## 实战案例：数据挖掘项目：构建人才(用户)流失模型

### \* 项目：构建人才(用户)流失模型对数据进行分析及预测

## 1.项目描述：

企业的快速发展离不开人才的支撑，可是现在我国的企业的人才流失严重，人才流失问题现在已经成为了关系企业发展的一个重大的问题。这些企业要想在目前激烈的竞争中快速发展，就需要依靠自身的人力资源的来竞争。只有拥有比对方更强，更优秀，更具有创造力的人才，才能在竞争中取得优势。所以如何有效解决我国企业人才流失问题是一个很迫切的任务。人才流失已经成了很多企业正在面临的困境，关键人才的流程对企业的影响尤为明显。 无论在IT互联网领域还是传统领域、事业单位，均面临关键人才的流失，作为公司的核心的人力资源部门，我们需要把控员工的基本情况，对员工的情况进行实时监控和预测，人才流失模型从公司的角度和员工自身角度分别入手，阐释了在那些重要维度能够保持流失率的下降，常规的做法比如增强企业文化，提高薪资，提高年终奖等，通过模型给出人力资源部门一定的建议。

1. **技术说明：**

项目通过公司内部人资资源系统数据，通过数据的基本ETL(数据清洗过滤和汇总)对数据进行基本的预处理，通过python的numpy、pandas、matplotlib和seaborn进行各维度数据分析，经过数据分析得到分类特征较好的特征数据，对数值型数据、类别型数据、有序性数据分别进行处理和分析，使用label encoder和one encoder分别对类别数据进行特征编码，处理组合后的数据特征后形成特征向量，通过Python的Scikit-learn机器学习库的机器学习算法寻找数据之间存在的关系，从而为公司人力资源及决策层提供信息建议及决策建议。

## **3.需求分析：**

3.1 分析各个维度的数据对人才流失的影响

3.2 通过训练数据建立的模型以及所给的测试数据，构建人才流失模型，最终预测测试数据相应的员工是否已经离职（0未离职，1离职）。

## 3.数据集描述

### 3.1 数据集描述:

\* 数据竞赛中[提供的数据集]

\* 数据主要包括**影响员工离职的各种因素**（工资、出差、工作环境满意度、工作投入度、是否加班、是否升职、工资提升比例等）以及员工是否已经离职的对应记录。数据分为训练数据和测试数据，分别保存在train.csv和test.csv两个文件中。训练数据主要包括1100条记录，31个字段。

### 3.2构建数据字典

\* Attrition：员工是否已经离职，1表示已经离职，0表示未离职，这是目标预测值；

\* Age：员工年龄

\* BusinessTravel：商务差旅频率，Non-Travel表示不出差，Travel\_Rarely表示不经常出差，Travel\_Frequently表示经常出差；

\* Department：员工所在部门，Sales表示销售部，Research & Development表示研发部，Human Resources表示人力资源部；

\* DistanceFromHome：公司跟家庭住址的距离，从1到29，1表示最近，29表示最远；

\* Education：员工的教育程度，从1到5，5表示教育程度最高；

\* EducationField：员工所学习的专业领域，Life Sciences表示生命科学，Medical表示医疗，Marketing表示市场营销，Technical Degree表示技术学位，Human Resources表示人力资源，Other表示其他；

\* EmployeeNumber：员工号码；

\* EnvironmentSatisfaction：员工对于工作环境的满意程度，从1到4，1的满意程度最低，4的满意程度最高；

\* Gender：员工性别，Male表示男性，Female表示女性；

\* JobInvolvement：员工工作投入度，从1到4，1为投入度最低，4为投入度最高；

\* JobLevel：职业级别，从1到5，1为最低级别，5为最高级别；

\* JobRole：工作角色：Sales Executive是销售主管，Research Scientist是科学研究员，Laboratory Technician实验室技术员，Manufacturing Director是制造总监，Healthcare Representative是医疗代表，Manager是经理，Sales Representative是销售代表，Research Director是研究总监，Human Resources是人力资源；

