内容简介

数据检视主要是通过对数据质量以及字段情况进行了解，同时对数据进行一些简单的数据清理工作，最后根据主业务问题和建模的需要提出数据改造方案。

1. 主要内容

- 数据质量和字段情况；
- 数据清理；
- 提出数据改造方案。

2. 学习目标

学完本节，能解决以下问题：

- 有限的离散型数据中如果存在错误的类别，如何对其进行异常值处理？
- 如何对长文本的字段，通过查看数据长度较短的方式快速筛选异常数据？

提出数据改造方案

在完成了数据质量和字段情况检视，以及数据清理等工作后，我们需要根据建模的需要对每一个字段进行必要的改造。值得一提的是，为了减少计算成本，本次案例不再关注数据分割和重构对模型的影响。

1. 岗位薪资字段改造

现有的岗位薪资字段是包含数字的文本，描述岗位薪资的最低值和最高值，如下图所示：

```
array(['1K-2K', '20K-40K', '30K-50K', '25K-50K', '18K-30K', '25K-45K',
       '35K-60K', '20K-35K', '2K-3K', '15K-25K', '20K-21K', '35K-70K',
       '15K-30K', '30K-60K', '25K-35K', '5K-6K', '3K-5K', '6K-10K',
       '35K-50K' '10K-20K' '20K-30K' '2K-4K' '13K-14K' '40K-80K'
```

一般情况下，岗位薪资应该是一个具体的数值，即定比变量，所以需要根据 job_salary 中的文本，进行数据转换，得到新的岗位薪资字段，类似于下图所示：

```
array([ 1.5, 30. , 40. , ..., 10. , 15. , 20. ])
```

2. 公司概述和岗位概述字段改造

我们很难用计算机直接处理长文本，所以对公司概述 company_overview 和岗位概述 job_info 两个文本进行文本向量化处理，生成新的公司概述和岗位概述字段。

但是同时也会带来一个问题，正如我们知道的，文本向量化后的字段会变成成千上万个维度。上一个案例中关于先验概率不同的情况下，朴素贝叶斯模型的准确率一致，解释为上千个维度之间的差异远远弱化了先验概率一个数值对模型的影响，而本次案例除了这两个字段外，还有 company_financing_stage、company_people、job_edu_require、job_exp_require 这四个字段都是一维的。因此文本向量化后，还需要对公司概述和岗位概述降维。

关于这两个字段的降维方法在后面会继续介绍到，这里需要知道的就是关于该两字段的降维不仅仅是简单的降维，而是在降维的同时尽量不要丢失该两字段与岗位薪资之间关联。

3. 文本数据转换为数值

根据之前得到的字段情况和数据清理结果，可以发现字段 company_financing_stage、company_people、job_edu_require、job_exp_require 都是离散型数据，以及公司概述和岗位概述也有可能转换为离散型变量，都属于定类或定序变量，对于离散型数据我们可以通过 OneHotEncoder 方法进行标准化处理。

数据检视与清理总结

1、有限的离散型数据中如果存在错误的类别，如何对其进行异常值处理？

可以使用 pandas 库中的 query()方法筛选出正确类别，例如案例中的公司融资情况字段的唯一值：

```
# 数据异常分析：'company_financing_stage'
job_desc['company_financing_stage'].unique()
```

```
array(['已上市', 'D轮及以上', 'C轮', 'B轮', '不需要融资', 'A轮', '天使轮', '20-99人',
       '1000-9999人', '未融资', 'company_financing_stage', '10000人以上',
       '0-20人', '500-999人', '100-499人'], dtype=object)
```

其中除了红框内的字段，剩余类别均为错误类别。因此可以将正确类别建立成一个列表，随后在 query()中通过@引入列表变量，对正确类别进行筛选。代码如下：

```
# 删除异常数据
fnc = ['已上市', 'D轮及以上', 'C轮', 'B轮', '不需要融资', 'A轮', '天使轮','未融资']
job_desc.query('company_financing_stage in @fnc', inplace=True)
job_desc.shape
```

```
(3315, 7)
```

经过处理后，本案例的数据集剩余 3315 个样本。

2、如何对长文本的字段，通过查看数据长度的方式快速筛选异常数据？

可以通过 pandas 库中 Series 类的 str.len()方法 ,它可以直接返回 Series 中每一行字符串的长度。例如案例中公司概述字段是一个长文本，可以对数据长度进行查看，筛选出异常的样本，例如案例中的代码：

```
# 数据异常分析：'company_overview'
job_desc[job_desc['company_overview'].str.len() < 20]
```

