Python机器学习实战案例

——保险产品推荐

业务背景分析

- 保险产品的多样性、客户特征的复杂性以及需求差异使得保险推荐存在相当大的不确定性，如何精准识别用户、降低销售风险、提升推荐成功率，成为当前一个非常热门的研究和应用话题。
- 通过对用户本身属性和过往保险购买记录分析客户特点，可以对广大用户进行个人信息的有效筛选，从购买保险的用户群体中提取共同的特征，进而针对这些特征规律提高投放精准性。
- 本案例是针对移动房车险的预测，其中保险公司提供了以家庭为单位的历史数据，包括家庭的各类特征属性和历史投保记录。通过对历史数据的分析，建立移动房车险的预测模型。

数据概况

- 数据分为训练数据和测试数据两部分，其中训练数据5822条，测试数据4000条。
- 每一条数据有86个字段，1~43字段为用户基本属性，包括财产、宗教、家庭情况、教育程度、职位、收入水平等；44~85字段为用户历史保险购买情况，第86个字段为目标预测字段，取值为0和1，表示用户是否购买该房车险。
- 这是一个典型的二分类问题。

字段取值类型

数值类型	字段说明
实际数值	➢ 家庭房产数量：1~10 ➢ 平均房产数量：1~6 ➢ …
范围取值	➢ 年龄：1~6（1 表示 20~30、…、6 表示 7~80） ➢ 金额（欧元）：0~9（0 表示 0、1 表示 1~49、2 表示 50~99、…、9 表示超过 20000） ➢ …
类别取值	➢ 客户主类别：1~10（代表功成享受、退休信教、保守家庭、…） ➢ 客户子类别：1~64（代表高收入、单身青年、中产阶级、丁克、…） ➢ …
百分比取值	➢ 0~9 取值（0 表示 0,1 表示 1~10%、…、9 表示 100%）

数据提取

- 对数据的整体情况进行分析。

```
import pandas as pd
data = pd.read_excel('data/data.xlsx')
print(data.shape)
print(data.describe())
```

```
(5822, 86) (4000, 86)
       客户次类别         房产数         每房人数         平均年龄         客户主类别  \
count  5822.000000  5822.000000  5822.000000  5822.000000  5822.000000
mean     24.253349     1.110615     2.678805     2.991240     5.773617
std      12.846706     0.405842     0.789835     0.814589     2.856760
min       1.000000     1.000000     1.000000     1.000000     1.000000
25%      10.000000     1.000000     2.000000     2.000000     3.000000
50%      30.000000     1.000000     3.000000     3.000000     7.000000
75%      35.000000     1.000000     3.000000     3.000000     8.000000
max      41.000000    10.000000     5.000000     6.000000    10.000000

       客户次类别         房产数         每房人数         平均年龄         客户主类别  \
count  4000.000000  4000.00000  4000.000000  4000.000000  4000.000000
mean     24.253000     1.10600     2.675750     3.004000     5.787000
std      13.022822     0.42108     0.767306     0.790025     2.899609
min       1.000000     1.00000     1.000000     1.000000     1.000000
25%      10.000000     1.00000     2.000000     3.000000     3.000000
50%      30.000000     1.00000     3.000000     3.000000     7.000000
75%      35.000000     1.00000     3.000000     3.000000     8.000000
max      41.000000    10.00000     6.000000     6.000000    10.000000
```

数据预处理（1）

- 将训练集和测试集合并以方便后续的数据探索和预处理。

```
train['source'] = 'train'
test['source'] = 'test'
data = pd.concat([train, test], ignore_index=True, sort=False)
```

