CSX-425 DATA MINING AND WAREHOUSING LAB

LABORATORY MANUAL

FOR VII SEMESTER B. Tech CSE

NIT JALANDHAR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DR B R AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY

JALANDHAR

Lab Manual

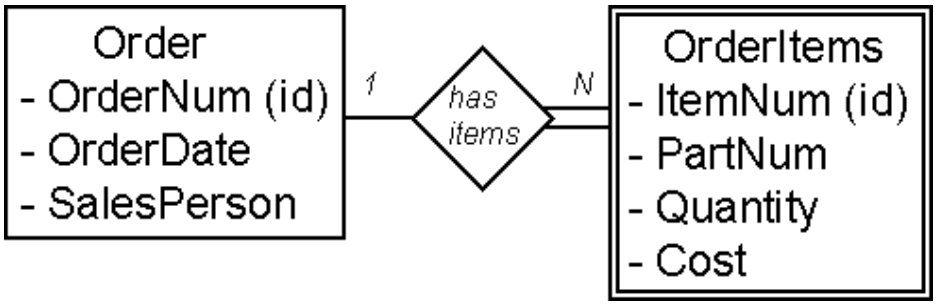
Assignment number	Topic	Date
1	Building a Database Design using ER Modeling and Normalization Techniques	
2	Implementation of functions ,Procedures, Triggers and Cursors	
3	Load Data from heterogenous sources including text files into a predefined warehouse schema.	
4	Design a data mart for a bank to store the credit history of customers in a bank .Use this credit profiling to process future loan applications.	
5	Feature Selection and Variable Filtering (for very large data sets)	
6	Association Mining in large data sets	
7	Interactive Drill-Down, Roll up, Slice and Dice operations	
8	Generalized EM & k-Means Cluster Analysis	
9	Generalized Additive Models (GAM)	
10	General Classification and Regression Trees (GTrees)	
11	General CHAID (Chi-square Automatic Interaction Detection) Models	
12	Interactive Classification and Regression Trees	
13	Goodness of Fit Computations	

LAB-1

Building a Database Design using ER Modeling and Normalization Techniques

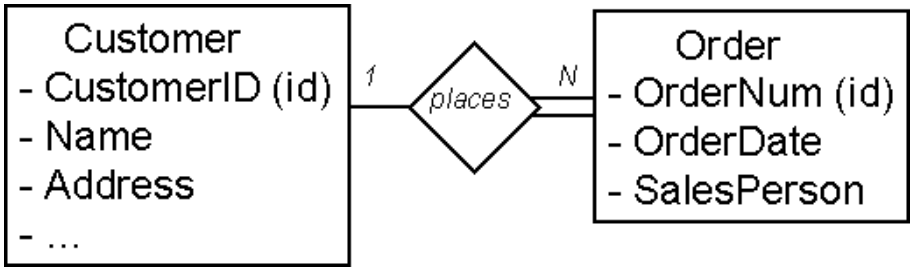
ER diagram:

Chen Notation



- ORDER (OrderNum (key), OrderDate, SalesPerson)
  ORDERITEMS (OrderNum (key)(fk) , ItemNum (key), PartNum, Quantity, Cost)
  - In the above example, in the ORDERITEMS Relation: OrderNum is the Foreign Key and OrderNum plus ItemNum is the Composite Key.

Chen Notation



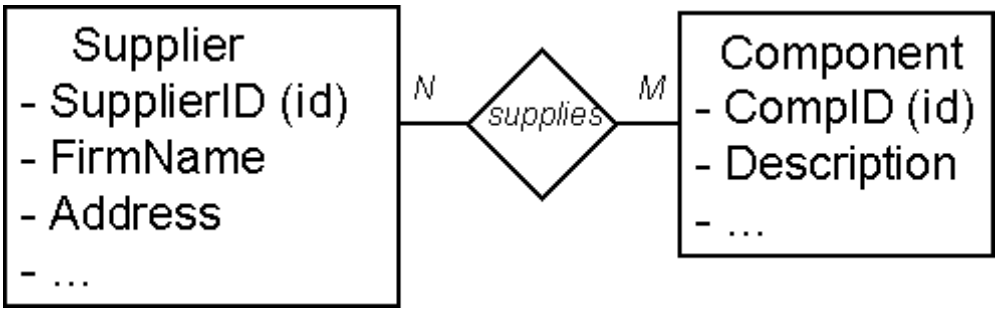
In the ORDER Relation: OrderNum is the Key.

Representing Relationships

- 1:1 Relationships. The key of one relation is stored in the second relation. Look at example queries to determine which key is queried most often.
- 1:N Relationships.
  Parent - Relation on the "1" side.
  Child - Relation on the "Many" side.
- Represent each Entity as a relation.
  Copy the key of the parent into the child relation.
- CUSTOMER (CustomerID (key), Name, Address, ...)
  ORDER (OrderNum (key), OrderDate, SalesPerson, CustomerID (fk))

- M:N Relationships. Many to Many relationships can not be directly implemented in relations.
- Solution: Introduce a third Intersection relation and copy keys from original two relations.

Chen Notation



- SUPPLIER (SupplierID (key), FirmName, Address, ...) COMPONENT (CompID (key), Description, ...) SUPPLIER_COMPONENT (SupplierID (key), CompID (key))
- Note that this can also be shown in the ER diagram. Also, look for potential added attributes in the intersection relation.

CONCEPTS OF NORMALIZATION

Checking Normalization of a database table (Third normal form).

Procedure:

Consider a book database table as given below.

Author Last Name	Author First Name	Book Title	Subject	Collection or Library	Building
Berdahl	Robert	The Politics of the Prussian Nobility	History	PCL General Stacks	Perry- Castañeda Library
Yudof	Mark	Child Abuse and Neglect	Legal Procedures	Law Library	Townes Hall
Harmon	Glynn	Human Memory and Knowledge	Cognitive Psychology	PCL General Stacks	Perry- Castañeda Library
Graves	Robert	The Golden Fleece	Greek Literature	Classics Library	Waggener Hall
Miksa	Francis	Charles Ammi Cutter	Library Biography	Library and Information Science Collection	Perry- Castañeda Library
Hunter	David	Music Publishing and Collecting	Music Literatur e	Fine Arts Library	Fine Arts Building
Graves	Robert	English and Scottish Ballads	Folksong	PCL General Stacks	Perry- Castañeda Library

By examining the table, we can infer that books dealing with history, cognitive psychology, and folksong are assigned to the PCL General Stacks collection; that books dealing with legal procedures are assigned to the Law Library; that books dealing with Greek literature are assigned to the Classics Library; that books dealing with library biography are assigned to the Library and Information Science Collection (LISC);and that books dealing with music literature are assigned to the Fine Arts Library.

Moreover, we can infer that the PCL General Stacks collection and the LISC are both housed in the Perry-Castañeda Library (PCL) building; that the Classics Library is housed in Waggener Hall; and that the Law Library and Fine Arts Library are housed, respectively, in Townes Hall and the Fine Arts Building.

Thus we can see that a transitive dependency exists in the above table : any book that deals with history, cognitive psychology, or library biography will be physically housed in the PCL building (unless it is temporarily checked out to a borrower); any book dealing with legal procedures will be housed in Townes Hall; and so on. In short, if we know what subject a book deals with, we also know not only what library or collection it will be assigned to but also what building it is physically housed in.

A problem with transitive dependency is that, there is duplicated information: from three different rows we can see that the PCL General Stacks are in the PCL building. For another thing, we have possible deletion anomalies: if the Yudof book were lost and its row removed from table, we would lose the information that books on legal procedures are assigned to the Law Library and also the information the Law Library is in Townes Hall. As a third problem, we have possible insertion anomalies: if we wanted to add a chemistry book to the table, we would find that the above table nowhere contains the fact that the Chemistry Library is in Robert A.Welch Hall. As a fourth problem, we have the chance of making errors in updating: a careless data-entry clerk might add a book to the LISC but mistakenly enter Townes Hall in the building column.

To solve this problem decompose the above table into three different tables as follows

Table A

Author Last Name	Author First Name	Book Title
Berdahl	Robert	The Politics of the Prussian Nobility Yudof
	Mark	Child Abuse and Neglect
Harmon	Glynn	Human Memory and Knowledge Graves
	Robert	The Golden Fleece
Miksa	Francis	Charles Ammi Cutter

Table B

Book Title	Subject
The Politics of the Prussian Nobility	History
Child Abuse and Neglect	Legal Procedures Human
Memory and Knowledge	Cognitive Psychology The
Golden Fleece	Greek Literature
Charles Ammi Cutter	Library Biography
Music Publishing and Collecting	Music Literature English
and Scottish Ballads	Folksong

Table C

Subject	Collection or Library
History	PCL General Stacks
Legal Procedures	Law Library Cognitive
Psychology	PCL General Stacks Greek
Literature	Classics Library
Library Biography	Library and Information Science Collection Music
Literature	Fine Arts Library
Folksong	PCL General Stacks

Table D

Collection or Library	Building
PCL General Stacks	Perry-Castañeda Library
Law Library	Townes Hall
Classics Library	Waggener Hall
Library and Information Science Collection	Perry-Castañeda Library
Fine Arts Library	Fine Arts Building

LAB-2

Implementation of functions ,Procedures, Triggers and Cursors

PROCEDURES
Procedures are simply named PL/SQL blocks. They are created and owned by a particular schema. Like the DML functions, right to execute a procedure can be granted / revoked.

Syntax for creating a procedure:

CREATE OR REPLACE PROCEDURE <procedure name> (<parameter1 name> <mode> <data type>, <parameter1 name> <mode> <data type>, …) IS
<Variable declarations>
BEGIN
Executable Commands
EXCEPTION
Exception handlers
END.

OR REPLACE - This option re-creates the procedure, maintaining the privileges previously granted.

Parameter list - If a procedure contains more than one parameter, commas should be used to separate them. You cannot specify a constraint on the data type.
For example, it is **illegal** to do the following:

CREATE OR REPLACE PROCEDURE pro_sample(emp_ssn NUMBER(2), …..)

Instead, you should do this:

CREATE OR REPLACE PROCEDURE pro_sample(emp_ssn NUMBER, …..)

Syntax for executing a procedure:

SQL> EXECUTE <procedure name> or EXEC <procedure name>

Let's look at the following unnamed block:

```
SET SERVEROUTPUT ON
DECLARE
   temp_emp_sal NUMBER(10,2);
BEGIN
   SELECT emp_salary
   INTO temp_emp_sal
   FROM employee
   WHERE emp_ssn =  '999666666';
   IF temp_emp_sal > 4000 THEN
      DBMS_OUTPUT.PUT_LINE('Salary > 4000');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Salary < 4000');
   END IF;
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      DBMS_OUTPUT.PUT_LINE('Employee not found ');
END;
/
```

FUNCTIONS

A function is similar to a procedure, except it returns only one value. A function can accept zero to many parameters, and it must return one and only one value. The data type of the return value must be declared in the header of the function.

Syntax for creating a function:

```
CREATE OR REPLACE FUNCTION <function name> (<parameter1 name> <mode> <data type>, <parameter1 name>
<mode> <data type>,…)
        RETURN <function return value data type> IS
          <Variable declarations>
BEGIN
          Executable Commands
          RETURN (return value);
          …
EXCEPTION
          Exception handlers
END.
```

It is not necessary for the RETURN statement to appear in the last line of the execution section, and there may also be more than one RETURN statement. Like procedure, OR REPLACE is optional when creating a function.

An example follows:

```
-- func1.sql
CREATE OR REPLACE FUNCTION myfunc1
   RETURN NUMBER
IS
   ret_sal NUMBER(10,2);
BEGIN
   SELECT emp_salary INTO ret_sal
   FROM employee
   WHERE emp_ssn = '999666666';
   RETURN (ret_sal);
END myfunc1;
/
```

The output of the above example is:

```
SQL> @func1

Function created.

SQL> var par_sal varchar2(20);
SQL> EXECUTE :par_sal := myfunc2;

PL/SQL procedure successfully completed.

SQL> print par_sal;

par_sal
----------
   55000
```

DATABASE TRIGGERS

A database trigger is a stored PL/SQL program unit associated with a specific database table. ORACLE executes (fires) a database trigger automatically when a given SQL operation (like INSERT, UPDATE or DELETE) affects the table. Unlike a procedure, or a function, which must be invoked explicitly, database triggers are invoked implicitly.

The syntax for creating a trigger (the reserved words and phrases surrounded by brackets are optional):

```
CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE|AFTER} triggering_event ON table_name
[FOR EACH ROW]
[WHEN condition]
DECLARE
        Declaration statements
BEGIN
        Executable statements
EXCEPTION
        Exception-handling statements
END;
```

LAB-3

Load Data from heterogenous sources including text files into a predefined warehouse schema

Data Load

Data Load is the process that involves taking the transformed data and loading it where the users can access it. If the architecture contains a staging database, then loading is a two step process –

- Load data to the transformed data to the Staging Database.
- Load the data in the staging database to the warehouse/mart.

If the the staging area is a file systems, then we directly load the data to the warehouse/mart. During the load we prevent end users to access the warehouse/mart tables on which the load is happening. The reason for this is to avoid

- Incomplete data analysis – While the load is in progress, the fact and dimension tables do not have the complete data. If the users perform their analysis on the partially loaded data the results would be incomplete/inaccurate.
- Query Performance issue – If a user fires a select query on the dimension and fact tables for required for his analysis and during that an Insert or an Update operation is already happening. This would degrade the performance of both the Select query as well as the Insert/Update queries.

Data Load Operations

The operations involved in loading data to the warehouse/mart are as follows

- Insert Operation – One of the basic operation is to insert or update data in the tables. In DWH, we can perform insert for new data or insert newer versions of dimension data (SCD). We use an INSERT statement to perform this.
- Update Operation – An operation used to overwrite values already present the tables. In DWH, we can perform Update for data that has changed. We use an UPDATE statement to perform this.
- Upsert Operation – This is a special operation, where if a row is already present then we Update all the contents of the row else we inset it into the table.
- Bulk Load Utility – The above mentioned Insert, Update and Upsert operations are performed one row at a time. While dealing with large amount of data, processing one row at a time can become a performance bottleneck. Instead we use database bulk loading utilities. These take the input dataset (flat file) and load the contents of the file into the table.

