20152410 배형준 Data Mining HW2

Main text

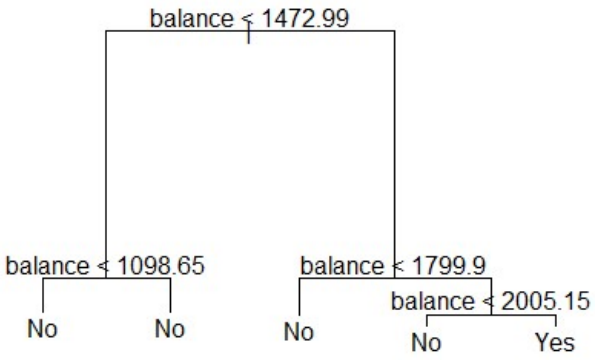
1. Construct (a) a naive Bayes classifier, (b) a classification tree classifier, and (c) a logistic regression classifier.

(a) a naive Bayes classifier

naiveBayes 함수를 이용하여 naïve Bayes classifier 를 학습하였다. 모델의 기본 가정에 의해 student 변수는 'No', 'Yes'로 이뤄진 범주형 변수이므로 binomial dstn 을 따른다고 가정했고 balance, income 변수는 연속형 변수이므로 각각 독립적인 normal dstn 을 따른다고 가정했다. 코드 결과를 살펴보면 default 의 값에 따라 student 변수의 binomial dstn 의 추정된 분포를 확인할 수 있고, default 값에 따라 balance, income 변수의 normal dstn 의 추정된 평균과 분산을 확인할 수 있다.

(b) a classification tree classifier

tree 함수를 이용하여 classification tree 를 학습하였고 노드 분할의 기준 통계량을 deviance 를 사용하였다. 학습된 tree 의 규칙은 아래와 같다.



깊이 1 에서 balance 변수가 1472.99 미만 여부로 쪼개졌고, 깊이 2 에선 balance 변수가 각각 1098.65 미만 여부, 1799.9 미만 여부에 따라 쪼개졌다. 마지막으로 깊이 3 에선 balance 변수가 1799.9 초과인 경우를 대상으로 2005.15 미만 여부에 따라 쪼개졌다. 가지고 있는 predictors 가 3 개인데 balance 변수만 사용한 것으로 보아 좋은 모델이라고 말할 순 없다고 생각한다.

(c) a logistic regression classifier

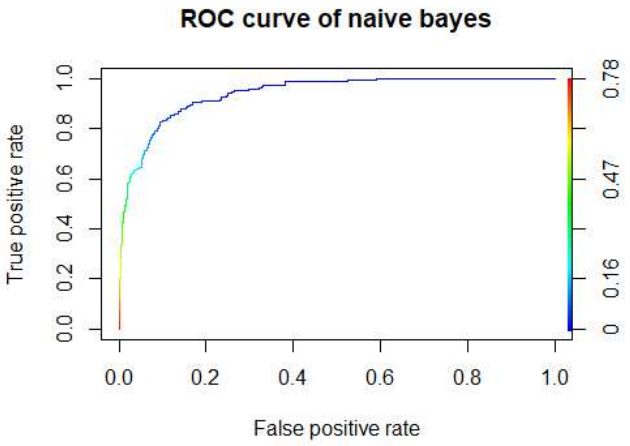
glm(family='binomial') 함수를 사용해 logistic regression 을 학습하였다. 결과는 아래의 표와 같다.

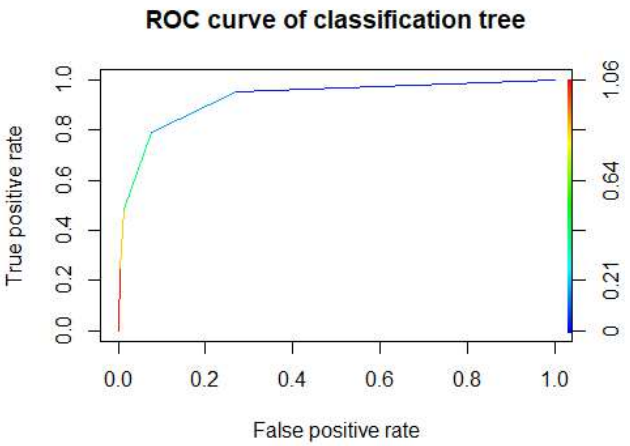
	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	-1.15E+01	7.33E-01	-15.726	< 2e-16	***
studentYes	-9.30E-01	3.33E-01	-2.791	0.00525	**
balance	6.19E-03	3.56E-04	17.407	< 2e-16	***
income	5.70E-06	1.14E-05	0.499	0.61767	