判断数据是否有空白值或缺失值，数据缺失将影响数据分析的特征处理

```
nan_count = data.isnull().sum().sort_values(ascending=False)
nan_ratio = nan_count/len(data)
nan_data = pd.concat([nan_count, nan_ratio], axis=1, keys=['count', 'ratio'])
print(nan_data.head(10))
```

	count	ratio
source	0	0.0
一辆车	0	0.0
非熟练劳工	0	0.0
社会阶层A	0	0.0
社会阶层B1	0	0.0
社会阶层B2	0	0.0
社会阶层C	0	0.0
社会阶层D	0	0.0
租房子	0	0.0
房主	0	0.0

数据预处理（2）

- 统计每个特征对应的类别数目。

```
count = data.apply(lambda x:len(x.unique())).sort_values(ascending=False)
print(count.head(10))
```

```
客户次类别      40
其它关系占比     10
社会阶层D       10
收入低于30      10
有子女          10
租房子          10
高等教育        10
投保寿险        10
房主            10
高管            10
dtype: int64
```

数据预处理（3）

- 数据均为数值型，因此统计数据的分布情况。下列代码使用Pandas统计数据的训练集的偏度。

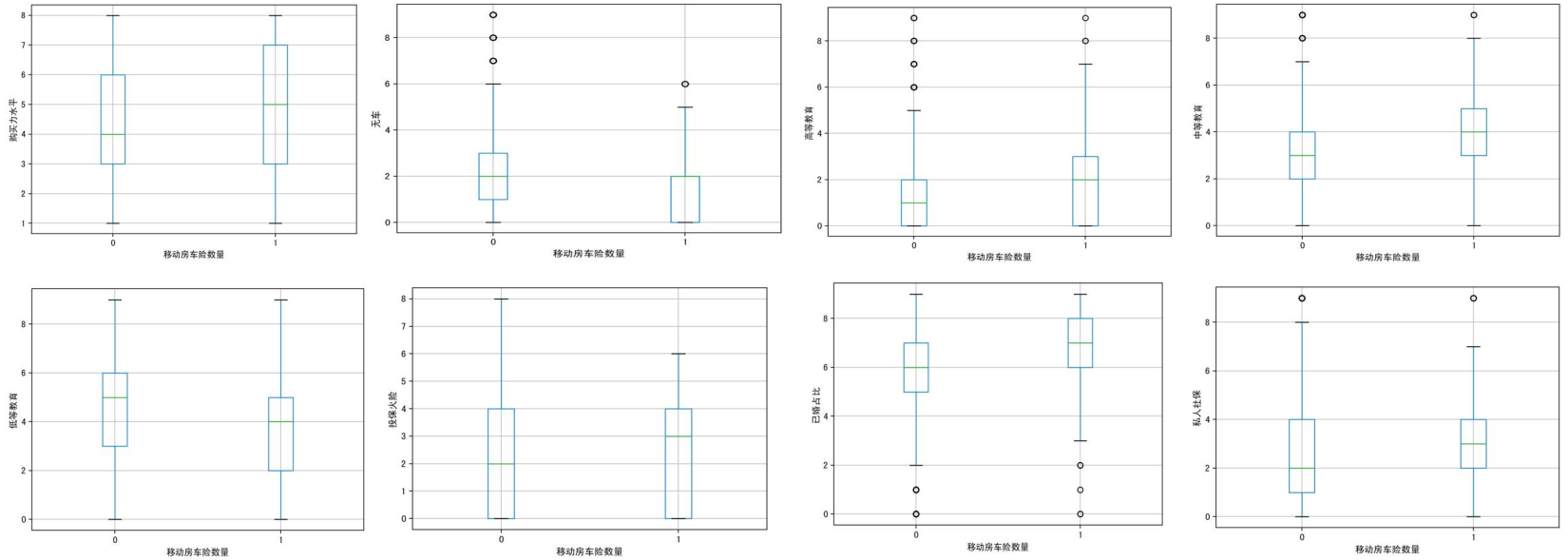
```
train = data.loc[data['source'] == "train"]
test = data.loc[data['source'] == "test"]
train.drop(['source'], axis=1, inplace=True)
skewness = train.iloc[:, :-1].apply(lambda x: x.skew())
skewness = skewness.sort_values(ascending=False)
print(skewness.head(10))
```

```
投保冲浪险          60.643858
投保冲浪险数量        44.030286
投保卡车险数量        33.861678
投保农机险数量        29.458616
投保卡车险          26.926855
投保农机险          19.228598
投保身残险数量        18.719503
投保个人意外险        18.631563
投保货车险数量        16.735105
投保财产险          16.654733
投保身残险          16.001588
投保船险           15.919394
投保船险数量         14.626767
第三方商业险数量       14.337519
投保个人意外险数量      13.598054
dtype: float64
```

