# 泰坦尼克号数据训练1

当算法逐渐框架化，变成调参的把戏，****数据清洗**** 就成为了所谓的数据挖掘的精髓，是你与别人的模型拉开差距的地方，也是你建模功力的最佳展示。  
如何洗数据？当然不是立白洗衣液，雕牌洗衣皂之类的。

数据清洗是一个漫长、耗时、经验积累的过程，本文包括：

* 导入数据
* 理解特征、数据类型
* 缺失值处理
* 数据类型的统一性处理
* 分类变量的处理(unary | binary | nomial | categorical | ordinal)
* 连续变量的处理(interval)

排序不分先后，有些步骤会重复出现  
比如观察缺失值，缺失值填充等

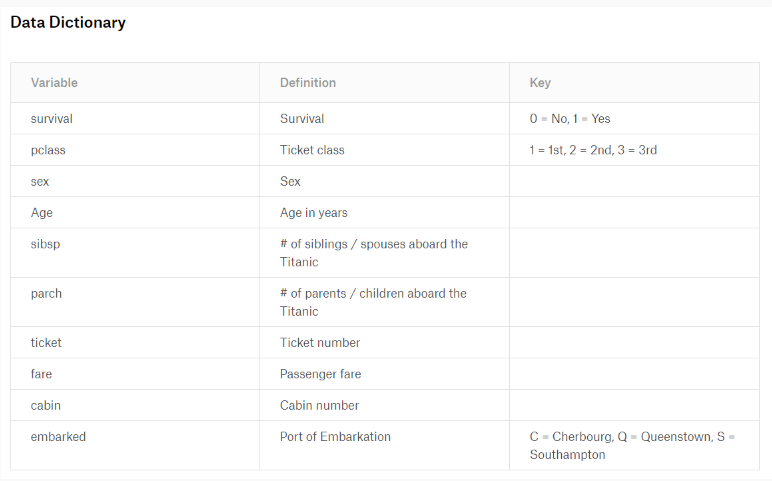
#### **1. 导入数据**

[train.csv](https://github.com/YZHANG1270/Markdown_pic/blob/master/DataCleaning/train.csv" \t "http://codewithzhangyi.com/2018/05/25/%E5%9F%BA%E4%BA%8ER%E7%9A%84%E6%95%B0%E6%8D%AE%E6%B8%85%E6%B4%97%EF%BC%881%EF%BC%89/_blank) [下载](https://www.kaggle.com/c/titanic/data" \t "http://codewithzhangyi.com/2018/05/25/%E5%9F%BA%E4%BA%8ER%E7%9A%84%E6%95%B0%E6%8D%AE%E6%B8%85%E6%B4%97%EF%BC%881%EF%BC%89/_blank)  
本文的样本数据为kaggle入门赛Titanic的数据，地球人都懂的……吧？

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 | # 导入包 packages<-c("ggplot2","dplyr","varhandle") UsePackages<-function(p){  if (!is.element(p,installed.packages()[,1])){  install.packages(p)}  require(p,character.only = TRUE)} for(p in packages){  UsePackages(p) }  library(ggplot2) library(dplyr) library(varhandle) |

以下的数据清洗方法、思路包含却不仅限于这个数据样本，按照整体分析框架会拓展到其他常用的特征

|  |  |
| --- | --- |
| 1 2 3 4 | # 我将下载的数据放在"D:/1\_kaggle/titanic/data"的路径下  setwd("D:/1\_kaggle/titanic") # 设置路径 train = read.csv('data/train.csv') # 导入csv格式的数据 |



