

1. Introduction

1.1. Background

In this situation we use the “SEMMA” architecture. It is SAS architecture and methods to solve a data mining problem, and link it to a business suggestion. The diagrams that SAS tools suggests, are appropriate to convert the statistical results to the business solution.

1.2. Objectives

The main goal in this problem is finding the pumps which need to be repaired, so less people be affected in terms of access to a clean and potable water.

However it is a short-term goal. After finding good relations according to “need to repair” and some specific input variables, we can plan some programs to better manage the topics related to the problem.

These issues can be improved after finding the good and clear relationships between target variables and input variables:

1. Installer: maybe the high rate of fault is related to some specific installers. So we can plan for managing them better (change them, replacing them ...).

Note: this input variable is completely related to the date of installing.

2. Payment way: if the payment way relates highly to the way that installers install and maintain the pumps, the managers can decide about it to improve it.

3. Funder: maybe it relates the way and volume that funder pay to the installer and after that it affects the types of pumps. So managers can plan for it.

4. Longitude and latitude of the location: Maybe they are related in that way some environmental and natural factors involve the faulting problem. So the government can manage them.

2. Method

“Most data mining methods learn by example. The neural network or decision tree generator or what have you is fed thousands and thousands of training examples. Each of the training examples is clearly marked as being either a responder or a nonresponder. After seeing enough of these examples, the tool comes up with a model in the form of a computer program that reads in unclassified records and updates each with a response score or classification” (Data Mining Techniques, p 64).

2.1. Data

The input variables we have:

1. What kind of pump is operating?
2. When it was installed
3. How it is managed

amount_tsh - Total static head (amount water available to water point)

date_recorded - The date the row was entered (the date of the last status)

funder - Who funded the well (maybe nominal - funder of pump)

Maybe because of funder, and the way or amount his or her funding, the quality of pumps has been influenced.

gps_height - Altitude of the well

installer - Organization that installed the well

longitude - GPS coordinate (maybe if the exact point of geographical location of the pump is highly correlated with the target variable, the managers can think about the other environmental factors that is related to that specific location, e.g. we see that faults are more in some areas due to their longitudes or latitudes).

latitude - GPS coordinate

wpt_name - Name of the water point, if there is one (It cannot be related)

basin - Geographic water basin (maybe nominal)

subvillage - Geographic location (maybe nominal)

region - Geographic location (maybe nominal)

region_code - Geographic location (coded) (maybe nominal)

district_code - Geographic location (coded) (maybe nominal)

lga - Geographic location (maybe nominal)

ward - Geographic location (maybe nominal)

population - Population around the well (maybe nominal)

recorded_by - Group entering this row of data (maybe the group has recorded a little or wrong information about the well, so there is a strange behavior of being in good status for pumps) (maybe nominal)

scheme_management - Who operates the water point (maybe nominal)

scheme_name - Who operates the water point (maybe nominal)

permit - If the water point is permitted (maybe binary)

construction_year - Year the water point was constructed

extraction_type - The kind of extraction the water point uses (maybe nominal)

extraction_type_group - The kind of extraction the water point uses (maybe nominal)

extraction_type_class - The kind of extraction the water point uses (maybe nominal)

management - How the water point is managed (maybe nominal)

management_group - How the water point is managed (maybe nominal)

payment - How the users of the water point pay (maybe the way of payment delays in getting the money, so the managers cannot manage the pumps well) (maybe nominal)

payment_type - How the users of the water point pay (rejected)

water_quality - The quality of the water

quality_group - The quality of the water

quantity - The quantity of water

quantity_group - The quantity of water

source - The source of the water

source_type - The source of the water

source_class - The source of the water

waterpoint_type - The kind of water point

waterpoint_type_group - The kind of water point

Target:

status_group (nominal) with three value

2.2. Target variable

Because our target variable (status) has three values, we have to decide about converting it to other types or not. In fact we do not know how being in a “needs repair” status can affect the population who use the pump’s water. Maybe being in a “non-functional” status, affects less people in some areas than being in a “needs repair” in some other areas.

In fact, because the goal is maximizing the number of users who access to clean, potable water; and we do not know how much a “needs repair” pump affects this goal, we have to include the “needs repair” status to the “non-functional”.

So the main hypothesis in assuming “status_group” variable as a **binary** target variable is even “needs repair” pumps affect the cleanness and potability of the water, and we do not know how much.

Also we assume that if the “recorded_by” group has mentioned that a pump needs repair, then it means that not repairing the pump can affect the accessibility to clean and potable water. And this means “needs repair” has passed a threshold. This threshold means we can assume the target variable as a binary one.

2.3. Input variables

We have assumed that some of variables cannot be related to effects that take a pump into a “non-functional” or “needs repair”. Because having many input variables means complexity of the model and also raising the variance of the target variable finally.

In fact the relationships between the status of the pumps and all the input variables are complex and still not fully understood. When applying DM techniques, variable and model selection are critical issues. Variable selection is useful to discard irrelevant inputs, leading

to simpler models that are easier to interpret and that usually give better performances. So we try to ignore some unrelated (in our understanding) input variables in our model.

The variable we are going to ignore:

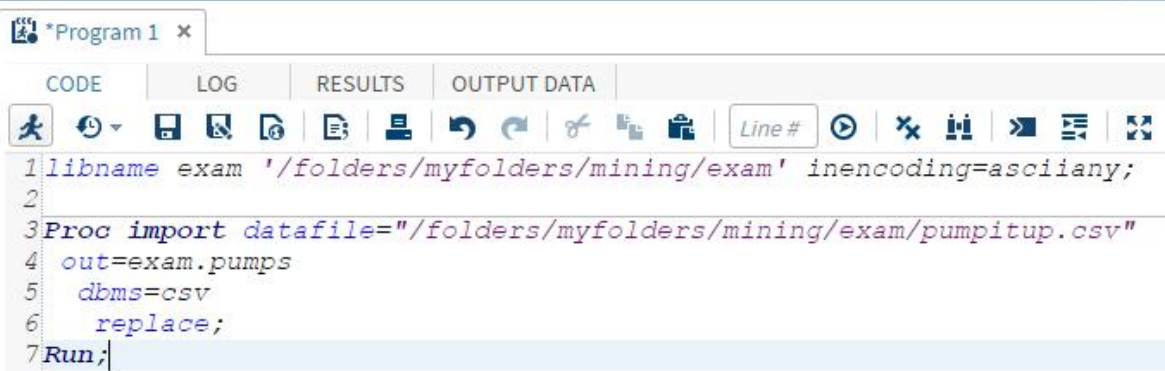
1. Funder - Who funded the well (maybe nominal - funder of pump) (It cannot be related)

Maybe it is related to the installer, so we reject it as an independent input variable. After calculation, if the rates of faulting is dependent on installers, we can inform the funders.

2.4. Data pre-processing

Because the main goal of the problem is to identify the pumps they need repair, and also we have to do it in an optimum way (that means identifying and repairing the pumps must have the most effects on people who use it), we have to check some issues about the data.

Converting data to a SAS table:



```

1 libname exam '/folders/myfolders/mining/exam' inencoding=asciiany;
2
3 Proc import datafile="/folders/myfolders/mining/exam/pumpitup.csv"
4   out=exam.pumps
5   dbms=csv
6   replace;
7 Run;

```

Before anything we want to know how many records we have.

```

proc sql;
  select count(id) from exam.pumps;
quit;

```

59400

So we have 59400 pumps (water point).

Apparently each record is related to a separate pump. We assume that people in one geographical region (region code) or district (district code) can use other pumps in the region (or district) if they are in a good (functional) status.

It is a strong and important assumption, because from the problem statement we can recognize that there is a priority that does not just depend on the number of population are using the pump. If the problem is this much easy, we do not need converting the problem to a DM problem to solve it. We could easily sort the pumps that are not in "functional" status based on the number of population who use them!

So we have to give higher priorities to pumps that are not in “functional” status, and also they are in regions (or districts) that in that region the number of people who do not access to clean and potable water is more.

So based on our assumption:

The people of a district can use another pumps in their district (if their main pump is not in “functional” status), but probably in a more difficult way,

We have to calculate the number of “non-functional” and “needs-repair” pumps in each region.

Note: we have assumed that the people in one district cannot use pumps in other district, even two region are very close to each other.

First we look at the number of people in each district:

```
proc sql;
  select region_code, district_code, sum(population) as population
        , count(id) as number_of_pumps
  from exam.pumps
  group by region_code, district_code
  order by region_code, district_code;
quit;
```

region_code	district_code	population	number_of_pumps
1	0	0	23
1	1	0	888
1	3	0	361
1	4	0	347
1	5	0	358
1	6	0	224
2	1	189	189
2	2	217803	1206
2	3	90952	109
2	5	48052	201
2	6	177195	310
2	7	206702	1009

Note: it seems that for some records the number of pumps is equal to the number of people (seems strange - region code = 2, district code = 1). Or even there are many pumps which anyone use them! (Region code = 1)

We calculate how many records we have that the number of people are less or equal the number of pumps (in comparison to whole number of pumps we have: 59400).

```

proc sql;
  create view pumps_more_poeple as
  select region_code, district_code, sum(population) as population
    , count(id) as number_of_pumps
    from exam.pumps
  group by region_code, district_code
  having population <= number_of_pumps;
quit;

proc sql;
  select count(*) from pumps_more_poeple;
quit;

```

42

So just 42 records we have that their data is strange.

Another assumption we can make is it is possible that two neighbor region (or district) can use each other's pumps if they are in the same longitude and latitude.

First we test how many distinct longitude and latitude we have.

```

select longitude, latitude from pumpitup
group by longitude, latitude

```

57520

Note: I am switching to SQL Server 2014, in cases the queries takes a long time for running.

After that we want to check how many distinct region has in each longitude and latitude.

```

select longitude, latitude, count(distinct region_code) from [dbo].[pumpitup]
group by longitude, latitude
having count(distinct region_code) > 1

```

Results		
longitude	latitude	(No column name)
0	-2e-8	3

Just 3 records.

So our assumption is not useful.

Note: we explore the data of two columns (region and district) and assume that we have to use this two column together as a distinct region column. So after this we use "region" as a combination of two fields: "region" and "district".

