基于分类算法对贵州茅台股份的数据挖掘研究

Hsinyuan

摘要

本文选取 2020 年整年共 249 个交易日的贵州茅台股份金融数据，采用开盘价、最高价、最低价、收盘价、昨日收盘价、涨跌额、涨跌幅、成交量、成交额等现有数据，计算 MACD、KDJ、OBV 及 CCI 等技术指标，按照 75% 置信度，预测一年期内可能造成资产价值的最大损失，并进行回测检验，对真实损失超过预测损失值的数据进行标记。建立 logit、VM、KNN、决策树、神经网络等模型和算法，根据历史数据，预测是否会超过预期损失值，并对模型效果进行评价，通过数据挖掘，达到管控股市交易风险的目的。

关键词： 风险价值；违约；分类模型；模型检验

1 研究背景

投资总会伴随着风险，投资风险是指对未来投资收益的不确定性，在投资中可能会遭受收益损失甚至本金损失的风险。如何进行风险识别和把控，是决定投资者是否成功的关键因素之一。

本文选取 2020 年整年共 249 个交易日的贵州茅台（股票代码为：600519.SH）股份金融数据。在给定的时间期限内，对不利的市场变动可能造成资产价值的最大损失进行估计，能够把控金融风险，避免交易中的重大亏损。

本文的研究路线为：

首先，计算该股票的某一置信度下的风险价值。在该风险价值之内（即最大可能损失值），我们认为该股票的风险尚可承受。下一步，对该评估方法进行回测，将该股票次日跌破风险价值预测阈值的数据标记为“违约”，进一步评估其信用风险。最后，建立模型预测违约。“违约”意味着股票跌幅超过预期，交易产生风险。在实际应用中，利用模型预测违约，能够掌握股票预期大致走向，更好管控股票市场投资带来的风险。

2 数据说明

2.1 数据来源

本文采用财经数据获取接口 Tushare 共享平台，接入 2020 年整年共 249 个交易日的贵州茅台（股票代码为：600519.SH）股份金融数据，共包括股票代码（ts_code）、交易日（trade_date）、开盘价（open）、最高价（high）、最低价（low）、收盘价（close）、昨日收盘价（pre_close）、涨跌额（change）、涨跌幅（pct_chg）、成交量（vol）、成交额（amount）等 11 个字段，如表1所示。

表 1 字段说明

字段	说明	字段	说明
ts_code	股票代码	pre_close	昨日收盘价
trade_date	交易日	change	涨跌额
open	开盘价	pct_chg	涨跌幅
high	最高价	vol	成交量
low	最低价	amount	成交额
close	收盘价		

2.2 数据准备

读取数据后，对数据按照日期升序整理，撰写代码计算股票日收益率（rate）、MACD、OBV、CCI、RSV[1]、K[2]D[3]J 等技术指标。

计算公式如下：

$$rate = \frac{close}{pre\_close} - 1 \tag{1}$$

$$MACD = 2 \times (DIF - DEA) \tag{2}$$

$$OBV = vol \times \frac{(close - low) - (high - close)}{high - close} \tag{3}$$

$$CCI = \frac{\frac{high+low+close}{3} - MACD}{0.015 \times \frac{MACD-close}{5}} \tag{4}$$

$$RSV = \frac{close - low_n}{high_n - low_n} \tag{5}$$

$$K_1 = 50, K_i = \frac{2}{3}K_{i-1} + \frac{1}{3}RSV_i, i \geq 2 \tag{6}$$

$$D_1 = 50, D_i = \frac{2}{3}D_{i-1} + \frac{1}{3}K_i, i \geq 2 \tag{7}$$

$$J_i = 3 \times K_i + 2 \times D_i \tag{8}$$

撰写代码，进行计算。

```
> #### 计算MACD ####
> mt_ascend$MACD <- NA
> dt_macd <- data.frame(MACD(mt_ascend$close))
> DIF <- dt_macd$macd
> DEA <- dt_macd$signal
> MACD <- 2*(DIF-DEA)
> mt_ascend$MACD <- MACD
> mt_ascend <- na.omit(mt_ascend)
> #### 计算OBV ####
> mt_ascend$OBV <- NA
> OBV <- function(v,c,l,h){
+   res <- v*((c-l)-(h-c))/(h-c)
+   return(res)
+ }
> mt_ascend$OBV <- OBV(v=mt_ascend$vol,
+                      c=mt_ascend$close,
+                      l=mt_ascend$low,
+                      h=mt_ascend$high)
> #### 计算CCI ####
> mt_ascend$CCI <- NA
> CCI <- function(h,l,c,m){
+   res <- (sum(h+l+c)/3-m)/((m-c)/5)/0.015
+   return(res)
+ }
> mt_ascend$CCI <- CCI(h=mt_ascend$high,
+                      l=mt_ascend$low,
+                      c=mt_ascend$close,
+                      m=mt_ascend$MACD)
```

