

**2020年数据科学导论期末大作业**



### 姓 名： 钟颖怡 吕文秀

**班 级： 2018211318 2018211318**

**学 号： 2018211601 2018211536**

**日 期： 2020年10月29日**

目录

[一、题目背景 3](#_Toc6055268)

[二、需求说明 3](#_Toc6055280)

[三、作业过程和细节分析 4](#_Toc6055280)

[1、数据预处理 4](#_Toc6055272)

[1）特征提取 4](#_Toc6055272)

[2）处理缺失值 4](#_Toc6055272)

[3）异常值检测和数据离散化 6](#_Toc6055272)

[2、建模过程 7](#_Toc6055272)

[1）学习器模型选择 7](#_Toc6055272)

[2）筛选模型 8](#_Toc6055272)

[3）集成尝试 8](#_Toc6055272)

[4）调参 9](#_Toc6055272)

[5）输出结果尝试 1](#_Toc6055272)0

[6）重新开始数据预处理 1](#_Toc6055272)0

[7）模型选择和参数调整 1](#_Toc6055272)0

[8）输出结果 1](#_Toc6055272)3

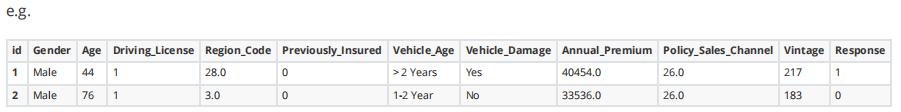
[附录：调试日志 1](#_Toc6055293)3

## 一、题目背景

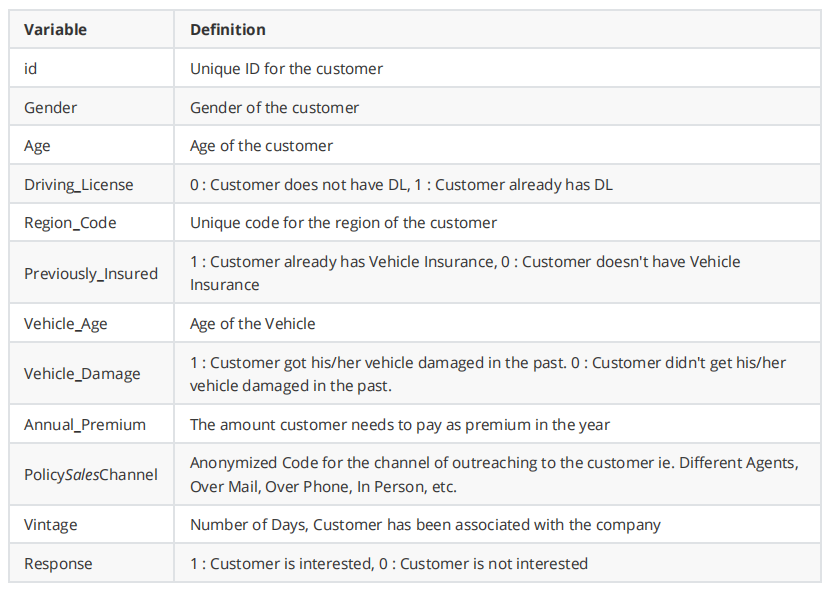
这是一家为客户提供健康保险的保险公司，现在他们需要你的帮助来建立一个模型来预测过去一年的投保人（客户）是否会对公司提供的汽车保险感兴趣

## 二、需求说明

通过已投健康保险的客户的个人以及车辆信息，来预测该客户是否会对公司提供的汽车保险感兴趣



数据字段描述



我们需要通过训练集来训练模型，然后预测测试集中的response，其中训练集包括304888条数据，测试集有76221条数据，我们需要给出测试集中每一个id的response为1的概率，即该客户对公司的汽车保险也感兴趣的概率



## 作业过程和细节分析

1. 数据预处理：

拿到题目，我们先通过观察各个属性和数据，分别把数据分了类，发现在所有属性中，只有Annual\_Premium是连续的，其余属性都是离散的，我们又简单查看了一下缺失值，发现只有Driving\_License和Policy\_Sales\_Channel是包含缺失值的，并且这两个属性的缺失值都占0.28%左右