#### **2. 理解特征、数据类型**

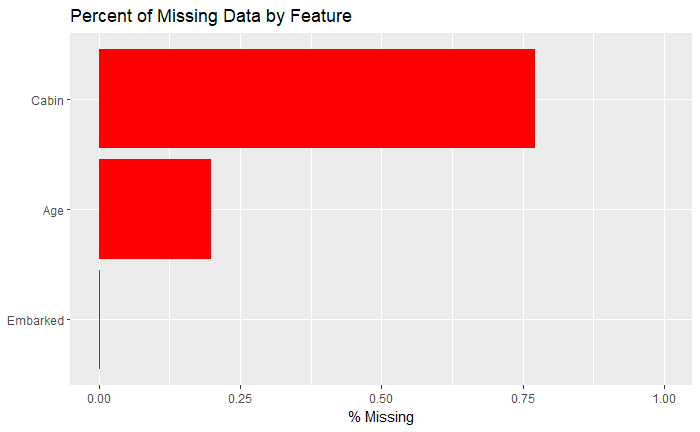
|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 | ncol(train) # 字段数量 列数 # 12  nrow(train) # 样本数量 行数 # 891  colnames(train) # 字段名 列名 # "PassengerId" "Survived" "Pclass" "Name" # "Sex" "Age" "SibSp" "Parch"  # "Ticket" "Fare" "Cabin" "Embarked"   cbind(apply(train,2,function(x)length(unique(x))),sapply(train,class)) # 获取每列的数据种类数，和 数据类型 # PassengerId "891" "integer" # Survived "2" "integer" # Pclass "3" "integer" # Name "891" "factor"  # Sex "2" "factor"  # Age "89" "numeric" # SibSp "7" "integer" # Parch "7" "integer" # Ticket "681" "factor"  # Fare "248" "numeric" # Cabin "148" "factor"  # Embarked "4" "factor" |

* 训练集中乘客的特征有：PassengerId、Pclass、Name、Sex、Age、SibSp、Parch、Ticket、Fare、Cabin、Embarked
* 12个字段/特征，12列，891行
* integer | numeric 为连续变量/数值型特征
* factor 为分类变量/离散型的特征，一般需要额外处理才能训练

#### **3. 缺失值处理**

（1）观察缺失值比例

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 | # 建立观察缺失值比例的函数 na.plot na.plot=function(data){  missing1=sapply(data,function(x)sum(x == '')/nrow(data))  missing2=sapply(data,function(x)sum(sum(is.null(x)), sum(is.na(x)))/nrow(data))  if(sum(is.na(missing1))>0){  missing1[is.na(missing1)] = 0  }  missing = missing1 + missing2  print(missing)   missing=missing[order(missing,decreasing = T)]  nadata=missing[missing>0]  na\_df=data.frame(var=names(nadata),na=nadata,row.names = NULL)  ggplot(na\_df)+  geom\_bar(aes(x=reorder(var,na),y=na),stat='identity', fill='red')+  labs(y='% Missing',x=NULL,title='Percent of Missing Data by Feature') +  coord\_flip(ylim = c(0,1))  }  # 调用函数 na.plot(train) |

[](https://github.com/YZHANG1270/Markdown_pic/blob/master/DataCleaning/003.png?raw=true)

* 字段 Cabin/ Age/ Embarked 有缺失值
* 缺失值缺少越好，没有缺失值最好

缺失值存在于DataFrame的形式：

* NA
* NULL
* ‘’

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 | # 以Cabin为例  train$Cabin[is.na(train$Cabin)] # 筛选出Cabin为NA的缺失值Cabin # train$Cabin[!is.na(train$Cabin)] # 反逻辑  train$Cabin[is.null(train$Cabin)] # 筛选出Cabin为NULL的缺失值Cabin  # train$Cabin[!is.null(train$Cabin)]# 反逻辑  train$Cabin[train$Cabin == ''] # 筛选出Cabin为''的缺失值Cabin # train$Cabin[train$Cabin != ''] # 反逻辑  filter(train,!is.na(Cabin)&!is.null(Cabin)&(Cabin != ''))$Cabin # 筛选出非缺失值的Cabin值 |

这样做数据清洗不免有些繁琐，如果逻辑合理，可以把NULL值和’’值都处理成NA值

|  |  |
| --- | --- |
| 1 2 3 4 5 6 | # 三种缺失值的区别在于length长度 length(NA) == 1  length(NULL) == 0  length('') == 1 |

（2）当高缺失值占比出现时（例：missing proportion > 95%），一般考虑删除该特征

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 | # 建立删除缺失值占比高于某比例的特征的函数 # 下面的函数设置的阈值是 0.75  na.drop=function(data){  missing1=sapply(data,function(x)sum(x == '')/nrow(data))  missing2=sapply(data,function(x)sum(sum(is.null(x)), sum(is.na(x)))/nrow(data))  if(sum(is.na(missing1))>0){  missing1[is.na(missing1)] = 0  }  missing = missing1 + missing2  missing=cbind(colnames(data),as.numeric(missing)[!is.na(as.numeric(missing))])%>%as.matrix()  print(missing)  valid=missing%>%as.data.frame%>%filter(missing[,2]<0.75) # 缺失值占比阈值设置   valid$V1<-unfactor(valid$V1)  data=data%>%select(valid$V1)%>%as.data.frame() }  train=na.drop(train) # 实施删除高缺失值特征 na.plot(train) # 观察是否被删除 |