[1] low_n、$high_n$ 分别指前 n 日的最低价和最高价。

[2] K_{i-1} 表示前一天的 K 值

[3] D_{i-1} 表示前一天的 D 值

```
> mt_ascend[mapply(is.infinite,mt_ascend)] <- NA
> mt_ascend <- na.omit(mt_ascend)
> #### 计算RSV指标 ####
> mt_ascend$RSV <- NA
> for (i in 1:length(mt_ascend$close)){
+   mt_ascend$RSV[i] =
+     (mt_ascend$close[i]-min(mt_ascend$close[1:i]))/
+     (max(mt_ascend$close[1:i])-min(mt_ascend$close[1:i]))
+ }
> mt_ascend <- na.omit(mt_ascend)
> #### 计算KDJ指标 ####
> # mt_ascend$K[1] = 1/3 * mt_ascend$RSV[1]
> mt_ascend$K <- NA
> mt_ascend$K[1] <- 50
> for (i in 2:length(mt_ascend$close)){
+   mt_ascend$K[i] = 2/3 * mt_ascend$K[i-1] + 1/3 * mt_ascend$RSV[i]
+ }
> # mt_ascend$D[1] = 1/3 * mt_ascend$K[1]
> mt_ascend$D <- NA
> mt_ascend$D[1] = 50
> for (i in 2:length(mt_ascend$close)){
+   mt_ascend$D[i] = 2/3 * mt_ascend$D[i-1] + 1/3 * mt_ascend$K[i]
+ }
> mt_ascend$J <- NA
> for (i in 1:length(mt_ascend$close)){
+   mt_ascend$J[i] = 3 * mt_ascend$K[i] - 2 * mt_ascend$D[i]
+ }
> mt_ascend <- na.omit(mt_ascend)
```

3 数据分析

3.1 VaR 计算

VaR 指的是一定时期内某类频率的收益率分布的下分位点，比如过去一年内日收益率的下 75% 分位点，度量的是 75% 置信度下每日持仓该股票的最大可能损失值 [1]。计算 VaR，可以用来评估市场风险的大小。

本文采用 VaR 的非参数计算方法。计算步骤为：将长度为 n 的收益率序列从小到大排序，第 $[n \times a]$ 和 $[n * timesa] + 1$ 的均值即为 $1 - a$ 置信度下的 VaR。

```
> mt_ascend <- mt[order(mt$trade_date),] # 对数据按照日期升序整理
> mt_ascend <- mt_ascend %>%
+   select(trade_date,open,high,low,close,pct_chg,change,vol)
> #### 计算收益率 ####
> mt_ascend$r <- NA
> mt_ascend$r[2:length(mt_ascend$trade_date)]<-
+ mt_ascend$close[2:length(mt_ascend$trade_date)] /
+ mt_ascend$close[1:length(mt_ascend$trade_date)-1] - 1
> mt_ascend <- na.omit(mt_ascend)
> #### 计算75%置信度的日度VaR ####
> var <- -qnorm(0.25,mean(mt_ascend$r),sd(mt_ascend$r))
> var
[1] 0.009709702
```

由 2020 年贵州茅台股份的日收益率，计算 75% 置信度的日度 VaR。75% 置信度的日度 VaR 为 0.0097。说明在 2020 年，茅台股份在一天内损失超过 0.0097 的概率为 25%。

3.2 回测

在交易策略、投资策略或风险建模中，回测旨在估计策略或模型在过去一段时间内的表现。只有能准确地预测风险的 VaR 模型才是有效的 [2]。因此，模型的运用过程就是一个不断检验证明的过程。模型验证是检验一个模型是否正确的一般过程，回测检验是一种规范的统计方法，它事实上是通过将实际发生的损失，与统计预测的损失进行比较，从而验证模型的有效性。对 VaR 模型来说，这包括把 VaR 模型的历史预测与真实的数据进行比较，也就是将利用模型进行事前预测得到的 VaR

结果与事后真实发生的损失进行统计意义上的比较，从而检验模型的预测能力是否能符合我们的要求。

回测对于 VaR 使用者和风险管理者对所建立的 VaR 模型进行有效性核查和检验来说，是至关重要的。如果不是这样，就要重新对模型进行假设错误、参数错误和模型错误的检验，将增加问题的复杂性。同时，回测检验过程也可以为模型的改进提供一些思路 [3]。

根据以上对 VaR 的计算，评估信用风险。信用风险指的是在金融交易中，由对手方可能的违约带来的风险。信用事件可以狭义地定义为债券的违约，即债券发行机构无法支付承诺的利息支付或本金偿还。此处，我们将次日跌幅超过 VaR 预测的阈值，标记为“违约”。

```
> #### 是否违约 ####
> mt_ascend$de <- NA
> mt_ascend$de_1 <- NA
> for (i in 1:length(mt_ascend$close)){
+   if ((mt_ascend$pct_chg[i] - var) > 0){
+     mt_ascend$de[i] <- 1
+     mt_ascend$de_1[i] <- "违约"
+   }
+   else{
+     mt_ascend$de[i] <- 0
+     mt_ascend$de_1[i] <- "不违约"
+   }
+ }
```

