Cajamar Challenge: Financial Products Predictive Modelling applying Deep Learning

Florent Micand
Foundations of Data Science's Master's Thesis
University of Barcelona
fmicand@gmail.com

Abstract

Responding to a bank's challenge, a predictive model is presented to address the task of predicting the next financial product an existing customer is willing to buy, given his purchases' history and sociodemographic information. After stating the problem and identifying the bank's needs, we explore in detail the dataset to detect trends and patterns which motivate the creation of new features, and the Neural Network based's model proposed. The results will show the importance of the Recurrent Neural Network part of the proposed architecture to overcome other models. During the solution's development, the business, ethical and technical specificities of the task have been constantly considered.

1 Introduction and background

1.1 Problem Statement

In February 2017, the Spanish bank *Cajamar Caja Rural* launched a challenge which main objective was to predict the next financial product customers are willing to purchase [1]. The challenge's participants will have access to historical purchases made by the bank's customers and sociodemographic information related to them (age range, seniority range, incomes range, gender and professional activity). Based on this data, the goal of the challenge is to build a predictive model that will output the most recommended product to propose to their current customers, in order to optimize the bank Cross Selling and Up Selling strategies.

1.2 Cajamar Caja Rural

Cajamar Caja Rural [2] is the leading rural savings bank and credit union in Spain. It was established as a result of the merger between local rural savings banks from several regions in Spain. It has 1.4 million members, 4 million customers, has assets amounted to 39 billion euros, managed business volume of 69 billion euros, solvency ratio of 12.93%, 1120 branches and 6020 employees.

Cajamar Caja Rural attends the needs of any sector of the economy and population, although its activities are oriented to the financing, savings and investment needs of the economy of families, professionals and the self-employed, and small and medium businesses. As a social economy company, it dedicates more attention to local productive sectors and especially to the agricultural and food production sector.

1.3 Background

Financial products recommendations are a key element in the banking industry. A main activity of banks consists on selling financial products to their existing clients, and on proposing engaging products to new prospect clients. Some decades ago, rural savings banks offered few and simple

products, targeted to well-defined customer profiles, and through a single channel (in person at the banks' branches). In the last years, the portfolio of products offered by a bank has increased significantly, the number of customers profiles is higher and the profiles more complex, and products can be sold through several channels (online, by phone, in person). The challenge of selling the right product to the right person has been redefined, and data modelling can add value in this task.

Apart from having the banking departments already implementing financial products recommendation systems, we can find a similar competition like Cajamar Challenge in the well-known Kaggle platform. The Santander Product Recommendation Kaggle competition [3] was launched mid-2016, and participants had to predict which products Santander's existing customers will use in the next month based on their past behaviour. The winner used an ensembling model of twelve neural networks and eight gradient boosting models, when most of the participants were using gradient boosting models only. Kaggle competitions are very focus on the final score, which explains this complex architecture to get few decimal score's improvements. In our case, we'll present a solution centred in obtaining a reasonably good score, as well as a fair and explicable model.

2 Definition of objectives and motivation

We want to find the next financial product a customer should be recommended with, so it's meaningful and highly probable for him to purchased it, based on his purchases' history and profile.

We first perform an exploratory analysis of the data to check if there are some trends, relations between features, that motivate building a modelling system for this dataset. Secondly, we want to explore the data processing that is needed to reflect correctly the data insights. We'll then propose a model based on Neural Networks to learn the correlations existing in the data, and finally present the results.

2.1 Exploratory Analysis

The dataset consists of 3.350.601 transaction records from customers based on Spain during their relationship with the bank entity, in a period between 1954 and 2017. There are 8 features as we can see in Table 1.

Table 1: Dataset's features description

Field Name	Description	# Classes
ID_Customer	Customer identifier	676370
Cod_Prod	Product identifier	94
Cod_Fecha	Purchase date (MM-YYYY)	635
Socio_Demo_01	Age range	5
Socio_Demo_02	Seniority range (in years)	5
Socio_Demo_03	Annual incomes range	5
Socio_Demo_04	Gender	2
Socio_Demo_05	Activity	4

Typically, a bank customer buys several financial products, so we'll have several entries per user. We run some consistency checks on the data to make sure there are no null values, nor duplicated entries, nor contradictions in the sociodemographic features (across the user's records).

