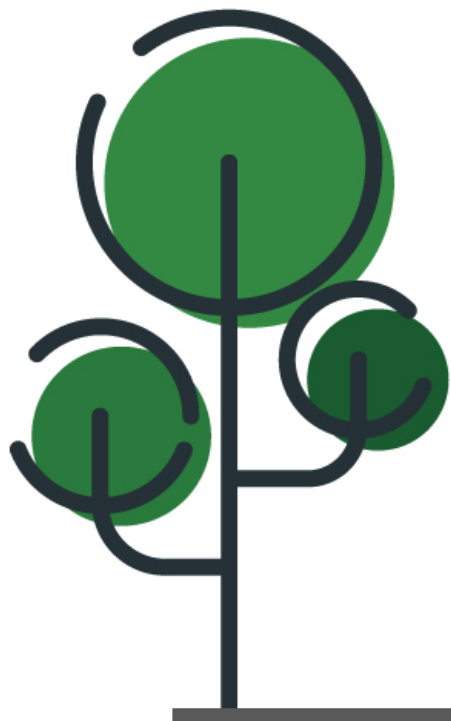


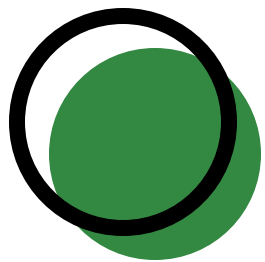
2  
0  
1  
8



# 基于深度学习和 强化学习的员工 离职解决策略



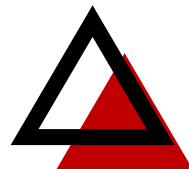
熊楚原、周昊



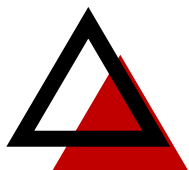
# CONTENTS



01.特征工程



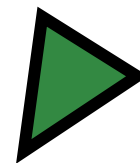
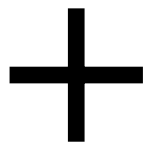
02.预测离职率



03.自动得到策略

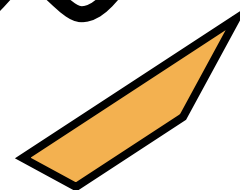


04.成果与总结



# PART ONE

## 特征工程



# 01.特征工程

## 数据集来源

- IBM Watson 大数据分析平台
- 美国1100个公民的31个特征

	A	B	C	D	E	F	G	H	I	J	K
1	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EducationField	EmployeeNumber	Environment	Gender	JobInvolvement
2	37	0	Travel_Rarely	Research & Development	1	4	Life Sciences	77	1	Male	2
3	54	0	Travel_Frequently	Research & Development	1	4	Life Sciences	1245	4	Female	3
4	34	1	Travel_Frequently	Research & Development	7	3	Life Sciences	147	1	Male	1
5	39	0	Travel_Rarely	Research & Development	1	1	Life Sciences	1026	4	Female	2
6	28	1	Travel_Frequently	Research & Development	1	3	Medical	1111	1	Male	2
7	24	0	Travel_Rarely	Sales	4	1	Medical	1445	4	Female	3
8	29	0	Travel_Rarely	Research & Development	9	5	Other	455	2	Male	2
9	36	0	Travel_Rarely	Sales	2	2	Medical	513	2	Male	2
10	33	0	Travel_Rarely	Research & Development	4	4	Medical	305	3	Female	2
11	34	0	Travel_Rarely	Research & Development	2	4	Technical Design	1383	3	Female	3
12	24	1	Travel_Rarely	Human Resources	22	1	Human Resources	1714	4	Male	1

