UNIVERSITY OF CAEN NORMANDIE

DEPARTMENT OF PHYSICS

PROJECT 2

- report for Data analysis and machine learning course -

Professor:

Morten Hjorth-Jensen

Advisers:
Kristine B. Heine, Bendik Samseth

Students:

Igor Daskalovski, Aidana Shagalakova, Miloš Tomić, Kristina Vučković

February 2019

CONTENTS

1 ABSTRACT

This work analyzes the crisis of Taiwan bank in 2005. The aim of this project was to find the optimal method in predicting credit card defaults and compare results with Yeh and Leh as a reference. In this work we applied different techniques, namely, Logistic regression and Neural networks, including accuracy estimation, normalization and data preparation.

2 INTRODUCTION

More than a decade ago the Taiwan bank faced with a crisis caused by credit card over-usage; the result was an inability to cover credit card loans. Therefore, in order to avoid such situations, crisis management started to use a risk prediction strategy based on several criteria such as repayment records, age and current occupation of the client. In this work we tried to reproduce the result obtained by Yeh and Lien (2007) [1], related on Taiwan bank dataset and tried to find an alternative way of more effective prediction of credit card debt repayment.

Therefore, the purposes of this project are:

1. Structuring given credit card dataset (for detailed information see Project 1)
2. Apply logistic regression
3. Integrate the backpropagation neural network
4. Introduce a random forest as alternative way of data analysis
5. Summarize the results from all methods and make a comparison, where it is possible, with already published work.

This report was organized in the way to help the reader delve into the work done by our team. We suggest to read the brief theoretical aspects in Chapter I before moving to methods of data analysis and project's code lines given in Chapter II. In chapter III the reader will find the obtained result analysis as well as comparison with above mentioned work and future perspectives of the created code for data analysis as whole.

3 THEORY

As it was mentioned above, in this work we used logistic regression, neural network and random forest. The chapter does not include linear regression as well as structuring data, and theory of errors because a detailed explanation on these matters is given in Project 1 (https://github.com/Daskalovski13/Project1).

3.1 LOGISTIC REGRESSION

Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Dichotomous means there are only two possible classes. It is a special case of linear regression where the target variable is categorical in nature. It uses a log of odds as the dependent variable. Logistic Regression predicts the probability of occurrence of a binary event utilizing a logit function. [2]

Linear Regression Equation:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots. + \beta_n X_n \quad (1)$$

where, y is dependent variable and X_1, X_2 ... and X_n are explanatory variables.

Sigmoid Function:

$$p = \frac{1}{1+e^{-y}} \quad (2)$$

Apply Sigmoid function on linear regression

$$p = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots.+\beta_n X_n)}} \quad (3)$$

Properties of Logistic Regression:

- The dependent variable in logistic regression follows Bernoulli Distribution.
- Estimation is done through maximum likelihood.
- No R Square, Model fitness is calculated through Concordance, KS-Statistics.

Linear Regression Vs. Logistic Regression

Linear regression gives you a continuous output, but logistic regression provides a constant output. An example of the continuous output is house price and stock price. Examples of the discrete output is predicting whether a patient has cancer or not, predicting whether the customer will churn. Linear regression is estimated using Ordinary Least Squares (OLS) while logistic regression is estimated using Maximum Likelihood Estimation (MLE) approach.

Maximum Likelihood Estimation Vs. Least Square Method

The MLE is a "likelihood" maximization method, while OLS is a distance-minimizing approximation method. Maximizing the likelihood function determines the parameters that are most likely to produce the observed data. From a statistical point of view, MLE sets the mean and variance as parameters in determining the specific parametric values for a given model. This set of parameters can be used for predicting the data needed in a normal distribution.

Ordinary Least squares estimates are computed by fitting a regression line on given data points that has the minimum sum of the squared deviations (least square error). Both are used to estimate the parameters of a linear regression model. MLE assumes a joint probability mass function, while OLS doesn't require any stochastic assumptions for minimizing distance (see fig.1).

