Implementation of the Neural Network Algorithm in Advanced Databases

Nedeljko Šikanjić[1], Zoran Ž. Avramović[2], Esad F. Jakupović[2]

[1]PhD Student at Pan-European University APEIRON, Banja Luka

[2]Professor at Pan-European University APEIRON, Banja Luka

Contribution to the state of the art

DOI: **10.7251/JIT1802054S** UDC: **004.738.5:551.588:551.506**

Abstract: The progress of humanity is closely related to the progress of technology. The highlight of the development of technology will occur when the machines are able to do what they learn and know; think and make decisions on their own without human help. In this paper we will try to analyze how neural networks work and how they will further develop in terms of application in advanced databases. We will also explore how one of the major IT companies is developing and continuing to use neural networks.

Keywords: Neural network, Advanced database, SQL.

Introduction

People owe their development to their intelligence and by coping the world around them by their implementation of technology. One of the more advanced attempts is to imitate human behavior and its implementation in technologies is an example of a neural network. The neural network in technology is based on the imitation of the human brain and how it functions. Although it has made a lot of progress, it is not even close to the functioning of the human brain, which remains the biggest challenge, but its behavior is improving over time.

Neuron Network

The neural network was created in 1943 when Waren McCulloh and Walter Pitts created a model for neural networks based on a mathematical and algorithmic model. In their model, Alan Turing later proposed a model which represents a functional construction according to the exchange of information flowing to the input, condition and output.

This model also works with the human brain. [1] The estimation is that the human brain has about 100 billion neurons. Neuron consists of the body (soma), dendrita (inlays) and axons (exits).

The artificial neuron is created on the same principle. The artificial neuron functions in the way that at the input it receives the value that it sums up and then forwards it to the output. Here we can find a layer system - an input layer, where we can have multiple input points, hidden layers (which represent the body of a neuron) and an output layer, where we can have more output points. The relationships between neurons are represented by numbers. These numbers are defined as the weight. The weight principle is set so that if the weight is greater then the effect of neurons on the other is also higher.

The way a neural network works is that it is "trained" for the first time, for example to learn how information should be processed, and the second time it is "tested" or performs certain tasks. This method is called propagation in advance.

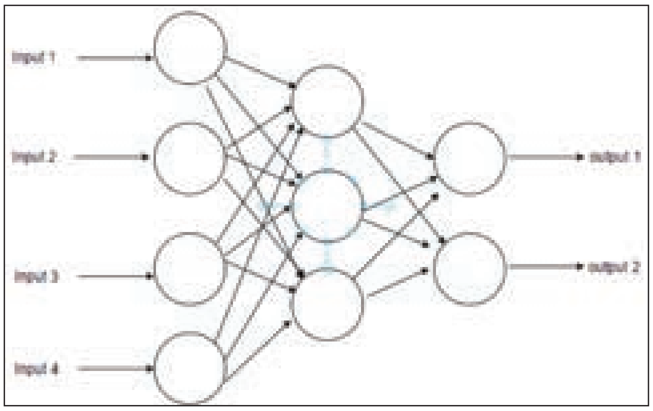


Image 1: *Neural network with hidden layer*

The way the neural network compares the output results through several iterations is called propagation backward. In this case, the information is returned to the neuron input. This is a system that people often use in attempts where, after a certain attempt, a person accurately or approximately accurately performs a certain operation.

Implementation of Algorithms of Neuron Networks in Basics - Sql Server

In this paper, we will work with an example of neural networks.What is amazing when it comes to machine learning is that it manages to transform us into adaptive thinkers [2]. We have created an example on a test dataset containing information about current trial subscription clients with their various attributes. Based on this set of data, we will find the probability of whether the possible client can become a paying customer through the algorithm. We will use a data set for displaying the neural network as it is implemented in Microsoft SQL Server Machine Learning Services along with the programming language R.

Microsoft made an important step towards statistical analysis as well as with data science [3]. Microsoft has made it possible to benefit from processing the data from where data are already located, in the database itself.

Some of the advantages of this approach are the following:

- The path to the data and its processing is reduced, thus improving security and facilitating access
- It is possible to obtain information in the short term
- The solution is scalable by being enabled as open source, so it can be applied in all versions that are offered
- New knowledge is being created that is already in place without having impact on the processing of existing data

Microsoft ML (Machine Learning) is a service responsible for transforming data or models that the R language and the Python language are processing [4]. The primary task is to enable data formatting. The functionalities provided by this service are the parallel execution, manipulation of data on disk and in memory as processing of huge amount of data.

The tasks or algorithms performed by the machine learning service are the following:

- The binary classification algorithm that teaches where one of the two classes of data instances belongs.

For example: It uses output of single two values such as 0 or 1 to detect true or false when search is given

$$R(\boldsymbol{x}) = P(+1|\boldsymbol{x}) - P(-1|\boldsymbol{x})$$

Formula 1: *Binary classifier [5]*

In the case of formula presented we use values of -1 and +1 where binary classifier for calculating probability is presented

- The multiclass classification algorithm learns to predict the categories to which the data belongs. The data are from 0 to k-1 where k represents the number of classes.

$$R_i(\boldsymbol{x}) = \frac{\sum_j a_{ij} P(j|\boldsymbol{x})}{\sum_j |a_{ij}| P(j|\boldsymbol{x})}$$

Formula 2: *Multi-class classification*

When we need more then single value as prediction we could use multi-class classification. Two sets are presented in this formula where *i* represents a row of matrix with partitioning of *j* row.

- A regression learning algorithm that teaches to predict the value of the dependent variable from a set of independent variables. The result that is given is a function used to display the value of a new data instance whose variables are not known.

- The anomaly detection algorithm that serves to identify which data or class of data does not belong to a given pattern. The sphere of application of these algorithms includes detection of fraud via credit cards, forecasting bankruptcy, estimating the value of mortgages or houses, checking the email whether it belongs to spam, etc.

Implementation use case

R language in combination with SQL Server objects such as Stored procedures is a very efficient and fast tool for work. By using this method, we shorten the way to the data, we do not have to think where to publish R scripts (for example, the web service) and we do not have to worry more about the performance since SQL Server will do it for us.

