Modeling Guides for Neural Network

Neural network is a very powerful predictive modeling method. This paper provides guides for neural network modeling.

1. Data Preparation

Historical data is prepared in a relational database system as a ***database table***. A lightweight database system, such as MySQL, PostgreSQL, MS SQL Server, MS Access, etc., is recommended. This is done by extracting and copying data from operational database systems. Historical data must be cleaned before analysis (***data cleaning***). Erroneous and synonymous data should be corrected. Outliers should be replaced with normal values.

Historical data can be prepared as a ***training dataset*** and multiple ***test datasets***. If data is not large, the following method can be used. Entire historical data may be used as a training dataset. Two test datasets can be obtain by selecting every first and second records from the training dataset. Another three training datasets can be created by taking every first, second and third records. (Note that there is no need to have different test datasets physically. They can be easily defined with "Data filter" over the training dataset using SQL conditions.)

Each dataset is required to have a ***unique record identifier field*** whose values are unique. For example, 1, 2, 5, 10, 11, …. It is recommended to use "RID" as the name (if not illegal). This field is required for updating records and plotting charts. In addition, this record identifier field should be ***indexed*** or be a ***primary key*** for performance reasons.

Normally a predictive model consists of a ***target variable*** (aka ***dependent variable***) and a number of ***independent variables***. Based on values of independent variables, models predict values of the model's target variable.

Independent variables will be discussed in the next section. The dependent variable is the prediction target. For risk modeling purposes, target is coded with the following two variables;

- **RiskFlag** : 0 and 1. If risk, then 1. No risk is 0. An integer data type is recommended as it can reduce dataset size when imported into CMSR.
- **RiskClass** : "Risk" or "Safe".

In insurance, "Risk" means insurance claims. In credit loans, it refers default and/or delinquency. For customer retention, it will mean customer churn. Models will use **RiskFlag** as the target variable. **RiskClass** will be used to analyse model fitness.

Neural network is poor in predicting information that training data does not contain. That is, it is poor in predicting ***unseen cases***. It may not learn functional information properly if training data does not contain enough information. This problem can be overcome by adding "***artificial data for missing cases***" with desired prediction. Especially data with null values with desired prediction can make models robust.

After data preparation, import the training data into CMSR Data Miner to begin with variable relevancy analysis for independent variables. This is described in the next section.

2. Independent Variables: Relevancy Analysis and Feature Extraction

Use of a large number of independent variables can lead to model ***overfitting***. In other words, models cannot learn statistical patterns. This can result in that models remember individual cases. Especially categorical variables with a large number of category items in neural networks are major problems. There are several methods that can be applied to avoid overfitting of models;

1. Exclude irrelevant independent variables.
2. Reduce irrelevant items in categorical independent variables by redesigning item values.
3. Use a smaller number of hidden layer nodes.
4. Use training datasets with a large number of records.
5. Use multiple models (bagging).
6. Use deep learning techniques.

First two methods are to reduce the number of input nodes in neural network by using only relevant information. These matters are further discussed in the subsections of this section that will follow.

Third method is to reduce the size of hidden layer nodes of neural network. Large hidden layer nodes can make neural network learn very detailed information. Reducing the number can force neural network to learn statistical patterns. This is further described in section “7. Neural Network Configuration”.

Fourth method is to use large training datasets with a large number of data records. Insufficiently large datasets can lead to overfitting. Large training datasets are recommended.

Fifth method is to use multiple models and combine model outputs using aggregate functions such as maximum, minimum and average values. This is called as bagging and can reduce overfitting.

Sixth method is extension of bagging methods by introducing integration networks.

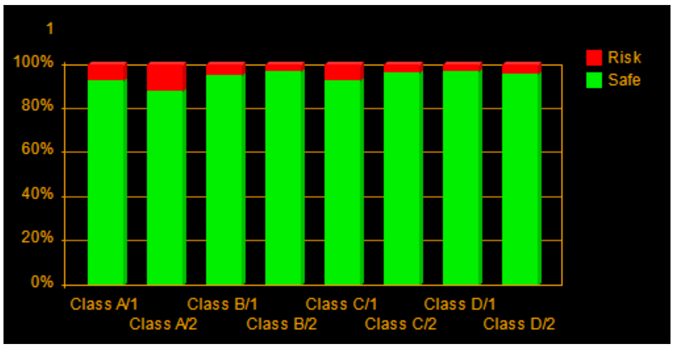
Bagging and deep learning methods are described in the paper “RME-EP and Deep Learning”. For more information, please read the paper.

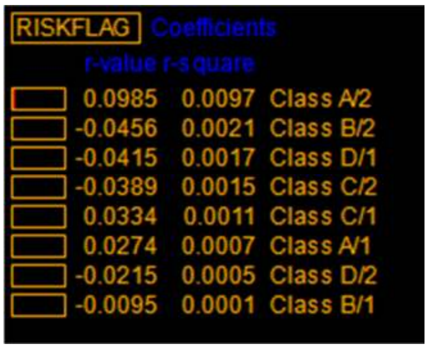
In the subsections that follow, independent ***variable selection*** and ***transformation*** methods will be described. This process can determine the success or failure of modeling.

2.1 Categorical variables

A categorical variable contains a finite number of category items. If the number is large, it will result in a large number of input and internal neural network nodes. Transforming category items into a small number of but more relevant variables is very important. This can be achieved by eliminating category variables and items which have no significance.