The Bulk loader utility takes two datasets as inputs, one containing the data to be loaded and the other containing control information. The control dataset names table and its columns where the data has to be loaded, it also instructs the database on how to load the data. The Load utility creates a reject file for records that had non-unique values for the primary key of the table.

There are three modes the load operation works in, namely

- APPEND – Here the rows are appended to the table.
- INSERT – Here the table must be empty and the data from the input dataset is loaded into the table.
- REPLACE – It empties the table and loads the data from the file.

We generally use the APPEND mode. Care must be taken when the load utility is run on a table, there should be no other SQL fired on the table.

- Referential Integrity Constraint – The referential integrity constraint checks if a value for a foreign key column is present in the parent table from which the foreign key is derived. This constraint is applied when we insert new rows or update the foreign key columns. While inserting or loading large amount of data, this constraint can pose a performance bottleneck. Hence we disable the Foreign Key constraint on tables dealing with large amounts of data, especially fact tables. We need to make sure that the purpose of referential integrity is maintained by ETL.
- Indexes – Indexes defined on tables help us speed up fetch data from tables. They could pose a performance issue when loading data to the table, especially while dealing with large amount of data (fact tables). We generally drop indexes on the table while inserting/updating/loading large amount of data because every time a row is inserted or the value of an indexed column has changed, the index will be recomputed. Instead we drop

the indexes on the table and rebuild it whence the load operation is over. This can significantly improve the performance of the load operation.

Types of ETL Load

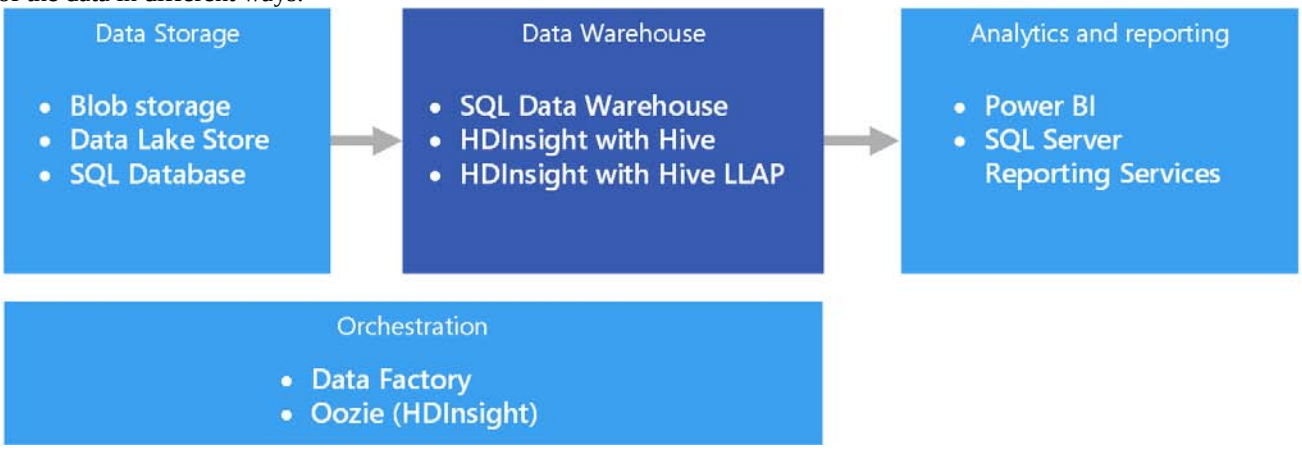
There are two types of ETL load's namely –

- History Load – The Data Warehouse/Mart is expected to house historical data. Based on the duration for how long the end users want to perform the analysis, we keep the data for that long. In other words, we would have users want to compare the Stores Monthly sales and compare it with the monthly sales of for the last 3 years. Here we would have to keep at least 3 years of data so that end users can perform their analysis. When we build the and implement data warehouse/mart, it is empty. We would not want to start building the history from the day it is implemented. In this case the end user would have to wait for 3 years from the day of implementation to perform this particular analysis. Hence we identify the source where we can find the history data, and perform a one time ETL to extract the required history data and load it to the warehouse.
- Incremental Load – Incremental load is the periodic load to keep the data warehouse updated with the most recent transactional data. This is an on going process that continues till the life of the warehouse/mart. The periodicity of the incremental loads is dependent on the availability time of the source data. we can have incremental loads on a
    - Daily basis.
    - Weekly basis.
    - Monthly basis.
    - Quarterly basis.
    - Yearly basis.

LAB-4

Design a data mart for a bank to store the credit history of customers in a bank .Use this credit profiling to process future loan applications.

A data warehouse is a central, organizational, relational repository of integrated data from one or more disparate sources, across many or all subject areas. Data warehouses store current and historical data and are used for reporting and analysis of the data in different ways.



To move data into a data warehouse, it is extracted on a periodic basis from various sources that contain important business information. As the data is moved, it can be formatted, cleaned, validated, summarized, and reorganized. Alternately, the data can be stored in the lowest level of detail, with aggregated views provided in the warehouse for reporting. In either case, the data warehouse becomes a permanent storage space for data used for reporting, analysis, and forming important business decisions using business intelligence (BI) tools.

Data marts and operational data stores

Managing data at scale is complex, and it is becoming less common to have a single data warehouse that represents all data across the entire enterprise. Instead, organizations create smaller, more focused data warehouses, called *data marts*, that expose the desired data for analytics purposes. An orchestration process populates the data marts from data maintained in an operational data store. The operational data store acts as an intermediary between the source transactional system and the data mart. Data managed by the operational data store is a cleaned version of the data present in the source transactional system, and is typically a subset of the historical data that is maintained by the data warehouse or data mart.

Choose a data warehouse when you need to turn massive amounts of data from operational systems into a format that is easy to understand, current, and accurate. Data warehouses do not need to follow the same terse data structure you may be using in your operational/OLTP databases. You can use column names that make sense to business users and analysts, restructure the schema to simplify data relationships, and consolidate several tables into one. These steps help guide users who need to create ad hoc reports, or create reports and analyze the data in BI systems, without the help of a database administrator (DBA) or data developer.

Consider using a data warehouse when you need to keep historical data separate from the source transaction systems for performance reasons. Data warehouses make it easy to access historical data from multiple locations, by providing a centralized location using common formats, common keys, common data models, and common access methods.

Data warehouses are optimized for read access, resulting in faster report generation compared to running reports against the source transaction system. In addition, data warehouses provide the following benefits:

- All historical data from multiple sources can be stored and accessed from a data warehouse as the single source of truth.
- You can improve data quality by cleaning up data as it is imported into the data warehouse, providing more accurate data as well as providing consistent codes and descriptions.
- Reporting tools do not compete with the transactional source systems for query processing cycles. A data warehouse allows the transactional system to focus predominantly on handling writes, while the data warehouse satisfies the majority of read requests.
- A data warehouse can help consolidate data from different software.
- Data mining tools can help you find hidden patterns using automatic methodologies against data stored in your warehouse.
- Data warehouses make it easier to provide secure access to authorized users, while restricting access to others. There is no need to grant business users access to the source data, thereby removing a potential attack vector against one or more production transaction systems.
- Data warehouses make it easier to create business intelligence solutions on top of the data, such as OLAP cubes.

LAB-5

Feature Selection and Variable Filtering (for very large data sets)

Machine learning works on a simple rule – if you put garbage in, you will only get garbage to come out. By garbage here, I mean noise in data.
This becomes even more important when the number of features are very large. You need not use every feature at your disposal for creating an algorithm. You can assist your algorithm by feeding in only those features that are really important. I have myself witnessed feature subsets giving better results than complete set of feature for the same algorithm. Or as Rohan Rao puts it – "Sometimes, less is better!"
Not only in the competitions but this can be very useful in industrial applications as well. You not only reduce the training time and the evaluation time, you also have less things to worry about!

Top reasons to use feature selection are:

- It enables the machine learning algorithm to train faster.
- It reduces the complexity of a model and makes it easier to interpret.
- It improves the accuracy of a model if the right subset is chosen.
- It reduces overfitting.

Filter Methods

Filter methods are generally used as a preprocessing step. The selection of features is independent of any machine learning algorithms. Instead, features are selected on the basis of their scores in various statistical tests for their correlation with the outcome variable. The correlation is a subjective term here. For basic guidance, you can refer to the following table for defining correlation co-efficients.

Feature\Response	Continuous	Categorical
Continuous	Pearson's Correlation	LDA
Categorical	Anova	Chi-Square

- **Pearson's Correlation:** It is used as a measure for quantifying linear dependence between two continuous variables X and Y. Its value varies from -1 to +1. Pearson's correlation is given as:

$$\rho_{X,Y} = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y}$$

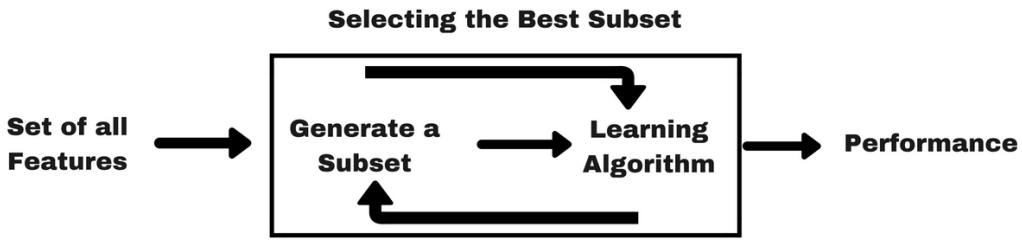
- **LDA:** Linear discriminant analysis is used to find a linear combination of features that characterizes or separates two or more classes (or levels) of a categorical variable.

- **ANOVA:** ANOVA stands for Analysis of variance. It is similar to LDA except for the fact that it is operated using one or more categorical independent features and one continuous dependent feature. It provides a statistical test of whether the means of several groups are equal or not.

- **Chi-Square:** It is a is a statistical test applied to the groups of categorical features to evaluate the likelihood of correlation or association between them using their frequency distribution.

One thing that should be kept in mind is that filter methods do not remove multicollinearity. So, you must deal with multicollinearity of features as well before training models for your data.

Wrapper Methods



In wrapper methods, we try to use a subset of features and train a model using them. Based on the inferences that we draw from the previous model, we decide to add or remove features from your subset. The problem is essentially reduced to a search problem. These methods are usually computationally very expensive.

Some common examples of wrapper methods are forward feature selection, backward feature elimination, recursive feature elimination, etc.

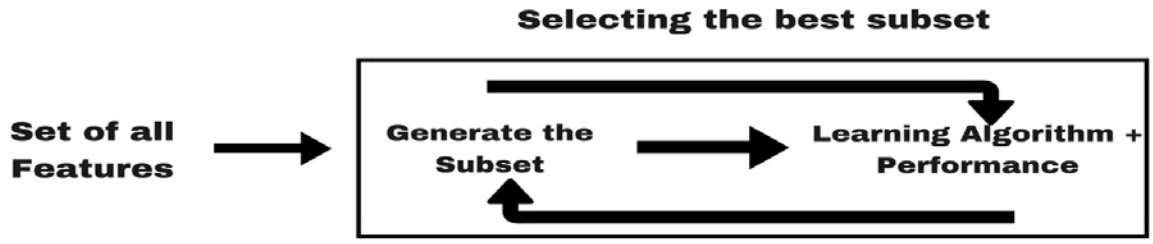
- **Forward Selection:** Forward selection is an iterative method in which we start with having no feature in the model. In each iteration, we keep adding the feature which best improves our model till an addition of a new variable does not improve the performance of the model.
- **Backward Elimination:** In backward elimination, we start with all the features and removes the least significant feature at each iteration which improves the performance of the model. We repeat this until no improvement is observed on removal of features.
- **Recursive Feature elimination:** It is a greedy optimization algorithm which aims to find the best performing feature subset. It repeatedly creates models and keeps aside the best or the worst performing feature at each iteration. It constructs the next model with the left features until all the features are exhausted. It then ranks the features based on the order of their elimination.

One of the best ways for implementing feature selection with wrapper methods is to use Boruta package that finds the importance of a feature by creating shadow features.

It works in the following steps:

1. Firstly, it adds randomness to the given data set by creating shuffled copies of all features (which are called shadow features).
2. Then, it trains a random forest classifier on the extended data set and applies a feature importance measure (the default is Mean Decrease Accuracy) to evaluate the importance of each feature where higher means more important.
3. At every iteration, it checks whether a real feature has a higher importance than the best of its shadow features (i.e. whether the feature has a higher Z-score than the maximum Z-score of its shadow features) and constantly removes features which are deemed highly unimportant.
4. Finally, the algorithm stops either when all features get confirmed or rejected or it reaches a specified limit of random forest runs.

Embedded Methods



Embedded methods combine the qualities' of filter and wrapper methods. It's implemented by algorithms that have their own built-in feature selection methods.

Some of the most popular examples of these methods are LASSO and RIDGE regression which have inbuilt penalization functions to reduce overfitting.

- Lasso regression performs L1 regularization which adds penalty equivalent to absolute value of the magnitude of coefficients.
- Ridge regression performs L2 regularization which adds penalty equivalent to square of the magnitude of coefficients.

