

Laboratory Manual

CS /IS C415

Data Mining

PoonamGoyal

Saiyedul Islam



EDD Notes

Birla Institute of Technology & Science
Pilani (Rajasthan)

Table of Contents

<i>Cover Page</i>	(i)
<i>Table of Contents</i>	(ii)
<i>Getting Started with IBM SPSS Modeler</i>	(iii)
1. Data Preprocessing	1
2. Classification: Decision Tree	6
3. Classification: K-Nearest Neighbor	11
4. Classification: Bayesian Network	14
5. Classification: Support Vector Machine.....	15
6. Classification: Classification and Regression Tree.....	18
7. Classification: Ensemble.....	21
8. Association Rule Mining: Apriori	23
9. Clustering: K-Means.....	25
10. Clustering: Self Organizing Maps.....	29
11. Real World Data Mining	31

Getting Started with IBM SPSS Modeler

The IBM SPSS Modeler is a data mining, modeling and reporting tool. It provides a nice GUI to carry out all the data mining tasks in form of Nodes and *Stream Flows*. **Nodes** are the icons or shapes that represent individual operations on the data. The nodes are linked together in a **stream** to represent the flow of data through each operation i.e. A set of actions (reading in, preprocessing, classification/association rule mining/clustering, reporting, etc.) on some input data is called a stream.

Modeler Interface -

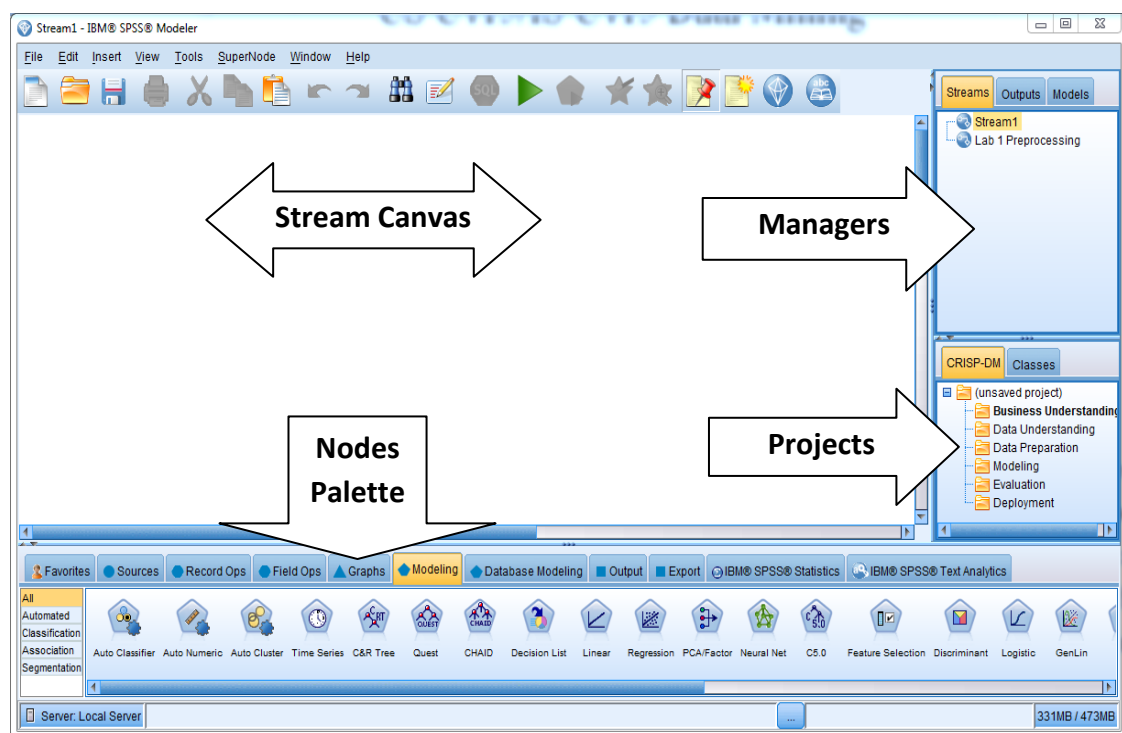


Fig 1 - SPSS Modeler Interface with main components

Modeling -

A model is a set of rules, formulas, or equations that can be used to predict an outcome based on a set of input fields or variables. For example, a financial institution might use a model to predict whether loan applicants are likely to be good or bad risks, based on information that is already known about past applicants.

To build a stream that will create a model, we need at least three elements:

- A source node that reads in data from some external source.
- A modeling node (classification, association, clustering, etc.) that generates a model nugget when the stream is run.
- [Optional] An output node if we want results in tabular or graphical form.

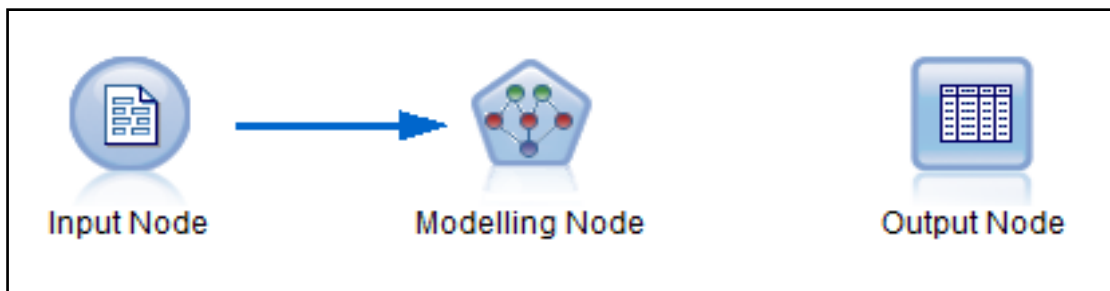







Fig 2 - An abstract stream

Source Nodes




Some important data source nodes are-





Symbol	Node type	Imports data from
	Database Node	MS SQL Server, DB2, Oracle (using ODBC)
	Variable File Node	Delimited text data (*.csv files)
	Excel Node	Microsoft Excel
	XML Node	XML files
	User Input Node	Generate synthetic data

Record Operation Nodes

Record operations nodes are used in data understanding and data preparation.







Some important record operation nodes are-

Symbol	Node type	Function
	Select Node	Selects or discards a subset of records from the data stream based on a specific condition <i>E.g. - Display all records having an attribute value above a threshold</i>
	Sample Node	Selects a subset of records using a sample type <i>E.g. - Display every fifth record</i>
	Aggregate Node	Replaces a sequence of input records with summarized output records <i>E.g. - Find class-wise mean and standard deviation of all records</i>

	Sort Node	Sorts record into ascending or descending order based on values of one or more fields
	Merge Node	Takes multiple input records from different sources and creates a single output record containing some or all of input fields
	Append Node	Concatenates sets of records
	Distinct Node	Removes duplicate records


Field Operation Nodes




These nodes are used to select, clean, or construct data in preparation for analysis. Some important field operation nodes are-

Symbol	Node type	Function
	Type Node	Specifies field metadata and properties. E.g. - measurement level (continuous, nominal, ordinal, or flag) for each field can be specified, options for handling missing values and system nulls can be set.
	Filter Node	Filters(discards) fields, rename fields, and maps fields from one source node to another
	Derive Node	Modifies data values or creates new fields from one or more existing fields <i>E.g. - Create a new field as the multiplication of two continuous fields</i>
	Filler Node	Replaces field values and changes storage (replace all blank values with a specific value)
	Binning Node	Creates new nominal fields based on the values of one or more existing continuous fields.
	Partition Node	Generates a partition field, which splits the data into separate subsets for the training, testing, and validation stages of model building.

Output Nodes




Output nodes provide the means to obtain information about data and models. Some important output nodes are-

Symbol	Node type	Function
	Table Node	Displays the data in tabular format, which can also be written to file

	Matrix Node	Creates a table that shows relationships between fields
	Analysis Node	Performs various comparisons between predicted values and actual values for one or more model nuggets
	Data Audit Node	Provides a comprehensive first look at the data, including summary statistics, histograms and distribution for each field, as well as information on outliers, missing values, and extremes.




Graph Nodes

These nodes are used for visualizing the data in a mathematical form. Some important graph nodes are-

Symbol	Node type	Function
	GraphboardNode	Offers many different types of graphs in one single node.
	Plot Node	Shows the relationship between numeric fields.
	Web Node	Illustrates the strength of the relationship between values of two or more symbolic (categorical) fields.

Export Nodes

These nodes provide a mechanism for exporting data in various formats to interface with other software tools. Some important export nodes are-

Symbol	Node type	Function
	Database Export Node	Writes data to an ODBC-compliant relational data source.
	Flat File Export Node	Outputs data to a delimited text file.
	Excel Export Node	Outputs data in Microsoft Excel Format (*.xls)

Lab 1

Data Preprocessing

Objective

Apply different kind of preprocessing techniques on given dataset.

Input Files-

Data file	=>	iris_data.csv
Metadata file	=>	iris_metadata.txt

Reading Data

1. Read the given metadata file and find out about the number of fields and their types (i.e. continuous, nominal, etc.), also have a look at input data file. Notice, field names are not mentioned.
2. Drag-n-drop a *Variable File Node* (from Sources tab in Nodes palette) on to *Modeler Canvas*.
3. Double-click on *Variable File Node* and go to "File" tab. Browse and select the given input data file.
4. Uncheck "Read field names from file", and check "Specify number of fields" and set the field count value according to information given in metadata file.
5. Go to "Data" tab and make sure that you are satisfied with the "Storage" type of each field. Otherwise, check "Override" and change the storage type.
6. Go to "Filter" tab and change the output field names according to the names mentioned in metadata file.
7. Go to "Types" and make sure "Measurement" of each field is correct (i.e. continuous, nominal, etc.)
8. [Optional] To know about range of values that each field have in this data, click on "Values" cell in front of the field, select "<Read>" and click on "Read Values" button.
9. [Optional] To mark a field as Classifier field (on which classification has to be performed), mark it as "Target" in "Role" cell.
10. Click "Apply" and "OK". Now, a source node with your input file name must appear on the modeler canvas.
11. [Optional] To view the output of this source node, add a "Table node"
12. Connect it to source node and "Run" the stream.