The following figures show relative “RiskClass” item distribution and “RiskFlag” correlation coefficients of a categorical independent variable;




RISKFLAG Coefficients

	r-value	r-square	
▭	0.0985	0.0097	Class A/2
▭	-0.0456	0.0021	Class B/2
▭	-0.0415	0.0017	Class D/1
▭	-0.0389	0.0015	Class C/2
▭	0.0334	0.0011	Class C/1
▭	0.0274	0.0007	Class A/1
▭	-0.0215	0.0005	Class D/2
▭	-0.0095	0.0001	Class B/1

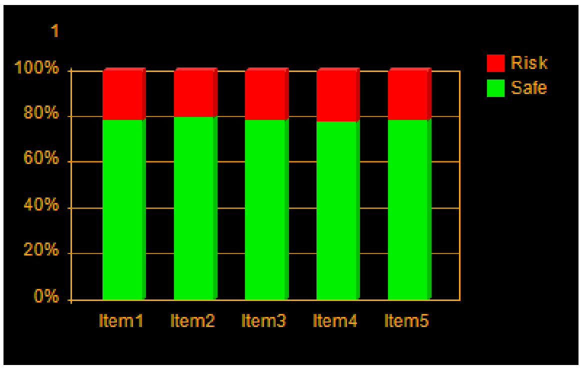
The first/left figure is a ***bar chart***, showing relative distribution of risk by category items. It was produced as follows. From CMSR, select a training dataset. Then select “Bars” from the “Analytics” menu. Select a category variable for ‘Item:’, “RiskClass” for ‘Category:’, and “1” for ‘Value(s)’. Press the “Draw” button. Then from the bar chart, select “Categoric proportion” for “Percent” and “Stacked/Cumulative” for “Stacking”.

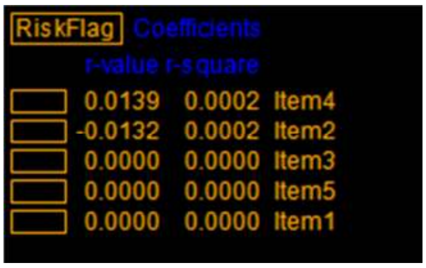
The second/right figure is a ***correlation analysis*** report. It shows correlation coefficients and squared values for each category item. It was produced as follows. From CMSR, select a training dataset. Then select “Correlation Analysis” from the “Analytics” menu. Select (=tick) a categorical variable for “Independent fields”. And select (=tick) “RiskFlag” for “Dependent fields”. Then press the “Analyze” button. From the main panel, click the box of “RiskFlag”. Then the right panel will show the report.

Correlation coefficients (=r-value) is 1 if there is perfect *positive* correlationship. When there is perfect *negative* correlationship, the value is -1. If there is no correlationship, it is 0. Thus larger absolute correlation coefficients mean higher correlationship. “r-square” is the squared values of “r-value”.

It is important to note that predictive models make different decisions based on the fact that different category items have different risk ratios. The left figure shows different category items having different risk ratios. Having this property, this category (of the previous page) variable is a good candidate for an independent variable. Although correlation coefficients are not large, still they are significant because when a larger number of items are involved, coefficients are normally small. This variable can be coded using the “1 of N” encoding scheme.

The following figures show another example where there is little difference in risk distribution amongst class items.



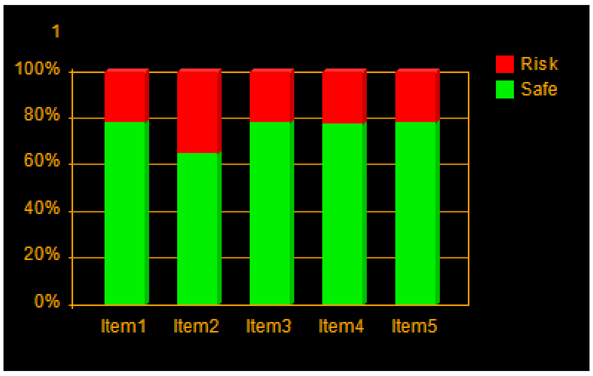


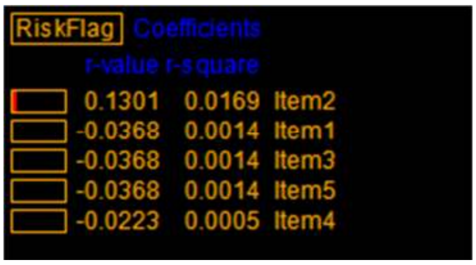
RiskFlag Coefficients

	r-value	r-square	
	0.0139	0.0002	Item4
	-0.0132	0.0002	Item2
	0.0000	0.0000	Item3
	0.0000	0.0000	Item5
	0.0000	0.0000	Item1

Unlike the previous example, this does not show any difference in risk ratios amongst category items. In addition, correlation coefficients are also small. It is recommended not to include this type of variables into models.

The following figures show an example of another category variable.





RiskFlag Coefficients

	r-value	r-square	
	0.1301	0.0169	Item2
	-0.0368	0.0014	Item1
	-0.0368	0.0014	Item3
	-0.0368	0.0014	Item5
	-0.0223	0.0005	Item4

Notice that the first figure shows the “Item2” has significantly higher risk ratio and coefficient than other items. Other items have “same or similar” risk ratios and smaller coefficients. This category variable can be transformed in two different ways. First method is to create another category variable consisting of “Item2” and “Others”. All items except “Item2” are converted as “Others”. Second method is to convert this to a numerical flag variable whose values take 0 and 1. If item value is “Item2”, new variable will have 1. Other items will be converted as 0. An integer data type is recommended for the new flag variable.