1. 特征提取：

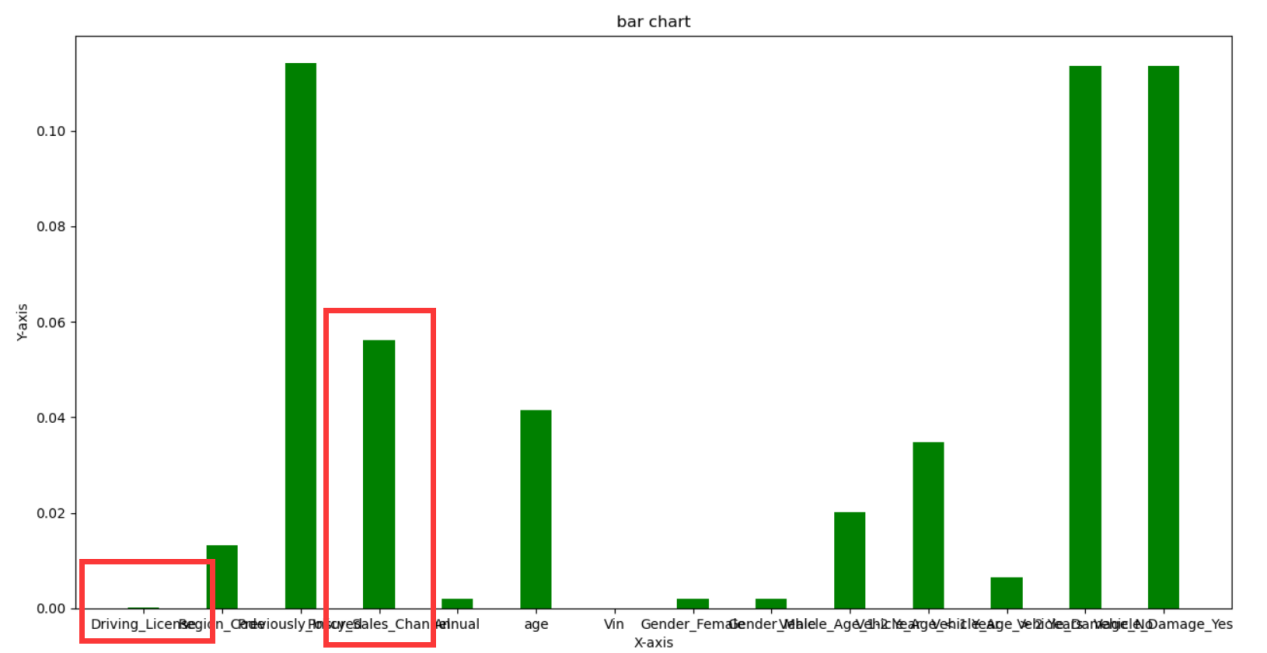
对于Gender，Vehicle\_Age，Vehicle\_Damage这三个属性的参数是不规范的，故使用pandas.get\_dummies函数对这三个属性实现one-hot encode方法

1. 处理缺失值：

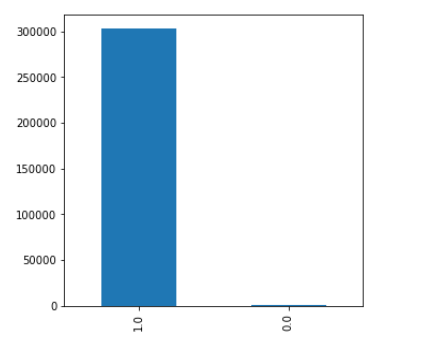
查看了缺失值之后，我们想要知道Driving\_License和Policy\_Sales\_Channel这两个属性和其他属性的相关性如何，因为如果有其他属性和包含缺失值的属性特别相关的话，该缺失属性甚至可以去掉，并且如果找到一些相关性都挺高的其他属性，也可以通过模型预测来填充包含缺失值属性的缺失值，同时，还能知道Driving\_License和Policy\_Sales\_Channel这两个属性和Response的相关性如何，如果相关性特别低，那么就可以直接去掉该属性

我们先使用了信息增益和K方检测两种方法来检测各个属性与Response的相关性：

使用信息增益方法：

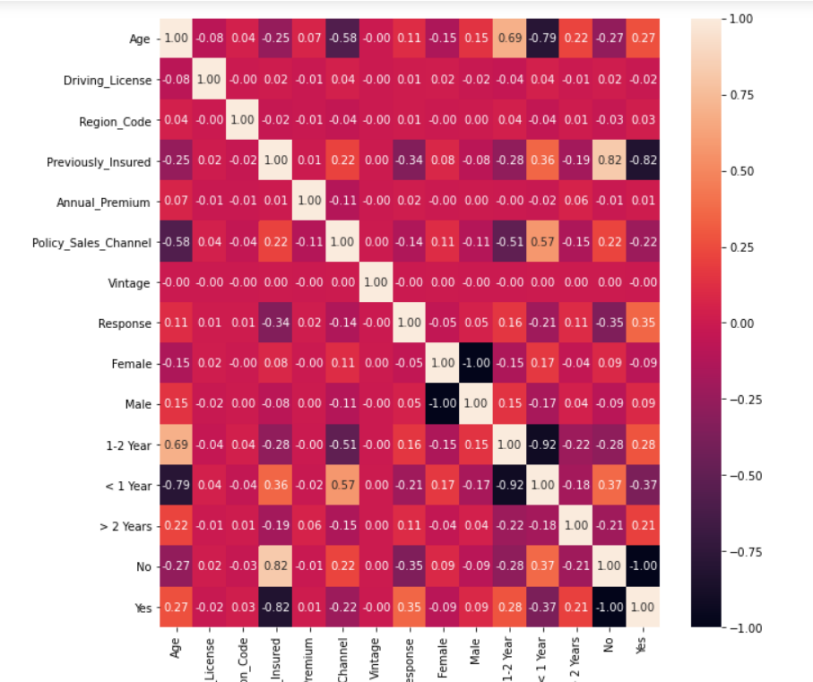


可以看到Policy\_Sales\_Channel属性和Response的相关性一般高，不能删，但Driving\_License和Response的相关性很低，于是我们便猜测Driving\_License属性可以删除，但是根据现实情况分析，我们觉得一个人是否有驾照和他是否会买汽车保险相关性不会很小，所以我们对Driving\_License属性的数据分布进行了检查：