2.4.1. Transformations and derived features

1. Longitude and latitude

As we see the longitude and latitude have their specific range.

```
select longitude, latitude from pumpitup
group by longitude, latitude
```

	longitude	latitude
1	36.6561165	-3.229434
2	33.8287819	-9.3043568
3	32.8787795	-4.3204367
4	33.9445392	-3.6813959
5	33.9495435	-9.4631308
6	31.8402758	-8.3493827
7	34.1892784	-1.7093497
8	33.9318493	-9.6087832
9	33.0317965	-2.9897719
10	33.9711379	-9.4356118

Because other scales for other variables are different from this range, it may affect the way each input variable influences target variable.

“It is very important to scale different variables so their values fall roughly into the same range, by normalizing, indexing, or standardizing the values. “ (Data mining techniques, p 393)

So we have to normalize variables to ensure that they are all in the same range.

We do it by two formulas:

Value 1 = (Value - min) / (max - min)

It gives a number from 0 to 1

Value 2 = log (value 1 / (1 - value 1))

It gives a number from “negative infinite” to “positive infinite”

We use this method by Transform node of SAS EM.

2. basin, subvillage, region, district

In the next step we have to decide about the variables we have to include in our model. We have to test which variables are inside others, e.g. if the “basin” is in “region” or vice versa.


```
select count(distinct basin), region from pumpitup
group by region
order by region
```

100 %

	(No column name)	region
1	3	Arusha
2	1	Dares Salaam
3	3	Dodoma
4	2	Iringa
5	2	Kagera

Each region includes some basins.

```
select count(distinct region), basin
group by basin
order by basin
```

100 %

	(No column name)	basin
1	7	Intenal
2	3	Lake Nyasa
3	4	Lake Rukwa
4	7	Lake Tanganyika
5	5	Lake Victoria
6	4	Pangani
7	9	Rufiji
8	3	Ruvuma / Southern Coast
9	6	Wami / Ruvu

So we cannot decide about the ignoring the basin, and region. So we include them in our model.

3. wpt_name

We check if it does mean or not.

```
select count(id), wpt_name from pumpitup group by wpt_name having count(id) > 1
```

100 %

	(No column name)	wpt_name
1	2	Kwa Mzee Daniel
2	2	Amkeni
3	2	Kwa Lekumo
4	3	Kwa Issa Hassan
5	4	Mpilipili

There are many repeated names with our records, so we can ignore this field.

4. region

Is there any relationship between “region” and “region_code”? So we can ignore “region”?

```

select count(distinct region_code), region from pumpitup group by region
having count(distinct region_code) > 1

select count(distinct region_code) from pumpitup

```

	(No column name)	region
1	2	Arusha
2	3	Lindi
3	3	Mtwara
4	2	Mwanza
5	3	Pwani
6	3	Shinyanga
7	2	Tanga

From 27 “region_code”, and 21 “region”, some of them are repeated. We can ignore region.

5. schema_name and schema_management

Each schema_management has a number of schema_name

```

select count(distinct [schema_name]), scheme_management from pumpitup group by scheme_management
having count(distinct [schema_name]) > 1

select count(distinct scheme_management) from pumpitup
select count(distinct [schema_name]) from pumpitup

```

	(No column name)	scheme_management
11		Trust
149		WUA
63		Other
260		Water Board
135		Company
224		Water authority
124		
93		Private operator
118		WUG
124		Parastatal
1786		VWC

But 124 schema_name does not have any shcema_management. So we have to impute the values of them if we want to apply regression models or neural networks in our model.

6. permit

Has a binary value.

7. exetration_type, group, class

extraction_type: 18

extraction_group: 13

extraction_class: 7

```

select count(distinct [extraction_type]) from pumpitup
select count(distinct [extraction_type_group]) from pumpitup
select count(distinct [extraction_type_class]) from pumpitup

--select count(distinct [extraction_type]), [extraction_type_group] from pumpitup
--group by [extraction_type_group]
--having count(distinct [extraction_type]) > 1

--select count(distinct [extraction_type_group]), [extraction_type_class] from pumpitup
--group by [extraction_type_class]
--having count(distinct [extraction_type_group]) > 1

```

100 %

Results Messages

	(No column name)	extraction_type_class
1	6	handpump
2	2	motorpump

We have three layer of hierarchical data. Because it is possible that each level has its effects on the quality of pumps, we include all of them.

8. "management" and "management_group"

```

select count(distinct [extraction_type_group]), [extraction_type_class] from pumpitup
group by [extraction_type_class]
having count(distinct [extraction_type_group]) > 1

select count(distinct [management]) from pumpitup
select count(distinct [management_group]) from pumpitup

select count(distinct [management]), [management_group] from pumpitup
group by [management_group]
having count(distinct [management]) > 1

```

00 % <

Results

Messages

(No column name)
1 12

(No column name)
1 5

(No column name)	management_group
1 4	commercial
2 2	other
3 4	user-group

We include all of them (the same reason of “extraction” variables).

9. “payment” and “payment_type”

```

select count(distinct payment) from pumpitup
select count(distinct payment_type) from pumpitup
|
select count(distinct payment), payment_type from pumpitup
group by payment_type
having count(distinct payment) > 1

```

00 %	<
Results	Messages
(No column name)	
1	7
(No column name)	
1	7
(No column name)	payment_type

We can ignore “payment_type”

10. "water_quality" and "quality_group"

```

select count(distinct water_quality) from pumpitup
select count(distinct quality_group) from pumpitup

select count(distinct water_quality), quality_group from pumpitup
group by quality_group
having count(distinct water_quality) > 1

```

100 % <

Results Messages

	(No column name)
1	8

	(No column name)
1	6

	(No column name)	quality_group
1	2	fluoride
2	2	salty

We include all of them (the same reason of "extraction" variables).

11. "quantity" and "quantity_Group"

```

select count(distinct quantity) from pumpitup
select count(distinct quantity_group) from pumpitup

select count(distinct quantity), quantity_group from pumpitup
group by quantity_group
having count(distinct quantity) > 1

```

100 % <

Results Messages

	(No column name)
1	5

	(No column name)
1	5

	(No column name)	quantity_group
--	------------------	----------------

We ignore "quantity_group".

12. "source", "source_type", "source_class"

```

select count(distinct source) from pumpitup
select count(distinct source_type) from pumpitup
select count(distinct source_class) from pumpitup

select count(distinct source), source_type from pumpitup
  group by source_type
  having count(distinct source) > 1

select count(distinct source_type), source_class from pumpitup
  group by source_class
  having count(distinct source_type) > 1

```

100 %

Results Messages

	(No column name)
1	10

	(No column name)
1	7

	(No column name)
1	3

	(No column name)	source_type
1	2	borehole
2	2	other
3	2	river/lake

	(No column name)	source_class
1	3	groundwater
2	3	surface

We include all of them (the same reason of “extraction” variables.

13. “waterpoint_type” and “waterpoint_type_group”

```

select count(distinct waterpoint_type) from pumpitup
select count(distinct waterpoint_type_group) from pumpitup
select count(distinct waterpoint_type), waterpoint_type_group from pumpitup
  group by waterpoint_type_group
  having count(distinct waterpoint_type) > 1

```

00 % <

Results Messages

	(No column name)
1	7

	(No column name)
1	6

	(No column name)	waterpoint_type_group
1	2	communal standpipe

We include all of them (the same reason of “extraction” variables.

14. status_group

```

select count(id), [status_group] from pumpitup group by [status_group]

```

100 % <

Results Messages

	(No column name)	status_group
1	22824	non functional
2	32259	functional
3	4317	functional needs repair

According discussion above, we replace the nominal variable “status_group” by binary variable “working” that 1 shows it does not work or needs repair!


```

data sasuser.pumps1;
  set sasuser.pumps;
  if status_group = "functional" then working = 0;
  if status_group = "functional needs repair" then working = 1;
  if status_group = "non functional" then working = 1;
run;

proc sql;
  select distinct status_group, working from sasuser.pumps1;
quit;

proc contents data=sasuser.pumps1 order=varnum;
run;

proc contents data=sasuser.pumps1 order=varnum;
run;

```

40	working	Num	8		
----	---------	-----	---	--	--

15. "amount_tsh"

```

select count(id), amount_tsh from pumpitup group by amount_tsh
order by amount_tsh

```

0.00 % <

Results		Messages
	(No column name)	amount_tsh
1	41639	0
2	3	0.2
3	1	0.25
4	3	1
5	806	10
6	816	100
7	1488	1000
8	57	10000
9	3	100000

The range is very different (high variance). So we have to normalize it, e.g. using log function.

16. "funder"

```

select [funder], count(id) from [dbo].[pumpitup] group by funder

```

0 % <

Results		Messages
funder	(No column name)	
Songea Municipal Council	35	
Ngos	6	
Wvc	4	
Ea	5	
Manvota Primary School	1	

It is a nominal variable.

17. "gps_height"

```
select [gps_height], count(id) from [dbo].[pumpitup] group by [gps_height]
```

	gps_height	(No column name)
1	1126	8
2	704	12
3	2117	6
4	1491	35
5	317	30

It is an interval variable.

18. "installer"

```
select [installer], count(id) from [dbo].[pumpitup] group by [installer]
```

	installer	(No column name)
1	WVC	3
2	EA	5
3	Yakwetu Contractor	1
4	Manyota primary School	1
5	Colonial government	3

It is a nominal variable.

19. "lga" and "ward"

```
select lga, count(id) from [dbo].[pumpitup] group by lga
select ward, count(id) from [dbo].[pumpitup] group by ward
```

6	Mkuranga	560
7	Ludewa	564
8	Monduli	189

	ward	(No column name)
1	Bangwe	9
2	Inyonga	42
3	Ilujam...	3
4	Mvun...	30
5	Kaagya	15

They are nominal variables.

20. "permit"

```
select permit, count(id) from [dbo].[pumpitup] group by permit
```

	permit	(No column name)
1	False	17492
2	True	38852
3		3056

It is binary and needs imputing.

21. "construction_year"

```
select [construction_year], count(id) from [dbo].[pumpitup] group by [construction_year]
```

	construction_year	(No column name)
33	2011	1256
34	1992	640
35	1990	954
36	1972	708
37	0	20709
38	1980	811

We use "age" variable as difference between 2016 and the construction year.

But we have about half of all records that they do not have any "construction_year". So maybe it is good idea that we do not use it.

2.4.2. Treatment of outliers and missing values

We have checked and explored data in the previous section, and we are going to use Transform node and also Impute node for standardization and imputing the null values; because we are going to use regression and neural networks models that null values can affect on their performance.

2.4.3. Sampling and data partitioning

Maybe we want to test our models with different number of training, validation, and test samples (a slightly). We discuss them when we make the models.