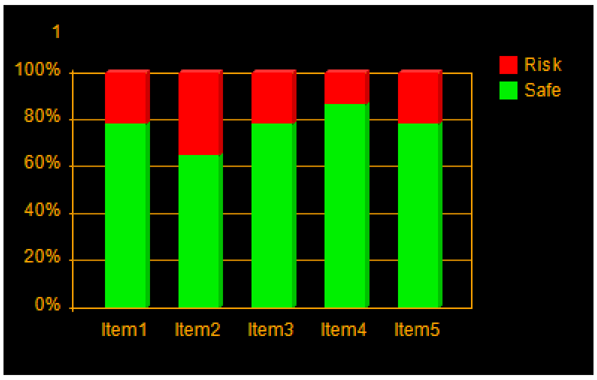
It is noted that these transformation methods are applicable when an item has significantly lower risk ratio while others have same or similar risk ratios as well. That is, when an item has significantly higher or lower risk ratio than others while other items have same or similar risk ratios.

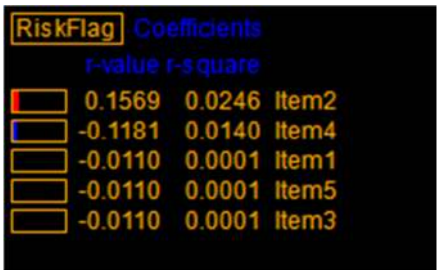
The following two RME-EP rules show these transformations. (Note that the first method is more suitable for decision trees, while the second is better with neural networks.)

```
// Convert "Item" as new category item "NewItem"
DECLARE "NewItem" AS STRING OUTPUT;
RULE r1:
        IF "Item" is NULL THEN
                SET "NewItem" AS NULL
        ELSE
                IF "Item" = 'Item2' THEN
                        SET "NewItem" AS 'Item2'
                ELSE
                        SET "NewItem" AS 'Others'
                END
        END;

// Convert "Item" as new numerical flag variable "NewItem"
DECLARE "NewItem" AS INTEGER OUTPUT;
RULE r2:
        IF "Item" is NULL THEN
                SET "NewItem" AS NULL
        ELSE
                IF "Item" = 'Item2' THEN
                        SET "NewItem" AS 1
                ELSE
                        SET "NewItem" AS 0
                END
        END;
```

The next example is when more than one item has significantly lower or higher risk ratios than the rest of items.




RiskFlag Coefficients

r-value	r-square	
0.1569	0.0246	Item2
-0.1181	0.0140	Item4
-0.0110	0.0001	Item1
-0.0110	0.0001	Item5
-0.0110	0.0001	Item3

Careful analysis shows that “Item2” and “Item4” have significantly higher and lower risk ratios than the rest of items. The rest items have same or similar risk ratios. “Item2” and “Item4” also have significant correlation coefficients.

The previous example can be extended to cater for multiple differing items as follows. As in the previous example, two methods can be applied. First method is to convert the categorical variable into another categorical variable whose values include “Item2”, “Item4” and “Others”. “Item2” and “Item4” will retain values. The other items will be converted as “Others”.

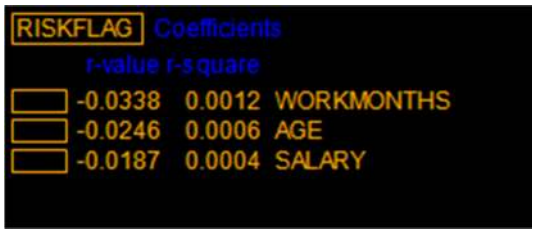
Second method is to create two numerical variables, one for “Item2” and another for “Item4”. Values will be set as 1 if true. Otherwise set to 0. The following rules show coding examples for RME-EP.

```
// Convert "Item" as new category item "NewItem"
DECLARE "NewItem" AS STRING OUTPUT;
RULE r3:
        IF "Item" is NULL THEN
                SET "NewItem" AS NULL
        ELSE
                IF "Item" = 'Item2' OR "Item" = 'Item4' THEN
                        SET "NewItem" AS "Item" // retain value
                ELSE
                        SET "NewItem" AS 'Others'
                END
        END;


// Convert "Item" as new numerical flag variables "NewItem2" and "NewItem4"
DECLARE "NewItem2", "NewItem4" AS INTEGER OUTPUT;
RULE r4:
        IF "Item" is NULL THEN
                { SET "NewItem2" AS NULL;
                  SET "NewItem4" AS NULL; }
        ELSE
                CASE
                WHEN  "Item" = 'Item2' THEN
                        { SET "NewItem2" AS 1;
                          SET "NewItem4" AS 0; }
                WHEN "Item"  = 'Item4' THEN
                        { SET "NewItem2" AS 0;
                          SET "NewItem4" AS 1; }
                ELSE
                        { SET "NewItem2" AS 0;
                          SET "NewItem4" AS 0; }
                END
        END;
```

2.2 Numerical variables

Unlike categorical variables, numerical variables do not have items. So bar charts cannot be used. We have to rely on correlation analysis. The following figure shows correlation coefficients of a number of numerical variables in relation to the "RiskFlag" target variable.