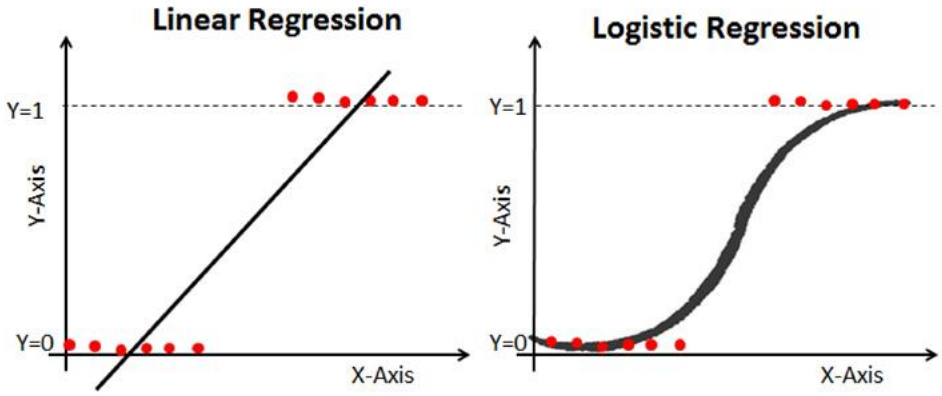

Figure 1 Linear Regression Vs. Logistic Regression

Sigmoid Function

The sigmoid function, also called logistic function gives an 'S' shaped curve that can take any real-valued number and map it into a value between 0 and 1. If the curve goes to positive infinity, y predicted will become 1, and if the curve goes to negative infinity, y predicted will become 0. If the output of the sigmoid function is more than 0.5, we can classify the outcome as 1 or YES, and if it is less than 0.5, we can classify it as 0 or NO.

Types of Logistic Regression:

- Binary Logistic Regression: The target variable has only two possible outcomes such as Spam or Not Spam, Cancer or No Cancer.
- Multinomial Logistic Regression: The target variable has three or more nominal categories such as predicting the type of Wine.
- Ordinal Logistic Regression: the target variable has three or more ordinal categories such as restaurant or product rating from 1 to 5.

3.2 NEURAL NETWORKS

The artificial neural network was designed as a computational model based on the brain to solve certain kinds of problems. [3,4]

It's probably pretty obvious that there are problems that are incredibly simple for a computer to solve, but difficult for human. Take the square root of 964,324, for example. A quick line of code produces the value 982, a number processing computed in less than a millisecond. There are, on the other hand,

problems that are incredibly simple for individuals to solve, but not so easy for a computer. Show any toddler a picture of a kitten or puppy and they'll be able to tell you very quickly which one is which. But what if a machine needs to perform one of these tasks? Scientists have already spent entire careers researching and implementing complex solutions.

The most common application of neural networks in computing today is to perform one of these "easy-for-a-human, difficult-for-a-machine" tasks, often referred to as pattern recognition. Applications range from optical character recognition (turning printed or handwritten scans into digital text) to facial recognition.

A neural network is a "connectionist" computational system. The program starts at the first line of code, executes it, and goes on to the next, following instructions in a linear fashion. A true neural network does not follow a linear path. Rather, information is processed collectively, in parallel

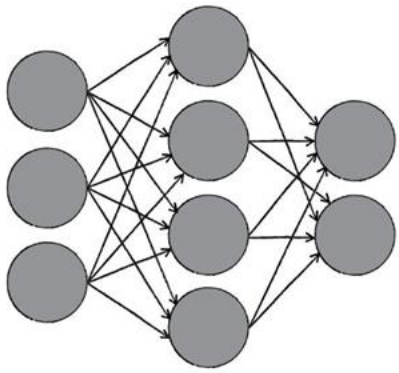

Figure 2 Pathway for the flow of information

throughout a network of nodes (the nodes, in this case, being neurons).

