



《联邦学习下特征工程的算法优化》

|  |  |
| --- | --- |
| **专 业** | **信息安全** |
| **姓 名** | **毛佳男** |
| **日 期** | **2022/09/01** |

**摘 要**

本文主要介绍一下在银联背景下的联邦学习的特征算法和算法的优化和改进，关于实现流程和算法优化都是基于500万基数的数据的情况下进行的。本文还包括一些纵向联邦的简单实现，其中以逻辑递归为主，同时也包含着对于联邦分箱的一些自己的理解。

**关键词：**联邦学习，特征分箱，数据处理，特征降维

**目 录**

[第1章 联邦学习 - 4 -](#_Toc14478)

[第2章 特征工程 - 4 -](#_Toc14478)

[2.1 特征预处理 - 4 -](#_Toc8865)

[2.2 特征选择 - 4 -](#_Toc23216)

[2.3 特征降维 - 4 -](#_Toc22557)

[第3章 联邦特征工程 - 4 -](#_Toc14478)

[第4章 特征工程数据处理流程 - 5 -](#_Toc14478)

[4.1 纵向求交-填充空白值-采样-归一化](#_Toc8865)

[- 5 -](#_Toc8865)

[4.2 纵向求交-填充空白值-特征分箱-特征选择](#_Toc23216)

[- 11 -](#_Toc23216)

[4.3 纵向求交-填充空白值-采样-归一化-分箱-特征选择](#_Toc22557)

[- 16 -](#_Toc22557)

[4.4 纵向求交-数据剥离-数据统计-填充空白值-采样-归一化-分箱-特征选择](#_Toc8865)

[- 22 -](#_Toc8865)

[4.5 纵向求交-采样-归一化-分箱-特征选择-one hot编码](#_Toc23216)

[- 29 -](#_Toc23216)

[第5章 特征降维 - 35 -](#_Toc14478)

[参考文献 - 38 -](#_Toc1841)

### 1.联邦学习

一个全新事物的产生的动力一定是市场的需求，随着法规和监管的愈发严格，人们对于数据安全和隐私的保护越来越看重，但是数据仅仅是在某个企业或者组织内部流通，无法互相沟通，我们成为哦这种现象为“数据孤岛”。因此联邦学习应运而生，简单来说联邦学习是一种分布式机器学习框架，可以让各企业在不共享数据的基础上进行联合训练，从而可以解决数据孤岛的难题。

### 2.特征工程

在机器学习的领域一直有这么一句话“数据和特征决定了机器学习的上限，但是模型和算法就是无限趋近这个上限而已”。其中特征工程指的就是将数据加工为模型训练的过程，目的是极大程度地提取出数据的信息供模型使用。

基于特征工程一般有三种处理方法，分别是：特征预处理、特征选择、特征降维。

#### 2.1特征预处理

我们要处理的原始数据中，只要数据量足够大（一般都是几百万条数据）一定会在其中夹杂中很多脏数据，所以我们需要进行异常特征样本处理、缺失值处理、标准化和归一化等操作

#### 2.2特征选择

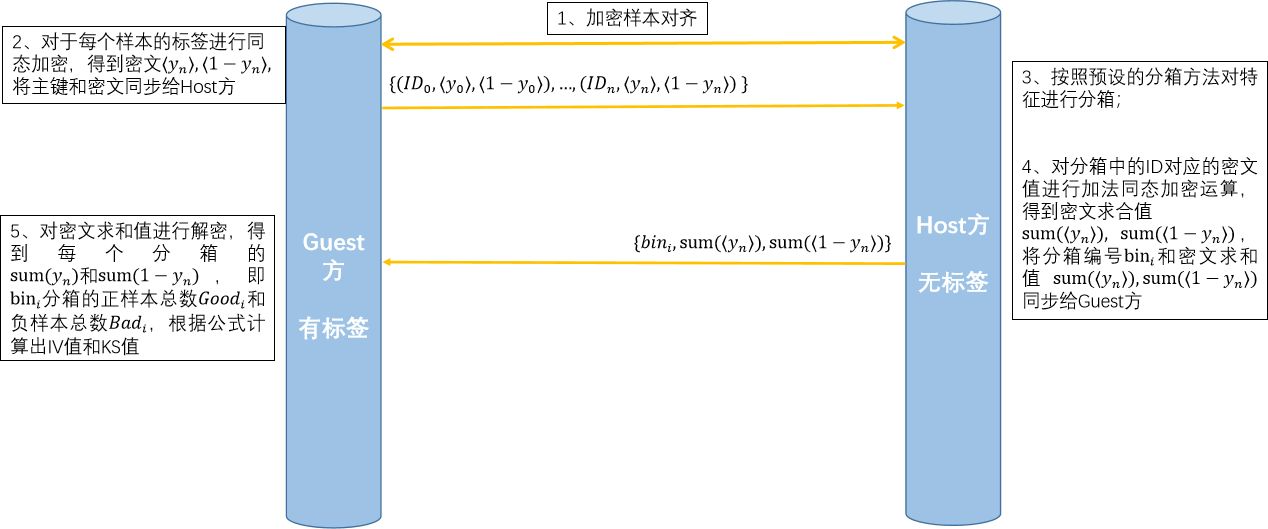
在进行特征预处理之后，我们需要进一步选择有意义的数据放进模型中训练。方法目前我们研究的就是特征单变量分析，顾名思义就是对一个变量进行分析，分析每个特征的分布情况对于标签的预测能力，常见的包括WOE、IV、KS和PSI等等，我们主要使用的是WOE和IV，详细的内容在后面联邦特征工程中会详细说明。