\* JobSatisfaction：工作满意度，从1到4，1代表满意程度最低，4代表满意程度最高；

\* MaritalStatus：员工婚姻状况，Single代表单身，Married代表已婚，Divorced代表离婚；

\* MonthlyIncome：员工月收入，范围在1009到19999之间；

\* NumCompaniesWorked：员工曾经工作过的公司数；

\* Over18：年龄是否超过18岁；

\* OverTime：是否加班，Yes表示加班，No表示不加班；

\* PercentSalaryHike：工资提高的百分比；

\* PerformanceRating：绩效评估；

\* RelationshipSatisfaction：关系满意度，从1到4，1表示满意度最低，4表示满意度最高；

\* StandardHours：标准工时；

\* StockOptionLevel：股票期权水平；

\* TotalWorkingYears：总工龄；

\* TrainingTimesLastYear：上一年的培训时长，从0到6，0表示没有培训，6表示培训时间最长；

\* WorkLifeBalance：工作与生活平衡程度，从1到4，1表示平衡程度最低，4表示平衡程度最高；

\* YearsAtCompany：在目前公司工作年数；

\* YearsInCurrentRole：在目前工作职责的工作年数

\* YearsSinceLastPromotion：距离上次升职时长

\* YearsWithCurrManager：跟目前的管理者共事年数；

## 4.项目结构

### 4.1项目流程：

* 数据获取(来源于公司内部的人力资源数据，通常没有缺失值的)
* 数据探索性分析---哪些是有缺失值，特殊标记
* 数据预处理---数据分析(数据图表展示给出相应的分析结果---哪些特征是有用的？)
* 特征处理：主要是对类别型数据进行处理
* 数据集的划分：使用20%部分作为测试集，80%作为训练集 （使用的是10则交叉验证得到结果）
* 模型训练：逻辑斯特回归、GBDT
* 模型校验：模型准确率、召回率、精确率、F1值、ROC曲线(横轴：真正率TRP，纵轴：假正率FPR)-----通过曲线和x轴围城的面积衡量分类性能的好坏，曲线面积叫做AUC值---面积大小代表准确率大小---Roc-Auc曲线
* **使用imblearn框架进一步采样过采样或欠采样或结合方式或集成采样方式对**
* 类别不均衡的问题进一步处理

### 4.2环境准备

Python环境Anaconda+安装不平衡数据处理第三方库Imbalance

## 代码实现

### 5.0数据的探索和思考正负样本比例

使用pandas基础属性查看数据的一些属性信息，如shape,info等

### 5.1数据可视化分析

采用基于matplotlib的Seaborn库

pip install seaborn 安装 如果使用Anaocnda无需安装

<http://seaborn.pydata.org/generated/seaborn.countplot.html#seaborn.countplot>

泰坦尼克号的数据分析

<http://seaborn.pydata.org/generated/seaborn.pairplot.html#seaborn.pairplot>

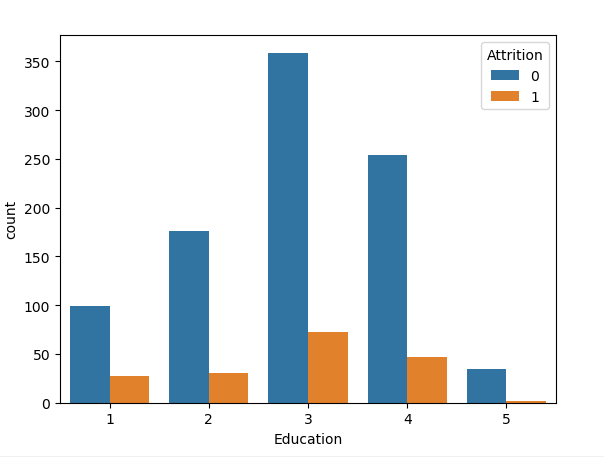
Iris鸢尾花的案例

#### (1)分析离职和受教育程度的分析

# 使用countplot方式进行分析

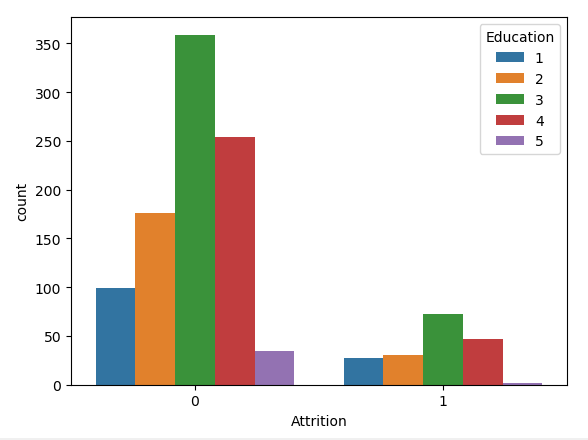
sns.countplot(x="Education", hue="Attrition", data=talent\_data)

plt.show()



sns.countplot(x="Attrition", hue="Education", data=talent\_data)

plt.show()



