# 数据挖掘在收入预测中的应用

贾文鑫 51184407128
2019 年 1 月 10 日

# 摘要

本文试图研究人们特别关注的收入预测问题，研究哪些变量对收入的影响较大，种族歧视和性别歧视在收入方面的表现。利用本学期学习的 10 种分类方法，对美国居民收入数据进行预测分类。首先预估模型，根据准确率、kappa 值和运行时间选取合适的模型进一步优化参数，随后将优化之后的模型应用到测试集中，根据准确率、AUC 值等评价指标评估模型的优良性，最后根据这些模型在测试集中得到的结果，进一步分析各类变量对收入的影响。得到结论，随机森林方法略微优于其他分类方法，各类方法大致处于同一水平，变量方面，人际关系和受教育程度对收入的影响最大，而性别和种族并没有显著的重要性。

本文包含 11 张图形，11 张表格。

**关键词**：收入；随机森林；判别分类；预测；模型评估

# 目录

# 第一章 引言

## 1.1 研究背景

收入一直是人们日常最关心的几个问题之一，生活中也经常能看到家长向刚工作的哥哥姐姐们询问工资情况，大多数同学选择来上海就读研究生生也是因为上海收入相对较高。那么收入到底和那些因素有关呢？如果能够研究得到收入与哪些因素是相关的，是否可以通过影响这些因素来达到提高自己收入的目的呢？

## 1.2 问题提出

本文所选取的数据中包含有年龄、受教育程度、工作阶层和种类、人际关系、种族、性别等十多个变量，来探究美国居民收入与哪些因素有关。通过采用本门课程所学习的 10 种分类方法，采用已知的信息，在训练集中训练模型，在测试集中预测收入是否大于 50k。通过对各类分类方法评价指标的分析，来挑选最优的估计模型。从直观上来看，受教育程度高的人理应要比没受过太多教育的人收入更高，不同职业之间应该也会有收入差异，而且种族与性别之间的差异也是我们所关心的，是否真的存在性别歧视和种族歧视导致男性比女性的收入高，白种人比有色人种收入高。

## 1.3 论文结构安排

本文结合本学期学习的各种分类方法，对美国居民收入进行预测，并且选取适当的分类评价指标，对模型预测效果进行评估。本文尝试对 1 万条数据进行训练，寻找最优预估的同时，测试各算法的运行速度，很好的模拟了现实操作。

本文共分为三个部分，结构安排如下：

第一章介绍了选题背景，收入的多少是人们十分关心的问题，那么是什么因素影响了收入自然而然地引起了人们的思考，本文选取数种分类方法来对收入进行预测。

第二章是本文的重点，主要是各种分类方法的实证分析，采用不同的分类器，

结合准确率、kappa 值、AUC 值等指标，评价各个模型的预估效果。并采用效果较好的分类方法在测试集中预测，并对这些分类模型进行评估。

第三章主要结合第二章模型的结果，就此次研究中所总结的经验教训予以简要陈述。

# 第二章 数据分析

## 2.1 数据来源与预处理

本文的数据来源于 http://archive.ics.uci.edu/ml/的 adult 数据集，共包含 16281 个数据，每一条数据包含 14 个因变量和 1 定型的自变量。

### 2.1.1 变量的介绍

每一条数据包含了 14 个因变量，这 14 个因变量既有连续变量又有分类变量，我们需要使用这 14 个变量去预测这个人的收入是否大于 50k。

表 1 变量名解释

| 变量名 | 变量解释 |
|---|---|
| age | 连续变量，被观测者年龄 |
| workclass | 分类变量，工作阶层 |
| fnlwgt | 连续变量，权重 |
| education | 分类变量，受教育程度 |
| education.num | 连续变量，受教育年限 |
| marital-status | 分类变量，婚姻状况 |
| occupation | 分类变量，职业 |
| relationship | 分类变量，家庭关系 |
| race | 分类变量，种族 |
| sex | 分类变量，性别 |
| capital-gain | 连续变量，资本收益 |
| capital-loss | 连续变量，资本亏损 |
| hours-per-week | 连续变量，每周工作时间 |
| native-country | 分类变量，原生国家 |

### 2.1.2 变量的初步筛选

观察收入数据的前几项可以看出，capital-gain 和 capital-loss 两项的值绝大多数取值为 0，而且从实际上来说，如果一个人资本收益较大，那么基本可以肯定他的收入大于 50k，这些变量对预测没有意义，所以选择将这两个变量去除。

### 2.1.3 训练集与测试集的构造

设立随机数种子为 1，在总数据中随机抽取 2/3 作为训练集，其余作为测试集。缺失值处理和归一化将在建立每一个模型时进行。

## 2.2 模型初步选择

首先针对每种方法都建立一个模型，观察它们在训练集中的效果，尽量采取交叉验证的方法获取模型在训练集上的正确率和 kappa 值，并选取其中效果最好的 5 个模型进行参数优化。

其中，KNN 方法和 SVM 方法需要对变量进行归一化处理，且有些方法不能处理数据的缺失值，需要提前对数据进行处理，将连续性变量归一化处理，或是将缺失数据所在的行清除。

表 2 模型预估效果

| 分类器 | 正确率 | kappa 值 | 运行时间 |
|---|---|---|---|
| naive bayes | 0.826494 | 0.4992671 | <1min |
| KNN | 0.8300797 | 0.5042374 | 5min |
| 决策树 | 0.8217131 | 0.4340324 | <1min |
| bagging | 0.8226096 | 0.4275556 | <1min |
| 随机森林 | 0.8356574 | 0.5266836 | 2min |
| adaboost | 0.8358566 | 0.5320491 | <1min |
| staking | 0.8311533 | 0.5010647 | 5min |
| SVM | 0.827988 | | 2min |
| xgboost | 0.821414 | | 2min |
| 神经网络 | 0.813526 | 0.4265689 | 20min |

正确率表示模型将训练局中的数据正确分类的比例，kappa 值表示相对于乱猜，模型正确分类的提高值 k=（p0-pe）/（1-pe）。

从上表可以看出，初步建立的模型中，神经网络不仅正确率和 kappa 值最低，而且运行时间也是最长的，所以选择将其去除。剩下的变量中，根据其模型效果和运行时间选择 KNN、bagging、随机森林、adaboost 和 SVM 这 5 种综合效果最好的 5 种方法进行下一步参数优化。

## 2.3 模型参数优化

### 2.3.1 KNN 的参数优化

KNN 方法使用前需要对数据进行归一化处理。选取欧式距离，最大的邻居数设为 100，在"rectangular", "triangular", "epanechnikov", "optimal", "cos", "inv", "gaussian","triweight", "biweight"这些核函数中选取最优的 k 和 kernel。

最终输出最优参数 k=66，kernel="inv"，模型的准确率也提升到了 0.832。

### 2.3.2 Bagging 的参数优化

Bagging 方法不需要对数据标准化处理，也能够处理数据的缺失值。设置 minsplit=1,maxdepth=10，即最小分裂 case 为 1，最大树高度为 10，利用 10 折交叉验证寻找最优的参数 m.