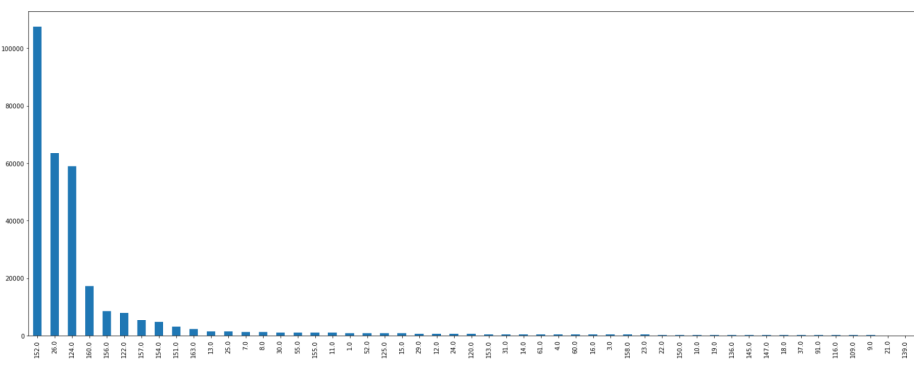
从图中可以发现，Driving\_License属性中1的占比非常大，所以我们有理由怀疑就是因为1的占比太大了所以导致信息增益检验中该属性和Response的相关性不强。

相关性热力图：



发现Driving\_License属性和其他属性并不是完全无关，于是综合讨论 我们决定用众数1来进行Driving\_License属性的缺失值填充。

Policy\_Sales\_Channel，结合实际：和客户的联系方式缺失很可能说明此条数据没有可信性，所以将缺失Policy\_Sales\_Channel的样本去掉。

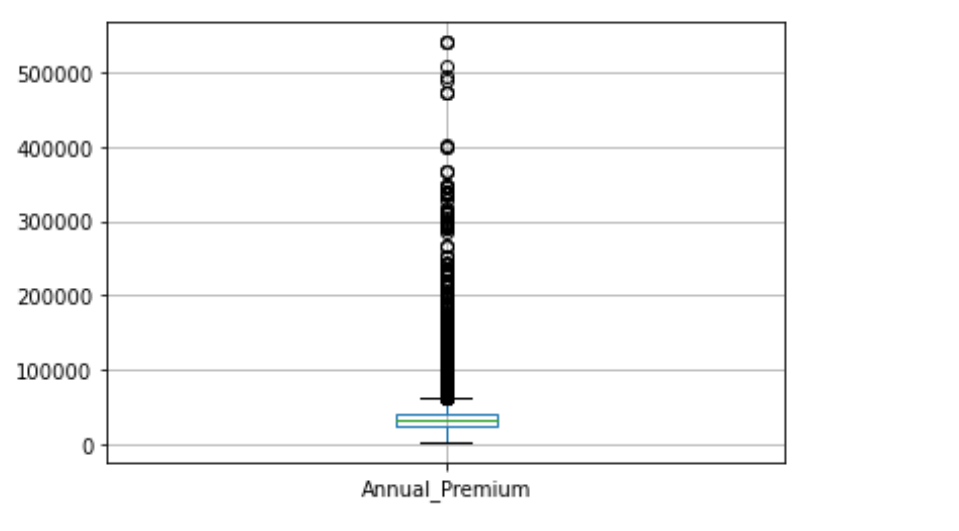


1. 异常值检测和数据离散化：

之所以把这两个步骤合在一起写是因为我们发现异常值和离散化有关。

我们先选出了需要进行离散化的属性：Annual\_Premium，Age，Vintage

刚开始，我们进行的尝试是将Annual\_Premium和Vintage属性进行等频划分，Age属性进行等宽离散化，但是令人惊讶的是，Annual\_Premium和Vintage属性在离散化后出现了很多缺失值，于是我们思考，这些缺失值是由于离散化而来的，那就说明等频的方法不适合这两个属性，我们猜测出现缺失值的原因是这两个属性的数值分布太过分散和极端，于是我们用箱型图来查看了一下，验证了我们的猜想



于是我们就想到了聚类，数值分散但是在每一块又很集中，这不正是聚类吗？于是我们把Annual\_Premium和Vintage属性都进行了KMeans离散化，使用kmeans进行聚类，增加了结果的随机性，可能会出现分类效果很好的模型

Annual\_Premium离散化：

1. **if** i == 'Annual\_Premium':
2. Annual = df[i]
3. Annual\_re = Annual.values.reshape((Annual.index.size, 1))
4. k = 10
5. k\_model = KMeans(n\_clusters = k, n\_jobs = 4)
6. df['Annual'] = k\_model.fit\_predict(Annual\_re)
7. del df[i]