* Cabin缺失77%，将被删除

（3）重要特征 & 高缺失值占比：将该特征转换成binary特征，有数值为1，缺失值为0

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 | train = read.csv('data/train.csv') #恢复原df  # 建立函数：筛选出高缺失值占比的特征 na.toBinary=function(data){  missing1=sapply(data,function(x)sum(x == '')/nrow(data))  missing2=sapply(data,function(x)sum(sum(is.null(x)), sum(is.na(x)))/nrow(data))  if(sum(is.na(missing1))>0){  missing1[is.na(missing1)] = 0  }  missing = missing1 + missing2  missing=cbind(colnames(data),as.numeric(missing)[!is.na(as.numeric(missing))])%>%as.matrix()  print(missing)    toBinary=missing%>%as.data.frame()%>%filter(missing[,2]>0.75 & missing[,2]<1) # 缺失值占比阈值设置,这里是选取缺失值比例大于0.75，小于1的字段，仅仅作为筛选出Cabin字段   toBinary$V1<-unfactor(toBinary$V1)  print(toBinary$V1) } na.toBinary(train) # "Cabin"  # 假设Cabin为重要信息，对缺失值进行二值化 train$Cabin = as.numeric(!is.na(train$Cabin)&!is.null(train$Cabin)&(train$Cabin != '')) |

* 这里只是拿Cabin做个例子，实际上并不会把这个信息二值化
* 当出现重要特征 & 高缺失值占比时，才进行这个处理

（4）缺失值填充

缺失值填充有很多方法：

* 数值型特征：可以用 均值/最大值/最小值/众数 填充
* 时间序列特征：例如苹果今天的价格缺失，可以用昨日的价格填充
* 将缺失值归为一类：可以用不曾出现也不会出现的值，比如：缺失年龄用 999 填充，缺失体重用 -1 填充
* 缺失比例小的连续变量：可用有数值的数据进行对缺失值的回归预测填充，例：班上某同学的身高
* 重要特征 & 高缺失值占比：将缺失值二值化，在上面已经举例说明
* 其他：可以用任何合理的逻辑进行填充

（数值型特征）均值/最大值/最小值/众数的获取

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 | # 以 Age 为例  tmp\_Age = filter(train,!is.na(Age)&!is.null(Age)&(Age != ''))$Age # 筛选出非空年龄值  mean(tmp\_Age) # 29.69912 max(tmp\_Age) # 80 min(tmp\_Age) # 0.42  # 众数没有直接可以调用的function，需要自己写 # create getmode function getmode<-function(v){  uniqv<-unique(v)  uniqv[which.max(tabulate(match(v,uniqv)))] }  getmode(tmp\_Age) # 24   # 将Age的缺失值以众数填充 train$Age[is.na(train$Age)] = getmode(tmp\_Age) |

#### **4. 数据类型的统一性处理**

数据类型有两种属性：

* 在数据里展示的类型
* 在实际意义中的类型

当两种类型不统一时，需要将其统一，或者做相应标识

例：

* Pclass，舱位等级这个特征在数据集中以 1/2/3 的整数展示，显示类型为”integer”，为连续变量，但按字面解释这个特征实际上属于categorical分类变量，等级1/等级2/等级3，意义上类型应该是”factor”，两种属性不统一。
* SibSp特征描述乘客的兄弟姐妹或者配偶数量，显示类型为”integer”，字面类型也是数值型，连续变量，两种类型统一。

数据类型应该统一如下：

* 连续变量：Age、SibSp、Parch、Fare（已经全部为数值型变量）
* 分类变量：PassengerId、Pclass、Name、Sex、Ticket、Cabin、Embarked（有数值型有离散型）

#### **5. 分类变量的处理**

****分类变量会出现多种情况：****  
（1）情况：分类信息需要提取  
在Name中，有Mr/Mrs/Miss等称呼信息可以判别乘客的性别，虽然在这里Sex性别的信息未缺失，但是在结果出来前所有的信息都不能轻易丢弃  
而且，按照Miss/Mrs可以大概预测缺失的年龄，对于名字中有Mrs的乘客的缺失年龄填充就不会出现10岁之类的