Example : The example below uses the chi squared (chi^2) statistical test for non-negative features to select 4 of the best features from the Pima Indians onset of diabetes dataset.

```
# Feature Extraction with Univariate Statistical Tests (Chi-squared for classification)
import pandas
    import numpy
    from sklearn.feature_selection import SelectKBest
    from sklearn.feature_selection import chi2
    # load data
    url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-
    diabetes.data.csv"
    names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
    dataframe = pandas.read_csv(url, names=names)
    array = dataframe.values
    X = array[:,0:8]
    Y = array[:,8]
    # feature extraction
    test = SelectKBest(score_func=chi2, k=4)
    fit = test.fit(X, Y)
    # summarize scores
    numpy.set_printoptions(precision=3)
    print(fit.scores_)
    features = fit.transform(X)
    # summarize selected features
    print(features[0:5,:])
```

You can see the scores for each attribute and the 4 attributes chosen (those with the highest scores): plas, test, mass and age.

```
[  111.52   1411.887    17.605    53.108  2175.565   127.669     5.393
181.304]
[[ 148.     0.    33.6   50. ]
[  85.     0.    26.6   31. ]
[ 183.     0.    23.3   32. ]
[  89.    94.    28.1   21. ]
[ 137.   168.    43.1   33. ]]
```

LAB-6

Association Mining in large data sets

Association rule mining is to find out association rules that satisfy the predefined minimum support and confidence from a given database. The problem is usually decomposed into two sub problems.

- Find those item sets whose occurrences exceed a predefined threshold in the database; those item sets are called frequent or large item sets.
- Generate association rules from those large item sets with the constraints of minimal confidence.

Suppose one of the large item sets is $L_k = \{I_1,I_2,...,I_k\}$; association rules with this item sets are generated in the following way: the first rule is $\{I_1,I_2,...,I_{k-1}\} => \{I_k\}$. By checking the confidence this rule can be determined as interesting or not. Then, other rules are generated by deleting the last items in the antecedent and inserting it to the consequent, further the confidences of the new rules are checked to determine the interestingness of them. This process iterates until the antecedent becomes empty.

Since the second sub problem is quite straight forward, most of the research focuses on the first sub problem. The Apriori algorithm finds the frequent sets *L* in Database *D*.

- Find frequent set L_{k-1}.
- Join Step
  - Ck is generated by joining Lk − 1with itself
- Prune Step.
  - Any (k − 1) -itemset that is not frequent cannot be a subset of a frequent k- itemset, hence should be removed.

where

- (Ck: Candidate itemset of size k)
- (Lk: frequent itemset of size k)
-

Input :

A large supermarket tracks sales data by SKU(Stoke Keeping Unit) (item), and thus is able to know what items are typically purchased together. Apriori is a moderately efficient way to build a list of frequent purchased item pairs from this data. Let the database of transactions consist of the sets {1,2,3,4}, {2,3,4}, {2,3}, {1,2,4}, {1,2,3,4}, **and {2,4}.**

Output

Each number corresponds to a product such as "butter" or "water". The first step of Apriori to count up the frequencies, called the supports, of each member item separately:

Item	Support
1	3
2	6
3	4
4	5

We can define a minimum support level to qualify as "frequent," which depends on the context. For this case, let min support = 3. Therefore, all are frequent. The next step is to generate a list of all 2-pairs of the frequent items. Had any of the above items not been frequent, they wouldn't have been included as a possible member of possible 2-item pairs. In this way, Apriori *prunes* the tree of all possible sets..

Item	Support
{1,2}	3
{1,3}	2
{1,4}	3
{2,3}	4
{2,4}	5
{3,4}	3

This is counting up the occurrences of each of those pairs in the database. Since minsup=3, we don't need to generate 3-sets involving {1,3}. This is because since they're not frequent, no supersets of them can possibly be frequent. Keep going:

Item	Support
{1,2,4}	3
{2,3,4}	3

LAB-7

Interactive Drill-Down, Roll up, Slice and Dice operations

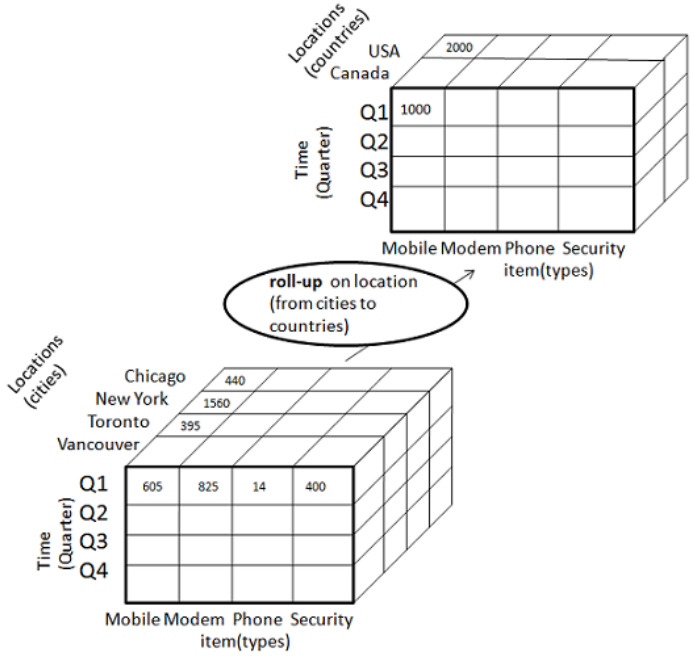
OLAP is an acronym for On Line Analytical Processing.Online Analytical Processing: An OLAP system manages large amount of historical data, provides facilities for summarization and aggregation, and stores and manages information at different levels of granularity. Since OLAP servers are based on multidimensional view of data, we will discuss OLAP operations in multidimensional data.
Here is the list of OLAP operations −

- Roll-up
- Drill-down
- Slice and dice
- Pivot (rotate)

Roll-up
Roll-up performs aggregation on a data cube in any of the following ways−

- By climbing up a concept hierarchy for a dimension
- By dimension reduction

The following diagram illustrates how roll-up works.



- Roll-up is performed by climbing up a concept hierarchy for the dimension location.
- Initially the concept hierarchy was "street < city < province < country".
- On rolling up, the data is aggregated by ascending the location hierarchy from the level of city to the level of country.
- The data is grouped into cities rather than countries.
- When roll-up is performed, one or more dimensions from the data cube are removed.

QUERY SYNTAX:

SELECT …GROUP BY ROLLUP (GROUPING_COLUMN_REFERENCE_LIST);

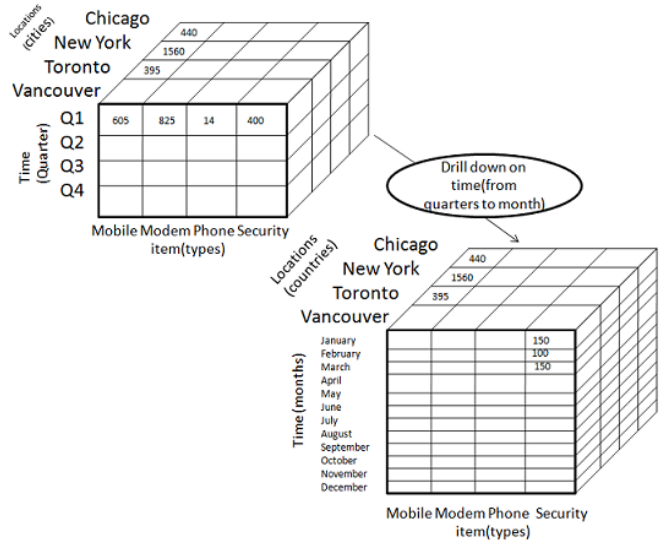
EXAMPLE:

SELECT TIME, LOCATION, PRODUCT ,SUM(REVENUE) AS PROFIT FROM SALES GROUP BY ROLLUP(TIME, LOCATION, PRODUCT);

Drill-down
Drill-down is the reverse operation of roll-up. It is performed by either of the following ways−

- By stepping down a concept hierarchy for a dimension
- By introducing a new dimension.

The following diagram illustrates how drill-down works –



- Drill-down is performed by stepping down a concept hierarchy for the dimension time.
- Initially the concept hierarchy was "day < month < quarter < year."
- On drilling down, the time dimension is descended from the level of quarter to the level of month.
- When drill-down is performed, one or more dimensions from the data cube are added.
- It navigates the data from less detailed data to highly detailed data.

QUERY SYNTAX:

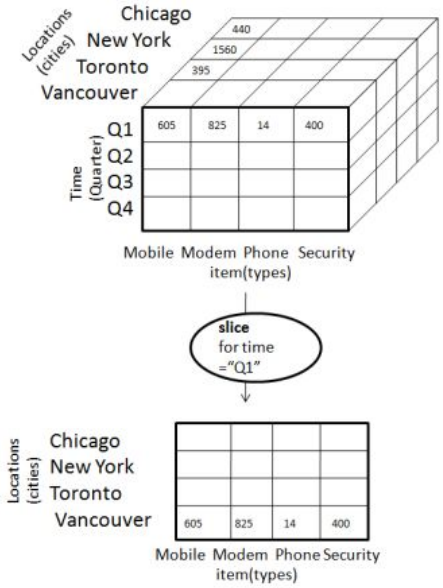
SELECT … GROUP BY ROLLDOWN(COLUMNS);

EXAMPLE:

SELECT TIME, LOCATION, PRODUCT ,SUM(REVENUE) AS PROFIT FROM SALES GROUP BY ROLLDOWN(TIME, LOCATION, PRODUCT);

Slice

The slice operation selects one particular dimension from a given cube and provides a new sub-cube. Consider the following diagram that shows how slice works.



- Here Slice is performed for the dimension "time" using the criterion time = "Q1".
- It will form a new sub-cube by selecting one or more dimensions.

QUERY SYNTAX:

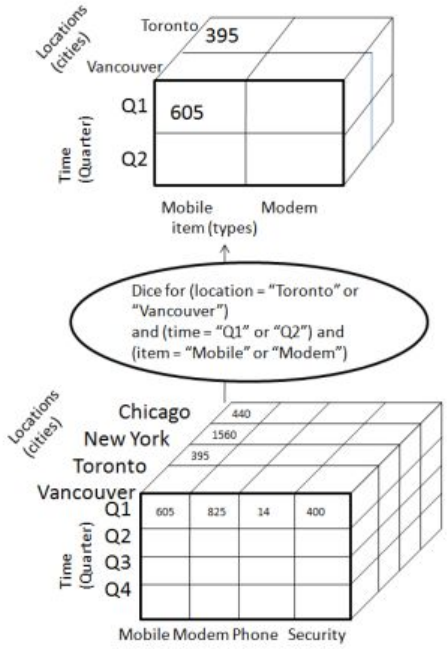
SELECTION CONDITIONS ON SOME ATTRIBUTES USING <WHERE CLAUSE> <GROUP BY> AND AGGREGATION ON SOME ATTRIBUTE

EXAMPLE:

SELECT PRODUCTS, SUM(REVENUE) FROM SALES WHERE PRODUCTS= 'OPV' GROUP BY PRODUCTS ;

Dice

Dice selects two or more dimensions from a given cube and provides a new sub-cube. Consider the following diagram that shows the dice operation.



The dice operation on the cube based on the following selection criteria involves three dimensions.

- (location = "Toronto" or "Vancouver")
- (time = "Q1" or "Q2")
- (item =" Mobile" or "Modem")

QUERY SYNTAX:

SELECTION CONDITIONS ON SOME ATTRIBUTES USING <WHERE CLAUSE> GROUP BY AND AGGREGATION ON SOME ATTRIBUTE

EXAMPLE:

SELECT PRODUCTS, SUM(REVENUE) FROM SALES WHERE PRODUCTS= 'EL' AND LOCATION='EUROPE' GROUP BY PRODUCTS;

LAB-8

Generalized EM & k-Means Cluster Analysis

The EM Algorithm
The EM algorithm is a general method of finding the maximum likelihood estimate of the parameters of underlying distributions from a given data set. We assume all variables are independent of each other and all data are from k joint distributions. The fundamental algorithm iterates between two steps.

1. M-algorithm (Maximization step)

$$u_i = \frac{\sum_j Z_{ij} x_j}{\sum_j Z_{ij}}, \qquad \sigma_i^2 = \frac{\sum_j Z_{ij}(x_j - u_i)^2}{\sum_j Z_{ij}}$$

2. E-algorithm (Expectation step)

$$Z_{ij} = \frac{p(x_j \mid c_i) p(c_i)}{p(x_j)}$$

where u_i is the mean of distribution i. σ_i^2 is the variance of distribution i. Z_{ij} is the estimated weight (probability) of observation j belonging to cluster i. c_i represents the cluster i. $p(x_j)$ represents the probability.

$$likelihood = \sum_j \log \sum_i p(x_j \mid c_i) p(c_i)$$

If the increase value of likelihood is less than the value you specified, stop the iteration and get the final clustering. Also, if the number of the iterations is equal to the maximum number of the iterations, stop the iteration and get the final clustering.

The k-Means Algorithm
The k-means algorithm is a clustering method used to partition a collection of objects into k groups according to the distance measure specified. The fundamental algorithm iterates between two steps.

1. Compute cluster centroids

a. The initial centroids are set up using the method specified by the user. There are three initial center methods: choose N observations to maximize the initial distance, randomly choose N observations, and choose the first N observation. Here, N represents k.

b. After assigning all objects to the nearest centroid, compute the new centroids using all the members assigned to them. For continuous variables, the centroid value is simply the mean of the values of all members assigned to this cluster. For categorical variables, the centroid value is the first mode of all members assigned to it.

2. Assign each object to the nearest centroid

a. The nearest centroid is decided using the specified distance measure method. The continuous variable values are all normalized before the calculation. Note that c represents a centroid and x represents an observation.
i. If i^{th} variable is categorical, (xi - ci) equals 0 if they have the same value, otherwise equals 1.

ii. If i^{th} variable is continuous, xi and ci are normalized first using the minimum and maximum values of this variable.

Euclidean distance:
distance(x, c) = {åi (xi - ci)2 }½

Squared Euclidean distance:
distance(x, c) = åi (xi - ci)2

City-block (Manhattan) distance:

distance(x, c) = åi |xi - ci|

Chebychev distance:
distance(x, c) = Maximum|xi - ci|

b. If all the observations belong to the cluster it belonged to before this iteration, stop the iteration. Also, if the number of iterations equals those specified by the user, stop the iteration. Renew the centroids, and get the final clustering.