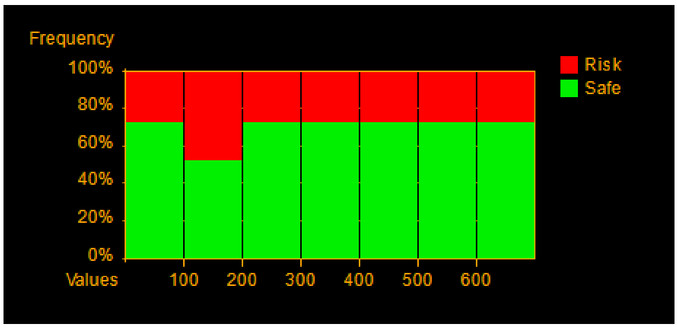
```
RISKFLAG  Coefficients
       r-value  r-square
☐ -0.0338  0.0012  WORKMONTHS
☐ -0.0246  0.0006  AGE
☐ -0.0187  0.0004  SALARY
```

This report was produced as follows. From CMSR, select a training dataset. Then select "Correlation Analysis" from the "Analytics" menu. Select (=tick) **all the numerical variables** for "Independent fields". And select (=tick) "RiskFlag" for "Dependent fields". Then press the "Analyze" button. From the main panel, click the box of "RiskFlag". Then the right panel will show this report.

Determining numerical variables which will be included in models is tricky business. In general, the absolute value of a correlation coefficient (=r-value) is less than 0.01 (or even less 0.005), it may not be worth for inclusion in models.

Normally, for neural network, numerical variables are used as they are. Binning is not recommended as it will increase the number of input nodes. (Note that for decision tree, however, binning is strongly recommended.)

There may be cases where transformations described for the categorical variables (described in the previous subsection) can be applied for numerical variables. For this purpose, histogram charts can be used. The following figure shows a histogram chart for a numerical variable "Variables".



This chart was produced as follows. From CMSR, select a training dataset. Then select "Histograms" from the "Analytics" menu. Select a numerical variable for "Value:", "RiskClass" for "Category:". Then from the bar chart, select "Categoric proportion" for "Percent".

This chart shows that values between 100 and 200 interval has much higher risk ratio than other intervals. Notice that other intervals have same or similar risk ratio. So treating the interval between 100 and 200 specially makes sense. We may transform this numerical field into new numerical field such that value is 1 if the original value is between 100 and 200. The rest will be 0. Note that if the interval has much lower risk ratio, the same transformation can be applied.

The following RME-EP coding implements this transformation;

```
// Convert "Variable" as new numerical flag variable "NewVariable"
DECLARE "NewVariable" AS INTEGER OUTPUT;
RULE r5:
        IF "Variable" is NULL THEN
                SET "NewVariable" AS NULL
        ELSE
                IF "Variable" >= 100 AND "Variable" < 200 THEN
                        SET "NewVariable" AS 1
                ELSE
                        SET "NewVariable" AS 0
                END
        END;
```

When more than one interval has significantly higher or lower risk ratios, the same transformation described for categorical variables can be used. Note that this type of transformation makes sense only if other intervals have same or similar risk ratios.

2.3 Transformation Plan Table

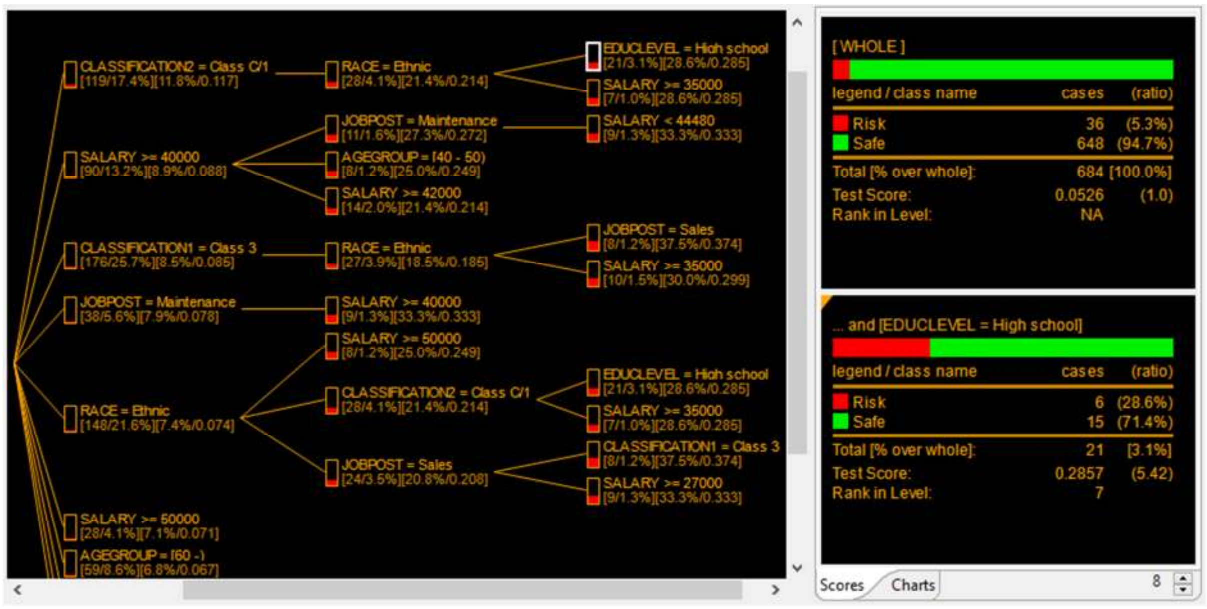
The following is an example table of data transformation plan. Similar sheet may be used to suit your requirement.