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 | train = read.csv('data/train.csv')  which(grepl('Mrs.', train$Name)) # 筛选出Mrs乘客的Index train$Name[which(grepl('Mrs.', train$Name))] # 筛选出Mrs乘客的名字，这部分人的性别可以标识为 女性，在这里暂不做改动 which(grepl('Mrs.', train$Name[is.na(train$Age)])) # 筛选出年龄为缺失值的Mrs乘客的Index  # 筛选出年龄为缺失值的Mrs乘客的年龄，下一步进行赋值 train$Age[is.na(train$Age)][which(grepl('Mrs.', train$Name[is.na(train$Age)]))]   # 用无缺失值的Mrs群体的年龄的均值 来填充 Mrs群体的缺失年龄 # 这样比直接用全体样本的年龄均值来填充缺失的样本年龄要合理一些 train$Age[is.na(train$Age)][which(grepl('Mrs.', train$Name[is.na(train$Age)]))] = mean(train$Age[!is.na(train$Age)][which(grepl('Mrs.', train$Name[!is.na(train$Age)]))])   # 同理可以对 Miss群体和 Mr群体进行年龄填充 train$Age[is.na(train$Age)][which(grepl('Miss.', train$Name[is.na(train$Age)]))] = mean(train$Age[!is.na(train$Age)][which(grepl('Miss.', train$Name[!is.na(train$Age)]))]) train$Age[is.na(train$Age)][which(grepl('Mr.', train$Name[is.na(train$Age)]))] = mean(train$Age[!is.na(train$Age)]) train$Age[is.na(train$Age)] = mean(train$Age[!is.na(train$Age)])  # 将年龄整数化 train$Age = as.integer(train$Age) |

（2）情况：分类的类数level太多

ID类型：

* PassengerId：一个样本为一类，每个样本的ID不一样，无效信息直接删除
* 身份证ID：与单纯的Index字段不一样，不能直接删除，能从里面提取出生年月日，年龄，性别，户籍省份等，信息提取完毕后可以删除
* userID：有些用户ID里也包含用户的注册信息，比如注册年月日等，需要对数据敏感

