ID2223 – Scalable Machine Learning and Deep Learning

Lab 1. An End to End Classification

Course Coordinators:
Amir H. Payberah
Kamal Hakimzadeh
Jim Dowling

Group *ScalableGroup*

Mohamed Gabr
Óttar Guðmundsson

2018-11-28

For this short report, we'll cover the last part of lab 1, section 9 named *An End to End Classification*. We wanted to write a small report based on the classification task, since the examples in part 1 to 8 are pretty straightforward, simply filling in parts of the code where it is needed. This part sorted into a few categories.

1. Load the data.
This part was the easiest, as spark read the csv file in a single simple command without any problems.

2. Carry out some exploratory analyses (e.g., how various features and the target variable are distributed).
Now the first and most important part of machine learning is data preparation. Before we wanted to try out some of the models that we were introduced to, first we had to understand the data that we were working on. To start our analysis, we computed a small information about the dataset.

```
Number of rows are: 30000
The number of parameters including the index are: 25
Number of records with a credit higher than 200000 are: 9622
The percentage of females: 60.3%
The percentage of males: 39.6%
```

The first thing to notice is that the data set is relatively small, as 30.000 data points are not really a big data set of today's standard. It has 25 parameters (as explained in the ccdefault.txt file) so there are a lot of parameters that can be used for training.

There were some interesting statistics that we took away from our analysis. We noticed that the dataset had way more females and about one third had their limit balance over 200.000. This hinted that maybe, just maybe, females from the dataset had a higher limit balance than the males. To test our hypothesis, we got the average limit balance grouped by genders

```
Average Credit By Gender
+--------+------------------+
|SEX DISP|     AVG LIMIT BAL|
+--------+------------------+
|  Female|170086.46201413427|
|    Male| 163519.8250336474|
+--------+------------------+
```

This seemed correct, hinting that females were the dominant factor for higher incomes. What else was there to look at? We were interested to see if the marriage status did affect the income, so we grouped by the love life.

```
Average Credit By Marital Status
+-------------+------------------+
|MARRIAGE DISP|     AVG LIMIT BAL|
+-------------+------------------+
|      Married|182200.89318398127|
|       Single|156413.66073665748|
|      Unknown|132962.96296296295|
|        Other| 98080.49535603715|
+-------------+------------------+
```

Wait a sec, something is weird here right? The ccdefault.text explains the data should have three types of marriage statues. However, we found out that there were indeed four labels that described that attribute. This hinted that maybe the data set was not as clean as we had hoped for. Further analysis ahead, do the same for the education!

```
Average Credit By Education
+---------------+------------------+
| EDUCATION DISP|     AVG LIMIT BAL|
+---------------+------------------+
|         Others|220894.30894308942|
|       Unknown0|217142.85714285713|
|Graduate School|212956.06991025034|
|       Unknown5| 168164.2857142857|
|        Unknow6|148235.29411764705|
|     University| 147062.4376336422|
|    High School|126550.27049013626|
+---------------+------------------+
```

We were right. Just like before, it seemed like the labels for education was incorrectly classified. For this attribute, 3 extra labels were marked at the unknown, hinting that education might not be an important factor here. Note that the category “*others”* has the highest average income