The customers have bought from 1 to 29 products, and the average is 5 products per user. Although there are 94 different products, the first four most purchased products (product's codes 601, 301, 201 and 2302) represent 50% of the transactions. The socio demographic variables are quite unbalanced too. Most of the transactions are from users older than 30 (age ranges from 3 to 5), seniority higher than 10 years (seniority ranges 4 and 5), low-average annual incomes between 6000 to 12000 euros (incomes range 2). Among all the customers, 56% are men and 79% belong to the "Personal" activity (activity range 1). Other possible activities are "Farmer", "Self-employed" and "Commerce".

We can detect relationships between sociodemographic features and purchased products. For instance, the top 5 most purchased products change for the youngest customers compared to the oldest ones. A simple example is the mortgage product, which is usually purchased by mid-aged people.

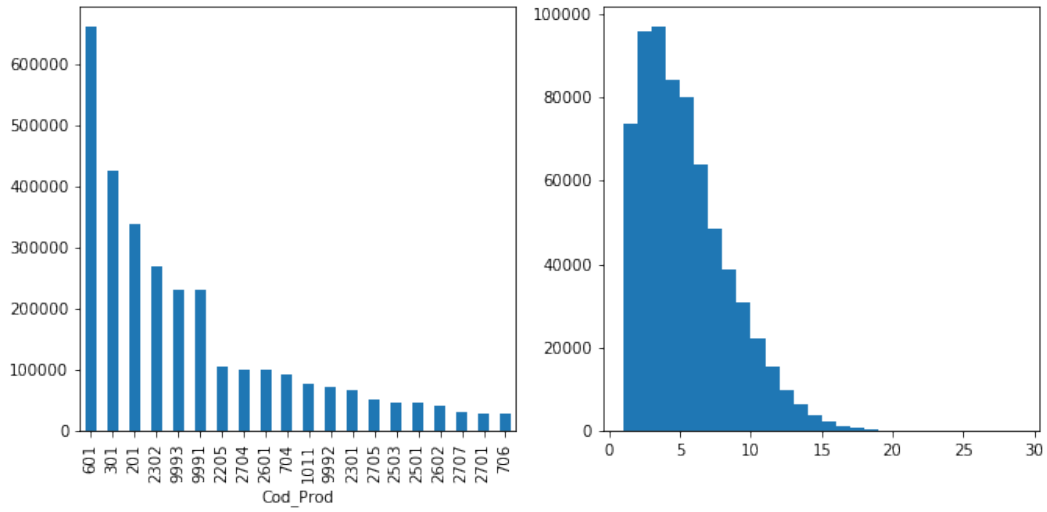

Figure 1: Left: Number of customers for the Top 20 products. Right: Histogram of products per customer.

The product 601 is the most purchased product for users who have purchased 1 product only. As expected, the purchased products share change when we consider users who have bought more than 5, or more than 10 products. We can also investigate if some products are typically bought together analysing pairs of products per user. We find out this behaviour is frequent and the most purchased pairs are 201-601, 601-2302 and 301-601.

We can use the month and year purchase date to check if buying is affected by temporality. A monthly distribution of all transactions shows us how March is the month where a higher amount of transactions is done. As expected, the seasonality effect is more pronounced inspecting each product individually. A yearly distribution of all transactions displays how most of them are from the last decade, with also more specificities inspecting the figures on a product base. The economic crisis in Spain (2008 – 2014), the launch and discontinuation of financial products and the job market seasonality are some of the temporal effects reflected on the data.

As we have records from 1954, we can find financial products that have been discontinued. 18 products over the 94 total products haven't been sold during the last year. We'll take this situation into consideration to make sure we don't recommend them.

We've tried to state the insights of the exploratory analysis in few words, but a better understanding can be got from the figures available in the code repository [4].

2.2 Data Processing

The exploratory analysis confirms that some relationships, dependencies and trends exist on this dataset. To map them correctly and feed our model, we consider in this section the needed data pre-processing and features engineering.

We create a new dataset grouping the transactions per user, where the last product purchased by each customer is not included because this subset of data will be the labels, i.e. the values to learn and predict. The product, month and year of each transaction are aggregated in lists, per user. As we consider that the purchase history is key to a good prediction, we first discarded transactions where the customers had purchased only once (whose only product would have been the label). It represents dropping 2.2% of the transactions. We then discard the gender feature for ethical reasons we'll discuss later.

