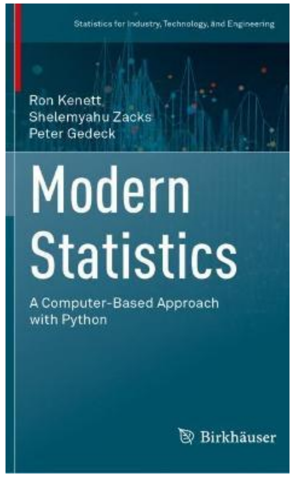
A Biomed Data Analyst Training Program

Supervised learning

Professor Ron S. Kenett

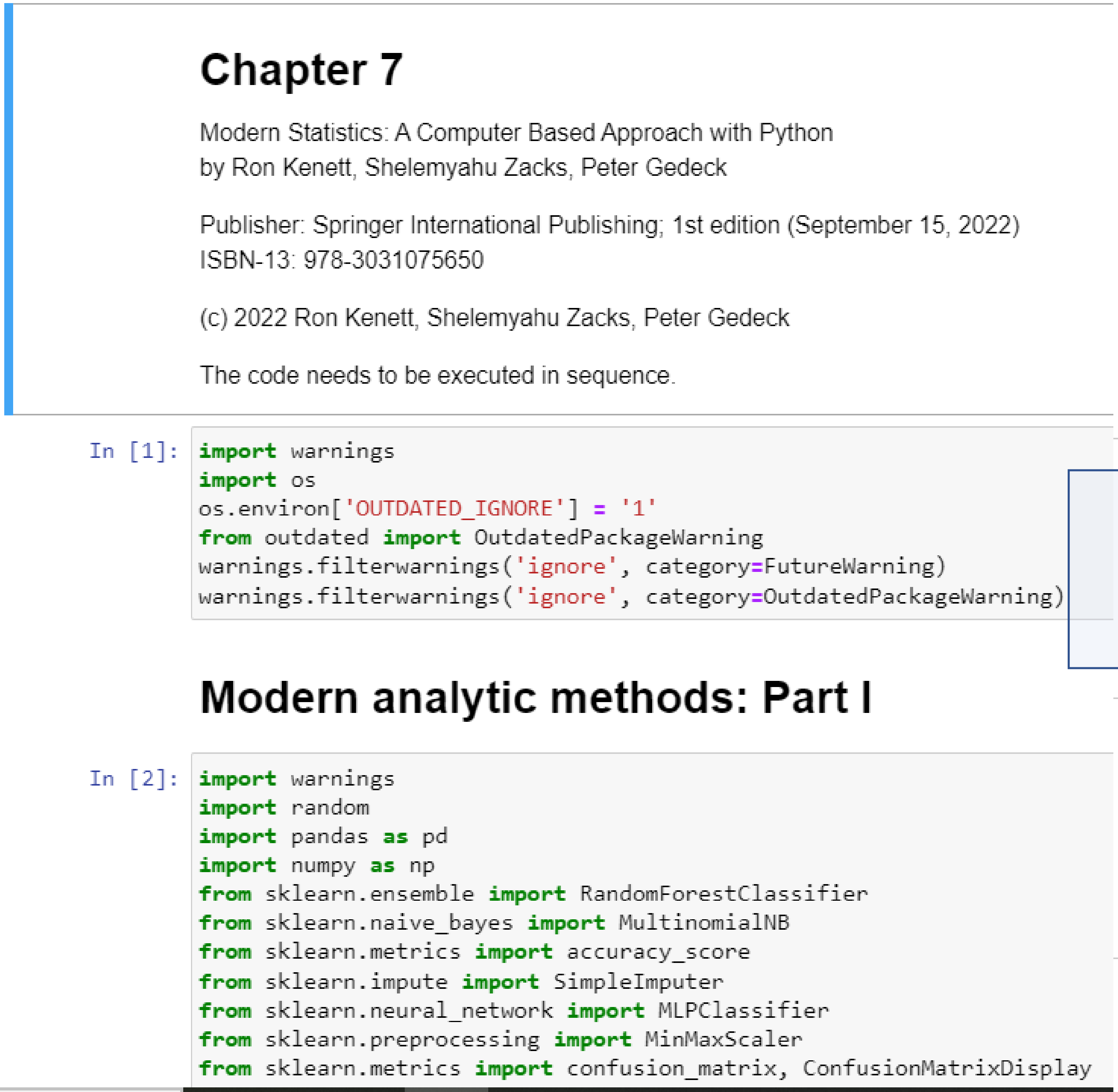

Statistics for Industry, Technology, and Engineering
Ron Kenett
Shelemyahu Zacks
Peter Gedeck
Modern Statistics
A Computer-Based Approach with Python
Birkhäuser

Chapter 7
Modern Analytic Methods: Part I

Preview This chapter is a door opener to computer age statistics. It covers a range of supervised and unsupervised learning methods and demonstrates their use in various applications.

7.1 Introduction to Computer Age Statistics

Big data and data science applications have been facilitated by hardware developments in computer science. As data storage began to increase, more advanced software was required to process it. This led to the development of cloud computing and distributed computing. Parallel machine processing was enhanced by the development of Hadoop, based on off-the-shelf Google File System (GFS) and Google MapReduce, for performing distributed computing.



Chapter 7

Modern Statistics: A Computer Based Approach with Python
by Ron Kenett, Shelemyahu Zacks, Peter Gedeck

Publisher: Springer International Publishing; 1st edition (September 15, 2022)
ISBN-13: 978-3031075650

The code needs to be executed in sequence.

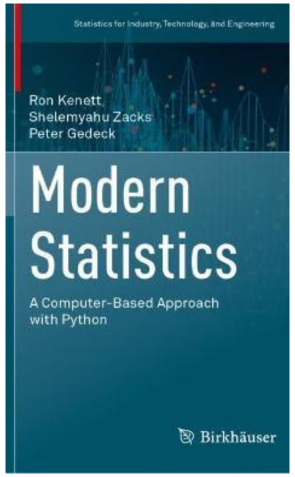
In [1]:
```python
import warnings
import os
os.environ['OUTDATED_IGNORE'] = '1'
from outdated import OutdatedPackageWarning
warnings.filterwarnings('ignore', category=FutureWarning)
warnings.filterwarnings('ignore', category=OutdatedPackageWarning)
```

Modern analytic methods: Part I

In [2]:
```python
import warnings
import random
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.impute import SimpleImputer
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

Modern Analytic Methods: Part I

Ron Kenett, Shelemyahu Zacks, Peter Gedeck
Pages 361-393



Statistics for Industry, Technology, and Engineering

Ron Kenett
Shelemyahu Zacks
Peter Gedeck

Modern Statistics

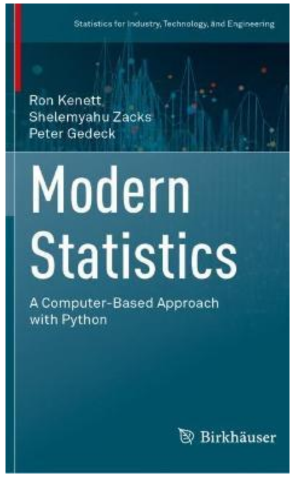
A Computer-Based Approach with Python

Birkhäuser

Chapter 7
Modern Analytic Methods: Part I

7.5 Decision Trees

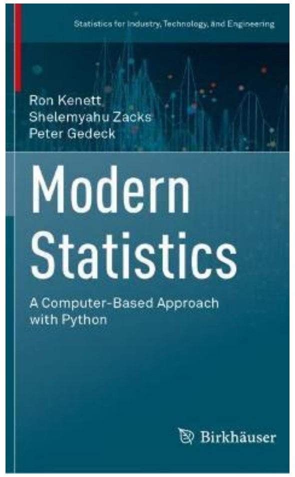
Partition models, also called decision trees, are non-parametric tools used in supervised learning in the context of classification and regression. In supervised learning you observe multiple covariate and one or more target variables. The goal is to predict or classify the target using the values of covariates. Decision trees are based on splits in covariates or predictors that create separate but homogeneous groups. Splits are not sensitive to outliers but are based on a "greedy" one -step look ahead, without accounting for overall performance. Breiman et al. (1984) implement a decision tree procedure called CART (Classification And Regression Trees). Other procedures are C4.5 and CHAID (Chi-square Automatic Interaction


Statistics for Industry, Technology, and Engineering
Ron Kenett
Shelemyahu Zacks
Peter Gedeck
Modern Statistics
A Computer-Based Approach with Python
Birkhäuser

Chapter 7
Modern Analytic Methods: Part I

Example 7.2 Data set **SENSORS.csv** consists of 174 measurements from 63 sensors tracking performance of a system under test. Each test generates values for these 63 sensors and a status determined by the automatic test equipment. The distribution of the test results is presented in Fig. 7.5. Our goal is to predict the outcome recorded by the testing equipment, using sensor data. The test results are coded as Pass (corresponding to "Good," 47% of the observations) and Fail (all other categories, marked in grey). The column **Status** is therefore a dichotomized version of the column **Test result**.