#### 2.3特征降维

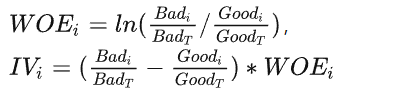
虽然我们进行了特征预处理和特征选择，从一个很庞大的数据集中选出了一部分数据对于模型训练有很大意义。但是由于我们数据的基数过大，所以还是会导致最后产生的特征矩阵过大，造成计算效率较低等问题，我们可以通过特征降维的方法去解决，我们后面会提及PCA主成分分析方法。

### 3.联邦特征工程

什么是联邦特征工程，简单来说就是通过联邦的手段借助特征工程让双方的信息进行安全交互，一般来说我们把持有标签的一方称作Guest方，（那么什么叫做标签呢？简单的说，银行的信贷标签，电商的消费数据都可以作为标签）无标签的一方称作Host方，Guest方想要评估Host方拥有的数据，所通过的评估指标就是我们刚才所提到的WOE/VI值。操作步骤如下：



这里简单你解释一下上述流程，因为这不是本文的重点。首先的加密样本对齐就是通过某些手段使得host和guest端样本数据完全相同，简单来说就是共用一个样本。然后对guest端标签进行同态加密，然后数据处理进行特征分箱，然后对密文同态加密再传给guest，然后guest端对于之前的分箱中的正负样本数带入WOE和IV值的计算公式中判断。WOE和IV值的公式如下:



其中Badi和Goodi是每个分箱中好坏样本的数量，BadT和GoodT是所有样本的好坏样本数量。

以上内容算是对于联邦特征工程有一个大概的了解，下面我们来具体说说特征分箱中的数据处理。

### 4.特征工程数据处理流程

联邦学习按照数据分成横向联邦学习、纵向联邦学习和迁移联邦学习。我们主要讲述的是纵向联邦学习。后面提到的dsl文件中包含的就是我们所需要的组件以及继承关系，而conf文件中包含组件中的参数及应用。

#### 4.1纵向求交-填充空白值-采样-归一化

采样过程是为了让特征数量的比例更加接近，避免失衡的数量比。

归一化的目的是为了改变特征的数值范围，如果两个特征的数量值差距太大，在逻辑回归中将会出现问题。

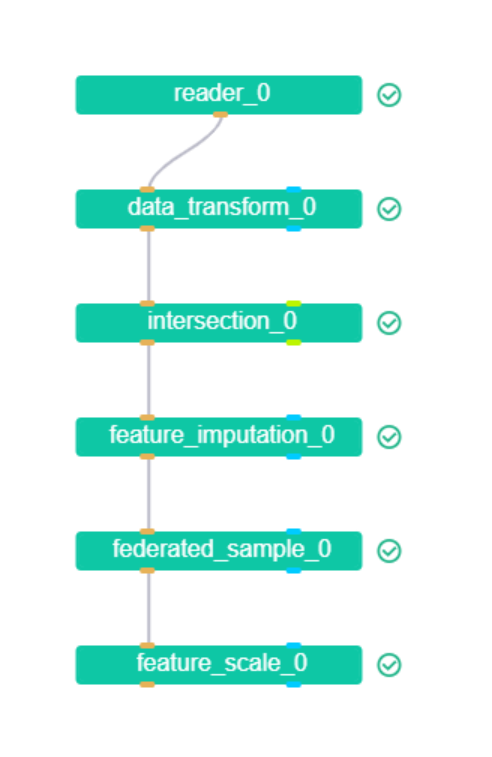
根据数据特点，可以在参数之中修改采样比例，从而达到自己所期望的样本特征比。

在这里如何选择两种归一化的方法？

standard scale其实是最常用的方法，改变特征值的分布，使其变为均值0，方差1，在分类、聚类算法中常用。

min-max scale不太常用，他是为了改变特征的数值范围来使用的。

所以说，以下流程适用于我们需要进行逻辑回归过程的特征。



dsl文件如下：

{

"components": {

"reader\_0": {

"module": "Reader",

"output": {

"data": [

"data"

]

}

},

"data\_transform\_0": {

"module": "DataTransform",

"input": {

"data": {

"data": [

"reader\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

},

"intersection\_0": {

"module": "Intersection",

"input": {

"data": {

"data": [

"data\_transform\_0.data"

]

}

},

"output": {

"data": [

"data"

]

}

},

"feature\_imputation\_0": {

"module": "FeatureImputation",

"input": {

"data": {

"data": [

"intersection\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

},

"federated\_sample\_0": {

"module": "FederatedSample",

"input": {

"data": {

"data": [

"feature\_imputation\_0.data"