Age等宽离散化：

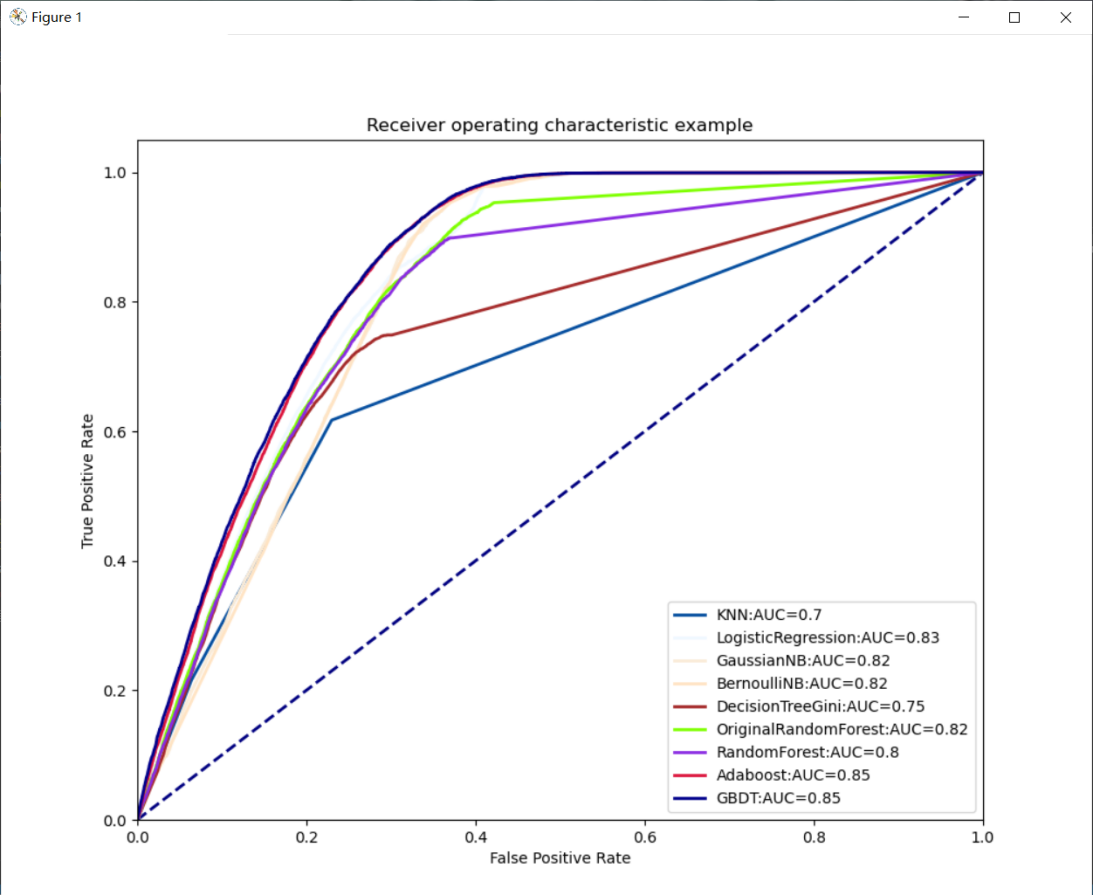
1. elif i == 'Age':
2. age = df[i]
3. df['age'] = pandas.cut(age, 15, labels=range(15))
4. del df[i]

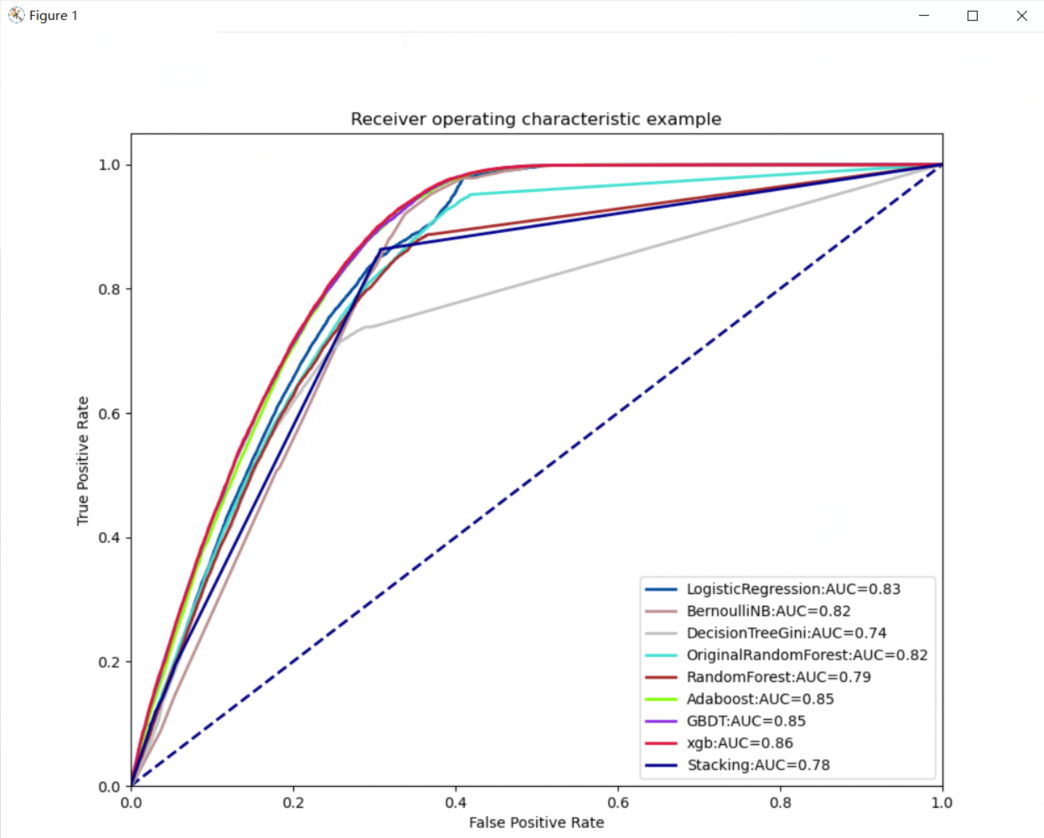
Vintage离散化：

1. elif i == 'Vintage':
2. Vin = df[i]
3. Vin\_re = Vin.values.reshape((Vin.index.size,1))
4. k = 10
5. k\_model = KMeans(n\_clusters = k, n\_jobs = 4)
6. df['Vin'] = k\_model.fit\_predict(Vin\_re)
7. del df[i]
8. 建模过程：
9. 学习器模型选择：把知道的模型都添加进去（不带参数），根据ROC曲线观察选择排名靠前的模型来进行进一步优化