2312	天使轮	专注于医疗大数据平台搭建与数据分析	20-99人	学历：本科	经验：1年以内	完成医院数据分析报告，要求熟练应用office工具，良好的文案基础，掌握Tableau等数据...	7K-8K
2504	未融资	纪元时空是国内领先的动漫游公司。	100-499人	学历：本科	经验：1-3年	1. 主导整个项目数据收集和分析流程，精准提炼数据和分析，形成结论，并跟踪反馈整个过程；2....	15K-20K
2540	未融资	专注于网络安全，信息安全方向	20-99人	学历：大专	经验：1-3年	职位描述：1. 在分布式存储、计算框架上，结合硬件环境，深度优化机器学习算法，提升机器学习性...	8K-16K
2964	已上市	公司介绍	1000-9999人	学历：本科	经验：5-10年	1、3年及以上工作经验。2、熟悉RWA、熟悉新资本协议/BASEL II&III/资产负债/...	12K-20K
2996	未融资	临床医学研究一站式管理平台	20-99人	学历：本科	经验：1-3年	您的职责1.负责海量医疗数据的清洗、统计、挖掘等算法研发。2.利用机器学习模型在医学研究等方...	15K-30K
3013	A轮	注册资本500万美金	20-99人	学历：本科	经验：1-3年	工作职责：1、对接公司核心部门需求，深入理解产品、业务需求，建立业务运营监测	16K-32K

这里我们将字符串长度设置为 20，即小于 20 的数据都会被筛选出来。可以看到索引为 2964 的样本公司概述为“公司介绍”，可以视为异常值进行删除：

```
# 删除公司概述异常的记录
job_desc.drop(index=2964, inplace=True)
job_desc.shape
```

```
(3314, 7)
```

删除后，数据集剩余 3314 行。

内容简介

基于数据改造方案，本节会从建模的角度，将本案例涉及的 7 个变量进行数据转换以及数据分割。主要内容如下：

1. 主要内容

- 改造岗位薪资字段；
- 将公司概述和岗位概述两个长文本字段进行文本向量化，并将其从高维映射到与岗位薪资相关联的一维数据；
- 文本数据转换为数值；
- 数据分割。

2. 学习目标

学完本节，能解决以下问题：

- 如何将岗位薪资“**k-**k”文字型变量转换为具体数值的定比变量？
- 在保证字段与岗位薪资相关联的前提下，如何将文本向量化后的高维字段改造为一维数据？

公司和岗位概述字段改造分析

由于公司概述和岗位概述两字段都是长文本型数据，在改造方式上基本一致，因此就公司概述字段进行分析。

1. 构建降维模型

根据上一个案例，公司概述字段在经过 TF-IDF 处理后被转换成了上千维的词语特征，无法和其他字段结合作为自变量输入到回归模型中。因此首先考虑将高维特征转换为一维特征，转换的结果要尽可能不丢失公司概述字段与岗位薪资相关的信息。

为了达到这样的目的，可以考虑构建这样一个模型：

输入是公司概述的高维特征，输出是一维特征。

表示基于输入高维特征的公司概述对岗位薪资的预测。这样通过结合公司概述与岗位薪资，可以尽可能反应概述和岗位薪资相关性高的信息。

2. 转换后变量类型

其次需要转换后得到的变量应该是什么类型。本质上，公司概述是对公司基本情况的描述，不涉及定量方面的信息（加减乘除运算不具有实际意义），所以为了使转换后得到的一维特征更好地反应公司概述字段的基本性质，我们设定转换后得到的新变量是定类/定序变量，即是离散的。

至此，我们要构建的转换模型，其输入是公司概述的 TF-IDF 高维向量，输出是一个离散型变量，反应了公司概述特征中能够用于预测岗位薪资的信息。

这个模型本质上是一个分类模型，构建的方式和上一个案例几乎相同，为了训练这个模型，需要对公司概述字段做 TF-IDF 转换作为输入，对岗位薪资进行离散化处理作为输出。

3. 转换模型选择

简单起见，我们的转换模型使用朴素贝叶斯算法。

首先，对岗位薪资进行离散化处理，从低到高切分为数量相等的几类。然后，构建模型根据公司概述字段预测岗位薪资，从而将公司概述字段转换成定类变量。

为了确定哪种转换结果的效果最好，我们尝试转换得到 3 种新的公司概述字段，取值情况分别为 4、5、6 种取值，因此对应的在对岗位薪资进行离散化处理时，也分别将其离散化为 4、5、6 类。