Variable	Items	Transformation	New Variables	Values	Remarks
Vocation	Farmer Miner Driver Teacher …	Transform Farmer and Miner	FarmerFlag MinerFlag	1/0 1/0	
…	…	…	…	…	

It is important to note that federal and state laws and regulations may prohibit using certain variables in credit and insurance scoring. For example, in USA, the Equal Credit Opportunity Act prohibits creditors from discriminating customers for the reasons of race, color, religion, national origin, gender, marital status, age, etc. Please check your local laws and regulations.

2.4 Checking with Hotspot Drill-down Analysis

CMSR Hotspot drill-down analysis can identify profiles of high risk customer segments in terms of categorical items and numerical value ranges. It can identify segments with highest ratios of risk. The following figure shows a hotspot drill down chart.


CLASSIFICATION2 = Class C/1 [119/17.4%][11.8%/0.117]
RACE = Ethnic [28/4.1%][21.4%/0.214]
EDUCLEVEL = High school [21/3.1%][28.6%/0.285]
SALARY >= 35000 [7/1.0%][28.6%/0.285]
SALARY >= 40000 [90/13.2%][8.9%/0.088]
JOBPOST = Maintenance [11/1.6%][27.3%/0.272]
SALARY < 44480 [9/1.3%][33.3%/0.333]
AGEGROUP = [40 - 50) [8/1.2%][25.0%/0.249]
SALARY >= 42000 [14/2.0%][21.4%/0.214]
CLASSIFICATION1 = Class 3 [176/25.7%][8.5%/0.085]
RACE = Ethnic [27/3.9%][18.5%/0.185]
JOBPOST = Sales [8/1.2%][37.5%/0.374]
SALARY >= 35000 [10/1.5%][30.0%/0.299]
JOBPOST = Maintenance [38/5.6%][7.9%/0.078]
SALARY >= 40000 [9/1.3%][33.3%/0.333]
SALARY >= 50000 [8/1.2%][25.0%/0.249]
RACE = Ethnic [148/21.6%][7.4%/0.074]
CLASSIFICATION2 = Class C/1 [28/4.1%][21.4%/0.214]
EDUCLEVEL = High school [21/3.1%][28.6%/0.285]
SALARY >= 35000 [7/1.0%][28.6%/0.285]
JOBPOST = Sales [24/3.5%][20.8%/0.208]
CLASSIFICATION1 = Class 3 [8/1.2%][37.5%/0.374]
SALARY >= 27000 [9/1.3%][33.3%/0.333]
SALARY >= 50000 [28/4.1%][7.1%/0.071]
AGEGROUP = [60 -) [59/8.6%][6.8%/0.067]
Scores
Charts
8

[WHOLE]

legend / class name	cases	(ratio)
🟥 Risk	36	(5.3%)
🟩 Safe	648	(94.7%)
Total [% over whole]:	684	[100.0%]
Test Score:	0.0526	(1.0)
Rank in Level:	NA	

... and [EDUCLEVEL = High school]

legend / class name	cases	(ratio)
🟥 Risk	6	(28.6%)
🟩 Safe	15	(71.4%)
Total [% over whole]:	21	[3.1%]
Test Score:	0.2857	(5.42)
Rank in Level:	7	

This chart was produced as follows. From CMSR, select a training dataset. Then select “Hotspot drill-down analysis” from the “Analytics” menu. Select (or tick) ***all potential independent variables*** from “Profiling fields”. Select “RiskClass” for “Field:” in “Hotspot target”. Choose “>> Risk” for “Value:”. Then select “Highest Probability” for “Hotspot criteria”.

Category items and numerical variables appearing in this report have high significance as they have higher risk ratios. They are recommended to be included in your independent variable list. Check whether they are already included. If not, go back to variable relevancy analysis for missing items and re-evaluate.

2.5 Implementing Transformation to Training Dataset

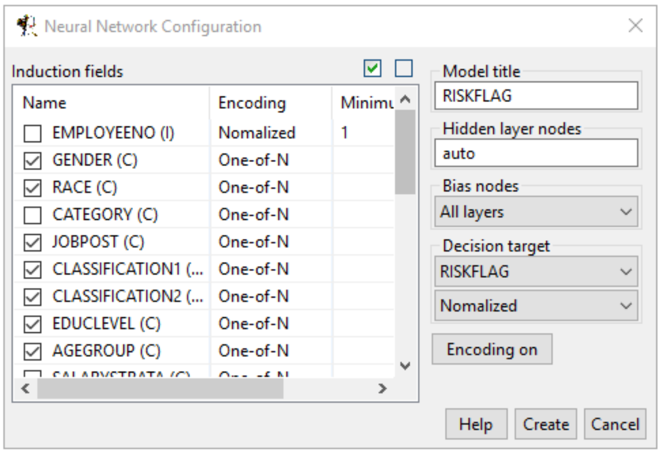
Once data transformation plan is ready, it’s time to implement and transform the training dataset database table. This is done in the following sequence;

1. Create new database table fields for newly created transformation variables.
2. Write a RME-EP model containing data transformation rules.
3. Apply the RME-EP model to the training dataset table using RME-EP’s “Apply to Database” function.

7. Neural Network Configuration

Neural network modeling starts with network configuration. To this end, you need to import training dataset again. This time, importing data should include variables of newly transformed values as described in the previous section. Once data import is completed, you are ready to configure a network and start training.