Running a stream

- To run whole stream, use the "Run Stream" button in toolbar
- To run down-stream from a particular node, right click on it and select "Run From Here"

Connecting Node N_1 to N_2

- Right-click on N_1 , select "Connect" and then select N_2 , or
- Select N_1 , press "F2" and then select N_2 .

Analyzing Input

1. To understand the input data, use "Data Audit Node" from "Output" tab in node palette and connect it to the source node.
2. Double-click on data audit node and go to "Settings" tab.
3. Either select "Default" for auditing all fields or select "Use custom fields" and select required fields.
4. Check all options in "Display".
5. Go to "Quality" tab and select "Outlier and Extreme Detection method". Specify the values for outliers and extremes.
6. Click "Apply" and "OK". Now, a data audit node will appear on the modeler canvas. "Run" the stream. (Ctrl+E)
7. Observe output ("Audit" and "Quality" tabs) carefully and try to determine necessity of preprocessing.

From "Audit" tab, determine answers of following questions.

- a. Is cardinality of nominal attributes correct?
 - E.g.- cardinality of Boolean attributes should be 2
 - See under column labeled as "Unique"
- b. Are Min, Max, Std. deviation, variance, etc. are under permissible limits?
- c. Does input data needs Sampling (Is it too large)?
- d. How values are distributed for each attribute?
 - See the graphs

From "Quality" tab, determine answers of following questions.

- a. Are there any "Outliers" and/or "Extreme" values?
- b. Are there any "Missing/Null" values?

Handling Missing Values

Remove records having a NULL value

1. Select a "Select Node" from "Record Ops" tab in nodes palette.
2. Connect source node to the select node.
3. Double click on select node and go to "Settings" tab.
4. In order to discard records with missing values select "Discard" and write the expression which selects such records. Expressions can be easily written using Expression Builder Tool.

- a. Click on *"Expression Builder"* button. Left hand side lists all functions with their return values and right hand side lists fields in the input dataset. Connectors specified in middle are used to connect two expressions.
 - b. To select all records with missing Field1 value, select *"@NULL"* function from function list and insert it into expression. Select Field1 from field list and insert it as argument of above mentioned function. *"@NULL"* is used for numeric values only, for string values use *"STRING1 matches STRING2"* and specify STRING2 as "".
 - c. To select all records with any one of the missing field values, write expressions as explained above, joined by *"OR"* operation.
 - d. Verify the expression by clicking *"Check"*. Remove syntax error, if any. Click *"OK"*.
5. Click *"Apply"* and *"OK"*. Now, a select node will appear on the modeler canvas.
 6. [Optional] To view the output of this select node, add a *"Table node"*. Connect select node to the table node and *"Run"* the stream from the select node.

Replace Numeric attributes by mean value

1. To obtain the mean of each field, connect a *"Set Globals"* node to the stream. Select required fields and operations (Mean, Sum, Min, Max and SDev). Click *Run*.
1. Select a *"Filler Node"* from *"Field Ops"* tab in nodes palette.
2. Connect source node to this filler node.
3. Double click on filler node and go to *"Settings"* tab.
4. Select all numeric fields in *"Fill in fields"*.
5. Select *"Replace"* condition as *"Blank and null values"*.
6. In *"Replace with"* box, open the *Expression Builder* window. Select *Globals* in right-hand side panel and select Global Mean of appropriate fields. In case of multiple fields selection use @FIELD keyword. Write following expression in replace with box -

`@GLOBAL_MEAN(@FIELD)`
7. Click *"Apply"* and *"OK"*. Now, a filler node will be appear on the modeler canvas.

Remove Nominal attributes having null value

1. In order to remove nodes with missing nominal values, select a *"Select Node"* from *"Record Ops"* tab in nodes palette.
2. Connect source node to this select node. Go to *"Settings"* tab.
3. Select *"Discard"* and write the expression which selects nominal attributes with null values using *"STRING1 matches STRING2"* and specify STRING2 as "".
4. Click *"Apply"* and *"OK"*. Now, a select node will appear on the modeler canvas.
5. [Optional] To view the output of this select node, add a *"Table node"*. Connect this node to the table node and *"Run"* the stream from the select node.

Discretization (Binning)

1. Select a *"Binning Node"* from *"Field Ops"* tab in nodes palette and connect select node of previous step to this node. Go to *"Settings"* tab.
2. Select all the numeric fields with *"Continuous"* values as *"Bin Fields"*.
3. As per the requirement, you may choose among many Binning methods like *"Fixed-width"* or *"Tiles"* or *"Ranks"* or *"Mean/Std Deviation based"*. For now, select *"Fixed-width"* method. Read *Help* to know what these term mean.
4. Select *"Number of bins"* as appropriate.
5. [Optional] Go to *"Bin Values"* tab, click on *"Read Values"* button. You can see the lower and upper range of each of the bins for each of the binned field.
6. Click *"Apply"* and *"OK"*. Now, a binning node will appear on the modeler canvas.
7. New Binned values are appended as new fields in the dataset. To remove old fields, select a *"Filter Node"* from *"Field Ops"* tab in nodes palette.
8. Connect binning node to this filter node and double click on it.
9. Click on *"Filter"* column of old fields to filter them out. Click *"Apply"* and *"OK"*.
10. [Optional] To view the output of this node, add a *"Table node"*. Connect this node to the table node and *"Run"* the stream from this node.

Sampling

1. Select a *"Sample Node"* from *"Record Ops"* tab in nodes palette. Connect select node of Step 1 to this sample node. Double click on sample node and go to *"Settings"* tab.
2. As per the requirement, you may choose among the *"Simple"* or *"Complex"* sample method. For now, select *"Simple"* method.
3. You can select the sampling criterion as either *"First n"* or *"1-in-n"* or *"Random %"*. Try to understand difference between all the operations.
4. Click *"Apply"* and *"OK"*. Now, a sample node will appear on the modeler canvas.
5. [Optional] To view the output of this sample node, add a *"Table node"*. Connect this node to the table node and *"Run"* the stream.

Normalization

4. Select a *"Filler Node"* from *"Field Ops"* tab in nodes palette. Connect select node of Step 1 to this filler node. Double click on sample node and go to *"Settings"* tab. Select all numeric fields in *"Fill in fields"*. Select *"Replace"* condition as *"Always"*.
5. In *"Replace with"* box, write your normalization expression. For example, for 0-1 normalization you should write
$$\frac{(@FIELD - @GLOBAL_MIN(@FIELD))}{(@GLOBAL_MAX(@FIELD) - @GLOBAL_MIN(@FIELD))}$$
6. Click *"Apply"* and *"OK"*. Now, a filler node will appear on the modeler canvas.

7. [Optional] To view the output of this filler node, add a *"Table node"*. Connect this node to the table node and *"Run"* the stream.

CorrelationDetermination

1. Select a *"Statistics Node"* from *"Output"* tab in nodes palette.
2. Connect select node of Step 1 to this node.
3. Double click on the node and go to *"Settings"* tab.
4. Under *"Examine"* box, select all numeric fields. You may check all statistics parameter under *"Statistics"* box.
5. Under *"Correlate"* box, again select all numeric fields.
6. You may adjust correlation strength parameters for weak, medium and strong correlation. Click *"OK"*.
7. Click *"Apply"* and *"OK"*. Now, a statistics node will appear on the modeler canvas.
8. Run the stream from this node and observe the correlations between different fields. It will help you in selecting features to be consider for further processing.

Assignment

1. Use *"Auto Data Preparation"* node and configure it to perform all types of preprocessing techniques explained above.
2. Replace outlier values with null values using *"Data Audit"*Node.
Hint: Go to *"Quality"* tab, select an action in *"Action"* Column. Generate *"Outlier and Extreme SuperNode"*. Connect source node to this super node.
3. Replace missing values by the mean of the values of records having same class value.
4. Replace missing values by majority voting of the values of records having same class value.
5. Perform Stratified Sampling and observe the difference in the output.
6. Perform binning using Tiles, Ranks and Mean based methods.
7. Perform z-score normalization without using Auto Data Preparation Node.

Lab 2

Decision Tree based Classification

Objective

Use the given data set to build a Decision tree based classification using C5.0 Algorithm and use it to predict the class of given cases. Also, analyze the difference on outcome with and without pruning.

Classification using IBM SPSS Modeler

The IBM SPSS Modeler offer a variety of classification models. These models use the values of one or more input fields to predict the value of one or more output, or target, fields. Some examples of these techniques are:

- Decision trees (C&R Tree, QUEST, CHAID and C5.0 algorithms)
- K-nearest neighbor
- Naive Bayesian classifier
- Neural networks
- Support vector machines
- Bayesian networks

Model Nuggets

A model nugget is a container for a model, that is, the set of rules, formulae or equations that represent the results of a model building operations. The main purpose of a nugget is for scoring data to generate predictions, or to allow further analysis of the model properties. Opening a model nugget on the screen enables you to see various details about the model, such as the relative importance of the input fields and/or Rule set used in creating the model. To view the predictions, you need to attach and execute a further process or output node.

