Classification

Eltecon Data Science Course by Emarsys

János Divényi

November 25, 2020

Homeworks from last week

- Presenters:
  - Bat-Erdene, Boldmaa - Kashirin, Andrey
  - Im Seongwon - Kim Yeonggyeong
  - Szőnyi Máté - Tran, Dung

Goal of the lesson

- introduce **decision trees** as nonlinear classifiers
- measure the performance of classification models by the **ROC curve**

Section 1

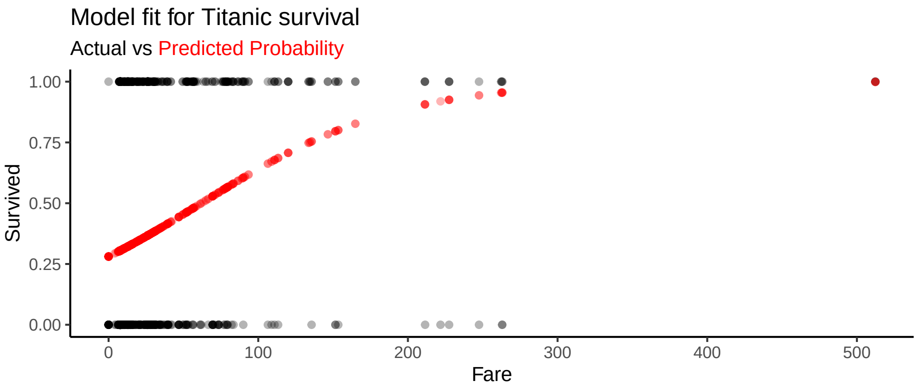
Classification

Recap: logistic regression to predict Titanic-survival

```
model <- glm(
    Survived ~ Fare,
    data = titanic_train,
    family = binomial(link = "logit")
)
```

```
predicted_prob <- predict.glm(
  model,
  newdata = titanic_train,
  type = "response"
)
```

Predictive fit



Evaluating binary models - Accuracy

```
calculateAccuracy <- function(actual, predicted) {
    N <- length(actual)
    accuracy <- sum(actual == predicted) / N

    return(accuracy)
}
```

```
predicted_class <- ifelse(predicted_prob > 0.5, 1, 0)
calculateAccuracy(titanic_train$Survived, predicted_class)
```

```
## [1] 0.6655
```

Evaluating binary models - Confusion Matrix

```r
table(
  titanic_train$Survived,
  predicted_class,
  dnn = c("actual", "predicted")
)
```

```
##        predicted
## actual    0   1
##       0 511  38
##       1 260  82
```

Non-linear classification: Decision Tree

Visual explanation by r2d3

Quiz

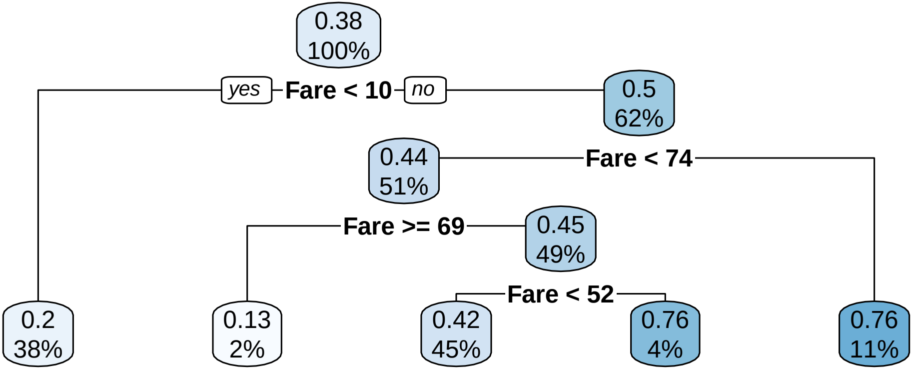
Estimate a decision tree model

```r
tree_model <- rpart(
    Survived ~ Fare, data = titanic_train
)
```

```
## n= 891
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 891 210.700 0.3838
##    2) Fare< 10.48 339  53.760 0.1976 *
##    3) Fare>=10.48 552 138.000 0.4982
##      6) Fare< 74.38 455 112.200 0.4418
##       12) Fare>=69.42 15   1.733 0.1333 *
##       13) Fare< 69.42 440 109.000 0.4523
##         26) Fare< 52.28 403  98.440 0.4243 *
##         27) Fare>=52.28 37   6.811 0.7568 *
##      7) Fare>=74.38 97  17.550 0.7629 *
```