One of the key elements of a neural network is its ability to learn. A neural network is not just a complex system, but a complex **adaptive** system, meaning it can change its internal structure based on the information flowing through it. Typically, this is achieved through the adjusting of weights. In the diagram (fig.2), each line represents a connection between two neurons and indicates the pathway for the flow of information. Each connection has a **weight**, a number that controls the signal between the two neurons. If the network generates a "good" output, there is no need to adjust the weights. However, if the network generates a "poor" output—an error, so to speak—then the system adapts, altering the weights in order to improve subsequent results. [4]

The basic unit of a neural net is a stylized "neuron" i that takes a vector of inputs $x = (x1; x2; \ldots; xd)$ and produces a scalar output $a_i(x)$. A neural network consists of many such neurons stacked into layers, with the output of one layer serving as the input for the next (see Figure 2). The first layer in the neural net is called the input layer, the middle layer(s) is/are often called "hidden layer(s)", and the final layer is called the output layer.

The exact function a_i varies depending on the type of non-linearity used in the neural network. However, in essentially all cases a_i can be decomposed into a linear operation that

weights the relative importance of the various inputs and a non-linear transformation $\sigma_i(z)$ which is usually the same for all neurons. The linear transformation in almost all neural networks takes the form of a dot product with a set of neuron-specific weights $w^{(i)} = \left(w_1^{(i)}, w_2^{(i)}, \ldots., w_d^{(i)}\right)$ followed by re-centering with a neuron-specific bias $b^{(i)}$:

$$z^{(i)} = w^{(i)} \cdot x + b^{(i)} \tag{4}$$

In terms of $z^{(i)}$ and the non-linear function $\sigma_i(z)$ we can write the full input-output function as:

$$a_i(x) = \sigma_i(z^{(i)}) \tag{5}$$

The basic idea of all neural networks is to layer neurons in a hierarchical fashion, the general structure of which is known as the network architecture. In the simplest feed-forward networks, each neuron in the input layer of the neurons takes the inputs x and produces an output $a_i(x)$ that depends on its current weights (eq. 4). The outputs of the input layer are then treated as the inputs to the next hidden layer. This is usually repeated several times until one reaches the top or output layer. The output layer is almost always a simple classifier of the logistic regression in case of categorical data or linear regression layer in case of continuous outputs. The use of hidden layers greatly expands the representational power of a neural net. The choice of exact network architecture for neural network remains an art that requires extensive investigations, but a general rule of thumb that seems to be emerging is that the number of parameters in the neural net should be large enough to prevent underfitting. [5]

Like all supervised learning procedures, the first thing one must do to train a neural network is to specify a loss function. Given a data point (x_i; y_i), the neural network makes a prediction $\hat{y}_i(w)$, where w are the parameters of the neural network. Recall that in most cases, the top output layer of our neural net is either a continuous predictor or a classifier that makes discrete (categorical) predictions. Depending on whether one wants to make continuous or categorical predictions, one must utilize a different kind of loss function.

For categorical data, the most commonly used loss function is the cross –entropy, since the output layer is often taken to be a logistic classifier for binary data with two type of labels, or a soft max classifier if there are more than two types of labels. The output of the top layer of the neural network is the probability $\hat{y}_i(w) = p(y_i|x_i; \boldsymbol{w})$ that data point I is predicted to be in category 1. The cross-entropy between true labels $y_i \in \{0; 1\}$ and the predictions is given by

$$E(w) = -\sum_{i=1}^{n} y_i log\, \hat{y}_i(w) + (1 - y_i) \log[1 - \hat{y}_i(w)] \tag{6}$$

More generally, for categorical data, y can take on M values so that $y \in \{0,1, \ldots, M-1\}$ for each datapoint i, define a vector y_{im} called a 'one-hot' vector, such that

$$y_{im} = \begin{cases} 1, & if\ y_i = \mathrm{m} \\ 0, & otherwise \end{cases} \tag{7}$$

We can also define the probability that the neural network assigns a datapoint to category m: $\widehat{y_{im}}(w) = p(y_i = m|x_i; w)$. Then, the categorical cross-entropy is defined as

$$E(w) = -\sum_{i=1}^{n} \sum_{m=0}^{M-1} y_{im} \log \widehat{y_{im}}(w) + (1 - y_{im}) \log[1 - \widehat{y_{im}}(w)] \tag{8}$$

As in linear and logistic regression, this loss function is often supplemented by additional terms that implement regularization.