To configure a neural network model, open the imported training dataset. If the training dataset is already opened, select the dataset by clicking the tab of the dataset. Then select “New neural network” from the “Modeling” menu. Or simply press the neural network button from the toolbar. Then following dialog window will appear;



Induction fields

	Name	Encoding	Minimu
☐	EMPLOYEENO (I)	Nomalized	1
☑	GENDER (C)	One-of-N	
☑	RACE (C)	One-of-N	
☐	CATEGORY (C)	One-of-N	
☑	JOBPOST (C)	One-of-N	
☑	CLASSIFICATION1 (...	One-of-N	
☑	CLASSIFICATION2 (...	One-of-N	
☑	EDUCLEVEL (C)	One-of-N	
☑	AGEGROUP (C)	One-of-N	

Model title: RISKFLAG

Hidden layer nodes: auto

Bias nodes: All layers

Decision target: RISKFLAG

Nomalized

From “Induction fields”, select a set of independent variables by ticking the boxes in front. Use transformed variables and exclude original variables here. If needed, change “Encoding”, “Minimum” and “Maximum” values. To change this information, press the “Encoding on” button and click a variable. Then change the variable options.

For numerical variables, pay attention to “Minimum” and “Maximum” values to cover unseen future values. They should not be too small or too large. If needed, adjust them.

Then select “Decision target” as “RiskFlag”.

Specify “Hidden layer nodes”. This specifies a number of hidden layer nodes, e.g., 35, etc. When “auto” is specified, the number will be automatically calculated as the 60% of total input nodes. When it is empty, internal hidden layer will be omitted. This creates “regression” neural network.

It is noted that too many hidden layer nodes can lead to overfitting. On the other hand, too few internal nodes will result in poor accuracy. So a proper internal node size is recommended. A proper size can be found through testing multiple models with different sizes. Try “auto”. Then try higher and lower numbers of internal nodes.

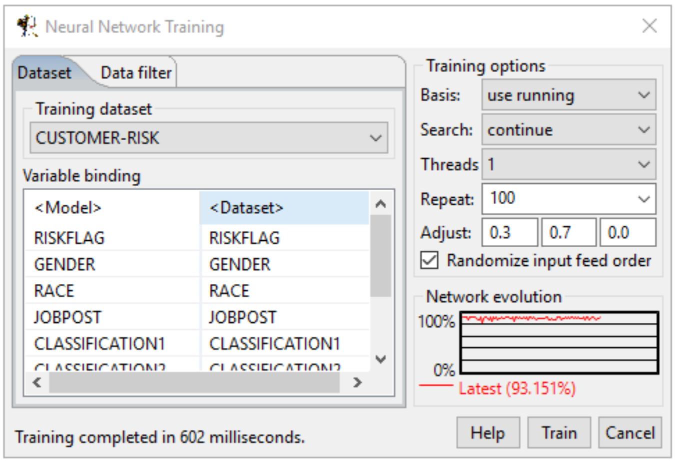
Finally, enter “Model title” and press the “Create” button.

Make sure to save newly created model for future use. To save, select the model as the font of the main frame. Then press the “Save model as” button. It is the first toolbar button.

8. Neural Network Training

Neural network training is repetitive process. You can train neural network many times. CMSR allows you to train network many different times, even using different training datasets.

To train a neural network model, open a training dataset and the model. Select the model to the front (of the main frame). Press the neural network button of the toolbar or select “Train neural network” from the “Modeling” menu. Or press the neural network button of the toolbar. Then the following dialog window will appear;



<Model>	<Dataset>
RISKFLAG	RISKFLAG
GENDER	GENDER
RACE	RACE
JOBPOST	JOBPOST
CLASSIFICATION1	CLASSIFICATION1

Basis: use running
Search: continue
Threads 1
Repeat: 100
Adjust: 0.3 0.7 0.0
☑ Randomize input feed order

Training completed in 602 milliseconds.

If needed, select “Training dataset” and “Variable binding”. If you use the same dataset used in creating the model, CMSR will automatically bind variables correctly. If you use different dataset with different variable names, you will need to select “Variable binding” manually.

The “Network evolution” shows recent accuracy ratio history and the “Latest” accuracy value (“Latest (93.151%)” in the figure). Reading this, you can determine whether network is fully trained.

The most important training parameter is “Error correction ratio”. This is the first value of “Adjust:” in the above dialog window. Start training with “0.5” as the “Error correction ratio”. If the “Latest” accuracy value fluctuates or does not change significantly, then move to next correction ratio “0.3” until the latest accuracy value fluctuates or does not change significantly again. Then move to “0.1”, “0.05”, and finally “0.01”. When completed, network training is done.

Training can automatically repeat a number of times for each training run. The “Repeat:” option specifies this. If you choose small numbers, you may have to repeat many training runs. If you choose too large numbers, training runs may take long time. A suitable number can be “1,000,000 / (records in training dataset)” or “5,000,000 / (records in training dataset)”. If the dataset is larger than that, you may choose 5 or 10.

Neural network models maintain two versions: the ***best*** and the ***running*** versions. The best version is automatically searched. If you want to reset the best version, select an option from “Search:”.