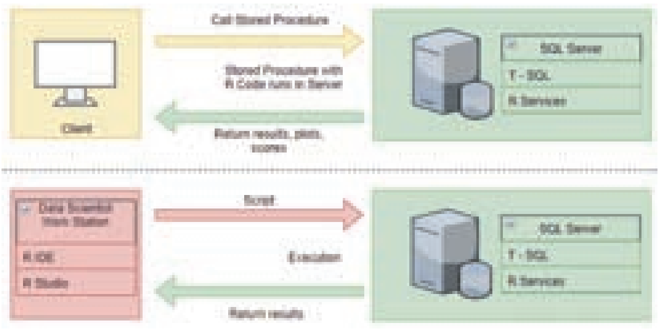


Image 2: *Diagram of the mode of operation and communication with SQL Server (R services)*

There are two options that we can use. One is to use the R Studio, which is very popular for the work of R language. Using libraries in R Studio, we can connect to SQL Server and execute the necessary commands and display the results in the R Studio. Another way is to use the SQL Server Management Studio environment, using functions and procedures, to process data using the R language in the form of enabled scripts, and to view data using various analysis and reporting tools that come with SQL Server.

In this example, we will use the Decision tree algorithm.

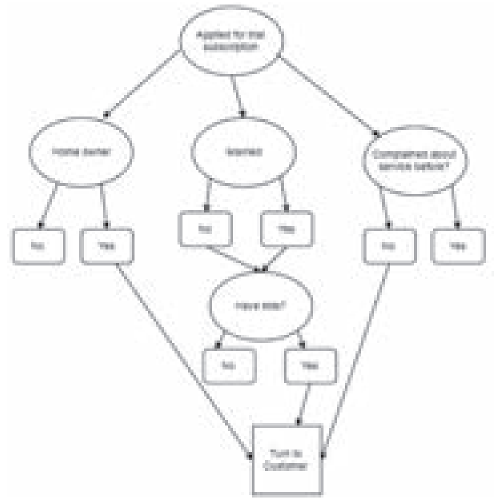


Image 3: *Decision tree diagram (partial) based on sample data*

As we can see on a decision tree diagram, we want to know, based on sample data, if this potential subscriber is a potential paying customer. With different sets of attributes we can determine if this is a valid model to predict an outcome with some percentage of certainty.

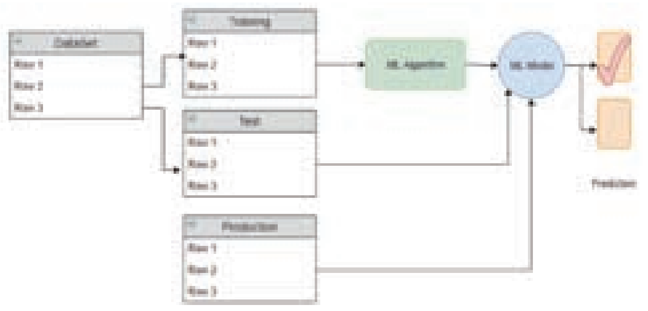
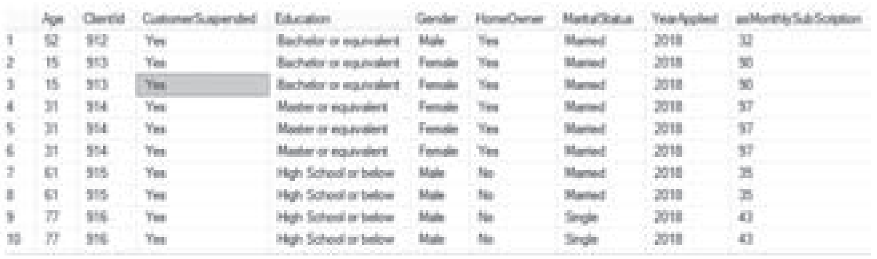


Image 4: *Training, testing, and prediction implementation diagram in SQL*

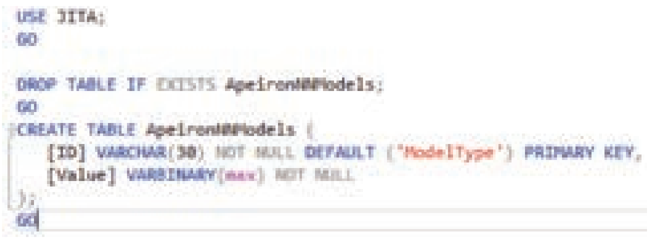
We will use a set of data to create a neural network. We will split this set of data into the training set and test set. The training set will use our algorithm to create a model and then use this model to test. Based on the testing, we can adjust our model so that we get the most accurate data. In the end, when we are satisfied with our model, then we can release it into production or production data to send it to us.

We will use a data model to predict whether a potential client will convert to a customer or not, based on a data set of some 25,000 data.



	Age	ClientId	CustomerSuspended	Education	Gender	HomeOwner	MaritalStatus	YearApplied	avMonthlySubScription
1	52	912	Yes	Bachelor or equivalent	Male	Yes	Married	2018	32
2	15	913	Yes	Bachelor or equivalent	Female	Yes	Married	2018	50
3	15	913	Yes	Bachelor or equivalent	Female	Yes	Married	2018	50
4	31	914	Yes	Master or equivalent	Female	Yes	Married	2018	57
5	31	914	Yes	Master or equivalent	Female	Yes	Married	2018	57
6	31	914	Yes	Master or equivalent	Female	Yes	Married	2018	57
7	61	915	Yes	High School or below	Male	No	Married	2018	35
8	61	915	Yes	High School or below	Male	No	Married	2018	35
9	77	916	Yes	High School or below	Male	No	Single	2018	43
10	77	916	Yes	High School or below	Male	No	Single	2018	43