最终输出最优参数 m=4，模型在训练集的准确率也有显著提升。

### 2.3.3 随机森林的参数优化

随机森林方法使用前需要去除数据的缺失值。设置 minsplit=1,maxdepth=10，即最小分裂 case 为 1，最大树高度为 10，ntree=500，寻找最优的 ntree 和 mtry 值使得训练集上的错误率最小。

**树的数量与错误率的关系：**

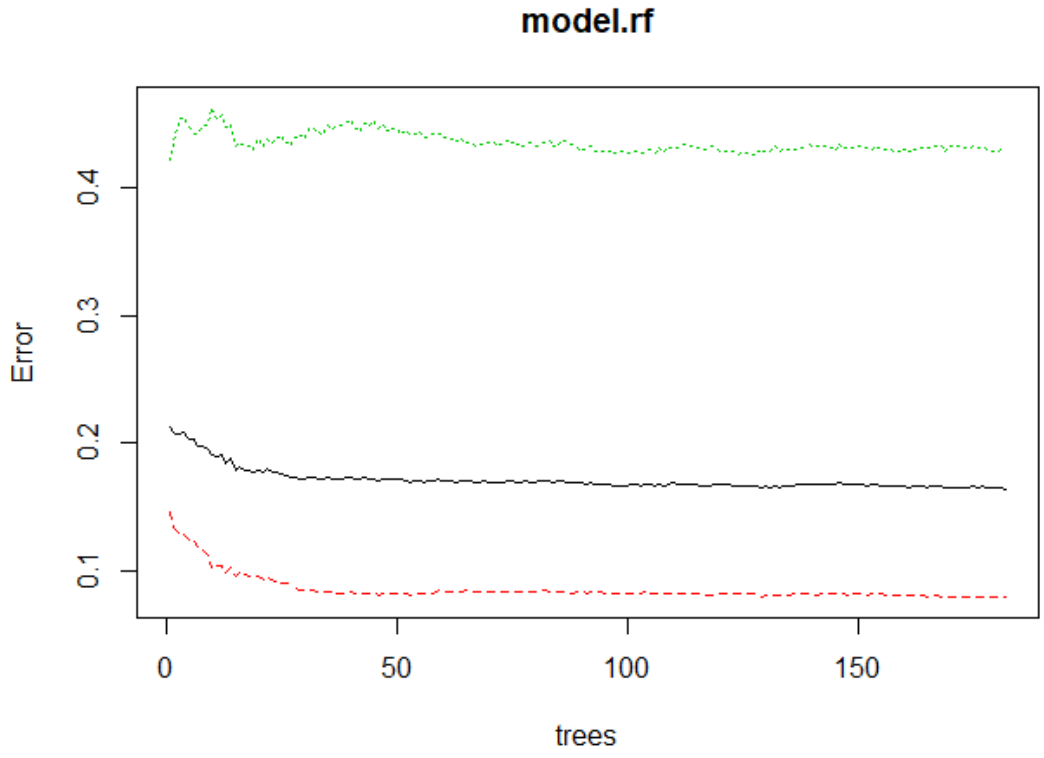



图 1 随机森林数的数量与错误率关系图

最终输出最优参数 ntree=182,mtry=2，模型的错误率降低到了到了 0.163。

### 2.3.4 Adaboost 的参数优化

Adaboost 方法同样使用前需要去除数据的缺失值。设置 minsplit=1,maxdepth=10，即最小分裂 case 为 1，最大树高度为 10，利用 2 折交叉验证，在"Breiman","Freund","Zhu"寻找最优的 coeflearn 参数，在 1：10 中寻找

最优的 mfinal 值。

**错误率与不同参数之间的关系：**

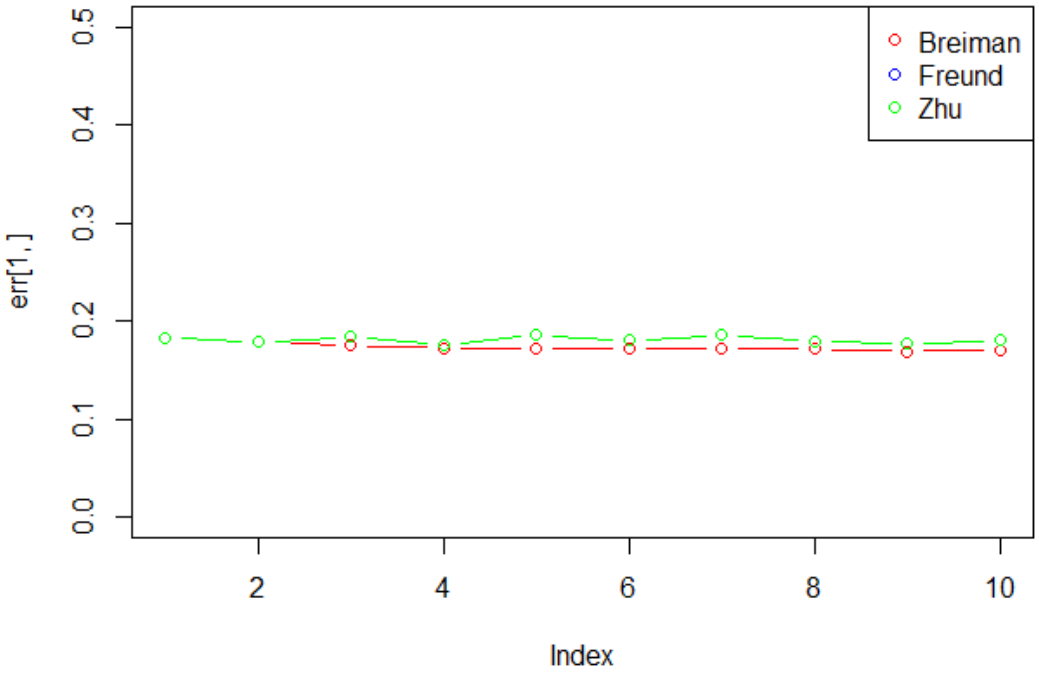



图 2 adaboost 错误率关系图

最终输出最优参数 coeflearn="Breiman",mfinal=9。

### 2.3.5 SVM 的参数优化

SVM 方法需要对数据标准化处理，使用前需要去除数据的缺失值。采用 2 折交叉验证，在-5：15 中以 2 为间隔寻找最优 cost 参数，在-15：3 中以 2 为间隔寻找最优 gamma 参数，使得正确率最高。

最终输出最优参数 cost=2**1,gamma=2**(-3)，共 4256 个支撑向量。

## 2.4 模型效果评估

表 3 模型效果对比

| 分类器 | 正确率 | kappa 值 | Precision | Recall | AUC |
|---|---|---|---|---|---|
| KNN | 0.8270916 | 0.5031449 | 0.6856287 | 0.5540323 | 0.87727 |
| bagging | 0.8181275 | 0.412946 | 0.76243981 | 0.38306452 | 0.74958 |
| 随机森林 | 0.8290837 | 0.51057 | 0.6883629 | 0.5629032 | 0.74000 |
| adaboost | 0.8219124 | 0.4917947 | 0.668616 | 0.5532258 | 0.87324 |
| SVM | 0.8306773 | 0.5128840 | 0.6953908 | 0.5596774 | 0.7400 |

上表给出了 5 个分类器在测试集中的效果，其中精准率

Precision=TP/(TP+FP)，召回率 Recall= TP/(TP+FN)，AUC 表示 ROC 曲线在 x 轴方向所围的面积。

从正确率来看，SVM 方法是最优的；从 kappa 值来看，SVM 方法是最优的；从 Precision 来看，bagging 方法是最优的；从召回率 recall 来看，随机森林方法是最优的；从 AUC 值来看，KNN 方法是最优的。

下面分别分析每个模型在测试集中的预测结果

### 2.4.1 KNN 模型在测试集中的预测结果

**混淆矩阵：**

表 4 knn 混淆矩阵

| Confusion Matrix and Statistics | | | |
|---|---|---|---|
| | | Reference | |
| Prediction | | 0 | 1 |
| | 0 | 3465 | 553 |
| | 1 | 315 | 687 |