#### **(2)各字段关系分析**

# 1.离职和年龄的关系

plt.subplot(221)

sns.boxplot(x="Attrition", y="Age", data=talent\_data)

# 2.离职和家庭和距离之间的关系

plt.subplot(222)

sns.boxplot(x="Attrition", y="DistanceFromHome", data=talent\_data)

# 3.离职和月收入的关系

plt.subplot(223)

sns.boxplot(x="Attrition", y="MonthlyIncome", data=talent\_data)

# 4.离职和曾经工作公司的关系

plt.subplot(224)

sns.boxplot(x="Attrition", y="NumCompaniesWorked", data=talent\_data)

plt.show()

#### **(3)各字段关系分析**

# 离职和婚姻状况分析

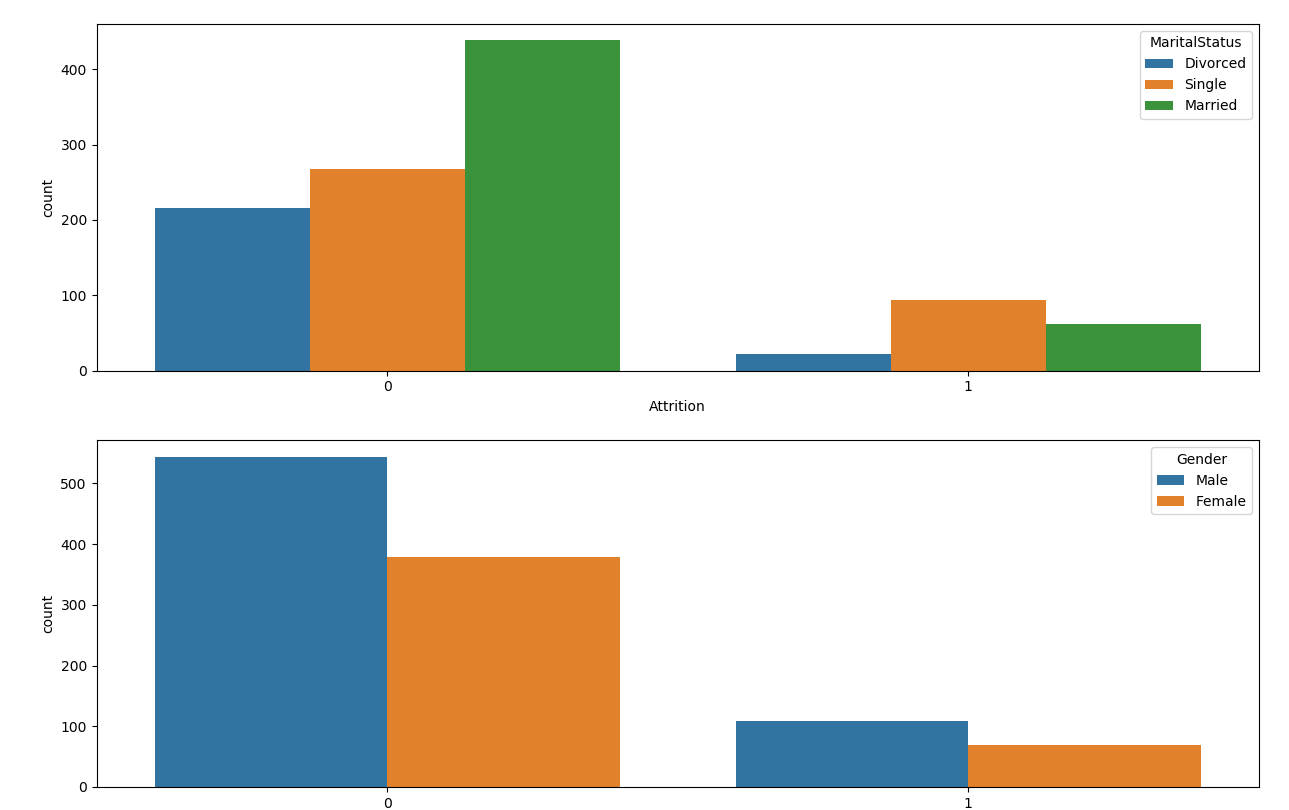
plt.subplot(2, 1, 1)

sns.countplot(x="Attrition", hue="MaritalStatus", data=talent\_data)

