Charles University in Prague

Faculty of Mathematics and Physics

MASTER THESIS

Šimon Rajčan

Magazine sales prediction

Department of Software Engineering

Supervisor of the master thesis: prof. RNDr. Jaroslav Pokorný, CSc.

Study programme: Computer Science

Specialization: Database systems

Prague year 2013

I would like to thank to my supervisor prof. RNDr. Jaroslav Pokorný, CSc for mentoring of this work.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In Prague date 9.8.2013
signature

Název práce: Predikce prodejností magazínů

Autor: Šimon Rajčan

Katedra / Ústav: Katedra softwarového inženýrství

Vedoucí diplomové práce: Prof. RNDr. Jaroslav Pokorný, CSc., Katedra softwarového inženýrství

Abstrakt: Mnoho nakladatelstev se dnes potýká s problémem predikce prodejností vydávaných magazínů. V mnoha případech tyto predikce vykonávají zaměstnanci na základě svých subjektivních odhadů. Nevýhodou tohoto přístupu jsou zvýšené náklady na tvorbu predikcí a následné zvýšené náklady způsobené nepřesnostmi v těchto predikcích. Cílen této práce je prozkoumat existující regresní metody automatické predikce a zvolit řešení predikce prodejností magazínů v Ruském nakladatelství Burda.

Klíčová slova: Predikce, magazíny, neuronové sítě

Title: Magazine sales prediction

Author: Šimon Rajčan

Department / Institute: Department of Software Engineering

Supervisor of the master thesis: Prof. RNDr. Jaroslav Pokorný, CSc., Department of Software Engineering

Abstract: Today, many magazine publishing houses faces the problem of future predictions of their products. In many cases, these predictions are made by employees based on their personal experiences and guesses. The problems of this attitude are high expanses on making the predictions and increased expanses when those predictions are wrong. The aim of this work is to study existing regression methods of automatic prediction and create a solution for predicting the magazine sales in Russian publishing house Burda.

Keywords: Prediction, magazines, neural networks

Table of Contents

1. Introduction

Today, many magazine publishers face the problem of predicting sales of their products. Those predictions are in many cases made by few employees' guesses and very often are wrong. Publishers are forced to produce more magazines, in case there will be bigger demand for them. This has big influence on their profits in both cases. When predictions are made to satisfy increasing demand and the real demand is decreased, there are many magazines unsold. When predictions are made for decreasing demand and however the real demand is increased, there is a profit loss, because potential customers are not able to buy demanded magazines. We have access to data from a Russian publisher house named Burda. In this company, predictions are made by employees in excel tables and they often are inaccurate. If datamining application software was able to make those predictions, it would save noticeable expenses on publisher's side. Today, there are several regression methods, which could solve this problem. The aim of this work is to study the most common regression models, analyze them, if they are useful for described problem, implement the most convenient ones and make experiments over the data provided by publisher Burda. The most effective algorithm or combination of algorithms, including adequate documentation, should become a basis for application software. If the software performed well, it would be delivered to Burda.

The goal is to find a technique where the predictions have the smallest percentage of mean error. We discussed several regression algorithms and we choose Backpropagation technique, K-nearest neighbor technique and Classification and Regression trees (CART) technique for further experiments. We made experiments using three datasets consisting of magazine-publisher-pairs. Each dataset represented a class of magazine-publisher-pairs (large sales, medium sales and small sales). All experiments were made in Matlab. After several experiments and data modifications, we found Backpropagation configuration, where this technique worked best on large and medium sized data. For small data, K-nearest neighbors performed the best. We were disappointed from performance of CART, which showed bad results even after several experiments. Experiments shown that, using any technique, the performance depends deeply on input attributes.

The Knowledge from the experiments was used for creating the Burdasales software implemented in Matlab. Burdasales can connect to MSSQL database, import data, calculate and display predictions for preconfigured magazine-consumer pairs. For large and medium sized data, backpropagation technique was used and for small data, K-nearest neighbor’s technique was used.

In Section 2, initial analyzes over the data from Burda will be discussed. There will also be an overview of data quality enhancements made on those data. In Section 3, known regression models and techniques will be described, and there will be discussion over their usability for described problem. In Section 4, an implementation of chosen models will be presented and there will be a discussion, where the best model or combination of them, which solves this problem, will be chosen. In Section 5, architecture and deployment of application software will be described. In Section 6 will be conclusion and future work.

2. Burda data analysis

This section describes data provided for datamining and their basic preprocessing modifications, which were made before using them for datamining process.

2.1 Row data provided by Burda

Publishing house Burda stores their data in MSSQL database. We did not have full access to this database. The database is managed by single database administrator and this person is the only one, who understands the database model and relationships between the data. At first, four excel files were provided for data analyzes, which were meant to determine possibilities for automated predictions of future sales. Each file contained data from one year, so together data from January 2009 to December 2012 were available for datamining process. Structure of each excel file was as follows:

Column name	**Colunm meaning**	**Note**
ID номенклатуры	Row ID	
Артикул	Magazine ID	There are more than 80 magazine types.
Наименование номенклатуры	Magazine name	Each magazine has name contained of text part and number part such as Добрые советы. Судоку-02/1 or Лиза Girl-10/11
Контрагент	Consumer ID	There is more than 200 consumers.
Наименование контрагента	Consumer name	
Первый день продаж	Date of beginning the sales	
Окончание продаж	Date of ending the sales	
Окончание ремиссии	Date of ending of returns	

Заказано	Number of ordered magazines	
Отгрузка	Number of delivered magazines	
Возврат	Number of returned magazines	Number of total sales can be subtracted from

Table 2.1.1

2.2 First look at the data

Magazine ID determines the exact type and number of magazine, for example 983051stands for Playboy-05/11, which means Playboy, week number 5, year 2011. If the last three characters from Magazine ID are cut, it determines the magazine type. Magazine types are Playboy, Lisa, Dasa and other. This value could be used for unification of magazines of the same type, but one magazine type consist of regular magazines and Specials. Special is an edition of magazine, which is issued parallel with common number and usually is devoted to some specific theme. Predictions of Specials is not in scope of this work, they are issued irregularly and are very hard to predict.

There are many missed values in *Date of ending the sales, Date of ending of returns*, *Number of ordered magazines* and sometimes even in *Number of returned magazines*.

2.3 Data preprocessing

There was several data quality enhancements made on row data before using them as input data for datamining process. This chapter is dedicated to those enhancements.

Firstly, the column names were translated to English. After that, an analysis of each column was made. The new structure of data and exact knowledge from data analyzes are listed in the table below:

New column name	Description

ID	Source ID.
ID_MAGAZINE	ID of magazine. From this ID can be deduced the magazine type by cutting the last three numbers. This method could not be used, because there are also magazine specials with the same ID as ordinary magazines. Magazine specials are Magazines published together with ordinary magazines and predicting their sales is not in the scope of this work.
MAGAZINE_NAME	Full name of the magazine.
STD_MAGAZINE_NAME	New column. As the ID_MAGAZINE is not suitable for identifying the ordinary magazine type, this row vas created from MAGAZINE_NAME. All numbers in magazine name were replaced by “X” character. This method expressly identifies the magazine type, because magazine specials have text addition to ordinary name, such as: “Autocont 11/10 Christmas edition”.
ID_CONSUMER	Consumer ID.
CONSUMER_NAME	Full consumer name.
FIRST_DAY_OF_SALES	
END_OF_SALES	
END_OF_RETURNS	
ORDERED	Number of ordered magazines. This number is usually provided by magazine consumer, but number of delivered magazines does not have to depend on this number.
DELIVERED	Number of magazines, delivered to consumer.
RETURNED	Number of magazines, returned to publisher.
SOLD	*DELIVERED - RETURNED*
ENRICHED_SOLD	Discussed in Data Content section
HOLIDAYS_1_W	New column. It is a number, which specifies an amount of non business days in one week, beginning from first day of sales. (weekends and holidays)
HOLIDAYS_2_W	New column. It is similar to HOLIDAYS_1_W, but specifies number of non business days in two weeks from the beginning of sales.

HOLIDAYS_3_W	New column. It is similar to HOLIDAYS_1_W, but specifies number of not business days in three weeks from the beginning of sales.

Table 2.3.2

2.4 Data content

The data contains of total 303341 rows. The data consists of 315 magazine types and 250 magazine consumers. Each row has non null *DELIVERED* and *RETURNED* column. The values in those rows vary widely, depends on type of magazine and consumer. For deeper examination magazine Lisa and JSC Daily Press consumer, magazine Rest Times and Aria-aif consumer and magazine Autocont and SRO Press were selected. Each pair of selected magazine and consumer represents a group. In the first magazine-consumer pair sales varies from 5471 to 69186, so it represents a group of higher sales. In the second pair, sales vary from 2426 to 7249 and it represents group of medium high sales. In the last pair, values vary from 80 to 1333 and it represents the group of less volume sales.

If there are zero returned magazines, it probably means, that some potential customers were not able to buy the magazine, because it was not available. Aria-aif is a big consumer, which owns many newsstands. So even if there are some magazines returned, there still is a chance that some newsstands sold all their magazines, and potential customers were lost. From publisher Burdas experiences, the „healthy“ number of returned magazines is 25% - 30% of delivered magazines, which means, that if this number is smaller, that there were some empty newsstands, which is not desirable. Numbers over 30% means, that the number of delivered magazines was unnecessarily big. The aim of this work is to predict an amount of magazines, that reflects the real demand plus the healthy amount of returned ones, so the publisher’s houses can maximalize their profits. For this reason, there was created new column called *ENRICHED_SOLD*. Value of *ENRICHED_SOLD* was calculated as follows:

If number of returned magazines is higher than 25% of sold magazines

ENRICHED_SOLD = SOLD

If number of returned magazines is less than 25% of sold magazines

*ENRICHED_SOLD = SOLD + ((0.25 – (RETURNED / SOLD) * 100)^2 * 0.00032) * SOLD*

This equation says, that the maximum amount of magazines, which can be added to sold number is 20% of sold number. And number of magazines added to *ENRICHED_SOLD* increases quadratic from 0% to 25%.

In next sections, we will discuss several datamining regression methods and their usability over those data and make experiments on selected magazine-consumer pairs. The best technique will be selected based on those experiments.

3. Regression models and techniques analyzes

This section describes common techniques and algorithms in this area. While describing some techniques and other works, which use those techniques, conclusions will be made and used for choosing the models, which will be implemented in experiments section.

3.1 K-nearest neighbors

K-nearest neighbors is one of the oldest and very simple methods for classifying objects, which can be used for making predictions. The main idea is that the prediction value is among historical records where the predictor values are nearest to the unclassified record. A simple example of this method is determining, if a loan should be granted or not. Imagine a situation, where client wants to make a loan and the only think that is known about him is his address. There are historical data about other clients available that say where those clients live and if they were able to pay back the loan or not. The client is likely to pay back the loan, if most of his neighbors were able to pay back their loans, rather then they were not. Of course, there are other important aspects, which affect ability to pay back a loan, than just geographical neighborhood, like income or total property. The success of this method deeply depends on choosing the best predictor attributes and metrics for determining the nearness of records.

Calculating the K-nn [2]

The training examples are vectors in multidimensional space. Each example has assigned prediction value. The training phase of the algorithm consists only of storing the feature vectors and assigned prediction values. In the classification phase, *k* is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point. A commonly used distance metric for continuous variables is Euclidean distance. For discrete variables, such as for text classification, another metric can be used, such as the overlap metric (or Hamming distance). A drawback of the basic "majority voting" classification occurs when the class distribution is skewed. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the k nearest neighbors due to their large number. One way to overcome this problem is to weight the classification, taking into account the distance from the test point to each of its k nearest neighbors. The class (or value, in regression problems) of each of the k nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point.

The suitability of algorithm for our problem

The algorithm is old and very simple to understand. Other advantage is that this algorithm is implemented in many datamining tools. It was used in Price prediction of residential property [1], with fairly good results. Price predictions and sales predictions could work on similar principles.

For reasons described above, this algorithm will be included in experiments section.

3.2 Decision trees

As K-nearest neighbors it is very old method, therefore there are many types of trees and many studies discuss their effectiveness. Decision trees are way of representing a series of rules that lead to a class or value. Decision trees can be diverted to those, where one node has zero or two sub nodes (binary trees), or those, where node can have zero to n nodes. They can also be diverted on univariate or multivariate trees. Univariate trees can use only one predictor value in a single split and multivariate can use various combinations of many predictor values. Decision trees are grown through an iterative splitting of data into discrete groups, where the goal is to maximize the “distance” between groups at each split. Decision trees used to predict continuous variables are called *regression trees*. The most used regression trees algorithms are Chi-Squared Automatic Interaction Detection (CHAID) and Classification and Regression Trees (CART).

There are studies, which say, that there is no notable difference between those two techniques, like [4] and some which are a little more favorable to CART, like [5]. We decided to study CART algorithm for our purposes.

Formal definition of Decision tree [18]

Let *D* = {*t1* ..*tn*} be a database, *{A1, A2, ...An}* be an attributes *C* = *{C1, ...Cm}* be a set of classes.