```
sensors = mistat.load_data('SENSORS.csv')
dist = sensors['testResult'].value_counts()
dist = dist.sort_index()
ax = dist.plot.bar(color='lightgrey')
ax.patches[dist.index.get_loc('Good')].set_facecolor('black')
plt.show()
```



Chapter 7
Modern Analytic Methods: Part I

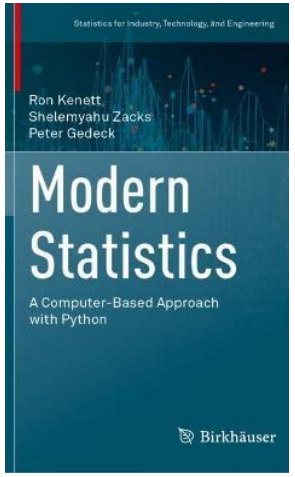
The goal is to predict the outcome recorded by the testing equipment, using sensor data. We can use `scikit-learn` for this. It has decision tree implementations for classification and regression. Here, we create a classification model for Pass-Fail using the 67 sensors.

```
from sklearn.tree import DecisionTreeClassifier, plot_tree, export_text

predictors = [c for c in sensors.columns if c.startswith('sensor')]
outcome = 'status'
X = sensors[predictors]
y = sensors[outcome]