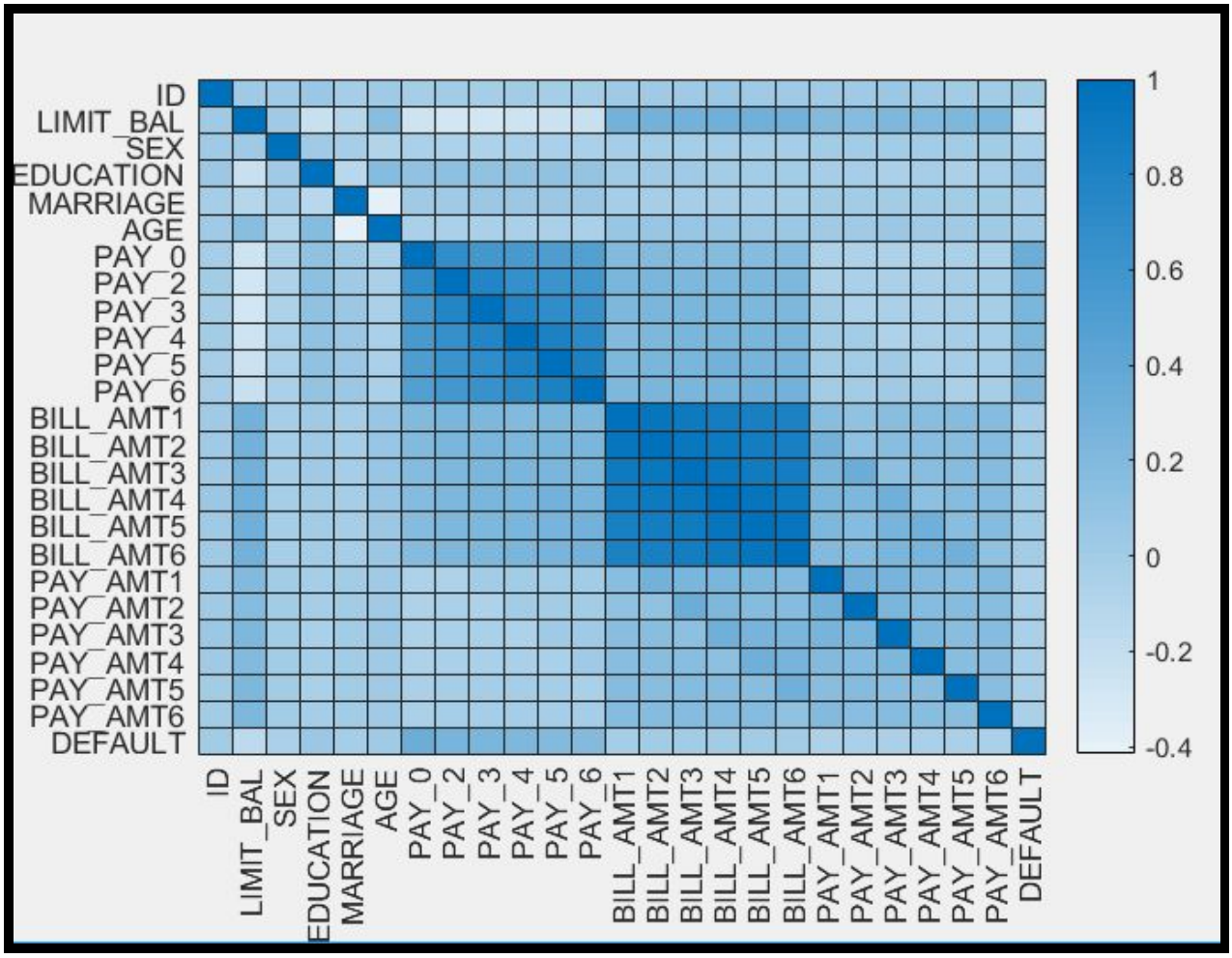
suggesting that maybe we are just wasting our time studying here… or there is a lot of black market going on that we are unaware of.

```
The average age is: 35.4855
The max age is: 79
The min age is: 21
```

Finally, some basic information about the age of the data set. Nothing out of the ordinary. But we were skeptical that the dataset was as clean as it posed to be, due to its different labels not mentioned in the description. So just before training, we needed to make sure that no value was missing.

```
Column ID has 0 missing values.
Column LIMIT_BAL has 0 missing values.
Column SEX has 0 missing values.
Column EDUCATION has 0 missing values.
Column MARRIAGE has 0 missing values.
Column AGE has 0 missing values.
Column PAY_0 has 0 missing values.
Column PAY_2 has 0 missing values.
Column PAY_3 has 0 missing values.
Column PAY_4 has 0 missing values.
Column PAY_5 has 0 missing values.
Column PAY_6 has 0 missing values.
Column BILL_AMT1 has 0 missing values.
Column BILL_AMT2 has 0 missing values.
Column BILL_AMT3 has 0 missing values.
Column BILL_AMT4 has 0 missing values.
Column BILL_AMT5 has 0 missing values.
Column BILL_AMT6 has 0 missing values.
Column PAY_AMT1 has 0 missing values.
Column PAY_AMT2 has 0 missing values.
Column PAY_AMT3 has 0 missing values.
Column PAY_AMT4 has 0 missing values.
Column PAY_AMT5 has 0 missing values.
Column PAY_AMT6 has 0 missing values.
Column DEFAULT has 0 missing values.
```

Surprise! It wasn't. As we were now sure that there were no values missing from the table, we could calculate the correlation between each attribute to determine if they were related to one another.