**图形分析：**

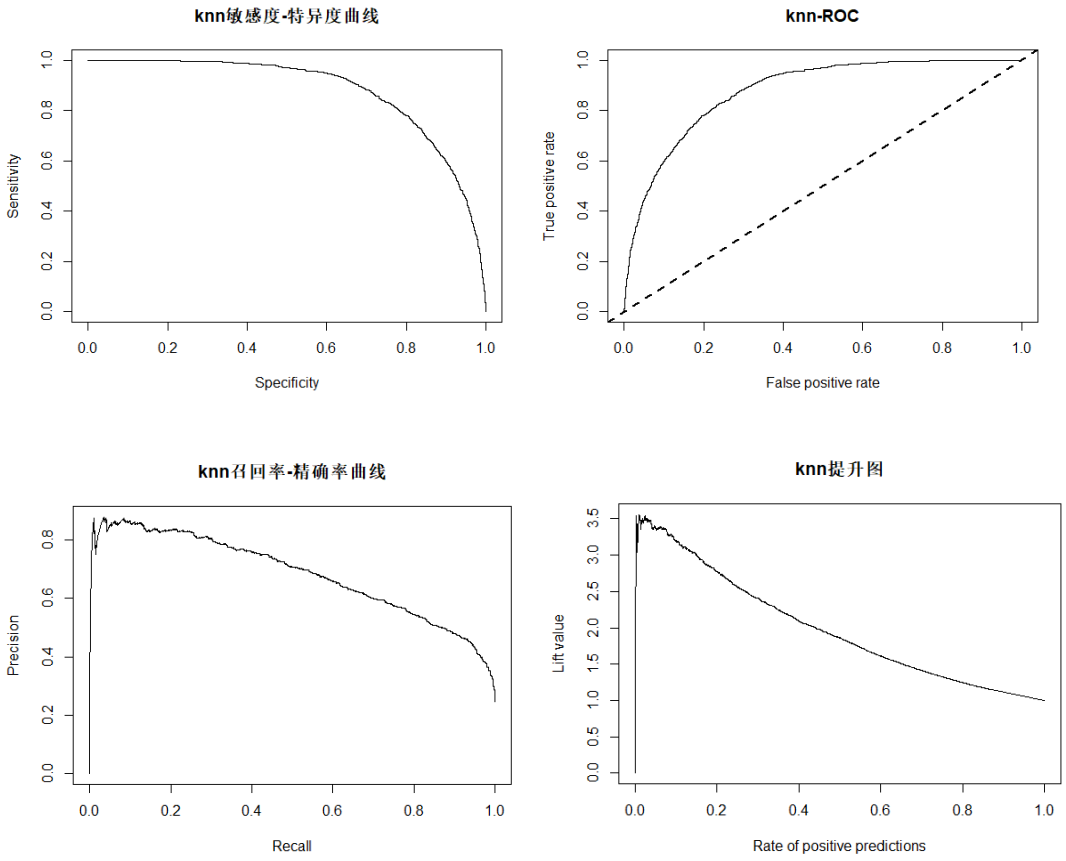



图 3 knn 图形分析

### 2.4.2 Bagging 模型在测试集中的预测结果

**混淆矩阵：**

表 5 bagging 混淆矩阵

| Confusion Matrix and Statistics | | | |
|---|---|---|---|
| | | Reference | |
| Prediction | | 0 | 1 |
| | 0 | 3632 | 765 |
| | 1 | 148 | 475 |

**变量重要性：**

表 6 knn 变量重要性

| 变量名 | 相对重要性 |
|---|---|
| age | 1.7746296 |
| workclass | 0 |
| fnlwgt | 0 |
| education | 12.2525035 |
| education.num | 13.4421459 |
| marital-status | 0 |
| occupation | 4.6432893 |
| relationship | 67.3491905 |
| race | 0 |
| sex | 0 |
| hours-per-week | 0.5382411 |
| native-country | 0 |

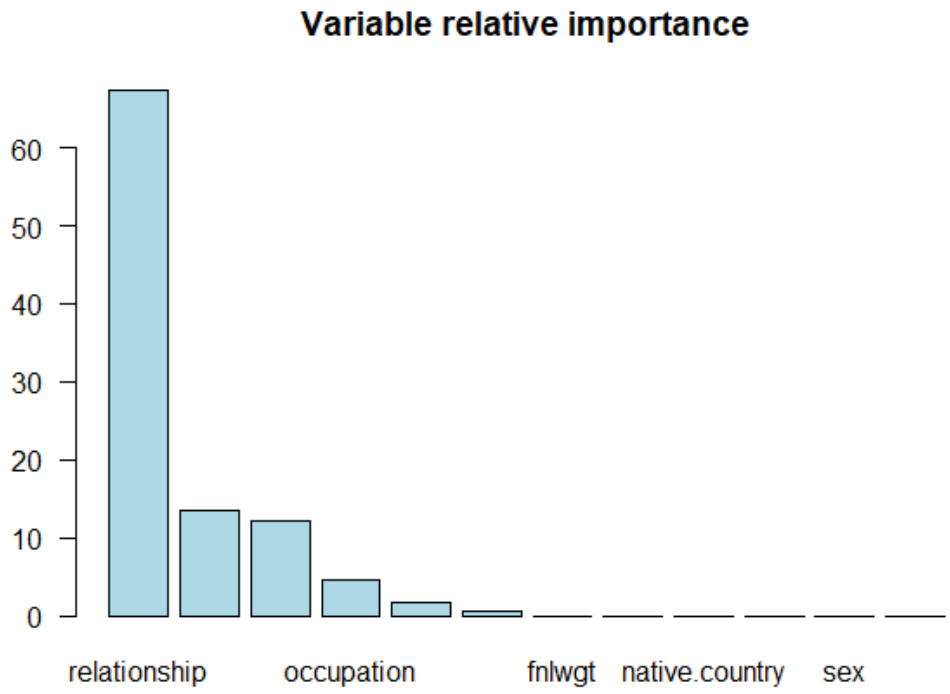
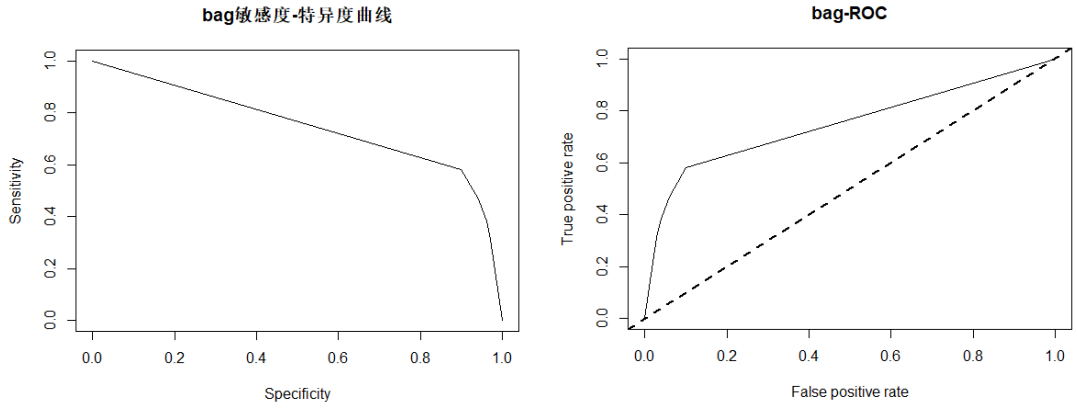



图 4 bagging 变量重要性柱状图

从变量重要性表中可以看出，relationship 是影响最重要的变量，说明良好的人际关系与收入是息息相关的，其次种族、性别、出生国家这几个变量的重要性为零，可见在美国，种族歧视、性别歧视的问题已经得到了有效的控制。