Example: Automatic selection of the best number of clusters from the data

Overview. This example is based on a classic example data set reported by Fisher (1936), which is widely referenced in the literature on discriminant function analysis. The Statistica Discriminant Function Analysis module Examples also discuss an analysis using this data set. The data set contains the lengths and widths of sepals and petals of three types of irises (Setosa, Versicol, and Virginic). The purpose of a discriminant function analysis using this data set usually is to learn how you can discriminate among the three types of flowers based on the four measures of width and length of petals and sepals. In this example, it is illustrated how the methods for determining the best number of clusters from the data, available in the Generalized EM and k-Means Cluster Analysis module of Statistica, can be used to identify the different types of iris if you do not know *a priori* that those three types exist.

In other words, suppose you only have the measurements for 150 different flowers of type iris, and are wondering whether these flowers "naturally" fall into a certain number of clusters based on the measurements available to you. In more general terms, suppose you have a set of measurements taken on a large sample of observations, and you are wondering whether any clusters of observations exist in the sample, and if so, how many. Thus, this type of research question will come up in various domains, such as marketing research where one might be interested in clusters of lifestyles or market segmentations; in manufacturing and quality control applications, one might be interested in any clusters or patterns of failures or defects occurring in the final product (e.g., on silicon wafers), etc.

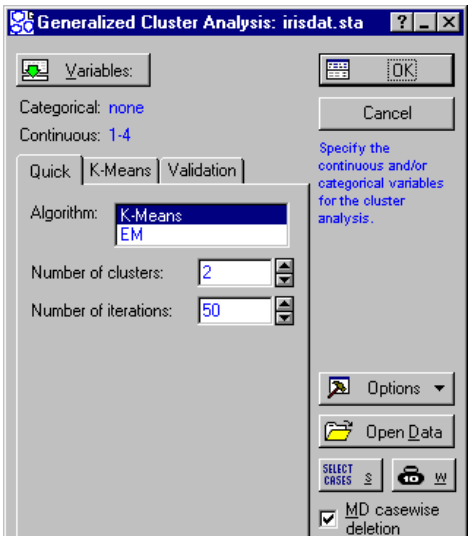
Specifying the analysis

Data file. The data file for this analysis is *Irisdat.sta*. A partial listing of the file is shown below. Open this data file via the *File* - *Open* menu; it is in the /Examples/Datasets directory of *STATISTICA*.

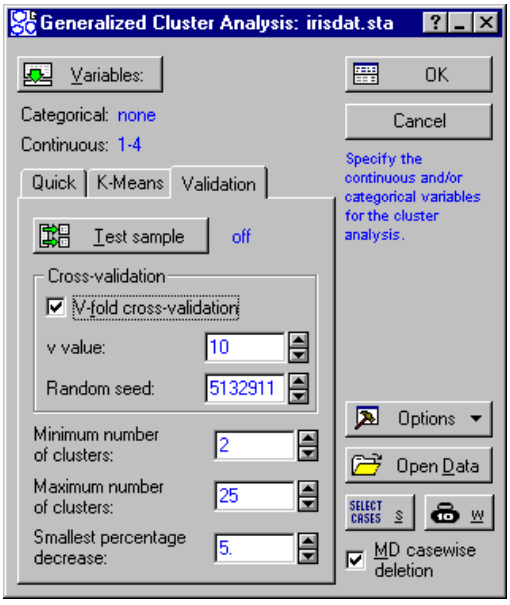
Data: irisdat.sta (5v by 150c)

	Fisher (1936) iris data: length & width of sepals and petals, 3 types of Iris				
	1 SEPALLEN	2 SEPALWID	3 PETALLEN	4 PETALWID	5 IRISTYPE
1	5.0	3.3	1.4	0.2	SETOSA
2	6.4	2.8	5.6	2.2	VIRGINIC
3	6.5	2.8	4.6	1.5	VERSICOL
4	6.7	3.1	5.6	2.4	VIRGINIC
5	6.3	2.8	5.1	1.5	VIRGINIC
6	4.6	3.4	1.4	0.3	SETOSA
7	6.9	3.1	5.1	2.3	VIRGINIC
8	6.2	2.2	4.5	1.5	VERSICOL
9	5.9	3.2	4.8	1.8	VERSICOL
10	4.6	3.6	1.0	0.2	SETOSA

Open the Cluster Analysis (Generalized EM, k-Means & Tree) module by selecting that command from the *Data Mining* menu. In the *Cluster Analysis* dialog box, click the Variables button. In the variable selection dialog box, select as continuous variables in the analysis variables 1 through 4: *Sepallen*, *Sepalwid*, *Petallen*, and *Petalwid*. Note that we are not selecting variable Iristype. Click the *OK* button. Consistent with the stated purpose of this example (see the Overview paragraph above), we will instead pretend that we have no prior knowledge regarding the "true" number of clusters (types of iris) contained in the data set.



Specifying v-fold cross-validation. Click on the *Validation* tab, and select the *V-fold cross-validation* check box. As explained in the Introductory Overview (see also the description of the *Validation* tab), this technique will divide the sample of 150 flowers into v "folds" or randomly selected samples of approximately equal size. *STATISTICA* will then perform repeated cluster analyses for each v-1 samples (i.e., leaving out one sample), and classify (assign to clusters) the observations in the sample that were not used to compute the respective cluster solution. That sample will be treated as a test sample, for which the average distance of observations from their respective (assigned) cluster centers will be recorded. This measure of "misclassification error" or "cost" will be averaged over all v replications of the analysis. *STATISTICA* will continue to perform these computations for increasing numbers of clusters until in successive cluster solutions (with *k* and *k+1* clusters) the percentage decrease in the misclassification error is less than the *Smallest percentage decrease*, specified on the *Validation* tab; at that point, *k* will be taken as the best number of clusters for the data.

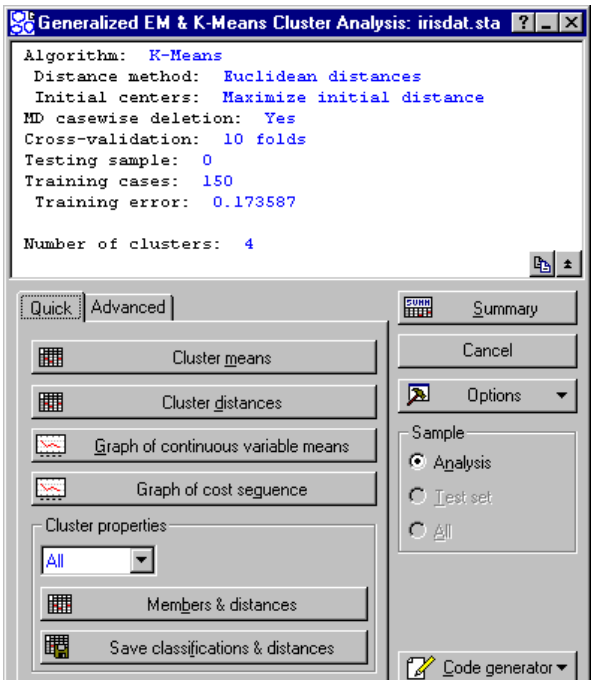


Now, click *OK* to perform the computations and display the *Results* dialog.

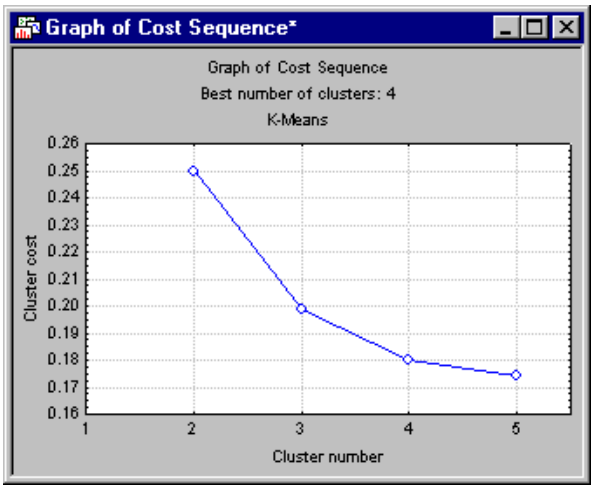
Reviewing Results

Note that, as discussed in the Introductory Overview, the results of your analysis may be different because the initial random assignment of observations (based on the random number seed) is different; you can enter the same random number seed (as shown in the above illustration) to achieve the same results.

In this particular case, the program extracted 4 clusters from the data.



Graph of cost sequence. Let us first look at the *Graph of the cost sequence*. As described in the Introductory Overview as well as the *Results - Quick* tab topic, this graph depicts the error function (average distance of observations in testing samples to the cluster centroids to which the observations were assigned) over the different cluster solutions.



It appears that the error function quickly drops from the 2- to the 3-cluster solution, and then it "flattens" out. Using the same logic as applied to the similar Scree plot computed in *Factor Analysis* (for determining the best number of factors), you could choose either the 3 or the 4-cluster solution for final review. For this example, let's accept the 4-cluster solution selected by the program.

Graph of continuous variable means. Click the *Graph of continuous variable means* button to display a line graph that shows the scaled cluster means for all continuous variables. These means are computed as follows:

$$\bar{x}'_{i,j} = \frac{\bar{x}_{i,j} - x_{i-min}}{x_{i-max} - x_{i-min}}$$

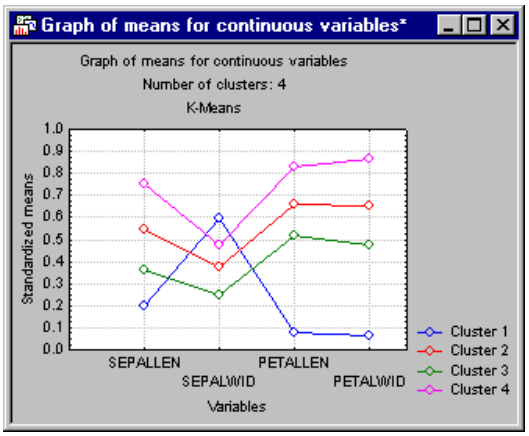
where

$\bar{x}'_{i,j}$ is the transformed (scaled) mean for continuous variable *i* and cluster *j*

$\bar{x}_{i,j}$ is the arithmetic ("unscaled") mean for continuous variable *i* and cluster *j*

x_{i-max}, x_{i-min} are the maximum and minimum observed values for continuous variable *i*

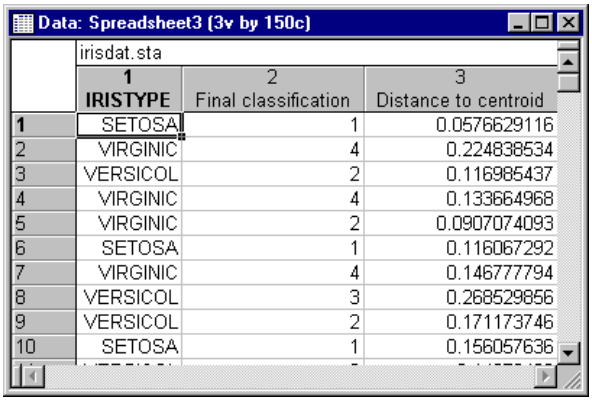
In other words, the plotted values depict the means scaled to the overall ranges of observed values for the respective continuous variables.



It appears that the pattern of means for Cluster 1 is quite distinct from that for the other clusters. You can also click the *Cluster distances* button to verify that the distance of Cluster 1 from all the others is larger than the distances between clusters 2, 3, and 4.

Comparing the final cluster solution with actual iris types. For this example, we deliberately ignored variable *Iristype*, i.e., the previous knowledge we had regarding the true number of clusters in the sample. It is interesting to review how "well" we did with the cluster analysis, in identifying the real iris types. In other research applications, it might be of interest to find appropriate labels for the final clusters by running additional analyses to relate the cluster assignments to other variables of interest. In this case, we will simply compute a crosstabulation table of the cluster assignments by the actual *Iristype* recorded in the *Irisdat.sta* data file.

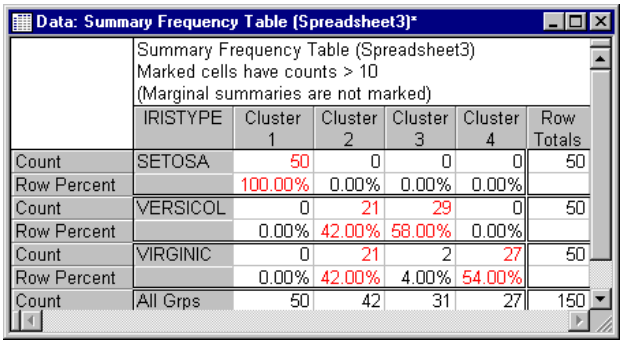
On the *Results* dialog - *Quick* tab, click the *Save classifications & distances* button, and then select *Iristype* as an additional variable to save along with the cluster analysis results (assignments, distances to cluster centers). Part of the resulting data file is shown below.


Data: Spreadsheet3 (3v by 150c)

irisdat.sta

	1 IRISTYPE	2 Final classification	3 Distance to centroid
1	SETOSA	1	0.0576629116
2	VIRGINIC	4	0.224838534
3	VERSICOL	2	0.116985437
4	VIRGINIC	4	0.133664968
5	VIRGINIC	2	0.0907074093
6	SETOSA	1	0.116067292
7	VIRGINIC	4	0.146777794
8	VERSICOL	3	0.268529856
9	VERSICOL	2	0.171173746
10	SETOSA	1	0.156057636

This data file will automatically be created as an input file for subsequent analyses (see also option *Data - Input Spreadsheet*). Select *Basic Statistics/Tables* from the *Statistics* menu, select *Tables and banners* to display the *Crosstabulation Tables* dialog, and then compute a cross tabulation table of variable *Iristype* by *Final classification*.