# 离职和性别关系

plt.subplot(2, 1, 2)

sns.countplot(x="Attrition", hue="Gender", data=talent\_data)



结论：婚姻状况为“单身”的在离职的人员中数据分布是不同于未离职的数据分布，该特征是区分性较好的特征。

同时，性别中男性较女性离职的概率较高的。

#### **(4)离职和其他变量的相关性分析**

sns.pairplot(data=talent\_data, hue="Attrition", vars=["Age", "MonthlyIncome",

"NumCompaniesWorked", # "PercentSalaryHike", "StandardHours", # "TotalWorkingYears",# "YearsAtCompany", # "YearsInCurrentRole"])

### 5.2特征工程

类别标签：Attrition 0 1-1表示已经离职，0表示未离职，这是目标预测值

数值型数据：Age、MonthlyIncome、NumCompaniesWorked、PercentSalaryHike、StandardHours、TotalWorkingYears、YearsAtCompany、YearsInCurrentRole、YearsSinceLastPromotion、YearsWithCurrManager

类别型数据：BusinessTravel、Department、EducationField、Gender、JobRole、MaritalStatus、Over18、OverTime

有序数据：DistanceFromHome、Education、EnvironmentSatisfaction、JobInvolvement、JobLevel、JobSatisfaction、RelationshipSatisfaction、StockOptionLevel、TrainingTimesLastYear、WorkLifeBalance、

# MaritalStatus 1100 non-null object

# BusinessTravel 1100 non-null object

# Department 1100 non-null object

# EducationField 1100 non-null object

# Gender 1100 non-null object

# JobRole 1100 non-null object

# Over18 1100 non-null object

# OverTime 1100 non-null object

#### 数据的处理1--不同类型特征选择

**import** pandas **as** pd  
  
talentData = pd.read\_csv(**"./train.csv"**)  
  
*# 0.类别标签  
# Attrition 1100 non-null int64  
# 1.数值型  
# Age 1100 non-null int64  
# MonthlyIncome 1100 non-null int64  
# NumCompaniesWorked 1100 non-null int64  
# PercentSalaryHike 1100 non-null int64  
# StandardHours 1100 non-null int64  
# TotalWorkingYears 1100 non-null int64  
# YearsAtCompany 1100 non-null int64  
# YearsInCurrentRole 1100 non-null int64  
# YearsSinceLastPromotion 1100 non-null int64  
# YearsWithCurrManager 1100 non-null int64*num\_cols = [**"Age"**, **"MonthlyIncome"**, **"NumCompaniesWorked"**, **"PercentSalaryHike"**, **"StandardHours"**,  
 **"TotalWorkingYears"**, **"YearsAtCompany"**, **"YearsInCurrentRole"**, **"YearsSinceLastPromotion"**,  
 **"YearsWithCurrManager"**]  
*# 2.类别型  
# BusinessTravel 1100 non-null object  
# Department 1100 non-null object  
# EducationField 1100 non-null object  
# Gender 1100 non-null object  
# JobRole 1100 non-null object  
# MaritalStatus 1100 non-null object  
# Over18 1100 non-null object  
# OverTime 1100 non-null object  
# cat\_cols=["BusinessTravel","Department","EducationField","Gender",  
# "JobRole","MaritalStatus","Over18","OverTime"]*cat\_cols = [**"Gender"**, **"MaritalStatus"**, **"OverTime"**]  
*# 3.有序性  
# DistanceFromHome 1100 non-null int64  
# Education 1100 non-null int64  
# EnvironmentSatisfaction 1100 non-null int64  
# JobInvolvement 1100 non-null int64  
# JobLevel 1100 non-null int64  
# JobSatisfaction 1100 non-null int64  
# PerformanceRating 1100 non-null int64  
# RelationshipSatisfaction 1100 non-null int64  
# StockOptionLevel 1100 non-null int64  
# TrainingTimesLastYear 1100 non-null int64  
# WorkLifeBalance 1100 non-null int64*ord\_cols = [**"DistanceFromHome"**, **"Education"**, **"EnvironmentSatisfaction"**,  
 **"JobInvolvement"**, **"JobLevel"**, **"JobSatisfaction"**, **"PerformanceRating"**,  
 **"RelationshipSatisfaction"**, **"StockOptionLevel"**, **"TrainingTimesLastYear"**,  
 **"WorkLifeBalance"**]  
*#将三种数据类型进行组合*total\_cols=num\_cols+cat\_cols+ord\_cols  
*#选择标签列*target\_cols=[**"Attrition"**]  
*#对已有的数据进行整合*useData=talentData[total\_cols+target\_cols]  
print(type(useData))  
print(useData.head())  
print(**"有多少列作为特征："**,len(total\_cols)) *#24*