Having defined an architecture and a cost function, we must now train the model. As with other supervised learning methods, we make use of gradient descent-based methods to optimize the cost function. Recall that the basic idea of gradient descent is to update the parameters w to move in the direction of the gradient of the cost function $\nabla_w \mathrm{E(w)}$.

Finally, we note that unlike in linear and logistic regression, calculating the gradients for a neural network requires a specialized algorithm, called backpropagation which forms the heart of any neural network training procedure. The backpropagation algorithm is a clever procedure that exploits the layered structure of neural networks to more efficiently compute gradients.

Backpropagation algorithm

Backpropagation is a common method for training a neural network. The goal of backpropagation is to optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs. [6]

Backpropagation is simply the ordinary chain rule for partial differentiation, and can be summarized using four equations. [7] In order see this, we must first establish some useful notation. We will assume that there are L layers in our network with $l = 1, \ldots, L$ indexing the layer. Denote by w_{jk}^l the weight for the connection from the k-th neuron in layer $l-1$ to the j-th neuron in layer l. We denote the bias of this neuron by b_j^l. By construction, in a feed-forward neural network the activation a_j^l of the j-th neuron in the l-th layer can be related to the activities of the neurons in the layer $l-1$ by the equation

$$a_j^l = \sigma\left(\sum_k w_{jk}^l\ a_j^{l-1} + b_j^l\right) = \sigma(z_j^l) \tag{9}$$

Where we have defined the linear weighted sum

$$z_j^l = \sum_k w_{jk}^l\ a_j^{l-1} + b_j^l \tag{10}$$

By definition, the cost function E depends directly on the activities of the output layer a_j^L. It of course also indirectly depends on all the activities of neurons in lower layers on the neural network through iteration of Eq. (8).

*Four backpropagation equations relating the gradients of the activations of various neurons a_j^l, weighted inputs z_j^l and the errorsΔ_j^l :

I. The error of neuron j in the layer l, Δ_j^l, as the change in the cost function the weighted input z_j^l:

$$\Delta_j^l = \frac{\partial E}{\partial z_j^l} = \frac{\partial E}{\partial a_j^l}\sigma'(z_j^l)$$

Where $\sigma'(x)$ denotes the derivative of the non-linearity $\sigma(\cdot)$ with respect to its input evaluated at x

II. The error function can also be interpreted as the partial derivative of the cost function with respect to the bias b_j^l

$$\Delta_j^l = \frac{\partial E}{\partial z_j^l} = \frac{\partial E}{\partial a_j^l}\frac{\partial b_j^l}{\partial a_j^l} = \frac{\partial E}{\partial b_j^l}$$

III. Since the error depends on neurons in layer l only though the activation of neurons in the subsequent layer l+ 1, we can use the chain rule to write

$$\Delta_j^l = \frac{\partial E}{\partial z_j^l} = \frac{\partial E}{\partial z_j^{l+1}}\frac{\partial z_j^{l+1}}{\partial z_j^l} = \sum_k \Delta_k^{l+1}\frac{\partial z_j^{l+1}}{\partial z_j^l} = \left(\sum_k \Delta_k^{l+1} w_{jk}^{l+1}\right)\sigma'(z_j^l)$$

IV. Differentiation of the cost function with respect to the weight w_{jk}^l as

$$\frac{\partial E}{\partial w_{jk}^l} = \frac{\partial E}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{jk}^l} = \Delta_j^l a_k^{l-1}$$