使用的模型如下图代码所示

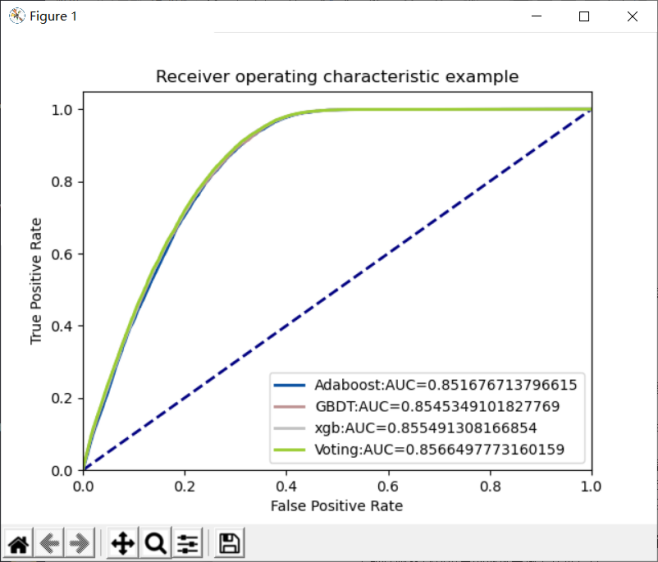
1. from sklearn.metrics import accuracy\_score,recall\_score,f1\_score
2. from sklearn.neighbors import NearestNeighbors,KNeighborsClassifier
3. from sklearn.ensemble import BaggingClassifier
4. from sklearn.naive\_bayes import GaussianNB,BernoulliNB
5. from sklearn.tree import DecisionTreeClassifier
6. from sklearn.svm import SVC
7. from sklearn.ensemble import RandomForestClassifier
8. from sklearn.ensemble import AdaBoostClassifier
9. from sklearn.linear\_model import LogisticRegression
10. from sklearn.ensemble import GradientBoostingClassifier
11. from sklearn.model\_selection import StratifiedKFold
12. from mlxtend.classifier import StackingClassifier
13. from sklearn import model\_selection
14. from sklearn.multiclass import OneVsRestClassifier
15. from xgboost import XGBClassifier
16. from scipy.sparse import hstack
17. from sklearn.preprocessing import OneHotEncoder
18. from sklearn.neural\_network import MLPClassifier
19. from sklearn.model\_selection import GridSearchCV
21. models=[]
22. models.append(('KNN',KNeighborsClassifier(n\_neighbors=3)))     #k近邻算法
23. models.append(('Bagging',BaggingClassifier(KNeighborsClassifier())))
24. models.append(('LogisticRegression',LogisticRegression()))    #logistic回归
25. models.append(('GaussianNB',GaussianNB()))    #朴素贝叶斯
26. models.append(('SVM:',SVC(kernel='linear', probability=True)))     #支持向量机
27. models.append(('BernoulliNB',BernoulliNB()))  #伯努利贝叶斯
28. models.append(('DecisionTreeGini',DecisionTreeClassifier()))      #决策树1
29. models.append(('DecisionTreeEntropy',DecisionTreeClassifier(criterion='entropy')))    #决策树2
30. models.append(('OriginalRandomForest',RandomForestClassifier()))
31. models.append(('RandomForest',RandomForestClassifier(n\_estimators=11,max\_features=None)))
32. models.append(('Adaboost',AdaBoostClassifier(n\_estimators=100)))
33. models.append(('GBDT',GradientBoostingClassifier()))
34. models.append(('xgb',XGBClassifier()))
35. models.append(('MLP',MLPClassifier(solver='adam', alpha=1e-5, hidden\_layer\_sizes=(100,5), random\_state=1)))
36. 筛选模型：各种模型得出的结果如下图





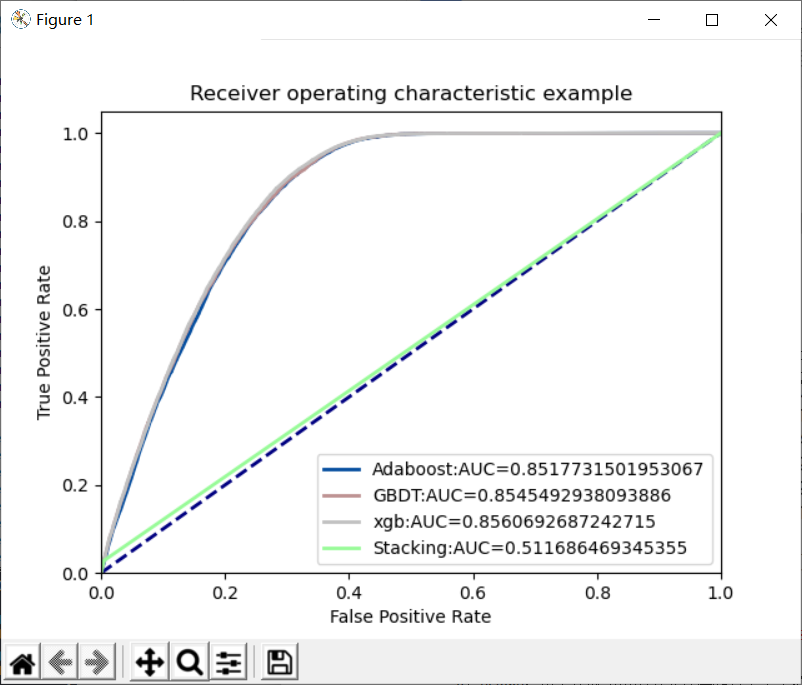
从图中可以看出，对于这些学习器，GBDT、adaboost和xgboost是相对来说结果较优的三个学习器，故选择这三种模型进一步优化