**图形分析：**



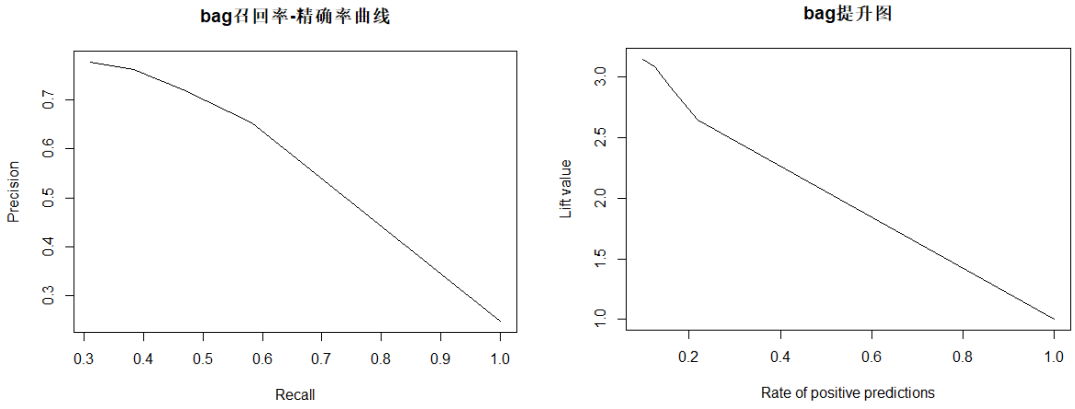



图 5 bagging 图形分析

### 2.4.3 随机森林模型在测试集中的预测结果

**混淆矩阵：**

表 7 随机森林混淆矩阵

| Confusion Matrix and Statistics | | | |
|---|---|---|---|
| | | Reference | |
| Prediction | | 0 | 1 |
| | 0 | 3464 | 542 |
| | 1 | 316 | 698 |

**变量相对重要性：**

表 8 随机森林变量重要性

| 变量名 | 相对重要性 |
|---|---|
| age | 0.0012837817，0.0680500576 |
| workclass | 0.0046625293，0.0038515196 |
| fnlwgt | 0.0019635672，0.0010249972 |
| education | 0.0276295398，0.0130539744 |
| education.num | 0.0322782584，0.0242303658 |
| marital-status | 0.0280280698，0.0786140828 |
| occupation | 0.01075126 ，0.0586479091 |
| relationship | 0.0209057694，0.0661022138 |
| race | 0.0005594742，-0.0001675230 |
| sex | 0.0081556166，-0.0018544703 |
| hours-per-week | 0.0009276968，0.0335627091 |
| native-country | 0.0015263009，0.0007588691 |

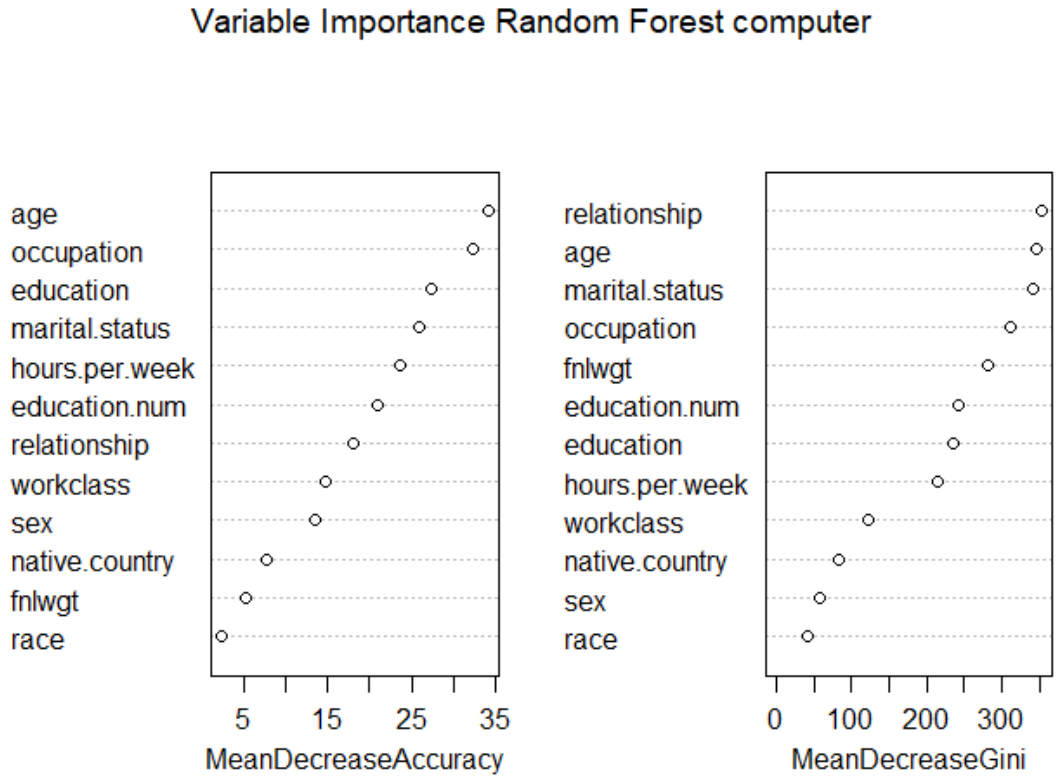



图 6 随机森林变量重要性图

ROC 曲线图：

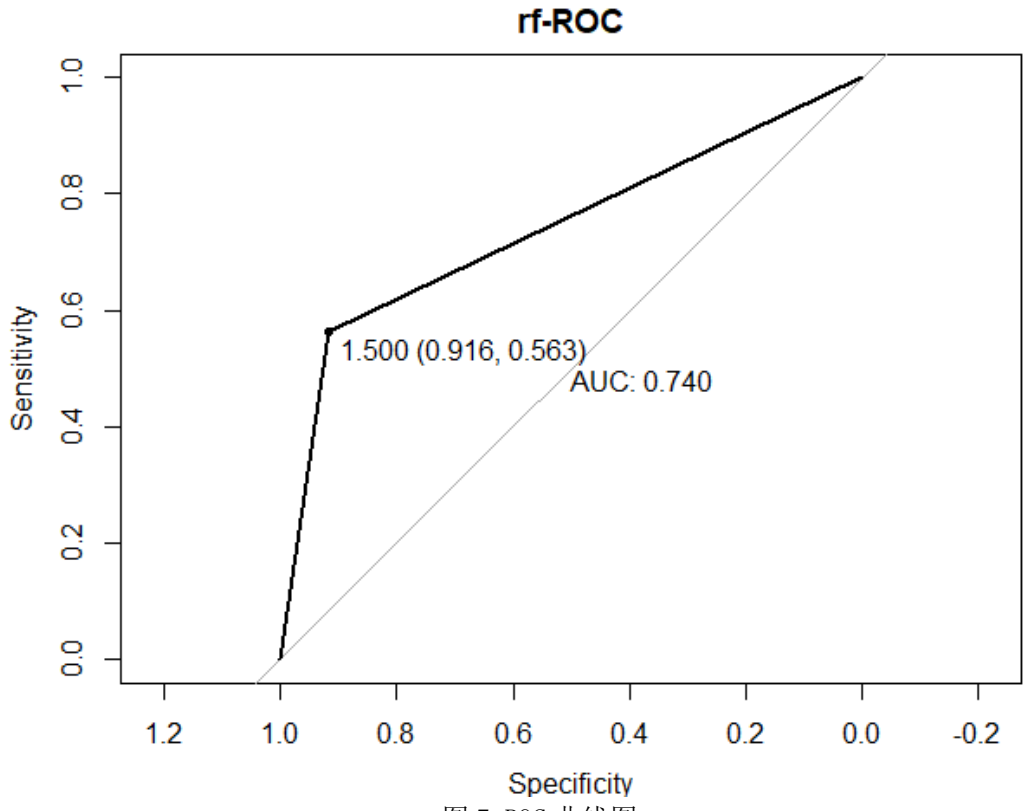



图 7 ROC 曲线图

与 bagging 的变量相对重要性表中得到的结果类似，人际关系、受教育程度是影响收入的最大因素，这也说明了一点，美国的阶级已经发生固化，仅仅依靠上层社会良好的人际关系就能够达到高收入。