Decision Tree is tree, where

- Each inside node has some value *Ai*
- Each edge has a predicate
- Each leaf has a value *Cj*

Solving classification or regression problem with decision tree consists of two steps:

- Building the tree using training data
- Using the tree to determine a class or predict the value

Building the tree algorithm

- Input : D // training data
- Output: T // decision tree

```
1. T = {}
2. Select best splitting criteria
3. T = create root and evaluate it with splitting criteria
4. T = Add edge for each splitting criteria and evaluate it
5. for each edge do
     a. D'= database created by using of splitting criteria on D;
     b. if ending criteria for the path
          i. T'= create leaf and evaluate it with class
     c. else
          i. T'= create decision tree D'.
6. T = add T' to edge
```

Using the tree for classification or regression

- In non leaf nodes are attributes used for splitting
- Edges relates to values of those attributes
- In leafs are information about assigned classes
- Tree is proceed from root to leafs according to attributes

Each way from root to leaf relates to one rule.

Entropy

Entropy is non orderliness in system.

$$H = -\sum_{t=1}^{T}(p_{\,t} \log_2 p_t)$$

p_t … likehood of class t appearance

t … number of classes

CART [17]

Leo Breiman and Jerome Friedman, later joined by Richard Olshen and Charles Stone, began work with decision trees when consulting in southern California in the early 1970s. They published a book and commerical software in 1984. Their program creates binary splits on nominal or interval inputs for a nominal, ordinal, or interval target. An exhaustive search is made for the split that maximizes the splitting measure. The available measures for an interval target are reduction in square error or mean absolute deviation from the median. The measure for an ordinal target is "ordered twoing.” The measures for a nominal target are reduction in the Gini index and "twoing.” Cases with missing values on an input are excluded during the search of a split on that input. Surrogate rules are applied to such cases to assign them to a branch.

Historically the most significant contribution is the treatment of over-fitting. The authors quickly concluded that an appropriate threshold for a stopping rule is unknowable before analyzing the data; therefore, they invented retrospective cost-complexity pruning: a large tree is created, then a subtree is found, then another sub-tree within the first, and so on forming a sequence of nested sub-trees, the smallest consisting only of the root node. A subtree in the sequence has the smallest overall cost among all sub-trees with the same number of leaves. For categorical targets, the cost is the proportion misclassified, optionally weighted with unequal misclassification costs. The final pruned tree is selected from subtrees in the sequence using the costs estimated from an independent validation data set or cross validation.

The program contains some other important features: prior probabilities that are incorporated into the split search; use of surrogate splits for missing values; a heuristic search for a split on linear combination of variables, the state of the art for about 15 years until the OC1 and SVM tree algorithms; evaluation of categorical trees in terms of purity of class probabilities instead of classification; bagging and arcing. The original commercial program is still available through Salford-Systems.

The suitability of CART to our problem

There are many implementations of CART algorithm and CART performed well in many studies, for example [16]. CART will be included in experiments section.

3.3 Backpropagation neural network

Neural networks are of particular interest because they offer a means of efficiently modeling large and complex problems in which there may be many predictor variables which many interactions.(Actual biological neural networks are incomparably more complex.) Neural networks may be used either for classification or for regressions problems.

Motivation

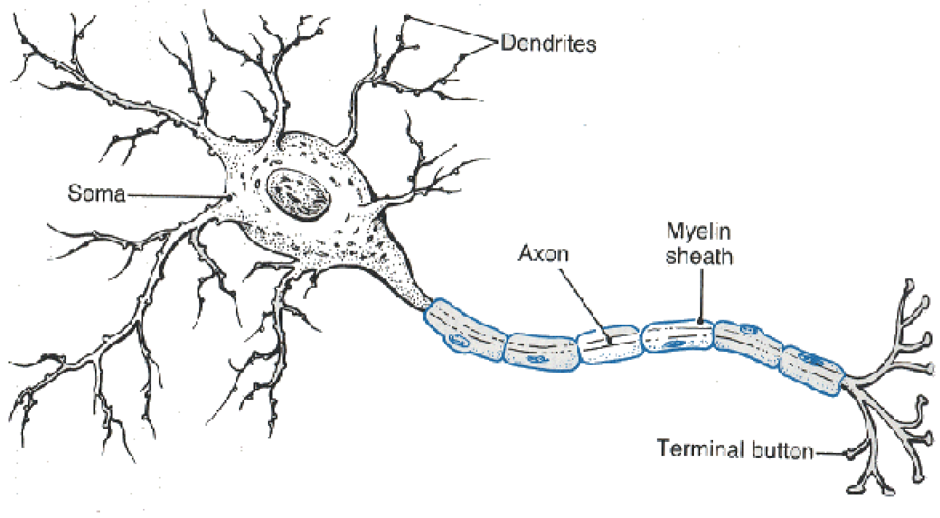


Figure 3.3.1 [7]

Neural networks are inspired by biological neurons. One biological neuron is displayed in Figure 3.9.1. The brain is a complex and huge connection of neurons, where each neuron collects information (signals) from other neurons with dendrites, and after a simple calculation inside its soma it generates its own information (signal) and distributes it to other neurons through its axon. So the axon terminal connects with other dendrites. This connection is called a synapse [6].

A synapse is the gate to transfer electro-chemical material from one cell to the other. This material can either downgrade (-) the soma load or upgrade (+) it. The synapses vary from each other by their effect which is called in AI terminology the *weight*. So each input signal, which enters the soma, is summed after it is multiplied by its weight. Now the cell has a load in itself. If this load is more than the cell can keep (greater than its threshold), a signal will be generated through the axon independent of how much the load is greater than the threshold [6].

So if we want to simulate this neuron, we will just use this simple math. Take the inputs, multiply each of them with its weight, sum up all, check if the total is bigger than the threshold or not. If yes fire the signal (1), if No keep silence (0). It is simple as illustrated in Figure 3.9.2.

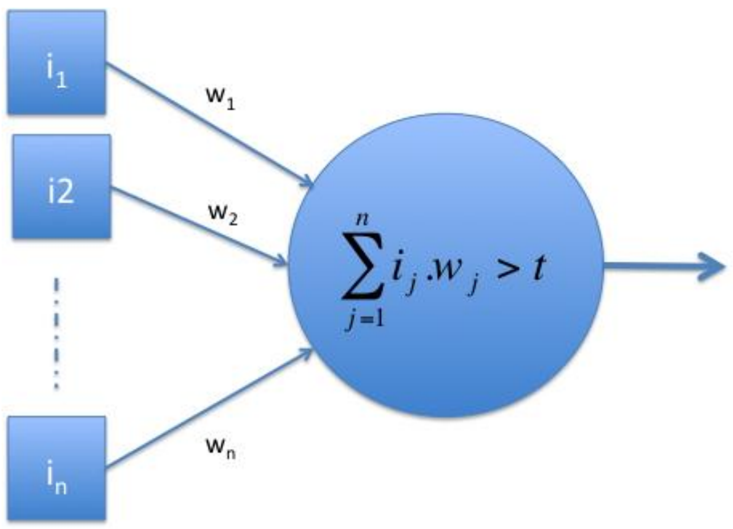


Figure 3.3.2[6]

Backpropagation

Backpropagation is probably the most known and used neural network. Using the backpropagation neural network consists of two processes. First is learning the network and second is using it.

Learning algorithm [8]

Learning algorithm can be divided into two parts:

- Propagation
    - Each propagation involves the following steps:
        - Forward propagation of a training pattern's input through the neural network in order to generate the propagation's output activations.
        - Backward propagation of the propagation's output activations through the neural network using the training pattern target in order to generate the deltas of all output and hidden neurons.

- Weight update
    - For each weight-synapse follow the following steps:
        - Multiply its output delta and input activation to get the gradient of the weight.

- Bring the weight in the opposite direction of the gradient by subtracting a ratio of it from the weight.

This ratio influences the speed and quality of learning; it is called the *learning rate*. The sign of the gradient of a weight indicates where the error is increasing. This is why the weight must be updated in the opposite direction. This process is repeated for each row in the training set. Each pass through all rows in the training set is called an *epoch*. The training set will be used repeatedly, until the performance of the network is satisfactory. At that point the neural network is considered to be trained to find the pattern in the test set.

Algorithm example

This is an example for 3-layer network, Figure 3.3.3.

```
initialize the weights in the network (often small random values)
  do
    for each example e in the training set
      O = neural-net-output(network, e)  // forward pass
      T = teacher output for e
      compute error (T - O) at the output units
      compute delta_wh for all weights from hidden layer to output layer  // backward pass
      compute delta_wi for all weights from input layer to hidden layer   // backward pass continued
      update the weights in the network
  until all examples classified correctly or stopping criterion satisfied
  return the network
```

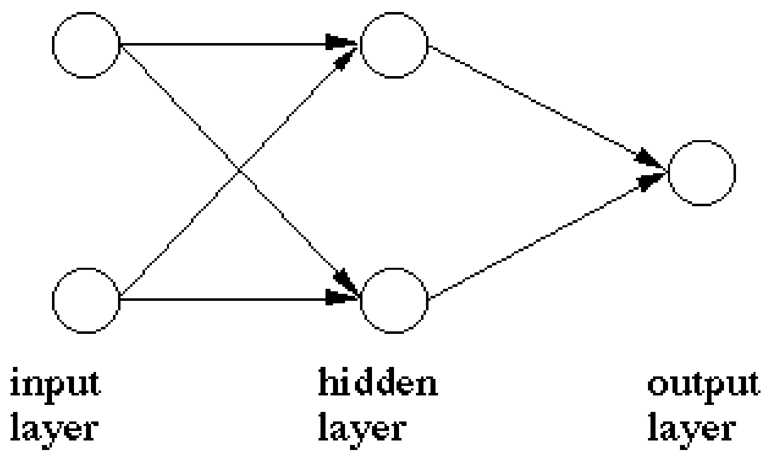


Figure 3.3.3[9]

The *architecture* (or topology) of a neural network is the number of nodes and hidden layers, and how they are connected. In designing a neural network, the number of hidden nodes and hidden layers, the activation function, and limits on the weights must be chosen.

Because so many parameters may exist in the hidden layers, a neural network with enough hidden nodes will always eventually fit the training set if left to run long enough. But how well it will do on other data? To avoid an overfitted neural network which will only work well on the training data, we must know when to stop training. Some implementations will evaluate the neural network against the test data periodically during the training. As long as the error rate on the test set is decreasing, training will continue. If the error rate on the test data goes up, even though the error rate on the training data is still decreasing, then the neural net may be overfitting the data, but this also is a matter of configuration.

Using the network

Using the learned network is very simple. It consists of one forward pass using the input values.

The suitability of technique for our problem

This technique will be used in experiments section. It was used on numerous studies of predictions witch good results, for example [3].

3.10 Random forest

The core of this section is taken from [15]

Random forests are an ensemble learning method for classification and regression. It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes output by individual trees. The original idea of this algorithm came from Leo Breiman and Adele Culter. ”Random Forests” is their trademark. This term is derived from random decision forests, which was proposed by Tin Kam Ho in 1995.

Bootstrap aggregating (Begging) [14]

Given a standard training set *D* of size n, bagging generates m new training sets *Di*, each of size n′ < n, by sampling from *D* uniformly and with replacement. By sampling with replacement, some observations may be repeated in each *Di*. If n′=n, then for large n the set *Di* is expected to have the fraction (1 - 1/e) (≈63.2%) of the unique examples of *D*, the rest being duplicates [13]. This kind of sample is known as a bootstrap sample. The m models are fitted using the above m bootstrap samples and combined by averaging the output (for regression) or voting (for classification).

Random forests method combines Breiman’s “bagging” and the random selection of features in order to construct a collection of decision trees with controlled variation.

Random forests are not a particular model, but they are more like a framework, which consists of interchangeable parts. Those parts can be mixed and matched to build many models. Constructing a model in this framework requires those choices:

1. The shape of the decision used in each node

2. The type of predictor to use in each leaf
3. The splitting objective to optimize in each node
4. The method of making randomness in the trees

Breiman's Algorithm

All trees are created with following algorithm

- Let the number of training cases be *N*, and the number of variables in classifier be *M*
- The number of input variables, which should be used to determine the decision at the node is m. m is much smaller than M
- From all training cases N choose n cases for training and use the rest for estimation of the error of the tree.
- For each node, from all M variables choose m variables and calculate the best split based on them in the training dataset.
- Each tree is fully grown and not pruned

For making prediction, using random forest, sample is calculated using all the trees and the final prediction is the mode vote of all of partial predictions.

The suitability of technique for our problem

This technique will not be included in experiments section. The technique is hard to implement and we already had bad results using the CART technique.

4. Experiments and implementation of chosen techniques

Experiments could not be effectively made and evaluated on all provided data. Popular magazines together with less popular and big consumers were chosen for this section. Firstly experiments were made on data from Lisa magazine for JSC Daily Press consumer and if algorithm performed well on those data, it was also verified on Dasa magazine with SRO Press consumer and Rest Times magazine with Aria-aif consumer.