#### 数据的处理2--筛选正负样本

*#筛选出正负样本2*posdata=useData[useData[**"Attrition"**]==1].reindex() *#<class 'pandas.core.frame.DataFrame'>*negdata=useData[useData[**"Attrition"**]==0].reindex() *#<class 'pandas.core.frame.DataFrame'>*print(**"="**\*100)  
print(type(posdata))  
print(posdata)  
print(type(negdata))  
print(negdata)  
*#如何取出pandas处理的数据之后的values？  
# posdata1=useData[useData["Attrition"]==1].values #<class 'pandas.core.frame.DataFrame'>  
# print(posdata1)  
# # [[34 6074 1 ... 3 3 1]  
# # [28 2596 1 ... 2 3 1]  
# # [24 1555 1 ... 2 3 1]  
# # ...  
# # [38 4855 4 ... 2 3 1]  
# # [22 2472 1 ... 2 3 1]  
# # [26 2042 6 ... 2 3 1]]*

#### 数据的处理3--数据集的划分在结合

*#对正负样本进行训练集和测试集的划分*train\_pos\_data=posdata.iloc[:int(len(posdata)\*0.8),:].copy()  
test\_pos\_data=posdata.iloc[int(len(posdata)\*0.8):].copy()  
train\_neg\_data=negdata.iloc[:int(len(negdata)\*0.8),:].copy()  
test\_neg\_data=negdata.iloc[int(len(negdata)\*0.8):].copy()  
*# len(train\_pos\_data) 142  
# test\_pos\_data 36  
# train\_neg\_data 737  
# test\_neg\_data 185  
#利用pandas的concat函数*train\_data=pd.concat([train\_pos\_data,train\_neg\_data])  
test\_data=pd.concat([test\_pos\_data,test\_neg\_data])  
print(test\_data.shape) *#(221, 25)  
# print(len(train\_data))  
# print(len(test\_data))  
# print(len(train\_pos\_data)/len(train\_neg\_data))*

#### 数据的处理4--特征编码(labelencoder)

*#cat\_cols = ["Gender", "MaritalStatus", "OverTime"]  
#目的：将类别型变形转化为数值性-onehot编码***from** sklearn.preprocessing **import** LabelEncoder  
**from** sklearn.preprocessing **import** OneHotEncoder  
Gender\_le=LabelEncoder()  
train\_data[**"Gender\_label"**]=Gender\_le.fit\_transform(train\_data[**"Gender"**])  
MaritalStatus\_le=LabelEncoder()  
train\_data[**"MaritalStatus\_label"**]=MaritalStatus\_le.fit\_transform(train\_data[**"MaritalStatus"**])  
OverTime\_le=LabelEncoder()  
train\_data[**"OverTime\_label"**]=OverTime\_le.fit\_transform(train\_data[**"OverTime"**])  
*# print(Gender\_le\_label)  
# print(MaritalStatus\_le\_label)  
# print(OverTime\_le\_label)  
#对于测试集的lableencoder编码*test\_data[**"Gender\_label"**]=Gender\_le.transform(test\_data[**"Gender"**])  
test\_data[**"MaritalStatus\_label"**]=MaritalStatus\_le.transform(test\_data[**"MaritalStatus"**])  
test\_data[**"OverTime\_label"**]=OverTime\_le.transform(test\_data[**"OverTime"**])  
  
print(test\_data.shape) *#(221, 28)*

#### 数据的处理4--简单聚合

*#简单聚合操作*print(**"Gender数据："**)  
print(train\_data.groupby(**"Gender\_label"**).size())  
*# Gender数据：  
# Gender\_label  
# 0 363  
# 1 516*print(**"MaritalStatus数据："**)  
print(train\_data.groupby(**"MaritalStatus"**).size())  
*# MaritalStatus  
# Divorced 184  
# Married 414  
# Single 281*print(**"OverTime数据："**)  
print(train\_data.groupby(**"OverTime"**).size())  
*# OverTime  
# No 636  
# Yes 243  
# dtype: int64*