# 01.特征工程



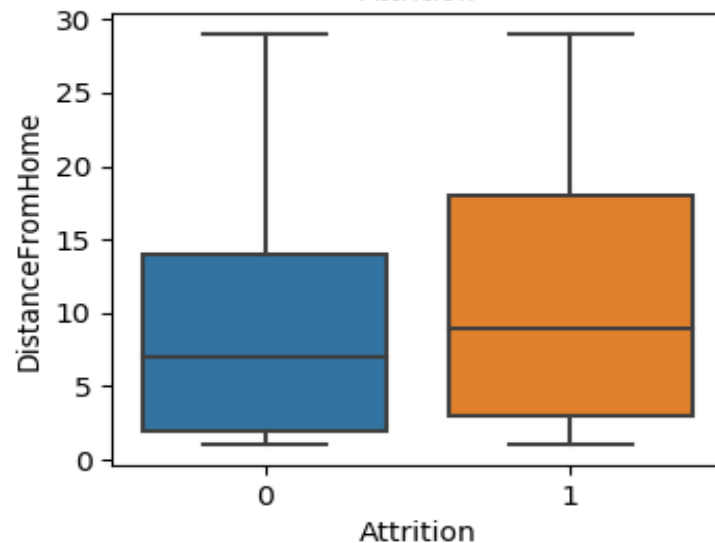
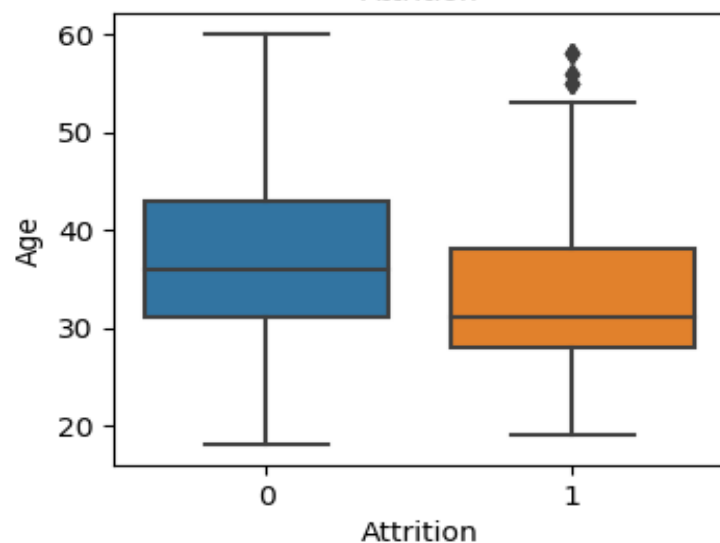
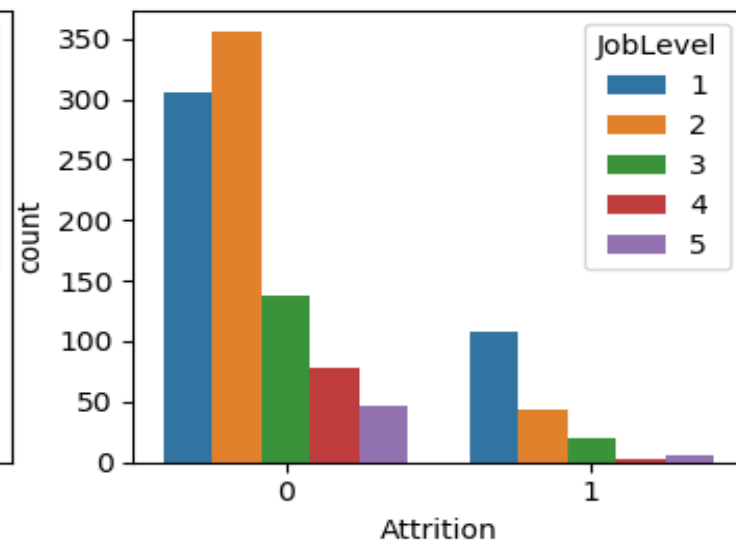
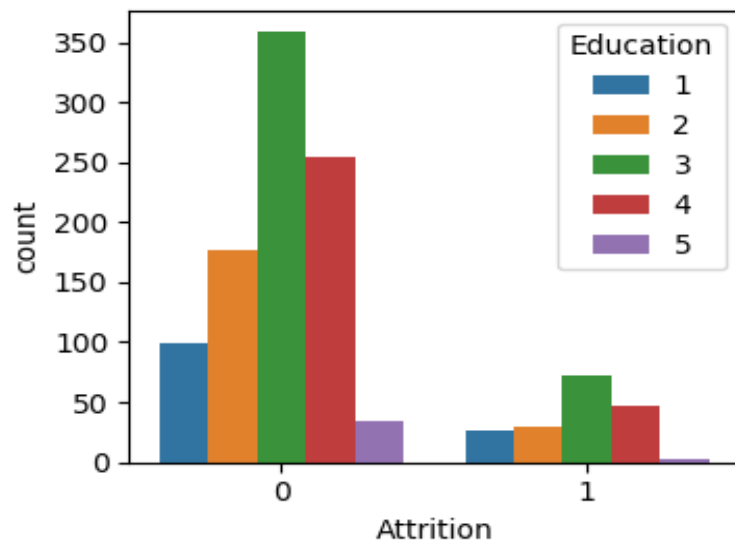
## 部分特征说明与预处理