### 2.4.4 adaboost 模型在测试集中的预测结果

**混淆矩阵：**

表 9 adaboost 混淆矩阵

| Confusion Matrix and Statistics | | | |
|---|---|---|---|
| | | Reference | |
| Prediction | | 0 | 1 |
| | 0 | 3440 | 554 |
| | 1 | 340 | 686 |

**变量相对重要性：**

表 10 adaboost 变量重要性

| 变量名 | 相对重要性 |
|---|---|
| age | 12.5677993 |
| workclass | 0.3916313 |
| fnlwgt | 0.2256467 |
| education | 10.5684174 |
| education.num | 7.4401988 |
| marital-status | 12.8484901 |
| occupation | 5.6381293 |
| relationship | 47.3729151 |
| race | 0 |
| sex | 0 |
| hours-per-week | 2.2848712 |
| native-country | 0.6619008 |

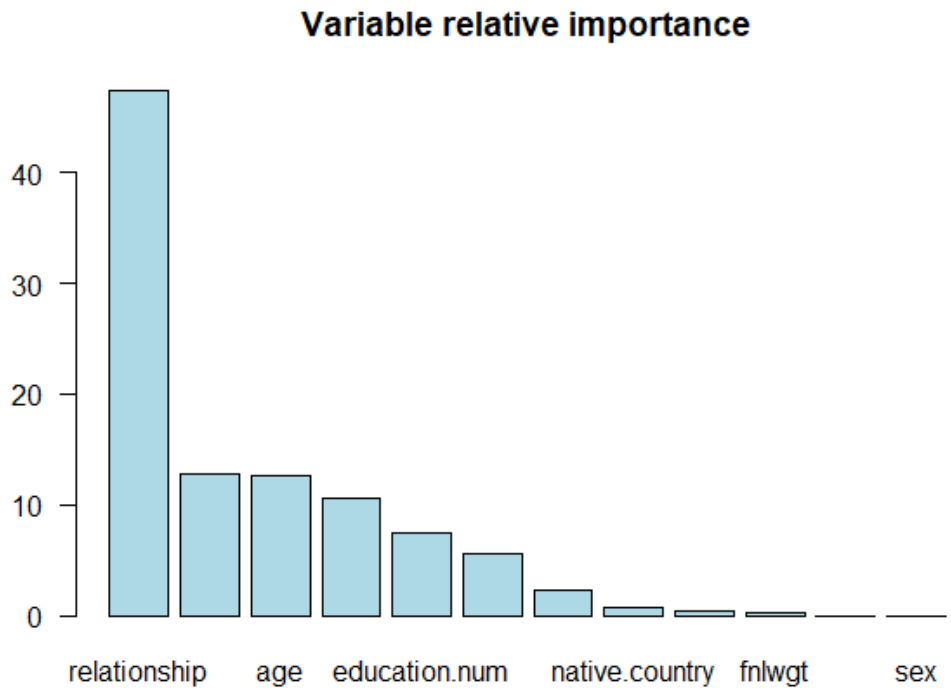
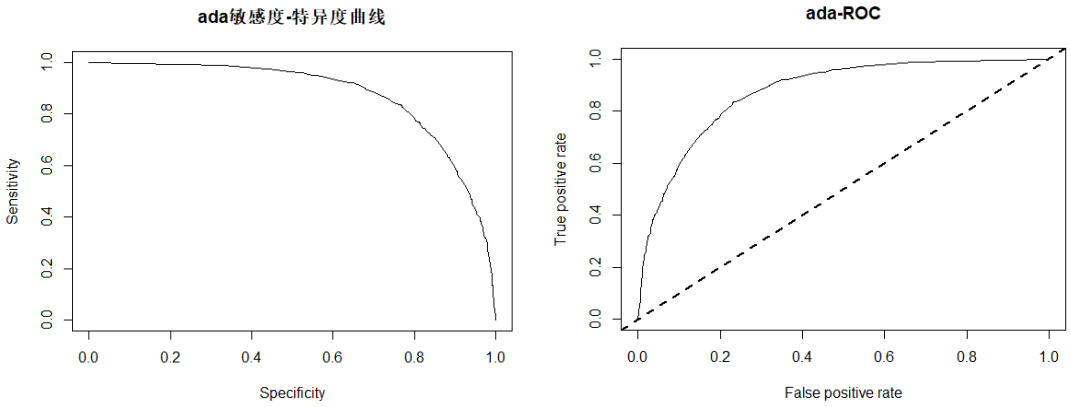



图 8 adaboost 变量重要性柱状图

与前两张变量相对重要性表所得结果类似，但值得注意的是，相对于 bagging 模型的相对重要性表，adaboost 的变量相对重要性略平均，但是种族和性别变量相对重要性依然是 0，再次验证了我们之前的观点。

**图形分析：**



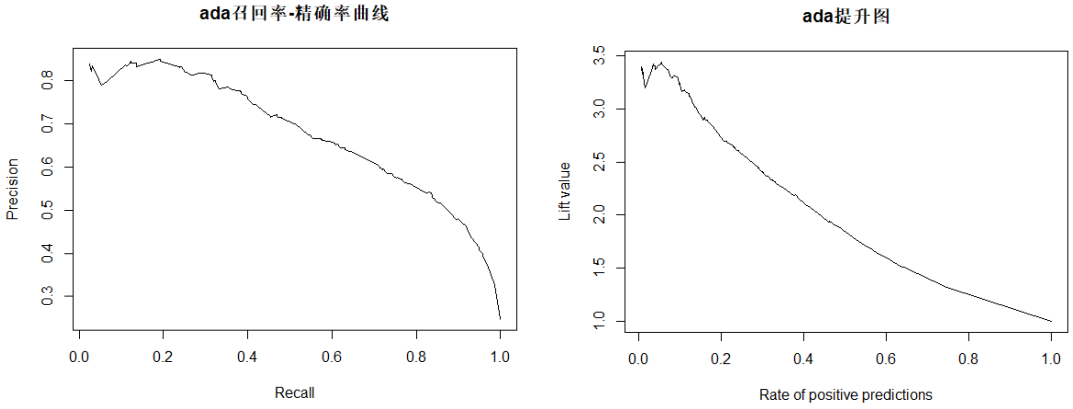



图 9 adaboost 图形分析

### 2.4.5 SVM 模型在测试集中的预测结果

**混淆矩阵：**

表 11 SVM 混淆矩阵

| Confusion Matrix and Statistics | | | |
|---|---|---|---|
| | | Reference | |
| Prediction | | 0 | 1 |
| | 0 | 3476 | 546 |
| | 1 | 304 | 694 |