数据预处理（4）

- 通过箱图对不同条件变量对目标变量的区分度。

```
plot_data = train[['购买力水平', '移动房车险数量']]
plot = plot_data.boxplot(column='购买力水平', by='移动房车险数量')
plot.set_xlabel('移动房车险数量')
plot.set_ylabel('购买力水平')
plt.show()
```



数据预处理（5）

- 可以发现购买移动房车险的家庭相对未购买移动房车险的家庭，有下列的一些特征：
- 购买力水平较高，平均收入较高；
- 平均教育水平较高；
- 投保火险的比例略高；
- 家庭成员中已婚的比例较高；
- 私人社保投保比例较高；
- 公共社保的投保比例较低；
- 农场主这类人群极少投移动房车险；
- 高管层次的人群比例较高。

数据预处理（6）

- 数据的维度较大，因此，使用特征选择降低数据的维度。
- 数据探索之后的特征变量维度为85，有较多的变量，因此考虑进行特征选择。计算每个特征与目标变量的相关系数，保留相关系数大于0.01的特征。

```
corr_target = train.corr()['移动房车险数量']
important_feature = corr_target[np.abs(corr_target) >= 0.01].index.tolist()
print(len(important_feature))
train = train[important_feature]
```

对测试集进行类似的处理

```
test = test[important_feature]
```

```
train.to_csv('train_preprocess.csv', encoding='utf_8_sig')
test.to_csv('test_preprocess.csv', encoding='utf_8_sig')
```

分类模型构建

- 使用sklearn构建最基础的决策树模型。

```
import pandas as pd
import numpy as np
from sklearn import tree
def load_data(path):
    data = pd.read_csv(path, encoding='utf-8')
    x, y = data.iloc[:, :-1], data.iloc[:, -1]
    return x, y
def build_model(x, y):
    classifier = tree.DecisionTreeClassifier()
    classifier.fit(x, y)
    return classifier
if __name__ == '__main__':
    train_x, train_y = load_data('data/train_preprocess.csv')
    classifier = build_model(train_x, train_y)
```

决策树测试

➢ 对于构建的模型，使用测试集数据进行评价。

```
import pandas as pd
import numpy as np
from sklearn import tree
from sklearn import metrics
from sklearn.model_selection import cross_validate
def load_data(path):
    data = pd.read_csv(path, encoding='utf-8')
    x, y = data.iloc[:, :-1], data.iloc[:, -1]
    return x, y
def build_model(x, y):
    classifier = tree.DecisionTreeClassifier()
    classifier.fit(x, y)
    return classifier
def test_model(classifier):
    test_x, test_y = load_data('data/test_preprocess.csv')
    scores = cross_validate(classifier, test_x, test_y, cv=5, scoring=('accuracy', 'precision', 'recall', 'f1', 'roc_auc'))
    return scores
if __name__ == '__main__':
    train_x, train_y = load_data('data/train_preprocess.csv')
    classifier = build_model(train_x, train_y)
    scores = test_model(classifier)
    print('Accuracy %.4f' % (np.mean(scores['test_accuracy'])))
    print('Precision %.4f' % (np.mean(scores['test_precision'])))
    print('Recall %.4f' % (np.mean(scores['test_recall'])))
    print('F1 %.4f' % (np.mean(scores['test_f1'])))
    print('AUC %.4f' % (np.mean(scores['test_roc_auc'])))
```

```
Accuracy 0.7910
Precision 0.0870
Recall 0.2647
F1 0.1310
AUC 0.5445
```