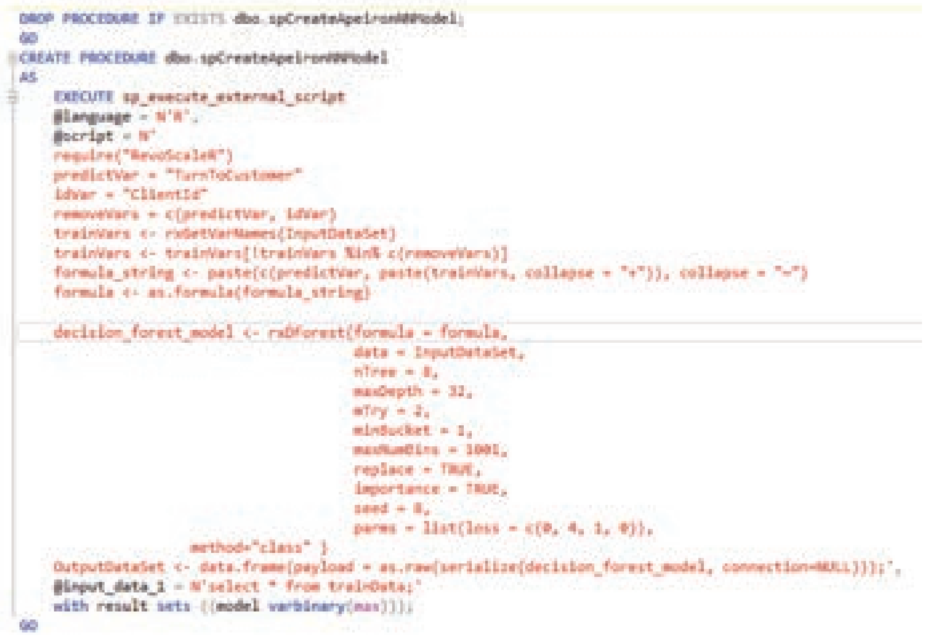
In the data model, we keep data with all the necessary attributes on the basis of which we can come up with prediction of which attributes are crucial for converting the client into a permanent customer. Some of the attributes that exist in the set of data are the level of education, gender, whether the client is the owner of the house or real estate, marital status, etc.



```
USE JITA;
GO

DROP TABLE IF EXISTS ApeironNNModels;
GO
CREATE TABLE ApeironNNModels (
    [ID] VARCHAR(30) NOT NULL DEFAULT ('ModelType') PRIMARY KEY,
    [Value] VARBINARY(max) NOT NULL
);
GO
```

In order to be able to create a model in machine learning, we need to have a data set for training and a data set for testing.

The data were divided into two sets of data, a training data set and a test data set to a ratio of 70% in training and 30% in the test set of data [6].



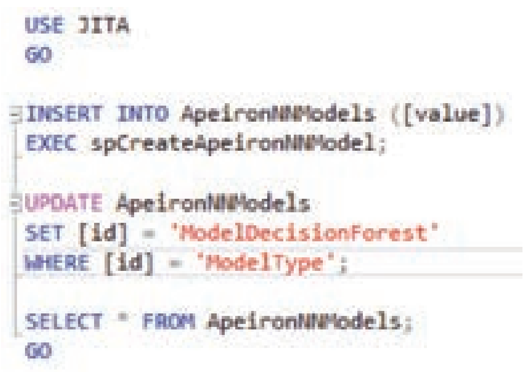
```
DROP PROCEDURE IF EXISTS dbo.spCreateApeironNNModel;
GO
CREATE PROCEDURE dbo.spCreateApeironNNModel
AS
    EXECUTE sp_execute_external_script
    @language = N'R',
    @script = N'
    require("RevoScaleR")
    predictVar = "TurnToCustomer"
    idVar = "ClientId"
    removeVars = c(predictVar, idVar)
    trainVars <- rxGetVarNames(InputDataSet)
    trainVars <- trainVars[!trainVars %in% c(removeVars)]
    formula_string <- paste(c(predictVar, paste(trainVars, collapse = "+")), collapse = "~")
    formula <- as.formula(formula_string)

    decision_forest_model <- rxDForest(formula = formula,
                                       data = InputDataSet,
                                       nTree = 8,
                                       maxDepth = 32,
                                       mTry = 2,
                                       minBucket = 1,
                                       maxNumBins = 1001,
                                       replace = TRUE,
                                       importance = TRUE,
                                       seed = 8,
                                       parms = list(loss = c(0, 4, 1, 0)),
                       method="class" )
    OutputDataSet <- data.frame(payload = as.raw(serialize(decision_forest_model, connection=NULL)));',
    @input_data_1 = N'select * from trainData;'
    with result sets ((model varbinary(max)));
GO
```

In this stored procedure, we have an implemented part of the R language that creates the model over the variables we passed to them. This model is used by the decision tree, based on which we can get predictions for the data we send to them.

Regarding the decision tree, there is a function in the R language that we use as part of the SQL store procedure is rxDForest. We send the following parameters to it:

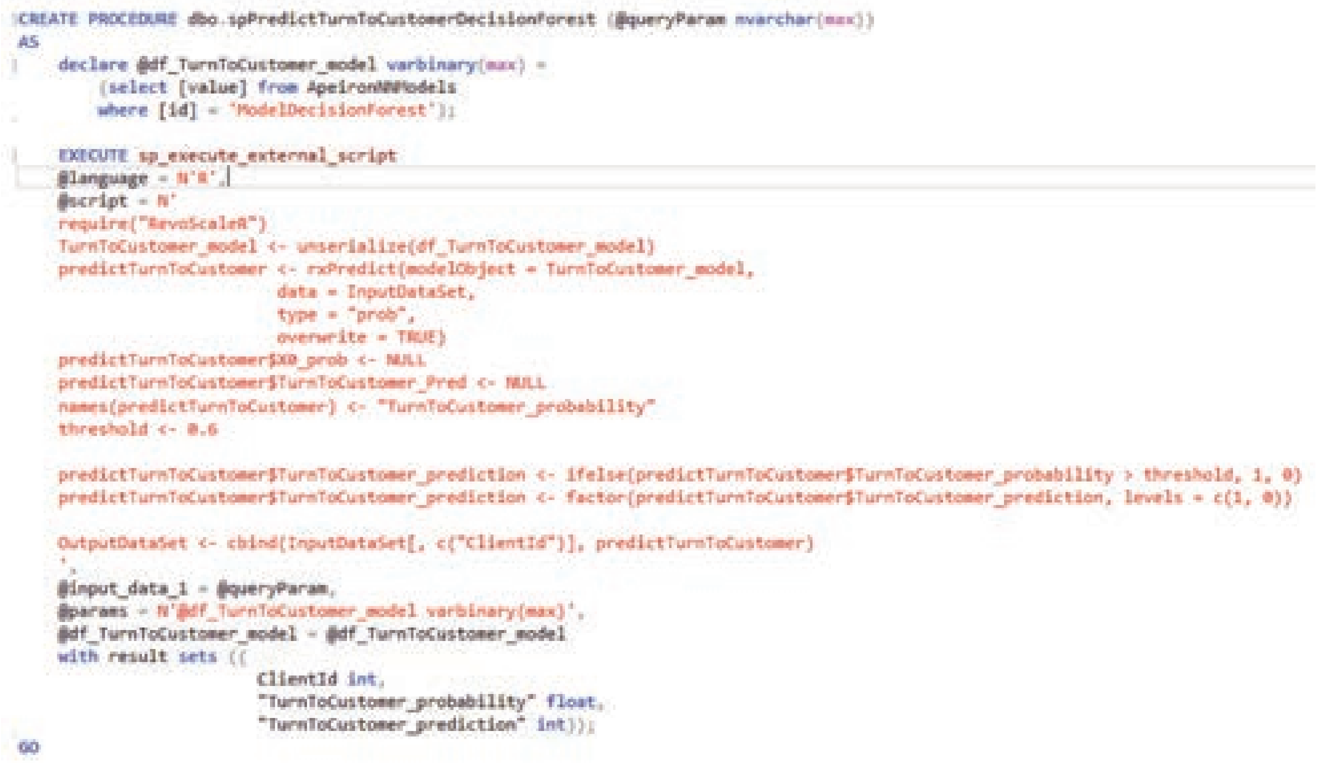
- Formula: the formula we pass to the columns that are key to the prediction model
- Data: data to be processed
- nTree: number of trees to which the model will grow