对于违约概率进行评估，可以使用分类模型预测违约概率，也可以使用如神经网络、支持向量机等机器学习的方法。

4 模型预测

标记出“违约”数据，将数据去除空缺值，标准化后，建立模型预测违约。

去除空缺值的数据共 200 行，随机抽取 70% 的观测放入学习数据集，剩余 30% 放入测试数据集。

4.1 Logit 分类模型

4.1.1 建立模型

在实际经济问题中，被解释变量为定性的（“违约”和“不违约”），适用 Logit 分类模型对其进行预测。

$$log[\frac{P(y=1)}{1-P(y=1)}] = c + \sum_{i=1}^{q} \beta_i X_i + \epsilon_t \tag{9}$$

如公式9，其中 y 为虚拟变量，$y=1$ 表示“违约”，$y=0$ 表示“不违约”；X 是一系列对于分类有影响的控制变量。此处，我们用成交量（vol）、涨跌额（change）、日盈利率（r）、MACD、OBV、CCI、KDJ 等技术指标预测违约。

用 R 中的 glm 函数可以估计 logit 模型。上一步的回测中，构建了一系列已经知道违约与否的样本的特征，用此训练样本，估计 β。带入新特征，拟合 $y=1$ 的概率 [4]。

经过调整，检查拟合后的模型。

```
> logit <- glm(de~vol+change+r+MACD+OBV+CCI+K+D,
               family=binomial(link='logit'),
                              train)
> summary(logit)
##
## Call:
## glm(formula = de ~ vol + change + r + MACD + OBV + CCI + K +
##     D, family = binomial(link = "logit"), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
##    0.00     0.00     0.00     0.00     8.49
##
## Coefficients:
```

```
##                Estimate Std. Error    z value Pr(>|z|)
## (Intercept) -8.296e+14  3.412e+07 -24316363   <2e-16 ***
## vol         -2.081e+08  3.809e+02   -546362   <2e-16 ***
## change      -4.981e+13  1.235e+06 -40335157   <2e-16 ***
## r            1.902e+17  1.968e+09  96632273   <2e-16 ***
## MACD        -1.063e+13  5.032e+06  -2112539   <2e-16 ***
## OBV          8.932e+08  2.397e+01  37265103   <2e-16 ***
## CCI         -9.521e+09  4.665e+02 -20408903   <2e-16 ***
## K            1.190e+14  7.915e+06  15030023   <2e-16 ***
## D           -1.299e+14  7.946e+06 -16353956   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 192.68  on 139  degrees of freedom
## Residual deviance: 432.52  on 131  degrees of freedom
## AIC: 450.52
##
## Number of Fisher Scoring iterations: 25
```

结果分析：模型中的 8 个预测变量（vol、change、r、MACD、OBV、CCI、K、D）的系数均通过显著性检验（即 p 值小于 0.1）。

用该模型对测试数据集进行预测。评估预测准确性，输出预测与实际情况对比的交叉表，即混淆矩阵。

```
> log <- predict(logit,test,type="response")
> # predict() 函数默认输出违约的对数概率
> # 指定参数type="response"即可得到预测"违约"的概率
> obs_p_rf <- factor(log > .5, levels=c(TRUE, FALSE),
                     labels=c("违约", "不违约"))
# obs_p_rf <- data.frame(prob=log,
#                        obs=test$de)
> table(test$de_1,obs_p_rf,dnn=c("Actual", "Predicted"))
##          Predicted
## Actual    违约  不违约
##    不违约     1     30
##    违约      26      3
```

由以上可得，模型正确判别了 26 个“违约”的数据和 30 个“不违约”数据。在验证集上，正确分类的模型（即准确率 Accuracy）为：

$$Accuracy = \frac{26+30}{60} = 93.33\% \tag{10}$$

4.1.2 模型检验

通过绘制 ROC 曲线图进行模型检验。ROC 曲线图是反映敏感性与特异性之间关系的曲线。横坐标 X 轴为假阳性率（误报率），X 轴越接近零准确率越高；纵坐标 Y 轴称为敏感度，也称为真阳性率，Y 轴越大代表准确率越好。