The sociodemographic features and products dependencies, the products sequentiality and the purchases temporality can be inferred from this new dataset, but we are missing the cooccurrence of products detected in the previous section. That's why for each sequence of products we create a set of pair combinations of them (where intermediate tokens can be skipped) called skipgrams, based on the

knowledge acquired in the NLP course [5] . For instance, the user who has purchased the products (2501, 2503, 601) will have the following set of skipgrams [(2501, 2503), (2501, 601), (2501, 1011), (2503, 601)]. We obtain 6034 different skipgrams.

We have seen that 18 products haven't been sold in the last year. As we don't want to recommend them, we drop 807 customers which label was one of those old products.

2.3 Model

We have considered and tested several machine learning models to solve this task, including Random Forest and Gradient Boosting models. The Deep Learning course [6] taken along this year motivated us to try a Neural Network architecture we'll explain below, implemented in Keras Functional model, and which has shown better results than other models.

The Neural Network architecture proposed will be fed with two different inputs: a sequential representation of products, purchase's months and purchase's years on one hand, and the sociodemographic information and skipgrams on the other hand. The sequential data will be processed by a Recurrent Neural Network, which output will be concatenated to the second input thanks to a concatenation layer. Three dense layers will be plugged on top of it to generate the final output.

2.3.1 Input 1

Recurrent Neural Networks are neural networks specialized for processing sequential data, and can deal with sequences of variable length [7]. In order to learn from the sequential information available in the sequences of products purchased by each user, and from the corresponding temporality, we feed an LSTM layer with batches of 3D tensors of shape $(batch\_size, sequence\_length, 3)$, representing by this way sequences of products, purchase's months and purchase's years. We pad the sequences to the longest sequence (28).

2.3.2 Input 2

In a second step, the sociodemographic data is transformed to a binary class matrix using Keras `to_categorical` method, and the skipgrams lists are transformed to a sparse binary class matrix with the `MultiLabelBinarizer` method from sklearn. Both matrices are concatenated together in a sparse format for efficiency (there are more than 6000 skipgrams classes).

2.3.3 Concatenation and Dense layers

The output of the LSTM is concatenated with Input 2 through a concatenation layer. Two *relu* activated dense layers, the second one with half units than the first one, are then stacked. A final *softmax* activated dense layer is added, with a number of units equal to the number of different classes in the labels set (76 and not 94, as we've removed previously those users whose last purchased product wasn't a product sold in the last year).

Regarding the labels set, we encode them and also use the `to_categorical` method to create a binary class matrix of labels, the format needed for Keras to handle it. The functional model is then compiled with an *adam* optimizer and a categorical crossentropy loss. The figure 2 shows the network architecture schema.

3 3. Discussion and results

3.1 Training

We've split the inputs in the following way: 50% for training, 25% for validation, and 25% for testing. As Keras doesn't handle directly sparse matrices, and input 2 is in sparse format, we've created a batch iterator that will feed the network by batches took randomly from the labels, Input 1, and Input 2 converted to dense. The iterator is used to feed the network for the training, validation and test sets.

During training, we make use of two Keras callbacks functions: `EarlyStopping` in order to stop the training when the validation accuracy doesn't improve after 1 epoch, and a personalized callback to