- maxDepth: maximum depth of trees used for calculating
- mTry: number of variables that are candidates for bending trees
- minBucket: number of observations in the node
- maxNumBins: control the maximum number of memory stores used for each variable
- replace: whether the observed variables will be replaced
- importance: a logical value that serves to evaluate the importance of the predictor
- seed: number used to initialize random numbers
- parms: additional parameters for bridging



```
USE JITA
GO

INSERT INTO ApeironNNModels ([value])
EXEC spCreateApeironNNModel;

UPDATE ApeironNNModels
SET [id] = 'ModelDecisionForest'
WHERE [id] = 'ModelType';

SELECT * FROM ApeironNNModels;
GO
```

The model is generated as binary data or output from stored procedure, which we will save in the table that will be used for sending parameters for test and production data.



```
CREATE PROCEDURE dbo.spPredictTurnToCustomerDecisionForest (@queryParam nvarchar(max))
AS
    declare @df_TurnToCustomer_model varbinary(max) =
        (select [value] from ApeironNNModels
        where [id] = 'ModelDecisionForest');

    EXECUTE sp_execute_external_script
    @language = N'R',
    @script = N'
    require("RevoScaleR")
    TurnToCustomer_model <- unserialize(df_TurnToCustomer_model)
    predictTurnToCustomer <- rxPredict(modelObject = TurnToCustomer_model,
                               data = InputDataSet,
                               type = "prob",
                               overwrite = TRUE)
    predictTurnToCustomer$X0_prob <- NULL
    predictTurnToCustomer$TurnToCustomer_Pred <- NULL
    names(predictTurnToCustomer) <- "TurnToCustomer_probability"
    threshold <- 0.6

    predictTurnToCustomer$TurnToCustomer_prediction <- ifelse(predictTurnToCustomer$TurnToCustomer_probability > threshold, 1, 0)
    predictTurnToCustomer$TurnToCustomer_prediction <- factor(predictTurnToCustomer$TurnToCustomer_prediction, levels = c(1, 0))

    OutputDataSet <- cbind(InputDataSet[, c("ClientId")], predictTurnToCustomer)
    ',
    @input_data_1 = @queryParam,
    @params = N'@df_TurnToCustomer_model varbinary(max)',
    @df_TurnToCustomer_model = @df_TurnToCustomer_model
    with result sets ((
                        ClientId int,
                        "TurnToCustomer_probability" float,
                        "TurnToCustomer_prediction" int));
GO
```

This stored procedure is used to predict based on the model we created.

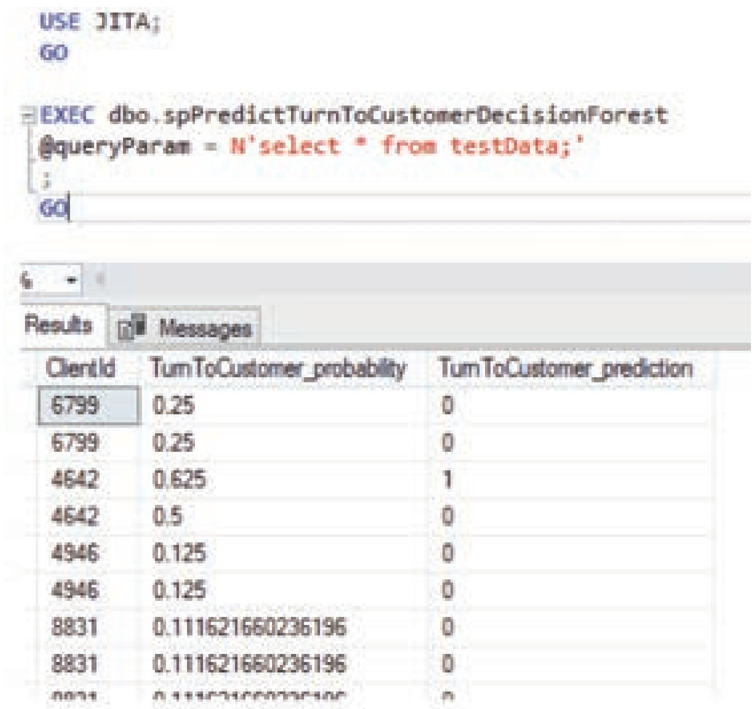
The function we use in this stored procedure is rxPredict which serves to predict the results using the model we have already created.

The rxPredict functions are forwarded to the following parameters:

- Model: the model we previously created, in this case rxDForest

- Data: the data we use as a source for the prediction
- Type: prediction type, in our case we set the value "probe"
- Overwrite: a logical value that serves to overcome the output value

An example of passing test data is shown below in the SQL query.



```
USE JITA;
GO

EXEC dbo.spPredictTurnToCustomerDecisionForest
@queryParam = N'select * from testData;'
;
GO
```

ClientId	TurnToCustomer_probability	TurnToCustomer_prediction
6799	0.25	0
6799	0.25	0
4642	0.625	1
4642	0.5	0
4946	0.125	0
4946	0.125	0
8831	0.111621660236196	0
8831	0.111621660236196	0

Based on the results we have received, we can further link this stored procedure to the reports to fine-tune the visual representation of the data, or we can further process the results both in the SQL environment and in the client application.