]

}

},

"output": {

"data": [

"data"

]

}

},

"feature\_scale\_0": {

"module": "FeatureScale",

"input": {

"data": {

"data": [

"federated\_sample\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

}

}

}

conf文件如下：

{

"dsl\_version": 2,

"initiator": {

"role": "guest",

"party\_id": 9999

},

"role": {

"host": [

10000

],

"guest": [

9999

]

},

"component\_parameters": {

"role": {

"guest": {

"0": {

"reader\_0": {

"table": {

"name": "give\_credit\_hetero\_guest",

"namespace": "experiment"

}

},

"data\_transform\_0": {

"with\_label": true,

"output\_format": "dense"

}

}

},

"host": {

"0": {

"reader\_0": {

"table": {

"name": "give\_credit\_hetero\_host",

"namespace": "experiment"

}

},

"data\_transform\_0": {

"with\_label": false

}

}

}

},

/\*

此处可以选择采样率，在fractions参数中，为二元组。前一个数表示列号，后一个表示采样率

\*/

"common":

{

"federated\_sample\_0": {

"mode": "stratified",

"method": "upsample",

"fractions": [

[

0,

1.5

],

[

1,

2.0

]

]

},

/\*

此处可以选择归一化操作进行的特征列，参数为scale\_col\_indexes,数字的list形式

或者scale\_names，string的list形式

默认为全选

\*/

"feature\_scale\_0": {

"method": "min\_max\_scale",

"mode": "cap",

"feat\_upper": 1,

"feat\_lower": 0

},

"feature\_imputation\_0": {

"missing\_fill\_method": "max",

"missing\_impute": [

0

]

}

}

}

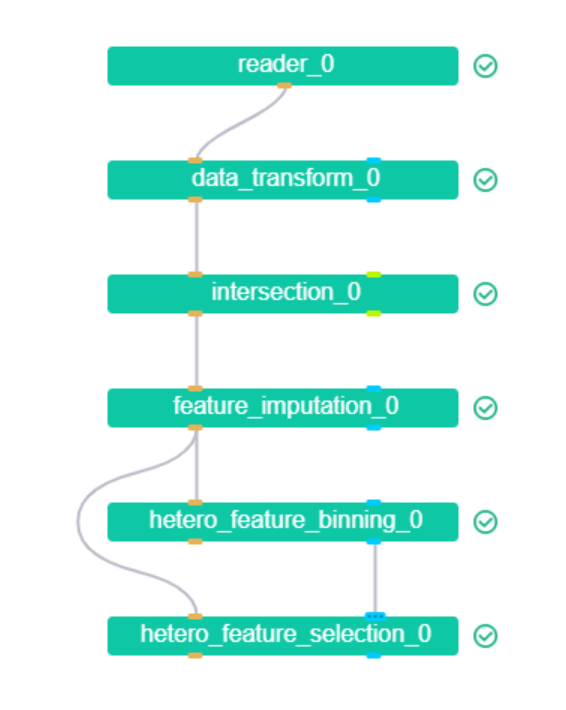
}

#### 4.2纵向求交-填充空白值-特征分箱-特征选择

特征选择的过程主要是为了去除掉“冗余”的特征，避免掉训练时间过长、模型过拟合、模型解释性差等问题。

在联邦学习中，我们使用iv值来代表一种特征的解释能力。并且在特征选择的过程中，我们采用过滤法（iv值）来滤出iv值较高，也就是解释能力较强的特征。

分箱过程也就是为了对iv值进行一个初步的划分和计算。所以需要加在特征选择之前。



dsl文件如下：

{

"components": {

"reader\_0": {

"module": "Reader",

"output": {

"data": [

"data"

]

}

},

"data\_transform\_0": {

"module": "DataTransform",

"input": {

"data": {

"data": [

"reader\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

},

"intersection\_0": {

"module": "Intersection",

"input": {

"data": {

"data": [

"data\_transform\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"cache": [

"cache"

]

}

},

"feature\_imputation\_0": {

"module": "FeatureImputation",

"input": {

"data": {

"data": [

"intersection\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

},

"hetero\_feature\_binning\_0": {

"module": "HeteroFeatureBinning",

"input": {

"data": {

"data": [

"feature\_imputation\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

},

"hetero\_feature\_selection\_0": {

"module": "HeteroFeatureSelection",

"input": {

"data": {

"data": [

"feature\_imputation\_0.data"

]

},

"isometric\_model": [

"hetero\_feature\_binning\_0.model"

]

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

}

}

}

conf文件如下：

{

"dsl\_version": 2,

"initiator": {

"role": "guest",

"party\_id": 9999

},

"role": {

"host": [

10000

],

"guest": [

9999

]

},

"component\_parameters": {

"role": {

"guest": {

"0": {

"reader\_0": {

"table": {

"name": "give\_credit\_hetero\_guest",

"namespace": "experiment"

}

},

"data\_transform\_0": {

"with\_label": true,

"output\_format": "dense"

}

}

},

"host": {

"0": {

"reader\_0": {

"table": {

"name": "give\_credit\_hetero\_host",

"namespace": "experiment"