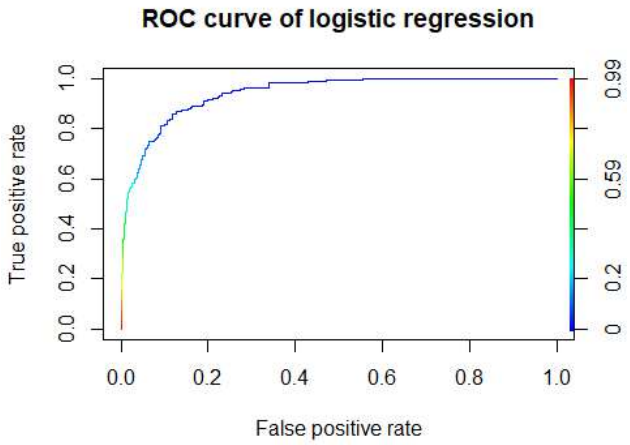
변수를 하나씩 살펴보겠다. Student 가 'Yes'이면 default 가 'No'일 확률이 'Yes'일 확률보다 크다. balance 가 증가할수록 default 가 'Yes'일 확률이 증가하고 income 이 증가할수록 default 가 'Yes'일 확률이 증가한다. student 변수와 balance 변수는 유의미한 변수이고, income 변수는 유의미하지 않은 변수이다.

2. For each classifier, make an ROC curve, calculate the AUC, and compare the three classifiers.

ROCR 패키지의 prediction(pred, true)와 performance(prediction, 'tpr', 'fpr') 함수를 사용하여 ROC curve 를 출력했다. 또한 performance(prediction, 'auc')를 사용하여 AUC 를 출력하였다.







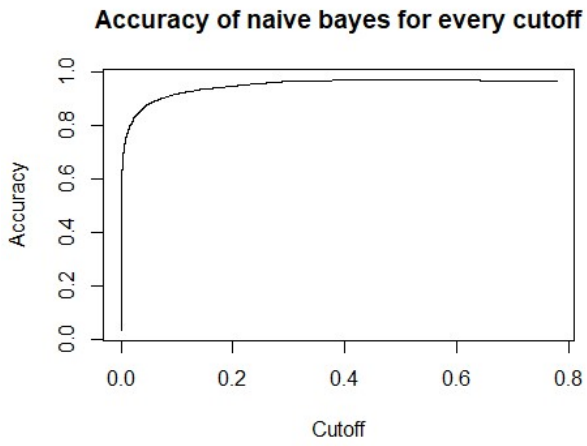
모델들의 ROC curve 를 살펴보도록 하겠다. 그래프의 색은 cutoff 를 의미하고 cutoff 에 따라 tpr, fpr 이 표현되었다. y=x 직선보다는 (0, 1)에 가깝게 curve 가 그려진 것으로 보아 well-seperated data 라고 판단할 수 있다. 눈에 띄는 점은 tree 의 ROC curve 인데, 'Yes'일 확률이 [0, 1] 구간 사이에 조밀하게 분포하지 않고 각 노드별로 class 에 대한 확률이 계산되다보니 curve 가 cutoff 에 따라 뚝뚝 끊기는 것을 볼 수 있다.

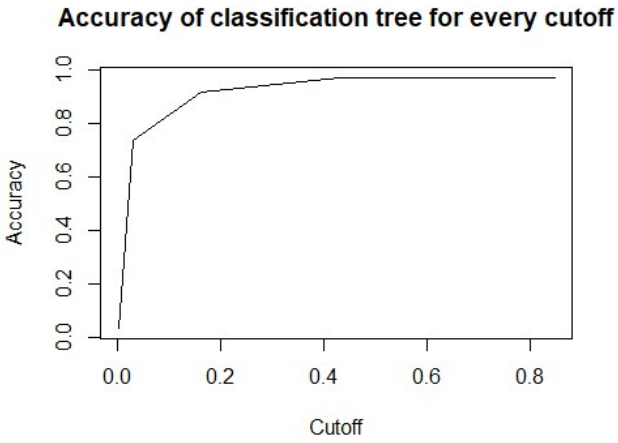
```
## AUC of naive bayes : 0.9409379
## AUC of classification tree : 0.9248116
## AUC of logistic regression : 0.941824
```