多类别少样本类型：

* 合并类别：  
  双十一购物者的省份信息，在江浙沪有爆炸性的收件数量，而在西藏、新疆近10个区域等地的收件数量很少，可以把这10个地区合并为一类：偏远地区

（3）情况：分类的类数level太少

* 单类别：所有样本都属于一个类别，无效类别信息，删除特征

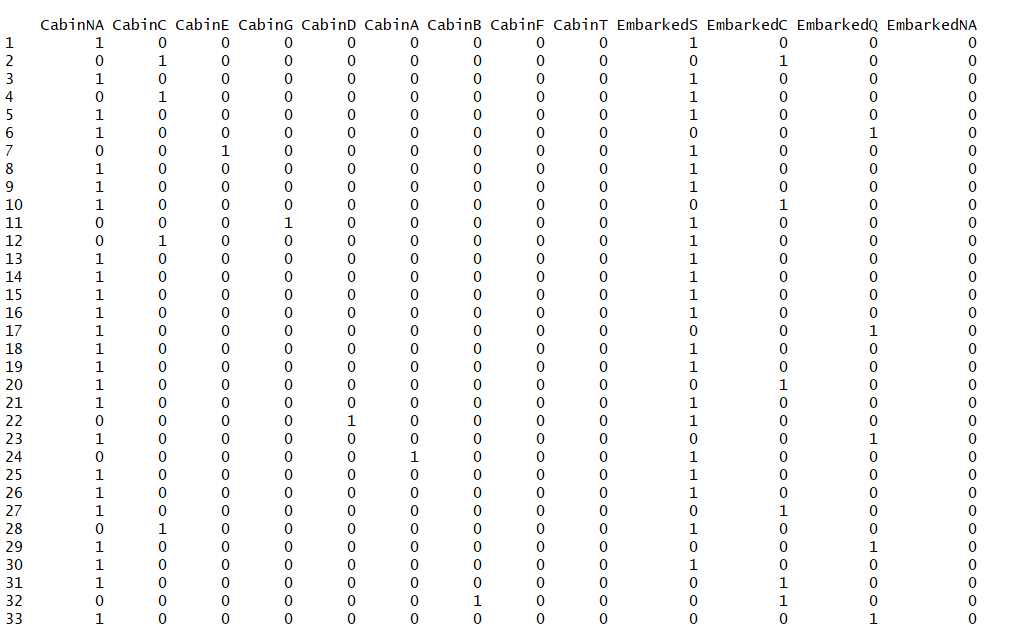
（4）情况：连续变量离散化/分箱

* 比如Age：可以把连续的年龄数值划分为：少年/青年/中年/老年，在决策树中这种操作会比较常见

****处理方法：****  
（1）onehot 独热编码处理  
one-hot encoding就是用h个变量来代表这h个level，比如3个level的变量就表示成100，010，001

这里以Cabin、Embarked为例，进行onehot处理

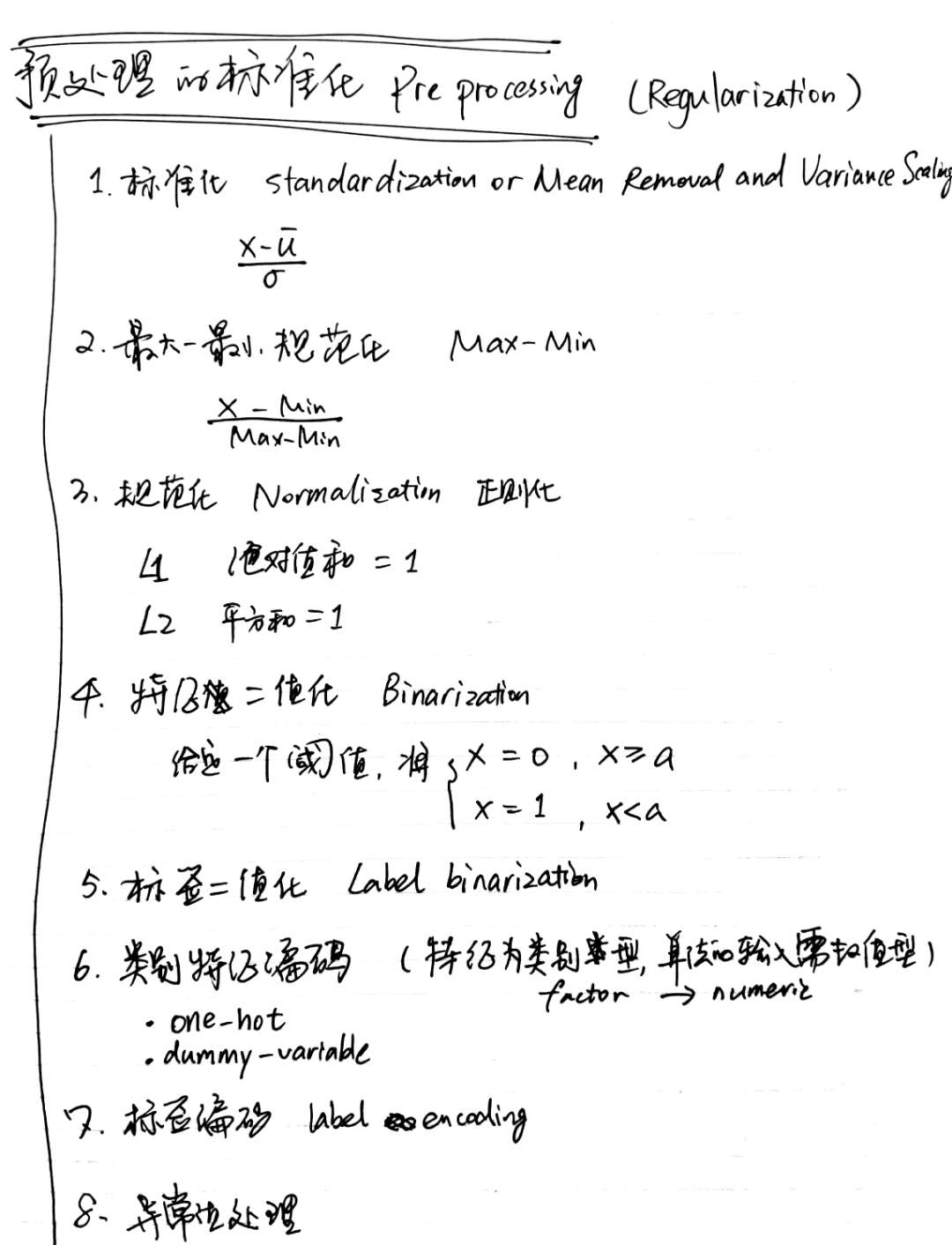
|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 | train = read.csv('data/train.csv')  # Cabin本身是由字母与数字组成的 # 字母更能代表舱位的一定信息，因此对Cabin做如下处理 # 对于有值的Cabin：仅保留字母信息 # 对于缺失值的Cabin：用 NA 作为一类填充  train$Cabin = ifelse(train$Cabin != "",substr(gsub('[0-9]', '', train$Cabin), 1, 1), 'NA') unique(train$Cabin) # 处理后Cabin的数值种类 # NA C E G D A B F T    # 保留Embark的字母信息，缺失值用NA填充 train$Embarked = ifelse(train$Embarked != "", substr(train$Embarked, 1, 1), 'NA') unique(train$Embarked) # 处理后Embark的数值种类 # S C Q NA   # 只有class类型为factor的特征才能做model.matrix处理，需要提前把character类型转换为factor类型 features = c('Cabin','Embarked') #需要做转换的特征名称 for (f in features){  if( (class(train[[f]]) == "character") || (class(train[[f]]) == "factor"))  {  levels = unique(train[[f]])  train[[f]] = factor(train[[f]], level = levels)  } }  # onehot处理 trainMatrix <- model.matrix(~ Cabin + Embarked, data=train,   contrasts.arg = lapply(train[,c('Cabin','Embarked')], contrasts, contrasts=FALSE))  trainMatrix <- trainMatrix[,-1] |