We could go on endlessly speculating this correlational heatmap, but the most important giveaway is that the PAY and BILL attributes are closely correlated. This makes us at ease as we know that the status of how people are classified are similar across periods and they have a connection to the BILL variable. Note that the correlational matrix was moved to Matlab to render the image. Now, onto the training.

3. Train a model to predict the target variable (risk of default).

The data is now ready and cleaned for training.
We first start by dividing the data into a training and test set. We split it 70% for the training and 30% for the test set randomly. Three different classification models are tested: logistic regression, binary tree and random forest.

3A. Employ three different models (logistic regression, decision tree, and random forest).

Logistic Regression: there are three different hyperparameters that we optimize using a cross validation which are: the threshold (positive classification if above and negative otherwise) , the maximum number of iterations and finally the regression parameter.
A worthy note is that the optimal regression parameter is 0, which means that there is no need of regularization since overfitting is not a problem in our case.

Decision Tree: We left the default values for the decision tree.

Random Forest: We tuned two different hyperparameters which are the number of trees and the maximum depth.

3B. Compare the models' performances (e.g., AUC).

In each of the models, the accuracy was the metric used to evaluate the error.
After the hyperparameter optimization for logistic regression, the test error is around 18%.
The error for decision trees is also around the same percentage. Finally, the optimized random forest also gets the same amount of misclassified test data (18%).
We weren't able to choose the best with such similarities in the performance using this metric.
We decided to calculate the Area Under ROC. The optimal value is 1:

For logistic regression we obtained 0.72 without the optimization
For decision trees we obtained 0.76 without optimization
For Random forests we obtained 0.76 without optimization

Again, we can see that the results are pretty similar which makes the preference of one model over the other harder. This may be due to a dataset that is easy to learn.

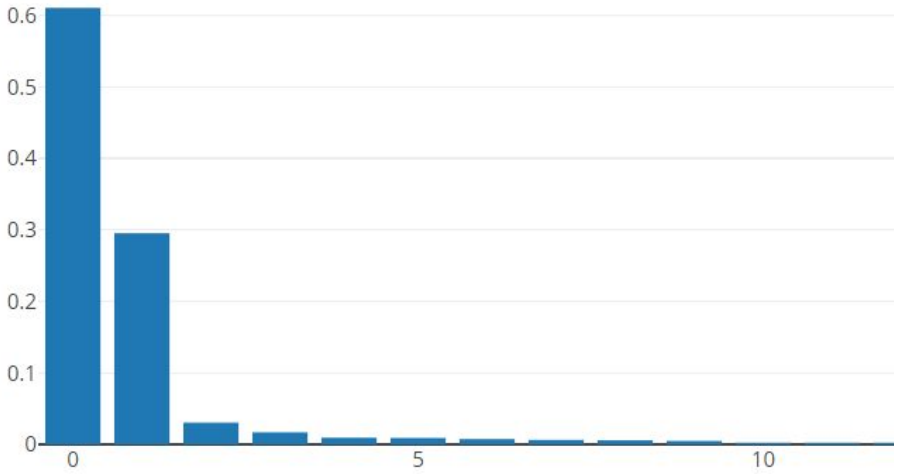
All the results are pretty accurate.
To take the decision, we evaluate the pros and cons of each technique and then try to make choice based on the advantages of one model compared to the other and its potential behavior on a similar dataset.

3C. Defend your choice of best model (e.g., what are the strength and weaknesses of each of these models?).