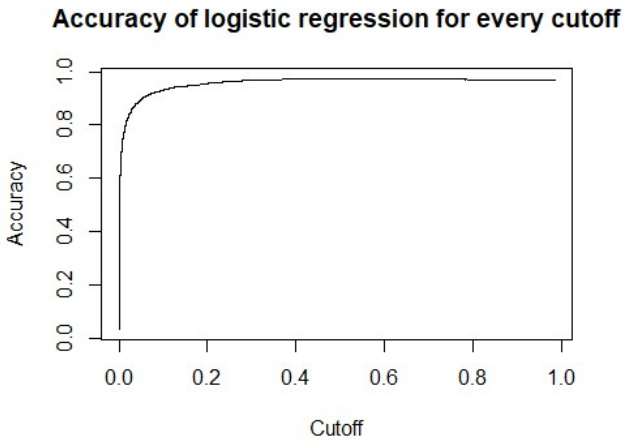
위의 출력값은 모델들의 AUC 이다. logistic regression 의 AUC 가 가장 큰 것을 확인할 수 있다. 값의 차이가 크지 않기 때문에 AUC 하나만 기준으로 삼아 어느 모델이 가장 우수한지 판단하는 것보단 다른 통계량도 같이 살펴보는게 좋다고 생각한다.

3. For each classifier, find the optimal cutoff value to maximize the accuracy. Compare the three classifiers.

ROC curve 를 계산할때와 마찬가지로 ROCR 패키지의 performance 함수를 사용하였다. 인수만 'acc'로 바꿔주면 cutoff 별로 accuracy 를 출력할 수 있다.







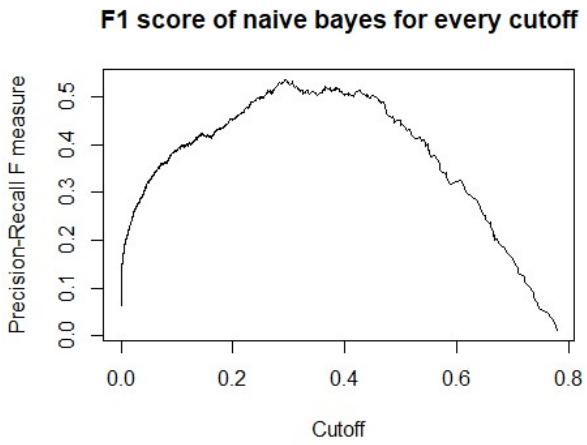
그래프와 아래 출력값을 통해 cutoff 에 해당하는 accuracy 를 알 수 있다. 3 개의 모델 모두 약 97%의 정확도를 가지지만 cutoff 값은 약 0.47, 0.85, 0.51 로 다른 것을 확인할 수 있다.

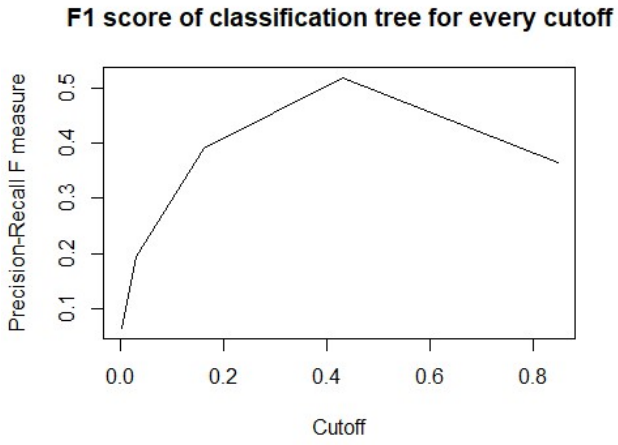
다만 default 변수의 불균형이 심하기 때문에 accuracy 를 기준으로 어느 모델이 가장 좋은지를 선정하는 것은 적절하지 않다고 평가할 수 있다.

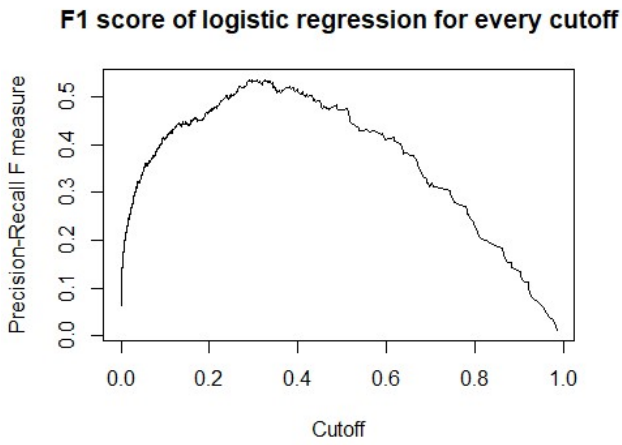
```
## Optimal cutoff of naive bayes :  0.4706127
## Maximum accuracy of naive bayes :  0.9734
## Optimal cutoff of classification tree :  0.8478261
## Maximum accuracy of classification tree :  0.9716
## Optimal cutoff of logistic regression :  0.5119097
## Maximum accuracy of logistic regression :  0.9736
```