The data source finally is:

Name	Role	Level
amount_tsh	Input	Interval
basin	Input	Nominal
construction_year	Rejected	Interval
date_recorded	Rejected	Interval
district_code	Input	Nominal
extraction_type	Input	Nominal
extraction_type_class	Input	Nominal
extraction_type_group	Input	Nominal
funder	Rejected	Nominal
gps_height	Input	Interval
id	ID	Interval
installer	Rejected	Nominal
latitude	Input	Interval
lga	Rejected	Nominal
longitude	Input	Interval
management	Input	Nominal
management_group	Input	Nominal
payment	Input	Nominal
payment_type	Rejected	Nominal
permit	Input	Binary
population	Input	Interval
quality_group	Input	Nominal
quantity	Input	Nominal
quantity_group	Rejected	Nominal
recorded_by	Rejected	Unary
region	Rejected	Nominal
region_code	Input	Nominal
scheme_management	Input	Nominal
scheme_name	Input	Nominal
source	Input	Nominal
source_class	Input	Nominal
source_type	Input	Nominal
status_group	Rejected	Nominal
subvillage	Input	Nominal
ward	Rejected	Nominal
water_quality	Input	Nominal
waterpoint_type	Input	Nominal
waterpoint_type_group	Input	Nominal
working	Target	Binary
wpt_name	Rejected	Nominal

Here we are not going to make a decision (confusion) matrix that is based on loss / profit calculation of guessing true about needing repair. So regarding our binary target variable, our models are based on misclassification, or other measures.

Sample

Also we are using from all records, and whenever needs, we use transform or impute nodes from SAS EM.

Partition

In data mining, a strategy for assessing the quality of model generalization is to partition the data source. A portion of the data, called the training data set, is used for preliminary model fitting. The rest is reserved for empirical validation and is often split into two parts: validation data and test data. The validation data set is used to prevent a modeling node from **overfitting** the training data and to compare models. The test data set is used for a final assessment of the model (Getting started with SAS, p 20).

Before anything we want to divide our data to three parts. Training (that our model learn from the data), validation (how good are our models), and test (for ultimately comparison of the models) we decide now about training: 70%, validation: 15%, test: 15%.

Data partition node is also useful when the number of hits was small relatively to all records, so it can be sure that the number of hits are the same in all partitions (and this a default method for data partitioning and called “stratify on target”). It does not seem that we have the same problem with our sample (discussed above, it is half approximately to have hits)



Explore

We have done this part before.

Modify

Because of above discussion, it is better that we normalize our data (e.g. using log). For example the range and the scales of “longitude” and “latitude” is so limit in comparison with “population”. So we use a transform node here.

Default Methods	
Interval Inputs	Log

Also for “schema_name” we have some null values, and want to impute them.

Missing values

When missing values must be replaced, the best approach is to **impute** them by creating a model that has the missing value as its target variable (data mining techniques, p 101).

Missing values cannot create any problem for decision trees, however because regressions and neural networks delete each record with missing values, the size of training sample is reduced and so the power of model predicting. This is more important when we want compare results of a model that ignores missing values with a decision tree (and we are going to do this in our case).

3. Results

3.1. Candidate models

3.1.1. Regression

Regressions are good for estimating (continuous variables), but also we can use them to predict a value for target variable.

Using regressions imply that the basic hypothesis of it must be meet. So we have used transform node.

Regressions have a strong hypothesis about the form of $f(X)$, so they are a parametric model.

1. Main effects: We start with a linear regression node, because we want to test if the relationship between input variables and the target variable is linear or not, and how much a linear relationship is a good fit for our model.

Regression type: linear

2. Two-way interactions: Means often as a relationship between an independent variable (IV) and dependent variable (DV), moderated by a third variable).

Two-Factor interactions: Yes

Regression type: linear

3. Quadratic terms:

Polynomial terms: yes

Polynomial degree: 2

Regression type: linear

4. Cubic terms:

Polynomial terms: yes

Polynomial degree: 3

Regression type: linear

Also we are going to use decision trees as well.

Here we do not have a high number of predictors, but we do not know about the correlation among input variables. We can try by adding gradually a variable to our model. There is some ways for doing so:

5. Forward-stepwise selection

Two-Factor interactions: no

Polynomial terms: no

Polynomial degree: 2

Regression type: linear

Selection model: Forward

Use Selection Defaults: yes

6. Backward-stepwise selection

Two-Factor interactions: yes

Polynomial terms: yes

Polynomial degree: 3

Regression type: linear

Selection model: Backward

Use Selection Defaults: No

“THE LOGIC OF SHRINKAGE METHODS

In regression models with many correlated predictors, the estimates of the coefficients can become unstable and exhibit high variance

A large positive coefficient on one variable can be cancelled out by a similarly large negative coefficient on another variable it is highly correlated with

Shrinkage estimators try to alleviate this problem by “regularizing” the regression, imposing size constraints on the parameters and shrinking them towards zero

Shrinkage methods are a “continuous” alternative to variable selection methods” (lecture 7, p 23).

“The **LARS** node can perform both variable selection and model-fitting tasks. When used for variable selection, the LARS node selects variables in a continuous fashion, where coefficients for each selected variable grow from zero to the variable's least square estimates. With a small modification, you can use LARS to efficiently produce LASSO solutions” (Getting Started with SAS EM, p 53).

“LASSO: Least absolute shrinkage and selection operator (Lasso)

Lasso was originally formulated for least squares models and this simple case reveals a substantial amount about the behavior of the estimator, including its relationship to ridge regression and best subset selection and the connections between lasso coefficient estimates and so-called soft thresholding. It also reveals that (like standard linear regression) the coefficient estimates need not be unique if covariates are collinear” ([https://en.wikipedia.org/wiki/Lasso_\(statistics\)](https://en.wikipedia.org/wiki/Lasso_(statistics))).

“AIC

Is an IN-SAMPLE GOODNESS-OF-FIT MEASURE which Estimates the degree to which the predictive accuracy of the model will generalize to new data ($AIC = -2 \ln(L) + 2p$)

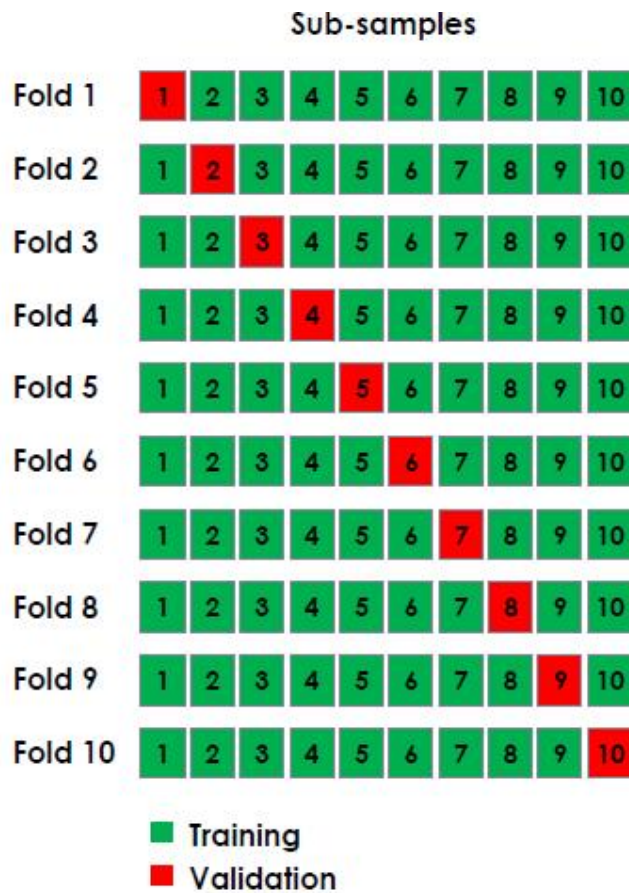
Finally we do not have AIC for LASSO and LARS, because AIC uses p parameters for estimating accuracy, but LASSO and LARS are types of Shrinkages and so they use all parameters” (lecture 8 - p 9).

“k-FOLD CROSS-VALIDATION

The original data set is randomly divided into k equally-sized sub-samples.

In each step (“fold”), $k-1$ sub-samples are merged into a training set and the remaining sub-sample is used as the validation set ‘

The process is repeated k times and the results are averaged over the k folds” (lecture 8, p 20).



7. LASSO

Variable Selection Method: LASSO

Model Selection Criterion: AIC

Path Stopping Criterion: Cross Validation

Cross Validation: Random (each training observation is randomly allocated to one of the n-cross validation folds).

Details: All (all details include Anova, fit statistics, parameter estimates ... for the top 10 candidates for inclusion or exclusion at each selection step is produced.

8. LARS

Variable Selection Method: LAR

“Regression models are based on least squares estimators. When the number of predictors or the correlation among them is high, we have more complexity. That means we have less bias and more variance. Unfortunately this situation affects the power of prediction accuracy. So we tend to reduce the variance by making some coefficients to zero. The other advantage is we can interpret and understand the model better, when it does not have so much input variables.

So one of the methods we use to reduce the complexity (the number of predictors) is forward-stepwise selection.

Forward-stepwise selection starts with the intercept and then sequentially adds the predictors that most improve the fit. Here after adding the new variable, we update other coefficients” (lecture 7, p 18).

“The **Dmine Regression** node enables you to compute a forward stepwise, least squares regression model. In each step, the independent variable that contributes maximally to the model R-square value is selected. The tool can also automatically bin continuous terms” (Data Mining Using SAS EM - p 12).

9. Dmine Regression

Use interactions: yes (inclusion of all possible two-way interactions of class variables)

That means we are considering the effects of relationship between two variable on each other, so how the dependent variables can be affected from them.