In all cases, the random forest is better than a decision tree, since a random forest is an aggregation of multiple decision trees, with an ability to limit the overfitting, reducing the variance (thanks to the aggregation) since each decision tree in the array of decision trees built in the random forest is based on a sample of the dataset.
The choice is now between logistic regression and the random forest. The advantage of a random forest is the same advantage of a decision tree, building a decision tree is quite fast. However, the global optimum can be easily missed; at each node the optimal question/feature must be chosen, optimality is based on a metric that is local (entropy before and after the node

split for instance). It means that optimality is only determined at each node, something very similar to greedy algorithms techniques. Decision trees/Random Forests can easily have an overfitting problem.
In our case, Random forest, a boosted algorithm, should outperform logistic regression, since we have a mixture of categorical and numerical features handled well by RF. One pro for logistic regression is to easily show the results, which could be helpful in finance or medicine for example, it is very fast with a model with a lot of sparse features. Also, logistic regression may outperform the random forest when there is a little amount of data and on a problem that is hard to predict (no enough correlation). This is not our case, we have a medium-sized task that is not hard to predict with a lot of data, thus we prefer the random forest.

4. What more would you do with this data? Anything to help you devise a better solution?
For the last part, we agreed that using around 23 parameters to train using Random Forest was a little bit too much so we needed to reduce the data. To do this, we used the dimension reduction technique called Principal Component Analysis (PCA). Spark has an inbuilt method to perform PCA on a data frame.



We noticed that by applying PCA on the data frame, the first 6 components span up to more than 97% of the whole vector space, meaning that we could possibly train the network using way fewer features. So by applying PCA again, but only setting the k to 6, we got the 6 biggest eigenvectors and transformed the dataset. This transformed data set was then trained using Random forest classifier which provided us with these results.

```
Area under ROC = 0.458438109799030 4
Test Error = 0.21693004751906286
Test Error = 0.21693004751906286
```

It seemed like our test error was still pretty similar and even worse off. The ROC area was also not as good as we had hoped for, but it was worth giving it a try. There are other methods of using dimension reduction methods but we'll stop at here.

Finally, we wanted to retry some training by seeing if we'd get different results based on the ratio of what a person has vs the bill amount.

```
val ccdefault1 = ccdefault.withColumn("PAYBILL1", ccdefault("PAY_AMT1") / ccdefault("BILL_AMT1"))
val ccdefault2 = ccdefault1.withColumn("PAYBILL2", ccdefault1("PAY_AMT2") / ccdefault1("BILL_AMT2"))
val ccdefault3 = ccdefault2.withColumn("PAYBILL3", ccdefault2("PAY_AMT3") / ccdefault2("BILL_AMT3"))
val ccdefault4 = ccdefault3.withColumn("PAYBILL4", ccdefault3("PAY_AMT4") / ccdefault3("BILL_AMT4"))
val ccdefault5 = ccdefault4.withColumn("PAYBILL5", ccdefault4("PAY_AMT5") / ccdefault4("BILL_AMT5"))

//
val ccdefaultFinal = ccdefault5.na.fill(0, Seq("PAYBILL1", "PAYBILL2", "PAYBILL3", "PAYBILL4", "PAYBILL5"))
ccdefaultFinal.select("PAYBILL1", "PAYBILL2", "PAYBILL3", "PAYBILL4", "PAYBILL5").show(5)
```

```
+-------------------+-------------------+-------------------+-------------------+-------------------+
|           PAYBILL1|           PAYBILL2|           PAYBILL3|           PAYBILL4|           PAYBILL5|
+-------------------+-------------------+-------------------+-------------------+-------------------+
|                0.0|0.22211476466795615|                0.0|                0.0|                0.0|
|                0.0| 0.5797101449275363| 0.37285607755406414| 0.3056234718826406|                0.0|
|0.05191696022435788|0.10693662222855921|  0.0737517516041006|0.06977880120019538| 0.0668985817500669|
|0.04256224728665674|0.04185930794269484|0.024345215150838897|0.03885003885003885|0.03691425808902241|
|0.23209933851688522|  6.469312169312169| 0.27905678805636946| 0.4297994269340974|0.03598662906090045|
+-------------------+-------------------+-------------------+-------------------+-------------------+
only showing top 5 rows
```

After testing out multiple training parameters, optimization and other inputs we were not reaching higer accuracy than around 80%. We concluded that more data samples were needed to reach more accuracy.