4. For each classifier, find the optimal cutoff value to maximize the F1 score. Compare the three classifiers.

위와 마찬가지로 ROCR 패키지의 performance 함수를 이용했으며 measure='f' 인수로 f1 score 를 출력하였다.







naive bayes 와 logistic regression 의 f1 score plot 은 전반적으로 비슷하게 생겼고, classification tree 는 모델의 특성 상 직선을 여러 개 연결한 모양의 f1 score plot 을 가졌다. cutoff 가 1 에 가까워질수록 precision 은 1 로 수렴하지만 recall 이 0 으로 수렴하여 f1 score 는 0 으로 수렴하고, cutoff 가 0 에 가까워질수록 precision 은 0 으로 수렴하고 recall 은 1 로 수렴하여 f1 score 는 0 으로 수렴한다. 각 모델의 최적 cutoff 와 maximum f1 score 는 아래와 같다.

```
## Optimal cutoff, Maximum F1 score of naive bayes :  0.2946947, 0.5359116
## Optimal cutoff, Maximum F1 score of classfication tree :  0.43, 0.5194805
## Optimal cutoff, Maximum F1 score of logistic regression :  0.2898158, 0.5360231
```

3 개의 모델의 f1 score 이 약 0.51~0.53 으로 큰 차이를 보이지 않지만 cutoff 가 0.2898158 일 때 0.5360231 을 f1 score 으로 가지는 logistic regression 이 가장 좋은 모델이라고 평가할 수 있다.

5. Write your conclusions and discussion.

Default 데이터를 naive bayes, classification tree, logistic regression 3 가지 모델로 적합시켰다. 그리고 AUC, Accuracy, F1 score 3 가지 통계량을 가지고 모델들을 평가하였다. AUC 를 기준으로 했을 때, Accuracy 를 기준으로 했을 때, F1 score 를 기준으로 했을 때 모두 logistic regression 이 가장 좋은 결과를 가진다. logistic regression 이 유일하게 계수의 통계적 검정이 가능한데 성능까지 가장 좋으니 3 개의 모델 중에선 가장 좋은 모델이라고 평가할 수 있다.

하지만 이 데이터 불균형이 심한 편이라 AUC, Accuracy 가 모델을 평가하기에 유효한 통계량인지 의심할 필요가 있다. logistic regression 의 F1 score 가 가장 높지만 0.5360231 밖에 되지 않아 좋다고 볼 수 없다('Yes'인 case 를 제대로 검출할 수 없다!). 그러므로 logistic regression 이 3 개 중엔 가장 좋지만 이 데이터를 대변하기에는 좋지 못한 모델이라고 생각한다.

따라서 데이터를 더 잘 설명하기 위해서 또는 더 좋은 성능을 얻기 위해서는 더 복잡한 모델을 사용할 필요가 있다. 뿐만 아니라 데이터가 불균형하다는 문제점이 있으므로 resampling 이나 oversampling 등의 방법을 적용한 뒤 모델을 학습하는 것도 좋은 방법이라고 생각한다.

Appendix : R codes

1. Construct (a) a naive Bayes classifier, (b) a classification tree classifier, and (c) a logistic regression classifier.

```
library(ISLR)

data = ISLR::Default

n = dim(data)[1]
train_size = 0.5
student = 20152410

set.seed(student)
train_index = sample(1:n, n*train_size, replace=FALSE)
train = data[train_index, ]
test = data[-train_index, ]
```