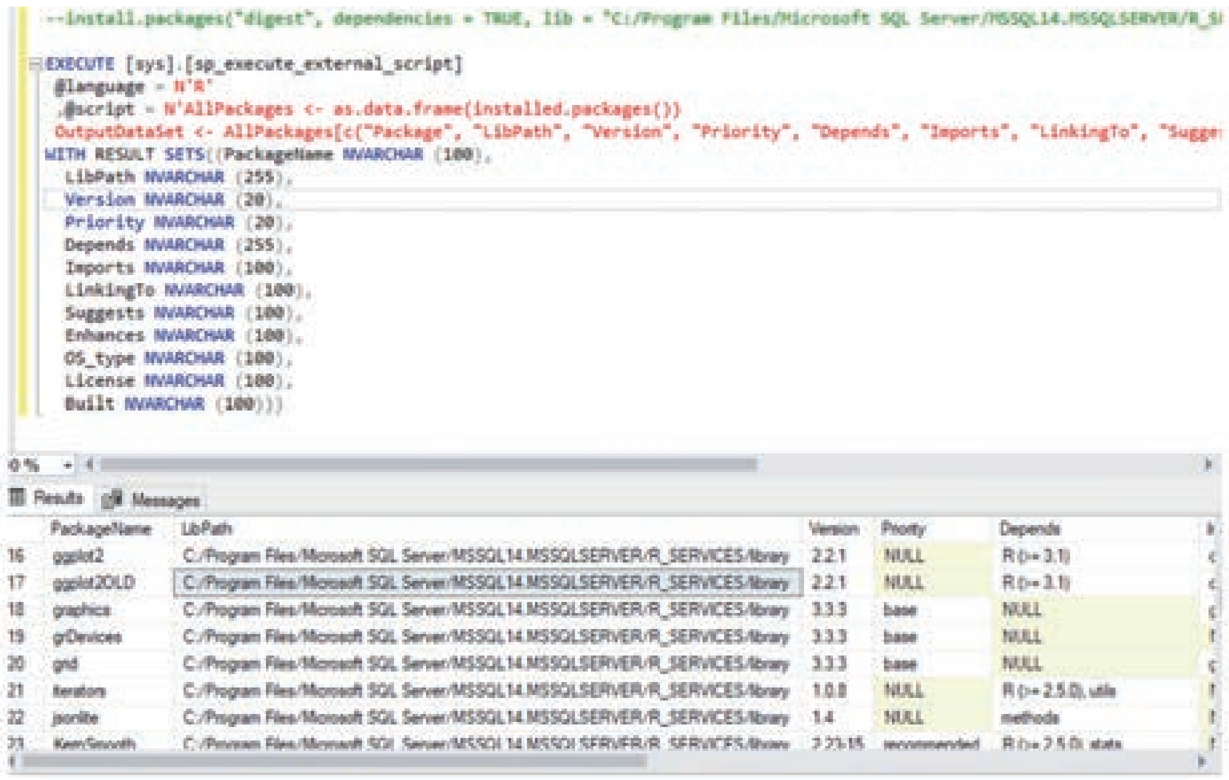
Generating visual representation of data

In order to create a visual representation of the results obtained, we can use the option within the SQL query.

It is important to note a hybrid approach using the R programming language within SQL Server where there are enough specific instructions to know.

For example, in order to generate plot diagrams in the R programming language, it is necessary to install certain libraries. For the plot diagram, we will use the ggplot2 library. The instruction for installing this library is as follows:
„install.packages("digest", dependencies = TRUE, lib = "PATH/Microsoft SQL Server/MSSQL14. MSSQLSERVER/R_SERVICES/library")“