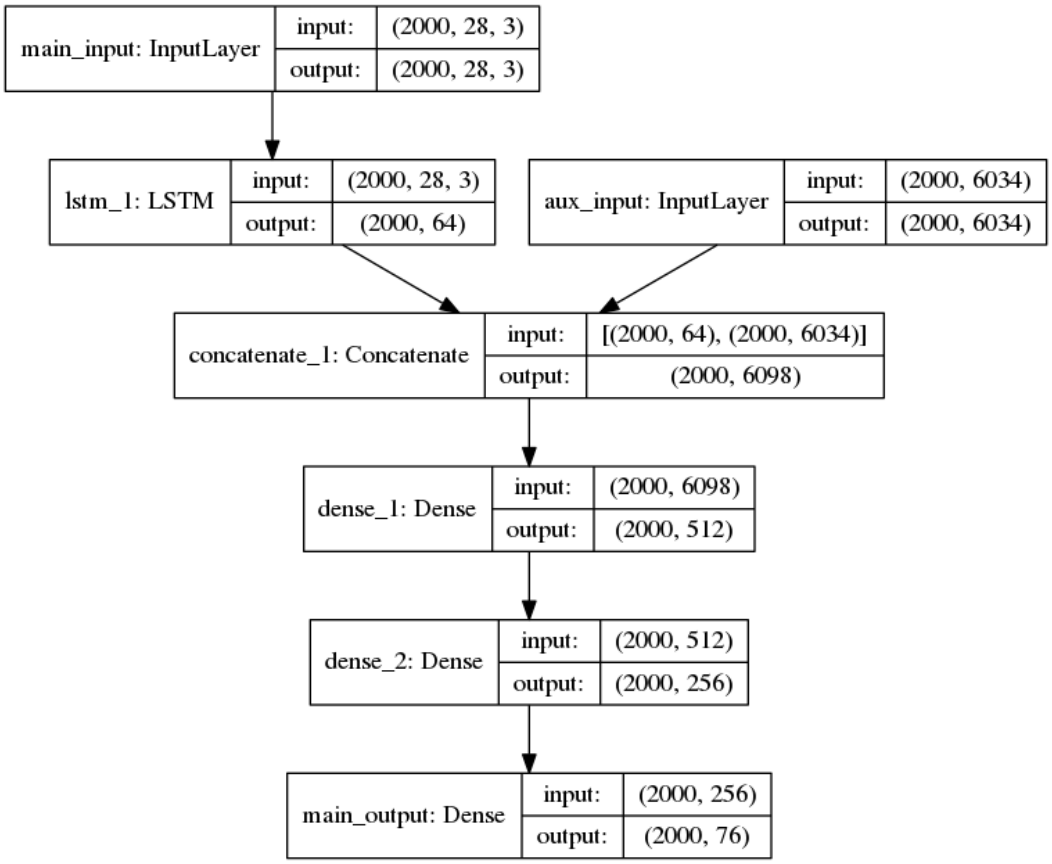

Figure 2: Neural Network's Architecture schema

get the training loss and the training accuracy at each batch step. The validation loss and validation accuracy is obtained at the end of an epoch only, so we have les log points for validation metrics.

3.2 Evaluation

For simplicity, we've decided to use the test accuracy as evaluation metric.

Other metrics could be though as future work, like calculating the accuracy over the next 3 products the user will purchase, as some challenge's participants proposed [1]. The inconvenient is that the average basket of products is 5 as we saw in the exploratory analysis section, so a significant part of the dataset would not be taken into account. Moreover, the catalogue of products is small (76 products if we remove products not purchased last year), and banking products are usually quite specific, so the pertinency of recommending several products is not clear.

Another interesting evaluation metric may be the share of users for which the predicted probability is higher than a certain threshold. By this way, the bank can recommend products to a subset of users only, with less cost (depending on the promotion method) and an higher probability of success.

3.3 RNN and Dense Units

In order to define which is the more suitable number of units to employ in the LSTM layer and the two dense layers, we've run the model with several combinations of units. You can see the results in table 2.

The best test accuracy is obtained using a 64 unit's LSTM, and 512 and 256 units for the two dense layers respectively. We expected that the best score would be given by the configuration with the highest number of units.

Table 2: Units in LSTM and Dense Layers

LSTM Units	Dense_1 Units	Dense_2 Units	Accuracy
64	256	128	0.5374
128	256	128	0.5378
64	512	256	0.5392
128	512	256	0.5386

A higher or lower contribution of the LSTM part in the final model depends on the share of the LSTM output in the concatenation layer. It appears that lowering the role of the LSTM (from 128 to 64 in term of output dimension) is positive for the final score, which is in accordance with the results we'll discuss in the following section.

3.4 Inputs and Network's parts comparison

Once we've found out the best configuration to use, and in order to understand better the proposed network architecture, we have separately ran Input 1 and his respective RNN followed by the dense layers on one side, Input 2 followed by the dense layers on another side, and finally the complete model. We want to know which input and respective part of the model brings the most relevant information to the prediction. The resulting accuracy and loss evolution can be seen below in figures 3 and 4 respectively.