Data: Summary Frequency Table (Spreadsheet3)*

Summary Frequency Table (Spreadsheet3)
Marked cells have counts > 10
(Marginal summaries are not marked)

	IRISTYPE	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Row Totals
Count	SETOSA	50	0	0	0	50
Row Percent		100.00%	0.00%	0.00%	0.00%	
Count	VERSICOL	0	21	29	0	50
Row Percent		0.00%	42.00%	58.00%	0.00%	
Count	VIRGINIC	0	21	2	27	50
Row Percent		0.00%	42.00%	4.00%	54.00%	
Count	All Grps	50	42	31	27	150

Shown above is the summary crosstabulation table, along with the (row) percentages of observations of each known *Iristype* classified into the respective clusters. As expected, *Cluster 1* (the one that showed the greatest distance from all others) is most distinct. 100% of all flowers of type *Setosa* were correctly classified as belonging to a distinct group or cluster. *Cluster 3* and *Cluster 4* apparently identify the flowers of type *Versicol* and *Virginic* that are easily "classifiable," while *Cluster 2* contains both flowers of type *Versicol* and *Virginic*. It appears that these two types of flowers are not as easily distinguished, a result that is consistent with those that were computed in the Discriminant Function Analysis - Example.

LAB-9

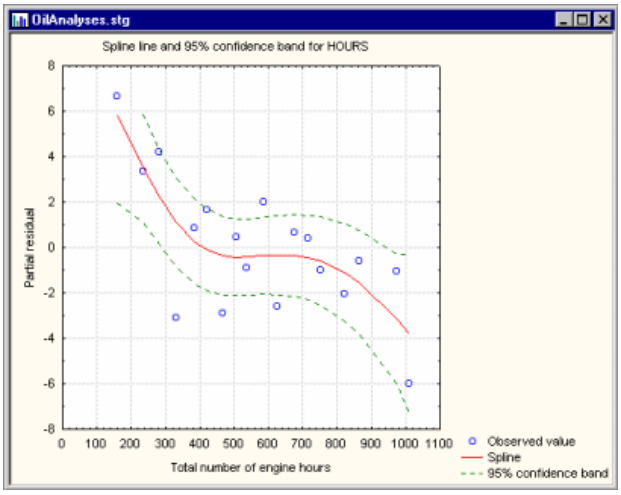
Generalized Additive Models (GAM)

We can combine the notion of additive models with generalized linear models, to derive the notion of generalized additive models, as:
gi(muY) = ∑i(fi(Xi))
In other words, the purpose of generalized additive models is to maximize the quality of prediction of a dependent variable Y from various distributions, by estimating unspecific (non-parametric) functions of the predictor variables which are "connected" to the dependent variable via a link function.

Estimating the non-parametric function of predictors via scatterplot smoothers. A unique aspect of generalized additive models are the non-parametric functions fi of the predictor variables Xi. Specifically, instead of some kind of simple or complex parametric functions, Hastie and Tibshirani (1990) discuss various general scatterplot smoothers that can be applied to the X variable values, with the target criterion to maximize the quality of prediction of the (transformed) Y variable values. One such scatterplot smoother is the cubic smoothing splines smoother, which generally produces a smooth generalization of the relationship between the two variables in the scatterplot. Computational details regarding this smoother can be found in Hastie and Tibshirani (1990; see also Schimek, 2000).

Fitting generalized additive models. Detailed descriptions of how generalized additive models are fit to data can be found in Hastie and Tibshirani (1990), as well as Schimek (2000, p. 300). In general there are two separate iterative operations involved in the algorithm, which are usually labeled the outer and inner loop. The purpose of the outer loop is to maximize the overall fit of the model, by minimizing the overall likelihood of the data given the model (similar to the maximum likelihood estimation procedures as described in, for example, the context of Nonlinear Estimation). The purpose of the inner loop is to refine the scatterplot smoother, which in the implementation of generalized additive models in Statistica is the cubic splines smoother. The smoothing is performed with respect to the partial residuals; i.e., for every predictor k Statistica finds the weighted cubic spline fit that best represents the relationship between variable k and the (partial) residuals computed by removing the effect of all other j predictors (j≠k). The iterative estimation procedure will terminate, when the likelihood of the data given the model cannot be improved.

Interpreting the results. Many of the standard results statistics computed by the Generalized Additive Models module are similar to those customarily reported by linear or nonlinear model fitting procedures. For example, Statistica will compute predicted and residual values for the final model, and display various graphs of the residuals to help the user identify possible outliers, etc. Refer also to the description of the residual statistics computed by the Generalized Linear/Nonlinear Models module for details.
The main result of interest, of course, is how the predictors are related to the dependent variable. Statistica will compute scatterplots showing the smoothed predictor variable values plotted against the partial residuals, i.e., the residuals after removing the effect of all other predictor variables.



This plot allows you to evaluate the nature of the relationship between the predictor with the residualized (adjusted) dependent variable values (see Hastie & Tibshirani, 1990; in particular formula 6.3), and hence the nature of the influence of the respective predictor in the overall model.

Generalized Additive Models Example

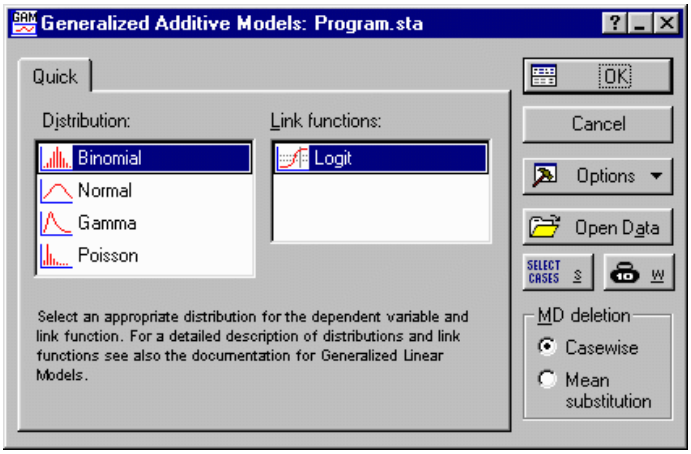
This example is based on a data set described in Neter, Wasserman, and Kutner (1985, page 357; however, note that those authors fit a linear regression model to the data); it is also discussed in the documentation for the Nonlinear Estimation module, in the context of the Quick Logit Regression examples. In this example we will fit a generalized additive logit model to the data, which you can compare with the results computed for a ("simple") logit regression model. Detailed examples of generalized additive logit models and for other distributions and link functions are provided in Hastie and Tibshirani (1990).

Suppose you want to study whether experience helps programmers complete complex programming tasks within a specified amount of time. Twenty-five programmers are selected with different degrees of experience (measured in months). They are then asked to complete a complex programming task within a certain amount of time. The binary dependent variable is the programmers' success or failure in completing the task. These data are recorded in the file Program.sta; shown below is a partial listing of this file.

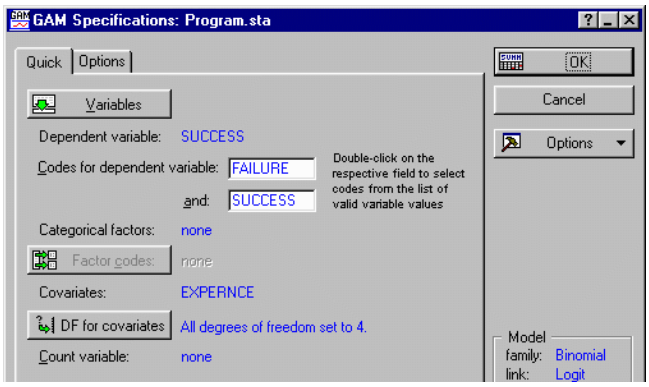
Data: Program.sta (2v by 25c)

	Success in a programing ta	
	1 EXPERNCE	2 SUCCESS
Frank	14	FAILURE
Henry	29	FAILURE
Tom	6	FAILURE
Beth	25	SUCCESS
Susan	18	SUCCESS
Harry	4	FAILURE
Paul	18	FAILURE
Pete	12	FAILURE
Diana	22	SUCCESS
Louise	6	FAILURE
Fred	30	SUCCESS
Hank	11	FAILURE
Steven	30	SUCCESS
Tod	5	FAILURE
Take	20	SUCCESS
Sam	13	FAILURE
Gail	9	FAILURE

Specifying the Analysis. Open the Program.sta data file. From the Data Mining menu, select Generalized Additive Models to display the Generalized Additive Models Startup Panel. Then select the Binomial distribution from the Distribution list; the Logit link function will automatically be selected.

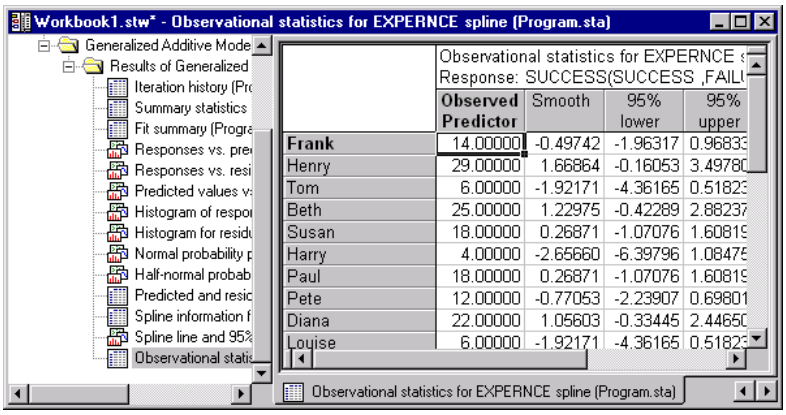


Click OK, to display the GAM Specifications dialog, and click the Variables button to display a standard variable selection dialog. Select the variables for the analysis: Select Success as the dependent variable, and Expernce as the continuous predictor variable (in the third list of the 4-variable lists selection dialog). Click the OK button.



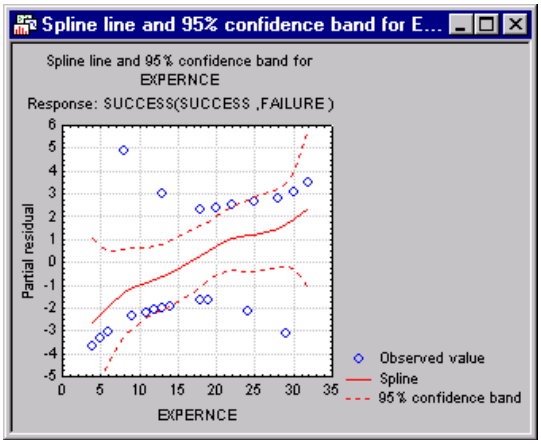
Note that STATISTICA automatically fills in the codes for the binomial dependent variable. During the computations, the value Failure in the dependent variable Success will be interpreted as 0, the value Success will be interpreted as 1. Hence, in the results, the greater the predicted (logit-) value, the greater is the probability of the programmers' success.

Reviewing Results. Click OK on the GAM Specifications dialog to start the computations. A series of results spreadsheets and graphs will be produced.



As you can see, a number of results spreadsheets and graphs are reported to provide a comprehensive picture of the quality of the fit of the model and to aid in the interpretation of results. The interpretation of results from fitting generalized additive models is complex and requires experience (note that these techniques were only developed fairly recently, and there is not a large body of literature and "experience" with these techniques); Hastie and Tibshirani provide detailed discussions on how to interpret the results from these types of analyses, and more importantly, how to use this information to evaluate the appropriateness of the solutions obtained. You can also refer to Schimek (2000) for more recent developments regarding these techniques, and their applications.

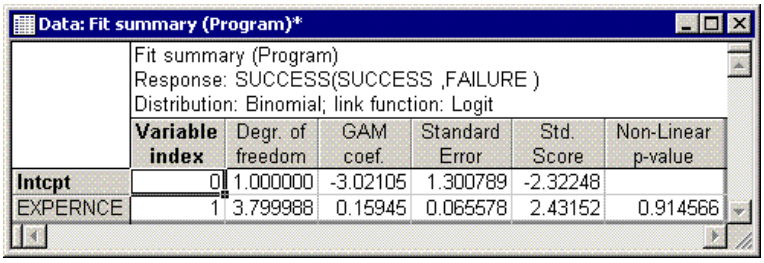
Let's only look at the result that is most characteristic for this method: The plot of the observed predictor values vs. the partial residuals (see also the GAM Introductory Overviews); this plot will also show the cubic spline fit for the final model.



To reiterate, this plot shows the final fitted cubic spline function, along with the observed predictor values, plotted against the partial residuals, i.e., against the residual values for the prediction of the (adjusted) dependent variable, after removing all other effects from the model (see Hastie & Tibshirani, 1990 for computational details; in particular formula 6.3 for the computation of adjusted dependent variable values). In this case, of course, there is only a single effect in the model. As you can see, the greater the experience of a programmer, the more likely is his or her success, as indicated by the monotone increasing cubic spline line.

You can also review the various observational and residual statistics that are computed to identify outliers or any general lack of fit, or groups of cases that are not well represented ("explained") by the model.

Fit Summary. Now display the results spreadsheet labeled Fit summary. As briefly mentioned in the Introductory Overview, one of the important issues to consider when applying generalized additive models is whether the added smoothing - and the parameter that needs to be estimated to find the best cubic spline smoother - are "worth it," i.e., produce a significantly better fit of the model to the data. In this case, judging from the partial residual plot, the relationship between the predictor variables and the partial residuals appears almost linear.



Data: Fit summary (Program)*

Fit summary (Program)
Response: SUCCESS(SUCCESS ,FAILURE)
Distribution: Binomial; link function: Logit

	Variable index	Degr. of freedom	GAM coef.	Standard Error	Std. Score	Non-Linear p-value
Intcpt	0	1.000000	-3.02105	1.300789	-2.32248	
EXPERNCE	1	3.799988	0.15945	0.065578	2.43152	0.914566

Indeed, the Nonlinear p-value in the Fit summary spreadsheet is almost 1; thus, in this case is not clear whether the additional complexity of the additive logistic model is worthwhile.