Classification using C5.0 Node

C5.0 node works by splitting the sample based on the field that provides the maximum information gain at each level. Each subsample defined by the first split is then split again, usually based on a different field, and the process repeats until the subsamples cannot be split any further. Finally, the lowest-level splits are reexamined, and those that do not contribute significantly to the value of the model are removed or pruned. It can produce following two types of models -

- Decision tree
It is a straightforward description of the splits found by the algorithm. Each terminal (or "leaf") node describes a particular subset of the training data, and each case in the training data belongs to exactly one terminal node in the tree.

- Rule set

A rule set is a set of rules that tries to make predictions for individual records. In some cases, more than one rule may apply for any particular record, or no rules at all may apply. If multiple rules apply, either prediction of first applicable rule or prediction of majority voting of all applicable rules becomes the class of record. If no rule applies, a default prediction is assigned to the record.

Partitioning the data

1. Select a "Partition Node" from "Field Ops" tab in nodes palette and connect source node to this node. Double click on partition node and go to settings tab.
2. Name the "Partition Field" as *Partition Field*. Select "Train and test" and specify "70:30" ratio for training and testing data splits.
3. Go to "Generate" option on menu bar and "Generate Select node for Training Partition" and "Generate Select node for Testing Partition". Click "Apply" and "OK". Now, a Partition node and two Select nodes will appear on the modeler canvas. View training and testing records using a table node.
4. Connect *Training Select* node and *Testing Select* node to the partition node.

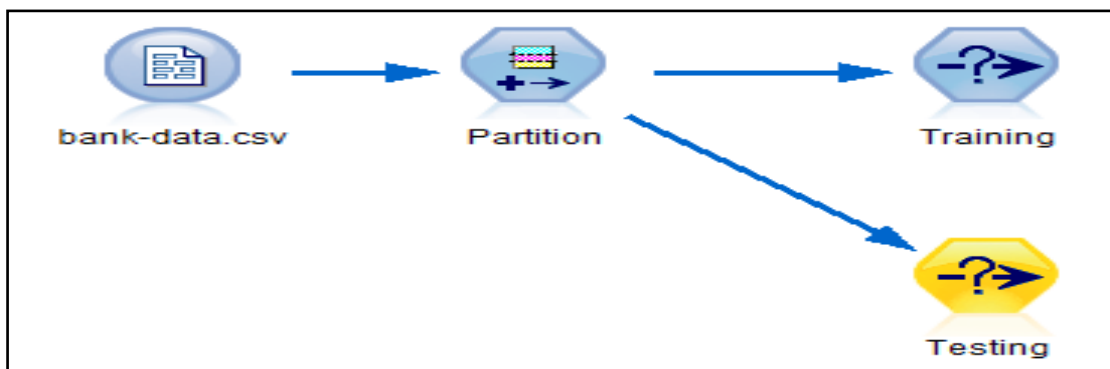


Fig 1- Stream after partitioning

Building Decision Tree

1. Select a "C5.0 Node" from "Classification" subgroup under "Modeling" tab in nodes palette and connect *Training select* node to this node.
2. Go to "Model" tab and select "Decision tree" as *Output type*. Uncheck *Use partitioned data* and *Build model for each split*. Uncheck "Calculate predictor importance" option on "Analyze" tab. Click "Apply" and "OK". Now, a C5.0 node with target field name will appear on the modeler canvas.
3. "Run" this stream from C5.0 node, a "Decision Tree Model Nugget" will be created and placed on stream canvas. Double click on model nugget and go to "Viewer" tab and see the decision tree. Analyze the tree carefully.

Generating Rule set

1. Go to "Model" tab of C5.0 node and select "Rule set" as *Output type*, this time. Select *Mode* as *Simple*. Go to *Annotations* and name the model as "C5.0 Simple mode". Click "Apply" and "OK". Copy and paste this C5.0 node and connect *Training Select* node to it.

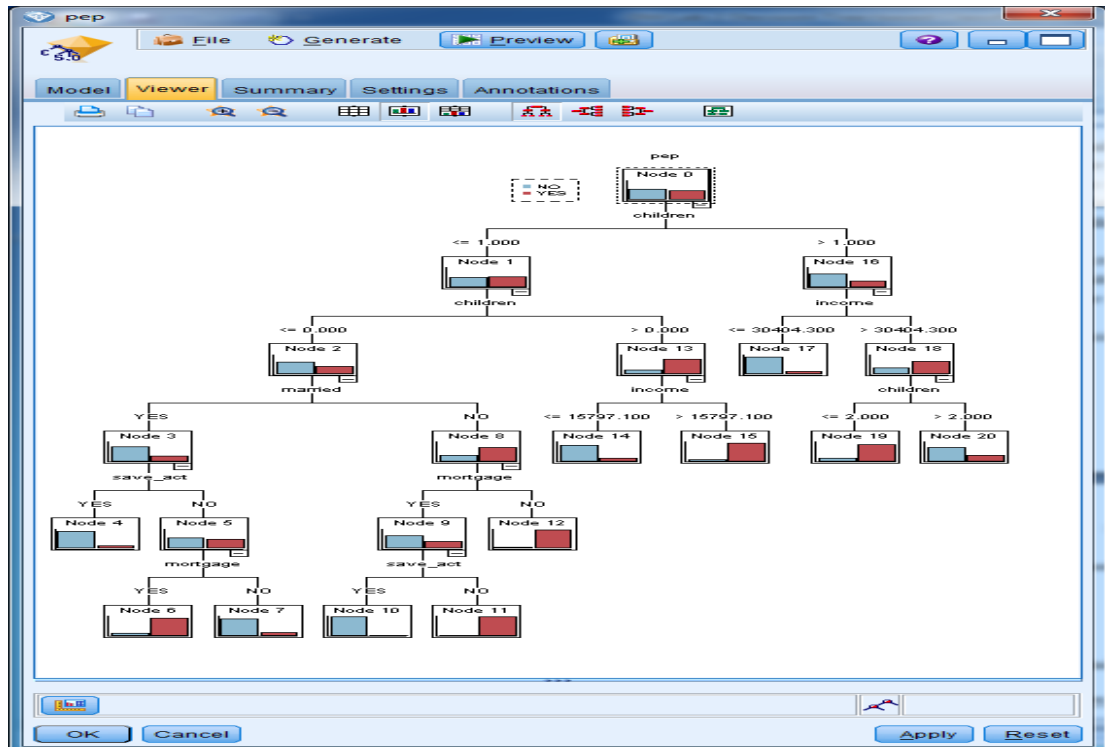


Fig 2- Decision tree

2. Go to "Model" tab of this copied C5.0 node and select "Rule set" as *Output type*, this time. Select *Mode* as *Expert*. Specify *Pruning severity* as 0 and *Minimum records per child branch* as 1. Uncheck *Use global pruning* and *Winnow attributes*. Uncheck *Use partitioned data* and *Build model for each split*.
3. Go to *Annotations* and name the model as "C5.0 Expert mode". Click "Apply" and "OK". This Expert mode node doesn't perform any type of pruning and creates a *unrestricted decision tree* (its rule set in this case).
4. "Run" this stream. Two model nuggets will get created. Go to their *Model* tab and observe the difference in Rule sets.
5. Take a *Merge* node from Record Ops tab in nodes palette. Connect both model nuggets to this node. Configure the merge node to merge the data on the basis of *Keys* and select all keys except the predicted class and corresponding confidence fields of both models. Rename these fields accordingly.
6. Connect merge node to an *Analysis* node. Run the stream and observe the accuracy of both the models.

Predicting class of test cases

1. Go to "Generate" option in the menu bar of *Simple mode* model nugget and select "Rule Trace Node". A SuperNode named as "RULE" will get generated on modeler canvas. Rename it as "RULE: Simple mode". A SuperNode groups multiple nodes into a single node by encapsulating sections of a data stream, in this case, all the rules specified by Rule Set.
2. Select this SuperNode and choose "Zoom-in" option from "SuperNode" menu from the menu bar of SPSS modeler. A stream of *Derive node* will appear, each driving a new field based on the conditions in Rule Set. Double-click on the nodes to see the deriving condition.
3. Third-last node derives a field having concatenation of all the rules or its value is "No rules". (This node uses concatenation operator ><). Second-last node derives another field which contains the concatenated rule set without the trailing semi-colon. Last node will be a *Filter* node which will be filtering out all the intermediate fields generated by different rules from final outcome.
4. Value of these derive fields contains predicted outcome of the rule along with its confidence value. In order to only compare the predicted value, we need to modify the output. [Optional] Add a *Table* node to last/second last node to see the output.
5. To make prediction of first applicable rule as the prediction of the record, select a "Derive" node from "Field Ops" tab under in nodes palette. Insert it in between second-last and last node. Choose "Flag" under *Derive as* option. Specify True value as "YES" and False value as "NO". Specify the true condition as

'RULE' matches "YES*"

This will put "YES" in newly derived field of all the records which had a string starting with YES in their RULE field.

6. Go to filter node, filter out RULE field and rename newly derived field as "*Simple mode prediction*". Zoom-out of SuperNode. Repeat all steps from 1 to 9 for *Expert mode* model nugget. Do the renaming correspondingly. Add *Table* nodes to these SuperNodes to see the predicted values.