# Train the model
clf = DecisionTreeClassifier(ccp_alpha=0.012, random_state=0)
clf.fit(X, y)

# Visualization of tree
plot_tree(clf, feature_names=list(X.columns))
plt.show()
```



Statistics for Industry, Technology, and Engineering
Ron Kenett
Shelemyahu Zacks
Peter Gedeck
Modern Statistics
A Computer-Based Approach with Python
Birkhäuser

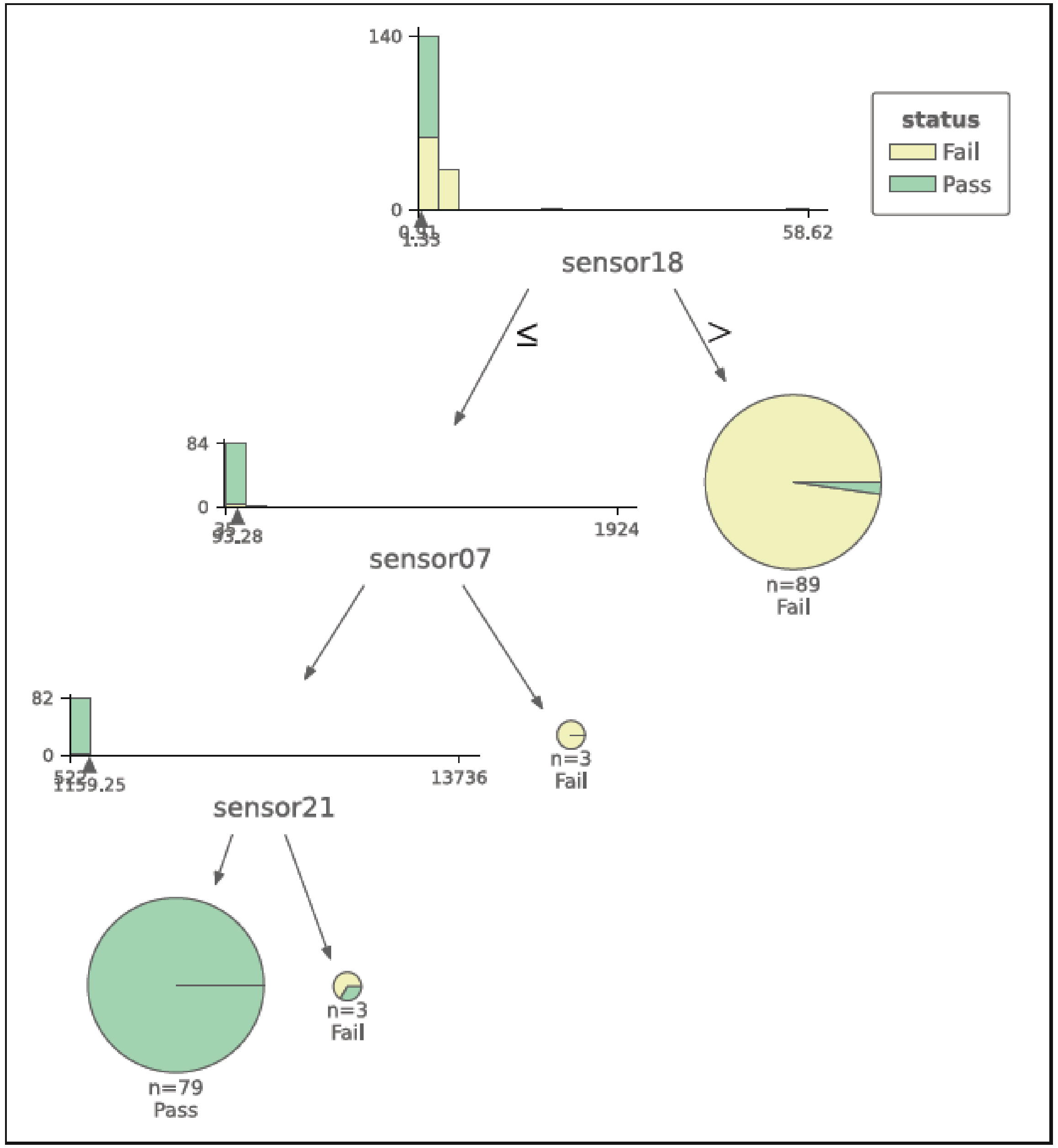
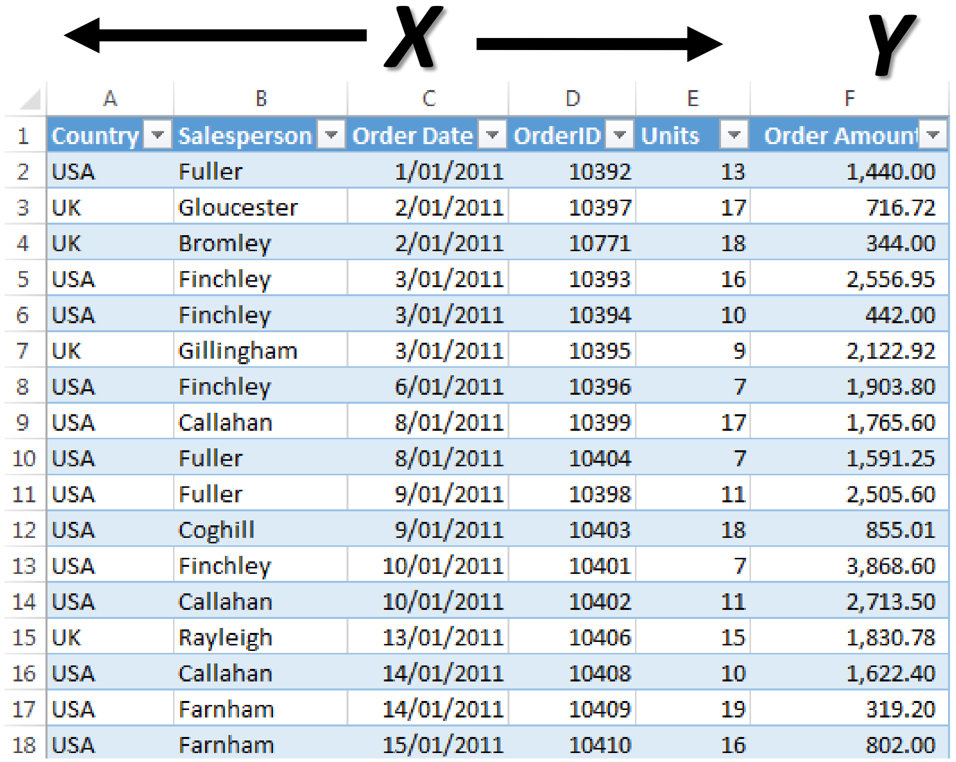
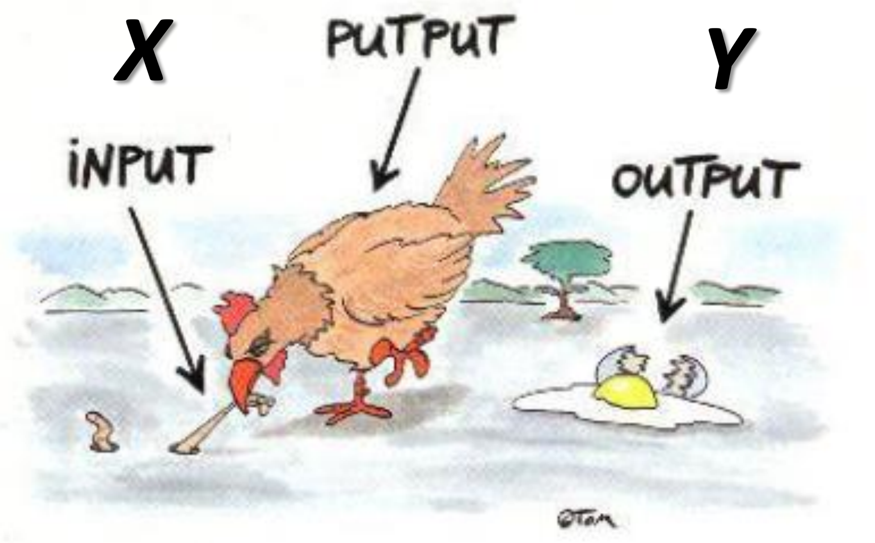


Fig. 7.7 Decision tree visualization of classification tree

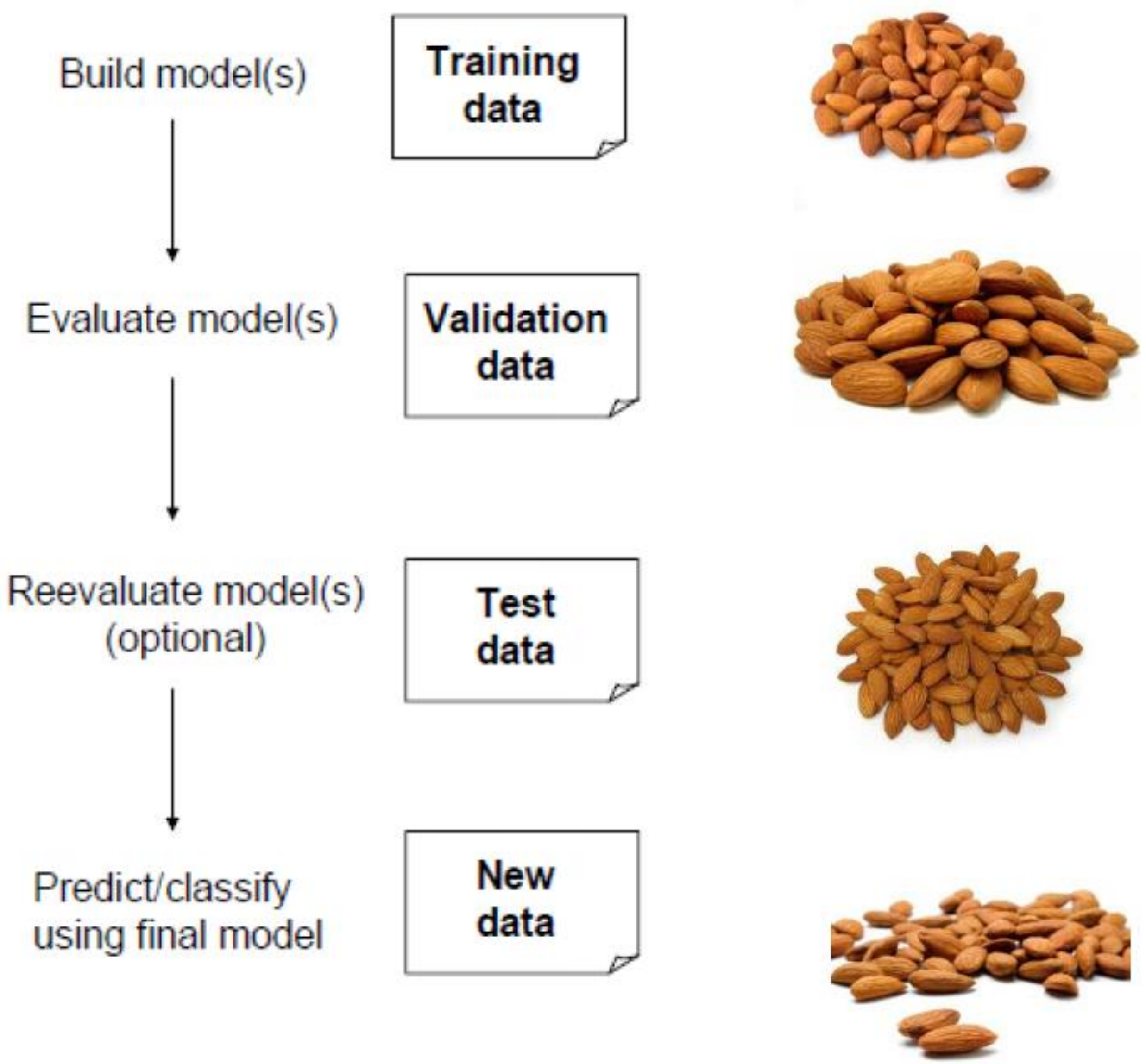
Supervised Learning



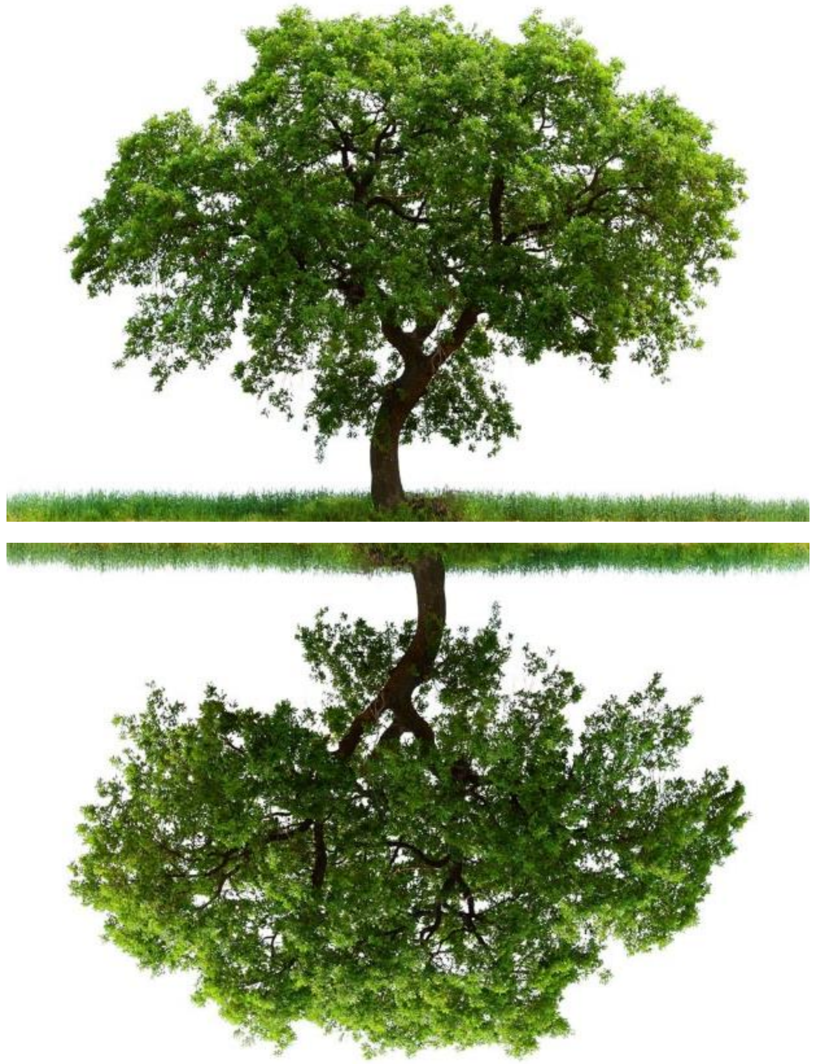
	A	B	C	D	E	F
1	Country	Salesperson	Order Date	OrderID	Units	Order Amount
2	USA	Fuller	1/01/2011	10392	13	1,440.00
3	UK	Gloucester	2/01/2011	10397	17	716.72
4	UK	Bromley	2/01/2011	10771	18	344.00
5	USA	Finchley	3/01/2011	10393	16	2,556.95
6	USA	Finchley	3/01/2011	10394	10	442.00
7	UK	Gillingham	3/01/2011	10395	9	2,122.92
8	USA	Finchley	6/01/2011	10396	7	1,903.80
9	USA	Callahan	8/01/2011	10399	17	1,765.60
10	USA	Fuller	8/01/2011	10404	7	1,591.25
11	USA	Fuller	9/01/2011	10398	11	2,505.60
12	USA	Coghill	9/01/2011	10403	18	855.01
13	USA	Finchley	10/01/2011	10401	7	3,868.60
14	USA	Callahan	10/01/2011	10402	11	2,713.50
15	UK	Rayleigh	13/01/2011	10406	15	1,830.78
16	USA	Callahan	14/01/2011	10408	10	1,622.40
17	USA	Farnham	14/01/2011	10409	19	319.20
18	USA	Farnham	15/01/2011	10410	16	802.00



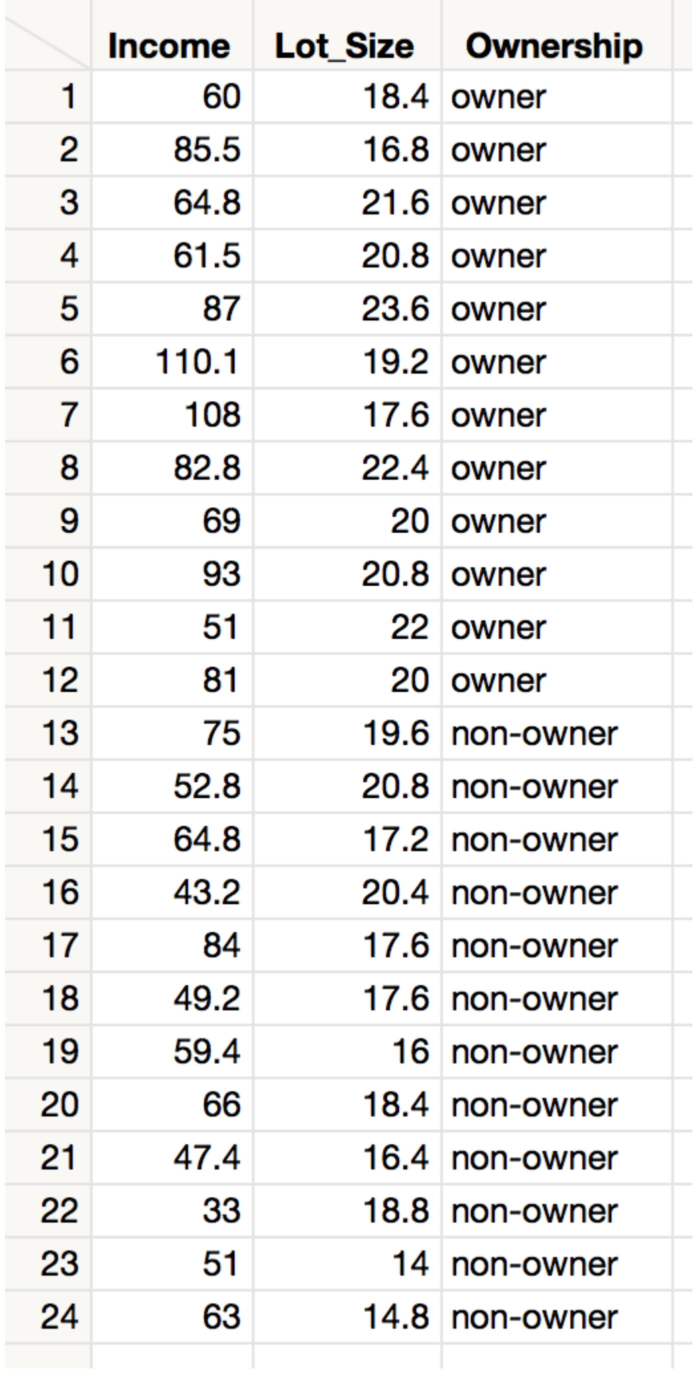
Data Stratification



Decision Trees



Income



	Income	Lot_Size	Ownership
1	60	18.4	owner
2	85.5	16.8	owner
3	64.8	21.6	owner
4	61.5	20.8	owner
5	87	23.6	owner
6	110.1	19.2	owner
7	108	17.6	owner
8	82.8	22.4	owner
9	69	20	owner
10	93	20.8	owner
11	51	22	owner
12	81	20	owner
13	75	19.6	non-owner
14	52.8	20.8	non-owner
15	64.8	17.2	non-owner
16	43.2	20.4	non-owner
17	84	17.6	non-owner
18	49.2	17.6	non-owner