岗位概述字段与公司概述字段采用相同的方法，因此可以定义一个函数，分别来实现岗位概述字段与公司概述字段的转换。

4. 转换模型评估

究竟离散化为几类是模型评估效果最好？可以构建一个线性回归模型，每种取值情况下，选择转换后的公司概述和岗位概述作为输入，岗位薪资的定比变量作为输出，计算拟合优度系数 R2 来计算回归方程的拟合程度，从而选择最好转换结果。

数据转换与分割总结

1、如何将岗位薪资“**k-**k”文字型变量转换为具体数值的定比变量？

字符串的切分与转换是数据准备中很普遍的内容，案例中的方法只是字符串切分的一种简单方法。主要思路是利用薪资的上下的数字求平均。例如案例中的代码：

```
# 将job_salary字段转换成数值型变量
job_desc['job_salary_avg'] = job_desc['job_salary'].apply(lambda x: np.mean(
    list(map(float,x.replace('K', '').split('-')))))
job_desc
```

首先使用 replace()函数将薪资字符串中的‘k’字符用空字符替换掉；

随后通过 split()函数将两个数字切分开；

由于切分后的两个元素仍为文本型无法进行计算，可以通过 map( ) 函数将两个文本元素都转换为浮点型；

在 Python3 中，使用 map()函数之后不能直接返回转换后的列表，而是返回迭代器，不能使用 np.mean()方法，因此需要 list()方法进行格式转换，这样对于“**k-**k”格式的文字型变量就转换为具体数值的定比变量。

最后通过 apply()函数将该方法应用到需要转换的字段，并将转换后的结果保存到新字段‘job_salary_avg’。

2、在保证字段与岗位薪资相关联的前提下，如何将文本向量化后的高维字段改造为一维数据？

经过一系列的分析，决定使用朴素贝叶斯模型将文本向量化后的高维字段映射在离散化后的岗位薪资字段上，并通过线性回归模型来挑选表现最好的岗位薪资离散化的种类个数。因此主要涉及三部分内容：

（1）离散化岗位薪资字段

使用 pandas 的 qcut()函数进行离散化，qcut 依据的是样本分位数进行离散化，

如：

```
pd.qcut(range(6), 3, labels=False)
array([0, 0, 1, 1, 2, 2], dtype=int64)
```

注：同理 pandas 的 cut()函数是依据样本实际取值等距进行离散化，如：

```
pd.cut([0,2,4,8,9,10], 3, labels=False)
array([0, 0, 1, 2, 2, 2], dtype=int64)
```

1-10 距离为 10，分为三份，以$\frac{10}{3}$和$\frac{20}{3}$为界限，≤$\frac{10}{3}$标记为 0，($\frac{10}{3}$，$\frac{20}{3}$]标记为 1，($\frac{20}{3}$，10]标记为 2。

而案例中的代码为：

gnb_y = *pd.qcut(job_desc['job_salary_avg'], k, labels=False)*

（2）使用公司概述和岗位概述字段分别拟合朴素贝叶斯模型，并使用该模型进行预测，这样，得到预测结果就是与岗位薪资相关的离散型数据。

（3）随后将使用映射后得到两个一维数据作为线性回归模型的输入变量，连续型岗位薪资作为模型得出变量，评估岗位薪资离散化为几类时效果最佳。

```
# 高维特征转换为一维特征
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import Lasso

for k in range(4,7):
    '''
    构建朴素贝叶斯模型，将高维特征转换为一维特征
    '''
    # 岗位薪资离散化
    gnb_y = pd.qcut(job_desc['job_salary_avg'], k, labels=False)
    # 构建朴素贝叶斯模型
    gnb = GaussianNB()
    # 公司概述转换
    gnb.fit(company_overview, gnb_y)
    company_level = 'company_' + str(k)
    job_desc[company_level] = gnb.predict(company_overview)
    # 岗位概述转换
    gnb.fit(job_info, gnb_y)
    job_level = 'job_' + str(k)
    job_desc[job_level] = gnb.predict(job_info)
    '''
    将转换后的公司概述、岗位概述作为输入，岗位薪资作为输出，评估模型效果
    '''
    # 获取输入输出数据
    X = job_desc[[company_level, job_level]]
    y = job_desc['job_salary_avg']
    # 创建并训练模型
    lasso = Lasso(random_state=0)
    lasso.fit(X, y)
    print('取值 %d 种的R平方为 %.4f' % (k, lasso.score(X, y)))
```

```
取值 4 种的R平方为 0.5950
取值 5 种的R平方为 0.6234
取值 6 种的R平方为 0.6180
```

最终决定取值为 5 种时效果最佳，因为岗位薪资被离散化为 5 类，文本向量化后的公司概述和岗位概述字段的取值也是 5 类。