Data used for experiments

Each dataset represents a class of magazine-consumer pairs. We define Large data as those, where average sales of one magazine number is above 20000, medium data as those, where average sales of one magazine number is between 2000 and 20000 and we define Small data as those, where average sales of one magazine number is less, then 2000.

Name	**Dataset 1**	**Dataset 2**	**Dataset 3**
Consumer name	JSC Daily Press	Aria-aif	SRO Press
Magazine name	Lisa	Rest Times	Autocont
Consumer_ID	0214	0019	0579
Magazine_ID **Sql regular expression**	Like '740[012345]% '	Like '742[012345678]%'	Like '743[012345]%'
Average sales of one magazine number	42702	603	4521
Class	Large data	Small data	Medium data

Table 4.0.1

4.1 Backpropagation technique

Matlab was chosen for this implementation, because it has Neural Network Toolbox, which provides Backpropagation technique. Another reason was that Matlab is also suitable for preprocessing of row data from Burda. It has graphical interface, so lot of technical work can be done through this interface. Each experiment was made by multiple training steps and the best, or interesting results are displayed in this section.

4.1.1 Dataset 1

Most of the experiments were made on top of these data, because we searched the best parameters and network layout on them.

4.1.1.1 Network 1

In first network, inputs and network arguments are selected intuitively.

Inputs:

- Last known value of ENRICHED_SOLD magazines. In Lisa it is 5 weeks ago.
- Value 6 weeks ago.
- Value 7 weeks ago
- Value 8 weeks ago
- Value 9 weeks ago
- Moving Average (10 last known values)
- Number of Holidays in week, when magazine was first published.
- Number of Holidays in 2 weeks, when magazine was first published.
- Number of Holidays in 3 weeks, when magazine was first published.

Building the network 1

The Network had 3 layers. It had 10 neurons in input layer, 10 neurons in hidden layer and one output neuron, which was desired value of predicted magazine sales. After several experiments, tansig transfer function was selected as better than purelin or logsig.

Training Parameters of network 1

Most of the parameters were set to default values, except max fail, which was set to 1000. And the goal was set to 1000.

Results of network 1

Matlab interface for neural networks – nntool splits the data set to 3 parts. 60% of data was used for network training, 20% for validation and last 20% for testing the network results. After several training attempts, we decided not to use any of common ways of endings the training process, because the results were very bad. We decided to stop the network training manually, when we saw, that the results on validation data does not improve. This means, that we can include the validation data to testing data, because they were not used directly for network training. All statistical results, from networks in this section are made from whole dataset. This means, that cases from training data are included in statistical results. This decision was made, because we think, that even statistics from those data can improve our future decision making about parameter selection, however in graphical results, training, validating and testing data are separated.

Minimal value	5471
Maximal value	69186
Mean error	5432,6
Percentage of mean error	12.5022 %
Standard deviation	4964,0
Variance	24641483,5

Median error	3619,7
Maximal error	25164

Table 4.1.2

Simple results

One way, to interpret the network strength is to compare it to simple way of prediction, where the prediction is the last known sales value. In this case it is 5 week old value. We will call this the simple results.

Mean error	9263,3
Percentage of mean error	20,87 %
Standard deviation	8309,3
Variance	69043785,3
Median error	6504
Maximal error	32368

Table 4.1.3

The graph below shows real results, network results and simple results

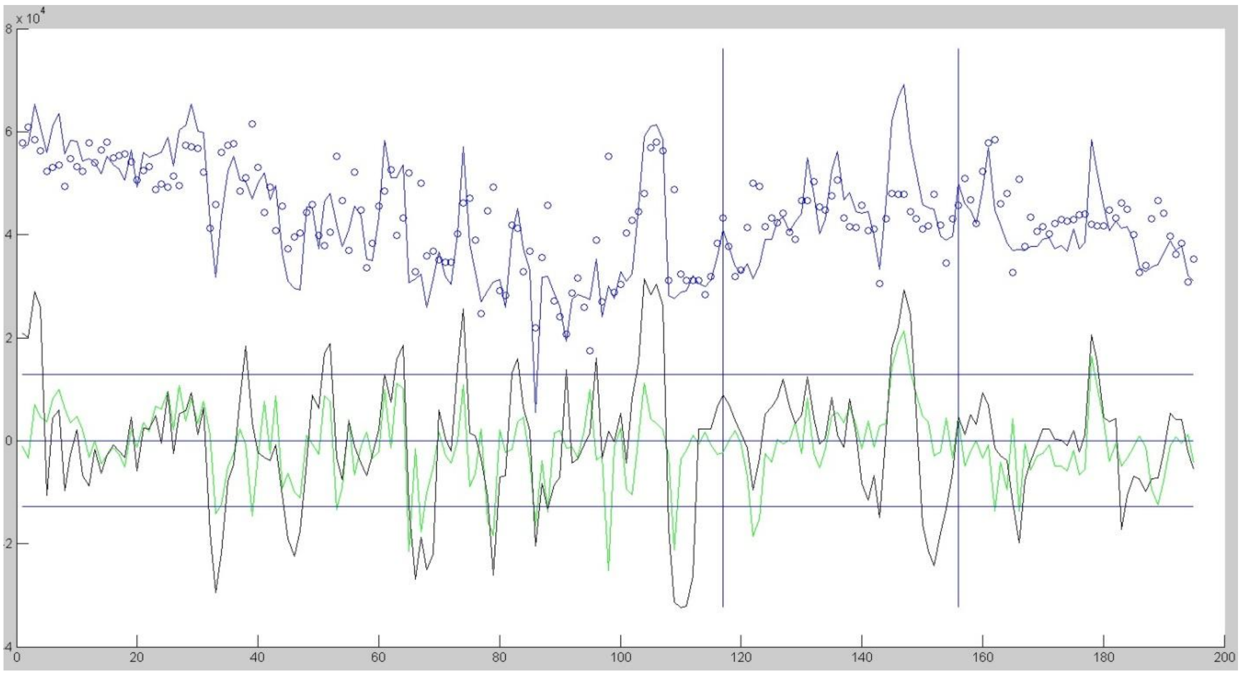


Figure 4.1.4

Where:

- Blue curve shows real results
- Blue circles shows network results
- Green curve shows network errors
- Black curve shows simple errors.
- Vertical lines splits the data in 3 parts (training data - 60%, validation data - 20%, testing data - 20%)

First network works fairly well, but there is space for enhancements. We changed the number of layers and numbers of neurons in hidden layers.

4.1.1.2 Network 2

In [3], backpropagation method is used for predicting the sales in retail store with good results. It is written there, that the best number of neurons in hidden layer is approximately half the number of input neurons. In their solution, there are much more input values, than in our case. For this reason, we decided to use 6 neurons in input layer.

Building the network 2

Inputs and training parameters are the same, as in previous experiment, except training goal, which was set to 2000 The network 2 has 10 input neurons, 6 neurons in hidden layer 1, 6 neurons in hidden layer 2 and one output layer.

Results of network 2

Mean Error	6209,07
Percentage of mean error	14,23%
Standard deviation	5822,14
Variance	33897363,79
Median error	4569,43
Maximal error	29484,51

Table 4.1.5

These results are even worse, than the network 1 results. On the graph below, it is seen, that it was probably caused by overfitting, because in many cases, the network 2 results are the same as simple results.

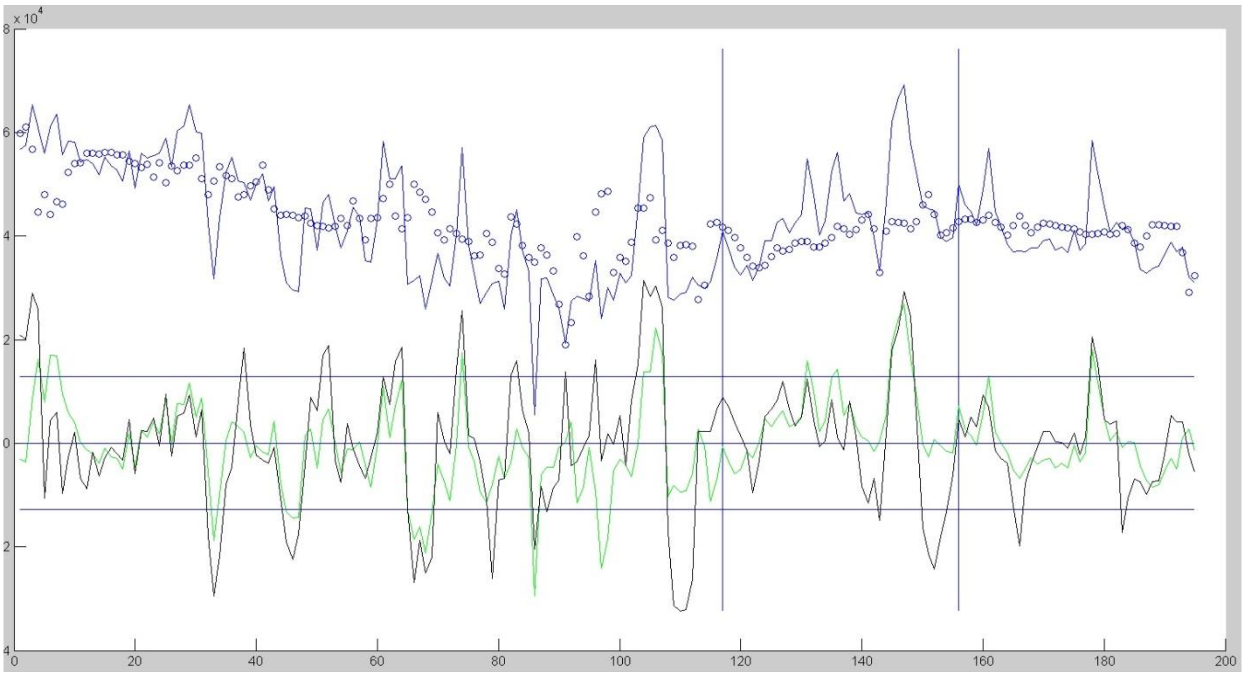


Figure 4.1.6

It is seen, on validation and testing data (last 40%), that predicted values are close together, the network 2 probably attaches more importance to moving average value.

4.1.1.3 Network 3

Adding second hidden layer did not help, so in third network, there was only one hidden layer. We left the number of neurons in hidden layer at 6. From the real results, it is seen, that in certain time in year, the sales increase periodically, so we decided to add time information to the network. Simple addition of date to the network would not help much, because for backpropagation network values like 30.6 and 1.7 are very different, but they are close in real time. Dates were transformed to day-number-in-year format, so for example date 1.3 has number 60.

Building the network 3

We changed the training goal to 1000 as in experiment 1 and all other training parameters were left as in experiment 2.

Results of network 3

Mean Error	6294,8
Percentage of mean error	14,51%
Standard deviation	5618,12
Variance	31563372,12
Median error	4863,96
Maximal error	32768,66

Table 4.1.7

Those results are the worst of all networks. It probably means that 6 neurons in hidden layer are not enough. Training of this network was not stopped by user, bud performance goal was met.

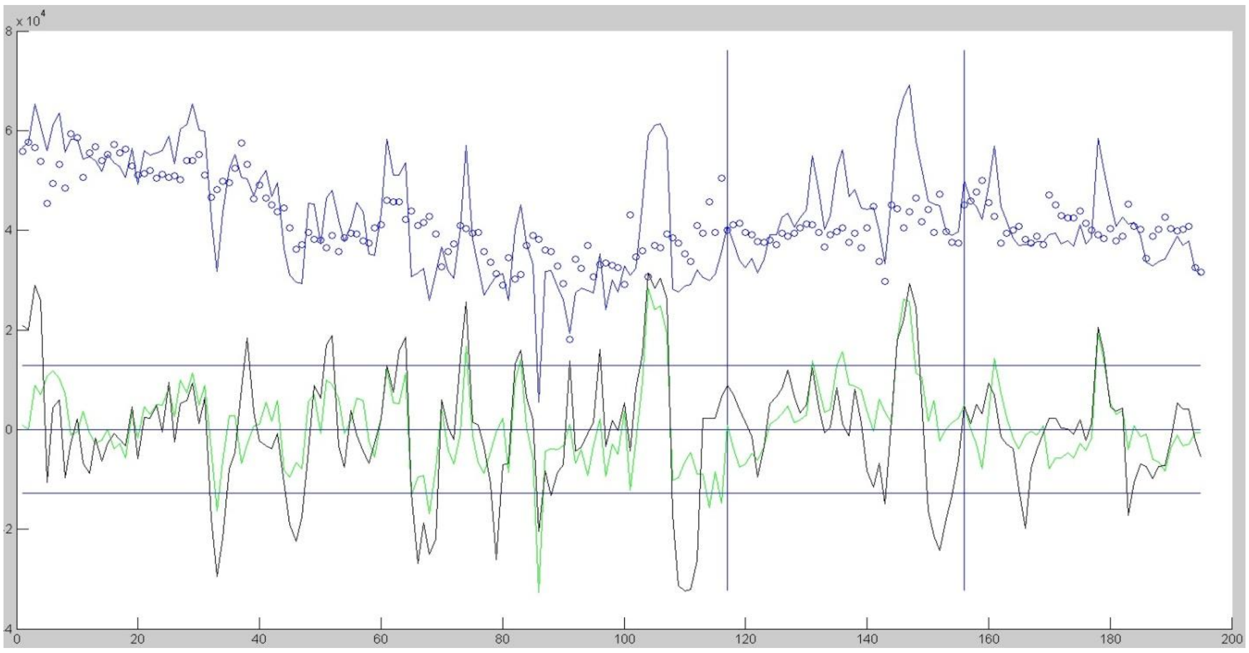


Figure 4.1.8

4.1.1.4 Network 4

The two last experiments shown, that 6 neurons in hidden layer are not enough. In fourth network we set the number of neurons in hidden layer to 10. First network shown, that backpropagation algorithm can work well, but it needs some better inputs. Added information about day number did not help. In network 4, we added second time information. We selected weeks in year, where sales raised, or dropped by 10% or more:

- Sales increase for 10% or more: +1 for the week number
- Sales dropped for 10% or more: -1 for the week number
- Other cases: no change
- Another option was, to add a value of sales exactly one year current day. But after close data examination it was shown, that there is no clear correlation between those two values. We call this value a *Sales increase number*

Building the network 4

The network has 13 input neurons, 10 neurons in hidden layer 1, 10 neurons in hidden layer 2 and one output neuron. Transfer function is tansig, for all layers. Performance goal was 1000.