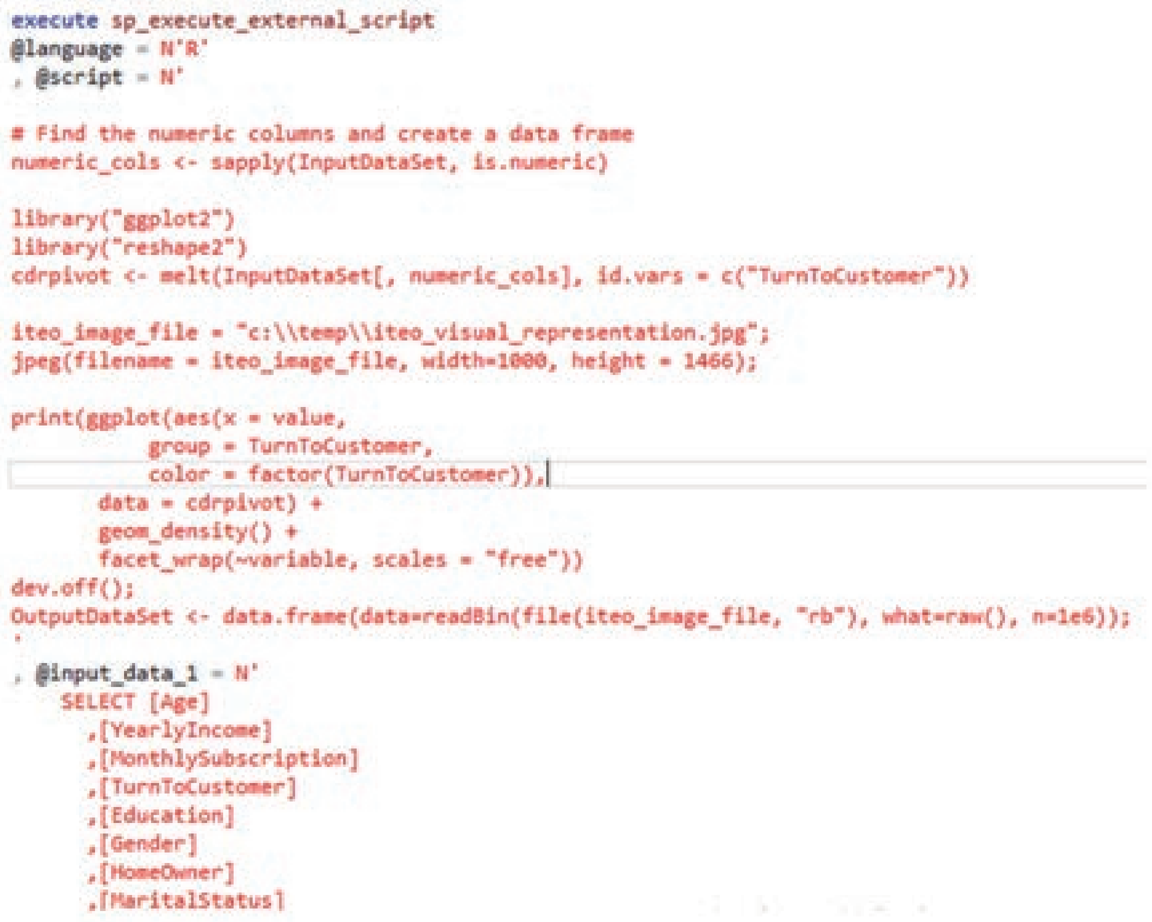
In order to know which libraries we can use in the SQL environment, it is enough to execute a query to list all the libraries available in our SQL environment.



```
--install.packages("digest", dependencies = TRUE, lib = "C:/Program Files/Microsoft SQL Server/MSSQL14.MSSQLSERVER/R_S

EXECUTE [sys].[sp_execute_external_script]
 @language = N'R'
 ,@script = N'AllPackages <- as.data.frame(installed.packages())
 OutputDataSet <- AllPackages[c("Package", "LibPath", "Version", "Priority", "Depends", "Imports", "LinkingTo", "Sugge
WITH RESULT SETS((PackageName NVARCHAR (100),
  LibPath NVARCHAR (255),
  Version NVARCHAR (20),
  Priority NVARCHAR (20),
  Depends NVARCHAR (255),
  Imports NVARCHAR (100),
  LinkingTo NVARCHAR (100),
  Suggests NVARCHAR (100),
  Enhances NVARCHAR (100),
  OS_type NVARCHAR (100),
  License NVARCHAR (100),
  Built NVARCHAR (100)))
```

```
0 %
Results  Messages
    PackageName  LibPath                                                              Version  Priority     Depends
16  ggplot2      C:/Program Files/Microsoft SQL Server/MSSQL14.MSSQLSERVER/R_SERVICES/library  2.2.1    NULL         R (>= 3.1)
17  ggplot2OLD   C:/Program Files/Microsoft SQL Server/MSSQL14.MSSQLSERVER/R_SERVICES/library  2.2.1    NULL         R (>= 3.1)
18  graphics     C:/Program Files/Microsoft SQL Server/MSSQL14.MSSQLSERVER/R_SERVICES/library  3.3.3    base         NULL
19  grDevices    C:/Program Files/Microsoft SQL Server/MSSQL14.MSSQLSERVER/R_SERVICES/library  3.3.3    base         NULL
20  grid         C:/Program Files/Microsoft SQL Server/MSSQL14.MSSQLSERVER/R_SERVICES/library  3.3.3    base         NULL
21  iterators    C:/Program Files/Microsoft SQL Server/MSSQL14.MSSQLSERVER/R_SERVICES/library  1.0.8    NULL         R (>= 2.5.0), utils
22  jsonlite     C:/Program Files/Microsoft SQL Server/MSSQL14.MSSQLSERVER/R_SERVICES/library  1.4      NULL         methods
23  KernSmooth   C:/Program Files/Microsoft SQL Server/MSSQL14.MSSQLSERVER/R_SERVICES/library  2.23-15  recommended  R (>= 2.5.0), stats
```

There are several options to generate a visual representation of the results obtained. One of them is through a SQL query.



```
execute sp_execute_external_script
@language = N'R'
, @script = N'

# Find the numeric columns and create a data frame
numeric_cols <- sapply(InputDataSet, is.numeric)

library("ggplot2")
library("reshape2")
cdrpivot <- melt(InputDataSet[, numeric_cols], id.vars = c("TurnToCustomer"))

iteo_image_file = "c:\\temp\\iteo_visual_representation.jpg";
jpeg(filename = iteo_image_file, width=1000, height = 1466);

print(ggplot(aes(x = value,
           group = TurnToCustomer,
           color = factor(TurnToCustomer)),
       data = cdrpivot) +
       geom_density() +
       facet_wrap(~variable, scales = "free"))
dev.off();
OutputDataSet <- data.frame(data=readBin(file(iteo_image_file, "rb"), what=raw(), n=1e6));
'
, @input_data_1 = N'
    SELECT [Age]
      ,[YearlyIncome]
      ,[MonthlySubscription]
      ,[TurnToCustomer]
      ,[Education]
      ,[Gender]
      ,[HomeOwner]
      ,[MaritalStatus]
```

In this SQL query, we use the R language programming commands to generate an image using the plot function to generate a diagram. Data is transmitted using the parameters used in the generic plot diagram. We can save this created diagram on a disk or in the database.