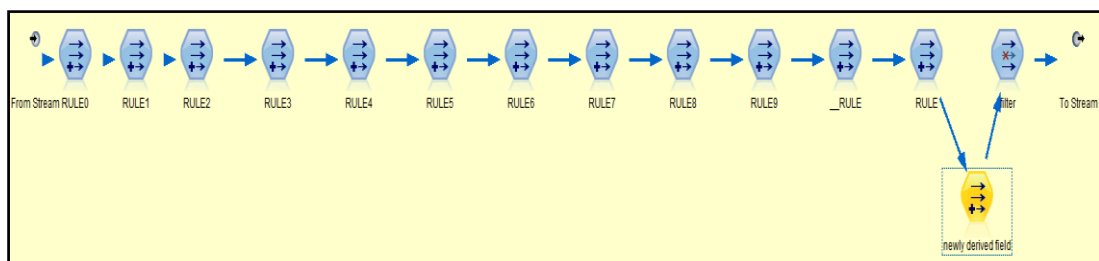


Fig 3- Zoom-in view of Simple mode SuperNode

Comparing performance

1. Select a "Merge" node from "Record Ops" tab in nodes palette. Connect *Simple mode* and *Expert mode* SuperNodes to this *Merge* node. Go to *Merge* tab and select "Keys" as Merge Method. Select all possible keys.
2. Select a "Matrix" node from "Output" tab in nodes palette. Under *Settings* select *Simple mode prediction* in Rows and *Expert mode prediction* in Columns.
3. Select *Cell-contents* as *Cross-tabulations*. Click on *Run* to see the results. Analyze it carefully.
4. Vary the "*Pruning Severity*", repeat all steps from beginning and see the change in this matrix.
5. Connect a *Table* node to *Merge* node. Under "*Highlight records where*", write appropriate condition for highlighting records where prediction of both the modes differ. Select "*Output to file*" and file type as *.html.

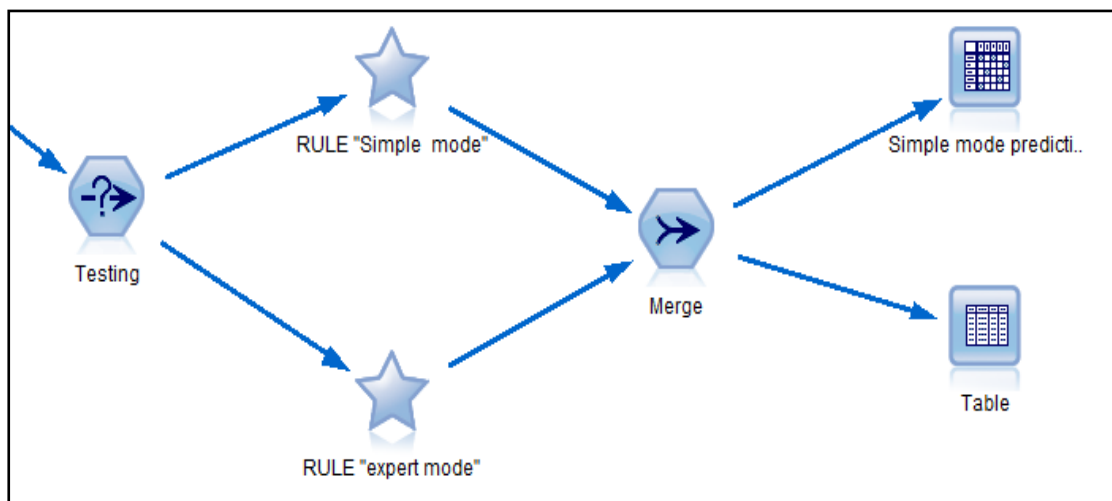


Fig 4- Stream for comparing results on testing data

Assignment

1. Pass both the partitions to the C5.0 models and use Analysis node to know the accuracy of model.
2. Observe the difference in model by performing following operations -
 - Using misclassification cost to heavily penalize false positives
 - Generating a Rule Set from already generated Decision Tree
 - Performing Majority Voting to select class of record in case of multiple rule applicability
 - Preprocessing the data before applying C5.0
 - Performing Global Pruning and Winnow attributes
3. Select a field as Split field and generate models for each split.

Lab 3

K-nearest Neighbor based Classification

Objective

Use the given data set to build a K-nearest neighbor (KNN) model and use it to predict the class of given cases.

KNN based Classification

The SPSS Modeler offers two type of analysis on KNN model-

- Predict a target field
 - To predict the value of a target field based on the values of its nearest neighbors
 - User can decide whether speed, accuracy, or a blend of both, are the most important factors when predicting a target field.
- Only identify the nearest neighbors
 - To determine K nearest neighbors for a particular input field

Data Preprocessing

1. Read in the input dataset using appropriate source node. Select an "Auto Data Prep" node from *Field Ops* tab in nodes palette and connect source node to it. Go to *Objectives* tab of the node and select *Optimize for accuracy*. Analyze the *Settings* tab and explore through its options. Go to *Analysis* tab and click on *Analyze Data* button in the menu bar. Select *Fields* option from the drop-down list on left-hand side panel.
2. Select *Do not use* option from the drop-down list adjoining the field names for fields having *Predictive Power* below a threshold (0.1 in this case). Select *Field Details* on right-hand side panel and explore through details of all the fields one by one.
3. Go to *Generate* menu item in the menu bar of the *Auto Data Prep* node and click on *Filter* node. Connect this node to auto data prep node to.

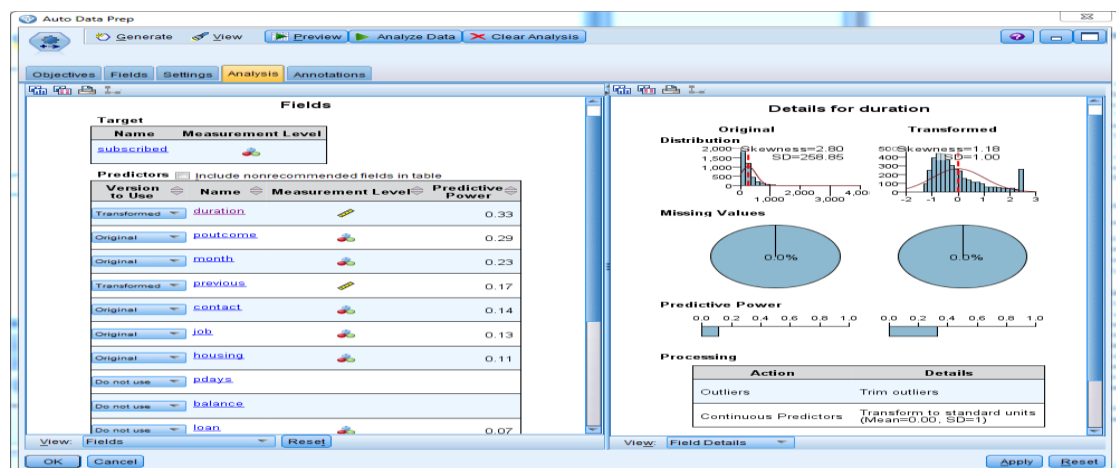


Fig 1 - Analysis tab of Auto Data Prep node

Joining Test Cases with Input Dataset-

1. Read the "to be predicted" dataset using appropriate source node. Select an "Append" node from *Record Ops* tab in nodes palette and connect "to be predicted dataset" and filter node of input data set to it. Go to "Inputs" tab of *Append* node and make sure that input dataset is above to be predicted dataset. To view the resulting fields, go to "Append" tab.
2. Add a "Type" node from *Field Ops* tab in node palette and connect *Append* node to it. Mark the *Role* of target field as *Target* under *Types* tab of this node.

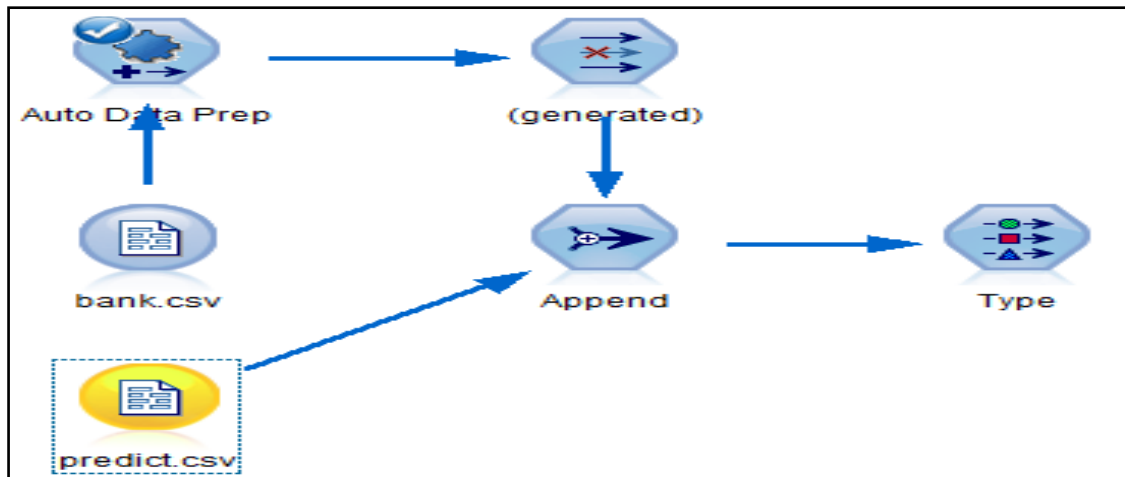


Fig 2 - Stream after step 3

Performing K-nearest neighbor Classification-

1. Select a "KNN Node" from "Classification" subgroup under "Modeling" tab in nodes palette and connect type node to it. Go to *Objectives* tab of this node and select "Predict a target field". Go to *Settings* tab and select *Model* subsection. Uncheck check boxes stating "Use partitioned data" and "Build model for each split". Go to *Neighbors* subsection and specify K as 5 and *Euclidean metric* as *Distance Computation* method.
2. Select either perform *Feature Selection* or *V-fold Cross-validation*. Select the later in this case and specify number of folds as 5. Run the stream. A *model nugget* with the name of target field will get created.
3. Go to *Model* tab of model nugget. Select any point on the left-hand side panel to see its *Peers Chart* and *Nearest Neighbors and Distance tables* on right-hand side panel.