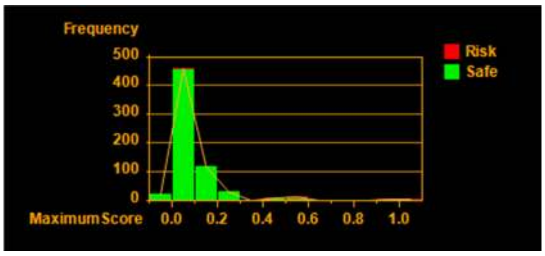
Make sure to save the model after network training. To save, press the “Save model as” button. It is the first toolbar button.

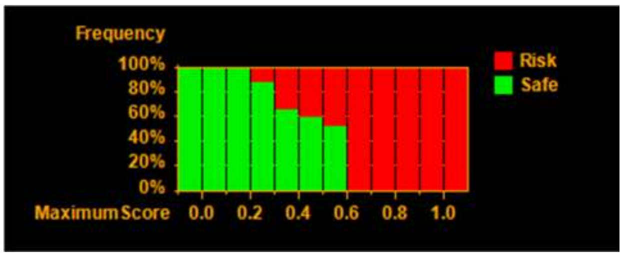
9. Model Fitness Testing

Once a neural network model is fully trained, the next step is to analyse model characteristics. This is called model ***fitness testing***. To test a model, the following preparation is needed.

(1) Create a database table field, say, "ModelScore" on the training dataset table. (Note that figures in this page use "MaximumScore" as this field.) A float data type such as "float" or "double" is required for this score field.
(2) Open the neural network model. Select the database and login. Press the "Apply to database" and update the new database table field "ModelScore" with model's output values.

This will store model's output values. Now it's time to analyse how the model's output values are distributed. The best way to view model desirability is to use ***histogram*** charts. The following charts are histograms showing predicted model scores and risk distribution.

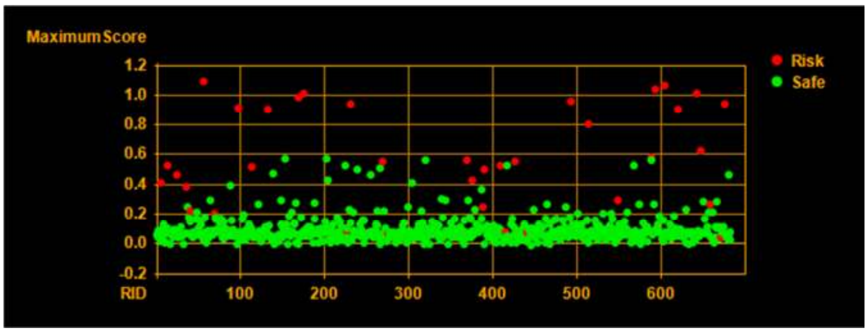




The left figure shows distribution of dataset samples over predicted "ModelScore" (="MaximumScore") values. Most samples scored less than 0.3. The right figure shows relative risk proportions by intervals. Red parts show risk proportions of intervals. The higher the predicted score is, the higher the risk (=red) proportion is. This is a very good attribute of good predictive models.

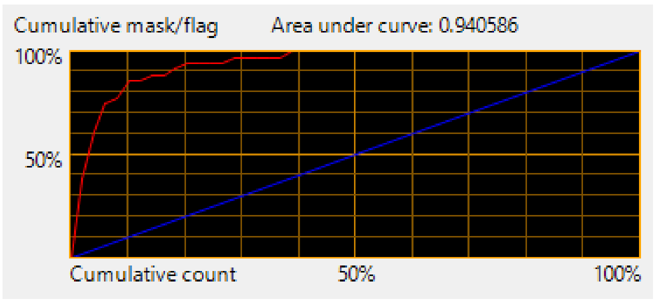
This chart was produced as follows. Select the database and login. From the "Database" menu, select "Histograms". From "Database table", select the training dataset database table. Select "ModelScore" (="MaximumScore") as "Value:". And select "RiskClass" as "Category:". Then the left figure chart will show up. To get the right figure chart, select "Categoric proportion" for the "Percent" option.

To see more detailed analysis of scores and risk distribution, the following ***scatter plot*** chart can be used. (Note that scatter plots are not suitable if there are too many samples. You may need to limit the sizes of plots using "Data filter". Record identifiers can be used as record filtering conditions.)



To produce this chart, first select the database that contains dataset tables and login. From the "Database" menu, select "Scatterplots". From "Database table", select the training dataset database table. Select a record identifier (RID) as "Independent (X):". Select "ModelScore" (="MaximumScore") as "Dependent (Y):". And select "RiskClass" as "Category:".

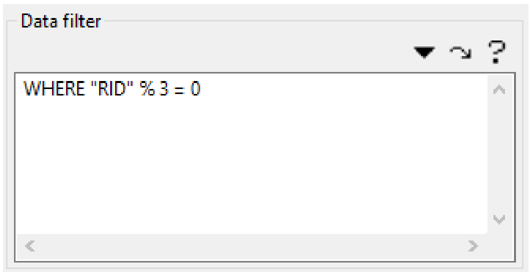
In addition to histogram and scatter plot charts, the following ***cumulative gains chart*** can be used to determine model fitness. This shows cumulative ratios of captured risk samples from descendingly sorted data on the "ModelScore" field. The red line shows the capture ratios of risk samples. Rapidly rising red line curve is considered as a good predictive model trait. This indicates how much risk samples are captured by highest scoring samples. "Area under curve" is the ratio of area under the red line curve. Maximum value is 1 and the larger number is better. This number is higher when higher scoring samples contain more/most risk samples.