根据曲线位置，把整个图划分成了两部分，曲线下方部分的面积被称为 AUC（Area Under Curve），用来表示预测准确性，AUC 值越高，也就是曲线下方面积越大，说明预测准确率越高。曲线越接近左上角（X 越小，Y 越大），预测准确率越高。

```
> #### 绘制ROC曲线 ####
> log <- roc(test$de,as.numeric(log))
> plot(log,
+      print.auc=TRUE,
+      auc.polygon=TRUE,
+      grid=c(0.1,0.2),
+      grid.col=c('green','red'),
+      max.auc.polygon=TRUE,
+      auc.polygon.col='chocolate',
+      print.thres=TRUE,
```

```
11 +       main='logit 模型 ROC 曲线')
```

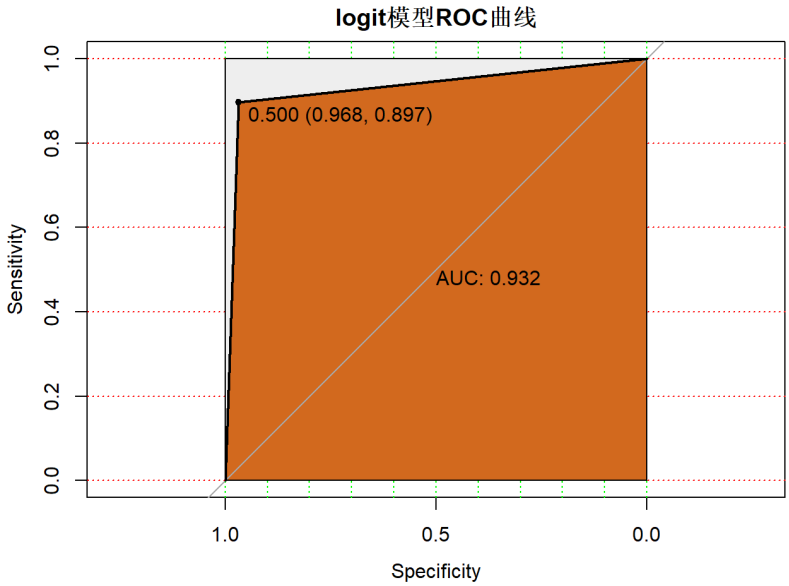

图 1　Logit 分类模型 ROC 曲线图

如图1所示，Logit 模型 ROC 曲线中，AUC 值为 0.932，表现出较好的预测准确率。

4.2　SVM 算法

SVM 是由模式识别中广义肖像算法发展而来的分类器，是一类按监督学习方式对数据进行二元分类的广义线性分类器。借助 R 语言中 e1071 包，可用于调用 SVM 算法。

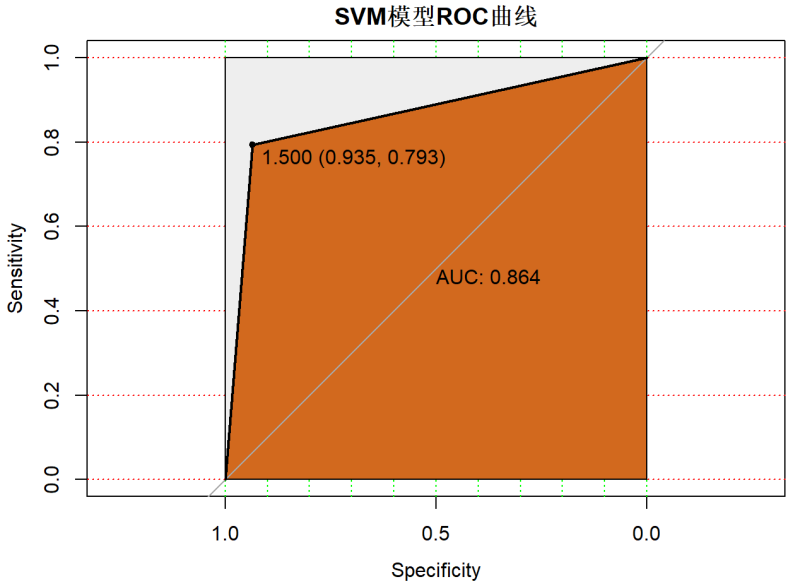

图 2　SVM 算法 ROC 曲线图

```
1 > #### SVM ####
2 > library(e1071)
3 > SVM <- svm(de~vol+change+r+MACD+OBV+CCI+K+D+J,
4 +            train,
5 +            type='C',
6 +            kernel='radial')
7 > svm <- predict(SVM,test)
```

```
> obs_p_rf <-data.frame(prob=svm,
+                       obs=test$de)
> table(test$de,svm,dnn=c('actuality','prediction'))
##           prediction
## actuality  0  1
##         0 29  2
##         1  6 23
```

如图2所示，SVM 算法 ROC 曲线中，AUC 值为 0.864。

4.3 神经网络算法

神经网络算法的主要思想是模拟人脑的工作特性，通过训练和学习掌握最佳的分类方法。其基本的组成单元是神经元，即分类函数。在多个神经元的作用下，并经过多次学习和分类，最终会找出给定的可用分类函数集内的最优的分类组合方法。