Print option: All (all details)

We add the same nodes until now, but with logistic type regression: Main effect, quadratic, cubic

Results:

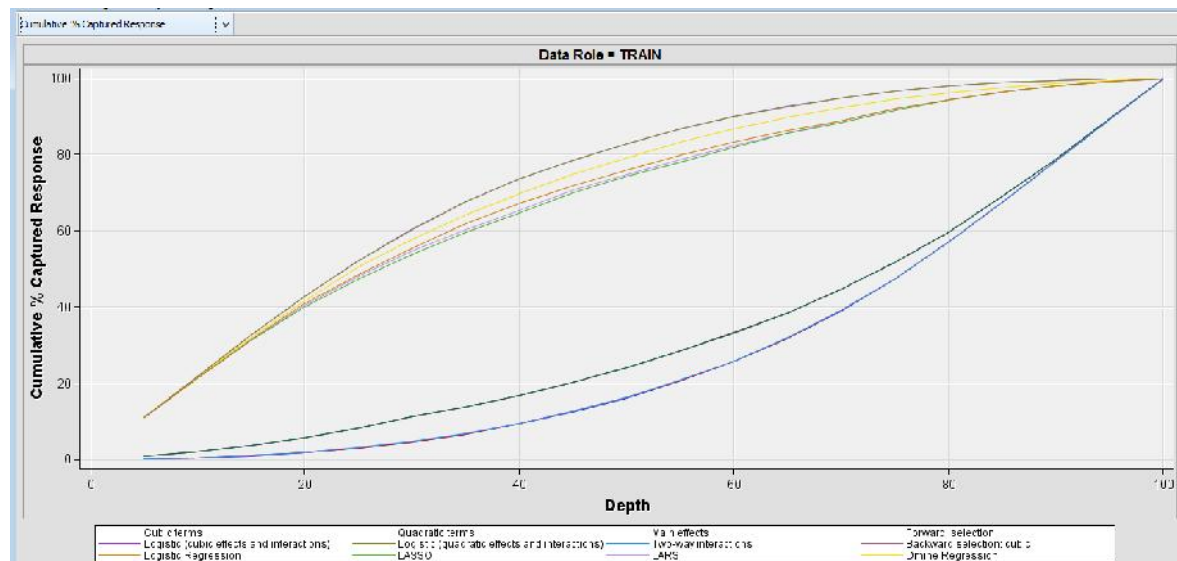
1. Model comparison

Model Description	Target Variable	Target Label	Selection Criterion: Test: Misclassification Rate
Logistic (cubic effects and interactions)	working		0.205341
Logistic (quadratic effects and interactions)	working		0.206127
Cubic terms	working		0.208595
Quadratic terms	working		0.209156
Two-way interactions	working		0.210839
Dmine Regression	working		0.224753
Logistic Regression	working		0.238555
Main effects	working		0.24136
Backward selection: cubic	working		0.24136
Forward, selection	working		0.24136
LARS	working		0.249776
LASSO	working		0.253366

Because logistic regression has better performance than linear one, we can guess the process of standardization and normalization is important. Data has measured in very different measures, so it is a good idea to standardize them.

After that cubic and quadratic regression have better performance. So we can conclude that the relationships among input variables is complicated. Also they are correlated with each other, and this can be due to bad selecting the input variables also.

2. cumulative % captured response



3.1.2. Nearest neighbor

Parametric models are useful because they have an easy interpretation of coefficients and they can fit to reality easily. Also we can test statistical significance about the fitness of the model easily.

The main problem with them is their robust hypothesis about the form of $f(X)$ and if the model does not correspond with reality and our target is the accuracy of prediction, then the model does not work well.

In contrast, non-parametric models does not assume explicitly a clear parametric form for $f(X)$, and they are more flexible for performing regression.

The most popular and the easiest non-parametric model is the k-nearest (or KNN regression).

“We have two concepts: similarity and preference. Nearest neighbor technique is based on similarity. We call it MBR or Memory Based Reasoning, because we categorize objects in that way we think they are similar to our past experience. Collaborative filtering adds more information including preference.

We apply these concepts in two steps:

1. Finding similarities between the new record and the old ones.
2. Combining these information with other neighbors.

Collaborative filtering is the same with nearest neighbor, the only different is grouping people for suggesting to them.

MBR uses two tools for this:

1. Distance Function: how it calculate the distance
2. Combination Function: the ability of combining the results of many neighbors to each other for finding the answer.

So the steps are:

1. Choosing the training set
2. Determining the distance function
3. Choosing the number of nearest neighbors
4. Determining the combination function" (Data mining techniques - p 286).

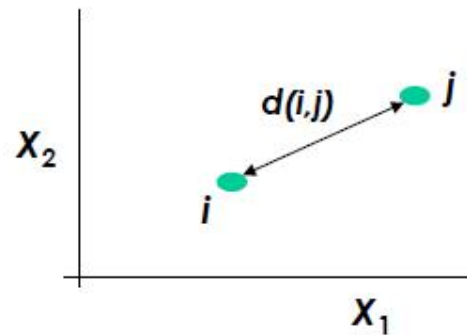
Choosing distance functions: Euclidean distance

If $q = 2$, d is the Euclidean distance:

$$d(i,j) = \sqrt{(|x_{i_1} - x_{j_1}|^2 + |x_{i_2} - x_{j_2}|^2 + \dots + |x_{i_p} - x_{j_p}|^2)}$$

Properties:

- $d(i,j) \geq 0$
- $d(i,i) = 0$
- $d(i,j) = d(j,i)$
- $d(i,j) \leq d(i,k) + d(k,j)$



Choosing combination functions: majority voting (Democracy)

"When MBR is used for classification, each neighbor casts its vote for its own class. The proportion of votes for each class is an estimate of the probability that the new record belongs to the corresponding class. When the task is to assign a single class, it is simply the one with the most votes. When there are only two categories, an odd number of neighbors should be pooled to avoid ties. So if we have c class, we have to get help from $c+1$ class for voting, because at least one class can have the majority" (Data mining techniques - p 286").

Also in the MBR (Memory-based Reasoning Node), the input variables must be standardized. So we are connecting the impute node to our candidate models (after transforming them). The reason is "the algorithm depends on the difference between all parts of Euclidean distance between the separate input variables and the probe x in which the best set of k nearest neighbor values are determined by the range of values in the input variables in the model" (Data Mining Using SAS Enterprise Miner).

Candidate models

K-nearest neighbor classifiers of increasing complexity with $k = \{16, 8, 4, 2, 1\}$, using **weighted Euclidean distances** and **majority voting**.

Weighted: yes

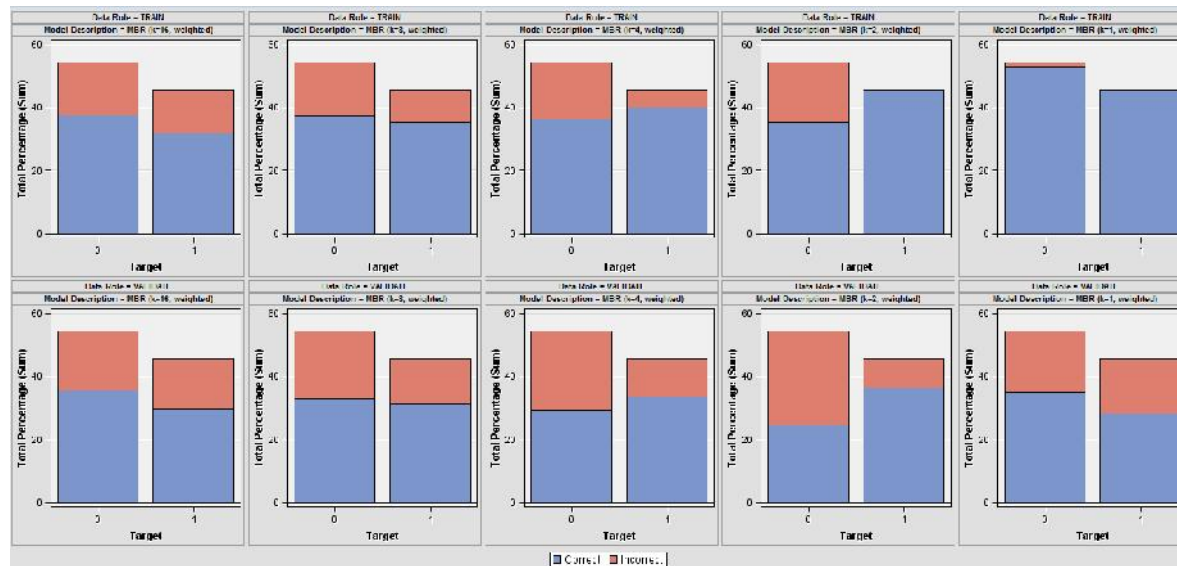
According to the amount of correlation the input variable has with target variable, node uses weighted, but this option is ignored if the class variable have more than two levels. And here seems that this occurs.

Fit statistics

Model Description	Target Variable	Target Label	Selection Criterion: Test: Misclassification Rate
MBR (k=8, weighted)	working		0.355925
MBR (k=16, weighted)	working		0.357047
MBR (k=4, weighted)	working		0.371522
MBR (k=1, weighted)	working		0.372195
MBR (k=2, weighted)	working		0.392729

It seems that the models does not perform well on validation sample. The best model ($k = 8$) has 35% misclassification!

Classification Results



As we discuss the model must be better in recognizing the pumps need to be repaired, and here the ($k=8$) is better than other models. However it does not perform well in recognizing the “functional” pumps.

3.1.3. Decision trees

“A decision tree is a structure that can be used to divide up a large collection of objects into successively smaller sets of objects by applying a sequence of simple IF-THEN decision rules.

A decision tree for classification consists of a set of rules for dividing a large heterogeneous group of objects into smaller groups of objects that are more homogeneous with respect to a particular target variable.

Most decision tree algorithms start with a recursive partitioning procedure, the split search, to grow the maximal tree

At any iteration of the search procedure, the best split is the one that leads to the best separation of the data into groups in which a single class of the target variable dominates

Potential splits can be compared in terms of their purity. The best split is the one that increases the purity of the subsets by the greatest amount and creates nodes that are not too small

It has a splitting criterion that make decision about choosing the best split in each iteration and for deciding when to stop the search for additional splits

MEASURES OF NODE IMPURITY

Let \hat{p}_{mk} represent the proportion of class k observations in a node m that represents a region R_m with N_m observations

Besides adjusted p values from a statistical test, other common measures of the impurity of a node m are

- **Misclassification rate:** $MCR = 1 - \hat{p}_{mk}$
- **Gini index:** $G = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk})$
- **Entropy:** $S = -\sum_{k=1}^K \hat{p}_{mk} \log \hat{p}_{mk}$

Because decision trees have a strong tendency to overfit the data in the training set, we want to prune the subdivision of tree in a very optimum way (with best performance for remaining tree). So the results are a series of sub-trees with reduced number of leaves.

From among these sub-trees, the final one is chosen. Model selection is based on performance in the validation set. Common model selection criteria include:

- Misclassification rate
- Gini index
- Average profit/loss
- Lift "(lecture 12).

"Decision trees don't make an assumption regarding the distribution of the data, so they are nonparametric."