(a) a naive Bayes classifier

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.6.3
```

```
model_nb = naiveBayes(default ~ ., data=train)
pred_nb = predict(model_nb, newdata=test, type='raw')
model_nb
```

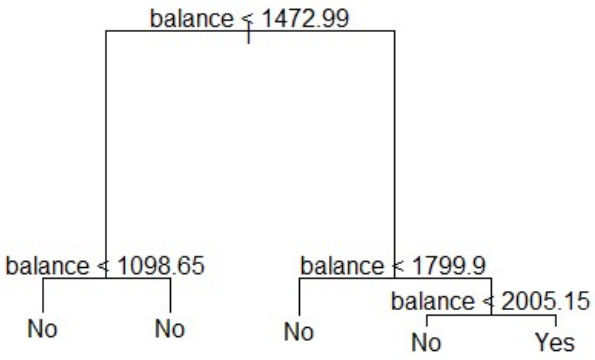
```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##     No    Yes
## 0.9666 0.0334
##
## Conditional probabilities:
##      student
## Y            No       Yes
##   No  0.7020484 0.2979516
##   Yes 0.6706587 0.3293413
##
##      balance
## Y          [,1]     [,2]
##   No   804.1743 451.4752
##   Yes 1744.2139 309.7295
##
##      income
## Y         [,1]     [,2]
##   No  33336.39 13336.79
##   Yes 32960.67 13787.46
```

(b) a classification tree classifier

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 3.6.3
```

```
model_tree = tree(default ~ ., data=train, split='deviance')
pred_tree = predict(model_tree, newdata=test)
plot(model_tree)
text(model_tree)
```



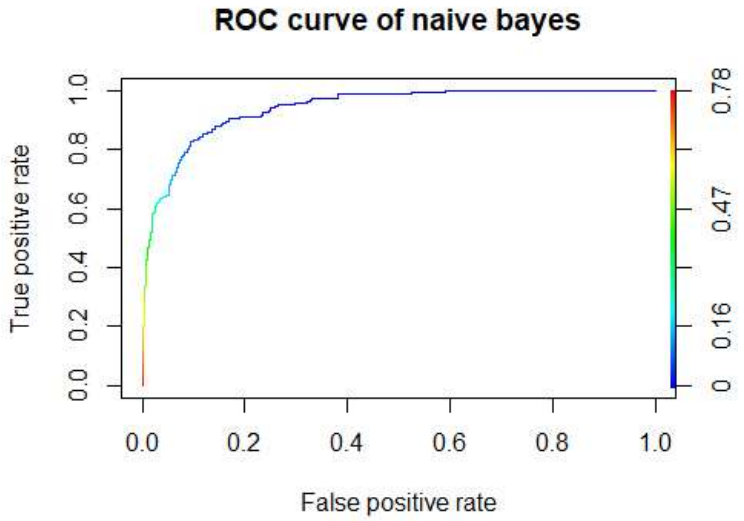
(c) a logistic regression classifier

```
model_logit = glm(default ~ ., data=train, family='binomial')
pred_logit = predict(model_logit, newdata=test, type='response')
summary(model_logit)
```

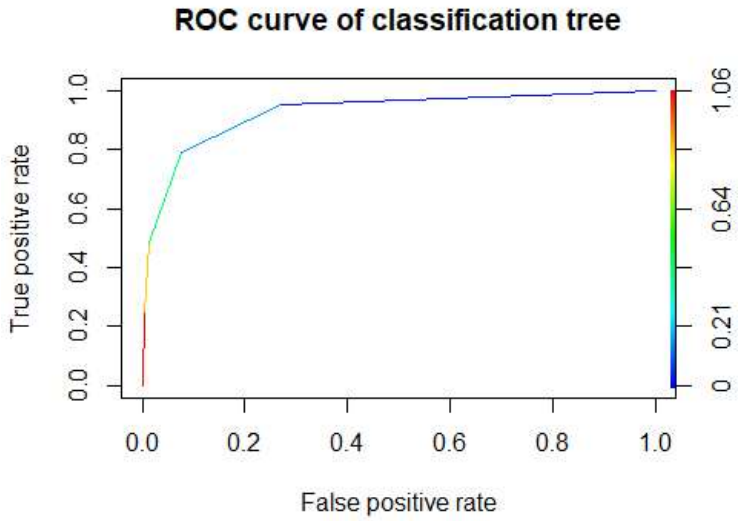
```
##
## Call:
## glm(formula = default ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2136  -0.1307  -0.0465  -0.0166   3.8036
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.153e+01  7.333e-01 -15.726  < 2e-16 ***
## studentYes  -9.303e-01  3.333e-01  -2.791  0.00525 **
## balance      6.188e-03  3.555e-04  17.407  < 2e-16 ***
## income       5.695e-06  1.141e-05   0.499  0.61767
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1463.7  on 4999  degrees of freedom
## Residual deviance:  752.3  on 4996  degrees of freedom
## AIC: 760.3
##
## Number of Fisher Scoring iterations: 8
```