Visualize

```
rpart.plot(tree_model)
```



Evaluate

```r
predicted_prob_tree <- predict(tree_model, newdata = titanic_train)
calculateAccuracy(titanic_train$Survived, predicted_prob_tree > 0.5
```

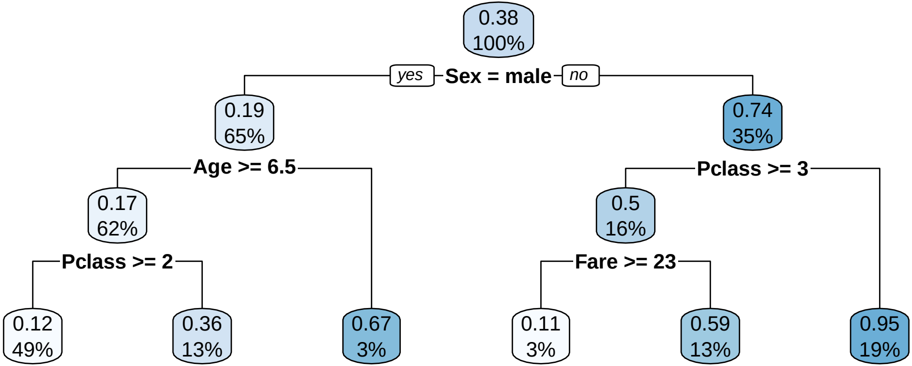
```
## [1] 0.6947
```

```r
table(
  titanic_train$Survived,
  predicted_prob_tree > 0.5,
  dnn = c("actual", "predicted")
)
```

```
##        predicted
## actual FALSE TRUE
##      0   517   32
##      1   240  102
```

Include other variables

```
extended_tree <- rpart(
    Survived ~ Fare + Sex + Age + Pclass, data = titanic_trai
)
rpart.plot(extended_tree)
```



Including more variable helps

```r
calculateAccuracy(
    titanic_train$Survived,
    predict(extended_tree) > 0.5
)
```

```
## [1] 0.8193
```

Including more variable helps

```
calculateAccuracy(
    titanic_train$Survived,
    predict(extended_tree) > 0.5
)
```

```
## [1] 0.8193
```

. . . or does it?

Including more variables helps

```
calculateAccuracy(
    titanic_train$Survived,
    predict(extended_tree) > 0.5
)
```

```
## [1] 0.8193
```

. . . or does it?

Recall: we have to evaluate the performance on a **different set of data** to avoid overfitting

Classify spam by decision trees

Recall from week 9

```
data <- fread("../week 8-10/data/spam_clean.csv")

# Seperate train-test set
train_proportion <- 0.8
n <- nrow(data)
set.seed(1234)
train_index <- sample(1:n, floor(n * train_proportion))

data_to_use <- data[, -c(2, 50:400)]  # exclude columns to speed up
data_train <- data_to_use[train_index,]
data_test <- data_to_use[-train_index,]
```

Estimate logistic regression as benchmark

```
spam_logit <- glm(
    is_spam ~ .,
    data = data_train,
    family = binomial(link = "logit")
)
```

Accuracy evaluated on a test set:

```
predicted_probs <- predict(spam_logit, newdata = data_test, ty
calculateAccuracy(
    data_test$is_spam,
    predicted_probs > 0.5
)
```

```
## [1] 0.9534
```