To produce this chart, first select the database that contains dataset tables and login. From the "Database" menu, select "Cumulative gains chart". From "Database table", select the training dataset database table. Select "ModelScore" (="MaximumScore") as "Model score:". And select "RiskFlag" as "Mask/flag:".

5.1 Testing with Test Datasets

Finally, repeat this validation process to test datasets. Test datasets can be defined with the "Data filter" option which is available from the "Data filter" tab of chart dialog windows.



WHERE "RID" % 3 = 0

For two way test datasets, use the following conditions. Note that "RID" is the record identifier. Follow syntax of your database system.

```
WHERE "RID" % 2 = 0
WHERE "RID" % 2 = 1
```

For three way test datasets, use the following conditions;

```
WHERE "RID" % 3 = 0
WHERE "RID" % 3 = 1
WHERE "RID" % 3 = 2
```

Gender-specific test datasets can be coded with the following conditions. (Do not use independent variables in these conditions.)

```
WHERE "GENDER" = 'Male'
WHERE "GENDER" = 'Female'
```

10. Model Integration and Deep Learning

Models can be deployed directly onto web and Android devices as described in the next section. However in certain cases, it is desirable to use RME-EP rule engine to incorporate additional features. First case is when models may have been developed using transformed data as described in section "2. Independent Variables: Relevancy Analysis and Feature Extraction". In this case, it is desirable that end users still enter data using original information. Using RME-EP, data is transformed and fed to models automatically. Rules used to create transformed training data can be used and extended with neural network models. The following example codes show this usage. The first rule "r1" transforms data and stored on "NewItem" variable. Then the second rule "r2" use "NewItem" as input to model "Model1". Notice that the original input variable "Item" is defined with original input item values.

```
// Convert "Item" as new numerical flag variable "NewItem"
DECLARE "Item" AS STRING INPUT VALUES('Item1', 'Item2', 'Item3', …);
DECLARE "NewItem" AS INTEGR;
RULE r1:
        IF "Item" is NULL THEN
                SET "NewItem" AS NULL
        ELSE
                IF "Item" = 'Item2' THEN
                        SET "NewItem" AS 1
                ELSE
                        SET "NewItem" AS 0
                END
        END;

// Evaluate Model1 using transformed "NewItem"
DECLARE "ModelScore" AS REAL OUTPUT;
RULE r2:
        IF TRUE THEN
                SET "ModelScore" AS PREDICT(Model1) USING (
                        "……..'' AS "……",
                        "………" AS "NewItem",
                        "……..'' AS "……" )
        END;
```

Second case is ***risk classification***. "ModelScore" values will normally range between 0 and 1. But can be a bit lower or higher 0 and 1 respectively. This value can be used to classify risk level into "Very high risk", "High risk", "Medium risk", "Low risk, etc. RME-EP rules can be used for this transformation.

Third case is to incorporate multiple models and take average, minimum, maximum values, etc. This is called as "***bagging***". RME-EP provides seamless integration of models with powerful SQL-based rule specification language.

Fourth case is implementation of ***Deep Learning*** models. Multiple models can be developed using different modeling techniques. Then models are integrated with integration models. Rules may be used to interpret model outputs. And so on.

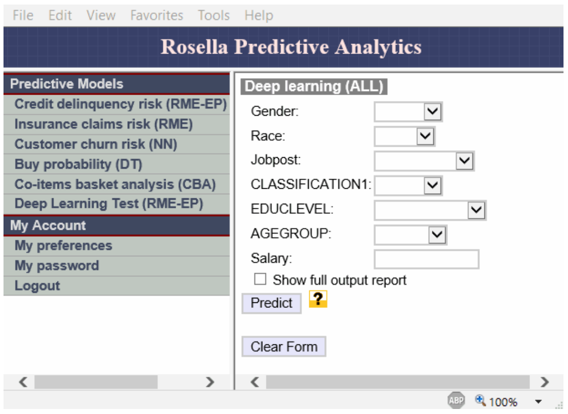
These three issues are fully described in the "RME-EP Deep Learning Guide" ("RME-EP-Deep-Learning.pdf") file. Please read the file.

11. Model Deployment

Once models are ready, next step is to deploy to end users. There are three forms of deployment for CMSR models.

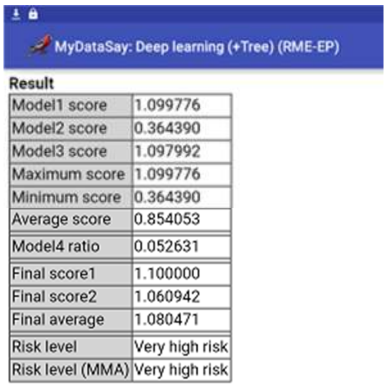
(1) Web Deployment

Models can be deployed using Rosella BI Sever on web. It's very simple to deploy. Just copying model and documentation files will do. The following shows an example screen of predictive model input on web browser.



(2) Android Deployment

MyDataSay Android application provides seamless deployment of models. The following figure shows an example output screen of a model.



Result

Model1 score	1.099776
Model2 score	0.364390
Model3 score	1.097992
Maximum score	1.099776
Minimum score	0.364390
Average score	0.854053
Model4 ratio	0.052631
Final score1	1.100000
Final score2	1.060942
Final average	1.080471
Risk level	Very high risk
Risk level (MMA)	Very high risk

(3) Integration with Host Applications

Rosella BI Server is based on Java J2EE. With JSP programing, models can be integrated into business applications. Template programs for JSON HTTP requests are also provided.

Rosella Software
www.roselladb.com