2. For each classifier, make an ROC curve, calculate the AUC, and compare the three classifiers.

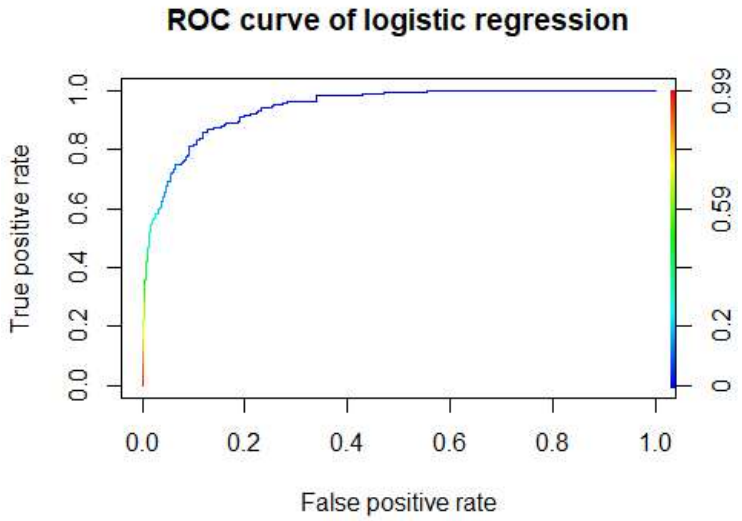
```
library(ROCR) # ROC curve of 3 classifiers
prediction_nb = prediction(pred_nb[, 'Yes'], test$default)
performance_nb = performance(prediction_nb, 'tpr', 'fpr')
plot(performance_nb, main='ROC curve of naive bayes', colorize=TRUE)
```




```
prediction_tree = prediction(pred_tree[, 'Yes'], test$default)
performance_tree = performance(prediction_tree, 'tpr', 'fpr')
plot(performance_tree, main='ROC curve of classification tree', colorize=TRUE)
```



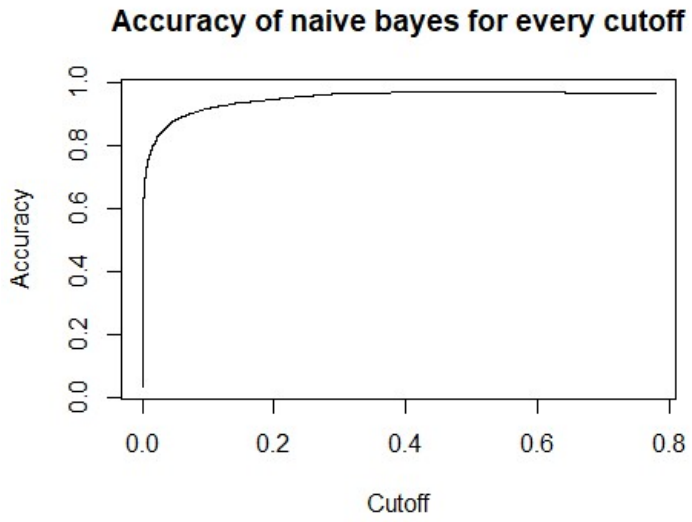
```
prediction_logit = prediction(pred_logit, test$default)
performance_logit = performance(prediction_logit, 'tpr', 'fpr')
plot(performance_logit, main='ROC curve of logistic regression', colorize=TRUE)
```



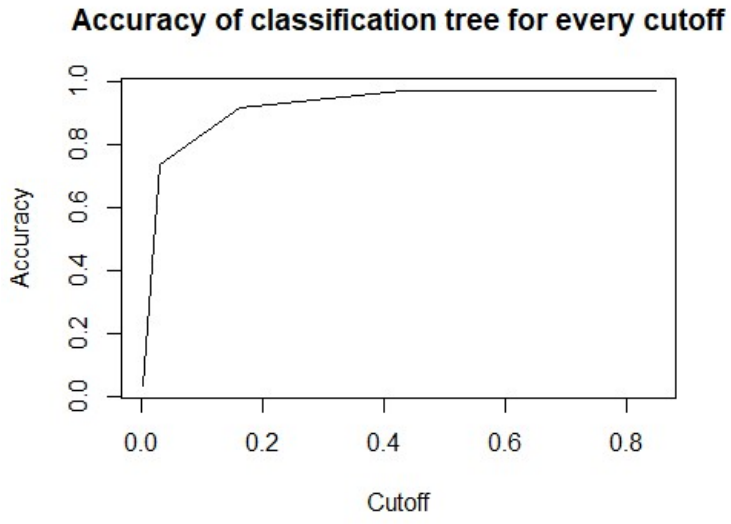

```
# AUC of 3 classifiers
auc_nb = performance(prediction_nb, 'auc')
auc_tree = performance(prediction_tree, 'auc')
auc_logit = performance(prediction_logit, 'auc')

cat('AUC of naive bayes : ', auc_nb@y.values[[1]])
cat('AUC of classification tree : ', auc_tree@y.values[[1]])
cat('AUC of logistic regression : ', auc_logit@y.values[[1]])


## AUC of naive bayes :  0.9409379
## AUC of classification tree :  0.9248116
## AUC of logistic regression :  0.941824
```