Reference: <http://stackoverflow.com/questions/13845816/are-decision-trees-e-g-c4-5-considered-nonparametric-learning>

Easy to understand, without need to imputing the missing values (Getting started, p 23)

Because we do have a binary variable as our target here, we do not need use a “Rules Builder” and also a “Meta Data” node, before using decision tree node.

Candidate models

Decision Tree properties:

1. **Maximum depth**: the number of levels of generation, however the final tree may have fewer levels because of pruning.

We set two types of depths = 5, 10

2. **Leaf size**: minimum number of training observations

We do not change this for our problem. (Default = 5)

3. **Number of surrogate rules**:

Because in decision trees we do not need to use impute node, so it is possible that we have null values. Setting this property identifies in each not-leave node of the tree, if splitting rule is based on an input that its value is null, then SAS EM uses this number of “surrogate rules” (for splitting).

This specification enables SAS Enterprise Miner to use up to four surrogate rules in each non-leaf node if the main splitting rule relies on an input whose value is missing. (Getting Started, p 24)

We do not change this for our problem. (Default = 0)

Our target is binary, so decision tree behaves with it as a nominal variable.

4. **Nominal target criterion** (ProbChisq, Gini, Entropy)

5. **Significance level**: 0.2 (default).

6. **Maximum branch**: 2 (binary), 5 (multi way)

7. **Method**: Assessment

Subtree properties group, shows the properties for pruning. Default method of pruning is “assessment”, and this means that algorithms in SAS select the best tree based on some optimality measures.

Other methods are “largest” and “N”.

Largest: automatically generates the largest tree

N: a tree with n leaves.

8. **Assessment measure**: misclassification

The default is “Decision”, but it is for when we have a confusion (decision) matrix (loss / profit matrix). We use misclassification for this property, because we want to compare it to other models.

Results

Model Description	Target Variable	Target Label	Selection Criterion: Test: Misclassification Rate
Multiway Tree (Gini, dep=10)	working		0.211849
Multiway Tree (Entropy, dep=10)	working		0.215103
Multiway Tree (split .20, dep=10)	working		0.215328
Multiway Tree (split .20, dep=5)	working		0.223743
Multiway Tree (Gini, dep=5)	working		0.224865
Binary Tree (Gini, dep=10)	working		0.226548
Multiway Tree (Entropy, dep=5)	working		0.228456
Binary Tree (Entropy, dep=10)	working		0.232383
Binary Tree (split .20, dep=10)	working		0.237208
Binary Tree (Gini, dep=5)	working		0.260435
Binary Tree (split .20, dep=5)	working		0.260435
Binary Tree (Entropy, dep=5)	working		0.266943

The best model (among decision trees) regarding our measure (misclassification) is the multiway tree (5 branches) with Gini criterion, and maximum depth = 10. Results show for all our candidates, when we increase the depth, we can make better predictions about the target.

“In fact setting the rules for subtrees make a pool of candidate subtrees in validation step, but just one of them (clearly) has the most power to predict the target variable value in the test sample. Each of the candidate subtrees is used to classify the records in the validation set. The tree that performs this task with the **lowest overall error** rate is declared the winner. The winning subtree has been pruned sufficiently to remove the effects of overtraining, but not so much as to lose valuable information.

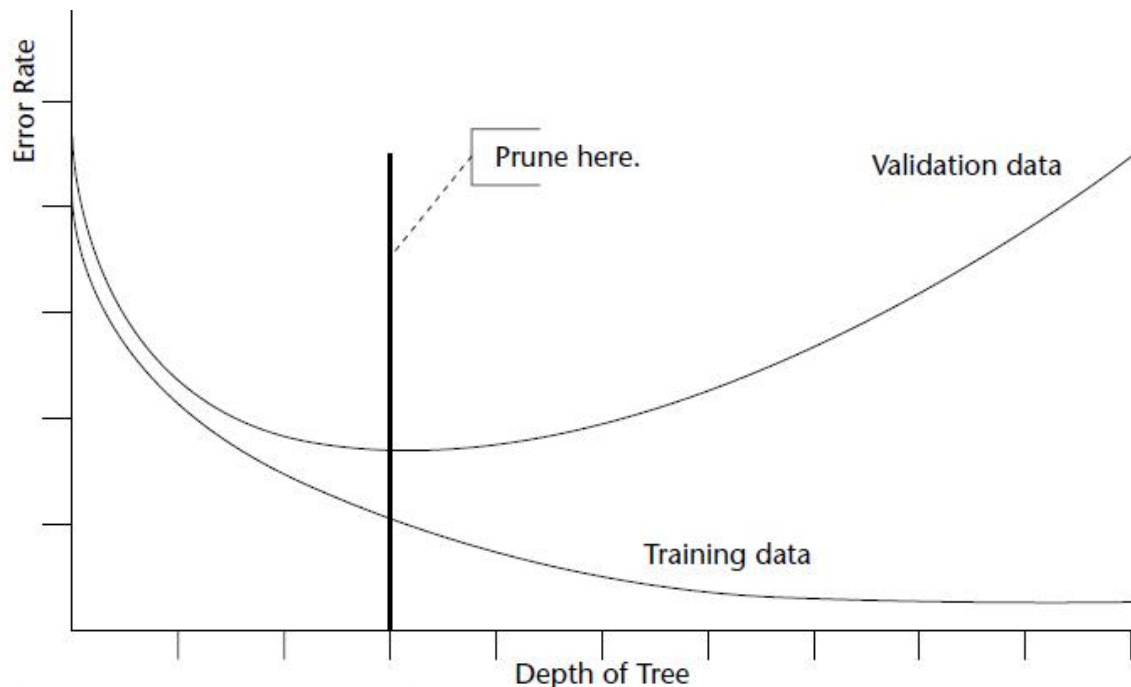


Figure 6.7 Pruning chooses the tree whose miscalculation rate is minimized on the validation set.

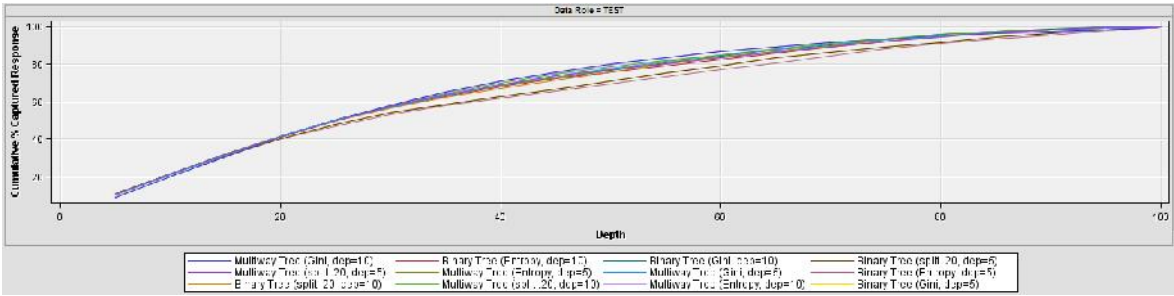
“(Data Mining Techniques, 219).

The diagram shows the point that pruning the subtrees is happen. It is similar to a tradeoff between bias and variance when we are tuning the parameters for our models. More learning from training data (more overfitting to it), less performance on new data. We have to put a tradeoff between learning useful data (to get more power for predicting), and overfitting.

So going to more depth always does not mean to get a better result. Maybe it can cause overfitting. We have selected 5 and 10 depth (level of tree) for our case. And we see that 10 is better than 5.

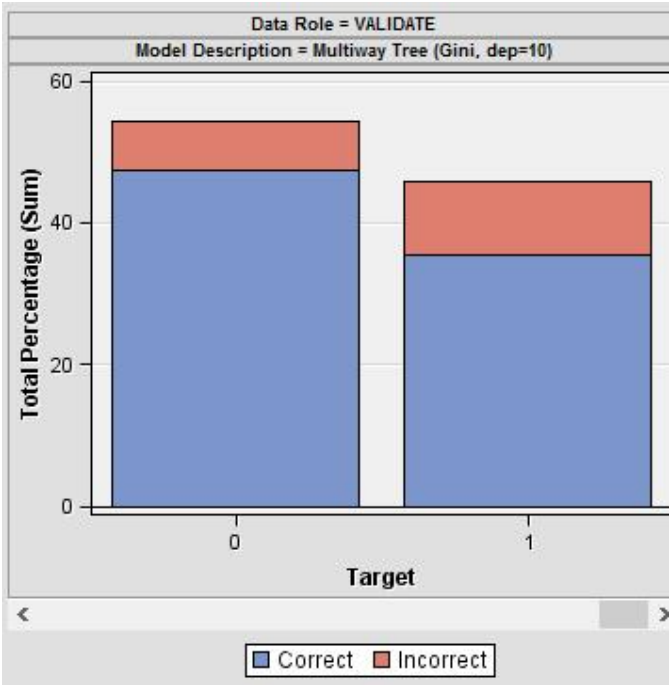
Also we see multiway trees have better performance in predicting. Here we have similar reason. When decision tree start to use decision rules for splitting the tree, it is going to make largest tree with a balanced number of hits (meeting the target variable, true - in case it is binary - and the same for interval targets) in each leave (branch). Increasing the number of branches can help to get a better result, regarding the number of involving (correlated with target) input variable, and also the size of training sample. “Theoretically, decision trees can assign records to an arbitrary number of classes. Increasing the number of branches, in cases that the number of training examples per class gets small” (Data Mining Techniques). Here we can see having 5 branches is better than a binary tree, probably because of large number of input variables and also the number of records. But we know certainly, that there is an optimum level for it.

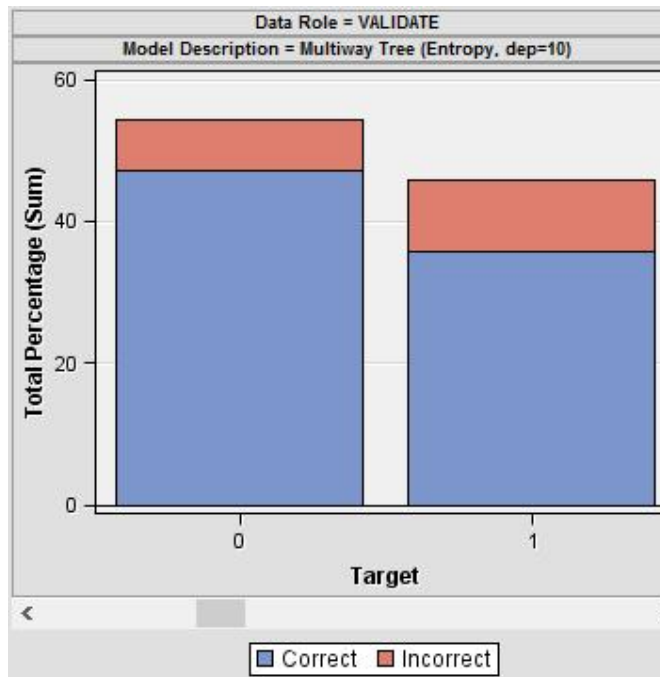
Cumulative % captured response here means if we sort the pumps with probabilities of needing repair, then our model how different is from the random model (without using any model). we have to compare it with the percent we have selected.