ID3决策树、逻辑回归模型测试结果

- 修改分类模型为ID3决策树、逻辑回归，重复上述实验。
- 可以看出更改模型并没有改变准确率和召回率低的情况，因此实验的数据出现了问题。再次分析实验的数据，发现样本集存在严重的不平衡问题，导致模型严重过拟合。

模型	评价指标	测试值
ID3决策树	Accuracy	0.8330
	Precision	0.1267
	Recall	0.3067
	F1	0.1794
	AUC	0.5865
逻辑回归	Accuracy	0.9405
	Precision	0.0000
	Recall	0.0000
	F1	0.0000
	AUC	0.5000

平衡数据集

- 如果不处理不平衡的数据，模型常常会直接返回占比高的值，导致模型过拟合，测试集上效果非常差。
- 对占比较少的类别进行上采样，对占比较大的类别进行下采样，然后将采样后的样本混合，使得样本集达到平衡。

```
print(train['移动房车险数量'].value_counts())
```

```
0    5474
1     348
Name: 移动房车险数量, dtype: int64
```

可见样本显著不平衡，投保数低于总数的6%。

```
from sklearn.utils import resample, shuffle
train_up = train[train['移动房车险数量'] == 1]
train_down = train[train['移动房车险数量'] == 0]
train_up = resample(train_up, n_samples=696, random_state=0)
train_down = resample(train_down, n_samples=1095, random_state=0)
train = shuffle(pd.concat([train_up, train_down]))
```

样本平衡后的模型测试结果

- 重新进行三个模型实验，可见三种模型的效果都有明显的提升，特别是召回率。

模型	评价指标	测试值
CART决策树	Accuracy	0.7432
	Precision	0.0970
	Recall	0.3992
	F1	0.1561
	AUC	0.5821
ID3决策树	Accuracy	0.7462
	Precision	0.1040
	Recall	0.4286
	F1	0.1674
	AUC	0.5975
逻辑回归	Accuracy	0.7678
	Precision	0.1265
	Recall	0.4916
	F1	0.2012
	AUC	0.6384

算法调参（1）

- 机器学习中参数对模型的影响巨大，调参也是费时费力的事情，对ID3决策树和逻辑回归模型的参数进行调整。
- 对于ID3决策树，首先调整决策树的最大深度，依次为5、10、15、20，可见随着最大深度的增加，精度先降后升，召回率先升后降，AUC变化不大，综合来看在最大深度为10左右有最好的表现。

模型	评价指标	测试值
ID3决策树最大深度=5	Accuracy	0.8285
	Recall	0.3866
	AUC	0.6215
ID3决策树最大深度=10	Accuracy	0.7115
	Recall	0.5000
	AUC	0.6124
ID3决策树最大深度=15	Accuracy	0.7472
	Recall	0.4286
	AUC	0.5980
ID3决策树最大深度=15	Accuracy	0.7615
	Recall	0.3908
	AUC	0.5879

算法调参（2）

- 随机调整切分时考虑的特征数、内部节点分裂时最少样本数、叶节点最少样本数等参数，可见实验结果没有特别大的变化。

模型	评价指标	测试值
ID3决策树最大深度=10切分时考虑的特征数=20	Accuracy	0.7718
	Recall	0.4454
	AUC	0.6189
ID3决策树最大深度=10内部节点分裂时最少样本数=50	Accuracy	0.6472
	Recall	0.5588
	AUC	0.6058
ID3决策树最大深度=10叶节点最少样本数=50	Accuracy	0.7210
	Recall	0.4958
	AUC	0.6155