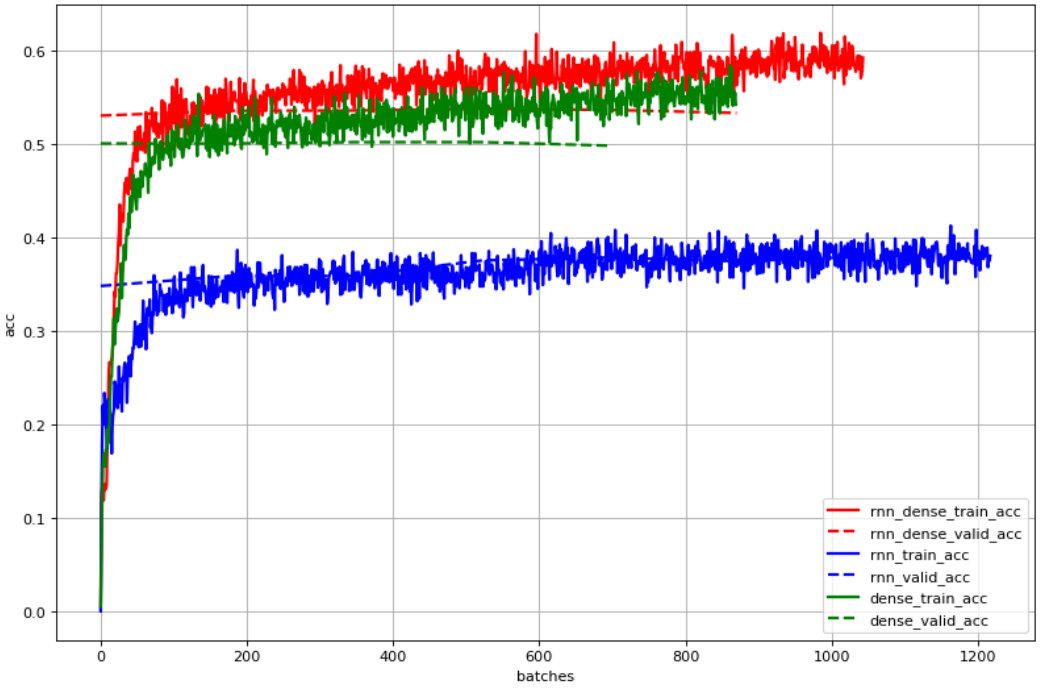

Figure 3: Training and validation accuracy across batches for different inputs and network parts

The dashed curves are the validation metrics while the continuous lines represent the training metrics. As explained before, the validation metrics are less granular because they are computed only at the end of each epoch, while the training metrics are obtained at each batch step. The `EarlyStopping` callback terminates the training when the validation accuracy doesn't improve after 1 epoch and the validation metrics isn't retrieved for the last epoch, that's why the represented validation metrics end before the training metrics in the figures. We can see how important is to monitor the validation accuracy as at some point the network is overfitting and learning the peculiarities of the training data, making the training accuracy improve and the training loss decrease constantly. The RNN model

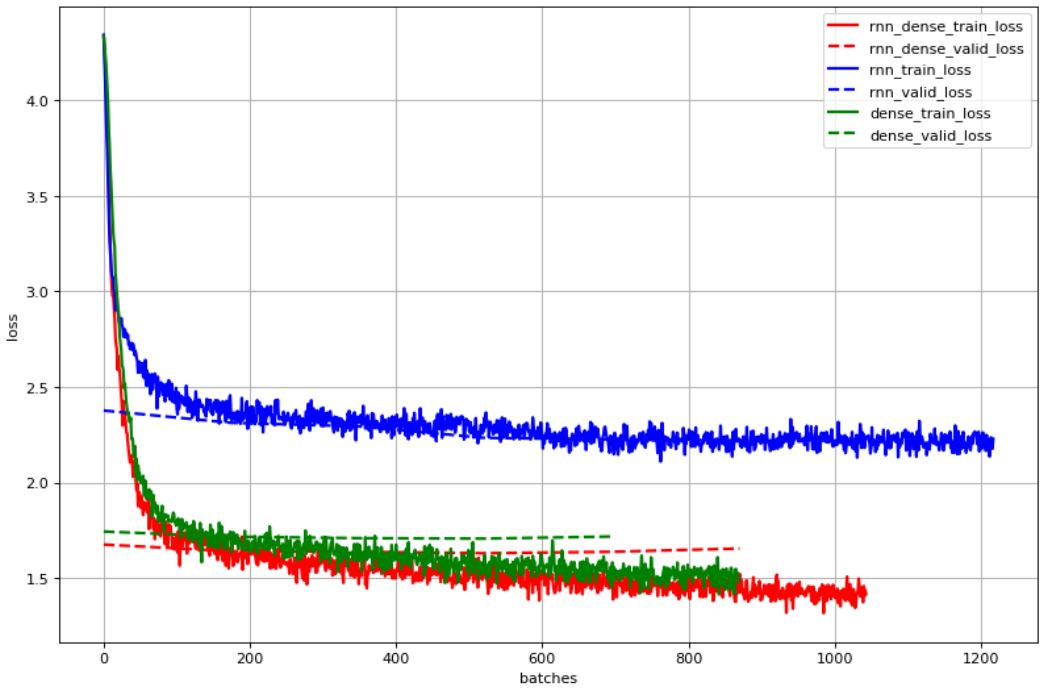

Figure 4: Training and validation losses across batches for different network parts

with Input 1 overfits later than the other part of the model, while the complete model overfits at an intermediate position between both parts.