LAB-10

General Classification and Regression Trees (GTrees)

The Stastica General Classification and Regression Trees module (GC&RT) will build classification and regression trees for predicting continuous dependent variables (regression) and categorical predictor variables (classification). The program supports the classic C&RT algorithm popularized by Breiman et al. (Breiman, Friedman, Olshen, & Stone, 1984; see also Ripley, 1996), and includes various methods for pruning and cross-validation, as well as the powerful v-fold cross-validation methods. In addition, with this program you can specify ANCOVA-like experimental designs (see MANOVA and GLM) with continuous and categorical factor effects and interactions, i.e., to base the computations on design matrices for the predictor variables. A general introduction to tree-classifiers, specifically to the QUEST (Quick, Unbiased, Efficient Statistical Trees) algorithm, is also presented in the context of the Classification Trees Analysis facilities, and much of the following discussion presents the same information, in only a slightly different context. Another, similar type of tree building algorithm is CHAID (Chi-square Automatic Interaction Detector; see Kass, 1980); a full implementation of this algorithm is available in the General CHAID Models module of Stastica.

Classification and Regression Problems
Stastica contains numerous algorithms for predicting continuous variables or categorical variables from a set of continuous predictors and/or categorical factor effects. For example, in GLM (General Linear Models) and GRM (General Regression Models), you can specify a linear combination (design) of continuous predictors and categorical factor effects (e.g., with two-way and three-way interaction effects) to predict a continuous dependent variable. In GDA (General Discriminant Function Analysis), you can specify such designs for predicting categorical variables, i.e., to solve classification problems.

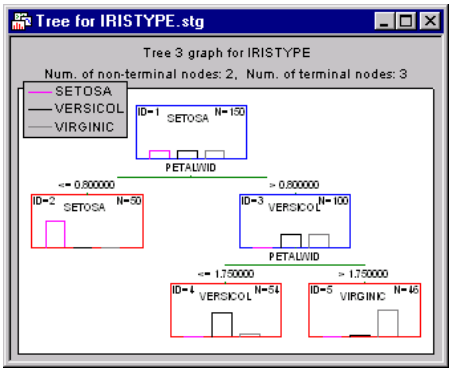
Regression-type problems. Regression-type problems are generally those where one attempts to predict the values of a continuous variable from one or more continuous and/or categorical predictor variables. For example, you may want to predict the selling prices of single family homes (a continuous dependent variable) from various other continuous predictors (e.g., square footage) as well as categorical predictors (e.g., style of home, such as ranch, two-story, etc.; zip code or telephone area code where the property is located, etc.; note that this latter variable would be categorical in nature, even though it would contain numeric values or codes). If you used simple multiple regression, or some general linear model (GLM) to predict the selling prices of single family homes, you would determine a linear equation for these variables that can be used to compute predicted selling prices. STATISTICA contains many different analytic procedures for fitting linear models (GLM, GRM, Regression), various types of nonlinear models (e.g., Generalized Linear/Nonlinear Models (GLZ), Generalized Additive Models (GAM), etc.), or completely custom-defined nonlinear models (see Nonlinear Estimation), where you can type in an arbitrary equation containing parameters to be estimated by the program. The General CHAID Models (GCHAID) module of STATISTICA also analyzes regression-type problems, and produces results that are similar (in nature) to those computed by GC&RT.

Classification-type problems. Classification-type problems are generally those where one attempts to predict values of a categorical dependent variable (class, group membership, etc.) from one or more continuous and/or categorical predictor variables. For example, you may be interested in predicting who will or will not graduate from college, or who will or will not renew a subscription. These would be examples of simple binary classification problems, where the categorical dependent variable can only assume two distinct and mutually exclusive values. In other cases one might be interested in predicting which one of multiple different alternative consumer products (e.g., makes of cars) a person decides to purchase, or which type of failure occurs with different types of engines. In those cases there are multiple categories or classes for the categorical dependent variable. Stastica contains a number of methods for analyzing classification-type problems and to compute predicted classifications, either from simple continuous predictors (e.g., binomial or multinomial logit regression in GLZ), from categorical predictors (e.g., Log-Linear analysis of multi-way frequency tables), or both (e.g., via ANCOVA-like designs in GLZ or GDA). The General CHAID Models module of STATISTICA also analyzes classification-type problems, and produces results that are similar (in nature) to those computed by GC&RT.

Classification and Regression Trees (C&RT)
In most general terms, the purpose of the analyses via tree-building algorithms is to determine a set of if-then logical (split) conditions that permit accurate prediction or classification of cases.

Classification Trees
For example, consider the widely referenced Iris data classification problem introduced by Fisher [1936; see also Discriminant Function Analysis and General Discriminant Analysis (GDA)]. The example data file Irisdat.sta reports the lengths and widths of sepals and petals of three types of irises (Setosa, Versicol, and Virginic). The purpose of the

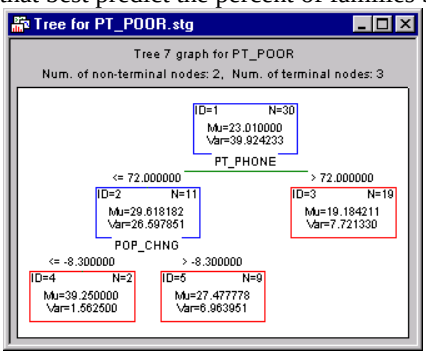
analysis is to learn how one can discriminate between the three types of flowers, based on the four measures of width and length of petals and sepals. Discriminant function analysis will estimate several linear combinations of predictor variables for computing classification scores (or probabilities) that allow the user to determine the predicted classification for each observation (see also the Discriminant Function Analysis Example for a discussion of this type of analysis). A classification tree will determine a set of logical if-then conditions (instead of linear equations) for predicting or classifying cases instead:



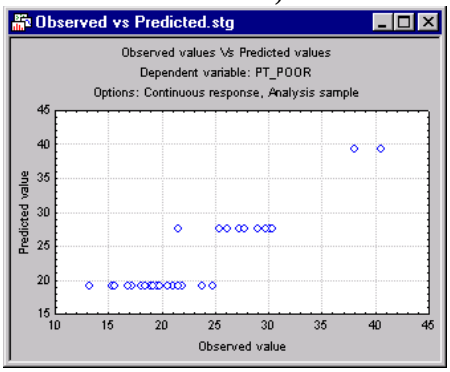
The interpretation of this tree is straightforward: If the petal width is less than or equal to 0.8, the respective flower would be classified as Setosa; if the petal width is greater than 0.8 and less than or equal to 1.75, then the respective flower would be classified as Versicol; else, it belongs to class Virginic.

Regression Trees

The general approach to derive predictions from few simple if-then conditions can be applied to regression problems as well. For example, refer to Example 1 of the Multiple Regression Analysis module (see also Example 2 for GC&RT). Example 1 is based on the data file Poverty.sta, which contains 1960 and 1970 Census figures for a random selection of 30 counties. The research question (for that example) was to determine the correlates of poverty, that is, the variables that best predict the percent of families below the poverty line in a county.



Again, the interpretation of these results is rather straightforward: Counties where the percent of households with a phone is greater than 72% have generally a lower poverty rate. The greatest poverty rate is evident in those counties that show less than (or equal to) 72% of households with a phone, and where the population change (from the 1960 census to the 1970 census) is less than -8.3 (minus 8.3).



LAB-11

General CHAID (Chi-square Automatic Interaction Detection) Models

The acronym CHAID stands for Chi-squared Automatic Interaction Detector. It is one of the oldest tree classification methods originally proposed by Kass (1980). According to Ripley, 1996, the CHAID algorithm is a descendent of THAID developed by Morgan and Messenger, (1973). CHAID will "build" non-binary trees (i.e., trees where more than two branches can attach to a single root or node), based on a relatively simple algorithm that is particularly well suited for the analysis of larger datasets. Also, because the CHAID algorithm will often effectively yield many multi-way frequency tables (e.g., when classifying a categorical response variable with many categories, based on categorical predictors with many classes), it has been particularly popular in marketing research, in the context of market segmentation studies.

Both CHAID and C&RT techniques will construct trees, where each (non-terminal) node identifies a split condition, to yield optimum prediction (of continuous dependent or response variables) or classification (for categorical dependent or response variables). Hence, both types of algorithms can be applied to analyze regression-type problems or classification-type.

Basic Tree-Building Algorithm: CHAID and Exhaustive CHAID
The acronym CHAID stands for Chi-squared Automatic Interaction Detector. This name derives from the basic algorithm that is used to construct (non-binary) trees, which for classification problems (when the dependent variable is categorical in nature) relies on the Chi-square test to determine the best next split at each step; for regression-type problems (continuous dependent variable) the program will actually compute F-tests. Specifically, the algorithm proceeds as follows:

Preparing predictors. The first step is to create categorical predictors out of any continuous predictors by dividing the respective continuous distributions into a number of categories with an approximately equal number of observations. For categorical predictors, the categories (classes) are "naturally" defined.

Merging categories. The next step is to cycle through the predictors to determine for each predictor the pair of (predictor) categories that is least significantly different with respect to the dependent variable; for classification problems (where the dependent variable is categorical as well), it will compute a Chi-square test (Pearson Chi-square); for regression problems (where the dependent variable is continuous), F tests. If the respective test for a given pair of predictor categories is not statistically significant as defined by an alpha-to-merge value, then it will merge the respective predictor categories and repeat this step (i.e., find the next pair of categories, which now may include previously merged categories). If the statistical significance for the respective pair of predictor categories is significant (less than the respective alpha-to-merge value), then (optionally) it will compute a Bonferroni adjusted p-value for the set of categories for the respective predictor.

Selecting the split variable. The next step is to choose the split the predictor variable with the smallest adjusted p-value, i.e., the predictor variable that will yield the most significant split; if the smallest (Bonferroni) adjusted p-value for any predictor is greater than some alpha-to-split value, then no further splits will be performed, and the respective node is a terminal node.

Continue this process until no further splits can be performed (given the alpha-to-merge and alpha-to-split values).

CHAID and Exhaustive CHAID Algorithms. A modification to the basic CHAID algorithm, called Exhaustive CHAID, performs a more thorough merging and testing of predictor variables, and hence requires more computing time. Specifically, the merging of categories continues (without reference to any alpha-to-merge value) until only two categories remain for each predictor. The algorithm then proceeds as described above in the Selecting the split variable step, and selects among the predictors the one that yields the most significant split. For large datasets, and with many continuous predictor variables, this modification of the simpler CHAID algorithm may require significant computing time.

General Computation Issues of CHAID
Reviewing large trees: Unique analysis management tools. A general issue that arises when applying tree classification or regression methods is that the final trees can become very large. In practice, when the input data are complex and, for example, contain many different categories for classification problems, and many possible predictors for performing the classification, then the resulting trees can become very large. This is not so much a computational

problem as it is a problem of presenting the trees in a manner that is easily accessible to the data analyst, or for presentation to the "consumers" of the research.

Analyzing ANCOVA-like designs. The classic CHAID algorithms can accommodate both continuous and categorical predictor. However, in practice, it is not uncommon to combine such variables into analysis of variance/covariance (ANCOVA) like predictor designs with main effects or interaction effects for categorical and continuous predictors. This method of analyzing coded ANCOVA-like designs is relatively new. However, it is easy to see how the use of coded predictor designs expands these powerful classification and regression techniques to the analysis of data from experimental.

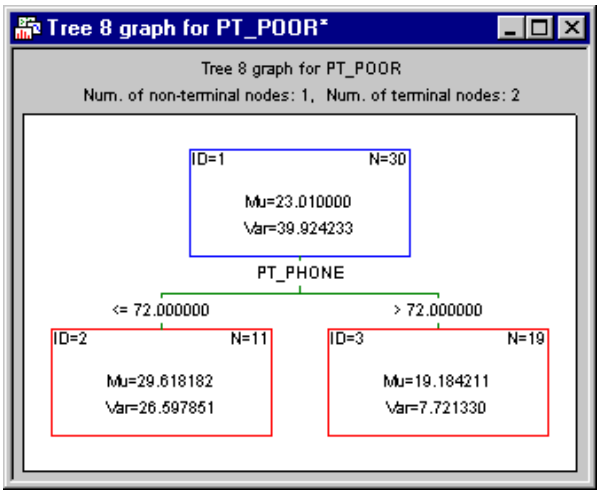
LAB-12

Interactive Classification and Regression Trees

The main purpose of the Interactive Trees (C&RT, CHAID) module is to provide interactivity and complete control over the tree-building process (see the Introductory Overview). Therefore, you may want to review the examples for the General Classification and Regression Trees (GC&RT), General CHAID (GCHAID) Models, and Classification Trees modules, and repeat the analyses described there using the Interactive Trees (C&RT, CHAID) facilities. You can then perform various "what-if" analyses, change particular branches in the results trees, etc., to assess how "unique" the respective solutions are, or how easy or difficult it is to "manually" derive trees of similar predictive validity, but with different split variables. We will demonstrate this process in this example, using the Poverty data set that is described, for example, in Example 2 in General Classification and Regression Trees (GC&RT).

Regression Tree for Predicting Poverty

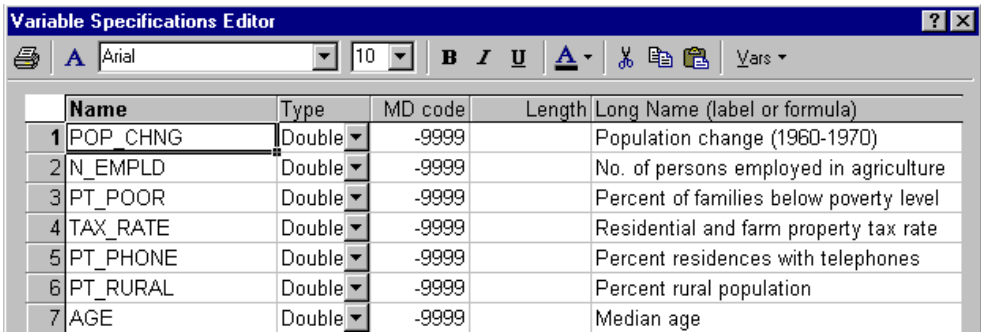
This example is based on a re-analysis of the data presented in Example 1: Standard Regression Analysis for the Multiple Regression module and GC&RT Example 2: Regression Tree for Predicting Poverty. It demonstrates how regression trees can sometimes create very simple and interpretable solutions. In Example 2 of the General Classification and Regression Trees (GC&RT) module, we automatically built the tree shown in the following illustration:



The solution is relatively simple and straightforward. However, we want to build a tree that is even simpler, in particular with respect to the specific cut-off or split values for each predictor. In practice, it is often convenient to use split values that are simple to communicate (e.g., explain to management) and "administer" (e.g., if PT_PHONE < 50% then ..., instead of if PT_PHONE < 72% then...), especially when such simplicity can be achieved with little loss in the quality of the overall predictive model.