算法调参（3）

- 对于逻辑回归模型，修改参数惩罚性进行实验，分别设置为L1惩罚与L2惩罚。可见在有L2惩罚项时有更低的召回率，而F1与AUC值相差不大。

模型	评价指标	测试值
逻辑回归L1惩罚	Accuracy	0.8037
	Precision	0.1299
	Recall	0.4034
	F1	0.1965
	AUC	0.6162
逻辑回归L2惩罚	Accuracy	0.8235
	Precision	0.1355
	Recall	0.3655
	F1	0.1977
	AUC	0.6090

模型比较（1）

- 除了决策树和逻辑回归，很多模型都可以解决二分类问题。sklearn中提供了一些开箱即用的分类模型，下面将尝试多层感知机、高斯朴素贝叶斯、支持向量机、随机森林和AdaBoost等模型。

```
from sklearn.neural_network import MLPClassifier
def build_model(x, y):
    classifier = MLPClassifier()
    classifier.fit(x, y)
    return classifier
```

多层感知机的测试结果

模型	评价指标	测试值
多层感知机隐层=1*100优化器：Adam激活函数：Relu初始学习率=0.001迭代次数=200	Accuracy	0.5130
	Precision	0.0909
	Recall	0.7983
	F1	0.1632
	AUC	0.6466

模型比较（2）

- 对于K-近邻（KNN）分类器最重要的参数为K的选择，K值太小容易过拟合，太大容易欠拟合，因此需要多次调参确定最佳的K值。

```
from sklearn.neighbors import KNeighborsClassifier
def build_model(x, y):
    classifier = KNeighborsClassifier(5)
    classifier.fit(x, y)
    return classifier
```

使用高斯朴素贝叶斯分类器

```
from sklearn.naive_bayes import GaussianNB
def build_model(x, y):
    classifier = MultinomialNB()
    classifier.fit(x, y)
    return classifier
```

模型	评价指标	测试值
高斯朴素贝叶斯	Accuracy	0.7458
	Precision	0.1101
	Recall	0.4622
	F1	0.1778
	AUC	0.6129

模型比较（2）

- 使用AdaBoosting和随机森林进行集成学习。由于集成模型强化了分类错误的样本的权重，使得模型的精度相比基学习器有提升，但是相应的召回率有所降低。

```
from sklearn.ensemble import AdaBoostClassifier
def build_model(x, y):
    classifier = tree.DecisionTreeClassifier(criterion='entropy', max_depth=10)
    classifier = AdaBoostClassifier(base_estimator=classifier, n_estimators=100)
    classifier.fit(x, y)
    return classifier
```

模型	评价指标	测试值
AdaBoost	Accuracy	0.8285
	Precision	0.1387
	Recall	0.3613
	F1	0.2005
	AUC	0.6097

模型比较（3）

- 随机森林使用自助采样法（Bootstrap sampling）随机从训练集中采样出训练数据拟合一个分类模型，重复该操作建立多个分类器，最终使用简单的投票法获得集成模型的最终分类结果。
- 可见随机森林与AdaBoosting算法类似，模型的精度有所提升，但是召回率有所下降？

```
from sklearn.ensemble import RandomForestClassifier
def build_model(x, y):
    classifier = RandomForestClassifier(criterion='entropy', max_depth=10, n_estimators=100)
    classifier.fit(x, y)
    return classifier
```

模型	评价指标	测试值
随机森林	Accuracy	0.8205
	Precision	0.1429
	Recall	0.4034
	F1	0.2110
	AUC	0.6251

总结

- 保险产品推荐是一个二分类问题，因此有相当多的模型可供选择，例如决策树、逻辑回归、神经网络、朴素贝叶斯、AdaBoosting和随机森林等。
- 在实际业务环境中，需要根据业务问题选择合适的模型，并进行调参。模型并非越复杂越好，参数的调整对模型的结果也有非常大的影响，需要数据分析人员耐心实践。