[](https://github.com/YZHANG1270/Markdown_pic/blob/master/DataCleaning/005.png?raw=true)

* Cabin变成length(unique(train$Cabin)) = 9 个onehot特征
* Embark变成length(unique(train$Embarked)) = 4 个onehot特征

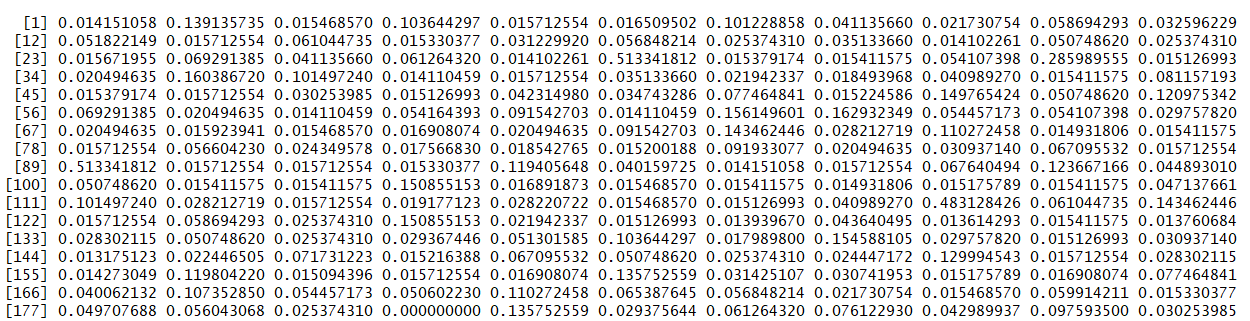
（2）dummy 虚拟变量编码处理  
dummy encoding就是把一个有h个level的变量变成h-1个变量，比如3个level的变量就表示成成10，01，或00

#### **6. 连续变量的处理**

当分类变量都变成 0/1 的数字，而连续变量（比如年龄，是0~100的分布）的值域远远大于0~1，难免显得不公平。这个说法是有科学依据的，有兴趣可以手动推导一下。  
因此一般会对连续变量进行 归一化(Regularization) | 标准化(Standardization) | 去中心化 | …(基本上是同一类思想，暂且都称为标准化)标准化有以下几种处理方式：  
  
[](https://github.com/YZHANG1270/Markdown_pic/blob/master/DataCleaning/006.jpg?raw=true)

这里以Fare为例，进行第二种：最大-最小化 | max-min 处理

|  |  |
| --- | --- |
| 1 | train$Fare = (train$Fare-min(train$Fare))/(max(train$Fare)-min(train$Fare)) |

[](https://github.com/YZHANG1270/Markdown_pic/blob/master/DataCleaning/007.png?raw=true)  
这样票价Fare的值域就处于0~1之间了

From: http://codewithzhangyi.com/2018/05/25/%E5%9F%BA%E4%BA%8ER%E7%9A%84%E6%95%B0%E6%8D%AE%E6%B8%85%E6%B4%97%EF%BC%881%EF%BC%89/