#### 数据的处理4--特征编码(one-hot编码)

ohe=OneHotEncoder()  
train\_cat\_feat=ohe.fit\_transform(train\_data[[**"Gender\_label"**,**"MaritalStatus\_label"**,**"OverTime\_label"**]]).toarray()  
test\_Cat\_feat=ohe.transform(test\_data[[**"Gender\_label"**,**"MaritalStatus\_label"**,**"OverTime\_label"**]]).toarray()  
print(train\_cat\_feat[:5,:])  
*# [[0. 1. 0. 0. 1. 0. 1.]  
# [0. 1. 1. 0. 0. 1. 0.]  
# [0. 1. 0. 1. 0. 1. 0.]  
# [1. 0. 0. 0. 1. 0. 1.]  
# [1. 0. 0. 1. 0. 0. 1.]]  
#整合所有的特征*train\_num\_cols=train\_data[num\_cols].values  
print(type(train\_num\_cols))  
print(train\_num\_cols)  
*# [[ 34 6074 1 ... 7 0 6]  
# [ 28 2596 1 ... 0 0 0]  
# [ 24 1555 1 ... 0 0 0]  
# ...  
# [ 37 8834 1 ... 5 7 7]  
# [ 49 19161 3 ... 4 4 3]  
# [ 38 2821 3 ... 2 2 2]]*

#### 数据的处理5--将处理数据合并

train\_ord\_cols=train\_data[ord\_cols].values  
test\_num\_cols=test\_data[num\_cols].values  
test\_ord\_cols=test\_data[ord\_cols].values  
**import** numpy **as** np  
train\_feats=np.hstack((train\_num\_cols,train\_ord\_cols,train\_cat\_feat))  
train\_target=train\_data[target\_cols].values  
test\_feats=np.hstack((test\_num\_cols,test\_ord\_cols,test\_Cat\_feat))  
test\_target=test\_data[target\_cols].values  
print(test\_feats)  
print(type(test\_feats))  
*# [[1.900e+01 1.675e+03 1.000e+00 ... 1.000e+00 0.000e+00 1.000e+00]  
# [3.400e+01 5.304e+03 8.000e+00 ... 1.000e+00 0.000e+00 1.000e+00]  
# [2.900e+01 2.439e+03 1.000e+00 ... 1.000e+00 0.000e+00 1.000e+00]  
# ...  
# [3.500e+01 4.014e+03 3.000e+00 ... 0.000e+00 0.000e+00 1.000e+00]  
# [3.800e+01 5.405e+03 2.000e+00 ... 0.000e+00 0.000e+00 1.000e+00]  
# [3.700e+01 6.334e+03 4.000e+00 ... 0.000e+00 1.000e+00 0.000e+00]]  
#查看训练集和测试集的比例*print(len(test\_feats))  
print(test\_feats.shape)  
print(len(train\_feats))  
print(train\_feats.shape)  
*# 221  
# (221, 28)  
# 879  
# (879, 28)*

#### 数据的处理6--建立模型

*#训练模型***from** sklearn.ensemble **import** RandomForestClassifier  
rf=RandomForestClassifier(n\_estimators=100)  
rf.fit(train\_feats,train\_target)  
y\_pred=rf.predict(test\_feats)  
print(**"model in train set score is:"**,rf.score(train\_feats,train\_target))  
print(**"model in test set score is:"**,rf.score(test\_feats,test\_target))  
**from** sklearn.metrics **import** confusion\_matrix  
print(confusion\_matrix(test\_target,y\_pred))  
*# model in train set score is: 0.9988623435722411  
# model in test set score is: 0.8597285067873304  
# [[181 4]  
# [ 27 9]]***from** sklearn.metrics **import** classification\_report,accuracy\_score  
print(classification\_report(test\_target,y\_pred))  
print(**"accuracy:"**,accuracy\_score(test\_target,y\_pred))  
*#逻辑斯特回归-LR***from** sklearn.linear\_model **import** LogisticRegression  
lr=LogisticRegression()  
lr.fit(train\_feats,train\_target)  
y\_pred\_lr=lr.predict(test\_feats)  
print(**"model in train set score is:"**,lr.score(train\_feats,train\_target))  
print(**"model in test set score is:"**,lr.score(test\_feats,test\_target))  
**from** sklearn.metrics **import** confusion\_matrix  
print(confusion\_matrix(test\_target,y\_pred\_lr))  
*# model in train set score is: 0.8816837315130831  
# model in test set score is: 0.8461538461538461  
# [[174 11]  
# [ 23 13]]*