**图形查看支持向量：**

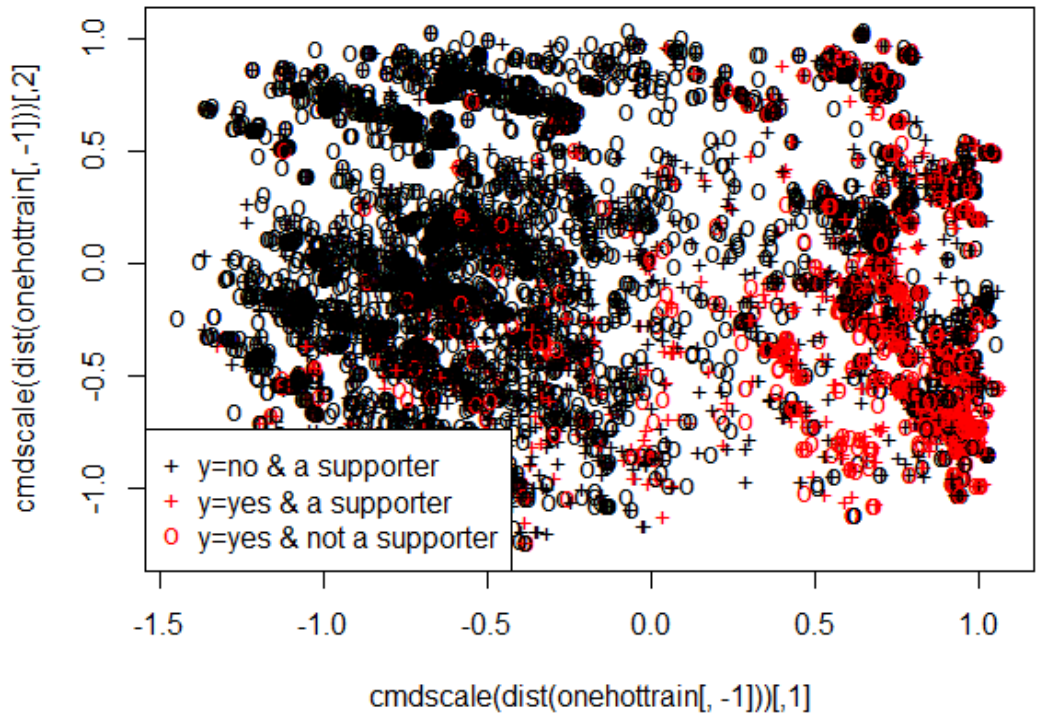



图 10 SVM 支撑向量图

ROC 曲线图：

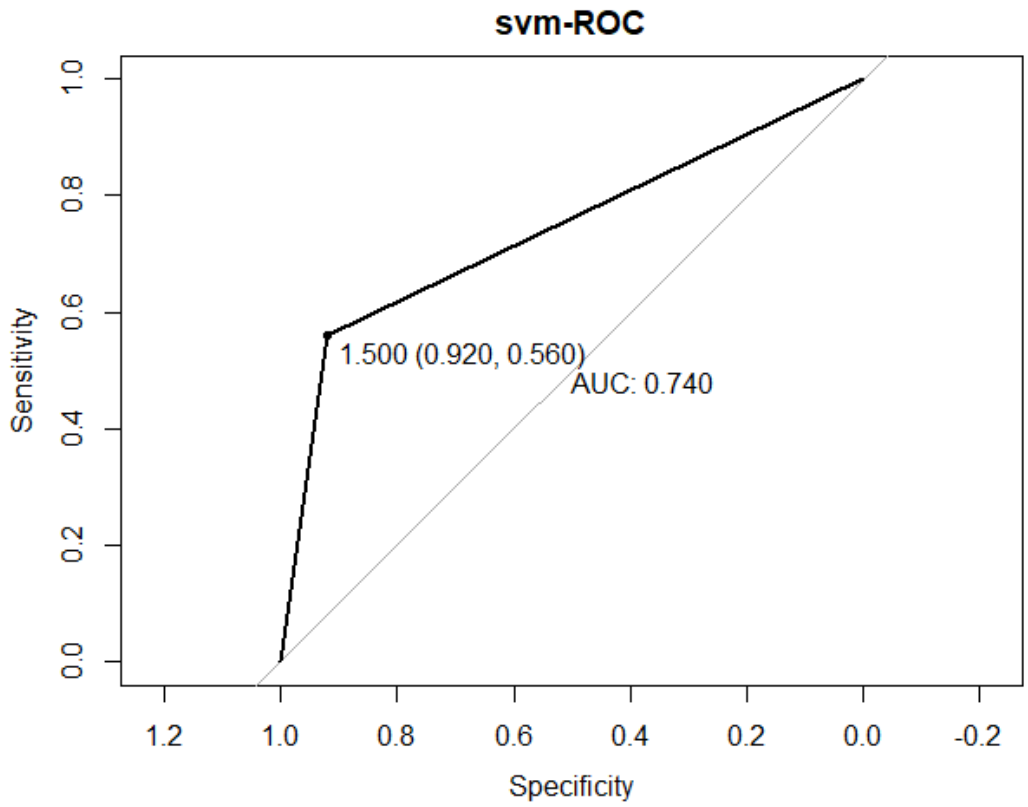



图 11 SVM-ROC 曲线图

综合来说，并没有那一个模型是特别优越的，每一个模型都存在优点和不足，如果非要选出一个的话，随机森林模型和 SVM 模型预测效果排在前列，每一个衡量参数都很不错，而且运行速度很快，模型的构造也容易理解，最优参数的寻找也十分便捷，而且随机森林模型不需要对参数进行标准化处理，这两个模型在本文中的缺点是不能够处理缺失值。

# 第三章 总结与展望

各种模型的正确率都在 0.83 左右，预测的正确率处于相对较高的水平，可以认为由这些变量来预测收入是可行的。但是并没有哪一种方法各类评价指标都显著高于其他方法的，在阅读文献的过程中我发现又有学者提出了一些新的分类方法，但时间紧迫，没有时间进一步探究其优良性。在模型的建立过程中，也遇到了一些问题，主要是内存和 cpu 的限制导致在参数优化的过程中需要大量的时间，SVM 方法建模的过程中因为数据量太大导致内存不够使用，这些问题需要性能更好的机器来解决。

变量的重要性分析，可以得出人际关系是影响收入的最大因素，这也是符合实际的，家庭因素与收入息息相关，毕竟大多数人不是在工作就是和家人在一起，这两方面占据了一个人的绝大多数时间其次，受教育程度是第二重要的变量，理所当然，受到高水平教育的人理应得到更多的收入。与一开始的预想不相符的是，种族和性别在收入方面的影响微乎其微，收入的高低似乎更加依赖自己的出身家庭和后天的受教育程度。当然这只是在美国的现象，中国的具体情况可能会有些许不同。

本文所得出的部分结论与预想的不符，其原因是美国人的收入确实不存在性别歧视和种族歧视，还是说我们中国的国情与美国不同，让我们潜意识里认为不同性别种族的收入应该有差异呢？这需要进一步研究中国人的收入受那些因素的影响才能得到结论。