Data file. The example is based on the data file Poverty.sta. Open this data file via the File - Open Examples menu; it is in the Datasets folder. The data are based on a comparison of 1960 and 1970 Census figures for a random selection of 30 counties. The names of the counties were entered as case names.

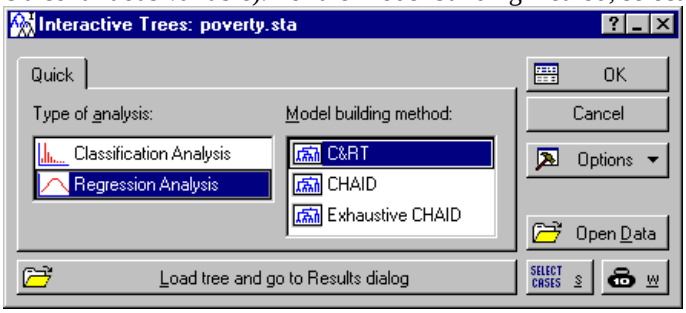
The following information for each variable is displayed in the Variable Specifications Editor (accessible by selecting All Variable Specs from the Data menu).



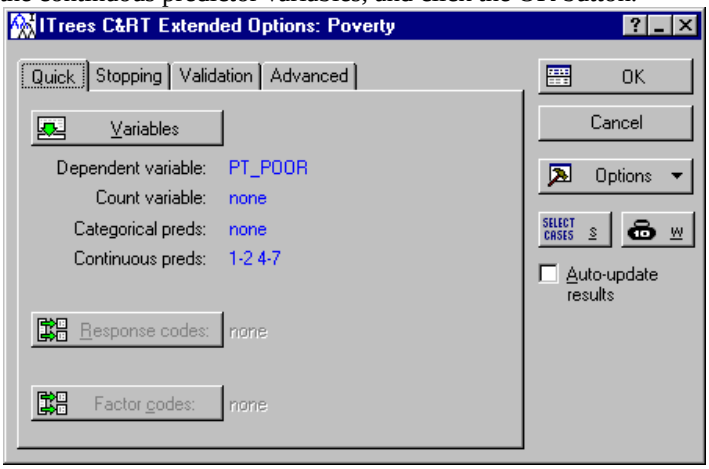
	Name	Type	MD code	Length	Long Name (label or formula)
1	POP_CHNG	Double	-9999		Population change (1960-1970)
2	N_EMPLD	Double	-9999		No. of persons employed in agriculture
3	PT_POOR	Double	-9999		Percent of families below poverty level
4	TAX_RATE	Double	-9999		Residential and farm property tax rate
5	PT_PHONE	Double	-9999		Percent residences with telephones
6	PT_RURAL	Double	-9999		Percent rural population
7	AGE	Double	-9999		Median age

Research question. The purpose of the study is to analyze the correlates of poverty, that is, the variables that best predict the percent of families below the poverty line in a county. Thus, you will treat variable 3 (Pt_Poor) as the dependent or criterion variable, and all other variables as the independent or predictor variables.

Setting up the analysis. Select Interactive Trees (C&RT, CHAID) from the Data Mining menu to display the Interactive Trees Startup Panel. Begin the analysis by specifying a classification and regression analysis (C&RT). On the Interactive Trees Startup Panel - Quick tab, select Regression Analysis from the Type of analysis list (since Poverty is a continuous variable). For the Model building method, select C&RT.



Click the OK button to display the Interactive Trees Specifications dialog box (in this case, the ITrees C&RT Extended Options dialog box). Next, click the Variables button and select PT_POOR as the dependent variable and all others as the continuous predictor variables, and click the OK button.

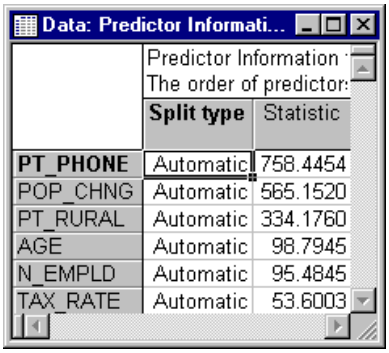


Click OK in the ITrees C&RT Extended Options dialog box to begin the analysis and to display the ITrees Results dialog box.

Manually Building the Tree

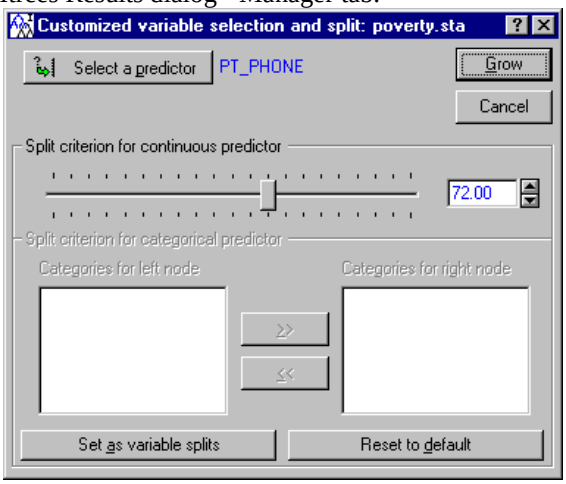
The Interactive Trees (C&RT, CHAID) module doesn't build trees by default, so when you first display the Itrees Results dialog, no tree will have been built (if you click the Tree graph button at this point, a single box will be displayed with a single root node).

Reviewing predictor statistics. Let's first review the initial predictor statistics. On the Itrees Results dialog - Manager tab, click the Predictor stats button.



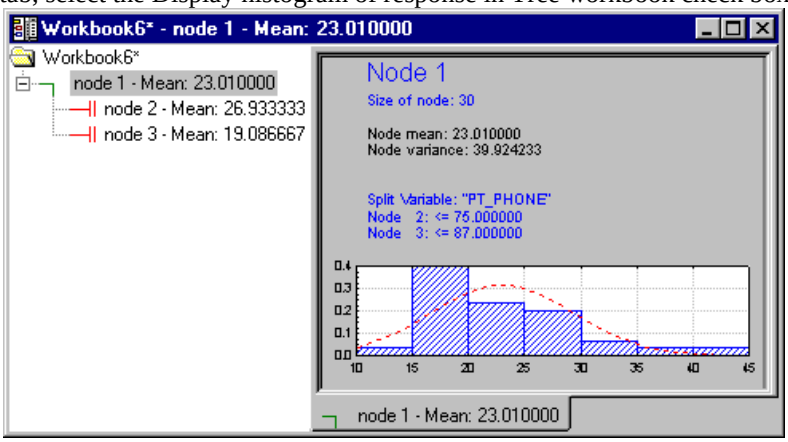
Selecting a split. This results spreadsheet shows the split statistics for the initial (Node 1) split; since this is a regression-type problem with a continuous dependent variable, the statistic shown is the sums-of-squares accounted for (explained) by the proposed split. Clearly, variable PT_PHONE (percent of residences with telephones) is the best (initial) predictor. To see what specific automatic split the program "proposes," click the Customize splits button on the

Itrees Results dialog - Manager tab.



By default, the best split for variable PT_PHONE would be at value 72.00, i.e., at 72% (of households with telephones). To simplify the final interpretation of the tree, let's round this value up to 75% (i.e., "if three-quarters or more of the households have telephones, then..."), i.e., set the Split criterion for continuous predictor to 75. Then exit the dialog by clicking the Grow button.

Reviewing the tree in the tree workbook browser. We will review the current tree via the Tree browser option, however, we also want to see the observed distributions of values. So, first select the ITrees Results dialog - Summary tab, select the Display histogram of response in Tree workbook check box, and then click the Tree browser button.

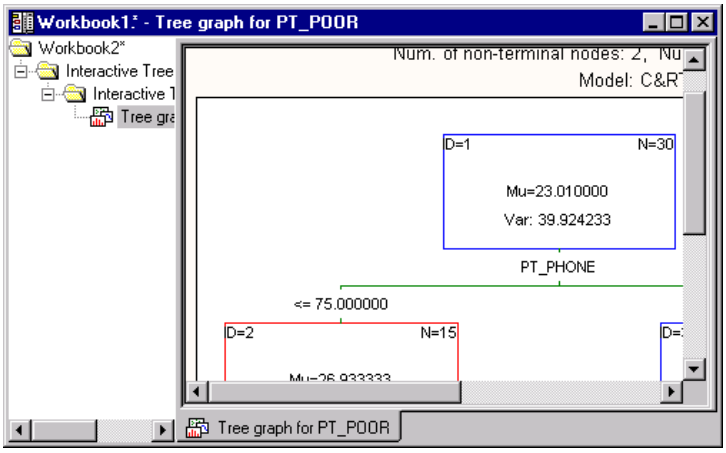


One of the useful features of the workbook tree browser (see also, Reviewing Large Trees: Unique Analysis Management Tools) is the ability to review "animations" of the final solution. Start by highlighting (clicking on) Node 1. Then use the arrow keys on your keyboard to move down the nodes of the tree. You can clearly see how the consecutive splits produce nodes of increasing purity, i.e., homogeneity of responses as indicated by the smaller standard deviation of the normal curve.

Automatically Growing (Completing) the Tree, Brushing the Tree

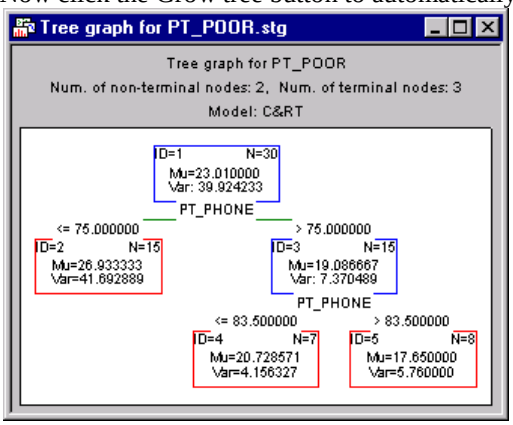
Now let's finalize the tree by automatically growing the tree to the final "stopping-point," consistent with the stopping criteria we accepted (by default) on the Interactive Trees Specifications dialog - Stopping tab; let's also use the tree-brushing tools for this purpose.

On the Itrees Results dialog - Manager tab, click the Brush tree button. If you selected (or didn't change the defaults) to display the results in a workbook, then the current tree will be displayed as a scrollable graph in the default output workbook.



Also, the Brushing Commands dialog is displayed. You can select any of the options and return to the tree brushing user interface to review the results (e.g., if you grow or prune the tree); note that the same options are also available on the shortcut menu, accessible by right-clicking on the brushing (cross-hair) cursor.

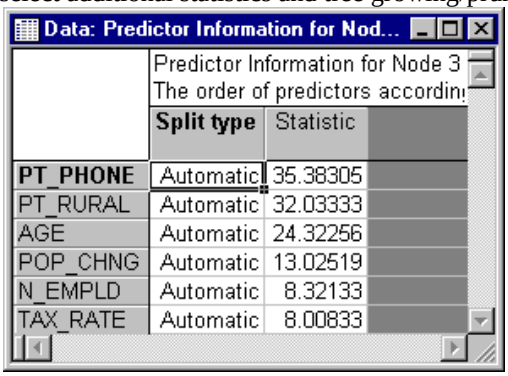
Now click the Grow tree button to automatically "finish" the tree. Shown below is the final, automatically grown tree.



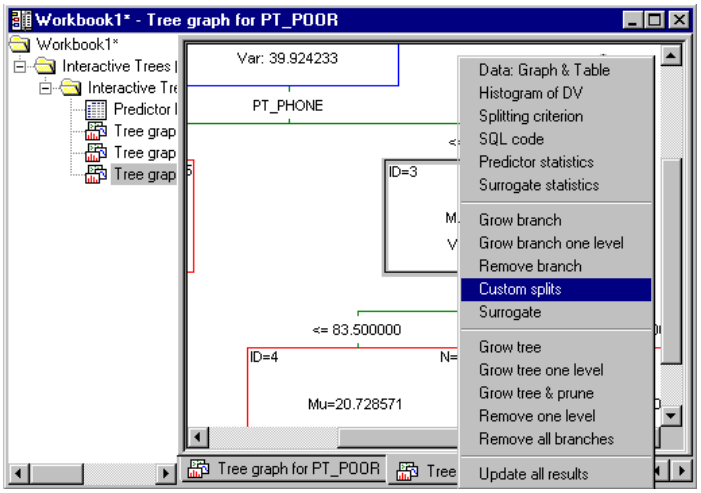
As you can see, the program split once more on the same variable PT_PHONE.