Tree model

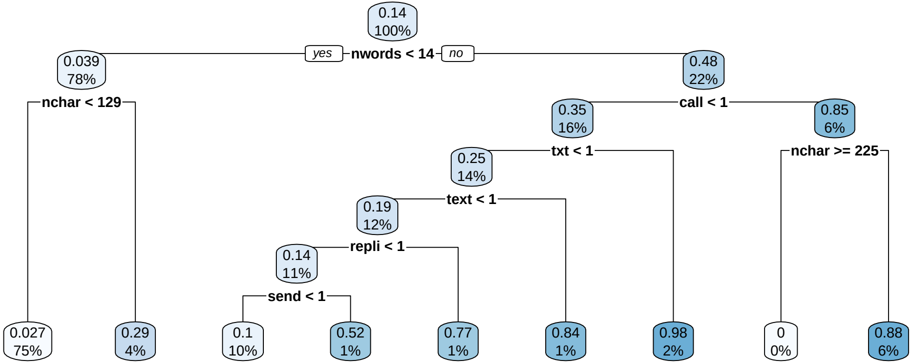
```
spam_tree <- rpart(
    is_spam ~ .,
    data = data_train
)
```

```
predicted_probs <- predict(spam_tree, newdata = data_test)
calculateAccuracy(
    data_test$is_spam,
    predicted_probs > 0.5
)
```

```
## [1] 0.9417
```

Performs worse but is easier to interpret

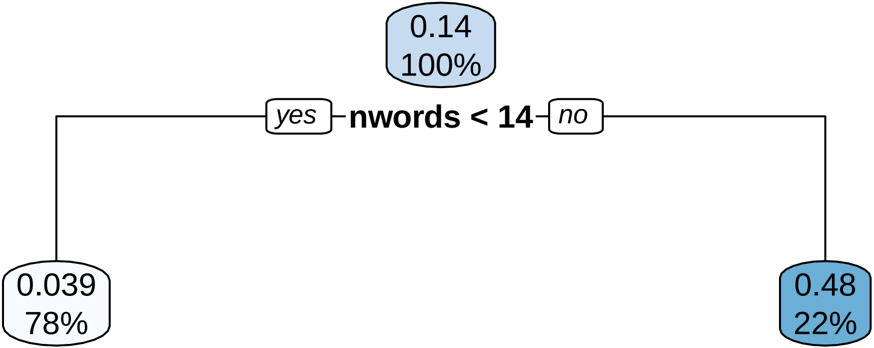
```
rpart.plot(spam_tree)
```



Tree "pruning"

The tree depth is controlled by the complexity parameter cp

```
rpart.plot(prune(spam_tree, cp = 0.1))
```



Overfitting

Estimate a "full" tree

```
spam_full_tree <- rpart(
    is_spam ~ .,
    data = data_train,
    control = rpart.control(
        minsplit = 2, minbucket = 1, cp = 0
    )
)
```

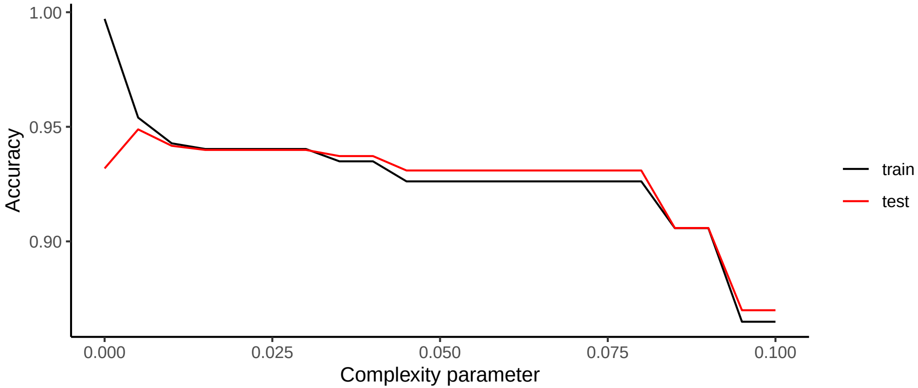
```
calculateAccuracy(
    data_train$is_spam,
    predict(spam_full_tree) > 0.5
)
```

```
## [1] 0.9971
```

Compare performance on train and test set

```
accuracy_by_params <- map_df(seq(0, 0.1, 0.005), ~{
    pruned_tree <- prune(spam_full_tree, cp = .x)
    data.table(
        cp = .x,
        train = calculateAccuracy(data_train$is_spam, predict(pruned_tree, data_trai
        test = calculateAccuracy(data_test$is_spam, predict(pruned_tree, data_test)
    )
})
```

Compare performance on train and test set



Section 2

Evaluate binary classification performance

Accuracy might not be that informative

- *"PCR-tests have above 95% accuracy".* - What does that mean?