特征名	特征含义	特征类别	处理
Age	年龄	数值型	归一化
Attrition	在职状态	0/1数值型	归一化
BusinessTravel	出差频率	类别型	字典化后归一化
Department	部门	类别型	字典化后归一化
DistanceFromHome	公司距家的距离	数值型	归一化
EducationField	研究领域	类别型	字典化后归一化
EmployeeNumber	员工号	数值型	归一化
EnvironmentSatisfaction	公司环境满意度	数值型	归一化
Gender	性别	类别型	字典化后归一化
JobInvolvement	职业与专业相关度	数值型	归一化
JobLevel	职位等级	数值型	归一化
JobRole	职位名	类别型	字典化后归一化
MaritalStatus	婚姻状态	类别型	字典化后归一化
MonthlyIncome	月薪	数值型	归一化

# 01.特征工程



部分特征相关性  
分析图



# 01.特征工程

- 特征选择、预处理、降维后，单个员工被抽象为了27元的向量
- 构造数据集分割函数，并作为模型的输入

```
[46, 21, 2, 4, 3, 2, 2, 4, 22, 4, 4, 80, 1, 13, 2, 4, 9, 7, 3, 7, 1, 0, 1, 1, 0, 1, 0]  
[37, 10, 4, 4, 3, 2, 1, 1, 15, 3, 2, 80, 0, 10, 4, 1, 10, 3, 0, 8, 1, 0, 1, 0, 4, 2, 0]  
[29, 1, 2, 2, 2, 1, 1, 1, 11, 3, 4, 80, 1, 1, 1, 3, 1, 0, 0, 0, 1, 0, 0, 1, 2, 1, 1]  
[41, 7, 2, 4, 3, 2, 3, 6, 14, 3, 2, 80, 0, 8, 6, 3, 2, 2, 2, 1, 1, 0, 1, 1, 4, 2, 0]  
[41, 1, 3, 3, 3, 2, 1, 0, 17, 3, 4, 80, 1, 10, 2, 3, 9, 3, 1, 7, 2, 1, 5, 1, 7, 0, 0]  
[54, 10, 3, 3, 3, 2, 1, 6, 19, 3, 4, 80, 0, 9, 3, 3, 5, 2, 1, 4, 1, 0, 1, 1, 4, 2, 1]
```

```
137 def getData(csvPath, label='Attrition', sampleType=resample, frac=0.7):  
138     reader = dataPreProcessing(readFile(csvPath))  
139     print(reader.keys())  
140     return getTrainAndTestData(*sampleType(reader, label), frac)
```

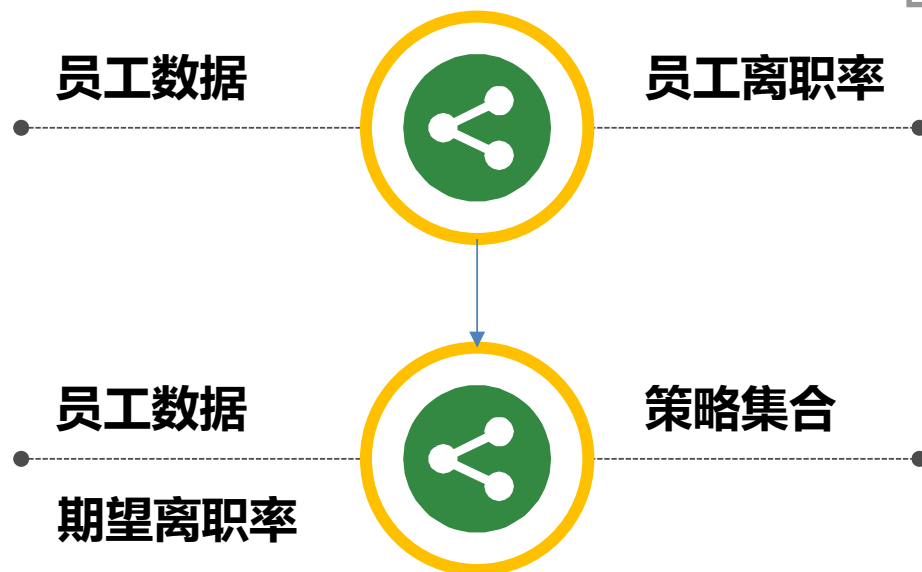
csvPath为数据集表的路径，label表示标签名，sampleType表示抽样类别，有重采样和欠采样两种，frac表示训练集和测试集的比例