19	59.4	16	non-owner
20	66	18.4	non-owner
21	47.4	16.4	non-owner
22	33	18.8	non-owner
23	51	14	non-owner
24	63	14.8	non-owner

Splitting on Continuous Variables

- Order records according to one variable, say lot size
- Split at the first value
- Measure the dissimilarity between the two subsets
- Split at the next value, and continue
- Repeat for the other variable(s)
- For all variables, the split value that drives the greatest dissimilarity in propensities (or probabilities) is selected as the split point

Shmueli, G., Bruce, P., Stephens, M. and Patel, N. (2016) Data Mining for Business Analytics: Concepts, Techniques, and Applications with JMP Pro, Wiley, USA https://www.wiley.com/en-us/Data+Mining+for+Business+Analytics%3A+Concepts%2C+Techniques%2C+and+Applications+with+JMP+Pro-p-9781118877524

Splitting on Categorical Variables

- Examine all possible ways in which the categories can be split.
- E.g., nominal categories A, B, C can be split 3 ways

  {A} and {B, C}

  {B} and {A, C}

  {C} and {A, B}

- With many categories, # of potential splits becomes huge

Splitting on Categorical Variables

- For ordinal data (ordered categories) there is an option for the splits to respect ordering
- Example: An ordinal predictor takes on the values 1, 2, 3, or 4
- The data can be split 3 ways:

  {1} and {2, 3, 4}

  {1, 2} and {3, 4}

  {1, 2, 3} and {4}

Gini Index

Gini Index for rectangle *A* containing *m* records

$$I(A) = 1 - \sum_{k=1}^{m} p_k^2$$

p = proportion of cases in rectangle *A* that belong to class *k*

- I(A) = 0 when all cases belong to same class
- Max value when all classes are equally represented (= 0.50 in binary case)

Entropy

p = proportion of cases (out of *m*) in rectangle *A* that belong to class *k*

$$entropy\ (A) = -\sum_{k=1}^{m} p_k\, log_2(p_k)$$

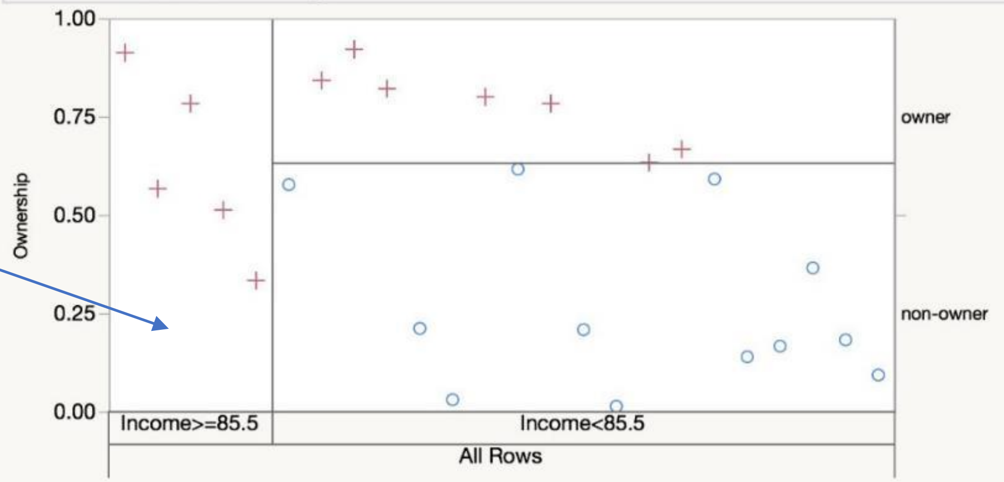