Accuracy might not be that informative

- *"PCR-tests have above 95% accuracy".* - What does that mean?
- I can always deliver a 99%+ accurate model to predict who will buy – until the purchase rate remains below 1% as usual (predicting no one will buy)

Accuracy might not be that informative

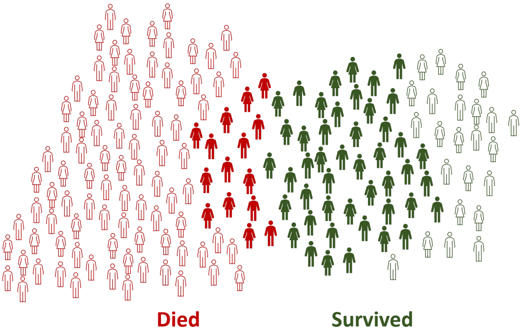
- *"PCR-tests have above 95% accuracy".* - What does that mean?
- I can always deliver a 99%+ accurate model to predict who will buy – until the purchase rate remains below 1% as usual (predicting no one will buy)
- Confusion matrix provides more detailed information by comparing actual and predicted labels

Confusion matrix

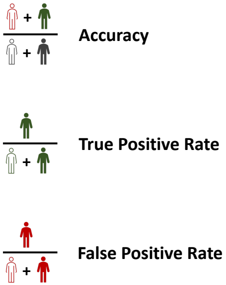
Recall the confusion matrix of the Titanic prediction task using the logistic regression model:

```
##        predicted
## actual FALSE TRUE
##      0   511   38
##      1   260   82
```

True Positive and False Positive Rate



predicted to survive



True Positive and False Positive Rate

Recall the confusion matrix of the Titanic prediction task using the glm:

```
##       predicted
## actual FALSE TRUE
##      0   511   38
##      1   260   82
```

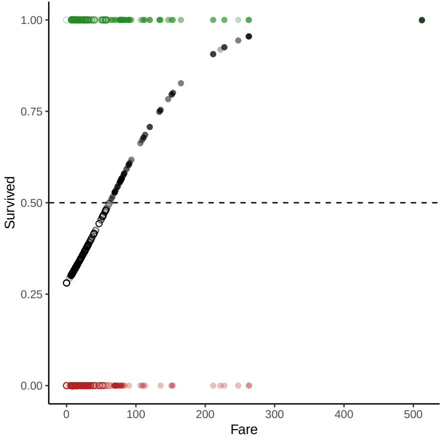
- True Positive Rate: `82/(260 + 82) = 23.98%`
- False Positive Rate: `38/(511 + 38) = 6.9%`

There is a trade-off between TPR and FPR

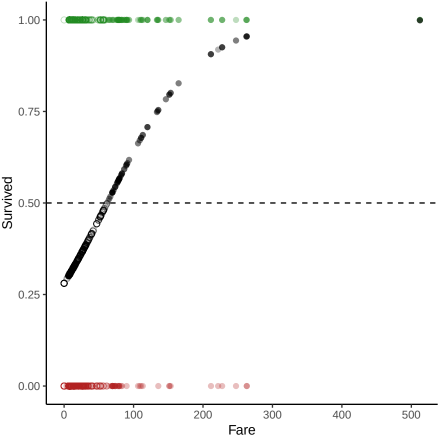
- Getting easier* about classifying someone as positive (or as a survivor) would definitely increase TPR - but also the FPR
  - It is easy to reach 100% true positive rate: just predict positive for everyone
- This trade-off is expressed by the ROC curve

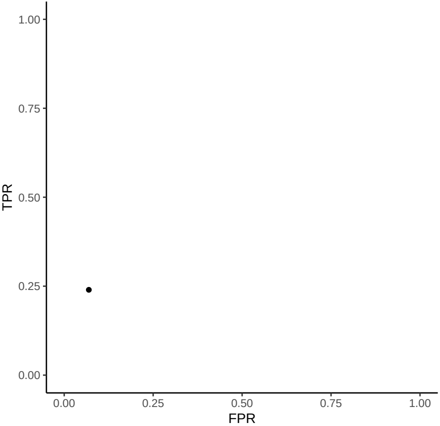
* just decrease the probability cutoff that we defaulted to 0.5

