# TDT4300
# Assignment 3
# Tino Lazreg

**1)**

## Decision Trees

`1 - (8/20)^2 - (12/20)^2 = 0,48`

**2)**

userID is uniquely identified, therefore if we split userID we will create nodes with 1 element each.

Gini of each node is equal 0, so total GINI for userID will be 0.

**3)**

| Age | total | yes | no | Gini |
|---|---|---|---|---|
| young | 7 | 4 | 3 | `1 - (4/7)^2 - (3/7)^2 = (24/49)` |
| Medium young | 6 | 5 | 1 | `1 - (5/6)^2 - (1/6)^2 = (5/16)` |
| old | 7 | 3 | 4 | `1 - (3/7)^2 - (4/7)^2 = (24/49)` |
| Total Gini: 7/20 * 24/49 + 6/20 * 5/16 + 7/20 * 24/49 = 0,4366 | | | | |

**4)**

| Student | total | yes | no | Gini |
|---|---|---|---|---|
| Yes | 10 | 8 | 2 | `1 – (8/10)^2 - (2/10)^2 = (8/25)` |
| No | 10 | 4 | 6 | `1 - (4/10)^2 - (6/10)^2 = (12/25)` |
| Total Gini: 10/20 * 8/25 + 10/20 * 12/25 = 0,4 | | | | |

**5)**

| Creditworthiness | total | yes | no | Gini |
|---|---|---|---|---|
| Pass | 10 | 6 | 4 | `1 – (6/10)^2 - (4/10)^2 = (12/25)` |
| High | 10 | 6 | 4 | `1 - (6/10)^2 - (4/10)^2 = (12/25)` |
| Total Gini: 10/20 * 12/25 + 10/20 * 12/25 = 0,48 | | | | |

**6)**

We wish to split on the attribute that gives us the most information. This means that we should pick the attribute which gives us the smallest Gini value. That would be UserID, but UserID doesn't really have any information value, because it is uniquely identified. The best attribute is then Student, since 0,4 is the lowest Gini value.

**7)**

I could make a decision tree with an algorithm, and then use that to find the answer . The dataset is not very large, so it is easy to just do it manually.

For user 21, we only have two records that fit the requirements, ID-11 and ID-20, and they both bought a PC. We should let user 21 buy a PC.

For user 22 ,we don't have any matches, so we shouldn't let him buy a PC.

## 2)

### Datasets

| Datasets | Iris | Diabetes | Spambase |
|---|---|---|---|
| Variables | 5 | 9 | 58 |
| Instances | 150 | 768 | 4601 |
| Type of data | 4 : Numeric, 1 : Nominal | 8: Numeric, 1 : Nominal | 57 : Numeric, 1 : Nominal |
| Most difficult | petallength | pedi | capital_run_length_average |

I determined most difficult based on which variable had the most distinct attributes.

## Classification

| Datasets | Iris | Diabetes | Spambase |
|---|---|---|---|
| J48 | confidence = 0,25 : 96,0784 %<br>confidence = 0,75 : 96,0784 % | confidence = 0,25 : 76,2452 %<br>confidence = 0,75 : 75,0958 % | confidence = 0,25 : 92,1995 %<br>confidence = 0,75 : 92,0716 % |
| k-NN | k = 1 : 96,0784 %<br>k = 5 : 98,0392 %<br>k = 10 : 96,0784 % | k = 1 : 72,7969 %<br>k = 5 : 75,0958 %<br>k = 10 : 74, 7126 % | k = 1 : 89,0026 %<br>k = 5 : 89,2583 %<br>k = 10 : 88,8747 % |
| SVM | c = 1 : 96,0784 %<br>c = 5 : 98,0392 %<br>c = 10 : 98,0392 % | c = 1 : 79,3101 %<br>c = 5 : 79,3103 %<br>c = 10 : 79,6935 % | c = 1 : 90,5371 %<br>c = 5 : 91,3683 %<br>c = 10 : 92,3274 % |

## Evalutation

In cross-validation, each record is used the same number of times for training and exactly once for testing. There are different methods to this approach. Two-fold cross validation, partion the data into two equal-sized subsets. We then choose one subset for testing and one for training, and then we swap the roles. The total errors are obtained by summing up the errors for both runs.

Another method is *k*-fold cross-validation where the data is partitioned in *k* equal-sized sets. During each run, one set is chosen for testing, and the rest are used for training. This gets repeated until every set has been used once for testing. The advantages of this method is that it uses most of the data for training, while every set gets tested at one point. The disadvantages is that it is computionally expensive because the procedure is repeated *k* times so each set is used for testing once, which can be expensive if *k* is significantly large.

Cross-validation might be a better approach with complex datasets because it limits problems like overfitting with its extensive runs of training sets and testing.

Sorted data sets can cause problems depending for cross-validation depending on how partitioning is done. If it is done randomly, ordering won't matter, but if it is done sequentially, it could cause problems. The partition size is also important, if we have many partitions, the k-fold cross-validation method will make several runs, and the fact that the data set is sorted won't be a big problem.

If we have three classes and perform a percentage split of 66%, that means that for the traning set 66 % of the instances in the data set is used, and the rest is used for testing. This will not be reasonable because the third class will mainly be used for testing, and the training set will be overrepresented by the first class.

## Best Classifiers

Of the three algorithms I used, I think SVM reported the best results. J48 is also very close. RandomForest reported better results than the three classifiers I used.