理论上来说，神经网络算法在一定深度下，满足一定的神经元个数就能拟合任意函数，所以广泛应用于数据预测，也可以用于二元分类。R 语言中 nnet 包即可调用神经网络算法。

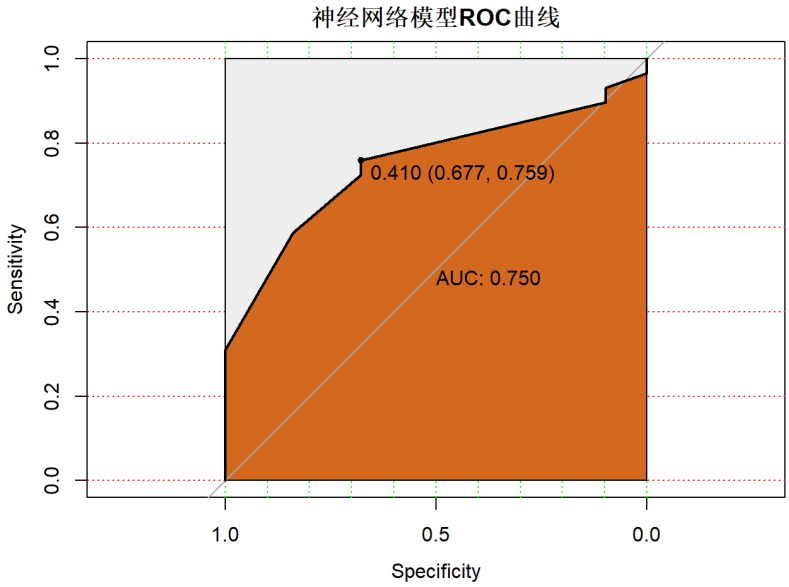

图 3　神经网络算法 ROC 曲线图

```
> #### NNET ####
> library(nnet)
> #### NNET ####
> NNET <- nnet(de~vol+change+r+MACD+OBV+CCI+K+D+J,
+              train,
+              size=6,
+              decay=0.01,
+              maxit=1000,
+              linout=T)
> nn <- predict(NNET,test)
> obs_p_rf <- factor(nn > .5, levels=c(TRUE, FALSE),
+                          labels=c("违约", "不违约"))
> table(test$de_1,obs_p_rf,dnn=c("Actual", "Predicted"))
##          Predicted
## Actual    违约  不违约
##    不违约    10      21
##    违约      22       7
```

如图3所示，神经网络算法 ROC 曲线中，AUC 值为 0.750。对比前两种算法，神经网络算法给出的模型预测效果并不理想。

4.4 KNN 分类算法

KNN 分类算法又称为最近邻估计法。KNN 是有监督学习的 K 近邻的机器学习算法。若果空间中某些样本具有相近的特征属性（样本距离比较近），我们可以认为它们的目标属性 Y 是相近的。因此，我们也可以用已有的最近 K 个样本的目标属性来预测待测样本的目标属性。

选取最近的 k 个点，距离定义一般如下：

$$kD(a,b) = \sqrt{\sum_{i=1}^{k}(X_i^a - X_i^b)^2} \tag{11}$$

最近的 k 个点中哪类最多，则判定该点属于哪一类。

利用 R 中 class 包，调用 KNN 算法。

```
> #### KNN ####
> maotai.knn <- kknn(de~vol+change+r+MACD+OBV+CCI+K+D+J,
+                    train,
+                    test)
> pre_knn <- fitted(maotai.knn)
> #### 绘制 ROC 曲线 ####
> knn_roc <- roc(test$de,as.numeric(pre_knn))
> plot(knn_roc,
+      print.auc=TRUE,
+      auc.polygon=TRUE,
+      grid=c(0.1, 0.2),
+      grid.col=c("green", "red"),
+      max.auc.polygon=TRUE,
+      auc.polygon.col="pink",
+      print.thres=TRUE,
+      main='KNN模型ROC曲线')
```

