# **Artificial Intelligence and Machine Learning Fundamentals**

**Activity 7**: Preparing Credit Data for Classification

This section will discuss how to prepare data for a classifier. We will be using

german.data from <https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/>, as an example and prepare the data for training and testing a classifier. Make sure all your labels are numeric, and the values are prepared for classification. Use 80% of the data points as training data.

1. Save german.data from <https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/>, and open it in a text editor like Sublime Text or Atom. Add the following first row to it:

CheckingAccountStatus DurationMonths CreditHistory CreditPurpose

CreditAmount SavingsAccount EmploymentSince DisposableIncomePercent

PersonalStatusSex OtherDebtors PresentResidenceMonths Property Age

OtherInstallmentPlans Housing NumberOfExistingCreditsInBank Job

LiabilityNumberOfPeople Phone ForeignWorker CreditScore

1. Import the data file using pandas and replace NA values with an outlier value:

import pandas

data\_frame = pandas.read\_csv('german.data', sep=' ')

data\_frame.replace('NA', -1000000, inplace=True)

1. Perform label encoding. We need to transform all labels in the data frame to integers. We could create all labels in a one dimensional array. However, this would be highly ineffective, because each label occurs in exactly one column. It makes a lot more sense to group our labels per column:

labels = {

'CheckingAccountStatus': ['A11', 'A12', 'A13', 'A14'],

'CreditHistory': ['A30', 'A31', 'A32', 'A33', 'A34'],

'CreditPurpose': ['A40', 'A41', 'A42', 'A43', 'A44', 'A45', 'A46',

'A47', 'A48', 'A49', 'A410'],

'SavingsAccount': ['A61', 'A62', 'A63', 'A64', 'A65'],

'EmploymentSince': ['A71', 'A72', 'A73', 'A74', 'A75'],

'PersonalStatusSex': ['A91', 'A92', 'A93', 'A94', 'A95'],

'OtherDebtors': ['A101', 'A102', 'A103'],

'Property': ['A121', 'A122', 'A123', 'A124'],

'OtherInstallmentPlans': ['A141', 'A142', 'A143'],

'Housing': ['A151', 'A152', 'A153'],

'Job': ['A171', 'A172', 'A173', 'A174'],

'Phone': ['A191', 'A192'],

'ForeignWorker': ['A201', 'A202']

}

1. Let's create a label encoder for each column and encode the values:

from sklearn import preprocessing

label\_encoders = {}

data\_frame\_encoded = pandas.DataFrame()

for column in data\_frame:

if column in labels:

label\_encoders[column] = preprocessing.LabelEncoder()

label\_encoders[column].fit(labels[column])

data\_frame\_encoded[column] = label\_encoders[

column].transform(data\_frame[column])

else:

data\_frame\_encoded[column] = data\_frame[column]

Let's verify that we did everything correctly:

data\_frame\_encoded.head()

CheckingAccountStatus DurationMonths CreditHistory CreditPurpose \

0 0 6 4 4

1 1 48 2 4

2 3 12 4 7

3 0 42 2 3

4 0 24 3 0

CreditAmount SavingsAccount EmploymentSince DisposableIncomePercent \

0 1169 4 4 4

1 5951 0 2 2

2 2096 0 3 2

3 7882 0 3 2

4 4870 0 2 3

PersonalStatusSex OtherDebtors ... Property Age \

0 2 0 ... 0 67

1 1 0 ... 0 22

2 2 0 ... 0 49

3 2 2 ... 1 45

4 2 0 ... 3 53

0 2 1 2 2

1 2 1 1 2

2 2 1 1 1

3 2 2 1 2

4 2 2 2 2

OtherInstallmentPlans Housing NumberOfExistingCreditsInBank Job \

0 2 1 2 2

1 2 1 1 2

2 2 1 1 1

3 2 2 1 2

4 2 2 2 2

LiabilityNumberOfPeople Phone ForeignWorker CreditScore

0 1 1 0 1

1 1 0 0 2

2 2 0 0 1

3 2 0 0 1

4 2 0 0 2

[5 rows x 21 columns]

label\_encoders

{'CheckingAccountStatus': LabelEncoder(),

'CreditHistory': LabelEncoder(),

'CreditPurpose': LabelEncoder(),

'EmploymentSince': LabelEncoder(),

'ForeignWorker': LabelEncoder(),

'Housing': LabelEncoder(),

'Job': LabelEncoder(),

'OtherDebtors': LabelEncoder(),

'OtherInstallmentPlans': LabelEncoder(),

'PersonalStatusSex': LabelEncoder(),

'Phone': LabelEncoder(),

'Property': LabelEncoder(),

'SavingsAccount': LabelEncoder()}

All the 21 columns are available, and the label encoders have been saved in an object too. Our data is now pre-processed. You don’t need to save these label encoders if you don't wish to decode the encoded values. We just saved them for the sake of completeness.

1. It is time to separate features from labels. We can apply the same method as the one we saw in the theory section:

import numpy as np

features = np.array(

data\_frame\_encoded.drop(['CreditScore'], 1)

)

label = np.array(data\_frame\_encoded['CreditScore'])

Our features are not yet scaled. This is a problem, because the credit amount distances can be significantly higher than the differences in age for instance.

We must perform scaling of the training and testing data together, therefore, the latest step when we can still perform scaling is before we split training data from testing data.

1. Let's use a Min-Max scaler from scikit's Preprocessing library:

scaled\_features = preprocessing.MinMaxScaler(

feature\_range=(0,1)).fit\_transform(features)

1. The final step is cross-validation. We will shuffle our data, and use 80% of all data for training, 20% for testing.

from sklearn import model\_selection

features\_train, features\_test, label\_train,

label\_test = model\_selection.train\_test\_split(

scaled\_features,

label,

test\_size = 0.2

)