Network 4 inputs

- Last known value of ENRICHED_SOLD magazines. In Lisa it is 5 weeks ago.
- Value 6 weeks ago.
- Value 7 weeks ago
- Value 8 weeks ago
- Value 9 weeks ago
- Moving Average (10 last known values)
- Number of Holidays in week, when magazine was first published.
- Number of Holidays in 2 weeks, when magazine was first published.
- Number of Holidays in 3 weeks, when magazine was first published.
- Moving Average (10 last known overall magazine sales)
- Moving Average (10 last known overall consumer sales)
- Day number in year
- Sales increase number

Results of network 4

Mean Error	3169,81
Percentage of mean error	6,95%
Standard deviation	4178,83
Variance	1746267,25
Median error	1414,03
Maximal error	26970,23

Table 4.1.9

These are the best results, of all 4 networks. It is clear, that adding a Sales increase number made a significant enhancement. These results were made after only 28 epochs of training. In previous networks, it took at least 100 epochs, to train the network.

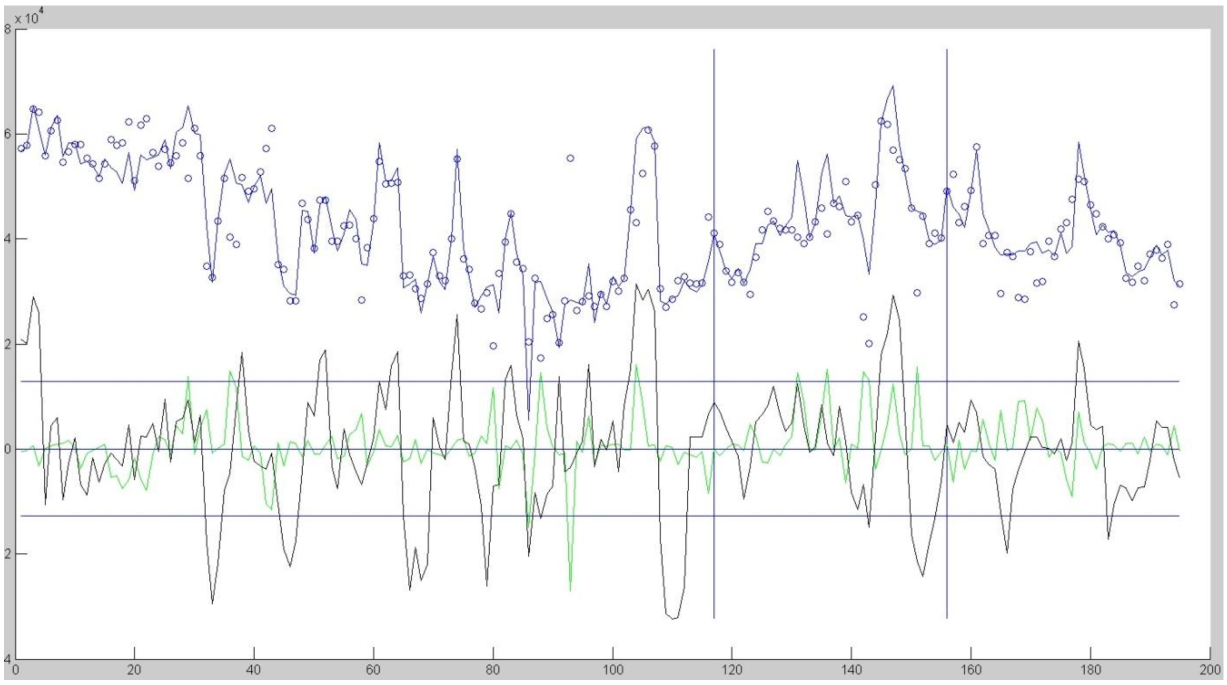


Figure 4.1.10

On testing, and validation data, it is seen, that results are almost perfect, except few cases. Those probably are some unpredicted events, which affected the sales.

4.1.1.5 Other attempts to enhance the network

We left the number of layers at two, but we tried to remove some original inputs, changed the numbers of neurons in several ways to enhance the performance, but any other networks did not reach the performance of fourth network, where all inputs were used and layers were in 13-10-10-1 layout.

4.1.2 Dataset 2

We build the network on with the same parameters and input columns as network 4, but this time we used data from magazine Dasa and consumer Yug Media Press.

Results of network 4 on dataset 2

Minimal value	80
Maximal value	1333
Mean error	116,02
Percentage of mean error	22.12%
Standard deviation	193,3
Variance	37367,66
Median error	29,09
Maximal error	804,99

Table 4.1.11

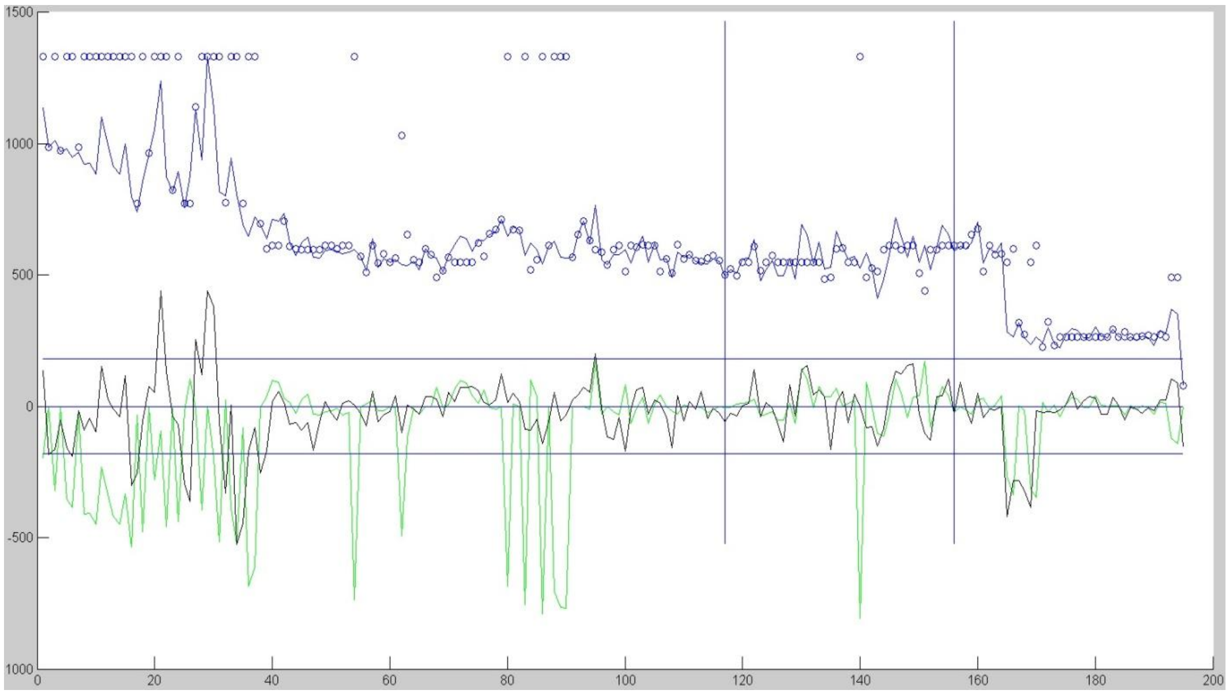


Figure 4.1.12

Those results are much worse, than any previous ones. It probably is because the value of sales is much smaller than in Lisa and JSC Daily press. After several attempts, the network was not able to learn predict other values then maximal ones. Values of sales vary between 80 and 1333, so this experiment showed that backpropagation technique does not performs well on small sales of magazines. We defined small sets of data as ones, where average sales are less than 2000.

4.1.3 Dataset 3

We build the network on with the same parameters and input columns as network 4, but this time we used data from magazine Rest times and consumer Aria-aif.

Results of network 4 on dataset 3

Minimal value	2426
Maximal value	7249
Mean error	306,95
Percentage of mean error	6,2%

Standard deviation	280,85
Variance	78880,92
Median error	239,34
Maximal error	1667,85

Table 4.1.13

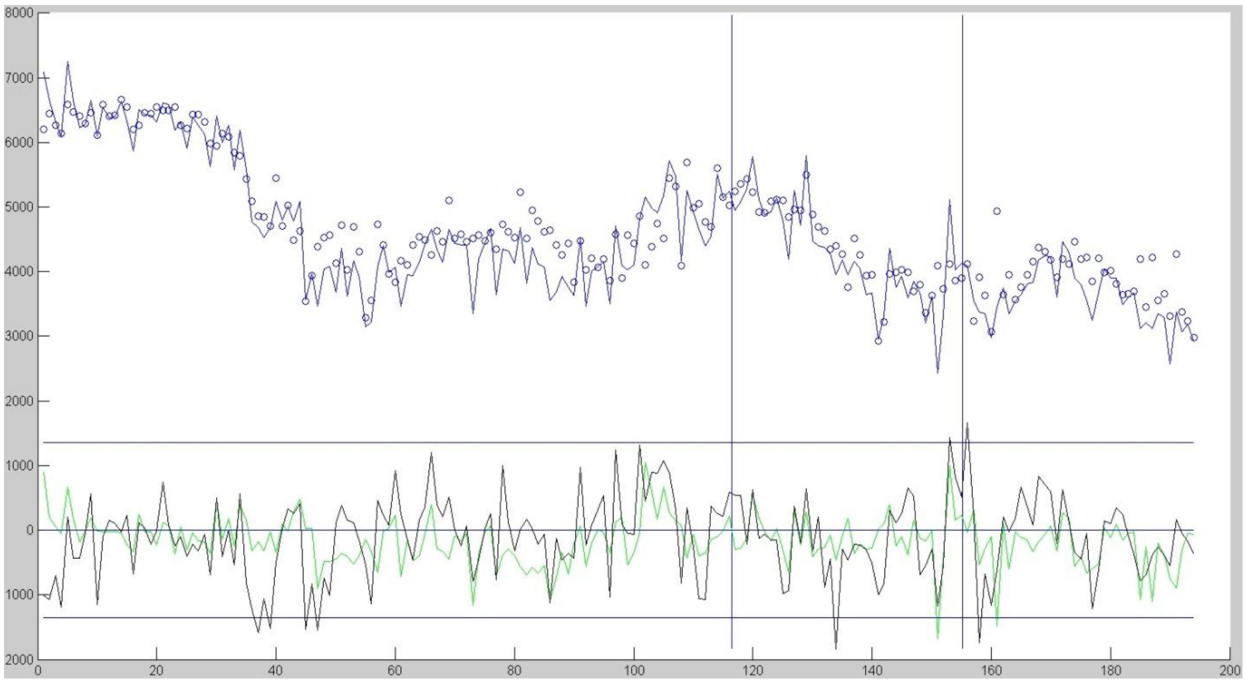


Figure 4.1.14

On third set of data, our network configuration performed very well. It is even better than on first set of data. Average error rate at 6.2% is very good and usable for purposes of Burda Publishers house. One of the reasons, why those results are good, is that the there are no big changes in volume of sales in short term. Values of sales varies between 2426 and 9364, which is more than in second set of data, but smaller than in third set. This experiment showed, that backpropagation network works well on medium sized sets of data.

4.1.4 The conclusion of Backpropagation technique

Backpropagation technique performed very well on first and third sets of data, but performed badly on small set of data. On first set, we also showed that results can improve significantly by adding even one important input. In our case, this input was Sales increase number.

4.2 K-nearest neighbors technique

This technique could be implemented in large scale of software, but we chose Matlab, because we already had preprocessed input data there from backpropagation technique and it provides comprehensible and simple graphical interpretation of data.

4.2.1 Dataset 1

We split the data to two parts. First 60% are training data and are only used for searching the nearest neighbors. Second 40% are testing data.