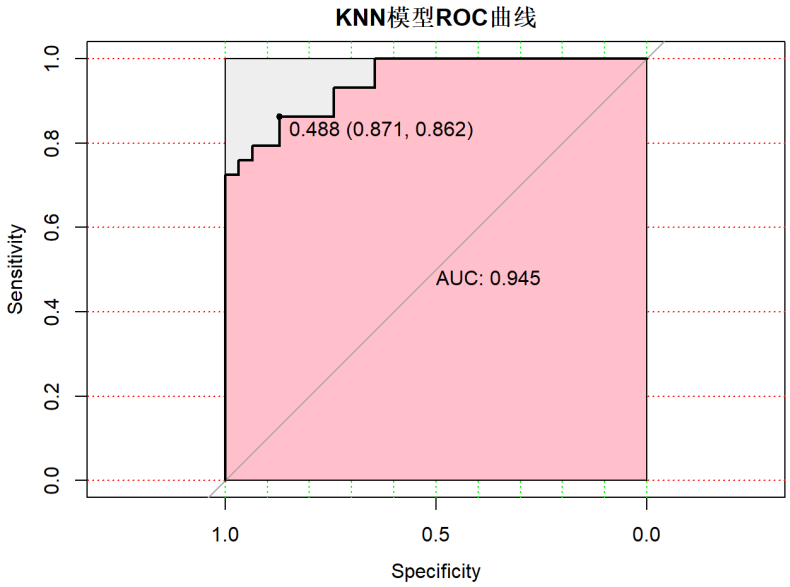

图 4　KNN 算法 ROC 曲线图

如图4所示，KNN 算法 ROC 曲线中，AUC 值为 0.945，预测效果较好。

4.5 决策树模型

决策树是一种通过对历史数据进行测算实现对新数据进行分类和预测的算法，通过对已有明确结果的历史数据进行分析，寻找数据中的特征。并以此为依据对新产生的数据结果进行预测。

```
> #### 决策树 ####
> tree <- rpart(de~vol+change+r+MACD+OBV+CCI+K+D+J,
```

```
+                train)
> pre_rf <- predict(tree,test)
> obs_p_rf <- data.frame(prob=pre_rf,obs=test$de)
> table(test$de,pre_rf,dnn=c('actuality','prediction'))
##           prediction
## actuality  0  1
##         0 31  0
##         1  0 29
> rf_roc <- roc(test$de,as.numeric(pre_rf))
> plot(rf_roc,
+      print.auc=TRUE,
+      auc.polygon=TRUE,
+      grid=c(0.1,0.2),
+      grid.col=c('green','red'),
+      max.auc.polygon=TRUE,
+      auc.polygon.col='chocolate',
+      print.thres=TRUE,
+      main='决策树模型ROC曲线')
```

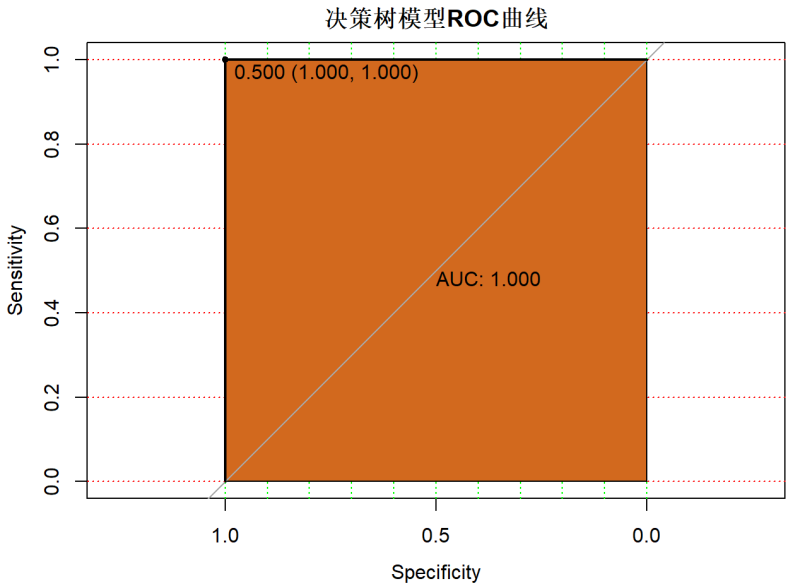

图 5　决策树算法 ROC 曲线图

如图5所示，决策树算法 ROC 曲线中，AUC 值为 1，预测效果是以上算法中最优的。

4.6　总结

本小节中，运用 logit 分类模型、KNN 算法、神经网络算法、SVM 算法、决策树算法，对数据是否违约进行预测。其中预测效果最好的为决策树算法，其次为 KNN 算法、logit 和 SVM 算法。

KNN 算法计算量大，虽然准确率较高，但是当样本不平衡时，如一个类的样本容量很大，而其他类样本容量很小时，有可能导致当输入一个新样本时，该样本的 K 个邻居中大容量类的样本占多数，从而出现误差。

神经网络算法并不是非常好的二元分类算法。因为对于“违约”与“不违约”的判断，要么是 1，要么是 0，而实际这种方式计算出来的结果可以是任意值，这样的概率意义不大。

因此，考虑使用决策树算法和 logit 分类模型进行预测。

5　改进方向

本文从实际出发，首先计算该股票的某一置信度下的风险价值。在该风险价值之内（即最大可能损失值），我们认为该股票的风险尚可承受。下一步，对该评估方法进行回测，将该股票次日跌破风险价值预测阈值的数据标记为“违约”，进一步评估其信用风险。最后，建立模型预测违约。“违约”意味着股票跌幅超过预期，交易产生风险。在实际应用中，利用模型预测违约，能够掌握股票