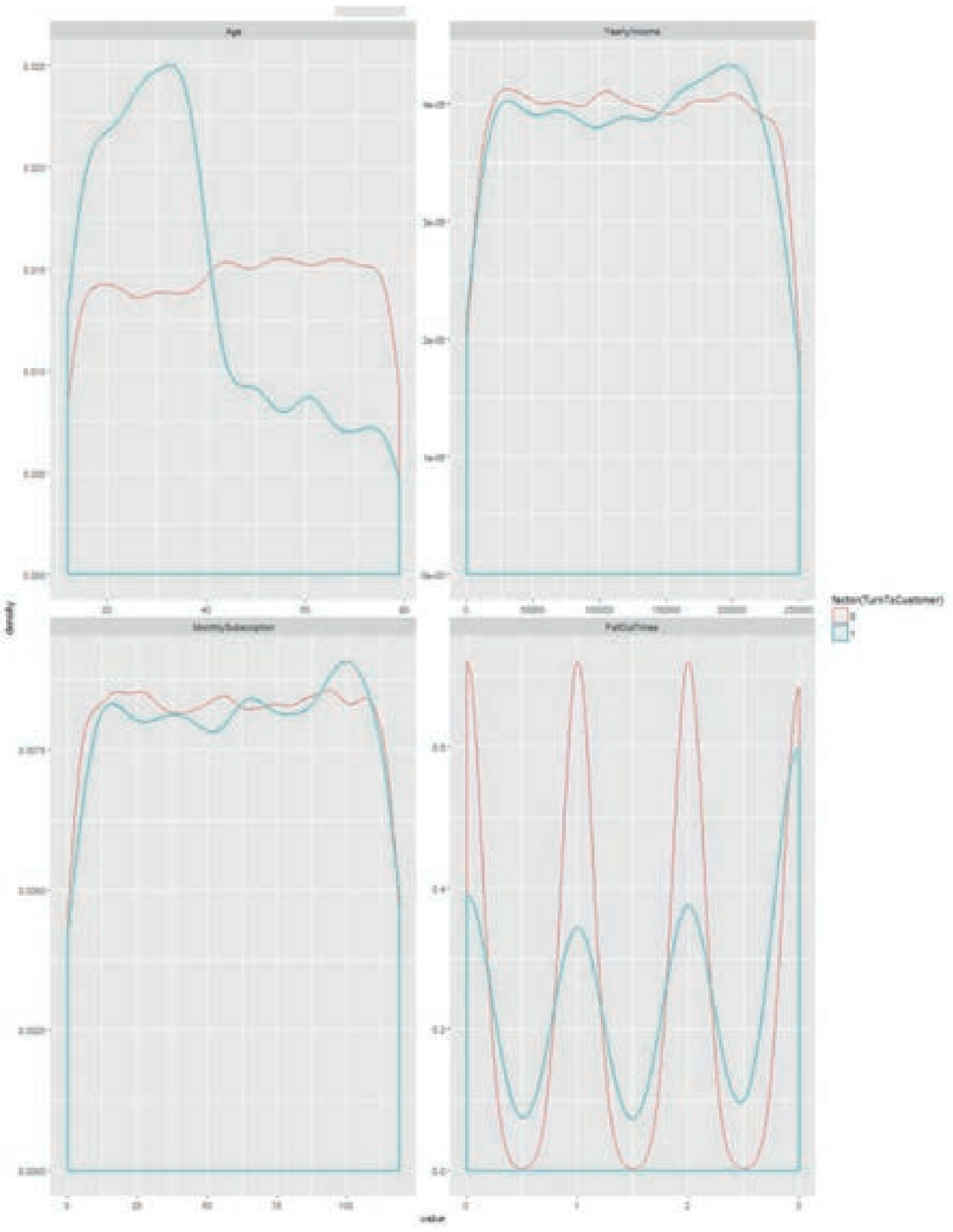
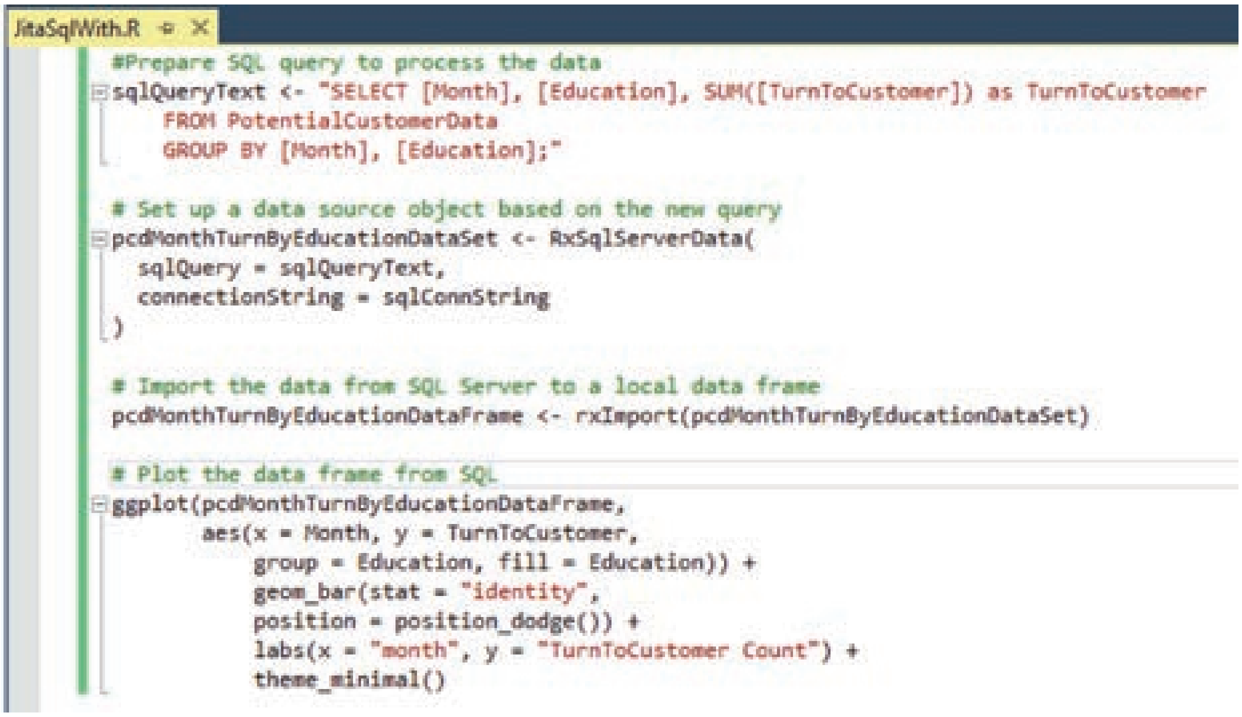