4.2.1.1 K-nn Experiment 1

The algorithm is very simple. For every value from testing data, it finds the nearest K neighbors within all older values, and returns mean value of those K values. Every new test case uses more values, as training data, as in last case. For example test case number 180 can use 179 cases as training data, but test case number 181 can use 180 cases as training data.

We used the same inputs, as in network 4 in previous section. K was set to 5 and we used Euclidean metric for all experiments in this chapter

K-nn Experiment 1 results

In Backpropagation technique, we displayed results on whole dataset. But in K-nn, we only used last 40% of data (testing data) for statistics.

Minimal value	5471
Maximal value	69186
Mean error	6335,48
Percentage of mean error	14,57%
Standard deviation	4922,42

Variance	24230259,15
Median error	5428,59
Maximal error	20562,2

Table 4.2.1

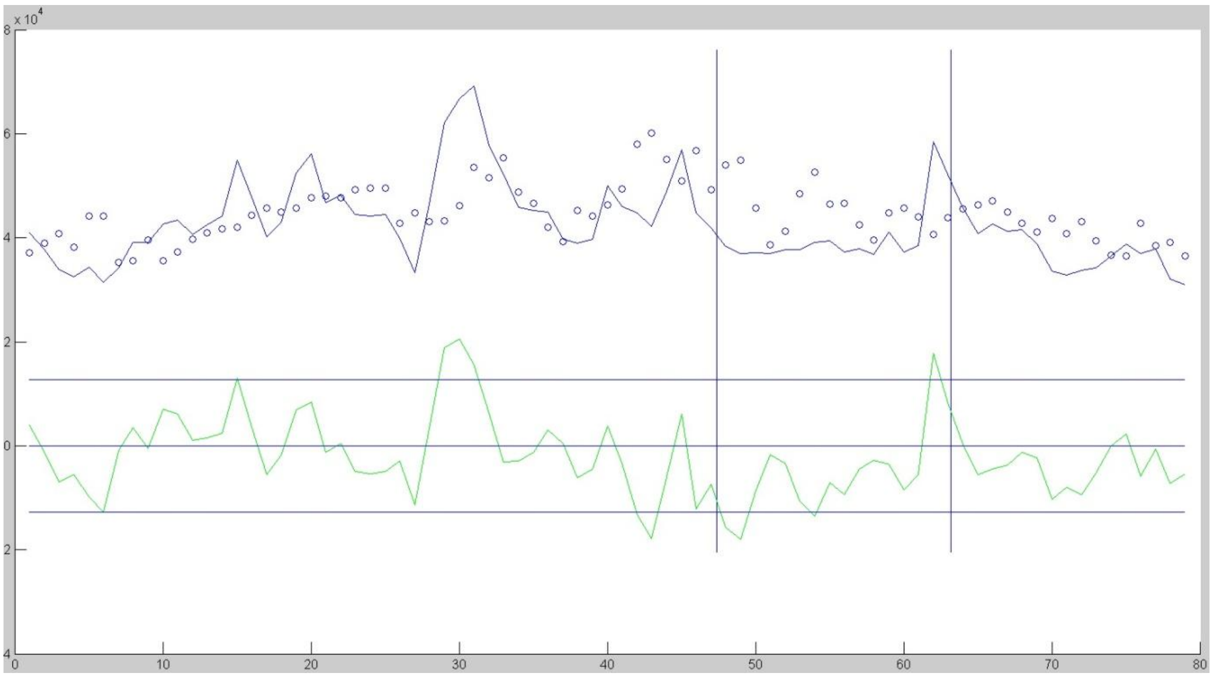


Figure 4.2.2

For K-nearest neighbors, we used slightly different graph. Only last 40% of data (testing data) are shown.

- Blue curve shows real results
- Blue circles shows predicted results
- Green curve shows prediction errors.

On Figure 4.2.2 is seen, that this experiment did not recognize any increase or decrease of sales. It seems that predicted values are some kind of average of last 5 input values. 13 inputs is probably a lot for this technique. In next experiment, we tried to reduce, the number of inputs.

4.2.1.2 K-nn Experiment 2

In this experiment, we tried to reduce the number of inputs, to increase the performance. We tried to remove several input values, mostly moving averages, because in experiment 1, the predictions were very similar to mean value of previous cases. The best results come from this inputs.

Experiment 2 inputs

- Last known value of ENRICHED_SOLD magazines. In Lisa it is 5 weeks ago.
- Value 6 weeks ago.
- Value 7 weeks ago
- Moving Average (10 last known values)
- Number of Holidays in 3 weeks, when magazine was first published.
- Sales increase number

We tried to set K to 3, 5 and 10, but the best results come from K set to 10.

K-nn Experiment 2 results

Mean error	7057,77
Percentage of mean error	15,86%
Standard deviation	5088,71
Variance	25895032,09
Median error	6212,59
Maximal error	21365,3

Table 4.2.3

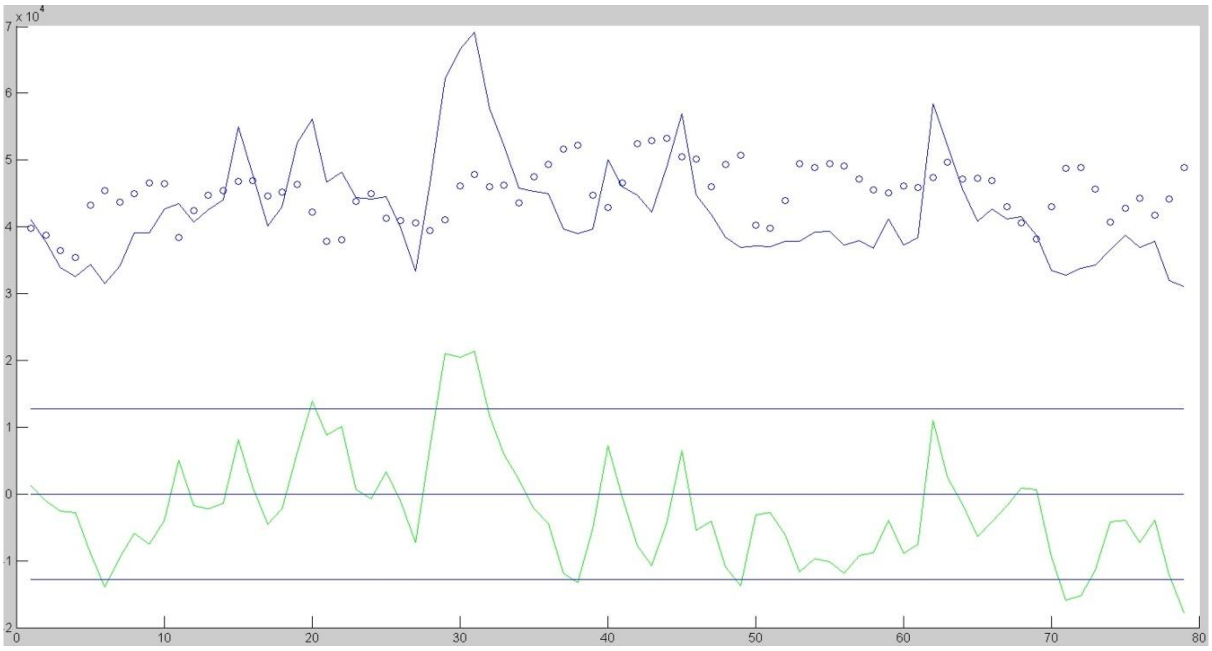


Figure 4.2.4

Decreasing the number of inputs did not help, so we decided to change an algorithm a bit. We decided to evaluate the nearness of neighbors, and assign more weight to the closer ones.

4.2.1.3 K-nn Experiment 3

We changed the algorithm of calculation the mean of K neighbors. The weight of neighbor raises quadratic regarded to his nearness:

```
weightIndex = K;
divisor = 0;
result_from_KNN = 0;
for i=1:length(IDX)  do
    neighborWeight = weightIndex * weightIndex;
    weightIndex = weightIndex - 1;
    divisor = divisor + neighborWeight;
result_from_KNN = result_from_KNN + (NNtargets(IDX(i)) * neighborWeight);
end
result_from_KNN = result_from_KNN / divisor;
results_from_KNN = [results_from_KNN, result_from_KNN];
```

K-nn Experiment 3 properties

We changed the number of inputs back to 13, as in backpropagation network number 4 and K was set to 10.

K-nn Experiment 3 results

Mean error	6150,72
Percentage of mean error	13.97%
Standard deviation	4527,67
Variance	20499858,47
Median error	4619,40
Maximal error	18397,96

Table 4.2.5

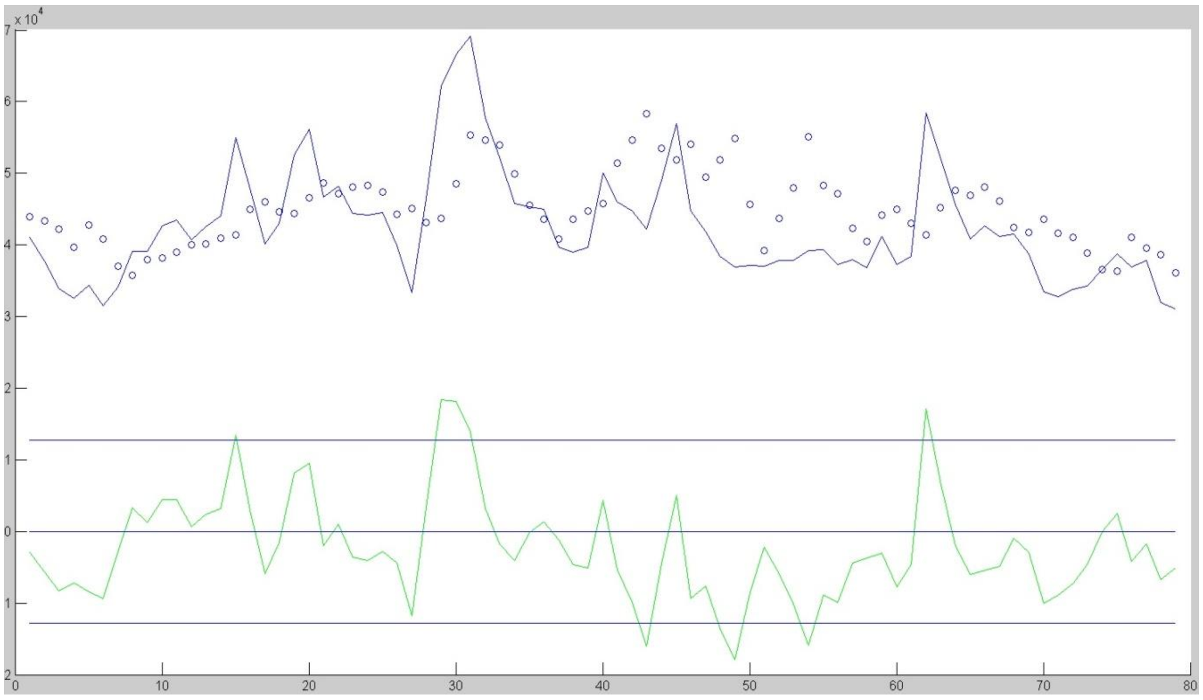


Figure 4.2.6

This algorithm performed better than previous ones, but the results still are not as good as with backpropagation method. From the figure is seen, that the predictions copy the peaks of a little, but in some cases, the predicted values are very wrong. This proved that the K-nearest neighbor method performs worse on big sized sets of data, than Backpropagation method.

4.2.2 Dataset 2

We used the same algorithm and parameters as in Experiment 3.

4.2.2.1 K-nn Experiment 4 results

Minimal value	80
Maximal value	1333
Mean error	68,93
Percentage of mean error	13,64%
Standard deviation	71.51
Variance	5114,87
Median error	42,72
Maximal error	313,18

Table 4.2.7

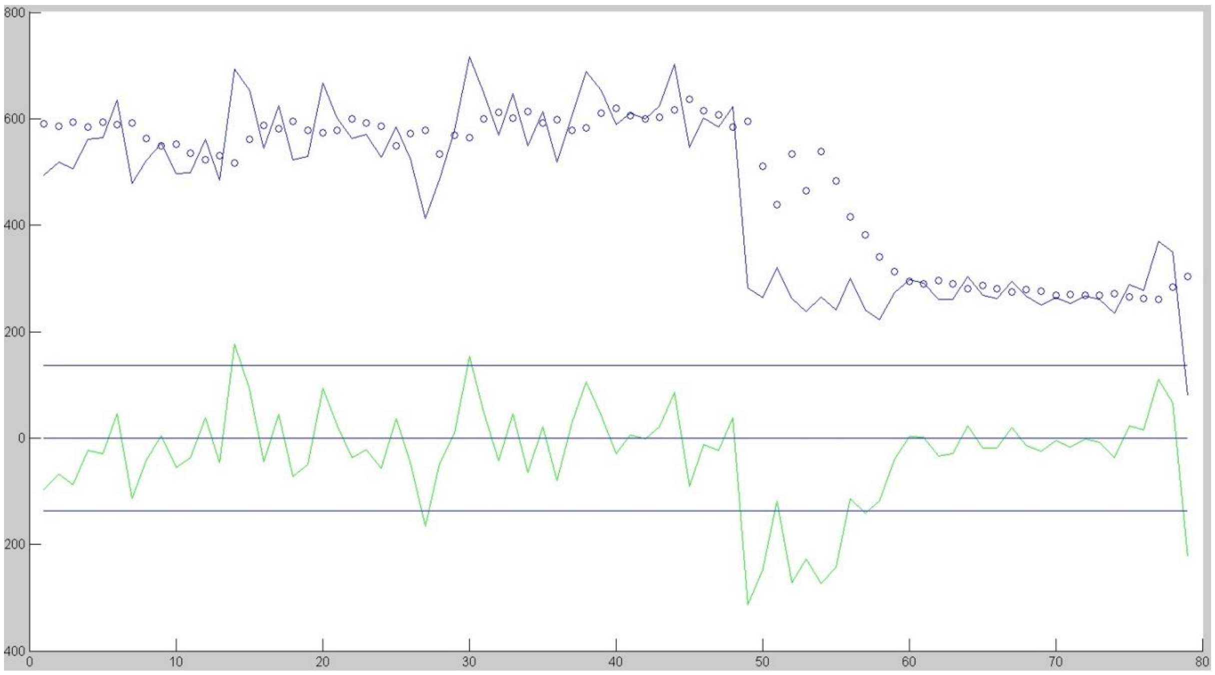


Figure 4.2.8

The figure 4.2.8 shows, that this method performs reasonably good on small sets of data. When there is a big decrease or increase in sales, it does not adapt immediately, but after few test cases the results are good again. The 13,6% mean percentage of error is better, than in backpropagation method.

