Machine Learning Project
Prediction of attrition

by Hussam Saoud
https://github.com/saoudh

1 Introduction

The project has the aim of predicting the attrition of employees, based on the data set provided by Ahassan Abdelglil[1].

The data set contains a huge number of features, although not all of them are relevant and needed for the classification process.

2 Implementation

The task was implemented using Scala as programming language and spark as Machine Learning framework. For the UI *scalaFX* (based on *javaFX*) is used.

The main class is the Scala-object *MyGUI* and triggers the training, evaluation and prediction processes from the *ClassificationProcess* class.

2.1 Data preprocessing

Prior to the classification process the data has to be preprocessed. First the features are read from the provided *csv*-file and saved as a data set of *Employee* objects using a corresponding schema. Leading and trailing white space has to be deleted to avoid *NumberFormatException* while parsing integer values. As the data source file contains the header labels the reader has to consider that while reading the file.

Some features as well as the label contain categorical values which has to be indexed as one-hot encoding.

2.2 Training

For the classification process a *ModelPipeline* from the *Spark* framework is used. The data is split into a training and test set with ration 0.7/0.3. For classification a *Decision Tree* is used.

A *VectorAssembler* is used to constrain the features to a set of features which are really relevant for the classification process. To figure out which features are relevant, the decision tree has to be built first.

2.3 Prediction

In the prediction process the raw data to be processed goes through the predefined ModelPipeline and the label is predicted by a *BinaryClassificationEvaluator.* If the data is given as an *Employee* object then it has to be transformed to a *DataFrame* before processing it through the pipeline.

1 https://www.kaggle.com/alhassanabdelglil/classification

3 Installation

The application can be installed with the containing *GradleWrapper.* To build the project go to the project folder in the terminal and execute:

./gradlew build

To run the application execute:

./gradlew run

4 Evaluation

A *BinaryClassificationEvaluator* and a 5-fold *CrossValidator* is used for evaluation of the model. To get the model with the best max depth parameter, values from 2 to 7 are used for this parameter.

The model has an accuracy on unseen data from the test data set of about 85 % (figure 1).

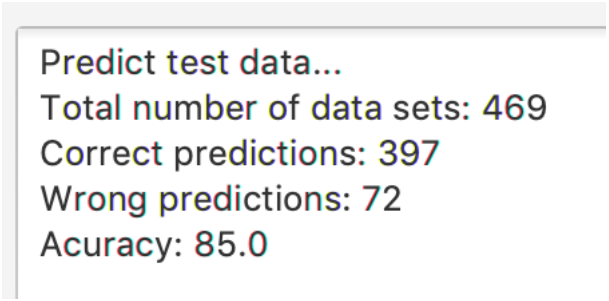

figure 1: Evaluation of the model with test data

The prediction of predefined default values set by triggering the button “set default values” has a probability of the predicted label being true of 79 % (figure 2).

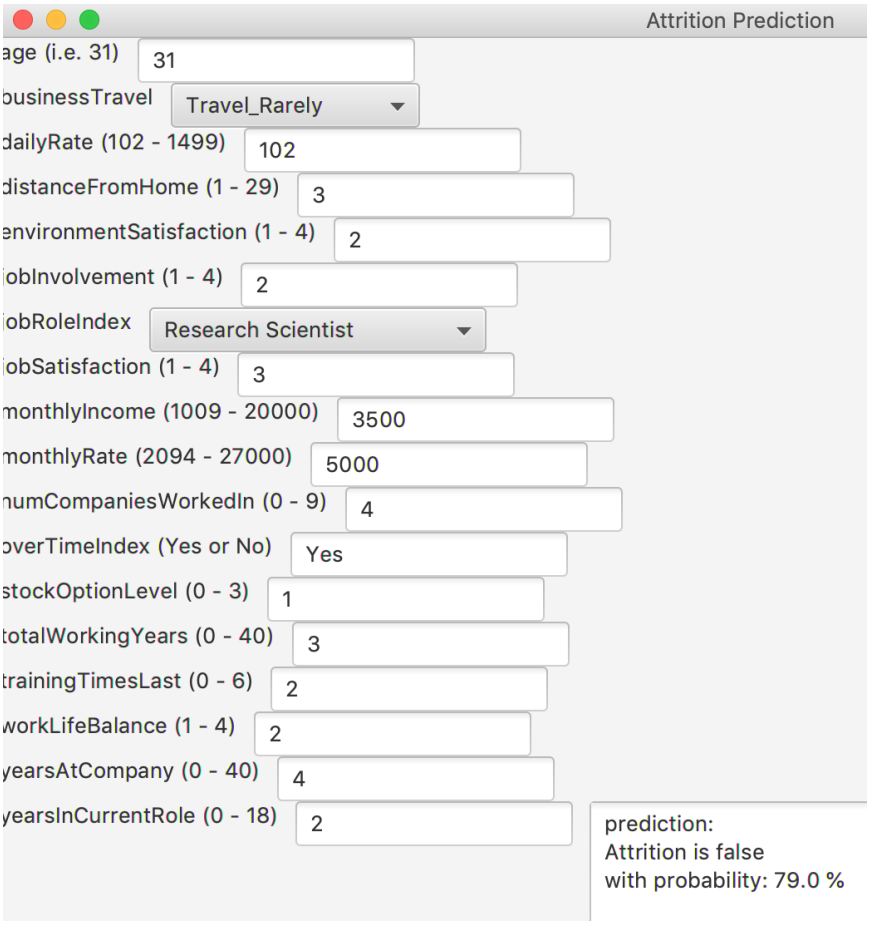

figure 2: prediction of default values