返回处理好的4个数据矩阵，分别表示X\_train, X\_test, Y\_train, Y\_test

# 01.特征工程

## 待解决问题

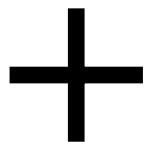
- 如何预测一个员工的离职率
- 如何自动生成成本较低的降低该员工离职率的策略集合



采用MLP三层网络模型

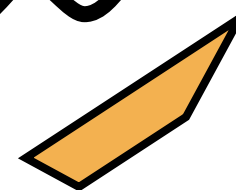
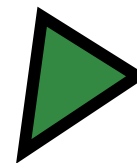
采用强化学习Sarsa算法





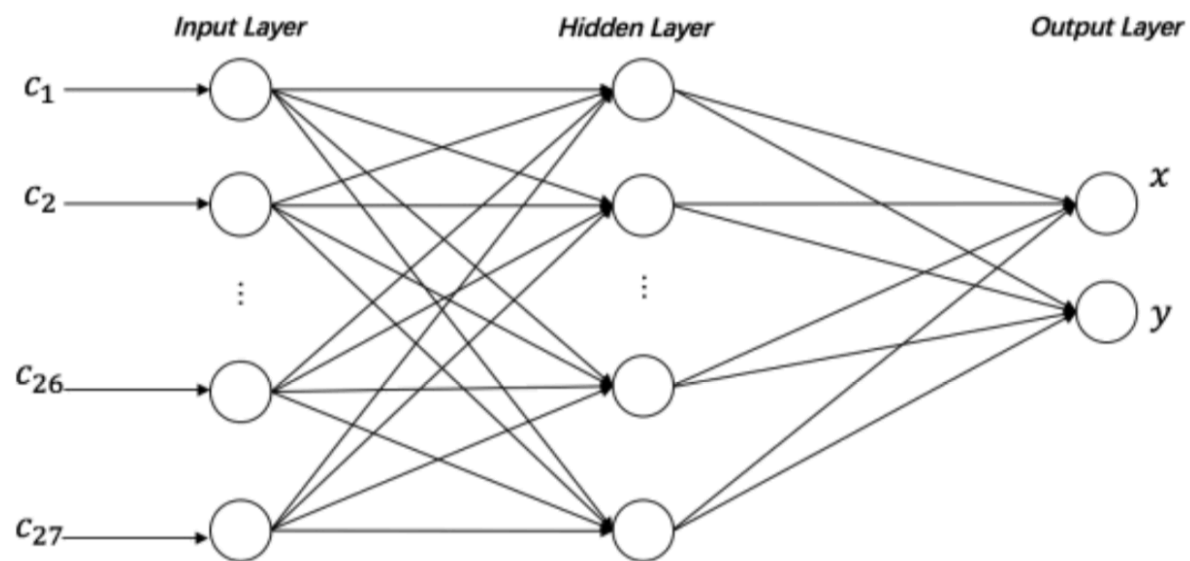
**PART TWO**

**预测离职率**



## 02.预测离职率

搭建网络



$$\text{Hidden} = \text{Sigmoid}(C * w_1 + b_1) \quad \text{Output} = \text{Softmax}(\text{Hidden} * w_2 + b_2)$$