- Entropy ranges between 0 (most pure) and $\log_2(m)$ (equal representation of classes)

Impurity and Recursive Partitioning

- Obtain overall impurity measure (weighted avg. of individual rectangles)
- At each successive stage, compare this measure across all possible splits in all variables
- Choose the split that reduces impurity the most
- Chosen split points become nodes on the tree

Rectangle A

Partition for Ownership



RSquare	N	Number of Splits
0.222	24	1

אין

יש

All Rows

Count	G^2	LogWorth
24	33.271065	1.408425

Level	Rate	Prob
non-owner	0.5000	0.5000
owner	0.5000	0.5000

Income>=85.5

Count	G^2
5	0

Level	Rate	Prob
non-owner	0.0000	0.0833
owner	1.0000	0.9167

Candidates

Term	Candidate G^2	LogWorth	Cut Point
Income	0	0	.
Lot_Size	0	0	.

Constrained by minimum size

Income<85.5

Count	G^2
19	25.00818

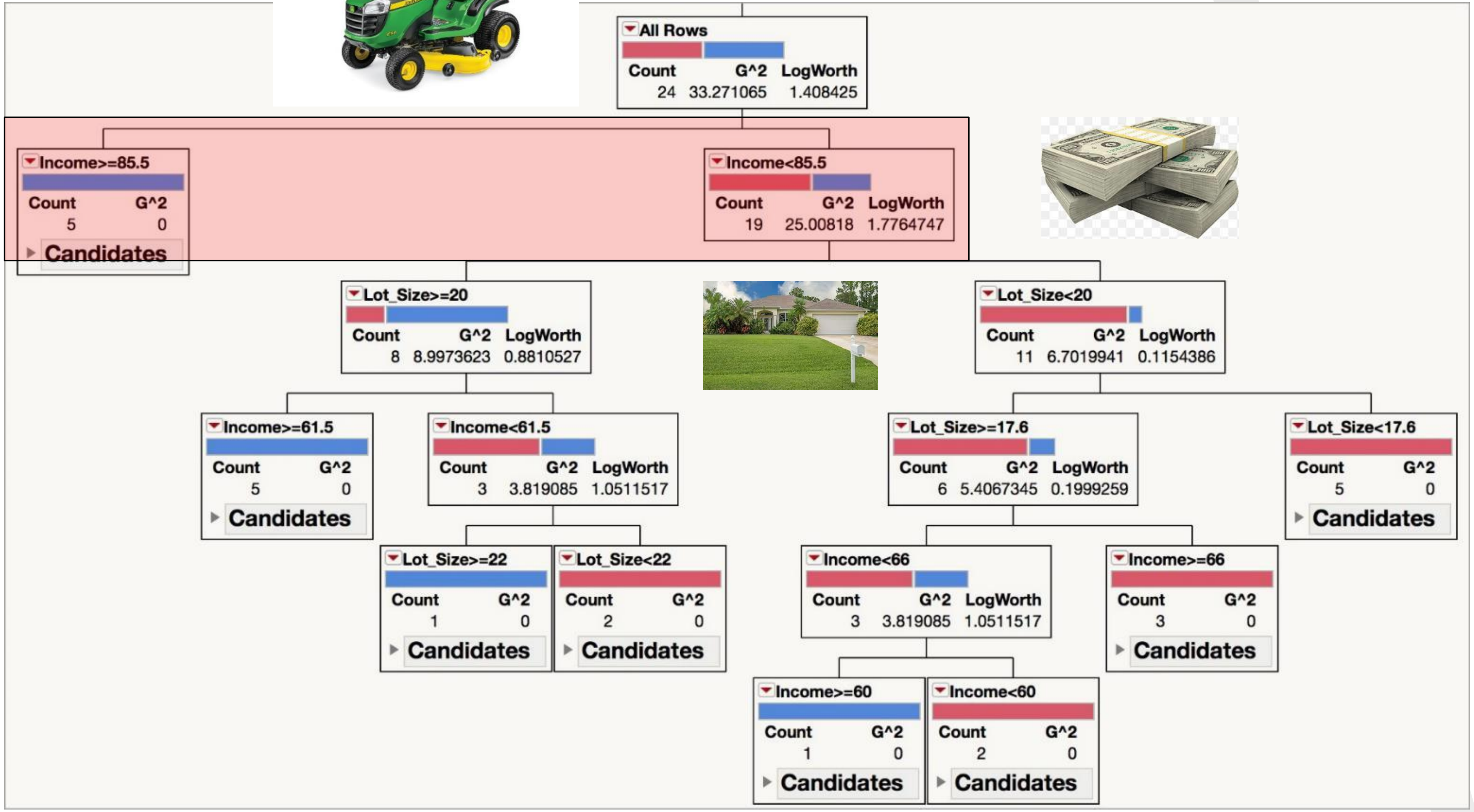
Level	Rate	Prob
non-owner	0.6316	0.6250
owner	0.3684	0.3750

Candidates

Term	Candidate G^2		LogWorth	Cut Point
Income	3.821653700		0.496857898	60
Lot_Size	9.308823071	*	1.776474652	20

Gini and Entropy measures are not available in JMP. JMP uses measures of dissimilarity (G^2 and Sum of Squares) rather than measures of impurity

Income



All Rows

Count	G^2	LogWorth
24	33.271065	1.408425

Income>=85.5

Count	G^2
5	0

Candidates

Income<85.5

Count	G^2	LogWorth
19	25.00818	1.7764747

Lot_Size>=20

Count	G^2	LogWorth
8	8.9973623	0.8810527

Lot_Size<20

Count	G^2	LogWorth
11	6.7019941	0.1154386

Income>=61.5

Count	G^2
5	0

Candidates

Income<61.5

Count	G^2	LogWorth
3	3.819085	1.0511517

Lot_Size>=17.6

Count	G^2	LogWorth
6	5.4067345	0.1999259

Lot_Size<17.6

Count	G^2
5	0

Candidates

Lot_Size>=22

Count	G^2
1	0

Candidates

Lot_Size<22

Count	G^2
2	0

Candidates

Income<66

Count	G^2	LogWorth
3	3.819085	1.0511517

Income>=66

Count	G^2
3	0

Candidates

Income>=60

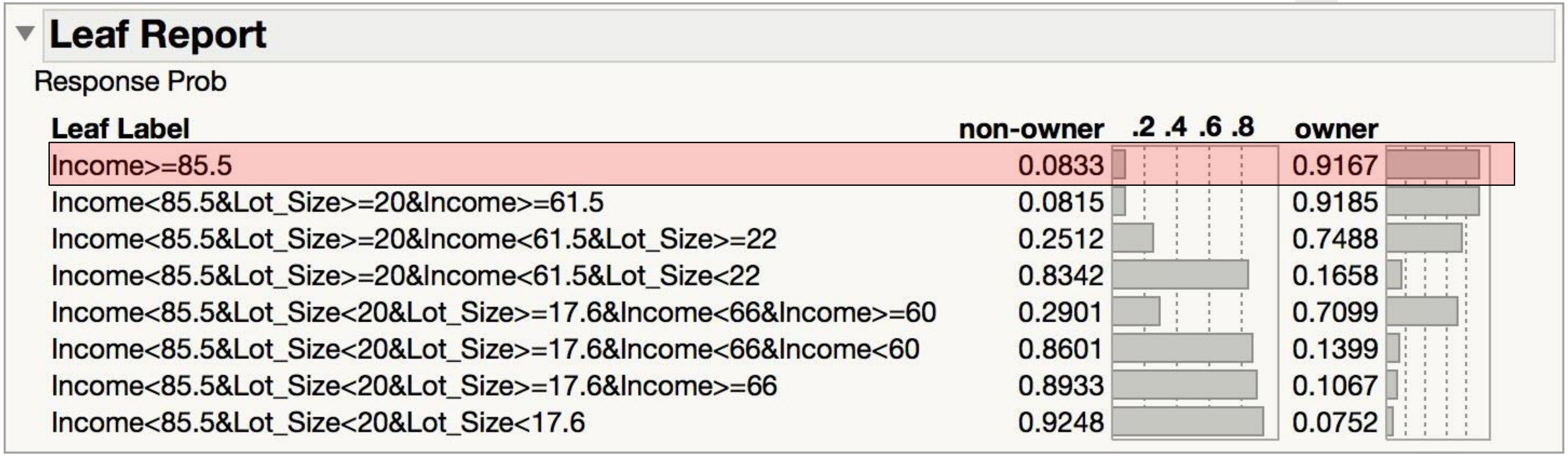
Count	G^2
1	0