预期大致走向，更好管控股票市场投资带来的风险。通过研究，本文提供了一种用于预测股票市场的实际可行的方案。

综合考虑，本文改进方向有以下两点。

一，在数据量上，本文运用 2020 年整年共 249 个交易日的贵州茅台股份金融数据，数据量较小。下一步将考虑使用近五年的数据进行预测。并且在预测模型的选择上，应该考虑增加数据量后的运算速度，综合准确率和运算速度选择最优算法。

二，在预测颗粒度上太大。本文通过多种算法预测是否“违约”，即股票的跌幅是否会超过预期（超过预期则记为“违约”），该预测颗粒度范围偏大，下一步将考虑预测更细致的股票涨跌走向。

参考文献

[1] 郑文通. 金融风险管理的 VAR 方法及其应用[J]. 国际金融研究, 1997(09):58-62.

[2] 徐光林. 回测检验在商业银行市场风险度量中的应用研究[J]. 金融理论与实践, 2010, 000(001):20-24.

[3] 薛玲. 回测检验研究及在风险价值模型中的应用[D]. 重庆: 重庆理工大学, 2010.

[4] 任晓萌. 基于逻辑样条回归的信用风险预测模型[D]. 大连: 大连理工大学, 2020.

A 代码

```
#### 导入宏包 ####
library(readr)# 读取数据
library(dplyr)# 清洗数据
library(kknn)
library(TTR)
library(AER)
library(pROC)
library(e1071)
library(nnet)
library(rpart)
setwd('C:/Users/86152/Desktop/0609')# 设置工作空间

mt <- read_csv("maotai.csv")# 读取数据
mt <- mt[,-1]
head(mt)
mt_ascend <- mt[order(mt$trade_date),] # 对数据按照日期升序整理
mt_ascend <- mt_ascend %>%
  select(trade_date,open,high,low,close,pct_chg,change,vol)
#### 计算收益率 ####
mt_ascend$r <- NA
mt_ascend$r[2:length(mt_ascend$trade_date)]<-
mt_ascend$close[2:length(mt_ascend$trade_date)] /
mt_ascend$close[1:length(mt_ascend$trade_date)-1] - 1
mt_ascend <- na.omit(mt_ascend)
#### 计算75%置信度的日度VaR ####
var <- -qnorm(0.25,mean(mt_ascend$r),sd(mt_ascend$r))
var
print(paste('75%置信度的日度VaR为',
            round(var,4),
            '。',
            '说明在2020年，茅台股份在一天内损失超过',
            round(var,4),
            '的概率为25%。',
            sep=' '))
#### 是否违约 ####
mt_ascend$de <- NA
mt_ascend$de_1 <- NA
for (i in 1:length(mt_ascend$close)){
  if ((mt_ascend$pct_chg[i] - var) > 0){
```

```
    mt_ascend$de[i] <- 1
    mt_ascend$de_1[i] <- "违约"
  }
  else{
    mt_ascend$de[i] <- 0
    mt_ascend$de_1[i] <- "不违约"
  }
}

#### 计算MACD ####
mt_ascend$MACD <- NA
dt_macd <- data.frame(MACD(mt_ascend$close))
DIF <- dt_macd$macd
DEA <- dt_macd$signal
MACD <- 2*(DIF-DEA)
mt_ascend$MACD <- MACD
mt_ascend <- na.omit(mt_ascend)
#### 计算OBV ####
mt_ascend$OBV <- NA
OBV <- function(v,c,l,h){
  res <- v*((c-l)-(h-c))/(h-c)
  return(res)
}
mt_ascend$OBV <- OBV(v=mt_ascend$vol,
                     c=mt_ascend$close,
                     l=mt_ascend$low,
                     h=mt_ascend$high)
#### 计算CCI ####
mt_ascend$CCI <- NA
CCI <- function(h,l,c,m){
  res <- (sum(h+l+c)/3-m)/((m-c)/5)/0.015
  return(res)
}
mt_ascend$CCI <- CCI(h=mt_ascend$high,
                     l=mt_ascend$low,
                     c=mt_ascend$close,
                     m=mt_ascend$MACD)
mt_ascend[mapply(is.infinite,mt_ascend)] <- NA
mt_ascend <- na.omit(mt_ascend)
#### 计算RSV指标 ####
mt_ascend$RSV <- NA
for (i in 1:length(mt_ascend$close)){
  mt_ascend$RSV[i] =
    (mt_ascend$close[i]-min(mt_ascend$close[1:i]))/
    (max(mt_ascend$close[1:i])-min(mt_ascend$close[1:i]))
}
mt_ascend <- na.omit(mt_ascend)
#### 计算KDJ指标 ####
# mt_ascend$K[1] = 1/3 * mt_ascend$RSV[1]
mt_ascend$K <- NA
mt_ascend$K[1] <- 50
for (i in 2:length(mt_ascend$close)){
  mt_ascend$K[i] = 2/3 * mt_ascend$K[i-1] + 1/3 * mt_ascend$RSV[i]
}
# mt_ascend$D[1] = 1/3 * mt_ascend$K[1]
mt_ascend$D <- NA
mt_ascend$D[1] = 50
for (i in 2:length(mt_ascend$close)){
  mt_ascend$D[i] = 2/3 * mt_ascend$D[i-1] + 1/3 * mt_ascend$K[i]
}