To sum up the Backpropagation Algorithm:

1. **Activation at input layer**: calculate the activations a_j^l of all the neurons in the input layer.
2. **Feedforward**: starting with the first layer, exploit the feed-forward architecture through Eq. (9) to compute z_j^l and a_j^l for each subsequent layer.
3. **Error at top layer**: calculate the error of the top layer using Eq. (I).
4. "**Backpropagate" the error**: use Eq. (III) to propagate the error backwards and calculate Δ_j^l for all layers.
5. Calculate gradient: use Eqs. (III) and (IV) to calculate $\frac{\partial E}{\partial b_j^l}$ and $\frac{\partial E}{\partial w_{jk}^l}$

We can now see where the name backpropagation comes from. The algorithm consists of a forward pass from the bottom layer to the top layer where one calculates the weighted inputs and activations of all the neurons. One then backpropagates the error starting with the top layer down to the input layer and uses these errors to calculate the desired gradients. This description makes clear the incredible utility and computational efficiency of the backpropagation algorithm. One can calculate all the derivatives using a single "forward" and "backward" pass of the neural network.

3.3 RANDOM FOREST

One of the most widely used and versatile algorithms in data science and machine learning is Random Forests (RF). Random Forests is an ensemble method widely deployed for complex classification tasks. A random forest is composed of a family of (randomized) tree-based classifier decision trees (discussed below). Decision trees are high-variance, weak classifiers that can be easily randomized, and as such, are ideally suited for ensemble-based methods. [5]

A decision tree uses a series of questions to hierarchically partition the data. Each branch of the decision tree consists of a question that splits the data into smaller subsets (See Fig. 3), with the leaves (end points) of the tree corresponding to the ultimate partitions of the data.

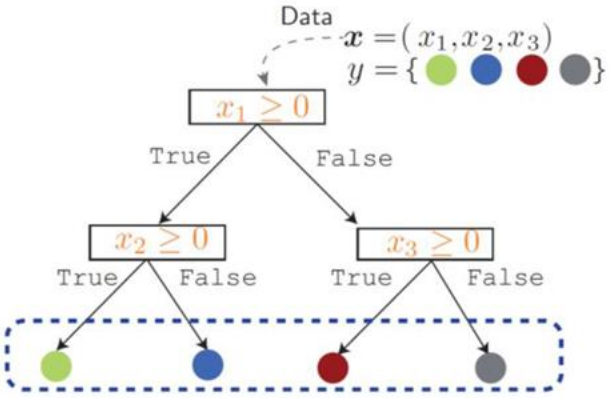

Figure 3 Example pf a decision tree.

When using decision trees for classification, the goal is to construct trees such that the partitions are informative about the class label (see Fig. 3). It is clear that more complex decision trees lead to finer partitions that give improved performance on the training set. However, this generally leads to overfitting, limiting the out-of-sample performance. For this reason, in practice almost all decision trees use some form of regularization (e.g. maximum depth for the tree) to control complexity and reduce overfitting. Decision trees also have extremely high variance, and are often extremely sensitive to many details of the training data. This is not surprising since decision trees are learned by partitioning the training data. Therefore, individual decision trees are weak classifiers. However, these same properties make them ideal for incorporation in an ensemble method.

In order to create an ensemble of decision trees, one must introduce a randomization procedure. Randomness is usually introduced into random forests in one of three distinct ways. The first is to use bagging and simply 'bag' the decision trees by training each decision tree on a different bootstrapped dataset. (however, it is more about bagged decision trees rather than random forest). The second procedure is to only use a different random subset of a features at each split in the tree. (this 'feature bagging' is the distinguishing characteristics of random forests, as the result reduction of correlations between decision trees. Finally, extremized random forests (ERFs) combine ordinary and feature bagging with an extreme randomization procedure where splitting is done randomly instead of using

optimality criteria. Even though this reduces the predictive power of each individual decision tree, it still often improves the predictive power of the ensemble because it dramatically reduces correlations between members and prevents overfitting.