### 网络结构

直接进行多层拟合，采用Sigmoid和Softmax激励函数

每次训练输入：

- 27维度向量
- 在职/离职独热标签

每次训练输出：

- 不离职指数 $x$
- 离职指数 $y$
- 其中 $x+y$ 恒为1

损失交叉熵( $y_$ 为标签)：

```
tf.reduce_mean(-tf.reduce_sum(y_*tf.log(y)))
```



## 02.预测离职率

### 训练与验证

训练10000次并将训练参数保存（防止过拟合设置0.75激活度）

```
39 #第三步、训练步骤
40 tf.global_variables_initializer().run()
41 if isTrain:
42     saver.restore(sess, tf.train.latest_checkpoint('./ckpt/'))
43 else:
44     for i in range(10000): #训练次数的改变可影响准确率
45         batch_xs, batch_ys = X_train, Y_train
46         train_step.run({x: batch_xs, y: batch_ys, keep_prob: 0.75})
47         if i % 500 == 0:
48             saver.save(sess=sess, save_path='./ckpt/mnist.ckpt', global_step=i)
```

分别在测试集和验证集上测试准确率

```
50 #第四步、对模型进行准确率预测
51 correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
52 accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
53 print(accuracy.eval({x: X_test, y_: Y_test, keep_prob: 1.0}))
54 print(accuracy.eval({x: X_train, y_: Y_train, keep_prob: 1.0}))
```



## 02.预测离职率

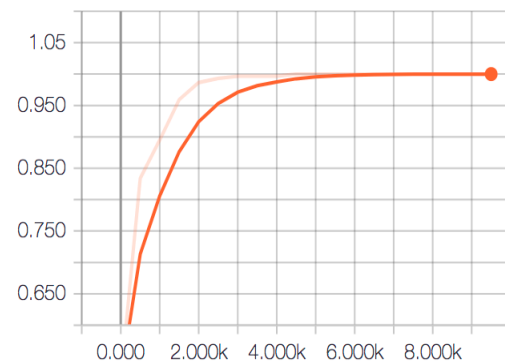
观察结果并封装

得到准确率，可以进行参数封装

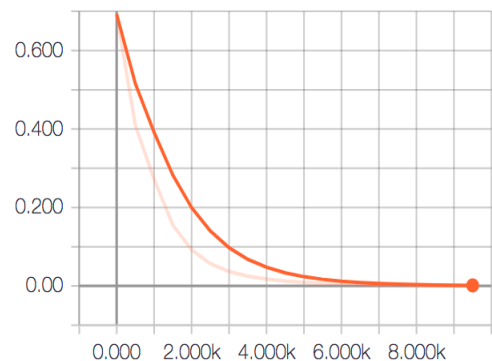
训练集上准确率	96.41%
测试集上准确率	94.96%

( TensorBoard上看到训练8k-1w次时效果最好 )

accuracy

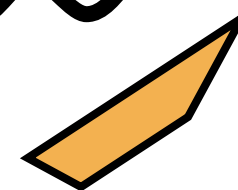
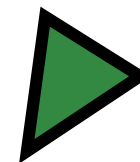
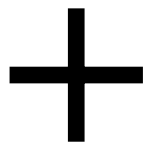


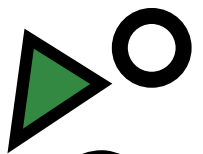
loss



调用保存参数可在该session上直接进行输入27维度数据得到离职率

```
40 tf.global_variables_initializer().run()  
41 if isTrain:  
42     saver.restore(sess, tf.train.latest_checkpoint('./ckpt/'))
```





## 03.得到策略

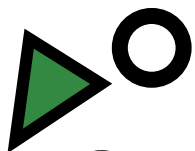
### 形式化定义

员工分析：

- 形式化定义员工为 员工 $\langle C, S, A \rangle$ ，C即表示员工属性，S表示员工状态即不离职率，A表示员工动作
- 定义员工动作成本相同且较小的一系列动作集合A为元动作，即pA，定义该成本为pCost
- 采取动作pA后，员工状态(离职率)可进行如此转换： $S' \leftarrow CalculateS(\langle C, S, A \rangle, pA)$
- CalculateS为问题一封装的函数，即输入一个员工和采取的动作，输出其数据变化后当前不离职概率