1. 集成尝试：对于GBDT、adaboost和xgboost这三种学习器，尝试通过集成算法进一步提高准确率
   1. 投票器：from sklearn.ensemble import VotingClassifier



从图中可以发现，用投票器AUC面积变高了一点（0.005的提升），ACC也提高了一点点（0.0003的提升）

* 1. Stacking:令人惊讶的是Stacking集成方法似乎并不适合这样做，或者我们技术有限不能很好的利用stacking来提高准确率，故我们决定用投票器来进行集成提升



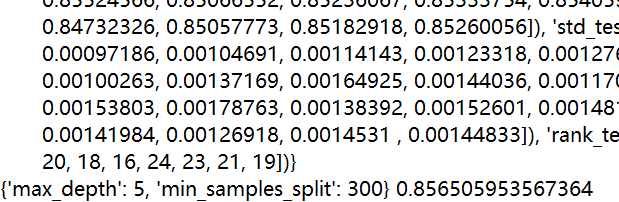
1. 调参：我们通过查阅相关资料，了解到一个自动调参的函数gridSearchCV
   1. 将步长初始值设置为0.1。对于迭代次数进行网格搜索。

b. 找到一个合适的迭代次数，对决策树进行调参。首先对决策树最大深度max\_depth和内部节点再划分所需最小样本数min\_samples\_split进行网格搜索。

c. 再对内部节点再划分所需最小样本数min\_samples\_split和叶子节点最少样本数min\_samples\_leaf一起调参。

d. 再对最大特征数max\_features进行网格搜索。  
e. 对子采样的比例进行网格搜索。

f. 减半步长，最大迭代次数加倍来增加我们模型的泛化能力。



1. 输出结果尝试：

我们在第一次进行到这一步时发现了一个严重的问题，所有的样本是一起做标准化和离散化的处理的，然后才随机分层抽样训练集和测试，这样会影响泛化能力的判断。所以先分层随机抽样分为训练集和测试集，再对训练集和测试集分别进行同样的预处理。

由于我们在数据预处理时对某些属性进行的是kMeans聚类离散化方法，所以当我们输入测试集真正想要预测结果时，对测试集的该属性离散化就会变得很困难，所以在重新进行数据预处理时，记录下kmeans聚类的质心。