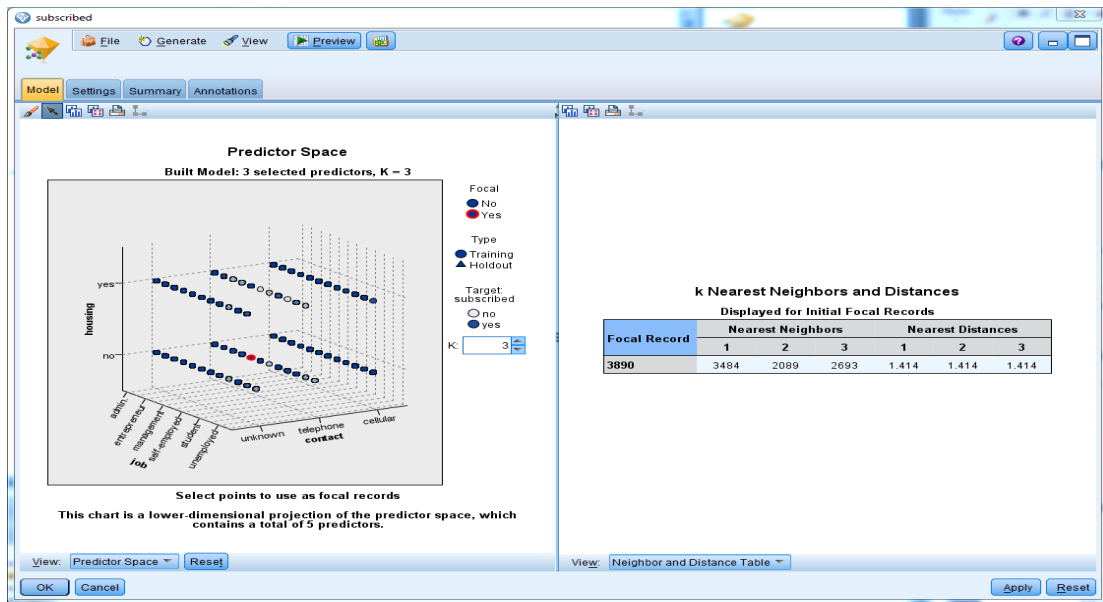


Fig 3 - Model tab of Model Nugget

Determining the class of test cases-

1. Connect a *Table* node the model nugget and *Run* it. Combined dataset with two additional fields (predicted value of target field and its probability) will be presented.
2. In order to get prediction values of only test cases, connect model nugget to a *Select* node. Go to its *Settings* tab, select *Mode* as *Include* and specify the expression as `@NULL(< target-field>)`. This will output records which didn't have target-field already predicted i.e. test cases.
3. Since, few fields had been dropped during preparation, in order to get back test cases with all the original fields select a *Merge* node and connect *Select* node of last step and *Source* node of test cases, to it. Go to its *Merge* tab and select *Keys* as *Merge Method* and select all keys as *Possible keys*.
4. Select a *Table* node and connect *Merge* node to it. Go to *Output* tab of table node and select *Output to file* and *File type* as `*.html`. Properly name these files with your ID, as you may be required to submit them for evaluation.

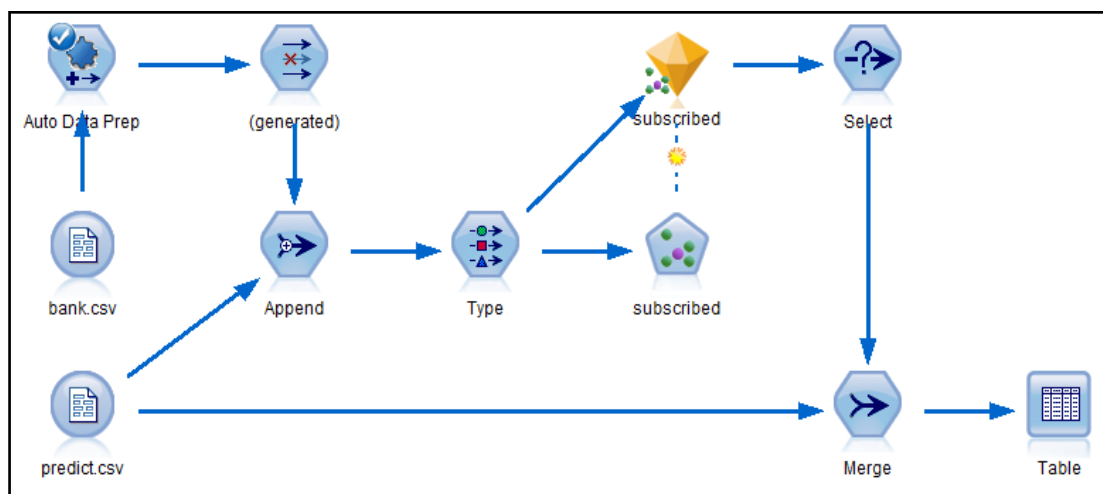


Fig 4 - Complete Stream

Lab 4

Bayesian Network based Classification

Objective

Use the given data set to build Bayesian network based classification model using different structuring methods and analyze the effect of feature selection on these methods.

Classification using Bayes net Node

The SPSS Modeler offers two type of structuring methods modes for Bayesian network model-

- Tree augmented naive (TAN) bayes model
 - It allows each predictor to depend on another predictor in addition to the target variable. It is an improvement over the standard Naïve Bayes model to improve the accuracy of standard Naive Bayes model.
- Markov Blanket model
 - This selects the set of nodes in the dataset that contain the target variable's parents, its children, and its children's parents. Essentially, a Markov blanket identifies all the variables in the network that are needed to predict the target variable.

Building Bayesian network model using TAN-

1. Read in the data. Partition it into 70:30 ratios using partitioning node.
2. Connect it with a "*Bayes net*" node from *modeling* tab of nodes palette.
3. Select *Structure type* as TAN under *Model* tab of bayes net node.
4. Uncheck *Include feature selection preprocessing step* option and *Run* the model.
5. Go to model nugget and analyze the bayesian network on left-hand side panel and field-wise conditional probabilities on right-hand side panel.
6. Connect an *Analysis* node and see the percentage of correctly classified records in training and testing partitions.
7. Take another Bayes net node and this time check the feature selection preprocessing option.
8. Analyze the difference in Bayes network created through the two approaches. Use Analysis node to see the difference in training and testing errors.

Building Bayesian network model using Markov Blanket-

1. Repeat the above mentioned procedure by selecting Markov blanket as *Structure type*.
2. Finally determine percentage variation in training and testing errors with these four approaches.

Lab 5

Support Vector Machine based Classification

Objective

Use the given data set to build SVM based classification model. Also, analyze the difference on outcome by varying different parameters of SVM generation.

Classification using SVM Node

The SPSS Modeler offers two type of modes for SVM model-

- Simple mode
- Expert mode

Partitioning the data-

1. Select a "*Partition Node*" from "*Field Ops*" tab in nodes palette and connect source node to this node.
2. Double click on partition node and go to settings tab.
3. Name the "Partition Field" as *Partition Field*. Select "*Train and test*" and specify "70:30" ratio for training and testing data splits.
4. Click "*Apply*" and "*OK*". Now, a Partition will be visible on the modeler canvas.
5. [Optional] To view which records have been marked as "training record" and which as "testing record", add a "*Table node*", "*Connect*" it to partition node and "*Run*" the stream.

Building SVM model in Simple mode-

1. Select a "*SVM Node*" from "*Classification*" subgroup under "*Modeling*" tab in nodes palette and connect *Training select* node to this node.
2. Go to "*Model*" tab and select "*Use partitioned data*" as *Output type*.
3. Go to "*Expert*" tab and select *Mode* as "*Simple*". Go to "*Annotations*" tab and rename the model as "*SVM Simple*".
4. Click "*Apply*" and "*OK*". Now, a SVM node will be visible on the modeler canvas.
5. "*Run*" this stream from SVM node, a "*SVM Model Nugget*" will be created and placed on stream canvas.

Building SVM model in Expert mode-

1. Repeat steps 1 and 2 as explained above.
2. Go to "*Annotations*" tab and rename the model as "*SVM Expert*".
3. Go to "*Expert*" tab and select *Mode* as "*Expert*".
4. Change different parameters to increase the *accuracy* of model.

5. "Run" this stream from SVM node, a "SVM Model Nugget" will be created and placed on stream canvas.
6. [Optional]Connect a *Analysis* node to this nugget and see the partition wise classification error.

