的输出是后一个工作的输入；基于并行式的工作可以同时进行，其使用同样的输入，所有工作完成后将各自的输出合并之后输出。sklearn提供了包pipeline来完成流水线式和并行式的工作。

## 1.2 数据初貌

在此，我们仍然使用IRIS数据集来进行说明。为了适应提出的场景，对原数据集需要稍微加工：



```
1 from numpy import hstack, vstack, array, median, nan  
2 from numpy.random import choice  
3 from sklearn.datasets import load_iris  
4  
5 #特征矩阵加工  
6 #使用vstack增加一行含缺失值的样本(nan, nan, nan, nan)  
7 #使用hstack增加一列表示花的颜色（0-白、1-黄、2-红），花的颜色是随机的，意味着颜色并不影响花的分类  
8 iris.data = hstack((choice([0, 1, 2], size=iris.data.shape[0]+1).reshape(-1,1), vstack((iris.data,  
array([nan, nan, nan, nan]).reshape(1,-1)))))  
9 #目标值向量加工  
10 #增加一个目标值，对应含缺失值的样本，值为众数  
11 iris.target = hstack((iris.target, array([median(iris.target)])))
```



## 1.3 关键技术

并行处理，流水线处理，自动化调参，持久化是使用sklearn优雅地进行数据挖掘的核心。并行处理和流水线处理将多个特征处理工作，甚至包括模型训练工作组合成一个工作（从代码的角度来说，即将多个对象组合成了一个对象）。在组合的前提下，自动化调参技术帮我们省去了人工调参的反锁。训练好的模型是贮存在内存中的数据，持久化能够将这些数据保存在文件系统中，之后使用时无需再进行训练，直接从文件系统中加载即可。

# 2 并行处理

并行处理使得多个特征处理工作能够并行地进行。根据对特征矩阵的读取方式不同，可分为整体并行处理和部分并行处理。整体并行处理，即并行处理的每个工作的输入都是特征矩阵的整体；部分并行处理，即可定义每个工作需要输入的特征矩阵的列。

## 2.1 整体并行处理

pipeline包提供了FeatureUnion类来进行整体并行处理：



```
1 from numpy import log1p  
2 from sklearn.preprocessing import FunctionTransformer  
3 from sklearn.preprocessing import Binarizer  
4 from sklearn.pipeline import FeatureUnion  
5
```

```

6 #新建将整体特征矩阵进行对数函数转换的对象
7 step2_1 = ('ToLog', FunctionTransformer(log1p))
8 #新建将整体特征矩阵进行二值化类的对象
9 step2_2 = ('ToBinary', Binarizer())
10 #新建整体并行处理对象
11 #该对象也有fit和transform方法，fit和transform方法均是并行地调用需要并行处理的对象的fit和transform方法
12 #参数transformer_list为需要并行处理的对象列表，该列表为二元组列表，第一元为对象的名称，第二元为对象
13 step2 = ('FeatureUnion', FeatureUnion(transformer_list=[step2_1, step2_2, step2_3]))