1. 重新开始进行数据预处理：



Driving\_License和Policy\_Sales\_Channel预处理方法不变；

Age, Region\_Code, Policy\_Sales\_Channel, Previously\_Insured不作处理，都当做离散值处理；

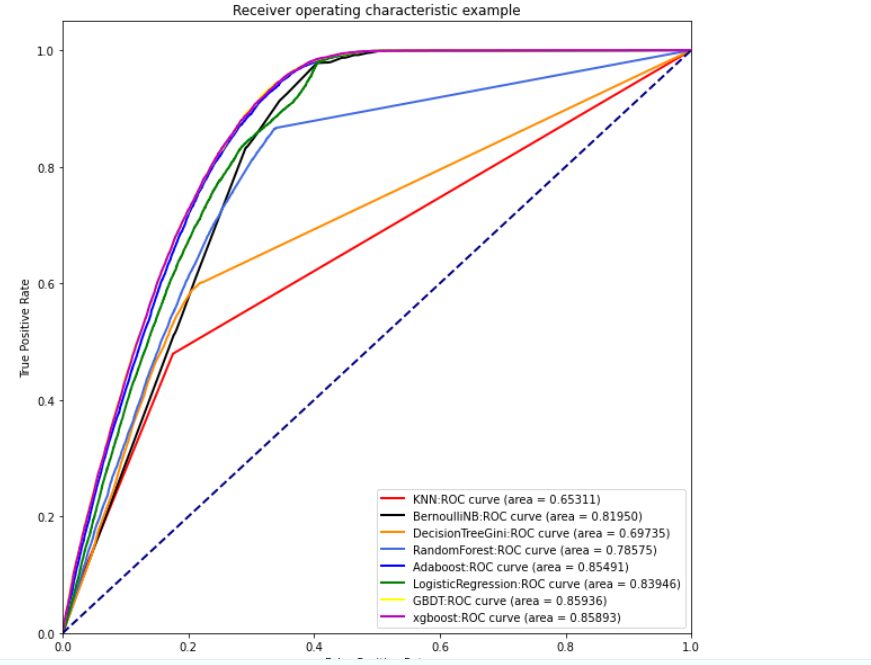
删去vintage属性；

对Annual\_Premium进行kmeans聚类。

测试集预处理方法与训练集相同，Annual\_Premium的聚类使用训练集kmeans聚类的结果。

1. 模型选择和参数调整：

筛选模型：各种模型得出的结果如下图



对于这些学习器，GBDT和xgboost是相对来说结果较优的两个学习器，故选择这两种模型进一步优化

使用的分类模型：GradientBoostingClassifier

调用GridSearchCV

经过网格搜索法调参：

learning\_rate = 0.05

min\_samples\_leaf = 70

max\_features = 12

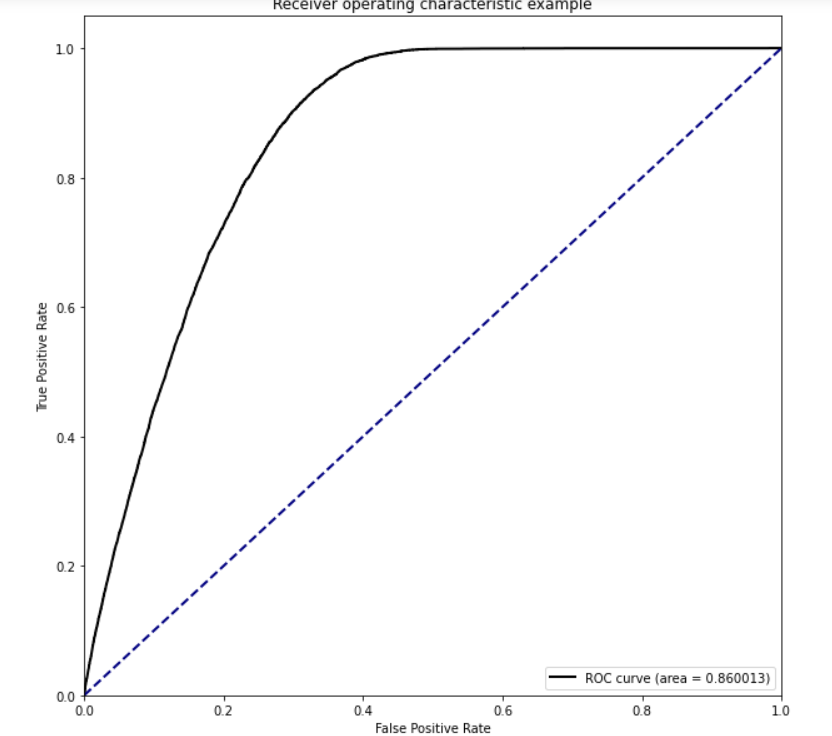
subsample= 0.8

random\_state=10

max\_depth=5

n\_estimators=300

min\_samples\_split=1400



使用的分类模型：XGBClassifier

调用GridSearchCV

经过网格搜索法调参：

learning\_rate=0.4,

n\_estimators=30,

max\_depth=5,

min\_child\_weight = 1,

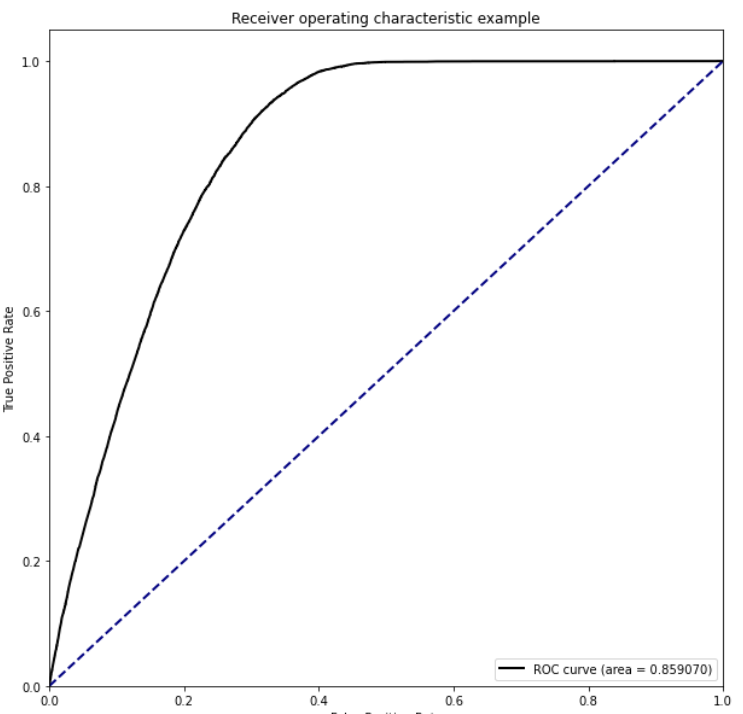
gamma=0，

subsample=0.9,

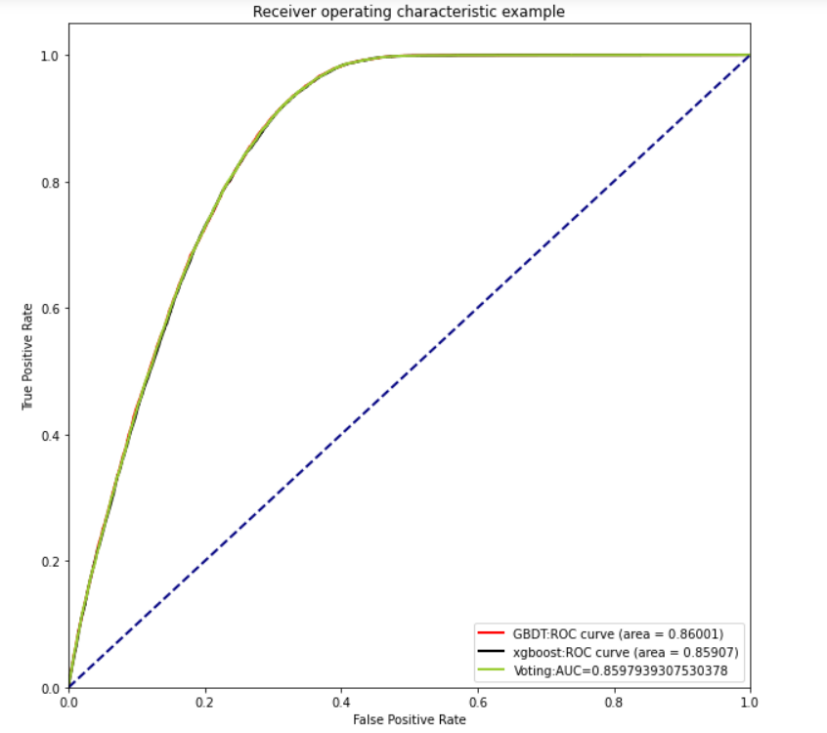
colsample\_btree=0.8,

scale\_pos\_weight=1

random\_state=20

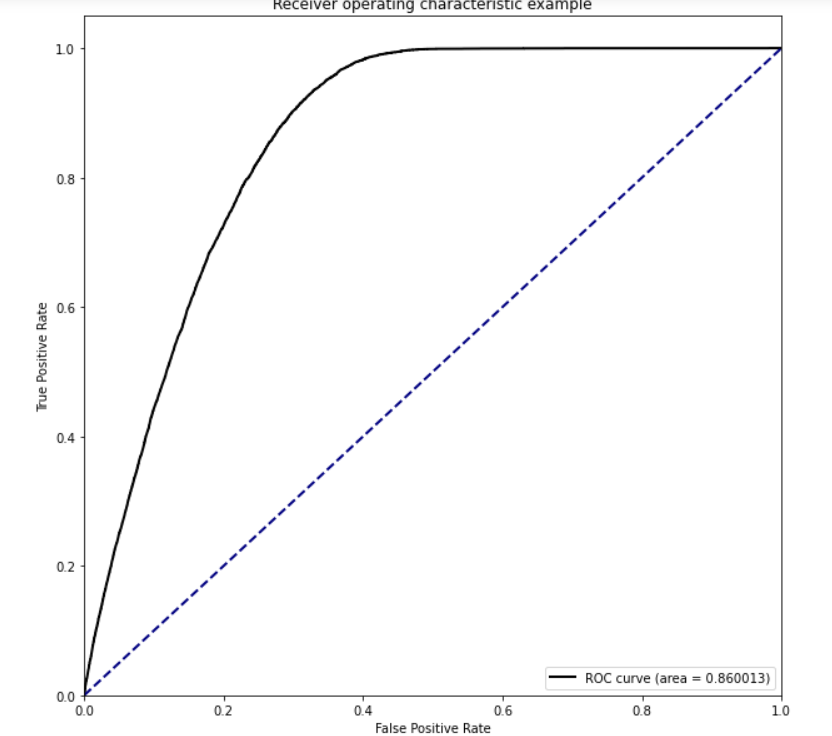


使用投票器：VotingClassifier



但是发现投票器集成的结果没有单独的GDBT模型好，所以最终决定单独使用GDBT模型

1. 输出结果：