4.2.3 Dataset 3

We used the same algorithm and parameters as in Experiment 3 and 4.

4.2.3.1 K-nn Experiment 5 results

Minimal value	2426
Maximal value	7249
Mean error	467,51
Percentage of mean error	11.15%
Standard deviation	385,42
Variance	148549,25
Median error	388,59
Maximal error	1848,98

Table 4.2.9

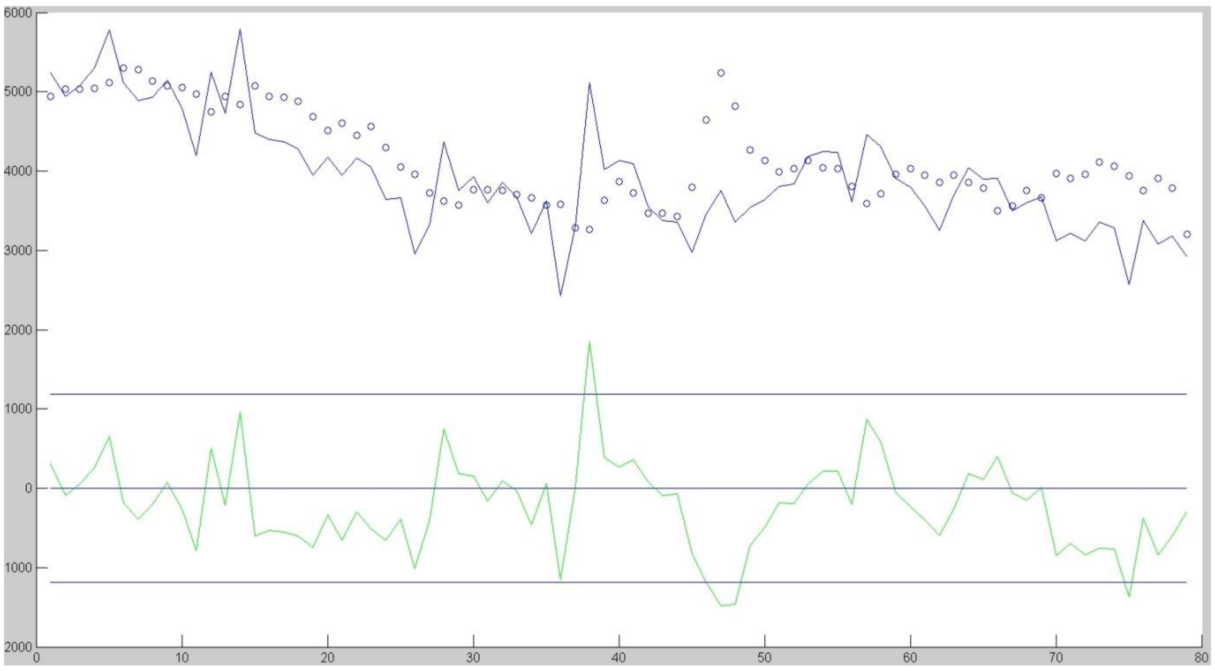


Figure 4.2.10

Percentage of mean error is smaller, than on first set of data, but volatility of input data is smaller.

4.2.4 The conclusion of K-nearest neighbor technique

K-nearest neighbor method performed worse, than backpropagation method on big and medium-sized data sets, but it performed better on small sets of data.

4.3 Classification and Regression trees

Just like in previous experiments, Matlab provides CART implementation.

4.3.1 Dataset 1

We used prepared data sets form Backpropagation technique. We split the data to two parts. First 60% are training data and are only used for searching the nearest neighbors. Second 40% are the testing data.

4.3.1.1 CART Experiment 1

In first experiment, we used the same inputs as in backpropagation experiment number 4. This input has 13 attributes.

CART Experiment 1 results

The results are made only on testing data.

Minimal value	31002
Maximal value	69186
Mean error	1.0671,76
Percentage of mean error	28.85%
Standard deviation	7796,27
Variance	60781927,11
Median error	8780,98

Maximal error	29584,33

Table 4.3.1

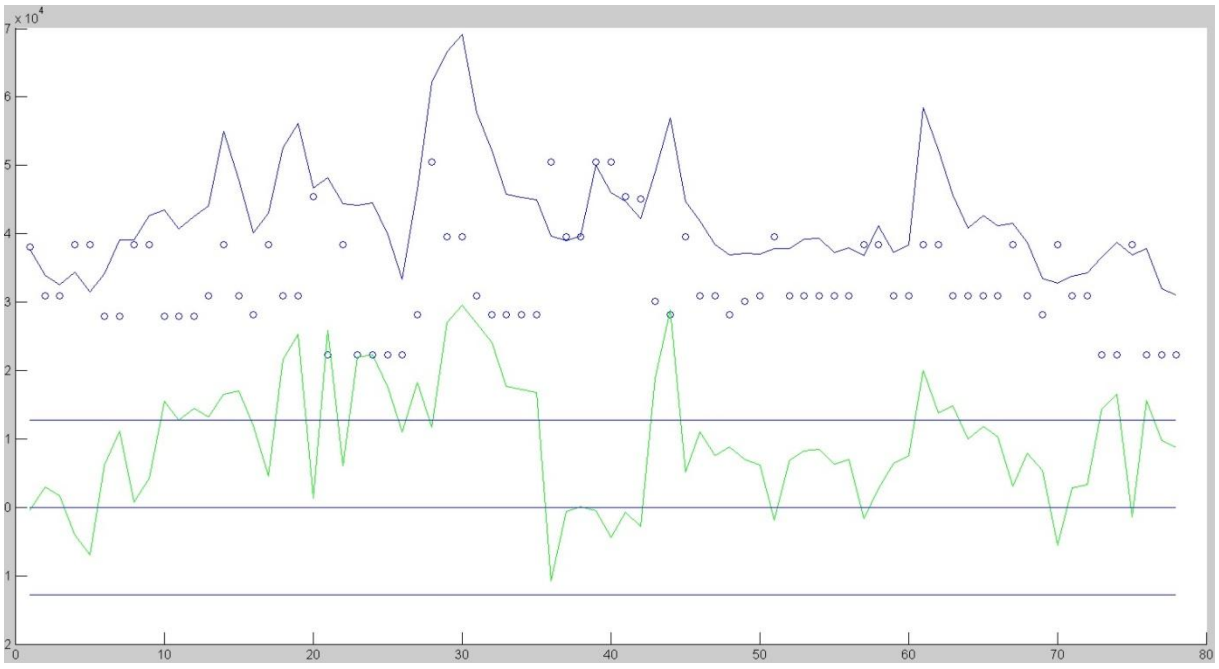


Figure 4.3.2

The results are the worst of all techniques. There probably are two reasons, why this is happening. Firstly, decision trees ted to overfit the training data [11]. Secondly, we used 60% of data for tree induction, while in some cases we could use more, like in K-nn technique.

4.3.1.2 CART Experiment 2

In this experiment we use the same input data, as in experiment 1, but for each test case, we create new tree, from all inputs available.

Experiment 2 results

Mean error	7268,57
Percentage of mean error	17,1%
Standard deviation	5776,17
Variance	33364244,58

Median error	6095,87
Maximal error	24331,5

Table 4.3.3

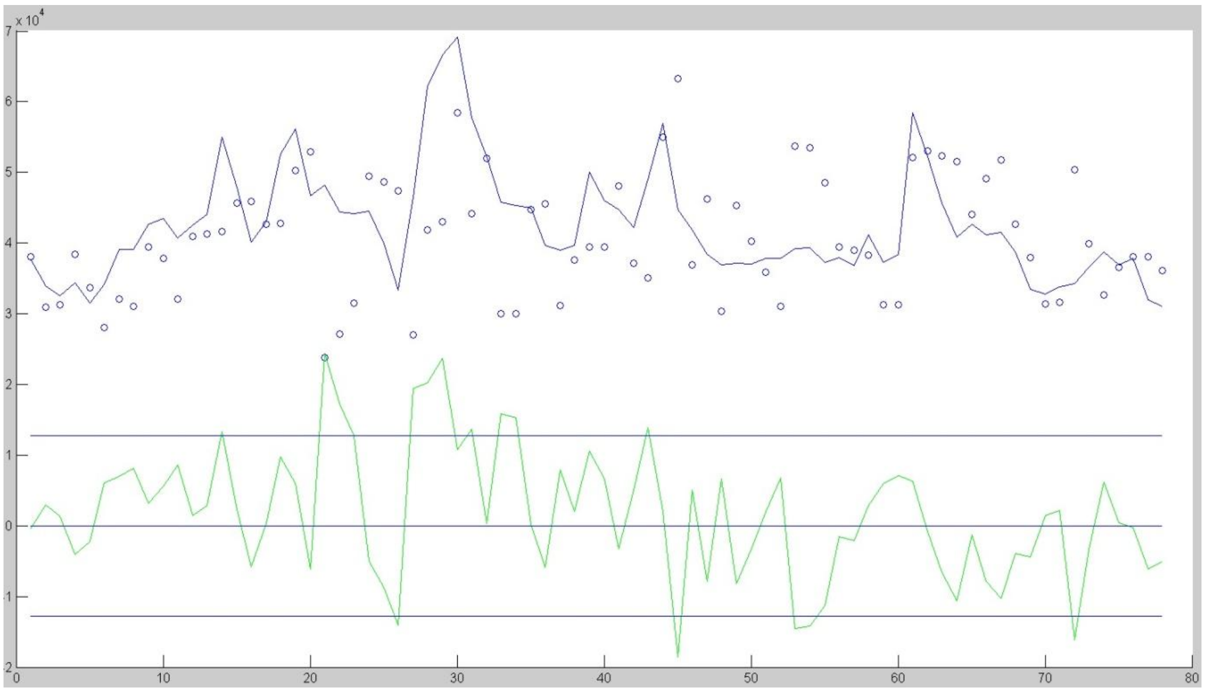


Figure 4.3.4

The performance increased significantly, but overall, it is still much worse, than backpropagation technique.

We skipped the third set of data, because it is clear, that this technique cannot outperform backpropagation technique on large and medium sized data. We will try, if it can get better, than K-nn technique on small data.

4.3.2 Dataset 2

CART technique performed bad on big data. We wanted to measure the performance on small data, to determine, if CART can outperform K-nn technique on them.

4.3.2.1 CART Experiment 3

We the same algorithm as in experiment 2 and 13-column input format, like in experiment 1 or experiment 2.