Table 1- Explanation of different expert mode parameters of SVM model

Parameter	Explanation
Stopping criteria	Determines when to stop the optimization algorithm. Values range from 1.0E-1 to 1.0E-6; default is 1.0E-3. Reducing the value results in a more accurate model, but the model will take longer to train.
Regularization parameter (C)	Controls the trade-off between maximizing the margin and minimizing the training error term. Value should normally be between 1 and 10 inclusive; default is 10. Increasing the value improves the classification accuracy (or reduces the regression error) for the training data, but this can also lead to overfitting.
Regression precision (epsilon)	Used only if the measurement level of the target field is Continuous. Causes errors to be accepted provided that they are less than the value specified here. Increasing the value may result in faster modeling, but at the expense of accuracy.
Kernel type	Determines the type of kernel function used for the transformation. Different kernel types cause the separator to be calculated in different ways, so it is advisable to experiment with the various options. Default is RBF (Radial Basis Function).
RBF gamma	Enabled only if the kernel type is set to RBF. Value should normally be between 3/k and 6/k, where k is the number of input fields. For example, if there are 12 input fields, values between 0.25 and 0.5 would be worth trying. Increasing the value improves the classification accuracy (or reduces the regression error) for the training data, but this can also lead to overfitting.
Gamma	Enabled only if the kernel type is set to Polynomial or Sigmoid. Increasing the value improves the classification accuracy (or reduces the regression error) for the training data, but this can also lead to overfitting.
Bias	Enabled only if the kernel type is set to Polynomial or Sigmoid. Sets the coef0 value in the kernel function. The default value 0 is suitable in most cases.
Degree	Enabled only if Kernel type is set to Polynomial. Controls the complexity (dimension) of the mapping space. Normally you would not use a value greater than 10.

Comparing performance-

6. Select a "Merge" node from "Record Ops" tab in nodes palette.
7. Connect *Simple mode* and *Expert mode* model nuggets to this *Merge* node.

8. Go to *Merge* tab of merge node and select "*Keys*" as Merge Method. Select all possible keys except predicted target field and its probability field (Fields starting with \$\$-sign).
9. Connect an *Analysis* node to the merge node. *Run* the stream
10. Carefully analyze the results i.e. performance of individual models and their comparison with original value of target field which was give in the dataset.
11. Vary the parameters on *Expert* node to generate better models.
12. *Export* analysis results to *.html file. Concatenate your ID in their names. You may be required to submit them for evaluation.

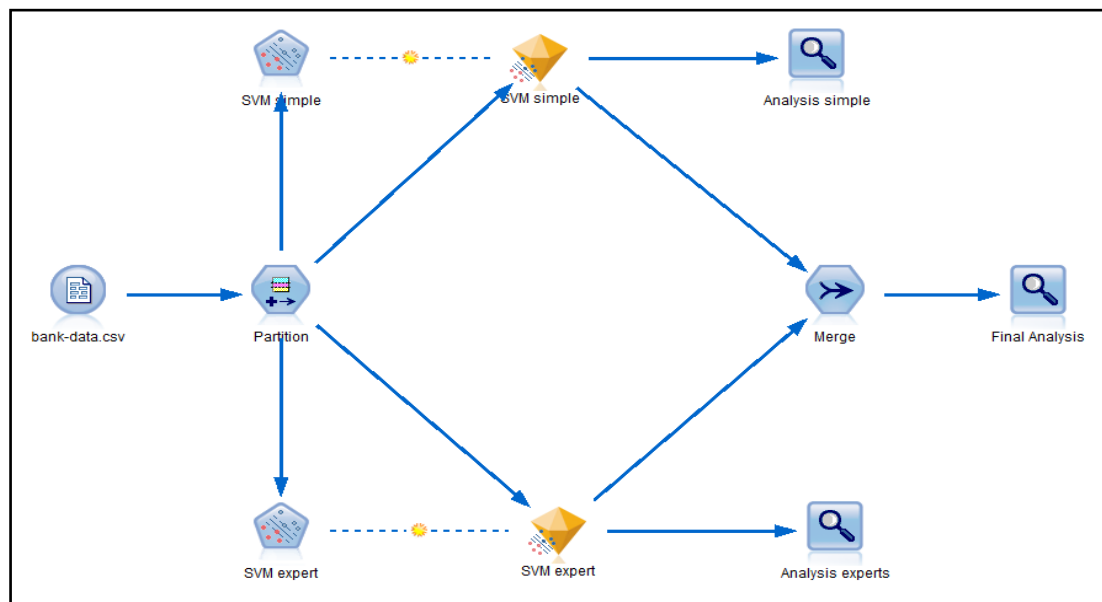


Fig 1- Stream for comparing SVM models

Assignment

1. Find out the default value of all the parameters in Simple mode.

Lab 6

C&R Tree based Classification

Objective

Explore different methods for performing tree based classification using C&R tree node.

C&R Tree Node

The Classification and Regression (C&R) Tree node is a tree-based classification and prediction method. It uses recursive partitioning to split the training records into segments with similar output field values. The C&R Tree node starts by examining the input fields to find the best split, measured by the reduction in an impurity index that results from the split. The split defines two subgroups, each of which is subsequently split into two more subgroups, and so on, until one of the stopping criteria is triggered. It provides following modes for building the decision tree -

- Single Tree
- With Boosting
- With Bagging
- Interactive tree building

Building Single C&R Tree

1. Read the given metadata file and read in data using appropriate source node. Connect it to a partitioning node and specify the ratio as 70:30.
2. Connect partition node to a C&R tree node. In *Build Options* tab, select *Build a single tree* as the objective. Run the stream.
3. Connect model nugget to an Analysis node.
4. Observe the Training and Testing errors. Tune following parameters to get better results
 - Maximum Tree Depth
 - Prune tree to avoid overfitting
 - Minimum records in child and parent
 - Overfit prevention set %
5. After getting best results, select *Output to file* option in analysis node and export the results into a *.html file.

C&R Tree with Boosting

1. Connect partition node to another C&R tree node. In *Build Options* tab, select *Enhance model accuracy(boosting)* as the objective. Specify the appropriate

combining rule for categorical and continuous targets in the *Ensembles* menu. Also, specify value for number of component models. Run the stream.

2. Connect model nugget to an Analysis node. Run the stream from model nugget.
3. Observe the change in accuracy from Single Tree model. Try to tune various parameters to get best possible model.

C&R Tree with Bagging

1. Connect partition node to another C&R tree node. In *Build Options* tab, select *Enhance model stability(bagging)* as the objective. Specify the appropriate combining rule for categorical and continuous targets in the *Ensembles* menu. Also, specify value for number of component models. Run the stream.
2. Connect model nugget to an Analysis node. Run the stream from model nugget.
3. Compare the accuracy value with previous two models. Try to tune various parameters to get best possible model.

Interactive Tree building

1. Connect partition node to another C&R tree node. In *Build Options* tab, select *Launch interactive session* as the objective and tune following parameters -
 - Maximum Tree Depth
 - Prune tree to avoid overfitting
 - Minimum records in child and parent
 - Overfit prevention set %
2. Run the stream. An interactive tree generation window will get open.
 - A tree with root node will be displayed in "Viewer" tab.
 - To grow the tree, from the menus select **Grow Tree**. The system builds the tree by recursively splitting each branch until one or more stopping criteria are met. At each split, the best predictor is automatically selected based on the modeling method used.
 - Alternatively, select **Grow Tree One Level** to add a single level.
 - To add a branch below a specific node, select the node and select **Grow Branch**.
 - To choose the predictor used for a split, select the desired node and select **Grow Branch with Custom Split**.
 - To prune a branch, select a node and select **Remove Branch** to clear up the selected node.
 - To remove the bottom level from the tree, select **Remove One Level**.
 - Select **Grow Tree and Prune** to prune based on a cost-complexity algorithm that adjusts the risk estimate based on the number of terminal nodes, typically resulting in a simpler tree.

- Go to "*Gains*" tab during tree generation process and you can see statistics for all terminal nodes in the tree. These statistics are "Tree Growing Set" and "Overfit Prevention Set".
 - Go to "*Risks*" tab during tree generation process and you can see the chances of misclassification at any level. You will "*Misclassification Matrix*" for both "Tree Growing Set" and "Overfit Prevention Set".
3. In the end, go to "*Generate*" menu and select "*Generate Model*". A new model nugget will be placed on the canvas. Connect partition node to it and connect it to an analysis node. Compare the accuracy of all four models.

Assignment

1. Select "Highest Probability Wins" and "Highest Mean Probability" in the combining rule for categorical target and observe the difference in the boosting and bagging model.

Lab 7

Ensemble based Classification

Objective

Understand the working of ensemble based classification by implementing

- Ensemble of same model applied on subsets of records (Bagging)
- Ensemble of different Models applied on same data
- Ensemble of same model applied on subset of fields (Random Forest)

Ensemble Node

The Ensemble node combines two or more model nuggets to obtain more accurate predictions than can be gained from any of the individual models. By combining predictions from multiple models, limitations in individual models may be avoided, resulting in a higher overall accuracy. Models combined in this manner typically perform at least as well as the best of the individual models and often better.

Ensemble of same model applied on subsets of records (Bagging)

1. Read the given metadata file and read in data using appropriate source node. Take a Partition node, connect source node to it and specify the partition ratio as 90:10. Generate Training and testing select nodes and connect partition node to both of them.
2. Connect training select node to *ten* Sampling nodes. In all nodes select *Complex* sample method, specify the Sample type as *Random* and Sample size as *Fixed (0.1)*. Make sure that random seed in all of them are different.
3. Take *five* Append Nodes and connect *one* pair of sampling nodes to each one of them. Connect testing select node to all the append nodes. Connect each append node to a type node and choose appropriate measurement type and role for each field.
4. Connect each type node to a classification node (preferably C5.0) and configure it to generate best possible model.
5. Save the project (stream). Run all the streams. Connect each model nugget to a select node and specify the condition as to select only testing records.
6. Connect all testing select nodes to a Merge node. Configure the merge node to merge the data on the basis of *Order*. Include whole input field set of one source and predicted fields from all sources, rename predicted fields as appropriate. Filter out rest of the fields.
7. Connect this merge node to an *Ensemble* node from Field Ops tab in nodes palette. Select appropriate *Target Field* and *Ensemble method* as Voting.
8. Connect the ensemble node to an *Analysis* node and Run the stream from merge node.