For example, for the binary tree entropy dep = 5, the cumulative % captured response is 69% at the depth of 50%. So this model at this depth works (69 - 50) 19% better. That means if we sort pumps using this model and select half of them, we can find needing repair pumps 19% more than the case we are selecting randomly half of pumps.

Classification results





Multiway tree entropy, dep = 10 is slightly better than our selected model (Gini), if we consider only misclassification rate at the second column (needing repair). Because it depends directly to our assumptions that government is going to repair pumps at the minimum cost.

3.1.4. Bagging

Ensemble methods is divided to two classes:

1. methods based on independent samples
2. methods based on running a whole series model (residuals)

Independent models:

1. bagging (in general)
2. random algorithms (random forest) (the most powerful of shelf data mining methods)

“The logic of ensemble models

In a typical data mining project, many competing models are fit to the data. Although these models can be ranked in terms of their predictive performance, it is unlikely that the “true model” will actually be among them. Ensemble methods acknowledge that all models have their strengths and weaknesses: each model may solve part of the prediction problem but may fail miserably at others. The solution is to aggregate many imperfect models into a “meta-model” that **combines their strengths** and **averages out their weaknesses**: an ensemble.

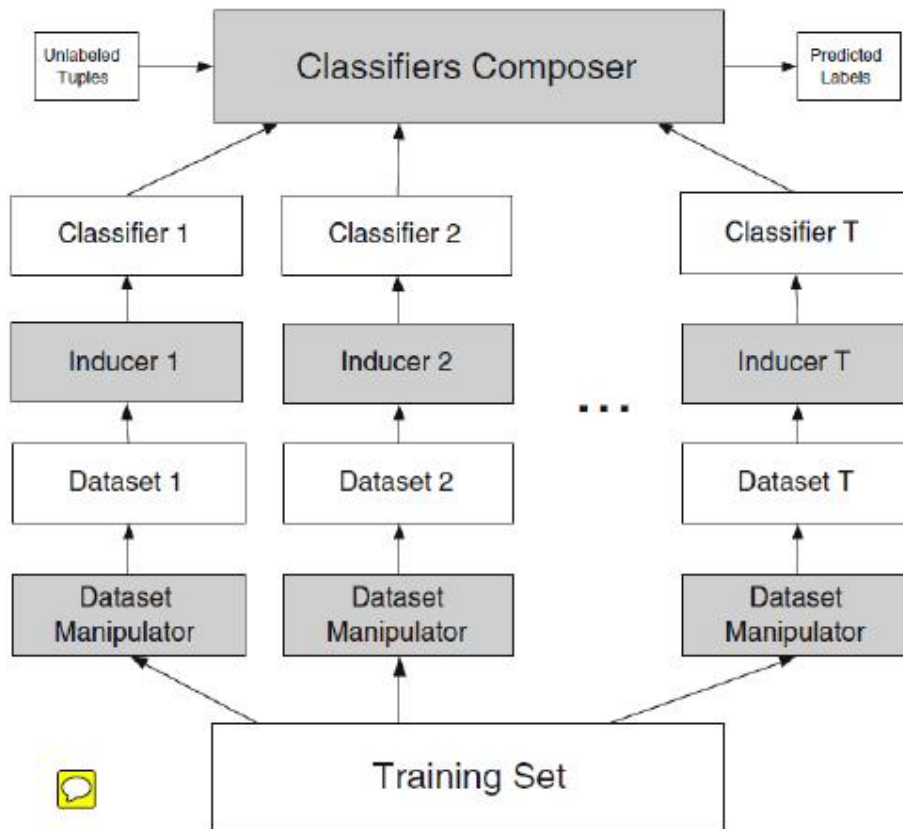
WHEN DOES SUCH “MODEL CROWDSOURCING” WORK?

Diversity: the base models in the ensemble must be sufficiently different from each other to make separate contributions

Independence: the estimation of the different base models should be as uncorrelated as possible.

Localization: it can be useful to let different base models adapt to particular regions of the search space so that their local bias becomes very low

Aggregation: the diverse models with low local bias must be combined in such a way that their aggregate variance is reduced



We can develop in each branch an independent model.

THE BAGGING ALGORITHM

Bagging utilizes standard non-parametric **bootstrapping** methodology. The learning sample (of size N) is regarded as a finite population. From the learning sample, B bootstrap samples of size N_b are drawn at random **with** replacement. In each bootstrap sample, the basis model is estimated.

The bagging estimate at an input x is the average prediction (in regression applications) or majority vote (in classification applications) over the B bootstrap samples" (lecture 16).

Step 1: from our decision trees model, we select two first best as a candidate for using in Bagging.

Step 2: identifying the number of Bootstrap samples

Step 3: the size of N_b in each Bootstrap samples

We have a learning sample with size N . in our case, N is total records (59400). After that we identify B samples each with the size of N_b .

In each bootstrap sample, the basis model is estimated. The bagging estimate at an input x is the average prediction (in regression applications) or majority vote (in classification applications) over the B bootstrap samples.

1. We add a **start group** node.

Mode: bagging

Index count: 2, 5 (the number of bootstrap samples) (B)

We start with small numbers for B

Also we start with 100% of data (with replacement). More sample can be more learning opportunity, but we are going to test other numbers later.

Results

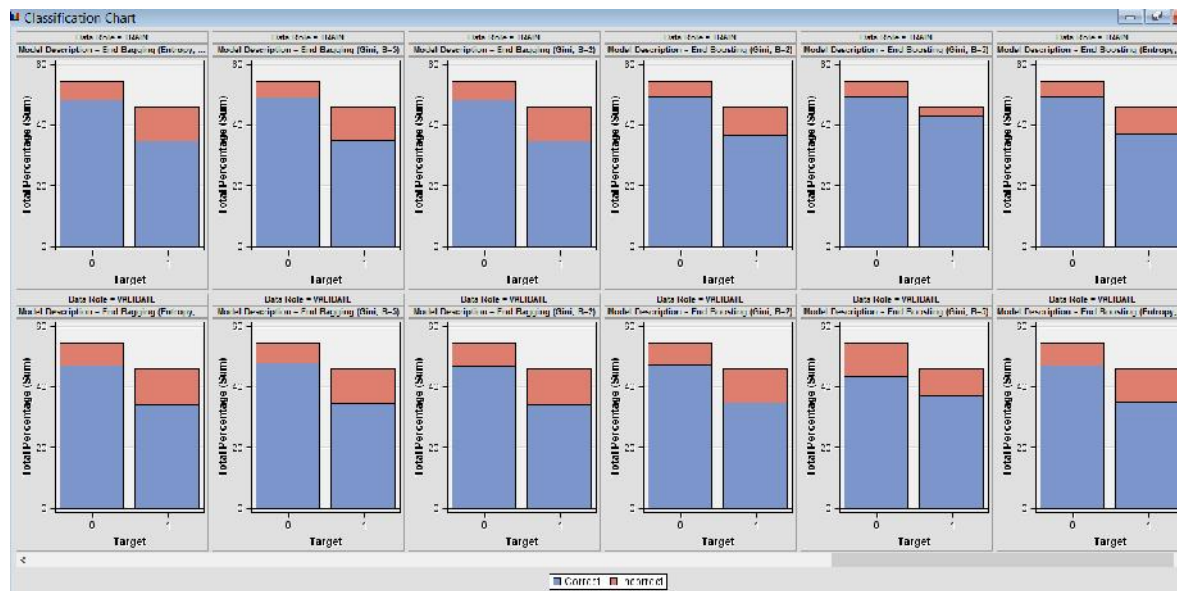
1. Model comparison

Model Description	Target Variable	Target Label	Selection Criterion: Test: Misclassification Rate
End Bagging (Gini, B=5)	working		0.200853
End Boosting (Entropy, B=2)	working		0.203995
End Boosting (Gini, B=2)	working		0.205902
End Bagging (Entropy, B=2)	working		0.209381
Multiway Tree (Gini, dep=10)	working		0.211849
Multiway Tree (Gini, dep=10)	working		0.211849
Multiway Tree (Entropy, dep=10)	working		0.215103
Multiway Tree (Entropy, dep=10)	working		0.215103
Multiway Tree (split .20, dep=10)	working		0.215328
End Bagging (Gini, B=2)	working		0.215889
End Boosting (Gini, B=5)	working		0.219143
Multiway Tree (split .20, dep=5)	working		0.223743
Multiway Tree (Gini, dep=5)	working		0.224865
Binary Tree (Gini, dep=10)	working		0.226548
Multiway Tree (Entropy, dep=5)	working		0.228456
Binary Tree (Entropy, dep=10)	working		0.232383
Binary Tree (split .20, dep=10)	working		0.237208
Gradient Boosting	working		0.237769
HP Forest (m=10)	working		0.238779
Binary Tree (Gini, dep=5)	working		0.260435
Binary Tree (split .20, dep=5)	working		0.260435
Binary Tree (Entropy, dep=5)	working		0.266943

Bagging and Boosting behave well (higher performance) regarding to other models (binary trees, multiway trees, Random Forest, Gradient Boosting). However we know Random Forest is better when the proportion of sample increases.

Also about the Gradient Boosting we know that shrinkage factor is an effective factor and the value of it must be in accordance of the number of iterations.

2. Classification results



We know that “end bagging, gini, b=5” has the best misclassification. Although it has higher misclassification rate (11.20%) in identifying the “need repair” status in comparison with “functional” status misclassification rate (6.52%). And we know the first one is more important, because we want to identify “need repair” pumps.

3.1.5. Random Forest

“THE RANDOM FOREST ALGORITHM

Random forests are a “tweaked version” of bagged decision trees, introducing additional elements of diversity and independence into the ensemble. Like in bagging, B new samples are drawn from the learning set. However, N_b is typically smaller than N , and sampling is simple random **without** replacement. In each sample, a subset of m input variables is chosen at random from the p original input variables. Using these, a **binary tree** is grown to considerable depth and not pruned. The classifications by the B sets of decision rules are aggregated via **majority voting**” (lecture 16).