Table 3: Inputs and Network's parts contributions

Network Part	Input	Accuracy
RNN + Dense	Input 1	0.3801
Dense	Input 2	0.5009
RNN + Dense	Inputs 1 and 2	0.5392

In Table 3 we can see how the sociodemographic information and the skipgrams allow us to reach a 50% accuracy through some simple dense layers. This score is similar to the one's presented by the challenge's finalists using a Xgboost model who have made use of the gender feature. The contribution of the RNN, which infers the sequential behaviour of products purchases along with the dates provides around 3.9 percentage points to the final accuracy score, overcoming the other models presented during the challenge (see Table 4).

Table 4: Challenge's solutions comparison

Challenge's participants	Model	Gender usage	Accuracy
First classified solution	Decision Trees	No	0.46
Second classified solution	Xgboost	Yes	0.50
Present solution	Neural Network	No	0.54

3.5 Other considerations

The age, seniority and incomes ranges have been considered as categorical data, but in fact they can be thought as continuous features. We've tried to use them as continuous variables but the model's accuracy was worse so we rejected the idea. We also tried to stack a dense layer on top of input 2, before concatenating the RNN output, but the results weren't improving neither.

This task was ran in a i7-3537U CPU, Nvidia Geforce 740M GPU, 8GB RAM Windows laptop in a Python 3.6 environment using, among other libraries, `keras 2.0.4` and `tensorflow 1.2.0`. A common execution to pre-process the data, train and evaluate the main model lasts 20 minutes approximately. The code created for this task is available at reference [4].

3.6 Gender Discussion

The Spanish Equality Law doesn't allow banking companies to take into consideration the gender as a variable to calculate primes or costs. We are building a recommender product with no monetary impact, as the recommended product will just be promoted. We can legally consider the gender feature in this model, but there would be some ethical concerns. The data is biased by the gender: we can see in this historical dataset how some products are more purchased by women, and other by men. If we take into account the gender in our model, we'll propagate this bias to our recommendations and making this partiality lasts. Although we ignore the gender, this bias can still be slightly implicit in some relationships in the dataset, which will be inferred by the model.

Although the gender feature would improve the model's score, we prefer to ignore it and let the bank improve, by this way, its corporate social responsibility.

4 Conclusions

We've seen it's possible to reach a decent result applying a deep learning model to a financial products prediction task, given a quite limited dataset and the business constraints. Thanks to a straightforward feature engineering consisting,mainly, in creating product's skipgrams, and to a RNN part in the Neural Network architecture to model sequential inputs, we obtain a 53.9% accuracy score which overcomes other presented models.

As future work, it can be interesting to know more about how the bank plan to promote the predicted products for their customers, and based on that adapt the model's evaluation as seen before. Motivated by the new European regulation, a new line of work would be to implement a solution focus on explicability in order to be able to explain why a customer is recommended a specific product. This problem can be addressed in neural network models on an individual base.

Acknowledgments

Erwan Guillotel and Jaume Puigbo Sanvisens, as part of our initial team to participate at this challenge.

Prof. Jordi Vitrià, as Master's Thesis tutor.

References

[1] Cajamar Datalab. *UniversityHack 2017 Datathon. Microsoft Predictive Modelling*. 2017. `http://www.cajamardatalab.com/datathon-cajamar-universityhack-2017/microsoft-predictive-modelling`.

[2] Cajamar Caja Rural. *About the Entity*. 2017. `https://www.cajamar.es/en/comun/informacion-corporativa/sobre-cajamar`.

[3] Kaggle Inc. *Santander Product Recommendation*. 2016. `https://www.kaggle.com/c/santander-product-recommendation`.

[4] Florent Micand. *Cajamar Challenge Repository Code*. 2017. `https://github.com/fmicand/cajamar_challenge`.

[5] M. Antònia Martí and Venelin Govatchev. *Natural Language Processing Course, Foundations of Data Science's Master*. University of Barcelona, 2017. `http://clic.ub.edu/`.

[6] Jordi Vitrià. *Deep Learning Course, Foundations of Data Science's Master*. University of Barcelona, 2017. `https://github.com/DeepLearningUB/DeepLearningMaster`.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.