# 附录 代码

```
library(caret)
library(klaR)
library(kknn)
library(e1071)
library(adabag)
library(randomForest)
library(rpart)
library(rpart.plot)
library(xgboost)
library(caretEnsemble)
library(ROCR)

data <- read.csv("data.csv")
data<-as.data.frame(data[,-c(11,12)])
names(data)[13] <- "y"
levels(data$y)[1] <- "0"
levels(data$y)[2] <- "1"

n <- round(2/3*nrow(data))
set.seed(1)
sub_train=sample(1:nrow(data),n)
datatrain=data[sub_train,]   #训练集
datatest=data[-sub_train,]   #测试集
#naive bayes-----------------------------------------------------------------------------
NB.model <-
NaiveBayes(y~age+fnlwgt+education.num+hours.per.week+.,datatrain,usekernel=T,f
L=1)
pred.NB <- predict(NB.model,datatrain[,-13])#预测
perf.values.NB=confusionMatrix(pred.NB$class,datatrain[,13],positive="1")
perf.values.NB
perf.values.NB$byClass
measure.NB <- postResample(pred.NB$class,datatrain[,13]) #预测效果
measure.NB

#KNN-------------------------------------------------------------------------------------
pre=preProcess(datatrain,method = "range")
newdatatrain=predict(pre,datatrain)
pre=preProcess(datatest,method = "range")
newdatatest=predict(pre,datatest)
```

```
model.tkknn <- train.kknn(y~.,kmax=100,newdatatrain,kernel = c("rectangular",
"triangular", "epanechnikov", "optimal", "cos", "inv", "gaussian","triweight",
"biweight"),distance=2,scale=T)
plot(model.tkknn)
model.tkknn #输出最优参数情况
table(newdatatrain[,13],model.tkknn$fitted.values[[66]]) #混淆矩阵
perf.values.KNN=confusionMatrix(model.tkknn$fitted.values[[66]],newdatatrain[,13]
,positive="1")
perf.values.KNN
perf.values.KNN$byClass
measure.KNN <- postResample(model.tkknn$fitted.values[[66]],newdatatrain[,13]) #
预测效果
measure.KNN

model.kknn <- kknn(y~.,newdatatrain,newdatatest[,-
13],k=66,scale=T,distance=2,kernel= "inv")
measure.kknn <- postResample(model.kknn$fitted.values,newdatatest[,13])
measure.kknn
perf.values.KNN=confusionMatrix(model.kknn$fitted.values,newdatatest[,13],positiv
e="1")
perf.values.KNN
perf.values.KNN$byClass

pred.knn <- prediction(model.kknn$prob[,2] , newdatatest[,13])
perf.auc.knn <- performance(pred.knn,measure="auc")#所得的是一个 list 对象
AUC.KNN=unlist(perf.auc.knn@y.values)#AUC 值
AUC.KNN

#作图
#敏感度-特异度曲线
perf1.knn <- performance(pred.knn,"sens","spec")
plot(perf1.knn,main="knn 敏感度-特异度曲线")
#ROC
perf2.knn <- performance(pred.knn,"tpr","fpr")
plot(perf2.knn,main="knn-ROC")
abline(a=0,b=1,lwd=2,lty=2)
#召回率-精确率曲线
perf3.knn <- performance(pred.knn,"prec","rec")
plot(perf3.knn,main="knn 召回率-精确率曲线")
#提升图
perf4.knn <- performance(pred.knn,"lift","rpp")
plot(perf4.knn,main="knn 提升图")
#决策树--------------------------------------------------------------------------
```

```
rp_rpart <- rpart(y~.,datatrain,minsplit=1,maxdepth=5,cp=0.001, parms
=list(split="gini"))
rpart.plot(rp_rpart,type=4,fallen.leaves=TRUE) #画出树状图

cpmatrix <- printcp(rp_rpart)
plotcp(rp_rpart)
mincpindex <- which.min(cpmatrix[,4])
cponeSE <- cpmatrix[mincpindex,4]+cpmatrix[mincpindex,5]
cpindex <- min(which(cpmatrix[,3]<=cponeSE))
cpmatrix[cpindex,1]#所确定的 cp 值
#剪枝
rp_rpart2 <- prune(rp_rpart,cp=cpmatrix[cpindex,1])
rpart.plot(rp_rpart2,type=4) #每类判定条件更明确

pre <- data.frame(predict(rp_rpart2,datatrain[,-13]))
pre$class <- factor(rep("0",nrow(datatrain)),levels=c("0","1"))
for (i in 1:nrow(datatrain)){
   if (pre[i,1]<pre[i,2])
      pre$class[i] <- factor("1",levels=c("0","1"))
}
perf.values.rp <- confusionMatrix(pre$class,datatrain[,13],positive="1")
perf.values.rp
perf.values.rp$byClass
measure.rp <- postResample(pre$class,datatrain[,13]) #预测效果
measure.rp

#bagging-----------------------------------------------------------------------------
m=10
errorvec <- rep(0,m)
for (i in 1:m){
   set.seed(1)
   bagcv <-
bagging.cv(y~.,data=datatrain,v=10,mfinal=i,control=rpart.control(minsplit=1,maxde
pth=10))
   errorvec[i] <- bagcv$error
}
errorvec
set.seed(1)
model.bag <-
bagging(y~.,data=datatrain,mfinal=4,control=rpart.control(minsplit=1,maxdepth=10))

perf.values <- confusionMatrix(factor(model.bag$class),datatrain$y,positive="1")
perf.values
perf.values$byClass
```

```
measure.bag <- postResample(model.bag$class,datatrain$y)
measure.bag

pred.bag <- predict(model.bag,datatest[,-13],type="class")
perf.values <- confusionMatrix(factor(pred.bag$class),datatest[,13],positive="1")
perf.values
perf.values$byClass

measure.bag <- postResample(pred.bag$class,datatest$y)
measure.bag

pred.bag <- prediction( pred.bag$prob[,2], datatest[,13])
perf.auc.bag <- performance(pred.bag,measure="auc")#所得的是一个 list 对象
AUC.bag=unlist(perf.auc.bag@y.values)#AUC 值
AUC.bag

model.bag$importance #变量相对重要性
importanceplot(model.bag) #变量相对重要性柱状图

#作图
#敏感度-特异度曲线
perf1.bag <- performance(pred.bag,"sens","spec")
plot(perf1.bag,main="bag 敏感度-特异度曲线")
#ROC
perf2.bag <- performance(pred.bag,"tpr","fpr")
plot(perf2.bag,main="bag-ROC")
abline(a=0,b=1,lwd=2,lty=2)
#召回率-精确率曲线
perf3.bag <- performance(pred.bag,"prec","rec")
plot(perf3.bag,main="bag 召回率-精确率曲线")
#提升图
perf4.bag <- performance(pred.bag,"lift","rpp")
plot(perf4.bag,main="bag 提升图")
#RF(最优)--------------------------------------------------------------------------------
datatrain <- datatrain[complete.cases(datatrain),]
datatest <- datatest[complete.cases(datatest),]

p <- ncol(datatrain)-1
err <- rep(0,p)
B <- rep(0,p)
for (i in 1:p){
   set.seed(1)
```

```
   rfmodel <-
randomForest(y~.,data=datatrain,ntree=500,mtry=i,importance=T,proximity=T,contro
l=rpart.control(minsplit=1,maxdepth=10))
   err[i] <- min(rfmodel$err.rate[,1])
   B[i] <- which.min(rfmodel$err.rate[,1])
}
err
B
mtry.optimal <- which.min(err)
ntree.optimal <- B[which.min(err)]
c(mtry.optimal,ntree.optimal)

set.seed(1)
model.rf <-
randomForest(y~.,data=datatrain,ntree=182,mtry=2,importance=T,proximity=T)
plot(model.rf)#看树的个数与 OOB mse 的关系

perf.values <- confusionMatrix(model.rf$predicted,datatrain[,13],positive="1")
perf.values
perf.values$byClass

measure.rf <- postResample(model.rf$predicted,datatrain$y)
measure.rf

pred.rf <- predict(model.rf,datatest[,-13],type="class")
perf.values <- confusionMatrix(pred.rf,datatest[,13],positive="1")
perf.values
perf.values$byClass

measure.rf <- postResample(pred.rf,datatest[,13])