```



## 2.2 部分并行处理

整体并行处理有其缺陷，在一些场景下，我们只需要对特征矩阵的某些列进行转换，而不是所有列。pipeline并没有提供相应的类（仅OneHotEncoder类实现了该功能），需要我们在FeatureUnion的基础上进行优化：

```

1 from sklearn.pipeline import FeatureUnion, _fit_one_transformer, _fit_transform_one, _transform_one
2 from sklearn.externals.joblib import Parallel, delayed
3 from scipy import sparse
4 import numpy as np
5
6 #部分并行处理，继承FeatureUnion
7 class FeatureUnionExt(FeatureUnion):
8     #相比FeatureUnion，多了idx_list参数，其表示每个并行工作需要读取的特征矩阵的列
9     def __init__(self, transformer_list, idx_list, n_jobs=1, transformer_weights=None):
10         self.idx_list = idx_list
11         FeatureUnion.__init__(self, transformer_list=map(lambda trans:(trans[0], trans[1]),
12 transformer_list), n_jobs=n_jobs, transformer_weights=transformer_weights)
13
14 #由于只部分读取特征矩阵，方法fit需要重构
15 def fit(self, X, y=None):
16     transformer_idx_list = map(lambda trans, idx:(trans[0], trans[1], idx),
17 self.transformer_list, self.idx_list)
18     transformers = Parallel(n_jobs=self.n_jobs)(
19         #从特征矩阵中提取部分输入fit方法
20         delayed(_fit_one_transformer)(trans, X[:,idx], y)
21         for name, trans, idx in transformer_idx_list)
22     self._update_transformer_list(transformers)
23     return self
24
25 #由于只部分读取特征矩阵，方法fit_transform需要重构
26 def fit_transform(self, X, y=None, **fit_params):
27     transformer_idx_list = map(lambda trans, idx:(trans[0], trans[1], idx),
28 self.transformer_list, self.idx_list)
29     result = Parallel(n_jobs=self.n_jobs)(
30         #从特征矩阵中提取部分输入fit_transform方法
31         delayed(_fit_transform_one)(trans, name, X[:,idx], y,
32                                     self.transformer_weights, **fit_params)
33         for name, trans, idx in transformer_idx_list)
34
35     Xs, transformers = zip(*result)
36     self._update_transformer_list(transformers)
37     if any(sparse.issparse(f) for f in Xs):
38         Xs = sparse.hstack(Xs).tocsr()

```

```
36         else:
37             Xs = np.hstack(Xs)
38         return Xs
39
40     #由于只部分读取特征矩阵，方法transform需要重构
41     def transform(self, X):
42         transformer_idx_list = map(lambda trans, idx:(trans[0], trans[1], idx),
self.transformer_list, self.idx_list)
43         Xs = Parallel(n_jobs=self.n_jobs)(
44             #从特征矩阵中提取部分输入transform方法
45             delayed(_transform_one)(trans, name, X[:,idx], self.transformer_weights)
46             for name, trans, idx in transformer_idx_list)
47         if any(sparse.issparse(f) for f in Xs):
48             Xs = sparse.hstack(Xs).tocsr()
49         else:
50             Xs = np.hstack(Xs)
51         return Xs
```



在本文提出的场景中，我们对特征矩阵的第1列（花的颜色）进行定性特征编码，对第2、3、4列进行对数函数转换，对第5列进行定量特征二值化处理。使用FeatureUnionExt类进行部分并行处理的代码如下：

```
1 from numpy import log1p
2 from sklearn.preprocessing import OneHotEncoder
3 from sklearn.preprocessing import FunctionTransformer
4 from sklearn.preprocessing import Binarizer
5
6 #新建将部分特征矩阵进行定性特征编码的对象
7 step2_1 = ('OneHotEncoder', OneHotEncoder(sparse=False))
8 #新建将部分特征矩阵进行对数函数转换的对象
9 step2_2 = ('ToLog', FunctionTransformer(log1p))
10 #新建将部分特征矩阵进行二值化类的对象
11 step2_3 = ('ToBinary', Binarizer())
12 #新建部分并行处理对象
13 #参数transformer_list为需要并行处理的对象列表，该列表为二元组列表，第一元为对象的名称，第二元为对象
14 #参数idx_list为相应的需要读取的特征矩阵的列
15 step2 = ('FeatureUnionExt', FeatureUnionExt(transformer_list=[step2_1, step2_2, step2_3], idx_list=
[[0], [1, 2, 3], [4]]))
```



### 3 流水线处理

pipeline包提供了Pipeline类来进行流水线处理。流水线上除最后一个工作以外，其他都要执行fit\_transform方法，且上一个工作输出作为下一个工作的输入。最后一个工作必须实现fit方法，输入为上一个工作的输出；但是不限定一定有transform方法，因为流水线的最后一个工作可能是训练！