Candidates

Income<60

Count	G^2
2	0

Candidates



Leaf Report

Response Prob

Leaf Label	non-owner	owner
Income>=85.5	0.0833	0.9167
Income<85.5&Lot_Size>=20&Income>=61.5	0.0815	0.9185
Income<85.5&Lot_Size>=20&Income<61.5&Lot_Size>=22	0.2512	0.7488
Income<85.5&Lot_Size>=20&Income<61.5&Lot_Size<22	0.8342	0.1658
Income<85.5&Lot_Size<20&Lot_Size>=17.6&Income<66&Income>=60	0.2901	0.7099
Income<85.5&Lot_Size<20&Lot_Size>=17.6&Income<66&Income<60	0.8601	0.1399
Income<85.5&Lot_Size<20&Lot_Size>=17.6&Income>=66	0.8933	0.1067
Income<85.5&Lot_Size<20&Lot_Size<17.6	0.9248	0.0752

Tree Structure

- Split points become nodes on the tree
- Leaves are the terminal nodes (there are no further splits)
- Read down tree to derive the decision rule

  E.g., Income < 85.5, Lot Size is >= 20, and Income >=61.5 , the probability that a household is an owner is 0.9185.

- Records within each node are from the training data (validation data are not used in building the tree)
- Default cutoff = 0.5 is used for classification

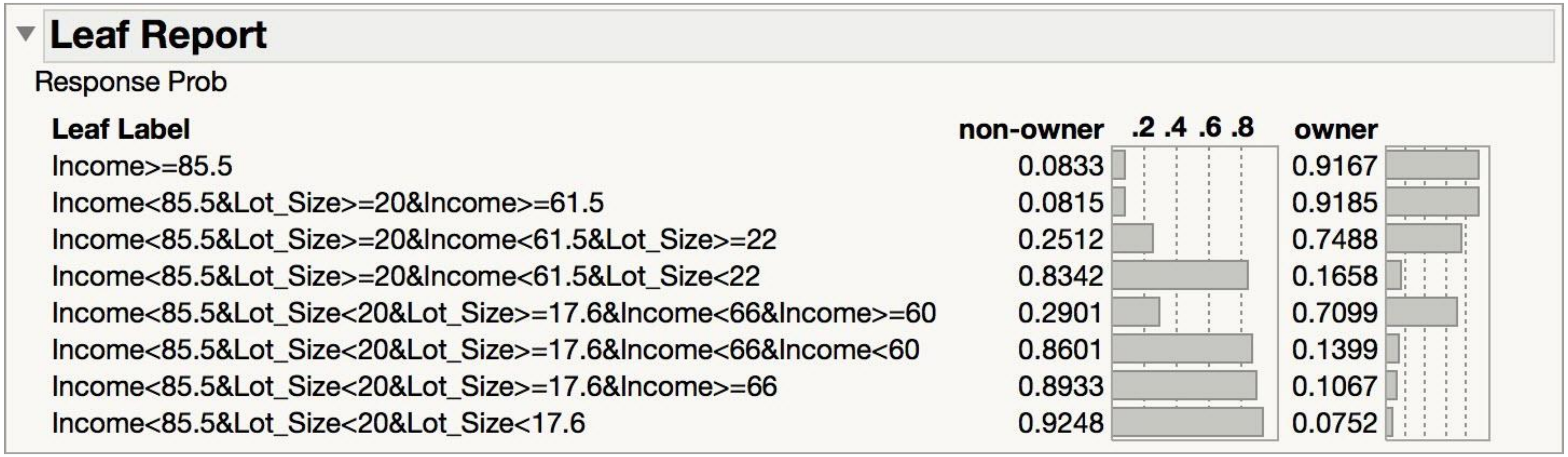
  In the previous example, the record would be classified as an owner.

The Riding Mowers

The leaf report provides a summary the splits

It displays the rules for classifying outcomes

For example, If Income < 85.5, Lot Size is < 17.6, the probability that a household is an owner is 0.0752. This record will be classified as a non-owner.



Leaf Report

Response Prob

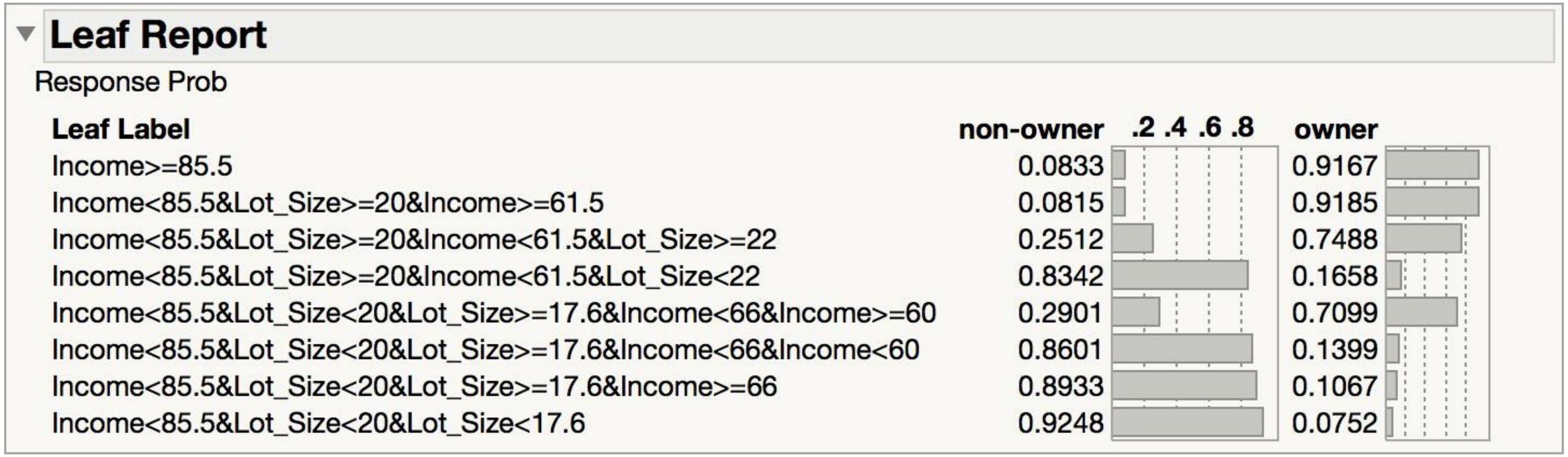
Leaf Label	non-owner	owner
Income>=85.5	0.0833	0.9167
Income<85.5&Lot_Size>=20&Income>=61.5	0.0815	0.9185
Income<85.5&Lot_Size>=20&Income<61.5&Lot_Size>=22	0.2512	0.7488
Income<85.5&Lot_Size>=20&Income<61.5&Lot_Size<22	0.8342	0.1658
Income<85.5&Lot_Size<20&Lot_Size>=17.6&Income<66&Income>=60	0.2901	0.7099
Income<85.5&Lot_Size<20&Lot_Size>=17.6&Income<66&Income<60	0.8601	0.1399
Income<85.5&Lot_Size<20&Lot_Size>=17.6&Income>=66	0.8933	0.1067