}

},

"data\_transform\_0": {

"with\_label": false

}

}

}

},

"common":

{

"feature\_imputation\_0": {

"missing\_fill\_method": "max",

"missing\_impute": [

0

]

},

"hetero\_feature\_binning\_0": {

"method": "quantile",

"compress\_thres": 10000,

"head\_size": 10000,

"error": 0.001,

"adjustment\_factor": 0.5,

"bin\_num": 10,

"bin\_indexes": -1,

"bin\_names": null,

"category\_indexes": null,

"category\_names": null,

"transform\_param": {

"transform\_cols": -1,

"transform\_names": null,

"transform\_type": "bin\_num"

},

"local\_only": false

},

"hetero\_feature\_selection\_0": {

"select\_col\_indexes": -1,

"select\_names": [],

"filter\_methods": [

"iv\_value\_thres"

],

"iv\_value\_param": {

"value\_threshold": 0.1

}

}

}

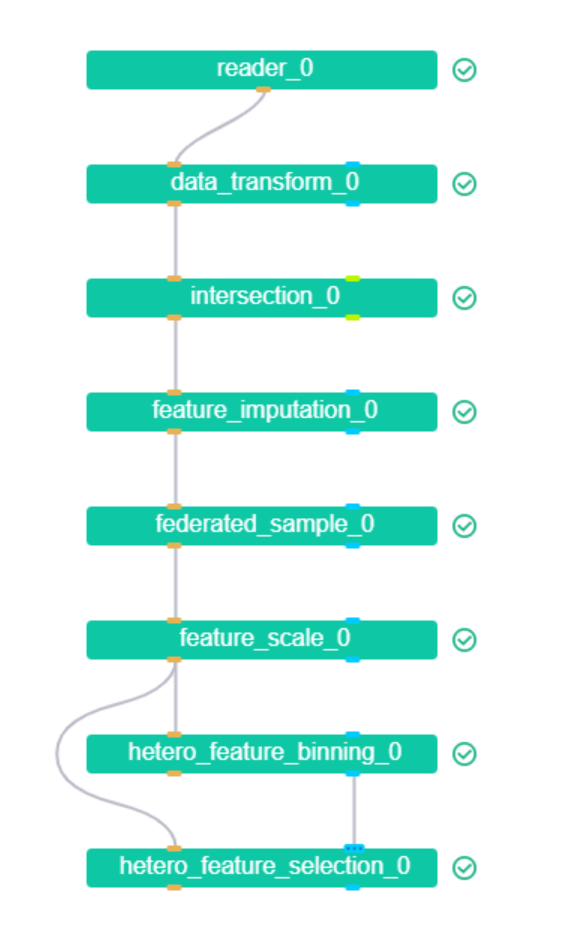
}

}

#### 4.3纵向求交-填充空白值-采样-归一化-分箱-特征选择

相当于是以上两种方法的一个中和。

对于归一化的特征再进行了一个分箱以及特征选择的过程。



dsl文件如下：

{

"components": {

"reader\_0": {

"module": "Reader",

"output": {

"data": [

"data"

]

}

},

"data\_transform\_0": {

"module": "DataTransform",

"input": {

"data": {

"data": [

"reader\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

},

"intersection\_0": {

"module": "Intersection",

"input": {

"data": {

"data": [

"data\_transform\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"cache": [

"cache"

]

}

},

"feature\_imputation\_0": {

"module": "FeatureImputation",

"input": {

"data": {

"data": [

"intersection\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

},

"federated\_sample\_0": {

"module": "FederatedSample",

"input": {

"data": {

"data": [

"feature\_imputation\_0.data"

]

}

},

"output": {

"data": [

"data"

]

}

},

"feature\_scale\_0": {

"module": "FeatureScale",

"input": {

"data": {

"data": [

"federated\_sample\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

},

"hetero\_feature\_binning\_0": {

"module": "HeteroFeatureBinning",

"input": {

"data": {

"data": [

"feature\_scale\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

},

"hetero\_feature\_selection\_0": {

"module": "HeteroFeatureSelection",

"input": {

"data": {

"data": [

"intersection\_0.data"

]

},

"isometric\_model": [

"hetero\_feature\_binning\_0.model"

]

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

}

}

}

conf文件如下：

{

"dsl\_version": 2,

"initiator": {

"role": "guest",

"party\_id": 9999

},

"role": {

"host": [

10000

],

"guest": [

9999

]

},

"component\_parameters": {

"role": {

"guest": {

"0": {

"reader\_0": {

"table": {

"name": "breast\_hetero\_guest",

"namespace": "experiment"

}

},

"data\_transform\_0": {

"with\_label": true,

"output\_format": "dense"

}

}

},

"host": {

"0": {

"reader\_0": {

"table": {

"name": "breast\_hetero\_host",

"namespace": "experiment"

}

},

"data\_transform\_0": {

"with\_label": false

}

}

}

},

"common": {

"federated\_sample\_0": {

"mode": "stratified",

"method": "upsample",

"fractions": [

[

0,

1.5

],

[

1,

2.0

]

]