3.4 DATA PREPARATION/PREPROCESSING

Some datasets have values that are missing, invalid, or otherwise difficult for an algorithm to process. If data is missing, the algorithm can't use it. If data is invalid, it causes the algorithm to produce less accurate or even misleading outcomes. [8]

Steps to consider while applying your ML algorithm:

1. Check the missing values in your data and clear them.
2. Clean the data and frame it in a structured manner to maintain the integrity.
3. Find the relevant features which account for the classification or regression during the training. Remove unwanted data to reduce the dimensions.
4. Depending on the data and your required output, choose the ML algorithm.
5. Split a small portion of training data into validation set to check if your model is overfitting during training. Correct it with regulariser, if there is any.
6. Keep the learning rate at optimum level to make sure the model does not overshoot or undershoot during error correction.

To avoid such problems and to be sure that data preparation was done well, we implemented pandas package (see appendix I) in our work. Scaling and normalizing a column in pandas python is required, to standardize the data, before we modelled a data. We used preprocessing method from scikitlearn package.

4 DATA EVALUATION

The metrics that you choose to evaluate your machine learning algorithms are very important. Choice of metrics influences how the performance of machine learning algorithms is measured and compared. They influence how one weights the importance of different characteristics in the results and your ultimate choice of which algorithm to choose.[10]

Metrics are demonstrated for both classification and regression type machine learning problems.

All recipes evaluate the same algorithms, Logistic Regression for classification and Linear Regression for the regression problems. A 10-fold cross-validation test harness is used to demonstrate each metric, because this is the most likely scenario where you will be employing different algorithm evaluation metrics.

A caveat in these recipes is the **cross_val_score** function used to report the performance in each recipe. It does allow the use of different scoring metrics, but all scores are reported so that they can be sorted in ascending order (largest score is best).

Some evaluation metrics (like mean squared error) are naturally descending scores (the smallest score is best) and as such are reported as negative by the cross_val_score()function. This is important to note, because some scores will be reported as negative that by definition can never be negative.

Classification Metrics

Classification problems are perhaps the most common type of machine learning problem and as such there are a myriad of metrics that can be used to evaluate predictions for these problems.

Classification accuracy

Classification accuracy is the number of correct predictions made as a ratio of all predictions made. It is suitable when there are an equal number of observations in each class (which is rarely the case) and that all predictions and prediction errors are equally important.

Logarithmic Loss

Logarithmic loss (or logloss) is a performance metric for evaluating the predictions of probabilities of membership to a given class. The scalar probability between 0 and 1 can be seen as a measure of confidence for a prediction by an algorithm. Predictions that are correct or incorrect are rewarded or punished proportionally to the confidence of the prediction.

Area under ROC Curve

Area under ROC Curve (or AUC for short) is a performance metric for binary classification problems. The AUC represents a model's ability to discriminate between positive and negative classes. An area of 1.0 represents a model that made all predictions perfectly. An area of 0.5 represents a model as good as random. ROC can be broken down into sensitivity and specificity. A binary classification problem is really a trade-off between sensitivity and specificity.

- Sensitivity is the true positive rate also called the recall. It is the number instances from the positive (first) class that actually predicted correctly.
- Specificity is also called the true negative rate. Is the number of instances from the negative class (second) class that were actually predicted correctly.

Confusion Matrix

The confusion matrix is a handy presentation of the accuracy of a model with two or more classes. The table presents predictions on the x-axis and accuracy outcomes on the y-axis. The cells of the table are the number of predictions made by a machine learning algorithm. For example, a machine learning algorithm can predict 0 or 1 and each prediction may actually have been a 0 or 1. Predictions for 0 that were actually 0 appear in the cell for prediction=0 and actual=0, whereas predictions for 0 that were actually 1 appear in the cell for prediction = 0 and actual=1. And so on.