部分元动作集合：

动作名	作用于员工特征	作用后特征变化
加设班车	家庭距离	5
外派接受培训	受教育程度	0.05
提高职位级别	职位级别	0.05
提高工资上涨比例	薪资水平	0.05
加强人际合作	人际关系满意程度	0.5
提高职位满意度	职位满意度	0.1
增强环境满意度	环境满意度	0.1



## 03.得到策略

算法

Sarsa算法描述：

Algorithm1: 原 SARSA 算法(on-policy) ↵

核心代码: ↵

1. → Initialize:  $Q(s,a), \forall s \in S; a \in A, Q(\text{terminal\_state},a) = 0$  ↵
2. → Repeat(for each episode): ↵
3. → ··· Initialize:  $S$  ↵
4. → ··· Choose  $A$  from  $S$  using policy derived from  $Q$ (e.g.,  $\epsilon$  - greedy) ↵
5. → ··· Repeat(for each step) ↵
6. → ····· Take action  $A$ , observe  $R, S'$  ↵
7. → ····· Choose  $A'$  from  $S'$  using policy derived from  $Q$ (e.g.,  $\epsilon$  - greedy) ↵
8. → ·····  $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$  ↵
9. → ·····  $S \leftarrow S', A \leftarrow A'$  ↵
10. → ··· Until  $S$  is terminal ↵
11. → Until end episode ↵



## 03.得到策略

### 算法

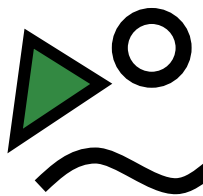
本问题下改进的Sarsa算法描述：（为什么不使用Q-learning）

Algorithm2: 改进后的 SARSA 算法

核心代码：

1. → Initialize:  $Q(s,a)$ ,  $\forall s \in S; a \in A$ ,  $Q(\text{terminal\_state},a) = 0$
2. → Repeat (for each episode):
3. → ... Initialize:  $S$
4. → ... Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
5. → ... Repeat (for each step):
6. → ... Take action  $A$
7. → ...  $S' \leftarrow \text{CalculateS}(< C, S, A >)$
8. → ...  $R \leftarrow (S - S')$
9. → ... Choose  $A'$  from  $S'$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
10. → ...  $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$
11. → ...  $S \leftarrow S', A \leftarrow A'$
12. → ... Until  $S$  is terminal
13. → Until end episode





## 03.得到策略

### 策略生成

Step 1: 初始化Q表,  $Q_{i,j} = 0, i = 1, 2, \dots, |S|, j = 1, 2, \dots, |A|$ 。

Step 2: 设置训练次数t次, 并设定目标值 $S_{end}$ ,  $S_p = S_{start}$ 。

Step 3: 若是首次选择动作, 则采取随机选择的方式 $a = \text{random}(A)$ , 如果并非首次选取动作, 则选择 $A[\text{argmax}(Q_{S_p})]$ 对应的动作a。

Step 4: 计算 $S_{temp} = \text{CalculateS}(\text{Agent} < C, S, a >)$ ,  $R = S_{temp} - S_p$ 。

Step 5: 计算预测值 $Q_{predict} = Q(S_p, a)$ 。

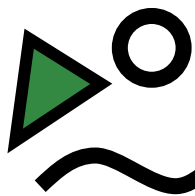
Step 6: 再次选取 $A[\text{argmax}(Q_{S_{temp}})]$ 对应的动作 $a_2$ 。

Step 7: 计算实测值 $Q_{target} = R + \gamma Q(S_{temp}, a_2)$ 。

Step 8: 按下式对Q表进行更新:  $Q(S_p, a) = Q(S_p, a) + \alpha[Q_{target} - Q_{predict}]$ 。

Step 9: 若 $S_p$ 与 $S_{end}$ 相等或是训练次数达到t次, 则进入Step 10, 否则更新中间状态 $S_p = S_{temp}$ , 并返回Step 3 进行再次训练。

