Examination

Linköping University, Department of Computer and Information Science, Statistics

Course code and name	TDDE01 Machine Learning
Date and time	2022-08-24, 8.00-13.00
Assisting teacher	Oleg Sysoev
Allowed aids	PDF of the course book + your help file (if submitted to LISAM in due time)
Grades:	
	5=18-20 points
	4=14-17 points
	3=10-13 points
	U=0-9 points

Provide a detailed report that includes plots, conclusions and interpretations. Give motivated answers to the questions. If an answer is not motivated, the points are reduced. Provide all necessary codes in the appendix.

Note: seed 12345 should be used in all codes that assumes randomness unless stated otherwise!

To start work in RStudio, type this in the Terminal application:

```
module add courses/TDDE01
rstudio
```

To submit your report:

1. Create one file (DOC, DOCX, ODT, PDF)
2. Use Exam Client to submit, and choose Assignment 1 in the drop box
3. Attach your report
4. Submit.
5. "Request Received" status implies that your report is successfully submitted.

Assignment 1 (10p)

The data file **rollercoasters.csv** contains information about various roller coasters from various amusement parks.

1. As you might observe, there are some data missing in the Duration column. Split the rows without missing data randomly into training and test (60/40). Compute an optimal decision tree with Duration as target and all other numerical columns as features by using the holdout principle. Report the graph showing the dependence of the training and validation error on the amount of leaves and report the amount of leaves in the optimal tree and which variables are actually used by the tree. Finally, use the optimal tree model to impute (i.e. fill in) the Duration values in the original data where they are missing and print first 10 observations from the updated Duration column. **(4p)**
2. Select rows nr "1" – nr "x" from the training and test data created in step 1, where x=20,25,30,…, 60. For each pair of training test/subsamples compute a decision tree with 2 leaves. Plot a dependence of the training and test errors on x and comment why the trends you observe look as they do. **(3p)**
3. Use the imputed original data computed from step 4 to fit a logistic regression model with Track as target and all numerical columns as inputs. Report the estimated probabilistic model. Assuming "Wood" to be the positive class, report the accuracy and F1-score. Is this model good according to the accuracy and F1? Which of these two numbers should we trust more here and why? **(3p)**
    a. **Hint: If you were unable to compute imputations in step 1, impute missing values with the mean of non-missing Duration values first, but the points will be reduced.**

Assignment 2 (10p)

KERNEL MODELS – 5 POINTS

In the course, you have learned about kernel models for classification and regression. Kernel models can also be used for density estimation, i.e. to model a probability distribution or density function p(x_*). In particular,

$$p(x_*) = \frac{1}{n}\sum_{i=1}^{n} k(x_*, x_i, h)$$

where the kernel function k() must integrate to 1. To ensure this, you will hereinafter consider k() to be the density function of a Gaussian distribution with mean equal to the training point x_i and standard deviation equal to the kernel width h, which is evaluated at the target point x_*. You can get it by using the command `dnorm` in R.

Run the code below to produce some training data consisting of 1500 samples from class 1 and 1000 samples from class 2. These points are stored in the variables data_class1 and data_class2. Sample 1500 new points from class 1 and 1000 new points from class 2. This is the validation data. Implement the kernel model presented above to estimate the density function of the data sampled from class 1. The kernel model should use only the training data. **Comment your code.** Using your implementation of the

kernel model, implement a function that computes the log likelihood of some data sampled from class 1 as function of *h*. Use this function to select the *h* value that maximizes the log likelihood of the training data from class 1. Use it again to select the *h* value that maximizes the log likelihood of the validation data from class 1. Do these two values coincide? Why or why not? Finally, answer the following question: Once you have kernel models for p(x_*|class=1) and p(x_*|class=2), how would you use them to produce posterior class probabilities, i.e. p(class|x_*)? You don't need to implement the answer.

```
set.seed(123456789)

N_class1 <- 1500

N_class2 <- 1000


data_class1 <- NULL

for(i in 1:N_class1){

  a <- rbinom(n = 1, size = 1, prob = 0.3)

  b <- rnorm(n = 1, mean = 15, sd = 3) * a + (1-a) * rnorm(n = 1, mean = 4, sd
= 2)

  data_class1 <- c(data_class1,b)

}


data_class2 <- NULL

for(i in 1:N_class2){

  a <- rbinom(n = 1, size = 1, prob = 0.4)

  b <- rnorm(n = 1, mean = 10, sd = 5) * a + (1-a) * rnorm(n = 1, mean = 15,
sd = 2)

  data_class2 <- c(data_class2,b)

}
```

NEURAL NETWORKS – 5 POINTS

In this assignment, you are asked to use the R package `neuralnet` to train a NN to learn the trigonometric sine function. To produce the learning data, sample 50 points uniformly at random in the interval [0, 10] and, then, apply the sine function to each point.

Your task is to estimate the generalization mean squared error of a NN with a single hidden layer of 10 units for the regression task described above. Use cross-validation with 2 folds. For the training, initialize the weights of the NN to random values in the interval [-1, 1]. Stop the training when the partial derivatives of the error function are below a threshold value of 0.001. **Comment your code.** Recall that cross-validation works as follows:

1. Divide the learning data into approximately equal sized folds D1 and D2.
2. Train the regressor on D1 and test it on D2.
3. Train the regressor on D2 and test it on D1.
4. Report the average error on the test folds.

Finally, name one advantage and one disadvantage of using cross-validation to estimate the generalization error.

Hint: Check the argument `threshold` in the documentation. Use the function `predict()` to compute the output of the trained NN for a given input vector. Use the default values for the arguments not mentioned here. Feel free to use the following template.

```
library(neuralnet)
set.seed(1234567890)

Var <- runif(50, 0, 10)
tr <- data.frame(Var, Sin=sin(Var))
tr1 <- tr[1:25,] # Fold 1
tr2 <- tr[26:50,] # Fold 2
```