# 附录：调试日志

**10月22日**

**集中讨论，观察数据集的属性，并对题目进行分析，完成了对特征间的相关性分析和缺失值处理**

|  |  |
| --- | --- |
| **问题** | **解决方案** |
| 对于缺失值的处理，我们想先查看含有缺失值的属性和Response与其他特征之间的相关性，但是想要检测相关性又不能有缺失值，学习了信息增益法和k方检验法以及数据可视化方法查看数据间相关性 | 先把缺失值全部删除，只是单纯用来看相关性的，在数据集的处理上不能全部删除 |

**10月23日**

|  |  |
| --- | --- |
| **问题** | **解决方案** |
| 对于Annual\_Premium和Vintage的离散化，使用等频划分会造成出现很多的缺失值，不知道怎么处理连续数据合适。 | 用kMeans聚类进行离散化，不出现一个缺失值；  对数据进行标准化，等宽聚类，等频聚类，kmeans聚类等分别查看模型拟合效果。 |

**10月24日**

|  |  |
| --- | --- |
| **问题** | **解决方案** |
| 对于Annual\_Premium和Vintage的离散化，使用等频划分会造成出现很多的缺失值 | 用kMeans聚类进行离散化，不出现一个缺失值 |

**10月25日**

|  |  |
| --- | --- |
| **问题** | **解决方案** |
| 在完成了用单个学习器学习并且得出结果后，想要通过一些集成学习方法来进行提升，但是集成的结果比原来的单个学习器低；  对GDBT和xgboost进行参数调优。 | 在把一些准确率较低的学习器删掉，只留下AUC较高的三种模型进行投票集成，结果有小幅度的提升；  调用库进行网格搜索（搜索很慢的同时进行手动调参） |

**10月26日**

|  |  |
| --- | --- |
| **问题** | **解决方案** |
| 在导出数据时发现，训练集kMeans离散化对于测试集不好分类 | 把train中的数据提前按照7.5：2.5的比例划分好，用那7.5的那部分数据进行kMeans聚类，然后那2.5的那部分数据用来验证，这样就能保证那2.5的数据和我们将要预测的那部分test是同一类型的数据 |

**10月27日**

|  |  |
| --- | --- |
| **问题** | **解决方案** |
| 重新进行数据预处理和模型构造 | GDBT和xgboost参数调优，投票器投票 |