#### 数据的处理7--(掌握)使用dictvec方法结合数据集的切分方法整合

**import** pandas **as** pd  
  
talentData = pd.read\_csv(**"./train.csv"**)  
  
*# 0.类别标签  
# Attrition 1100 non-null int64  
# 1.数值型  
# Age 1100 non-null int64  
# MonthlyIncome 1100 non-null int64  
# NumCompaniesWorked 1100 non-null int64  
# PercentSalaryHike 1100 non-null int64  
# StandardHours 1100 non-null int64  
# TotalWorkingYears 1100 non-null int64  
# YearsAtCompany 1100 non-null int64  
# YearsInCurrentRole 1100 non-null int64  
# YearsSinceLastPromotion 1100 non-null int64  
# YearsWithCurrManager 1100 non-null int64*num\_cols = [**"Age"**, **"MonthlyIncome"**, **"NumCompaniesWorked"**, **"PercentSalaryHike"**, **"StandardHours"**,  
 **"TotalWorkingYears"**, **"YearsAtCompany"**, **"YearsInCurrentRole"**, **"YearsSinceLastPromotion"**,  
 **"YearsWithCurrManager"**]  
*# 2.类别型  
# BusinessTravel 1100 non-null object  
# Department 1100 non-null object  
# EducationField 1100 non-null object  
# Gender 1100 non-null object  
# JobRole 1100 non-null object  
# MaritalStatus 1100 non-null object  
# Over18 1100 non-null object  
# OverTime 1100 non-null object  
# cat\_cols=["BusinessTravel","Department","EducationField","Gender",  
# "JobRole","MaritalStatus","Over18","OverTime"]*cat\_cols = [**"Gender"**, **"MaritalStatus"**, **"OverTime"**]  
*# 3.有序性  
# DistanceFromHome 1100 non-null int64  
# Education 1100 non-null int64  
# EnvironmentSatisfaction 1100 non-null int64  
# JobInvolvement 1100 non-null int64  
# JobLevel 1100 non-null int64  
# JobSatisfaction 1100 non-null int64  
# PerformanceRating 1100 non-null int64  
# RelationshipSatisfaction 1100 non-null int64  
# StockOptionLevel 1100 non-null int64  
# TrainingTimesLastYear 1100 non-null int64  
# WorkLifeBalance 1100 non-null int64*ord\_cols = [**"DistanceFromHome"**, **"Education"**, **"EnvironmentSatisfaction"**,  
 **"JobInvolvement"**, **"JobLevel"**, **"JobSatisfaction"**, **"PerformanceRating"**,  
 **"RelationshipSatisfaction"**, **"StockOptionLevel"**, **"TrainingTimesLastYear"**,  
 **"WorkLifeBalance"**]  
*#将三种数据类型进行组合*total\_cols=num\_cols+cat\_cols+ord\_cols  
*#选择标签列*target\_cols=[**"Attrition"**]  
*#对已有的数据进行整合*useData=talentData[total\_cols] *#X*useDataLabel=talentData[target\_cols] *#Y*print(useData.shape)*#(1100, 25)*print(useData.info())*##<class 'pandas.core.frame.DataFrame'>*print(useDataLabel.shape)*#(1100, 25)*print(useDataLabel.info())*##<class 'pandas.core.frame.DataFrame'>  
# RangeIndex: 1100 entries, 0 to 1099  
# Data columns (total 1 columns):  
# Attrition 1100 non-null int64***from** sklearn.model\_selection **import** train\_test\_split  
X\_train,X\_test,y\_train,y\_test=train\_test\_split(useData,useDataLabel,test\_size=0.2,random\_state=22)  
*# print(y\_train)  
# #目的：将类别型变形转化为数值性-onehot编码***from** sklearn.feature\_extraction **import** DictVectorizer  
dv=DictVectorizer(sparse=**False**)  
X\_train\_dv=dv.fit\_transform(X\_train.to\_dict(orient=**"records"**))  
*#AttributeError: 'str' object has no attribute 'items'*X\_test\_dv=dv.transform(X\_test.to\_dict(orient=**"records"**))  
print(X\_train\_dv)  
*# test\_data\_dv=pd.DataFrame(test\_data\_dv,columns=)  
# test\_data\_dv.to\_csv("tesedemo.csv",sep=",")*print(dv.feature\_names\_)  
*#训练模型1***from** sklearn.ensemble **import** RandomForestClassifier  
rf=RandomForestClassifier(n\_estimators=100)  
rf.fit(X\_train\_dv,y\_train)  
y\_pred\_rf=rf.predict(X\_test\_dv)  
y\_pred\_train\_rf=rf.predict(X\_train\_dv)  
*#校验*print(**"model in trainset score is:%.2f%%"**%(rf.score(X\_train\_dv,y\_train)\*100))  
print(**"model in testset score is:%.2f%%"**%(rf.score(X\_test\_dv,y\_test)\*100))  
*# model in trainset score is:100.00%  
# model in testset score is:87.27%***from** sklearn.metrics **import** confusion\_matrix  
print(confusion\_matrix(y\_test,y\_pred\_rf))  
print(confusion\_matrix(y\_train,y\_pred\_train\_rf))  
print(**'='**\*100)  
*#训练模型2***from** sklearn.linear\_model **import** LogisticRegression  
lr=LogisticRegression()  
lr.fit(X\_train\_dv,y\_train)  
y\_pred\_lr=lr.predict(X\_test\_dv)  
*#校验*print(**"model in trainset score is:%.2f%%"**%(lr.score(X\_train\_dv,y\_train)\*100))  
print(**"model in testset score is:%.2f%%"**%(lr.score(X\_test\_dv,y\_test)\*100))  
*# model in trainset score is:88.18%  
# model in testset score is:87.73%*