measure.rf

roc.rf <- roc(newdatatest[,13], as.numeric(pred.rf))
plot(roc.rf, print.auc = TRUE, print.thres = TRUE,main="rf-ROC")

model.rf$importance#各变量的重要程度
varImpPlot(model.rf, main="Variable Importance Random Forest computer")#作图

#adaboost--------------------------------------------------------------------
listcoeflearn <- c("Breiman","Freund","Zhu")
m <- 10
err <- matrix(0,nrow=length(listcoeflearn),ncol=m)
for (i in 1:length(listcoeflearn)){
   for (j in 1:m) {
```

```
        set.seed(1)
        cv.model <-
boosting.cv(y~.,data=datatrain,v=2,boos=T,coeflearn=listcoeflearn[i],mfinal=j,control
=rpart.control(minsplit=1,maxdepth=10))
        err[i,j] <- cv.model$error
    }
}
err
x <- 1:m
plot(err[1,],type="b",col="red",ylim=c(0,0.5))
points(err[2,],type="b",col="blue")
points(err[3,],type="b",col="green")
legend('topright',listcoeflearn,col=c("red","blue","green"),pch=rep(1,3))

model.ada    <-
boosting(y~.,data=datatrain,boos=F,coeflearn="Breiman",mfinal=9,control=rpart.cont
rol(minsplit=1,maxdepth=10))

perf.values <- confusionMatrix(factor(model.ada$class),datatrain[,13],positive="1")
perf.values
perf.values$byClass

measure.ada <- postResample(model.ada$class,datatrain$y)
measure.ada

pred.ada <- predict(model.ada,datatest[,-13],type="class")
perf.values <- confusionMatrix(factor(pred.ada$class),datatest[,13],positive="1")
perf.values
perf.values$byClass

measure.ada <- postResample(pred.ada$class,datatest[,13])
measure.ada

pred.ada <- prediction(pred.ada$prob[,2], datatest[,13])
perf.auc.ada <- performance(pred.ada,measure="auc")#所得的是一个 list 对象
AUC.ada=unlist(perf.auc.ada@y.values)#AUC 值
AUC.ada

model.ada$importance
importanceplot(model.ada)

#作图
#敏感度-特异度曲线
perf1.ada <- performance(pred.ada,"sens","spec")
```

```
plot(perf1.ada,main="ada 敏感度-特异度曲线")
#ROC
perf2.ada <- performance(pred.ada,"tpr","fpr")
plot(perf2.ada,main="ada-ROC")
abline(a=0,b=1,lwd=2,lty=2)
#召回率-精确率曲线
perf3.ada <- performance(pred.ada,"prec","rec")
plot(perf3.ada,main="ada 召回率-精确率曲线")
#提升图
perf4.ada <- performance(pred.ada,"lift","rpp")
plot(perf4.ada,main="ada 提升图")
#SVM（跑太慢）----------------------------------------------------------------------------
---
costset <- seq(-5,15,2)
gammaset <- seq(-15,3,2)
costexp <- 2**costset
gammaexp <- 2**gammaset
accuracy <- matrix(0,length(costset),length(gammaset))
for (i in 1:length(costset)) {
   for (j in 1:length(gammaset)){
      set.seed(1)
      cvmodel.svm <- svm(y~., newdatatrain,cross=
2,cost=costexp[i],gamma=gammaexp[j])
      accuracy[i,j] <- cvmodel.svm$tot.accuracy
   }
}
dimnames(accuracy)=list(costset,gammaset) #模型精度变量的列名和行名
accuracy
max(accuracy)#正确率最大值
which.max(accuracy)
set.seed(1)
model.svm <- svm(y~., newdatatrain,cost=2**1,gamma=2**(-3))
summary(model.svm)

pred.svm <- predict(model.svm,newdatatest[,-13])

perf.values <- confusionMatrix(pred.svm,newdatatest[,13],positive="1")
perf.values
perf.values$byClass

measure.svm <- postResample(pred.svm,newdatatest[,13])
measure.svm

roc.svm <- roc(newdatatest[,13], as.numeric(pred.svm))
```

```
plot(roc.svm, print.auc = TRUE, print.thres = TRUE,main="rf-ROC")

onehottrain <- model.matrix(~.-1,data=newdatatrain[,-13])#one-hot 编码后的自变量
onehottrain
plot(cmdscale(dist(onehottrain[,-1])), col = as.integer(newdatatrain[,13]), pch =
c("o","+")[1:5020 %in% model.svm$index +1])
legend("bottomleft",col=c(1,2,2),(byt = n),legend=c("y=no & a supporter","y=yes & a
supporter","y=yes & not a supporter"),pch = c("+","+","o"))

#stacking---------------------------------------------------------------------------
levels(datatrain$y) <- list(no="0", yes="1")

set.seed(1)
list.stacking <- caretList(x=datatrain[,-13],y=datatrain[,13],
trControl=trainControl(method="cv",savePredictions='final',classProbs = TRUE),
methodList=c("rpart","rf"))
list.stacking#不能加 kknn
model.stacking <- caretStack(list.stacking, method="glm")
model.stacking

#xgboost----------------------------------------------------------------------------
newdata <- read.csv("data 无-.csv")
newdata<-as.data.frame(newdata[,-c(11,12)])
names(newdata)[13] <- "y"
levels(newdata$y)[1] <- "0"
levels(newdata$y)[2] <- "1"

n <- round(2/3*nrow(newdata))
set.seed(1)
sub_train=sample(1:nrow(newdata),n)
train=newdata[sub_train,]   #训练集
test=newdata[-sub_train,]   #测试集

trainy <- as.numeric(as.factor(train[,13]))-1 #将因变量转化为 numeric,且取值为 0
和 1
trainx <- data.frame(train[,-13]) #训练集的 x
colnames(trainx) <- names(train[,-13])
colxattr <- c(2,4,6,7,8,9,10,12)
trainxattr <- model.matrix(~.,data=trainx[,colxattr])[,-1]
newtrainx <- cbind(trainx[,-colxattr],trainxattr)
newtrainx <- as.matrix(newtrainx)

param <-
list(seed=1,objective="binary:logistic",max_depth=10,min_child_weight=0.1)
```

```
modelcv.xgb <-
xgb.cv(data=newtrainx,label=trainy,params=param,nrounds=10,nfold=2)

#NN--------------------------------------------------------------------------------
trainy <- model.matrix(~y-1,newdatatrain) #测试集的 y
trainy <-data.frame(newdatatrain[,13])
trainx <- data.frame(newdatatrain[,-13]) #测试集的 x
colnames(trainx) <- names(newdatatrain[,-13])

colxattr <- c(2,4,6,7,8,9,10,12)
trainxattr <- model.matrix(~.,data=trainx[,colxattr])[,-1]
newtrainx <- cbind(trainx[,-colxattr],trainxattr)
newtrain <- as.data.frame(newtrain)


testy <- model.matrix(~y-1,newdatatest) #测试集的 y
testy <-data.frame(newdatatest[,13])
testx <- data.frame(newdatatest[,-13]) #测试集的 x
colnames(testx) <- names(newdatatest[,-13])

colxattr <- c(2,4,6,7,8,9,10,12)
testxattr <- model.matrix(~.,data=testx[,colxattr])[,-1]
newtestx <- cbind(testx[,-colxattr],testxattr)

formulatrain <- as.formula(paste(paste(dimnames(newtrain)[[2]][c(1,2)],collapse = "
+ "),"~", paste(dimnames(newtrainx)[[2]], collapse = " + ")))
set.seed(1)
model.nn <- neuralnet(formulatrain,data=newtrain,hidden=c(9,6),linear.output=F)
predcv <- compute(model.nn,newtrainx)
print(model.nn)

predcv$net.result
predcvclass <- c("0","1")[apply(predcv$net.result, 1, which.max)]
predcvclass
table(predcvclass,train$y)

predcvclass<-as.factor(predcvclass)
per.values <- confusionMatrix(predcvclass,train$y,positive="1")
per.values
per.values$byClass
```