},

"feature\_scale\_0": {

"method": "min\_max\_scale",

"mode": "cap",

"feat\_upper": 1,

"feat\_lower": 0

},

"feature\_imputation\_0": {

"missing\_fill\_method": "max",

"missing\_impute": [

0

]

},

"hetero\_feature\_binning\_0": {

"method": "quantile",

"compress\_thres": 10000,

"head\_size": 10000,

"error": 0.001,

"adjustment\_factor": 0.5,

"bin\_num": 10,

"bin\_indexes": -1,

"bin\_names": null,

"category\_indexes": null,

"category\_names": null,

"transform\_param": {

"transform\_cols": -1,

"transform\_names": null,

"transform\_type": "bin\_num"

},

"local\_only": false

},

"hetero\_feature\_selection\_0": {

"select\_col\_indexes": -1,

"select\_names": [],

"filter\_methods": [

"iv\_value\_thres"

],

"iv\_value\_param": {

"value\_threshold": 0.1

}

}

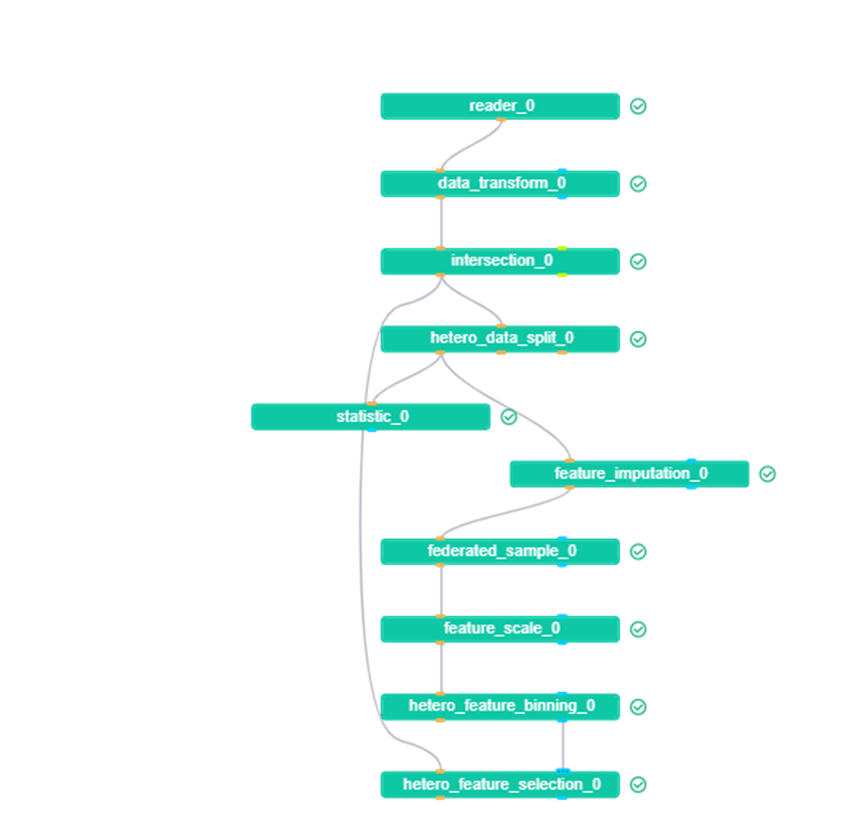
}

}

}

#### 4.4纵向求交-数据剥离-数据统计-填充空白值-采样-归一化-分箱-特征选择

在data\_split的基础之上加入了data\_statistic组件（当然也可以不加入data\_split组件直接进行data\_statistic），该组件可以计算数据特征，也可作为特征选择的依据



dsl文件如下：

{

"components": {

"reader\_0": {

"module": "Reader",

"output": {

"data": [

"data"

]

}

},

"data\_transform\_0": {

"module": "DataTransform",

"input": {

"data": {

"data": [

"reader\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

},

"intersection\_0": {

"module": "Intersection",

"input": {

"data": {

"data": [

"data\_transform\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"cache": [

"cache"

]

}

},

"hetero\_data\_split\_0": {

"module": "HeteroDataSplit",

"input": {

"data": {

"data": [

"intersection\_0.data"

]

}

},

"output": {

"data": [

"train\_data",

"validate\_data",

"test\_data"

]

}

},

"statistic\_0": {

"module": "DataStatistics",

"input": {

"data": {

"data": [

"hetero\_data\_split\_0.train\_data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

},

"feature\_imputation\_0": {

"module": "FeatureImputation",

"input": {

"data": {

"data": [

"hetero\_data\_split\_0.train\_data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

},

"federated\_sample\_0": {

"module": "FederatedSample",

"input": {

"data": {

"data": [

"feature\_imputation\_0.data"

]

}

},

"output": {

"data": [

"data"

]

}

},

"feature\_scale\_0": {

"module": "FeatureScale",

"input": {

"data": {

"data": [

"federated\_sample\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

},

"hetero\_feature\_binning\_0": {

"module": "HeteroFeatureBinning",

"input": {

"data": {

"data": [

"feature\_scale\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

},

"hetero\_feature\_selection\_0": {

"module": "HeteroFeatureSelection",

"input": {

"data": {

"data": [

"intersection\_0.data"