Step 10: 将最终结果集合Ans置为空, 再次从Step 3开始执行, 将每一步循环过程中采取的动作a加入Ans, 并将结果输出, 结束算法。



## 04.得到策略

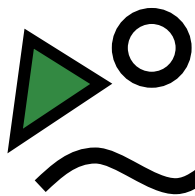
举例

对1号员工（部分特征）：

特征	值
家庭距离	46
受教育程度	2
职位级别	2
薪资水平	3
人际关系满意程度	4
职位满意度	2
环境认可度	4

第一次训练结果：

采取动作	当前不离职概率	目标不离职概率
增强环境满意度	0.014	0.8
外派接受培训	0.035	0.8
调动工作性质	0.072	0.8
增强自身体质	0.076	0.8
加强国际合作	0.145	0.8
增强职位满意度	0.153	0.8
加设班车	0.164	0.8
增强环境满意度	0.143	0.8
提高工资上涨比例	0.522	0.8
加强国际合作	0.358	0.8
加设班车	0.381	0.8
加强国际合作	0.403	0.8
增强自身体质	0.479	0.8
外派接受培训	0.505	0.8
外派接受培训	0.509	0.8
外派接受培训	0.518	0.8
外派接受培训	0.523	0.8
加设班车	0.524	0.8
加强国际合作	0.517	0.8
提高职位级别	0.732	0.8
增强环境满意度	0.724	0.8
增强自身体质	0.715	0.8
提高工资上涨比例	0.868	0.8



## 04.得到策略

举例

其他例子：

第三十次训练结果：

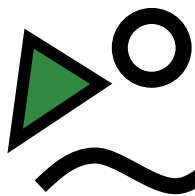
采取动作	当前不离职概率	目标不离职概率
增强环境满意度	0.014	0.8
加强人际合作	0.139	0.8
加强职位满意度	0.166	0.8
提高工资上涨比例	0.503	0.8
加强人际合作	0.655	0.8
增强自身体质	0.691	0.8
提高职位级别	0.832	0.8

第 1 次训练：

1 加设一些班车	S_p: 0.651297	S_end: 0.8
2 加设一些班车	S_p: 0.652658	S_end: 0.8
3 加设一些班车	S_p: 0.654277	S_end: 0.8
4 加设一些班车	S_p: 0.65617	S_end: 0.8
5 外出进行一点培训	S_p: 0.659646	S_end: 0.8
6 外出进行一点培训	S_p: 0.663805	S_end: 0.8
7 外出进行一点培训	S_p: 0.668687	S_end: 0.8
8 加强一点人际合作	S_p: 0.728813	S_end: 0.8
9 加设一些班车	S_p: 0.732535	S_end: 0.8
10 加设一些班车	S_p: 0.736491	S_end: 0.8
11 加设一些班车	S_p: 0.740671	S_end: 0.8
12 加设一些班车	S_p: 0.745067	S_end: 0.8
13 加强一点锻炼	S_p: 0.730831	S_end: 0.8
14 加设一些班车	S_p: 0.734552	S_end: 0.8
15 加设一些班车	S_p: 0.73853	S_end: 0.8
16 加设一些班车	S_p: 0.74276	S_end: 0.8
17 加设一些班车	S_p: 0.747235	S_end: 0.8
18 加强一点人际合作	S_p: 0.818069	S_end: 0.8

第 30 次训练：

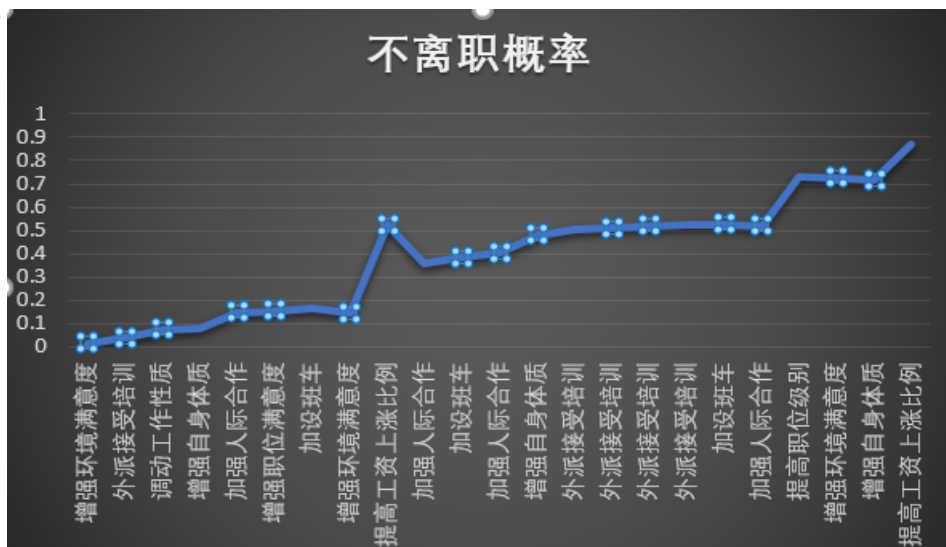
1 提高一点环境认可	S_p: 0.651738	S_end: 0.8
2 加强一点人际合作	S_p: 0.707067	S_end: 0.8
3 外出进行一点培训	S_p: 0.709624	S_end: 0.8
4 外出进行一点培训	S_p: 0.712719	S_end: 0.8
5 外出进行一点培训	S_p: 0.716394	S_end: 0.8
6 加强一点人际合作	S_p: 0.792873	S_end: 0.8
7 升职	S_p: 0.799595	S_end: 0.8
8 外出进行一点培训	S_p: 0.803007	S_end: 0.8