Classification report

Scikit-learn does provide a convenience report when working on classification problems to give you a quick idea of the accuracy of a model using a number of measures.
The classification_report() function displays the precision, recall, f1-score and support for each class.

5 DESCRIPTION OF THE DATA

The project dataset represented the Taiwanese bank clients data who were credit card holder including information regarding: gender, age, education, marital status, bills and payment history. The set was dated in the period of bank system crisis, particularly 2000s. In order to overcome with crisis and keep the market power, the bank began to introduce consumer loans as a mass product. The result was a misuse of credit card capabilities and insolvency of the client.

Dataset included 30000 clients with total 23 variables, namely:

- X1: Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit.
- X2: Gender (1 = male; 2 = female).
- X3: Education (1 = graduate school; 2 = university; 3 = high school; 4 = others).
- X4: Marital status (1 = married; 2 = single; 3 = others).
- X5: Age (year).
- X6–X11: History of past payment. We tracked the past monthly payment records (from April to September,2005) as follows:
- X6 = the repayment status in September, 2005; X7 = the repayment status in August, 2005;. . ;X11 = the repayment status in April, 2005. The measurement scale for the repayment status is: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above.
- X12–X17: Amount of bill statement (NT dollar).
- X12 = amount of bill statement in September, 2005;
- X13 = amount of bill statement in August,2005; . . .;X17 = amount of bill statement in April, 2005.
- X18–X23: Amount of previous payment (NT dollar). X18 = amount paid in September, 2005; X19 = amount paid in August, 2005; . . .;X23 = amount paid in April,2005.

Based on these variables, it is possible to train a program for further prediction of potential risk clients.

Therefore, our essential role was to build and train a program for further analysis of huge range of data with multiple variances.

6 RESULTS AND DISCUSSION

The primary goal of this project was to obtain the same result as in the given article [1] both for logistic regression and neural networks.

Logistic regression

Our results obtained by logistic regression displayed the same behavior as presented in the article. (see Fig.5) therefore it is possible to conclude logistic regression is not suitable for this kind of task. One of the main reasons could be in high reliance on a proper presentation of data.

Logistic regression is not able to handle a large number of categorical features/variables. It is vulnerable to overfitting. Moreover, it is not performing well with independent variables that are not correlated to the target variable and are very similar or correlated to each other.

Consequently, there is a probability to improve results in logistic regression in case of alternative way of identification of important independent variables.