]

},

"isometric\_model": [

"hetero\_feature\_binning\_0.model"

]

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

}

}

}

conf文件如下：

{

"dsl\_version": 2,

"initiator": {

"role": "guest",

"party\_id": 9999

},

"role": {

"host": [

10000

],

"guest": [

9999

]

},

"component\_parameters": {

"role": {

"guest": {

"0": {

"reader\_0": {

"table": {

"name": "breast\_hetero\_guest",

"namespace": "experiment"

}

},

"data\_transform\_0": {

"with\_label": true,

"output\_format": "dense"

}

}

},

"host": {

"0": {

"reader\_0": {

"table": {

"name": "breast\_hetero\_host",

"namespace": "experiment"

}

},

"data\_transform\_0": {

"with\_label": false

}

}

}

},

"common": {

"federated\_sample\_0": {

"mode": "stratified",

"method": "upsample",

"fractions": [

[

0,

1.5

],

[

1,

2.0

]

]

},

"feature\_scale\_0": {

"method": "min\_max\_scale",

"mode": "cap",

"feat\_upper": 1,

"feat\_lower": 0

},

"feature\_imputation\_0": {

"missing\_fill\_method": "max",

"missing\_impute": [

0

]

},

"hetero\_feature\_binning\_0": {

"method": "quantile",

"compress\_thres": 10000,

"head\_size": 10000,

"error": 0.001,

"adjustment\_factor": 0.5,

"bin\_num": 10,

"bin\_indexes": -1,

"bin\_names": null,

"category\_indexes": null,

"category\_names": null,

"transform\_param": {

"transform\_cols": -1,

"transform\_names": null,

"transform\_type": "bin\_num"

},

"local\_only": false

},

"hetero\_feature\_selection\_0": {

"select\_col\_indexes": -1,

"select\_names": [],

"filter\_methods": [

"iv\_value\_thres"

],

"iv\_value\_param": {

"value\_threshold": 0.1

}

}

}

}

}

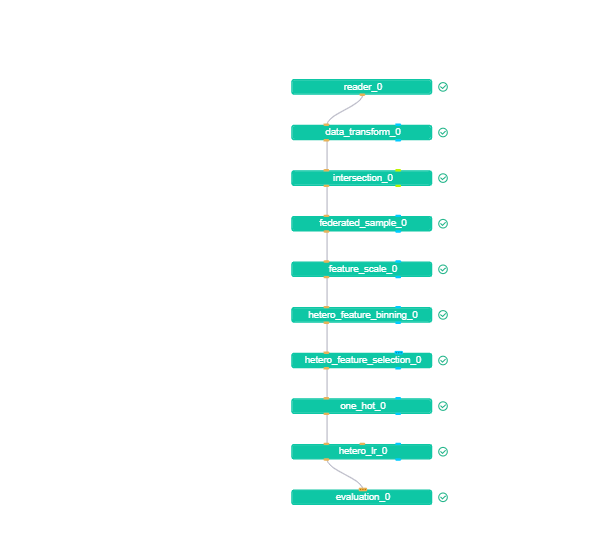
#### 4.5纵向求交-采样-归一化-分箱-特征选择-one hot编码

one hot编码是让一些无序的**离散特征**转化成为连续有序的离散特征，以有利于机器学习算法的分类器工作。

其也可以说在某种程度上扩充了特征，其中的一种类别就可以看待为一种特征。

为什么这里在one hot组件之前添加了特征分箱和特征选择的过程呢？因为give\_credit中的数据特征值都是连续值，直接使用one hot不可行，我们需要分箱之后，将所在箱号对应的值进行one hot编码，这样才是可行的。

在实际数据中其实可以直接采用one hot组件，例如对于性别这一特征。



dsl文件：

{

"components": {

"reader\_0": {

"module": "Reader",

"output": {

"data": [

"data"

]

}

},

"data\_transform\_0": {

"module": "DataTransform",

"input": {

"data": {

"data": [

"reader\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

},

"intersection\_0": {

"module": "Intersection",

"input": {

"data": {

"data": [

"data\_transform\_0.data"

]

}

},

"output": {

"data": [

"data"

]

}

},

"federated\_sample\_0": {

"module": "FederatedSample",

"input": {

"data": {

"data": [

"intersection\_0.data"

]

}

},

"output": {

"data": [

"data"

]

}

},

"feature\_scale\_0": {

"module": "FeatureScale",

"input": {

"data": {

"data": [

"federated\_sample\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

},

"hetero\_feature\_binning\_0": {

"module": "HeteroFeatureBinning",

"input": {

"data": {

"data": [

"feature\_scale\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

},

"hetero\_feature\_selection\_0": {

"module": "HeteroFeatureSelection",

"input": {

"data": {

"data": [

"hetero\_feature\_binning\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

},

"one\_hot\_0": {

"module": "OneHotEncoder",

"input": {

"data": {

"data": [

"hetero\_feature\_selection\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

},

"hetero\_lr\_0": {

"module": "HeteroLR",

"input": {

"data": {

"train\_data": [

"one\_hot\_0.data"