Image 3.2: *Generated plot diagram using SQL query*

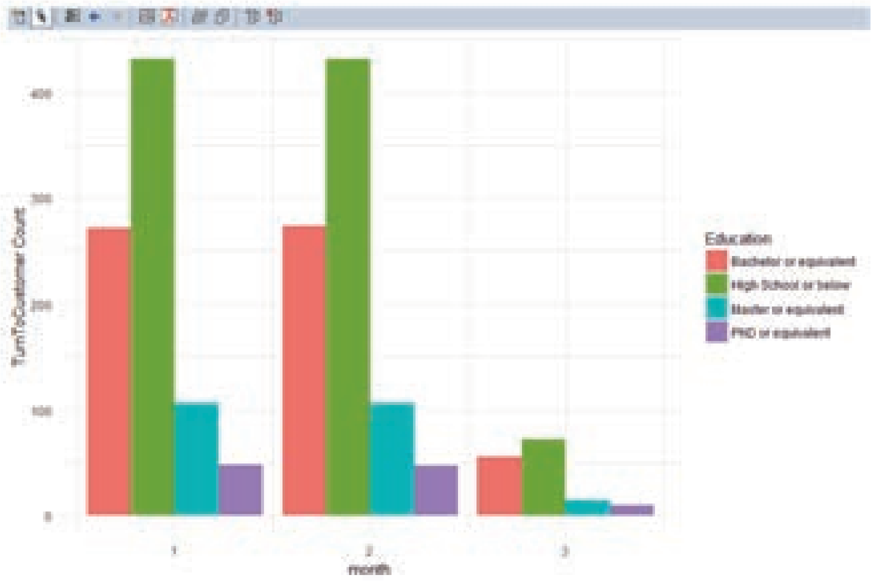
Another option is using Visual Studio. In addition to the existing project types, Visual Studio also has a project for the R programming language. Through the R tool, we can perform various commands, including data visualization.



```
JitaSqlWith.R

#Prepare SQL query to process the data
sqlQueryText <- "SELECT [Month], [Education], SUM([TurnToCustomer]) as TurnToCustomer
    FROM PotentialCustomerData
    GROUP BY [Month], [Education];"

# Set up a data source object based on the new query
pcdMonthTurnByEducationDataSet <- RxSqlServerData(
  sqlQuery = sqlQueryText,
  connectionString = sqlConnString
)

# Import the data from SQL Server to a local data frame
pcdMonthTurnByEducationDataFrame <- rxImport(pcdMonthTurnByEducationDataSet)

# Plot the data frame from SQL
ggplot(pcdMonthTurnByEducationDataFrame,
       aes(x = Month, y = TurnToCustomer,
           group = Education, fill = Education)) +
           geom_bar(stat = "identity",
           position = position_dodge()) +
           labs(x = "month", y = "TurnToCustomer Count") +
           theme_minimal()
```

Data is being prepared through the functions RxSqlServerData that enable loading the data into a data-set object, which then imports the local data frame.

After that we use the ggplot function by passing the data set object, defining what we are showing on the X and Y axes, along with other parameters such as labels and setting the theme of the diagram.



The diagram with the data that is generated can be further exported to other formats such as pdf, images or metadata [7].

Conclusion

In this paper we have shown how the neural network functions in advanced databases, which are the advantages of using neural networks in advanced databases and what is its further development. It is also important to know that the largest IT companies such as Microsoft have decided to implement their focus on the application of a neural network in a similar way, that large processing of data which the client stations find considerably difficult to process due to resources can move within the database and can thus apply security of access to data and use resources that their application algorithms or engine engine possess. For us as humans it is important to continue to strive to integrating different systems in order to unlock the knowledge that we could not imagine before.

References

[1] N. J. Nilsson, Introduction to Machine Learning, Robotics Laboratory, Department of Computer Science, Stanford University, 1998.

[2] Denis Rothman, Artificial Intelligence By Example: Develop machine intelligence from scratch using real artificial intelligence use cases, ISBN-13: 978-1788990547, May 2018

[3] Tomaz Kastrun, Julie Koesmarno, SQL Server 2017 Machine Learning Services with R: Data exploration, modeling, and advanced analytics, ISBN-13: 978-1787283572, February 2018

[4] https://blog.statsbot.co/bayesian-learning-for-statistical-classification-f2362d620428 accesed on 10.11.2018

[5] Raghav Bali, R Machine Learning By Example, ISBN-13: 978-1784390846, March 31, 2016

[6] https://docs.microsoft.com/en-us/sql/advanced-analytics/?view=sql-server-2017accessedon 15.11.2018

Submitted: November 21, 2018
Accepted: December 24, 2018

About the authors

Nedeljko Šikanjić holds a Magister degree in Informatics and Computer Science and has worked for more than 15 years as a Software and Database Architect/Engineer. His main fields of studies are in the area of advanced Databases and Software Architectures. He has been a holder of active Microsoft Certified Trainer Certificate since 2012 and has been teaching courses on various topics in Information Technologies.

Zoran Ž. Avramović was born in Serbia (Yugoslavia) on September 10th, 1953. He graduated from the Faculty of Electrical Engineering, University of Belgrade. At this Faculty he received a Master's degree, and then a PhD in technical sciences. He is:

- Academician of the Russian Academy of Transport (RTA, St. Petersburg, Russia, since 1995),
- Academician of the Russian Academy of Natural Sciences (RANS, Moscow, Russia, since 2001),
- Academician of the Yugoslav Academy of Engineering (YAE, Belgrade, Serbia, since 2004) (today: Engineering Academy of Serbia, EAS)
- Academician of the Academy of Electrotechnical Sciences of the Russian Federation (AES of the Russian Federation, Moscow, Russia, since 2007)
- Scientific Secretary of the Electrical Engineering Department of the Engineering Academy of Serbia.

Esad Jakupović, was born on 21 September 1950 in Ćeli, municipality of Prijedor, where he finished elementary and high school. He graduated from the Faculty of Industrial Pedagogy - the direction of physics in Rijeka in 1975. Postgraduate studies completed at the Faculty of Agricultural Sciences of the University of Zagreb in 1984 in the field of Mechanization of Agriculture. He defended his doctoral dissertation at the Faculty of Mechanical Engineering of the University of Ljubljana in 1991.
Since 2005 he has been employed as a regular professor at the Pan-European University "APEIRON" Banja Luka. At the same University, he served as Vice-Rector for Teaching, and the Vice President for Postgraduate and PhD Studies in the period 2005-2014.
He worked as the rector of the Pan-European University "APEIRON" from 2014 to 2018.

For citation

Nedeljko Šikanjić, Zoran Ž. Avramović, Esad F. Jakupović, Implementation of the Neural Network Algorithm in Advanced Databases, *JITA – Journal of Information Technology and Applications*, PanEuropien University APEIRON, Banja Luka, Republika Srpska, Bosna i Hercegovina, JITA 8(2018) 2:54-63, (UDC: 004.738.5:551.588:551.506), (DOI: 10.7251/JIT1802054S), Volume 8, Number 2, Banja Luka, december 2018 (45-96), ISSN 2232-9625 (print), ISSN 2233-0194 (online), UDC 004