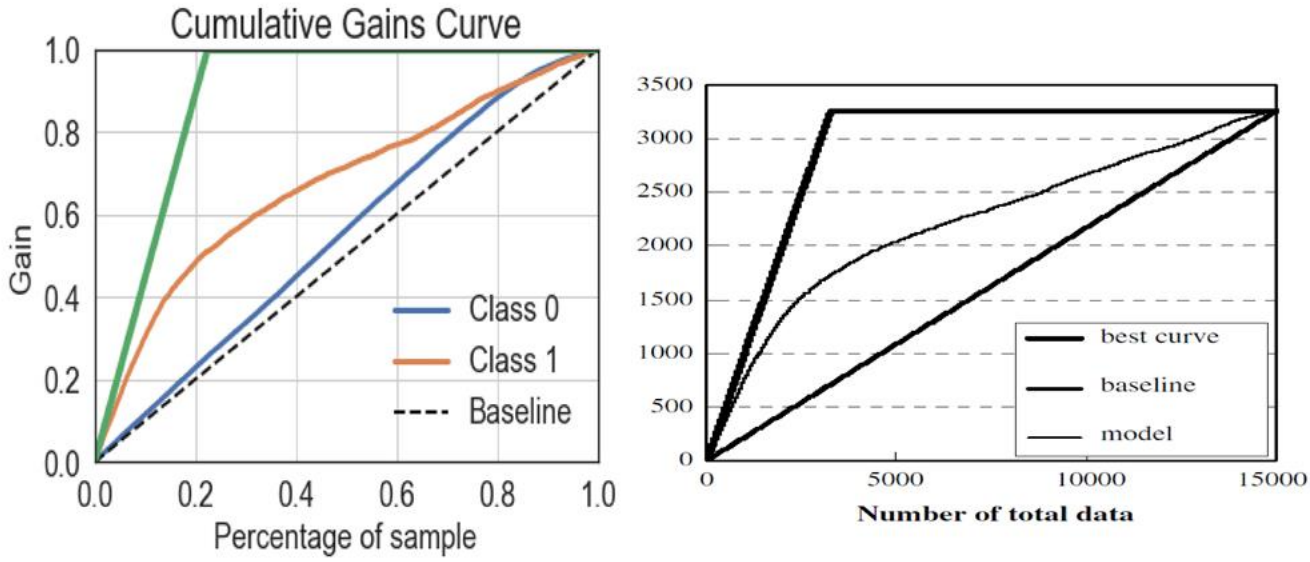

Figure 4: Cumulative gains chart of the results(left-project, right - article) from logistic regression.

Neural networks

We tested 3 different activation functions: sigmoid, tanh and softsign. The next step was to find the best architecture of our NN. Based on theoretical finding by Lippmann in the 1987 paper "An introduction to computing with neural nets" that shows that an MLP with two hidden layers is sufficient for creating classification regions of any desired shape we discovered that in order to deal with this task the simplistic model with 2 hidden layers was optimal choice. Then we moved to training our network. Training step was done with 100 epochs in order to find more preciously the global minimum. Obtained results are shown in the table below.

Table 1: Loss and accuracy for different activation functions

Activation function:	**Loss:**	**Accuracy:**
softsign	3.4841949780782064	0.7838333333333334
sigmoid	0.5145126843452453	0.7838333333333334
tanh	0.5230136462847392	0.7835

Based on Table 1, we see that all 3 functions show similar accuracy, but that tanh is a bit worse, than the other two. Other two activation functions have the same accuracy, but softsign has a much bigger loss, which means sigmoid is the best choice for this kind of NN.

Also, it is obvious that the neural network is better in prediction of results probability compare to logistic regression, as it was proved by Yeh and Lien [1].

7 CONCLUSION AND OUTLOOKS

In this project we were able to repeat the same results as Yeh and Lien in case of Logistic regression and Neural network. We did not perform Random Forests or Support Vector Machines analysis, which could possibly yield even better results.

Based on this data set we cannot be absolutely sure about suitability of algorithm and models for actual work, since we cannot fully rely on data collection in crisis year of 2005. Therefore, would be better to test algorithm on alternative dataset.

For better accuracy prediction of client's solvency, we recommend to take into account different parameters, such as salary rate and working experience, and status from other banks, for instance.

8 REFERENCES

1. Yeh Ch., Lien Ch. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. Expert Systems with Applications 36 (2009) 2473-2480
2. Navlani.A. Understanding Logistic Regression in Python. www.datacamp.com (2018)
3. Haghi A.K. et al. Application of Artificial Neural Networks and a Metaheuristic . Applied chemistry and chemical Engineering. Volume 3.p.359 (2018)
4. Shiffman D. The nature of code. Chapter 10. Neural Network (2012)
5. Mehta P. Neural Network Basics. A high – bias, low variance introduction to Machine Learning for physicists. p. 46 (2018)
6. Mazur M. A step by step Backpropagation Example. www.mattmazur.com (2015)
7. Mehta P. Deriving and implementing the backpropagation equations. A high – bias, low variance introduction to Machine Learning for physicists. p. 51 (2018)
8. www.datarobot.com/wiki/data-preparation/
9. Maladkar K. How to get started with preparing data for machine learning. Analytics India.(2018)
10. J. Brownlee. Metrics to evaluate machine learning algorithms in Python. Python Machine learning (2016) www.machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python/
11. Varun. Using pandas describe method to get dataframe summary. www.backtobazics.com (2018)