]

}

},

"output": {

"data": [

"data"

],

"model": [

"model"

]

}

},

"evaluation\_0": {

"module": "Evaluation",

"input": {

"data": {

"data": [

"hetero\_lr\_0.data"

]

}

}

}

}

}

conf文件：

{

"dsl\_version": 2,

"initiator": {

"role": "guest",

"party\_id": 9999

},

"role": {

"arbiter": [

10000

],

"host": [

10000

],

"guest": [

9999

]

},

"component\_parameters": {

"common": {

"federated\_sample\_0": {

"mode": "stratified",

"method": "upsample",

"fractions": [

[

0,

1.5

],

[

1,

2.0

]

]

},

"hetero\_lr\_0": {

"penalty": "L2",

"tol": 1e-05,

"alpha": 0.01,

"optimizer": "rmsprop",

"batch\_size": 320,

"learning\_rate": 0.15,

"init\_param": {

"init\_method": "random\_uniform"

},

"max\_iter": 10,

"early\_stop": "diff"

},

"feature\_scale\_0": {

"method": "min\_max\_scale",

"mode": "cap",

"feat\_upper": 1,

"feat\_lower": 0

}

},

"role": {

"guest": {

"0": {

"data\_transform\_0": {

"missing\_fill": true,

"outlier\_replace": true,

"with\_label": true

},

"reader\_0": {

"table": {

"name": "breast\_hetero\_guest",

"namespace": "experiment"

}

}

}

},

"host": {

"0": {

"data\_transform\_0": {

"missing\_fill": true,

"outlier\_replace": true,

"with\_label": false

},

"reader\_0": {

"table": {

"name": "breast\_hetero\_host",

"namespace": "experiment"

}

}

}

}

}

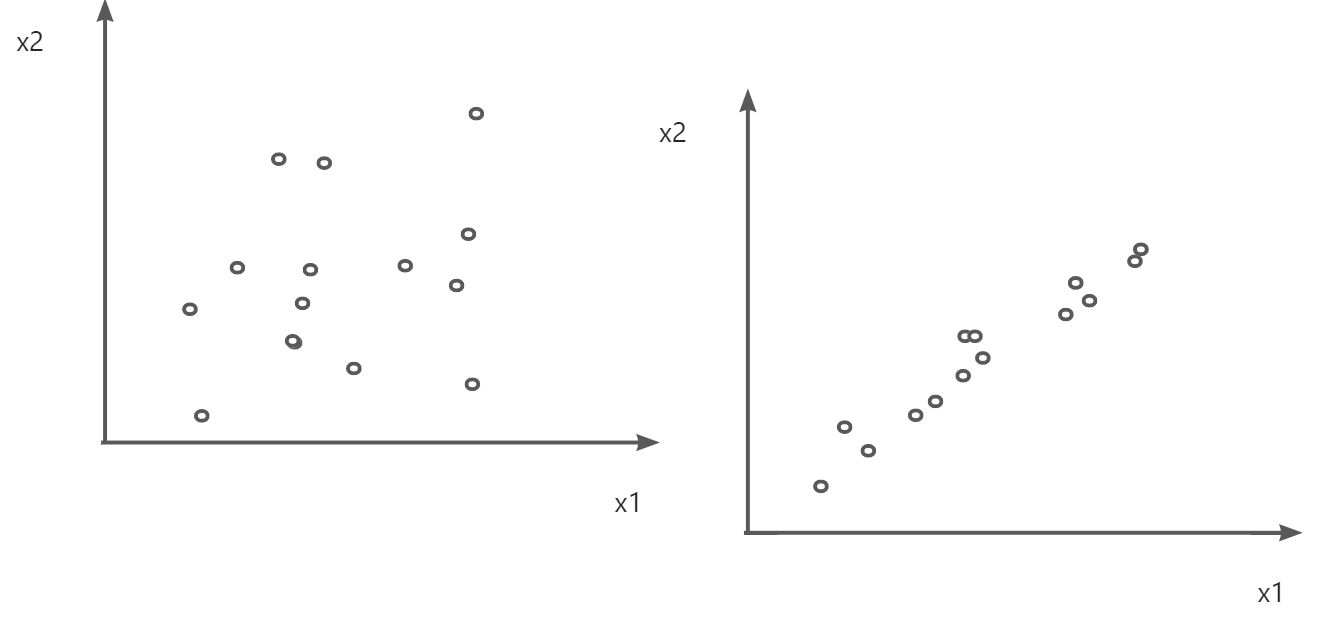
}

}

以上就是一些关于数据处理的流程及其代码实现，关于各个组件内部具体的参数设置以及数据处理这里就不过多阐述了，还有一些特征数据的处理比如说decimal、时间戳数据等等，在这里就不过多叙述了，主要的思想就是讲其转换成字符串处理。接下来我们再最后简单说说特征降维。

### 5.特征降维

关于特征降维的部分我门就主要说说关于PCA主成分分析的方法。首先我们先要考虑什么是降维？就是将一些无法给到有用信息的特征维度删去。接下来我们考虑为什么要降维？就是为了避免这些无用的特征浪费空间和计算能力。那么我们凭借什么去判断是有用还是无用呢？简单举个例子，如下图现在我们有两个特征x1和x2，圆形为数据点。