3. For each classifier, find the optimal cutoff value to maximize the accuracy. Compare the three classifiers.

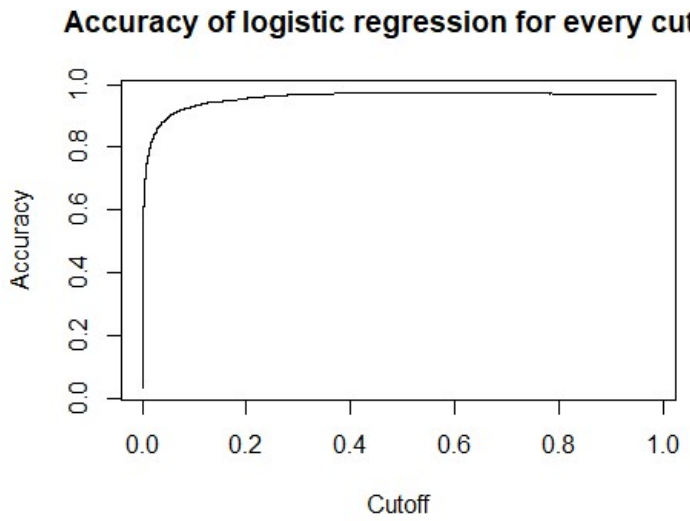
```
acc_nb = performance(prediction_nb, 'acc', 'cutoff')
plot(acc_nb, main='Accuracy of naive bayes for every cutoff')
```




```
acc_tree = performance(prediction_tree, 'acc', 'cutoff')
plot(acc_tree, main='Accuracy of classification tree for every cutoff')
```



```
acc_logit = performance(prediction_logit, 'acc', 'cutoff')
plot(acc_logit, main='Accuracy of logistic regression for every cutoff')
```

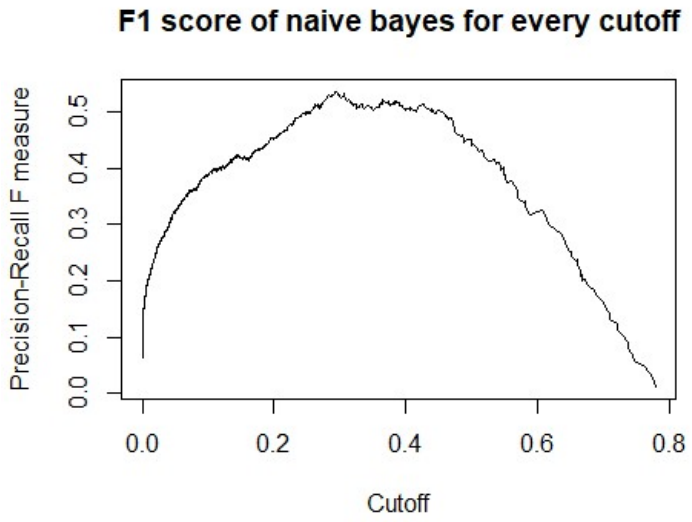



```
cat('Optimal cutoff of naive bayes : ', acc_nb@x.values[[1]][which.max(acc_nb@y.val
ues[[1]])])
cat('Maximum accuracy of naive bayes : ', max(acc_nb@y.values[[1]]))
cat('Optimal cutoff of classification tree : ', acc_tree@x.values[[1]][which.max(ac
c_tree@y.values[[1]])])
cat('Maximum accuracy of classification tree : ', max(acc_tree@y.values[[1]]))
cat('Optimal cutoff of logistic regression : ', acc_logit@x.values[[1]][which.max(a
cc_logit@y.values[[1]])])
cat('Maximum accuracy of logistic regression : ', max(acc_logit@y.values[[1]]))
```