Experiment 3 results

Minimal value	80
Maximal value	717
Mean error	80,37
Percentage of mean error	17,19%
Standard deviation	80,57
Variance	6491,91
Median error	50,5
Maximal error	352,25

Table 4.3.5

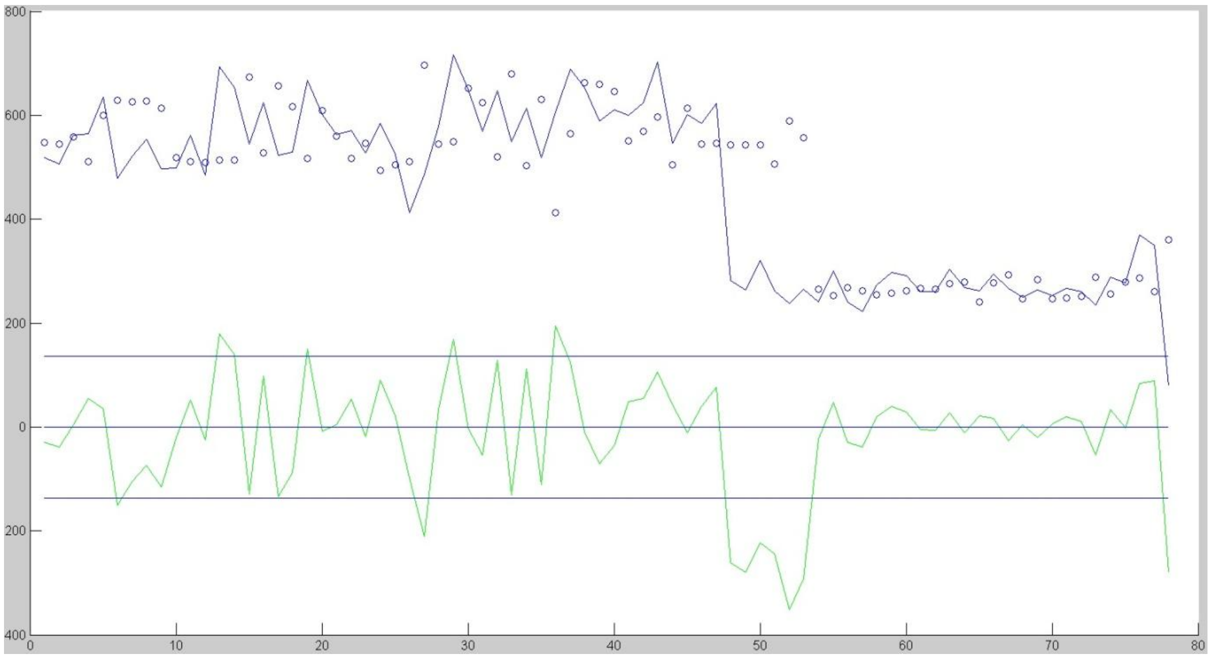


Figure 4.3.6

The results are better, than neural network results, but worse than K-nn. We decided not to continue with any other experiments with CART technique, because they perform badly for our problem.

4.3.3 The conclusion of CART technique

CART is not suitable for our problem. It performed badly on all experiments, so we finished the experiments even before trying all sets of data.

4.4 Experiments conclusion

The first method, we made experiments on was backpropagation. When we used first intuitive set of input values, default network learning configuration and intuitive network layout, we got the 12,5% of mean error rate on first dataset(large magazine-consumer pair). Removing some inputs made the results worse, so we tried to find more results and we tried to change the configuration and the structure to increase the performance. The turning point of our effort was adding a *Sales increase number*. The best results on first dataset were 6,9%, which we got with network number 4. Compared to 20,8% of simple results, this performance is very good. We used the same network layout and parameters on dataset 2 (small magazine-consumer pair), but it performed badly. Performance on third dataset (small magazine-consumer pair) was even better than on first one.

Second method was K-nearest neighbors. We made first experiment using the same inputs as in the best backpropagation experiment on first dataset. Firstly, we used the simplest K nearest method and K was set to 5. After several experiments, we changed K to 10 and improved the weight of the neighbors. Closer neighbors got more weight in resulted prediction. We also added every possible case to the training data, to increase the performance. Even after this effort, the resulted performance increased just lightly. The final performance was just under 14% of mean error rate. On second dataset, the mean error rate was 13,6% and on third dataset 11,1%. Backpropagation method clearly outperformed K-nearest neighbors on large and medium sized data, but K-nearest neighbors performed slightly better on small data.

Third method, we made experiments on was Classification and regression trees. This method was the biggest disappointment, because it showed bad results on all tested datasets. We finished the experiments even before trying it on all prepared datasets.

Best method for our problem

Based on our experiments, we decided to use neural backpropagation technique on large or medium sized datasets and K-nearest neighbor technique on small datasets. With K-nearest neighbors, it is simple, because algorithm does not need training for each magazine-consumer pair. With backpropagation technique, we need to train and store a network for each magazine-consumer pair.

In the next section, architecture of experiments and final solution will be discussed.

5. Architecture and deployment

In the fourth section, we tried several datamining regression methods. In this section we used that knowledge to implement the best model. Architecture of this implementation together with implementation of experiments will be described.

5.1 Architecture of experiments

All experiments were implemented in Matlab. The high level architecture is shown on Figure 5.1.1.

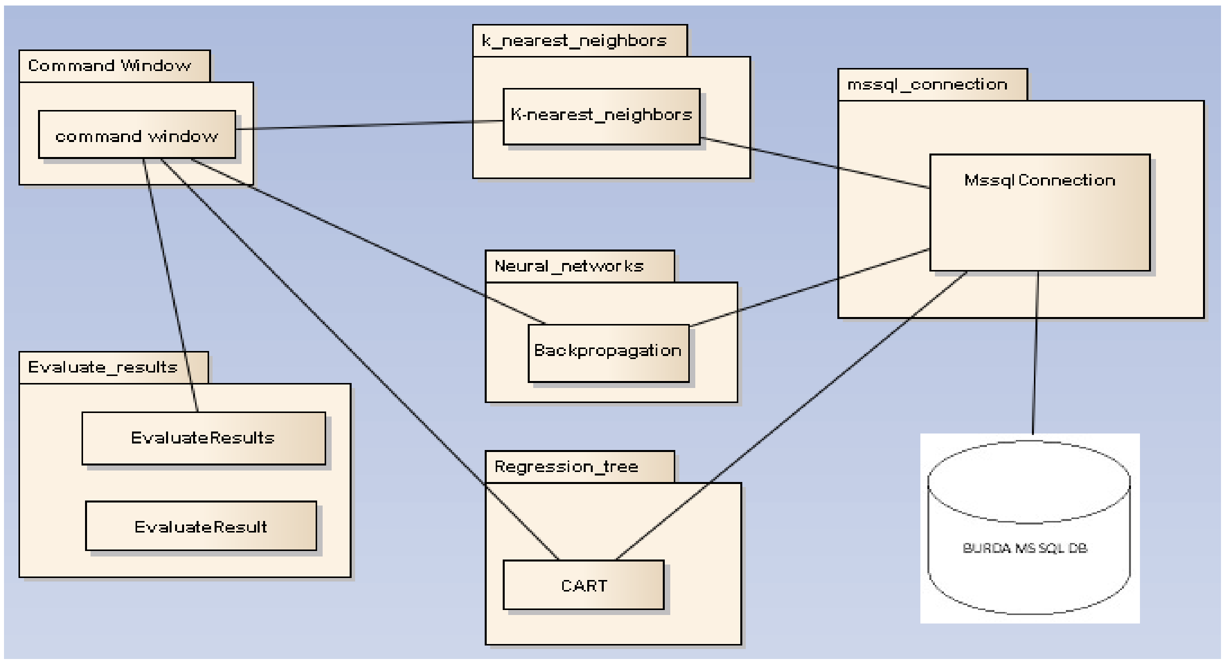


Figure 5.1.1

Burda DB and mssql_connection class is DB layer. K_nearest_neighbors, Backpropagation and CART is Business layer and Evaluate_results is Display layer.

For each technique a static class was created and experiments were called from command window.

5.1.1 Data Preprocessing

Some preprocessing steps were made before first experiment, and they are described in section 2. Preprocessing steps, which took place while making experiments are described here.

Excel

Some basic data enhancements were made in excel. All four excel tables were put to one. Column names were translated to English. And we added *holiday's* columns and *std_magazine_name* column. We did the first experiments over an excel database, but we had problems with performance, so we moved to MS SQL, the same DBMS as in Burda publishing store.

MS SQL Server 2012

In MS SQL Server was made only one preprocessing step, which was creation of *enriched_sold* column.

Matlab

We made most of data preprocessing steps in Matlab. Mssql_connection class only selects data from DB. All preprocessing steps are implemented in business layer. Columns *Sales increase number*, *daynumber*, All Moving Average values, and we created inputs from old values of *enriched_sold*.

5.1.2 Backpropagation

Matlab nntool was used for those experiments. Backpropagation.m class does only data preparation, for nntool. All used datasets and backpropagation networks are in package Backpropagation.m.

5.1.3 K-nearest neighbors

All business logic and data preparation logic is implemented in K_nearest_neighbors.m class. All used datasets are in package K_NN.m.

5.1.4 CART

For this method, data prepared for backpropagation were used, because inputs were very similar. Small data modifications were made in CART.m class.

5.1.5 Evaluate Results

Class EvaluateResults.m creates all statistical results for all techniques. Those results are stored in instance of EvaluateResutl.m class. This class also creates all the figures that are shown in section 4.

5.2 Architecture of Burdasales software

Burdasales is implemented in Matlab. The high level architecture is shown on Figure 5.2.1.

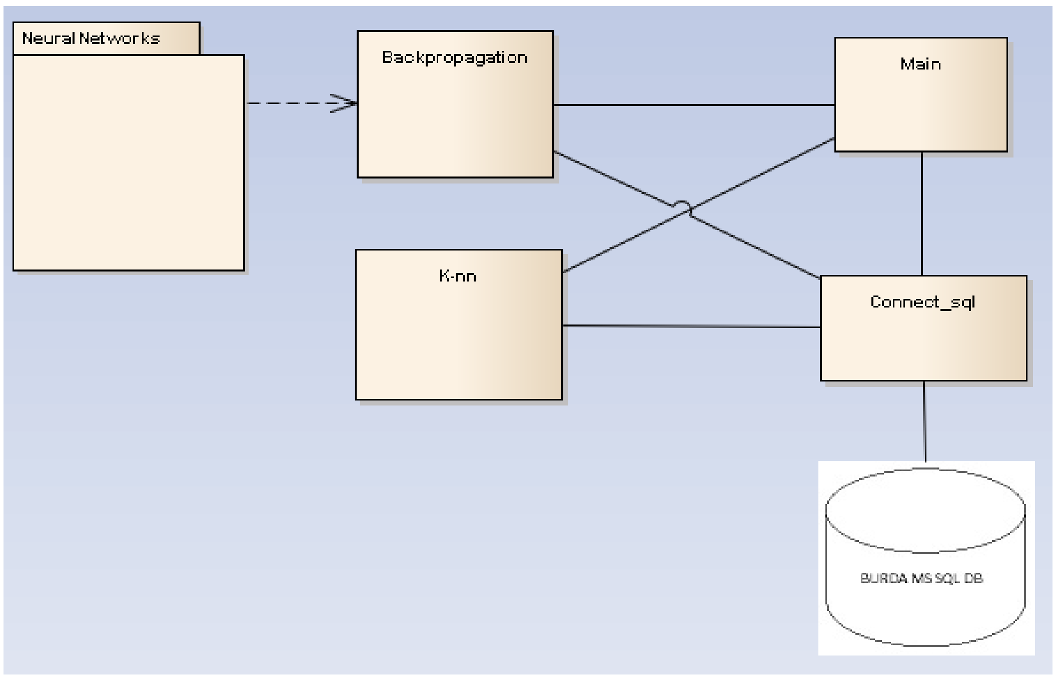
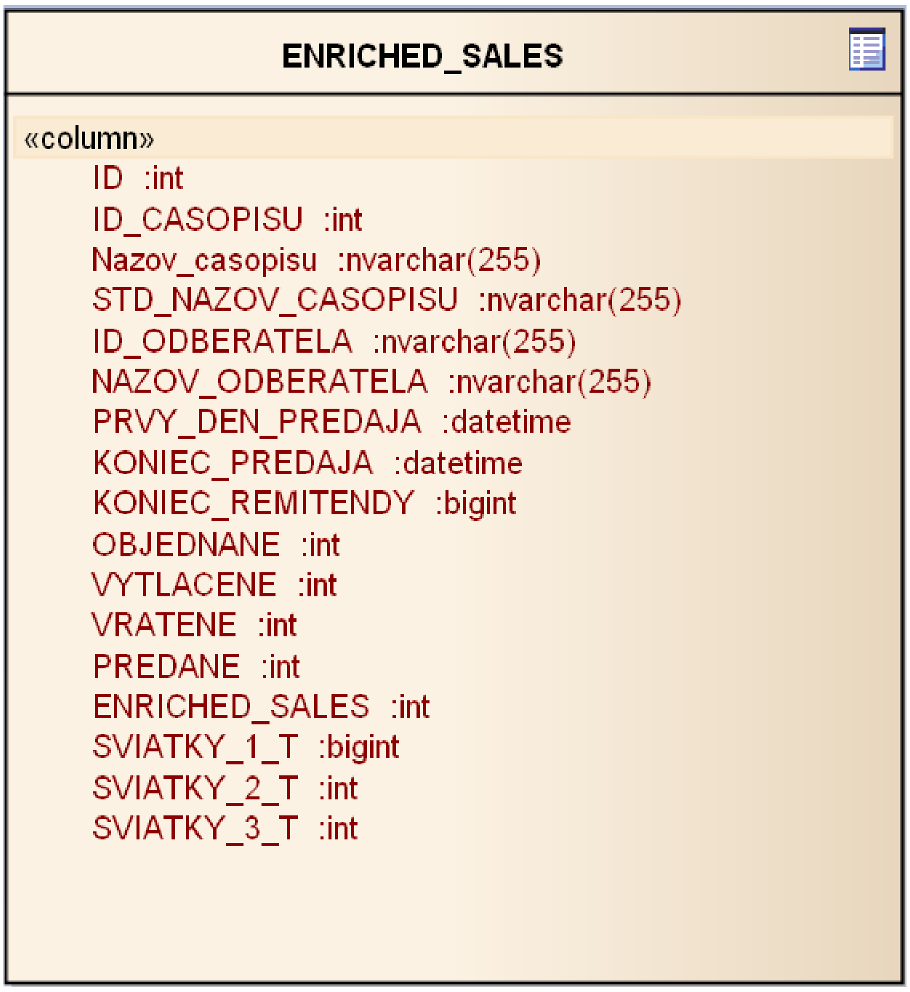


Figure 5.2.1

Burda MSSQL DB

This DB consists of two tables.