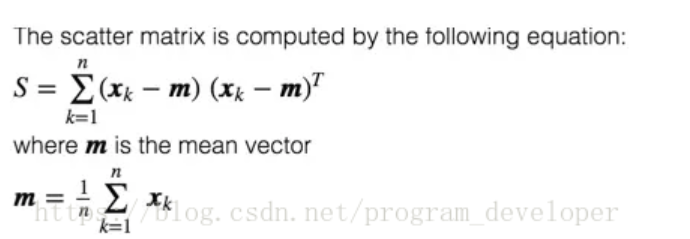


左图中的数据点比较均匀，看起来是几乎充满了整个空间。而在右图中的数据点，虽然其不是完全线性相关，但是可以将x1和x2之中的一维视作无用的，因为另一个特征可以表示它。所以我们可以用一个新特征，也就是这条线的维度，来取代x1和x2。

那么我们如何可以做到PCA降维呢？

首先是协方差，两个特征维度的协方差的绝对值越大，那么他们相关关系越大。而对于多维数据，其协方差是一个矩阵，叫做协方差矩阵，三维协方差矩阵如下：

协方差矩阵又可以由一个叫做散度矩阵的矩阵乘以N-1得到。而散度矩阵的计算如下图，是比较容易的。



特征值分解矩阵：

就是将矩阵A分解为下式：

其中，W是一个对角阵，对角线上的元素就是特征值。

PCA的具体流程如下（用特征值分解矩阵的方法）：

1) 去平均值(即去中心化)，即每一位特征减去各自的平均值。

2) 计算协方差矩阵。

3) 用特征值分解方法求协方差矩阵的特征值与特征向量。

4) 对特征值从大到小排序，选择其中最大的k个。然后将其对应的k个特征向量分别作为行向量组成特征向量矩阵P。

5) 将数据转换到k个特征向量构建的新空间中，即Y=PX。

还有另一种实现方式，是基于SVD对于协方差矩阵的分解，SVD分解矩阵的原理是：

对于每一个矩阵A，有：

若A为n\*m矩阵，那么U是一个n\*n的方阵，其中的正交向量叫做左奇异向量，W是n\*m的对角矩阵，对角线上的元素叫奇异值，V是一个m\*m的矩阵，其正交向量叫做右奇异向量。

在python中有关机器学习的库sklearn之中，有一个直接实现PCA的函数PCA，其使用方法如下：

from sklearn.decomposition import PCA

import numpy as np

X = np.array([[-1, 1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])

pca = PCA(n\_components=1)

pca.fit(X)

print(pca.transform(X))

其中的n\_components就是代表了降维后你所期望的特征维数。

其中，其内部实现如下：

import numpy as np

class PCA():

def \_\_init\_\_(self,n\_components):

self.n\_components = n\_components #降维的维度数

def fit\_transform(self,X):

self.n\_features\_ = X.shape[1] #数组的列数

# 求协方差矩阵

X = X - X.mean(axis=0) #计算每一列的均值

self.covariance = np.dot(X.T,X)/X.shape[0] # np.dot 矩阵乘法 shape[0]数组的行数

print("协方差矩阵为：\n",self.covariance)

# 求协方差矩阵的特征向量和特征值

eig\_vals,eig\_vectors = np.linalg.eig(self.covariance) #求特征值和特征向量的接口

print("协方差矩阵的特征值是：\n",eig\_vals)

print("协方差矩阵的特征向量是：\n", eig\_vectors)

# 获得降序排列特征值的序号

idx = np.argsort(-eig\_vals) # argsort()函数是将x中的元素从小到大排列，提取其对应的index(索引)，然后输出

print("特征值从大到小排序的序列号是：\n",idx)

self.n\_components\_ = eig\_vectors[:,idx[:self.n\_components]] #降维矩阵

#a[:,x]的含义为取每一维的第x个数据

print("降维矩阵为：\n",self.n\_components\_)

return np.dot(X,self.n\_components\_)

pca = PCA(n\_components=2)

X = np.array([[-1,2,66,-1],[-2,6,58,-1],[-3,8,45,-2],[1,9,36,1],[2,10,62,1],[3,5,83,2]])

newX = pca.fit\_transform(X

**参考文献**

[1] 林来兴. 空间控制技术[M]. 北京：中国宇航出版社，1992：25-42.

[2] 辛希孟. 信息技术与信息服务国际研讨会论文集：A集[C]. 北京：中国科学出版社，1999.

[3] 赵耀东. 新时代的工业工程师[M/OL]. 台北：天下文化出版社，1998 [1998-09-26]. http://www.ie.nthu.edu.tw/info/ie.newie.htm（Big5）.

……

[12] 谌颖. 空间交会控制理论与方法研究[D]. 哈尔滨：哈尔滨工业大学，1992：8-13.

[13] Kanamori H. Shaking Without Quaking[J]. Science，1998，279（5359）：2063-2064.