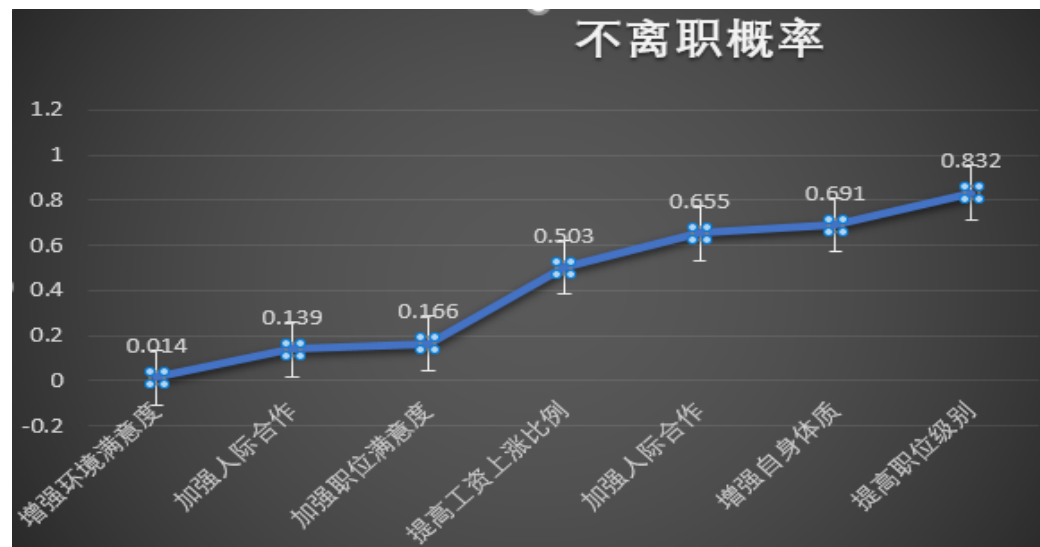
## 04.得到策略

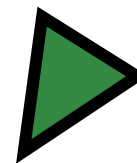
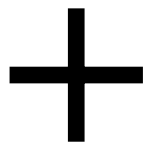
举例

第一次训练趋势图：



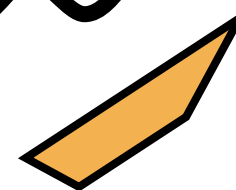
第三十次训练趋势图：





# PART **FOUR**




## 成果与总结



## 04.成果与总结

### 主要成果



-  处理了数据集并形式化定义了员工
-  能够得到一个员工的离职率
-  能够通过一个员工的数据和并结合离职率计算方法自动生成成本较小的策略集合

## 04.成果与总结

### ①预测准确率高

在训练集和测试集上准确率都达到了至少90%，比其他传统机器方法（如逻辑回归等）准确率大大的提升

### ①使用方便

直接调用函数输入数据表指定标签即可批量得到员工离职率，很方便

## 优点

### ①潜力大

在实验型数据集上可以达到很好的效果，进一步研究可以很大的降低企业的人力成本

### ①切合时代

该项目追求准确率高、自动化、实用性强等特点，非常适用于当今时代

2  
0  
1  
8



2018.07.20