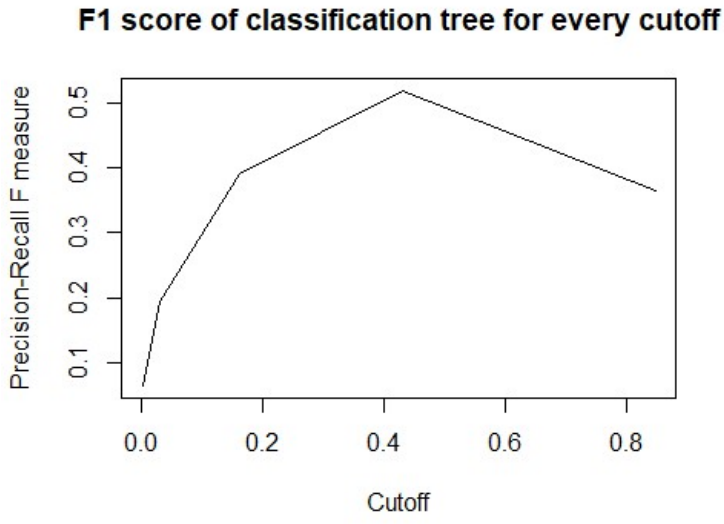
```
## Optimal cutoff of naive bayes :  0.4706127
## Maximum accuracy of naive bayes :  0.9734

## Optimal cutoff of classification tree :  0.8478261
## Maximum accuracy of classification tree :  0.9716

## Optimal cutoff of logistic regression :  0.5119097
## Maximum accuracy of logistic regression :  0.9736
```

4. For each classifier, find the optimal cutoff value to maximize the F1 score. Compare the three classifiers.

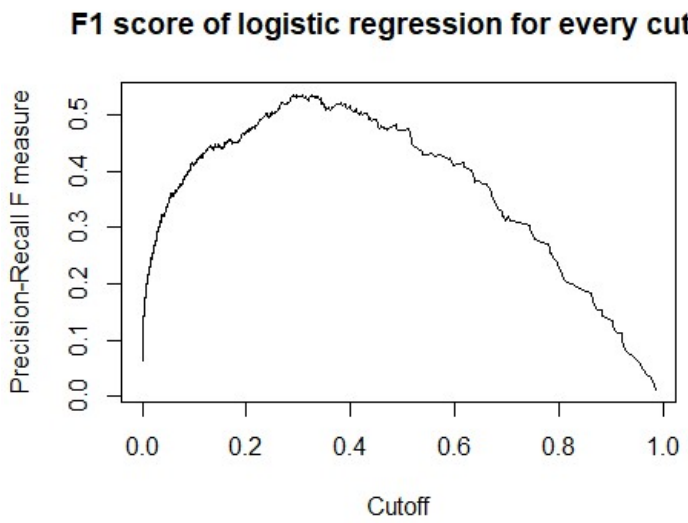
```
f1_nb = performance(prediction_nb, 'f', 'cutoff')
plot(f1_nb, main='F1 score of naive bayes for every cutoff')
```




```
f1_tree = performance(prediction_tree, 'f', 'cutoff')
plot(f1_tree, main='F1 score of classification tree for every cutoff')
```



```r
f1_logit = performance(prediction_logit, 'f', 'cutoff')
plot(f1_logit, main='F1 score of logistic regression for every cutoff')
```




```r
cat('Optimal cutoff of naive bayes : ', f1_nb@x.values[[1]][which.max(f1_nb@y.values[[1]])])
cat('Maximum F1 score of naive bayes : ', max(f1_nb@y.values[[1]], na.rm=TRUE))
cat('Optimal cutoff of classfication tree : ', f1_tree@x.values[[1]][which.max(f1_tree@y.values[[1]])])
cat('Maximum F1 score of classfication tree : ', max(f1_tree@y.values[[1]], na.rm=TRUE))
cat('Optimal cutoff of logistic regression : ', f1_logit@x.values[[1]][which.max(f1_logit@y.values[[1]])])
cat('Maximum F1 score of logistic regression : ', max(f1_logit@y.values[[1]], na.rm=TRUE))
```

```
## Optimal cutoff of naive bayes :  0.2946947
## Maximum F1 score of naive bayes :  0.5359116

## Optimal cutoff of classfication tree :  0.43
## Maximum F1 score of classfication tree :  0.5194805

## Optimal cutoff of logistic regression :  0.2898158
## Maximum F1 score of logistic regression :  0.5360231
```