ROC plot

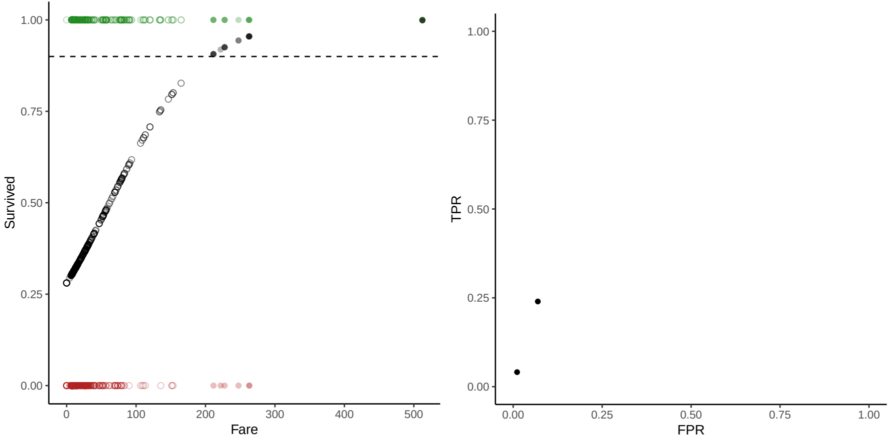


ROC plot

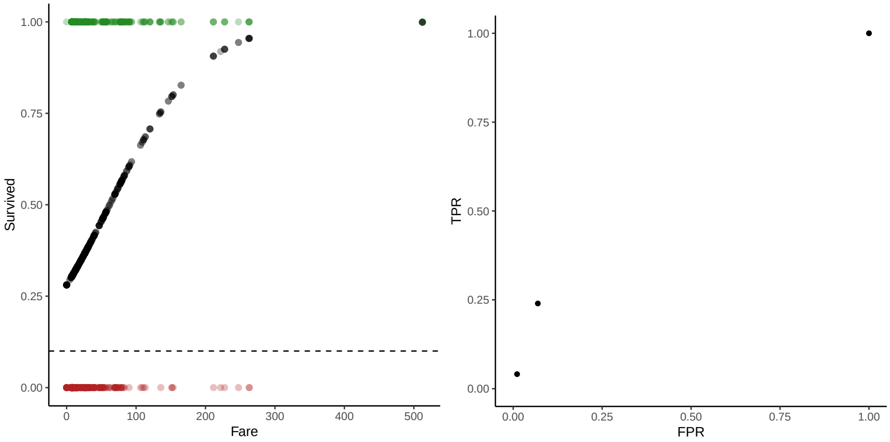

1.00
0.75
0.50
0.25
0.00
Survived
0
100
200
300
400
500
Fare


1.00
0.75
0.50
0.25
0.00
TPR
0.00
0.25
0.50
0.75
1.00
FPR

ROC plot

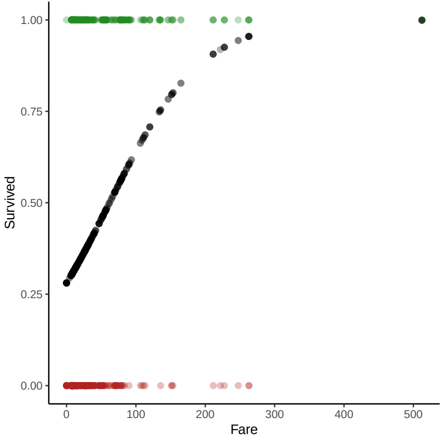

1.00
0.75
0.50
0.25
0.00
Survived
0
100
200
300
400
500
Fare
TPR
FPR
0.00
0.25
0.50
0.75
1.00