Modifying a Branch of the Tree

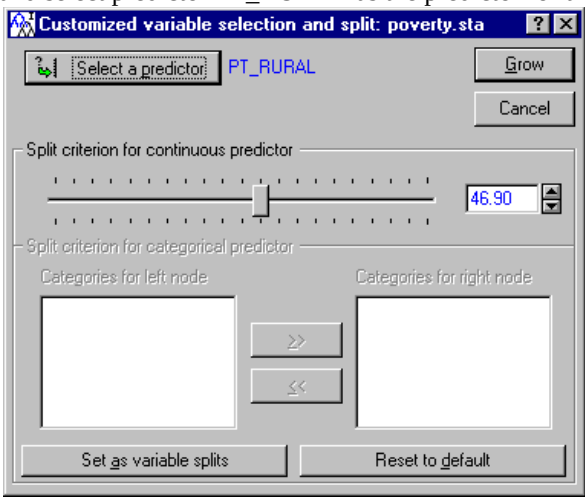
To see what other variable may have provided a good split at node ID=3, click the Brush tree button again, select node number 3, and then click the Predictor statistics button; note that in the tree brushing mode, the results spreadsheet will automatically be created in a stand-alone window, which you can position (and update) in a convenient place on the screen. After this spreadsheet is displayed, the program will automatically return to the tree brushing mode so you can select additional statistics and tree growing/pruning operations.



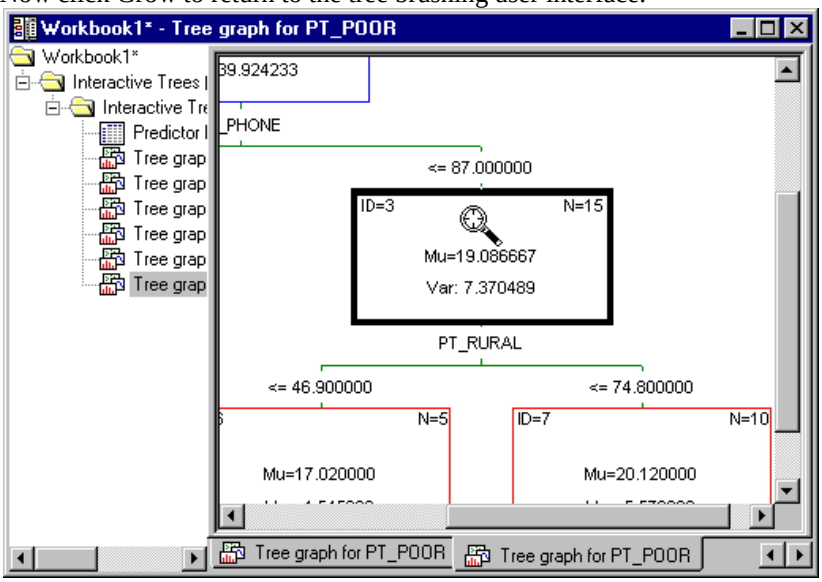
It appears that variable PT_RURAL (Percent of rural population) may provide a split of similar "quality" as did the (automatically chosen) split on variable PT_PHONE. To make the split based on variable PT_RURAL, select node ID=3 (if it is not highlighted/selected already), and then click the Custom splits button.



This will display the Customized variable selection and split dialog. In this dialog, click the Select a predictor button and select predictor PT_RURAL as the predictor for this split.



Now click Grow to return to the tree brushing user interface.



Now Cancel the tree brushing mode, and return to the ITrees Results dialog.

LAB-13

Goodness of Fit Computational Details

The Goodness of Fit module will compute various statistics for continuous and categorical variables (regression and classification problems) that reflect the quality or accuracy of the prediction or predicted classification.

Note on computations. In this module, casewise deletion of missing data is used to process the data. This may cause different results if you select different variables. For example, if you calculate goodness of fit statistics for Y, X1 and X2, then recalculate those statistics for Y, X1, X2 and X3, when X3 has missing data, the results will be different. If all cases are deleted by the casewise deletion method, an error message will be issued and the analysis will stop. Additionally, if a selected variable has no variance and correlation coefficient statistics have been selected, an error message will be issued and the analysis will stop.

Continuous variables. For continuous variables (regression problems), the following statistics are computed:

Least squares deviation (LSD), mean square error

$$LSD = \sum_{i=1}^{N} (E_i - O_i)^2 / (N-1)$$

N - Number of observations or sum of weights
E_i - Predicted value of case i
O_i - Observed value of case i

Average deviation, mean absolute error

$$AD = \sum_{i=1}^{N} |E_i - O_i| \ / (N-1)$$

N - Number of observations or sum of weights
E_i - Predicted value of case i
O_i - Observed value of case i

Relative squared error, mean relative squared error

$$RSE = \sum_{i=1}^{N} [(E_i - O_i) / E_i]^2 / (N-1)$$

N - Number of observations or sum of weights
E_i - Predicted value of case i
O_i - Observed value of case i

Relative absolute deviation, mean relative absolute error

$$RAD = \sum_{i=1}^{N} |E_i - O_i| / E_i \ / (N-1)$$

N - Number of observations or sum of weights
E_i - Predicted value of case i
O_i - Observed value of case i

Correlation coefficient (Pearson product moment correlation)

$$r = \sum_{i=1}^{N} (E_i - \overline{E}) * (O_i - \overline{O}) / \left[\sqrt{\sum_{i=1}^{N} (E_i - \overline{E})^2} * \sqrt{\sum_{i=1}^{N} (O_i - \overline{O})^2} \right]$$

N - Number of observations or sum of weights
E_i - Predicted value of case i
E- Mean of predicted values
O_i - Observed value of case i
O - Mean of observed values

Categorical variables. For categorical variables (classification problems), the following statistics are computed:

Pearson Chi-square

$$\chi^2_{N-1} = \sum_{i=1}^{N} (E_i - O_i)^2 / E_i$$

N - Number of observed classes
E_i - Number of observations in observed class i that are predicted to belong to class i (predicted or expected frequencies for observed class i)
O_i - Number of observations belonging to class i (observed frequencies)

Note that this value will become 0 (zero) when the classifier is perfect (i.e., when the expected classifications are identical to the observed classifications).

G-square (maximum likelihood Chi-square)

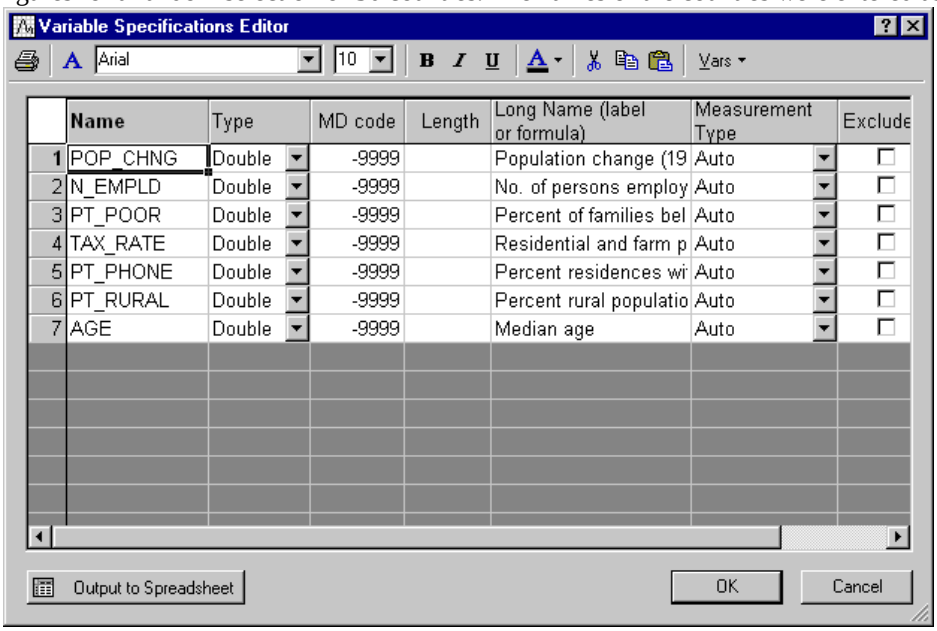
$$G^2 = 2\sum_{i=1}^{N} O_i * \ln(O_i / E_i)$$

N - Number of observed classes
E_i - Number of observations in observed class i that are predicted to belong to class i (predicted or expected frequencies for observed class i)
O_i - Number of observations belonging to class i (observed frequencies)

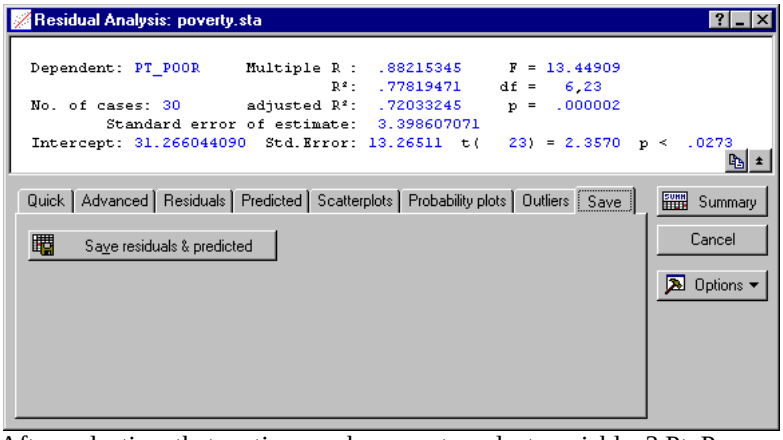
Percent disagreement (misclassification rate)

The percent disagreement measure is computed as the percent of observations for which the expected classifications are not equal to (disagree with) the observed classifications.

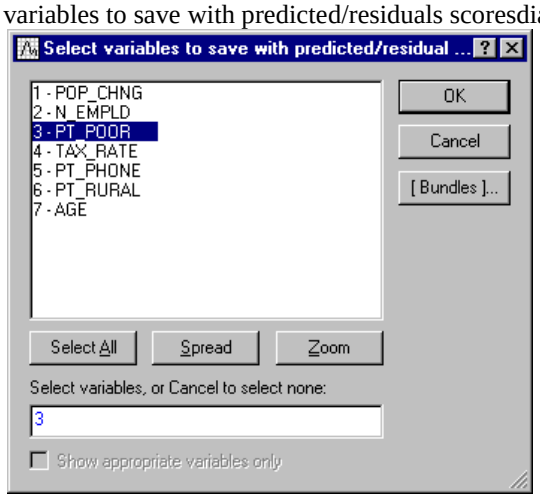
Example: Goodness of Fit Indices for Regression Predictions
The example uses the data file Poverty.sta, which contains data pertaining to the comparison of 1960 and 1970 Census figures for a random selection of 30 counties. The names of the counties were entered as case names.



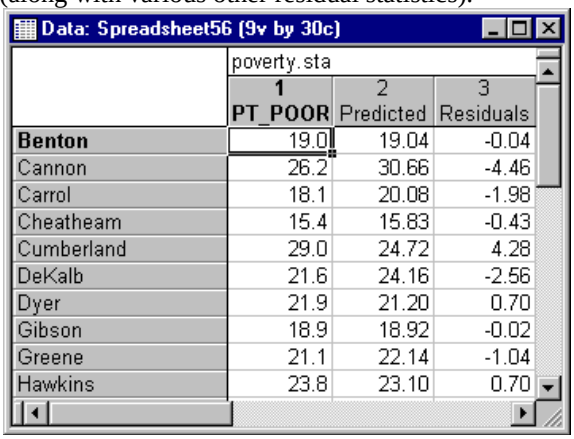
Research question. The purpose of the study was to analyze the correlates of poverty, that is, the variables that best predict the percent of families below the poverty line in a county. Thus, in the regression analysis, variable 3 (Pt_Poor) was treated as the dependent or criterion variable, and all other variables as the independent or predictor variables.
Regression analysis. Follow Example 1: Standard Regression Analysis in the Multiple Regression module to the point where the spreadsheet with predicted and residual values is computed. Then, on the Save tab of the Residual Analysis dialog box, click the Save residuals & predicted button.



After selecting that option, make sure to select variable 3 Pt_Poor as an additional variable to save in the Select variables to save with predicted/residuals scoresdialog box.



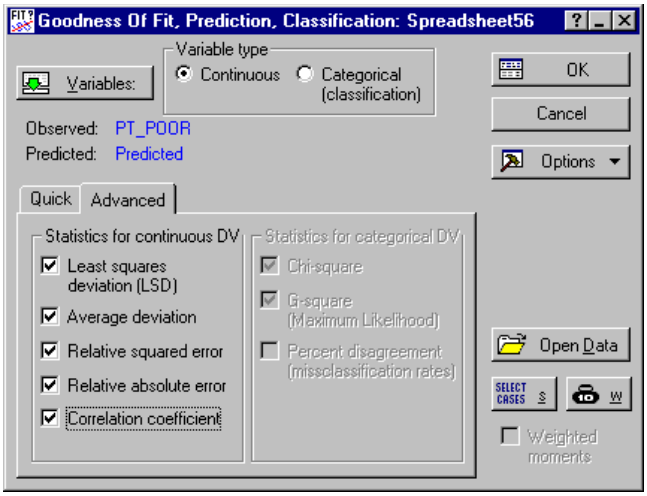
The resulting spreadsheet should then contain the observed and predicted values computed from the regression analysis (along with various other residual statistics).



	1 PT_POOR	2 Predicted	3 Residuals
Benton	19.0	19.04	-0.04
Cannon	26.2	30.66	-4.46
Carrol	18.1	20.08	-1.98
Cheatheam	15.4	15.83	-0.43
Cumberland	29.0	24.72	4.28
DeKalb	21.6	24.16	-2.56
Dyer	21.9	21.20	0.70
Gibson	18.9	18.92	-0.02
Greene	21.1	22.14	-1.04
Hawkins	23.8	23.10	0.70

Note that this results spreadsheet is automatically marked as Input, and when it is highlighted (i.e., as the top-most document in the STATISTICA application space), new analyses will automatically "connect" to this spreadsheet.

Goodness of fit computations. Next select Goodness of Fit, Classification, Prediction from the Data Mining menu to display the Goodness of Fit, Classification, Prediction Startup Panel. Ensure that the newly created spreadsheet with the observed and predicted values from the regression analysis is indeed the one selected for the analysis; if it is not, use option Open Data to select that spreadsheet. Click the Variables button and select variable Pt_Poor as the variable with observed values, and variable Predicted as the variable with predicted values. Next, on the Advanced tab of the Startup Panel, select all Statistics for continuous DV.



Next, click OK to display the Results dialog box, and click the Summary goodness of fit measures button.