### 5.3对业务进行建模：

机器学习--分类任务---集成学习：随机森林 LR(逻辑斯特回归)

校验模型：混淆矩阵（分对或分错）、准确率

#### 数据的处理6--建立模型

*#训练模型***from** sklearn.ensemble **import** RandomForestClassifier  
rf=RandomForestClassifier(n\_estimators=100)  
rf.fit(train\_feats,train\_target)  
y\_pred=rf.predict(test\_feats)  
print(**"model in train set score is:"**,rf.score(train\_feats,train\_target))  
print(**"model in test set score is:"**,rf.score(test\_feats,test\_target))  
**from** sklearn.metrics **import** confusion\_matrix  
print(confusion\_matrix(test\_target,y\_pred))  
*# model in train set score is: 0.9988623435722411  
# model in test set score is: 0.8597285067873304  
# [[181 4]  
# [ 27 9]]***from** sklearn.metrics **import** classification\_report,accuracy\_score  
print(classification\_report(test\_target,y\_pred))  
print(**"accuracy:"**,accuracy\_score(test\_target,y\_pred))  
*#逻辑斯特回归-LR***from** sklearn.linear\_model **import** LogisticRegression  
lr=LogisticRegression()  
lr.fit(train\_feats,train\_target)  
y\_pred\_lr=lr.predict(test\_feats)  
print(**"model in train set score is:"**,lr.score(train\_feats,train\_target))  
print(**"model in test set score is:"**,lr.score(test\_feats,test\_target))  
**from** sklearn.metrics **import** confusion\_matrix  
print(confusion\_matrix(test\_target,y\_pred\_lr))  
*# model in train set score is: 0.8816837315130831  
# model in test set score is: 0.8461538461538461  
# [[174 11]  
# [ 23 13]]*

### 5.4正负样本比例失衡？---如何处理？

欠采样:随机丢弃一些样本数据

过采样：类别值数据较少的部分重复采样(补充了更多数据)

## 5.过采样问题理解

参考过采样文档学习

python提供了一个框架----imbalance 0.3.2

## 6.项目总结