9. Carefully analyze the output. See the accuracy of each model and the accuracy of ensemble of these models. Try to tune each model individually in order to get most accurate model.

Ensemble of models applied on same data

13. In place of connecting all streams to same classification node in Step 5 above, now connect each stream to a different classification model (C5.0, KNN, SVM, C&R Tree, etc.).
14. Change the field names on merge node accordingly. Run all the streams.
15. Carefully analyze the output. See the accuracy of each model and the accuracy of ensemble of these models. Try to tune each model individually in order to get most accurate model.

Ensemble of same model applied on subset of fields (Random Forest)

1. Insert a Filtering Node in between all Append and Type Nodes in Bagging Model. Configure all filtering nodes as to allow target and partition field along with a mutually exclusive subset of fields and filter out remaining fields.
2. Connect all streams to C5.0 decision tree node. Do necessary settings on merge node and Run all the streams.
3. Carefully analyze the output. See the accuracy of each model and the accuracy of ensemble of these models. Try to tune each model individually in order to get most accurate model.

Assignment

Change Ensemble method to Confidence-weighted voting and Highest confidence wins and observe the changes in all three cases.

Lab 8

Apriori based Association Rule Mining

Objective

Use the given data set to generate association rules using Apriori algorithm.

Association Rule Mining using Apriori -

The Apriori node discovers association rules in the data. Association rules are statements of the form

if antecedent(s) then consequent(s)

Support refers to the percentage of records in the training data for which the antecedents are true. **Confidence** is based on the records for which the rule's antecedents are true and is the percentage of those records for which the consequent(s) are also true.

Building Apriori model

5. Read in the data. Identify the fields that represent a product (we are mining association rules in between products), set their role as "Both" in Types tab. For remaining fields mark role as "None".
6. Connect it with a "Apriori" node from *Modeling* tab of nodes palette.
7. On *Fields* tab, select all fields in *Consequents* as well as *Antecedents*. (Goal is to extract every possible rule).
8. On *Model* tab uncheck *Use partitioned data* and supply *Min Support%* and *Min Rule Confidence%* as 10% both.
9. Select *Simple* mode on *Expert* tab. *Run* the stream and see the model nugget. Use filtering options provided there to select rules having support and confidence % more than 50%.
10. Export the selected rules to *.html file.
11. Connect source node to a "Web Node" from *Graph* tab of nodes palette. Select all product fields, check "Show True Flags Only" and select Line Values are "Overall Percentage". Specify appropriate values for "Weak links below" and "Strong links above" parameters and Run the node.
12. Observe the web carefully. Strong links will result in bi-directional rules coming out of Apriori node.

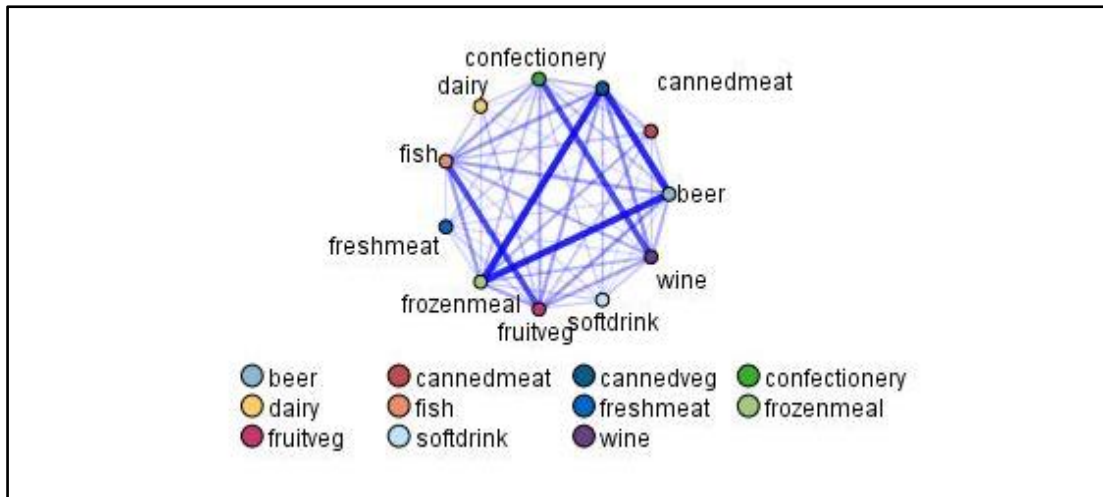


Fig 1- Output of a web node

Assignment

1. Go to *Expert* tab of model and select mode as *Expert*. Generate association rules through different *Evaluation measures* -

Evaluation Measure	Description
Rule confidence	The default method uses the confidence (or accuracy) of the rule to evaluate rules
Confidence difference	It is the absolute difference between the rule's confidence and its prior confidence.
Confidence ratio	It is the ratio of rule confidence to prior confidence
Information difference	It is based on the concept of Information gain
Normalized Chi-square	It is a statistical index of association between antecedents and consequents

Lab 9

K-means based Clustering

Objective

Find the optimal value of number of clusters (K) for K-means algorithm for a given data set through scripting in SPSS Modeler.

Scripting in SPSS Modeler

Like any other scripting language, scripting in SPSS helps in automating tasks that would be highly repetitive or time consuming to perform manually. Here, a script could be a Stream script or a Standalone script or a SuperNode script.

Creating & Saving streams

```
create stream <stream-name>
save stream as <stream-name with full path>
```

Local Scripting Variables

- Declaration `var<variable-name>`
- Initialization `set <variable-name> = <value>`
- Usage `^<variable-name>`

Variable names, such as `^newVar`, are preceded with a caret (^) symbol when referencing an existing variable whose value has already been set. The caret is not used when declaring or setting the value of the variable.

Creating Nodes

```
create NODE
create NODE at X Y
create NODE between NODE1 and NODE2
create NODE connected between NODE1 and NODE2
```

NODE can be any valid node like *variablefilenode*, *typenode*, *tablenode*, etc.

Nodes can be created using variables also. For example -

```
var x
set x = create variablefilenode
rename ^x as "Input"
position ^x at 200 200
```

Referencing Nodes and its properties

Nodes can be specified by name or by type or by its unique ID. In case of multiple nodes of same type, specification by type will give error. For example, all three commands below will set *space* as delimiter for the variable file node.

```
set ^x.delimit_space = true
set Input.delimit_space = true
set :variablefilenode.delimit_space = true
```

Properties of any node can be accessed using dot (.) operator. Like in above cases, the property `delimit_space` of `variablefilenode` has been set to true.

For complete list of properties of individual nodes, refer SPSS Properties reference manual.

Connecting Nodes

```
connect NODE1 to NODE2
connect NODE1 between NODE2 and NODE3
```

Conditional and Iterative command execution

- if...then...else

```
if EXPR then
    STATEMENTS 1
else
    STATEMENTS 2
endif
```

- for...endfor

```
for PARAMETER in LIST
    STATEMENTS
endfor

for PARAMETER from M to N
    STATEMENTS
endfor
```

K-means using Scripting

Given an input dataset, upper and lower limit of K, write a script to do following tasks -

10. Read given input dataset properly.
11. For each value of K from lower limit to upper limit
 - a. Connect source node to a K-means node. Specify value of K and run the model.
 - b. Connect source node to model nugget
 - c. Connect model nugget to a derive node and derive a new field as square of distance of the point from its cluster
 - d. Connect derive node to a "Set Globals" node and set it to set sum of squared errors(SSE) as a global variable
12. Connect a report node to source node. Configure it to write a *.CSV file as output where each line will contain value of K and corresponding SSE for all the values of K in the user specified range. Run the node.
13. Read above generated file into another source node.
14. Connect this source node to a plot node and configure it to draw a plot of number of cluster Vs. SSE. Value of K having abrupt change in slope represents an optimal value of number of clusters.

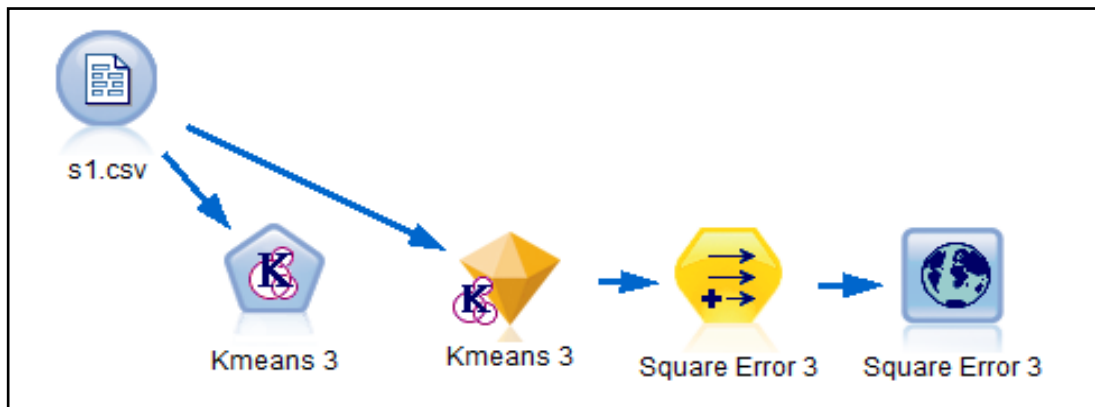


Fig 1 - Stream after one iteration of for loop in the above script

Complete Script

```

varsource_file_path
setsource_file_path = "C:\Users\Dell\Documents\DM lab
Manuals\Lab 6 Kmeans"
varsource_file_name
setsource_file_name = "\s1.csv"
varlower_limit
varupper_limit
setlower_limit = 2
setupper_limit = 5
varlname
vardname
var t
var temp
var d
varlname
var g
varvname
varymid
setymid = 150 * (^upper_limit/2)

create stream "Kmeans Analysis using Scripting"
setvname = create variablefilenode
position ^vname at 150 ^ymid
set ^vname.full_filename = ^source_file_path><
^source_file_name
set ^vname.read_field_names = false
set ^vname.delimit_space = true
set ^vname.multi_blank = true
set ^vname.default_value_mode = Read

fori from ^lower_limit to ^upper_limit
    set temp = 150 * (^i-1)
    setkname = create kmeansnode
    position ^kname at 250 ^temp
    set t = "Kmeans " >< ^i
    set ^kname.model_name = ^t
    connect ^vname to ^kname
    set ^kname.num_clusters = ^i
    set ^kname.gen_distance = true