Income<85.5&Lot_Size<20&Lot_Size<17.6	0.9248	0.0752

$$\text{Prob}_i = \frac{n_i + \text{prior}_i}{\sum(n_i + \text{prior}_i)}$$

where the summation is across all response levels; n_i is the number of observations at the node for the ith response level; and prior_i is the prior probability for the ith response level, calculated as follows:

$$\text{prior}_i = \lambda p_i + (1-\lambda)P_i$$

where p_i is the prior_i from the parent node, P_i is the Prob_i from the parent node, and λ is a weighting factor currently set at 0.9.



Leaf Report

Response Prob

Leaf Label	non-owner	owner
Income>=85.5	0.0833	0.9167
Income<85.5&Lot_Size>=20&Income>=61.5	0.0815	0.9185
Income<85.5&Lot_Size>=20&Income<61.5&Lot_Size>=22	0.2512	0.7488
Income<85.5&Lot_Size>=20&Income<61.5&Lot_Size<22	0.8342	0.1658
Income<85.5&Lot_Size<20&Lot_Size>=17.6&Income<66&Income>=60	0.2901	0.7099
Income<85.5&Lot_Size<20&Lot_Size>=17.6&Income<66&Income<60	0.8601	0.1399
Income<85.5&Lot_Size<20&Lot_Size>=17.6&Income>=66	0.8933	0.1067
Income<85.5&Lot_Size<20&Lot_Size<17.6	0.9248	0.0752

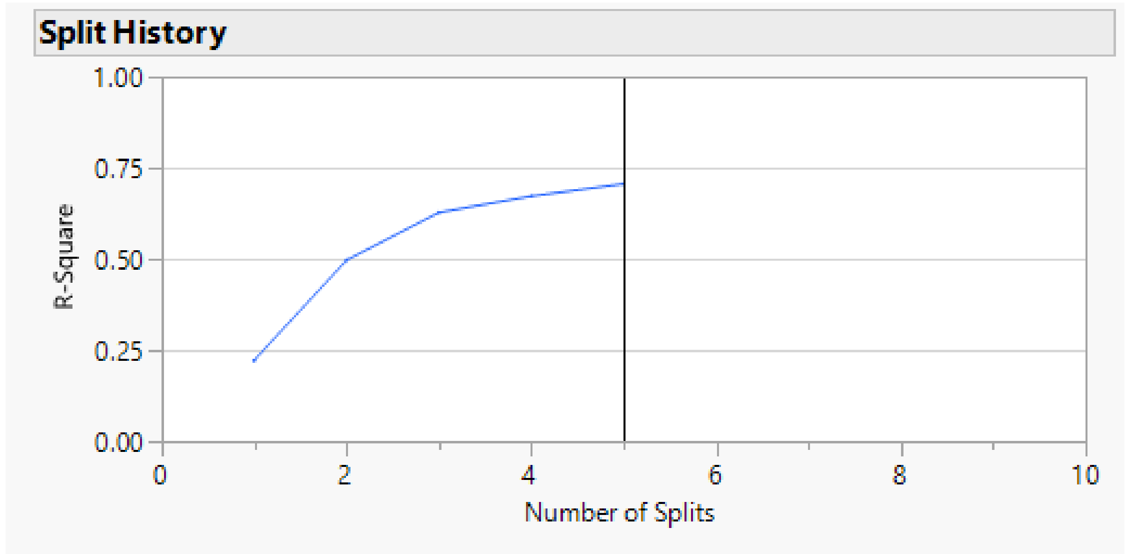
Stopping Tree Growth

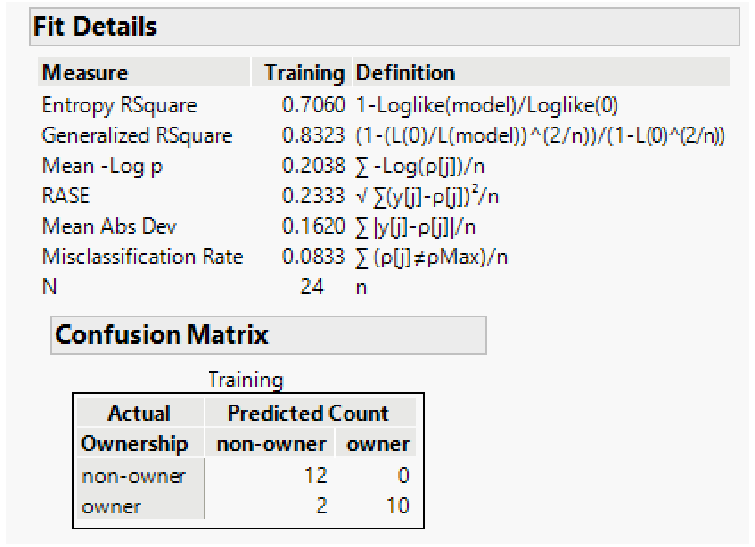
Natural end of process is 100% purity in each leaf

This **overfits** the data, which end up fitting noise in the data

Overfitting leads to low predictive accuracy of new data

Past a certain point, the error rate for the validation data starts to increase





Fit Details

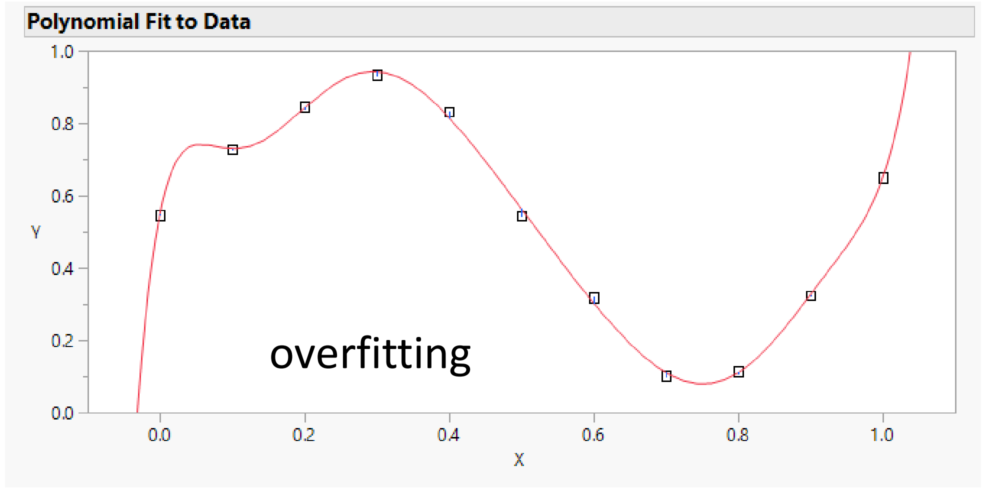
Measure	Training	Definition
Entropy RSquare	0.7060	1-Loglike(model)/Loglike(0)
Generalized RSquare	0.8323	(1-(L(0)/L(model))^(2/n))/(1-L(0)^(2/n))
Mean -Log p	0.2038	∑ -Log(ρ[j])/n
RASE	0.2333	√ ∑(y[j]-ρ[j])²/n
Mean Abs Dev	0.1620	∑ |y[j]-ρ[j]|/n
Misclassification Rate	0.0833	∑ (ρ[j]≠ρMax)/n
N	24	n

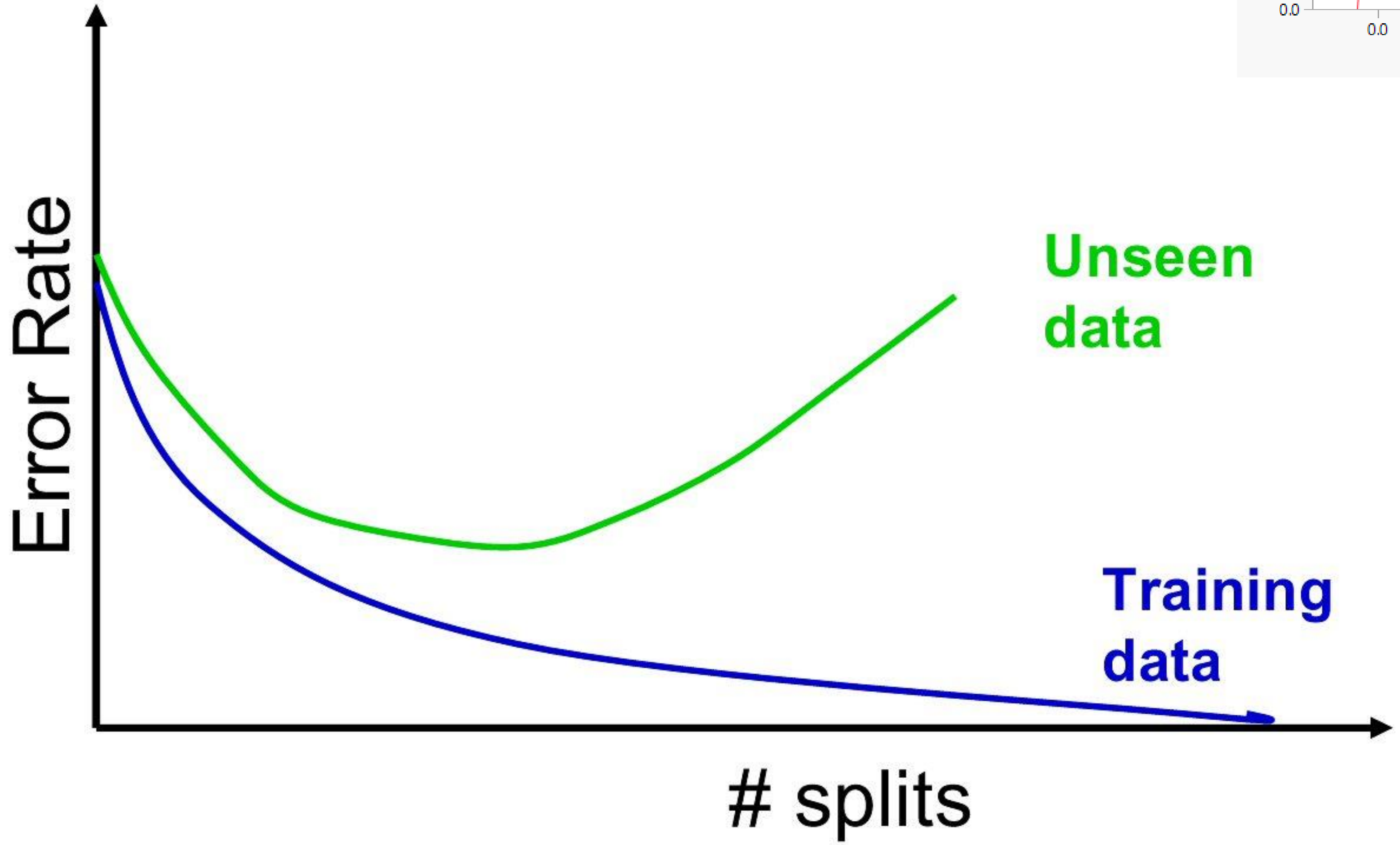
Confusion Matrix

Training

