Assignment 3

Machine Learning
MSc Business Analytics

Wolfram Wiesemann

1 Individual Assignment

Instructions: *This exercise should be done "by hand", that is, not using R or Python. All necessary calculations should be included in the submission, as well as brief explanations of what you do.*

The training data set in Table 1 on the next page provides a summary of the traffic conditions experienced on a major road across several days, times and weather conditions.

1. Create a full classification tree that estimates the traffic based on the day of the week, time of the day and the weather condition. Use the entropy as a purity measure to guide your splitting choices.

2. Construct a confusion matrix for the performance of your tree on the training data. What is the misclassification rate, and what is the sensitivity and specificity (assuming that 'yes' is the 'important' class)?

3. Construct the confusion matrix that results if your algorithm is applied to the test set in Table 2 on page 3.

day	**weather**	**time**	**traffic**
weekday	sunny	1pm	no
weekday	rainy	1pm	yes
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekday	rainy	1pm	yes
weekday	sunny	8am	no
weekend	sunny	8am	yes
weekend	sunny	1pm	yes
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekday	sunny	1pm	no
weekend	rainy	1pm	yes
weekday	rainy	1pm	yes
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekend	sunny	1pm	yes
weekday	rainy	8am	yes
weekday	sunny	8am	no
weekday	sunny	8am	no
weekday	sunny	1pm	no
weekday	sunny	8am	yes
weekend	rainy	8am	no
weekday	sunny	1pm	no
weekday	rainy	8am	yes
weekday	sunny	8am	yes

Table 1: Training set for the individual assignment.

day	**weather**	**time**	**traffic**
weekend	rainy	8am	no
weekday	sunny	8am	yes
weekend	sunny	1pm	yes
weekday	sunny	8am	no
weekend	sunny	1pm	yes
weekday	rainy	8am	no
weekday	sunny	8am	yes
weekday	sunny	1pm	no
weekday	sunny	1pm	no
weekday	sunny	1pm	no
weekend	rainy	8am	yes
weekday	sunny	8am	yes
weekday	sunny	1pm	no
weekday	rainy	1pm	yes
weekday	sunny	1pm	no

Table 2: Test set for the individual assignment.

2 Group Assignment

For this group assignment, download the loan data set[1] from

https://www.lendingclub.com/info/download-data.action

and follow the steps below to build a classification tree. This is a data set of loans that were given out to clients. We will attempt to identify clients with a high probability of default. This is a tough real-word data set, as those clients that are most likely to default have already been filtered out by the banks – the data set solely consists of loans that were actually approved!

2.1 Pre-Processing

Real-world data sets rarely come in a convenient format, and some form of data pre-processing is typically necessary. For this exercise, we want to undertake the following pre-processing steps:

1. We are going to base our prediction of `loan_status` on the columns `loan_amnt, term, subgrade, emp_length, home_ownership, verification_status` and `purpose`, and we are only interested in rows with a `loan_status` of `Fully Paid` (paid back) or `Charged Off` (defaulted).

2. Due to the relatively large number of rows and few missing values, we have the luxury to drop all rows with missing values. To this end, we:

   - drop all records with null values;
   - drop all records with a loan status other than `Fully Paid` and `Charged Off`;
   - drop all records where the employment length is not available;
   - convert `sub_grade` into a number;
   - convert the strings in `emplen` (employment length) and `term` (duration of the loan) to numbers.

Below is a Python script that you can use to perform the necessary clean-up:

```
import pandas as pd
df=pd.read_csv("LoanStats3a.csv")
df=df[['loan_amnt','term','sub_grade','emp_length'
       ,'home_ownership','verification_status','purpose','loan_status']]
df=df.dropna()
df=df[df.loan_status.isin(['Fully Paid','Charged Off'])]
df=df[df.emp_length!='n/a']
df['term']=df.term.apply(lambda x: int(x.split()[0]))
```

[1] Use the section 'DOWNLOAD LOAN DATA' with 'Year 2007-2011' (CSV: 9,648kb).

```
grades=['G','F','E','D','C','B','A']
df['gradeencoding']=df['sub_grade'].apply(lambda x: grades.index(x[0])
    +(0.7-0.1*float(x[1])))
def empllengthprocess(x):
    x=x.split('year')[0]
    if('+') in x:
        return 12
    if ('<') in x:
        return 0
    else:
        return int(x)
df['emplen']=df.emp_length.apply(lambda x: empllengthprocess(x))
df=df[['loan_amnt','term','verification_status','gradeencoding','emplen','
    purpose','home_ownership',
        #'emp_title',
        'loan_status']]
df.to_csv('Loans_processed.csv',index=False)
```

2.2 Prediction

Use R to construct the classification tree:

1. Import the pre-processed data set in R. Shuffle the records and split them into a training set (20,000 records), a validation set (8,000 records) and a test set (all remaining records).

2. Using a classification tree (look at the `C50` library), try to predict with an accuracy greater than $\frac{\text{\# of repaid loans}}{\text{\# of repaid loans + \# of charged off loans}}$ if a loan will be repaid. Do you manage to achieve this performance on the validation set? What about the training set?

3. The majority of loans in the data set are being repaid. By default, a classification tree algorithm uses 'majority votes' in the leaf nodes and thus classify loans in leaf nodes with more than 50% non-defaults as 'safe'. This strategy optimizes the default metric: the number of correctly classified loans.

   From a business perspective, however, we are interested in identifying loans with a high probability of default, even if the associated data record falls in a leaf node with more than 50% of 'safe' loan samples. R's `C50` library contains a 'cost matrix' parameter that allows you to change the optimized metric and thus put more weight on one type of error over the other (e.g., false positives or false negatives). Experiment with different cost matrices to achieve a sensitivity (also known as *recall*) of approximately 25%, 40% and 50% in your validation set.[2] Also report the percentage of the loans $\frac{n_{11}}{n_{11}+n_{21}}$ you would recommend to the bank for re-evaluation that were indeed charged off (also known as *precision*).

[2] For the confusion matrix

	Charged off (pred.)	Paid (pred.)
Charged off (act.)	n_{11}	n_{12}
Paid (act.)	n_{21}	n_{22}

, the sensitivity is $\frac{n_{11}}{n_{11}+n_{12}}$.

4. Pick a cost parameter matrix that you assess as the most appropriate for identifying loan applications that deserve further examination.

5. Evaluate the performance of your cost parameter matrix on the test set.