- ENRICHED_SALES - stores all data needed for datamining process
- PRODUCTION_PARAMS - stores magazine-consumer pairs, which can be predicted.

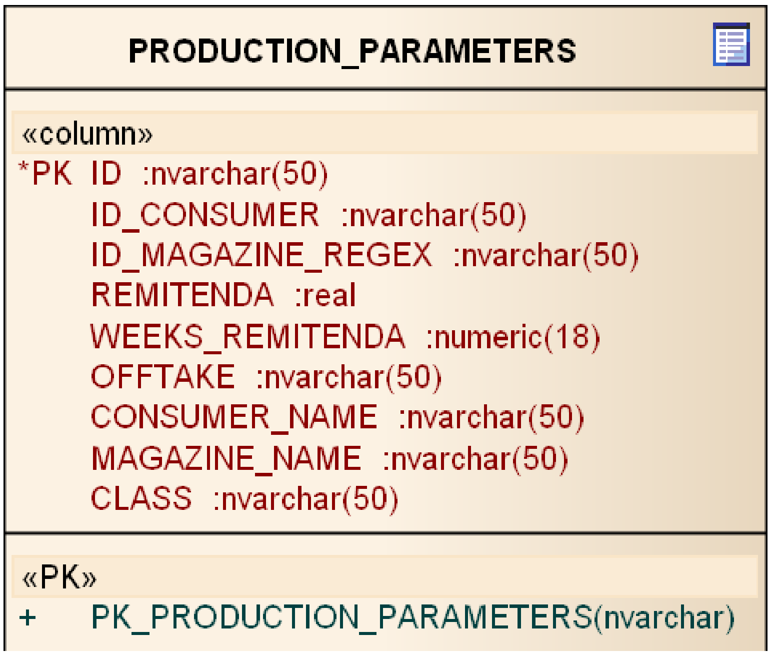


ENRICHED_SALES

«column»

ID :int
ID_CASOPISU :int
Nazov_casopisu :nvarchar(255)
STD_NAZOV_CASOPISU :nvarchar(255)
ID_ODBERATELA :nvarchar(255)
NAZOV_ODBERATELA :nvarchar(255)
PRVY_DEN_PREDAJA :datetime
KONIEC_PREDAJA :datetime
KONIEC_REMITENDY :bigint
OBJEDNANE :int
VYTLACENE :int
VRATENE :int
PREDANE :int
ENRICHED_SALES :int
SVIATKY_1_T :bigint
SVIATKY_2_T :int
SVIATKY_3_T :int

Figure 5.2.2



PRODUCTION_PARAMETERS

«column»
*PK ID :nvarchar(50)
ID_CONSUMER :nvarchar(50)
ID_MAGAZINE_REGEX :nvarchar(50)
REMITENDA :real
WEEKS_REMITENDA :numeric(18)
OFFTAKE :nvarchar(50)
CONSUMER_NAME :nvarchar(50)
MAGAZINE_NAME :nvarchar(50)
CLASS :nvarchar(50)

«PK»
+ PK_PRODUCTION_PARAMETERS(nvarchar)

Figure 5.2.3

Table displayed on Figure 5.2.3 configures the magazine-consumer pairs, which can be calculated in Burdasales.

Class definition

Main.m

This class provides public static methods for calculation predictions. Those methods are:

- `calculateOneByName(consumerName, magazineName)`
- `calculateOneById(magazineConsumerPairId)`
- `calculateAll()`

Backpropagation.m

This class handles the predictions for large and medium magazine-consumer pairs. The backpropagation networks are imported from NeuralNetworks package. Each network has serial number, which is associated to magazine-consumer pair ID in table PRODUCTION_PARAMETERS, Figure 5.2.3.

K-nearest_neighbors.m

This class handles the predictions for small magazine-consumer pairs. It works as the best K-nn technique described in Section 4.2, experiment 3.

Connect_db.m

This class holds all SQL queries, and provides all data from database..

5.3 Deployment of Burdasales

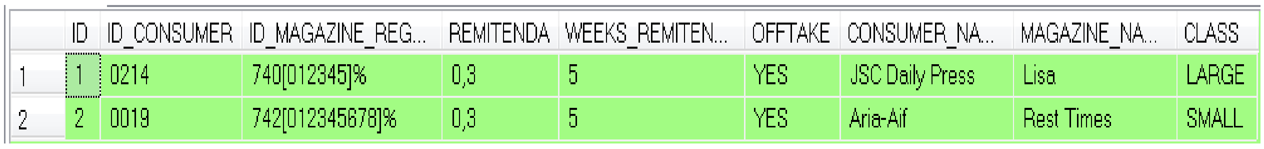
As finishing this work, datamining application was ready for testing mode in Burda publishing house. We were waiting for Burda to create a database needed for Burdasales calculations. Burda has to make a decision, which magazine-consumer pairs will be calculated. When we will have this list, we will train all the backpropagation networks needed for those calculations. At time of finishing this work, only one testing network was attached to the CD.

5.4 Burdasales User Manual

To run the program we need:

- Computer with Windows 7
- MSSQL server 2012 or higher with Burda database (Burda database for testing purposes is attached on CD)
- MSSQL server 2012 Management studio
- Matlab R2012a or higher

Burda publisher house uses MSSQL and there is an agreement, that Burda will deliver and maintain a database for Burdasales. The database has to be the same as described in 5.2, so both tables has to be maintained. The content of *PRODUCTION_PARAMETERS* table has to be known in advance, so we can create a particular backpropagation network, if needed. Some SQL scripts, which transforms Burda data to data we need for our calculations are on attached CD. This database has to be up to date.



	ID	ID_CONSUMER	ID_MAGAZINE_REG...	REMITENDA	WEEKS_REMITEN...	OFFTAKE	CONSUMER_NA...	MAGAZINE_NA...	CLASS
1	1	0214	740[012345]%	0,3	5	YES	JSC Daily Press	Lisa	LARGE
2	2	0019	742[012345678]%	0,3	5	YES	Aria-Aif	Rest Times	SMALL

Figure 5.4.1

Figure 5.4.1 shows table *PRODUCTION_PARAMS*. This table configures the magazine-publisher pairs, which could be predicted. The particular columns are:

- ID
- ID_CONSUMER
- ID_MAGAZINE_REGEXP – regular expression, which selects the magazine id’s, we want to make predictions for.
- REMITENDA – healthy remitenda for the magazine-consumer pair
- WEEKS_REMITENDA – number of weeks between the day, when the magazine is first published and the day, when the number of returned magazines is known
- OFFTAKE – Values are ‘YES’ and ‘NO’ and it configures if particular magazine-consumer pair should be calculated in calculateAll process.
- CONSUMER_NAME
- MAGAZINE_NAME
- CLASS – values are ‘LARGE’, ‘MEDIUM’, ‘SMALL’. Classes are described in Section 4, network 4.

Connection to database

Connection settings are stored in Matlab class Connect_db.m in connectDB method. This has to be set before running the predictions. Example of this setting is:

```
conn =
database('burda','burda','password','com.microsoft.sqlserver.jdbc.SQLServerDriver','j
dbc:sqlserver://localhost:1433;database=burda;integratedSecurity=true;');
```

where first 'burda' is instance and second 'burda' is database name.

Backpropagation networks

If the magazine-consumer pair, added to *PRODUCTION_PARAMETERS* has LARGE or MEDIUM class, the network for its calculation has to be trained using the inputs and configuration described in Section 4, network 4. There is one network trained for magazine-consumer pair with ID 1. The networks has to be named production_network and has to be stored in NeuralNetworks folder in file named network_ID.mat, where ID is ID of magazine-consumer pair in table *PRODUCTION_PARAMETERS*

Running the predictions

The Burdasales folder has to be copied from CD to computer and this folder with all subfolders has to be added to Matlab path. To make predictions, database has to be up to date and number of returned magazines for magazine numbers *WEEKS_REMITENDA* old has to be known.

Predictions can be made two ways. CalculateAll process calculates all magazine-consumer pairs, which are stored in PRODUCTION_PARAMETERS table and have OFFSET set to 'TRUE'. CalculateOne calculate prediction for only one stored magazine-consumer pair. Both processes are called from Matlab Command window.

CalculateOne examples:

```
>> result = Main.calculateOneById(1);
>> result = Main.calculateOneByName('Aria-aif', 'Rest
Times');
```

CalculateAll example:

```
>> results = Main.calculateAll();
```

Results will be stored in global variables result for calculateOne or results for calculateAll. If All results are calculated, they will be stored in matrix of instants of Prediction class. This class contains:

- Magazine-consumer pair ID
- Magazine name
- Consumer name
- Value of prediction

The prediction is calculated without healthy remitenda, so this number of magazines has to be added to result, if it is desired.

6. Conclusion

The main goal of this work was to study regression methods and find a solution for predicting future sales of magazines in Russian magazine publishing house Burda. This company works with more than 200 different magazine types and has more than 200 magazine consumers. When this work was created, all predictions, on which, number of printed magazines depends on, were made by employees in excel tables and were based purely on their experience and guesses. These predictions were commonly very wrong.

Firstly, data form Burda were summarized and analyzed. They were consolidate from four excel documents to one and after that, they were moved to MSSQL database. After that, some basic preprocessing data enhancements were made and some data from external sources were added.

Several techniques were searched and explored. We made experiments using backpropagation neural network technique, k-nearest neighbor technique and CART decision tree technique. The best results came from Backpropagation neural network and k-nearest neighbor. The biggest step forward, to our solution was adding information about sales increases in particular weeks in the year, which improved the performance significantly. We used 3 magazine-consumer datasets to measure the performance, and the best mean error results were 6,95% for dataset with average sales of 42702, technique used was backpropagation neural network, 6,2% for dataset with average sales of 4521, technique used was backpropagation neural network and 13,64% for dataset with average sales of 603, technique used k-nearest neighbors. After these experiments, we decided to create a solution, where on data over 2000 average sales, we used backpropagation technique and on data, where average sales are under 2000 we used k-nearest neighbor technique.

When finishing this work, our application was ready for testing mode in Burda publishing house.

We think that our results are very good, but there is space for improvements. We only made experiments on 3 regression methods, while there are numerous of them, which we did not even look at. Our experiments showed, that in every used technique, number and types of inputs has a big influence on performance. There are many inputs, which could be added to the data, to increase the performance. For example weather information, magazine content or global economical influences. These inputs are very hard to get, and are beyond the scope of this work.

7. Bibliography

1	Lin, Tzu-Chin, Price Prediction for Residential Properties consistent with Sales Comparison Approach
2	D. Coomans; D.L. Massart (1982), Alternative k-nearest neighbour rules in supervised pattern recognition
3	Hean-Lee Poh, Jingtao Yao and Teo Jasic ,Neural Networks for the Analysis and Forecasting of Advertising and Promotion Impact
4	CART® : A Recent Advance in Tree-Structured List Segmentation Methodology by Rosana Thrasher, 1991.
5	Direct Marketing Modeling with CART® and CHAID by Dominique Houghton and Samer Oulabi, 1993.
6	Introduction To Neural Networks – part 1:the neuron, Science of Delphi & Delphi of Science, 30.07.2013
7	Simon & Schuster, Foundations of Physiological Psychology. Needham Heights, Massachusetts, 2012
8	Backpropagation, Wikipedia 30.7.2013
9	Bill Wilson, Neural Networks and Error Backpropagation Learning, 2012
10	Ho, Tin Kam "Random Decision Forest". Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, (1995).
11	Scikit learn, 1.8 Decision Trees, 2.8.2013
12	Amit, Yali; Geman, Donald (1997). "Shape quantization and recognition with randomized trees". Neural Computation, 1997.
13	Aslam, Javed A.; Popa, Raluca A.; and Rivest, Ronald L. (2007); On Estimating the Size and Confidence of a Statistical Audit, Proceedings of the Electronic Voting Technology Workshop (EVT '07), Boston, MA, August 6, 2007.
14	Bootstrap aggregating Wikipedia, 30.7.2013
15	Random forest – Wikipedia, 30.7.2013
16	Jo-Ting Wei , Chih-Chien Weng , Hsin-Hung Wu, A Case Study of Using Classification and Regression Tree and LRFM, 2011

17	Padraic G. Neville, Decision Trees for Predictive Modeling, 1999
18	Iveta Mrazova, Datamining I – Decision Trees, 2012

8. List of Tables

9. Attachments

The accompanying CD contains text part of work, source code of experiments in Matlab, Burdasales source in Matlab and SQL scripts used to transform Burda row data to use for Burdasales application.