根据本文提出的场景，结合并行处理，构建完整的流水线的代码如下：

```
1 from numpy import log1p
2 from sklearn.preprocessing import Imputer
3 from sklearn.preprocessing import OneHotEncoder
```

```
4 from sklearn.preprocessing import FunctionTransformer
5 from sklearn.preprocessing import Binarizer
6 from sklearn.preprocessing import MinMaxScaler
7 from sklearn.feature_selection import SelectKBest
8 from sklearn.feature_selection import chi2
9 from sklearn.decomposition import PCA
10 from sklearn.linear_model import LogisticRegression
11 from sklearn.pipeline import Pipeline
12
13 #新建计算缺失值的对象
14 step1 = ('Imputer', Imputer())
15 #新建将部分特征矩阵进行定性特征编码的对象
16 step2_1 = ('OneHotEncoder', OneHotEncoder(sparse=False))
17 #新建将部分特征矩阵进行对数函数转换的对象
18 step2_2 = ('ToLog', FunctionTransformer(log1p))
19 #新建将部分特征矩阵进行二值化类的对象
20 step2_3 = ('ToBinary', Binarizer())
21 #新建部分并行处理对象，返回值为每个并行工作的输出的合并
22 step2 = ('FeatureUnionExt', FeatureUnionExt(transformer_list=[step2_1, step2_2, step2_3], idx_list=
[[0], [1, 2, 3], [4]]))
23 #新建无量纲化对象
24 step3 = ('MinMaxScaler', MinMaxScaler())
25 #新建卡方校验选择特征的对象
26 step4 = ('SelectKBest', SelectKBest(chi2, k=3))
27 #新建PCA降维的对象
28 step5 = ('PCA', PCA(n_components=2))
29 #新建逻辑回归的对象，其为待训练的模型作为流水线的最后一步
30 step6 = ('LogisticRegression', LogisticRegression(penalty='l2'))
31 #新建流水线处理对象
32 #参数steps为需要流水线处理的对象列表，该列表为二元组列表，第一元为对象的名称，第二元为对象
33 pipeline = Pipeline(steps=[step1, step2, step3, step4, step5, step6])
```



## 4 自动化调参

网格搜索为自动化调参的常见技术之一，grid\_search包提供了自动化调参的工具，包括GridSearchCV类。对组合好的对象进行训练以及调参的代码如下：

```
1 from sklearn.grid_search import GridSearchCV
2
3 #新建网格搜索对象
4 #第一参数为待训练的模型
5 #param_grid为待调参数组成的网格，字典格式，键为参数名称（格式“对象名称__子对象名称__参数名称”），值为可取的参数值列表
6 grid_search = GridSearchCV(pipeline, param_grid={'FeatureUnionExt__ToBinary__threshold':[1.0, 2.0, 3.0, 4.0], 'LogisticRegression__C':[0.1, 0.2, 0.4, 0.8]})
7 #训练以及调参
8 grid_search.fit(iris.data, iris.target)
```



## 5 持久化



```
1 #持久化数据
2 #第一个参数为内存中的对象
3 #第二个参数为保存在文件系统中的名称
4 #第三个参数为压缩级别，0为不压缩，3为合适的压缩级别
5 dump(grid_search, 'grid_search.dmp', compress=3)
6 #从文件系统中加载数据到内存中
7 grid_search = load('grid_search.dmp')
```



## 6 回顾

包	类或方法	说明
sklearn.pipeline	Pipeline	流水线处理
sklearn.pipeline	FeatureUnion	并行处理
sklearn.grid_search	GridSearchCV	网格搜索调参
externals.joblib	dump	数据持久化
externals.joblib	load	从文件系统中加载数据至内存

注意：组合和持久化都会涉及pickle技术，在sklearn的技术文档中有说明，将lambda定义的函数作为FunctionTransformer的自定义转换函数将不能pickle化。

## 7 总结

2015年我设计了一个基于sklearn的自动化特征工程的工具，其以Mysql数据库作为原始数据源，提供了“灵活的”特征提取、特征处理的配置方法，同时重新封装了数据、特征和模型，以方便调度系统识别。说灵活，其实也只是通过配置文件的方式定义每个特征的提取和处理的sql语句。但是纯粹使用sql语句来进行特征处理是很勉强的，除去特征提取以外，我又造了一回轮子，原来sklearn提供了这么优秀的特征处理、工作组合等功能。所以，我在这个博客中先不提任何算法和模型，先从数据挖掘工作的第一步开始，使用基于Python的各个工具把大部分步骤都走了一遍（抱歉，我暂时忽略了特征提取），希望这样的梳理能够少让初学者走弯路吧。

## 8 参考资料

1. 使用sklearn做单机特征工程
2. FunctionTransformer
3. Github:jasonfreak/ali2015