We add a HP Forest node.

Maximum number of Trees: 50 (the number of sample) (B)

Type of sample: proportion

Proportion of observation in each sample: 0.6 (N_b)

Number of variables to consider in split search: 10, 5 (m)

Note: HP Forest uses out-bag samples as validation, so actually we do not have test sample.

HP Forest uses out-of-bag validation of the results which allows you to train on more data than if you do a three-way partition.

In Enterprise Miner, random forests are specified using the "HP Forest" procedure from the HPDM tab. Note that it only accepts a two-sample partition (learning and validation) and no additional test set (lecture 16, p 19)

3.1.6. Adaptive Boosting

"Ensemble methods is divided to two classes:

1. Methods based on independent samples
2. Methods based on running a whole serious model (residuals)

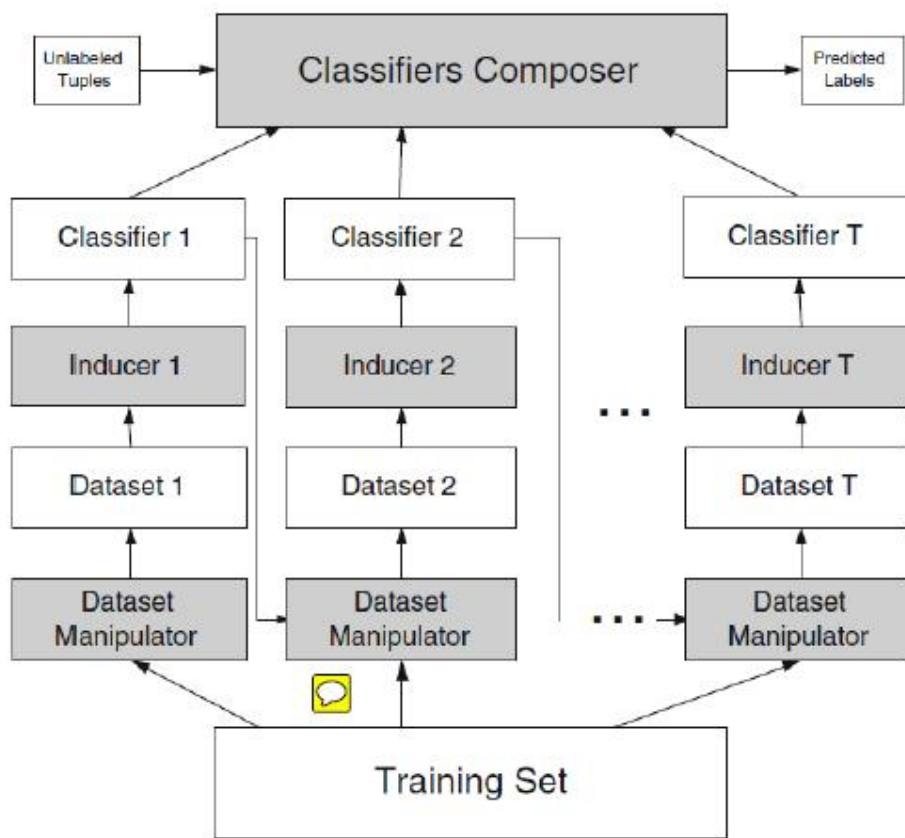
Dependent models:

1. Adaptive Boosting
2. Gradient Boosting

THE Boosting ALGORITHM

Boosting is built upon the principle of **slow learning**. A series of relatively weak basis models are fitted to the learning data in such a way that models that come later in the sequence **learn from the errors** made by models that came earlier in the sequence. To reduce overfitting, the estimated parameters of the basis models are **frozen** and not updated. The ensemble prediction is a **weighted average** of the predictions by the basis models

Dependent Ensemble Methods



We have a new overfit in each branch and work on fitting a new model using residuals on previous models. The performance of the model is based on the residuals. And after that a new data set is built.

The classified algorithm: AdaBoost

Stands for Adaptive Boosting and this is a method for typically decision tree so a classification method. And it work with weight.

First step in the basic algorithm is initialization of the weights, and in the first round we give them the weights equally, because we do not have any classification result yet.

The next iteration we figure classify, and the way each case is initially equal.

Then we calculate the error rate (misclassification rate),

Then we calculate the Logit (the real valued transformation of the error rate), then we update the rates in such a way that of the cases of misclassified to get pushed up, and the cases that could not be misclassified get the weight zero.

Then in the next iteration, we will fit a classifier to the data in such a way that the cases we already collect the classified, actually get zero weight. And the cases which are really classified actually enter to building the model.

And repeat the iterations until each case is classified. After ending the loop, we can see how our majority roots, simple algorithm classifies the cases.

This is a very simple algorithm that we classify the first round with our decision tree and then set weight zero to correct classified cases. And after that build the second tree on the misclassified cases of the first round and so on.

Parameters:

Index count: the number of iterations (default = 100) (we set at the first step = 2, 5)

3.1.7. Gradient Boosting

In a situation that we cannot find a parametric model, for our data, we try to estimate a sort of functional relationships between our input variables and our target variable. By use of a lots of essentially simple functions that together form the composite.

Another thing that Boosting does is it uses a regularization parameter. Because if you fit a serious of models to the residuals from the previous models, we get extremely strong overfitting.

Updating the residuals from the previous models can be slow down by a relatively small multiplier: A shrinkage factor $0 \leq v \leq 1$ is introduced that directly controls the learning rate in the boosting loop.

If $v = 1$: no regularization curve

If $v = 0$: no updating

Also v slowdowns the overfitting. The logic is each successively fitted model can only make a relatively small contribution to our whole sample.

The Lower you set this parameter, the more iterations you need. For example if $v=0.1$ we need 5000 iteration, otherwise it cannot improve quickly enough.

In each iteration, a tree is fit to the residual of the prediction from earlier iterations. The residual is defined in terms of the gradient of a loss function (binomial or multinomial deviance)

Like decision trees, boosting makes no assumptions about the distribution of the data (Getting Started, p 29). So we can use it in a balanced or non-balanced sample data and we do not need a transformation node, so we are connecting the sample data directly to it (however we have used transform node, due to decision trees)

Parameters:

N Iterations: 50 (depending on what sort of shrinkage factor, you have to set this parameter)

Shrinkage: 0.1 (very small) - we set 0.5

Train proportion: the proportion of training set that is sampled in each iteration (like in Random Forest) (in percent): 60 (we can play it)

Maximum branches: how many branches each tree is allowed to split in. (here at the first we set it to 5)

Maximum depth: how deep each tree is allowed to grow (here at the first we set it to 10)

Number of surrogate rules are backup rules that are used in the event of missing data (we do not have missing data here).

3.1.8. Neural Networks

“Neural networks are a class of parametric models that can accommodate a wider variety of nonlinear relationships between a set of predictors and a target variable than can logistic regression” (Getting Started with SAS EM, p 38).

So it includes the wider range of relationships than regressions.

“Building a neural network model involves two main phases. First, you must define the network configuration. We can think of this step as defining the structure of the model that we want to use. Then, we iteratively train the model.”

They are suitable for prediction, classification, and clustering.

Limitations

1. The number of variables must not high.
2. Enough data must be available (this model is good as much as accessible data - it is static and must be updated)

“A UNIT AND ITS ACTIVATION FUNCTION

Combination function + transfer function = activation function

Combination function: combines all the input into a single value, usually as a weighted summation.

Transfer function: calculated the output value from the result of the combination function

Output: the result is one output value, usually between -1 and 1" (lecture 13).

Training the network builds a model which can then be used to estimate the target value for unknown examples.

The process of training the network is actually the process of adjusting weights inside it to arrive at the best combination of weights for making the desired predictions. The network starts with a random set of weights, so it initially performs very poorly. However, by reprocessing the training set over and over and adjusting the internal weights each time to reduce the overall error, the network gradually does a better and better job of approximating the target values in the training set. When the approximations no longer improve, the network stops training.

We continue training steps until we make sure that there is not any other improvement. We are playing with the weights we have.

If one input variable have a value much more than others, then the neural network make the next worth repetitions that can be consumed for learning, for reducing its weight (for making its effect on output less). This is the first pattern the network learns, and because it is not very useful, it is better that we change weights ourselves.

Training is a process of iterating through the training set to adjust the weights.

Each iteration is sometimes called a generation.

Once the network has been trained, the performance of each generation must be measured on the validation set. Typically, earlier generations of the network perform better on the validation set than the final network (which was optimized for the training set).

Each of generations (outputs of each repetition) are compared with validation set. Usually the first repetitions are better, because of overfitting. Unfortunately it is the property of decision trees and neural networks that they can estimate each function regardless of its complication. But linear and logistic regressions do not have this property, because they assume specific forms of underlying functions, from the first.

"NEURAL NETWORKS CAN HANDLE ALL LEVELS OF MEASUREMENT: for both inputs and outputs, all ordinal, nominal, and interval. So we can use this model for our problem."

Results tend to be best if the target variable is relatively evenly distributed in the learning set. So it is a good idea to use a transform node.

"CHOOSING A NETWORK ARCHITECTURE"

Number of hidden layers

Always fit a model without hidden layers (= a GLM) as a baseline model. Regression models are powerful

One hidden layer will typically do. Prediction can only in special cases be improved by having more layers

Number of hidden units

Depends on the number of inputs, the number of cases in the training set, the complexity of the patterns, and the noise in the target variable.

Rule of thumb: start with zero and increase until the error in the validation set increases

CASCADE CORRELATION NETWORKS

Cascade correlation networks have a “self-organizing” architecture: hidden units are added step by step (while all previously trained weights and biases are frozen) until overfitting is detected in the validation sample

BLOCK NETWORKS

In block networks, new hidden units are added as new layers in the network, according to which activation function provides the most benefit.

The process continues until overfitting is detected in the validation sample.

FUNNEL NETWORKS

In a funnel network, hidden units are added step by step to form a funnel pattern, and are selected according to which activation function provides the most benefit.

The process continues until overfitting is detected in the validation sample” (lecture 13).

“Because neural networks are so flexible, SAS Enterprise Miner has two nodes that fit neural network models: the Neural Network node and the AutoNeural node” (Getting Started - p 39).

“We use three types of nodes in the diagram:

1. Neural Network

The Neural Network node trains a specific neural network configuration; this node is best used when you know a lot about the structure of the model that you want to define.