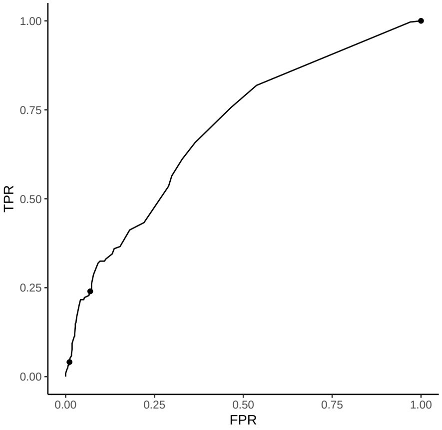
ROC plot



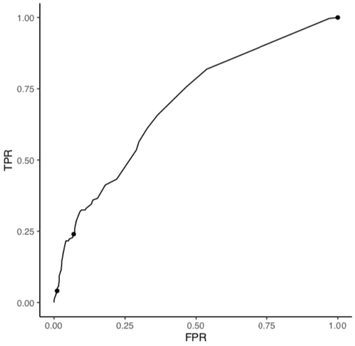
Survived
1.00
0.75
0.50
0.25
0.00
0
100
200
300
400
500
Fare
TPR
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
FPR

ROC plot

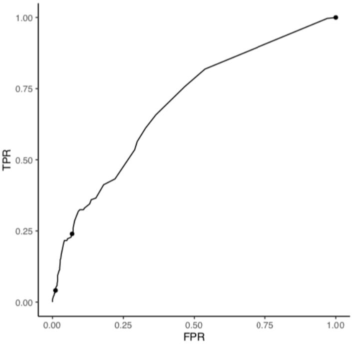

Survived
1.00
0.75
0.50
0.25
0.00
0
100
200
300
400
500
Fare

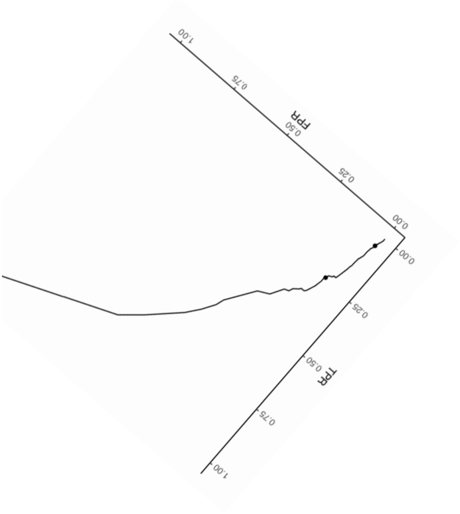

TPR
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
FPR

ROC plot

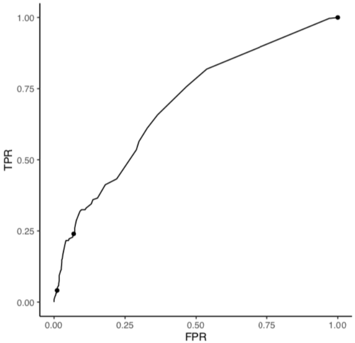


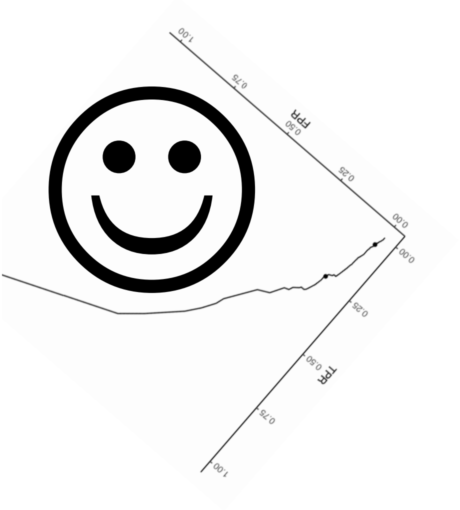
ROC plot


1.00
0.75
0.50
0.25
0.00
TPR
0.00
0.25
0.50
0.75
1.00
FPR


1.00
0.75
0.50
0.25
0.00
FPR
0.00
0.25
0.50
0.75
1.00
TPR

ROC plot





Quiz

ROC plot - live coding

Homework

- Work on your final project
- Remember: the first version of your written project is due on **4th December**

Resources

- Gareth J., Witten D., Hastie T. and Tibshirani R.: An Introduction to Statistical Learning Chapter 8.
- Machine Learning meets economics: https://blog.mldb.ai/blog/posts/2016/01/ml-meets-economics/
- FPR, TPR: https://www.youtube.com/watch?v=sunUKFXMHGk (StatQuest)
- ROC curve: https://www.youtube.com/watch?v=4jRBRDbJemM (StatQuest)

Thank you & Feedback