```

```

        execute ^kname
        insert model ^t:kmeansnode at 350 ^temp
        connect ^vname to ^t:applykmeansnode
        setdname = create derivenode
        position ^dname at 450 ^temp
        connect ^t:applykmeansnode to ^dname
        set d = "Square Error " >< ^i
        set ^dname.new_name = ^d
        set ^dname.result_type = Formula
        set ^dname.formula_expr = "'$KMD-" >< ^t >< "'*'$KMD-" ><
^t >< "'"

        setgname = create setglobalsnode
        position ^gname at 550 ^temp
        set ^gname.custom_name = ^d

        connect ^dname to ^gname
        set ^d:setglobalsnode.globals.^d = [Sum]
        set ^d:setglobalsnode.clear_first = False
        execute ^dname
    endfor

    createreportnode
    connect ^vname to :reportnode
    set :reportnode.output_mode = File
    set :reportnode.output_format = Text
    set :reportnode.full_filename = ^source_file_path>< "kmeans-
output.txt"
    set :reportnode.text = ""
    fori from ^lower_limit to ^upper_limit
        set :reportnode.text = :reportnode.text>< ^i>< ",
[@GLOBAL_SUM('Square Error " >< ^i>< ")]\n"
    endfor
    execute :reportnode

    setvname = create variablefilenode
    position ^vname at 650 ^ymid
    set ^vname.full_filename = ^source_file_path>< "kmeans-
output.txt"
    set ^vname.read_field_names = false
    set ^vname.delimit_comma = true
    set ^vname.multi_blank = true
    set ^vname.default_value_mode = Read
    set ^vname.type.field1 = Range
    set ^vname.type.field2 = Range
    set ^vname.new_name.field1 = "No of Clusters"
    set ^vname.new_name.field2 = "Sum of Squared Errors"

    createplotnode
    position :plotnode at 750 ^ymid
    connect ^vname to :plotnode
    set :plotnode.x_field = "No of Clusters"
    set :plotnode.y_field = "Sum of Squared Errors"
    set :plotnode.title = "Plot of SSE Vs. K"
    execute :plotnode

```

Lab 10

Self-organizing Map based Clustering

Objective

Use the given data set to build a Kohonen(SOM) Clustering model

Clustering using Kohonen Node -

The basic units of Kohonen network or Self-organizing Maps are neurons, and they are organized into two layers: the input layer and the output layer (also called the output map). All of the input neurons are connected to all of the output neurons, and these connections have strengths, or weights, associated with them. During training, each unit competes with all of the others to "win" each record. As training proceeds, the weights on the grid units are adjusted so that they form a two-dimensional "map" of the clusters.

Building SOM model

1. Read in the data. Connect it with a "*Kohonen*" node from *modeling* tab of nodes palette.
2. On *Model* tab uncheck *Use partitioned data* and check *Show feedback graph* (a visual representation of the two-dimensional array is displayed during training. The strength of each node is represented by color. Red denotes a unit that is winning many records (a strong unit), and white denotes a unit that is winning few or no records (a weak unit)). Since, this is displayed only if training takes significant time the input dataset should be large enough and at the same time number of output neurons shouldn't be too many to avoid extremely large training time.
3. Select *Simple* mode on *Expert* tab. *Run* the stream and see the feedback graph. A model nugget will get created in the end.
4. Go to *Model* tab of model nugget and select *Model Summary* on left hand side panel. Find out the value of *Silhouette measure of cohesion and separation* for current clustering output.
5. Select *Clusters* as View on left hand side panel. Hold on *ctrl* key while clicking on all columns in first row of cluster table. This will select first five clusters and there comparison will be displayed on right hand side panel. Carefully analyze the comparison.

Assignment

1. Adjust the parameters in *Expert* mode such that value of *Silhouette measure* gets maximized. Uncheck the *Show feedback graph* option to avoid waiting long during every execution of model.

Parameter	Description
Width and length	It is size of two-dimensional output map.
Learning rate decay	It is a weighting factor that decreases over time, such that the network starts off encoding large-scale features of the data and gradually focuses on more fine-level detail.
Phase 1	It is a rough estimation phase, used to capture the gross patterns in the data
Phase 2	It is a tuning phase, used to adjust the map to model the finer features of the data.
Neighborhood	This determines the number of "nearby" units that get updated along with the winning unit during training.

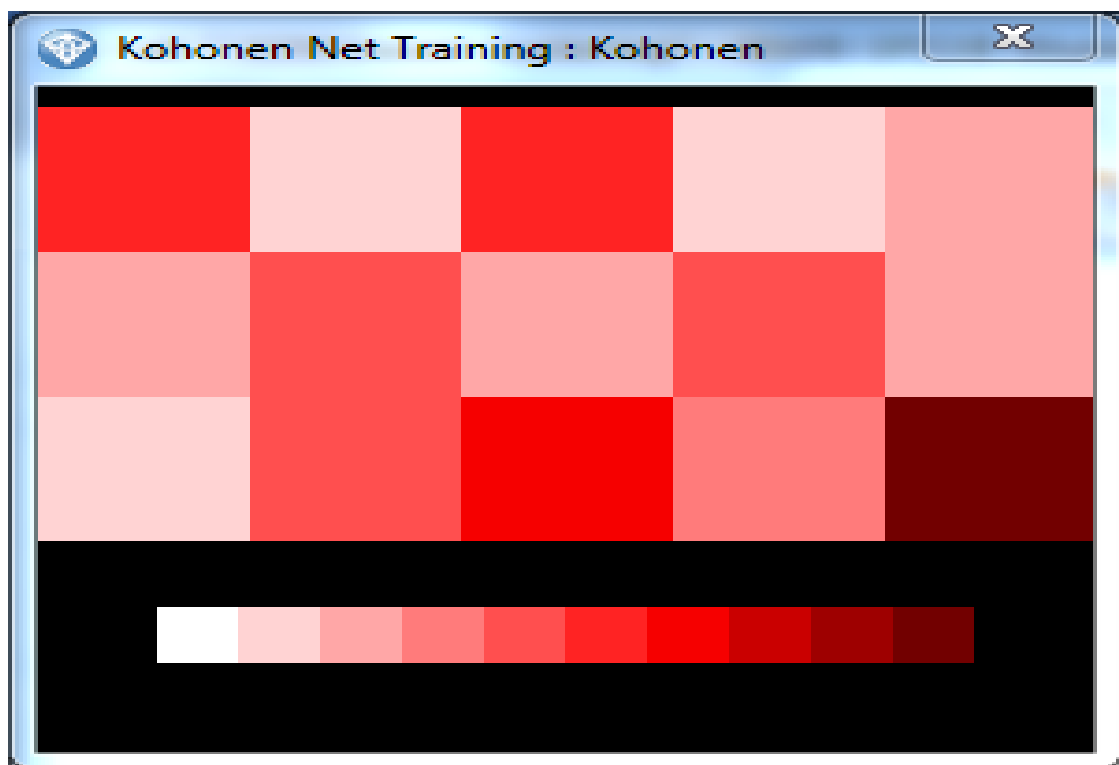


Fig 1- Feedback graph showing 5 x 3 output map during training

Lab 11

Real World Data Mining

Objective

Use learned data mining techniques to identify relevant patterns and useful information in a given data set.

Understanding the dataset

Go through the data set and try to find its domain. Understand the meaning of all fields individually. Identify the dataset in terms of the attributes present in the data, the number of instances, missing values, and other relevant characteristics.

Objectives of Data Mining Experiments

Figure out 3 to 5 specific, interesting questions **about the domain** that you want to answer with the data mining techniques. Try to choose questions that are about the domain not about the techniques or the experimental parameters.

Performance Metrics

Explain what performance metric(s) will be used and why to evaluate the models that you have constructed in order to answer questions identified (For example - accuracy, error rate, etc).

Preprocessing of the data

Identify the set of preprocessing techniques depending upon your choice of model and the given dataset. Explain the necessity of each such step.

Model Interpretation

Describe the resulting model (e.g., size of the model, readability, accuracy, etc.). Summarize the model in your own words focusing on the most interesting/relevant patterns. Elaborate on if and how the model answers the objectives identified.

Performance of the resulting model

- State the performance of the model. For example, elaborate on the confusion matrix and/or other relevant performance indicators.
- How long it took IBM SPSS Modeler to construct this model?
- How much the choice of preprocessing steps contributes to its performance?
- Compare the performance of the model with that of other models constructed.

Results

- What is/are the model(s) with the highest performance for each objective of each data set?
- Discuss how well a data mining technique worked on these datasets. What combination of parameters yielded particularly good results?

- Overall project conclusion: strength & weaknesses
- Make a list of tasks that will work for any dataset and another list of tasks particular to given dataset