2. AutoNeural

The AutoNeural node searches over several network configurations to find one that best describes the relationship in a data set and then trains that network.

3. DMNeural

The DMNeural node is another modeling node that you can use to fit an additive nonlinear model. The additive nonlinear model uses bucketed principal components as inputs to predict a binary or an interval target variable with automatic selection of an activation function” (Data Mining Using SAS EM).

We have binary target value, so we are going to use “bucketed”.

“The unit of a neural network combines its inputs into a single value, which it then transforms to produce the output; these together are called the activation function.

1. Combination function

The activation function has two parts. The first part is the combination function that merges all the inputs into a single value. Each input into the unit has its own weight. The most common combination function is the weighted sum. (The combination function in SAS EM)

1.1. Nominal or interval targets: General Linear combination function

1.2. Ordinal targets: Linear-Equal slopes

The combination function that is applied depends on the level of measurements of the target variable.

2. Transfer function

The second part of the activation function is the transfer function, which gets its name from the fact that it transfers the value of the combination function to the output of the unit. (Activation Function in SAS EM)

Three typical transfer functions: the sigmoid (logistic), linear, and hyperbolic tangent functions.

The activation function depends on the combination function and the level of measurements of the target variable.

The selection of the activation function must conform to the range of the target values, which corresponds to the distribution of the target variable even after standardization is applied" (Data Mining Techniques).

"2.1. Interval: no default

2.2. Nominal or binary: softmax

2.3. Ordinal: logistic

3. Error function

3.1. Continuous targets: normal error function or sum-of-squares error

3.2. Categorical targets: multiple Bernoulli error function" (Data Mining using SAS EM).

In our case, target is binary, so we are going to use the second case.

Neural Networks Nodes:

Node 1:

Model Selection Criterion: Misclassification

Optimization:

Maximum iteration: 5

Maximum time: 5 minutes

Network:

Architecture: Generalized Linear Model

GLIM: Generalized Linear Model architecture – has no hidden layers and the input and target variables are not standardized. This design is similar to least-squares regression modeling.

(Neural network modeling using SAS EM - p 160)

Target Layer Combination Function: linear

Target Layer Activation Function: softmax

Target Layer Error Function: Bernoulli

Node 2:

Network:

Architecture: Multilayer Perception (the default to the EM - MLP without any hidden unit or hidden layer is GLIM)

Number of hidden units: 2

Node 3:

Network:

Number of hidden units: 4

AutoNeural Nodes

Node 1:

Architecture: single layer

Maximum iteration: 5

Number of hidden units: 1

Node 2:

Architecture: Block layer

Node 3:

Architecture: Funnel layer

Node 4:

Architecture: Cascade layer

DMNeural Node

Results:

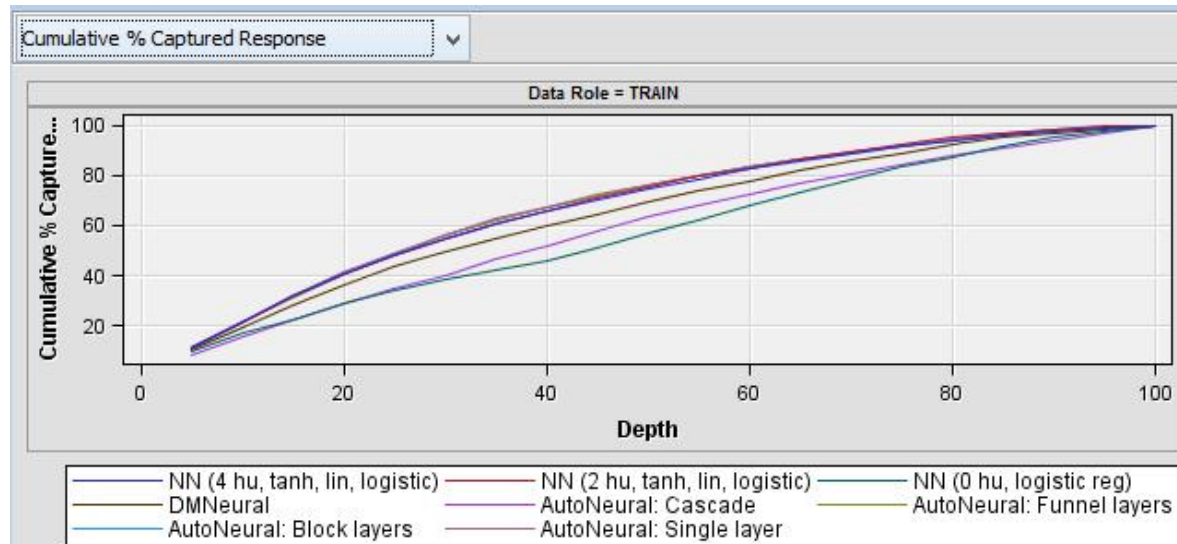
1. fit Statistics

Model Description	Target Variable	Target Label	Selection Criterion: Valid: Misclassification Rate
Model Description			
AutoNeural: Block layers	working		0.235466
AutoNeural: Funnel layers	working		0.236251
AutoNeural: Single layer	working		0.23771
NN (4 hu, tanh, lin, logistic)	working		0.24927
NN (2 hu, tanh, lin, logistic)	working		0.252413
DMNeural	working		0.307183
AutoNeural: Cascade	working		0.400224
NN (0 hu, logistic reg)	working		0.403928

As we see autoNeural has better performance to predict the “need repair” pumps. Also we can see that the difference between various Neural Networks, regarding the number of hidden units is high, and increasing the hidden units gives a better performance.

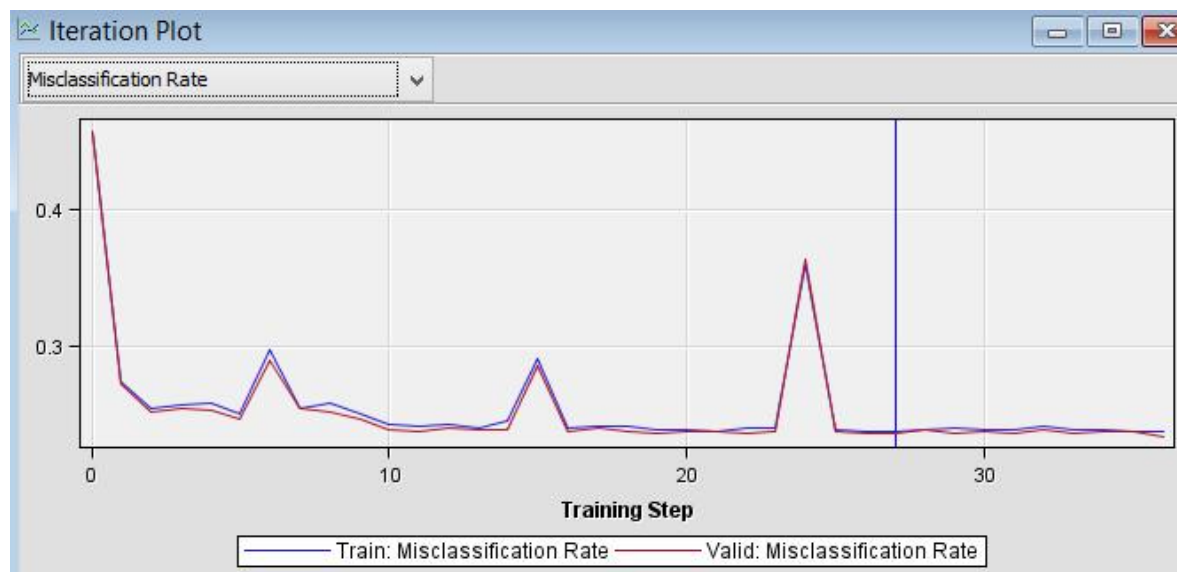
Better performance of AutoNeural nodes than the Neural Network nodes happens especially when we have a binary target variable.

2. cumulative % captured response



That means if we sort the pumps with our model (based on the probability of needing repair), then how much better our model can identify them in comparison with randomly selecting them. The difference is especially high between 30 to 40 percent of selected cases.

3. best model - iteration history



When we are going to detect the lowest misclassification. It depends on the limitation of time. So it is very natural that after 27 repetition the misclassification rate goes even less.

...

3.2. Model selection approach

We select the models which have the ability of learning from input variables, and predict a binary target variable. Our comparison among various models are the Misclassification measure.

3.3. Final Model

3.3.1. Overall predictive accuracy

3.3.2. Observed versus predicted target values

3.3.3. Improvement over baseline

4. Discussion

Discussion has come during developing models.

4.1. Assessment of model performance

Assessment of models' performance has come during developing models.

4.2. Contribution to the solution of the business problem

The main goal in this problem is decreasing the misclassification rate for identifying the pumps that need repair. After that the main topic is finding the input variables with more correlation with target variable. So the government can decide about these kind of input variables.

4.3. Deployment recommendations

1. neural networks - autoNeural: adding hidden units maybe improve the performance of the models. Also changing the activation functions and also combination functions is recommended for investigating probable improvements.

2. regression: changing some properties in forward and backward stepwise selection can improve the performance of the model. These properties include:

Two-Factor interactions: yes

Polynomial terms: yes

Polynomial degree: degrees more than 2

Regression type: logistic

Results show that logistic regressions are better than linear ones. So we suggest logistic regressions would be applied to forward and backward stepwise, LARS and LASSO ...

Also because cubic models are better than quadratic ones here (in our case), we suggest increasing the degree of regression.

3. nearest neighbor: regarding on the complexity of the relationships between input variables, sample size, being balanced sample sets, the number of cases relatively to the number of variables ... there is an optimum value for k . and in our case we find it 8. But maybe manipulating other factors (in addition to k) can change performance.

4. decision trees: we know the number of input variables in our case causes binary trees does not behave well. So we suggest using multiway trees and higher depth for decision trees.

4. bagging and boosting: behave better in our case regarding Random Forest and Gradient Boosting. We think it is because the proportion for Random Forest and Gradient Boosting, and the proportion must be increased.

After that we know shrinkage factor must be set regarding the number of iterations, so maybe setting the shrinkage factor to 0.1 and increasing the number of iterations can improve the Gradient Boosting.

4.4. Recommended follow-up activities

Finding another relationships among existing input variables (combining them), can help finding more accurate predictions about the target variable. Also converting quality (nominal) variables to more quantity (interval, ratio) variables can help to improve misclassification rate.