mt_ascend$J <- NA
for (i in 1:length(mt_ascend$close)){
  mt_ascend$J[i] = 3 * mt_ascend$K[i] - 2 * mt_ascend$D[i]
}
```

```
mt_ascend <- na.omit(mt_ascend)
#### logit ####
nrow(mt_ascend)
set.seed(1)
pre <- sort(sample(200,140))
train <- mt_ascend[pre,]
test <- mt_ascend[-pre,]
head(train)

logit <- glm(de~vol+change+r+MACD+OBV+CCI+K+D,
             family=binomial(link='logit'),
             train)
summary(logit)
log <- predict(logit,test,type="response")
#predict()函数默认输出违约的对数概率
#指定参数type="response"即可得到预测"违约"的概率
obs_p_rf <- factor(log > .5, levels=c(TRUE, FALSE),
                   labels=c("违约", "不违约"))
obs_p_rf
# obs_p_rf <- data.frame(prob=log,
#                        obs=test$de)

# table(test$de,log,dnn=c('actuality','prediction'))
table(test$de_1,obs_p_rf,dnn=c("Actual", "Predicted"))

#### 绘制ROC曲线 ####
log <- roc(test$de,as.numeric(log))
plot(log,
     print.auc=TRUE,
     auc.polygon=TRUE,
     grid=c(0.1,0.2),
     grid.col=c('green','red'),
     max.auc.polygon=TRUE,
     auc.polygon.col='chocolate',
     print.thres=TRUE,
     main='logit模型ROC曲线')
#### SVM ####
SVM <- svm(de~vol+change+r+MACD+OBV+CCI+K+D+J,
           train,
           type='C',
           kernel='radial')
svm <- predict(SVM,test)
obs_p_rf <-data.frame(prob=svm,
                      obs=test$de)
table(test$de,svm,dnn=c('actuality','prediction'))
#### 绘制ROC曲线 ####
svm <- roc(test$de,as.numeric(svm))
plot(svm,
     print.auc=TRUE,
     auc.polygon=TRUE,
     grid=c(0.1,0.2),
     grid.col=c('green','red'),
     max.auc.polygon=TRUE,
     auc.polygon.col='chocolate',
     print.thres=TRUE,
     main='SVM模型ROC曲线')
#### NNET ####
NNET <- nnet(de~vol+change+r+MACD+OBV+CCI+K+D+J,
             train,
             size=6,
             decay=0.01,
             maxit=1000,
             linout=T)
nn <- predict(NNET,test)
```

```
obs_p_rf <-data.frame(prob=nn,
                      obs=test$de)
table(test$de,nn,dnn=c('actuality','prediction'))
nn <- roc(test$de,as.numeric(nn))
plot(nn,
     print.auc=TRUE,
     auc.polygon=TRUE,
     grid=c(0.1,0.2),
     grid.col=c('green','red'),
     max.auc.polygon=TRUE,
     auc.polygon.col='chocolate',
     print.thres=TRUE,
     main='神经网络模型ROC曲线')
#### KNN ####
maotai.kknn <- kknn(de~vol+change+r+MACD+OBV+CCI+K+D+J,
                    train,
                    test)
pre_knn <- fitted(maotai.kknn)
#### 绘制 ROC 曲线 ####
knn_roc <- roc(test$de,as.numeric(pre_knn))
plot(knn_roc,
     print.auc=TRUE,
     auc.polygon=TRUE,
     grid=c(0.1, 0.2),
     grid.col=c("green", "red"),
     max.auc.polygon=TRUE,
     auc.polygon.col="pink",
     print.thres=TRUE,
     main='KNN模型ROC曲线')
#### 决策树 ####
tree <- rpart(de~vol+change+r+MACD+OBV+CCI+K+D+J,
              train)
summary(tree)
pre_rf <- predict(tree,test)
obs_p_rf <- data.frame(prob=pre_rf,obs=test$de)
table(test$de,pre_rf,dnn=c('actuality','prediction'))
rf_roc <- roc(test$de,as.numeric(pre_rf))
plot(rf_roc,
     print.auc=TRUE,
     auc.polygon=TRUE,
     grid=c(0.1,0.2),
     grid.col=c('green','red'),
     max.auc.polygon=TRUE,
     auc.polygon.col='chocolate',
     print.thres=TRUE,
     main='决策树模型ROC曲线')
```