Actual	Predicted Count	
Ownership	non-owner	owner
non-owner	12	0
owner	2	10

Full Tree Error Rate





CART - Classification and regression trees

- CART lets tree grow to full extent, then prunes it back
- Idea is to find that point at which the validation error begins to rise
- Generate successively smaller trees by pruning leaves
- At each pruning stage, multiple trees are possible
- Use *cost complexity* to choose the best tree at that stage

Cost Complexity

$$CC(T) = Err(T) + \alpha\, L(T)$$

CC(T) = cost complexity of a tree

Err(T) = proportion of misclassified records

L(T) – size of tree

α = penalty factor attached to tree size (set by user)

Among trees of given size, choose the one with lowest CC

Do this for each size of tree

CART - Classification and regression trees

- Nonparametric (no probabilistic assumptions)
- Automatically performs variable selection
- Uses any combination of continuous/discrete variables
  - Very nice feature: ability to automatically bin massively categorical variables into a few categories (zip code, business class, make/model…)
- Invariant to monotonic transformations of predictive variable
- Unlike regression, not sensitive to outliers in predictive variables

CART Overview

- Classification and Regression Trees are an easily understandable and transparent method for predicting or classifying new records
- A tree is a graphical representation of a set of rules
- Trees must be pruned to avoid over-fitting of the training data
- As trees do not make any assumptions about the data structure, they usually require large samples

Shmueli, G., Bruce, P., Stephens, M. and Patel, N. (2016) Data Mining for Business Analytics: Concepts, Techniques, and Applications with JMP Pro, Wiley, USA https://www.wiley.com/en-us/Data+Mining+for+Business+Analytics%3A+Concepts%2C+Techniques%2C+and+Applications+with+JMP+Pro-p-9781118877524

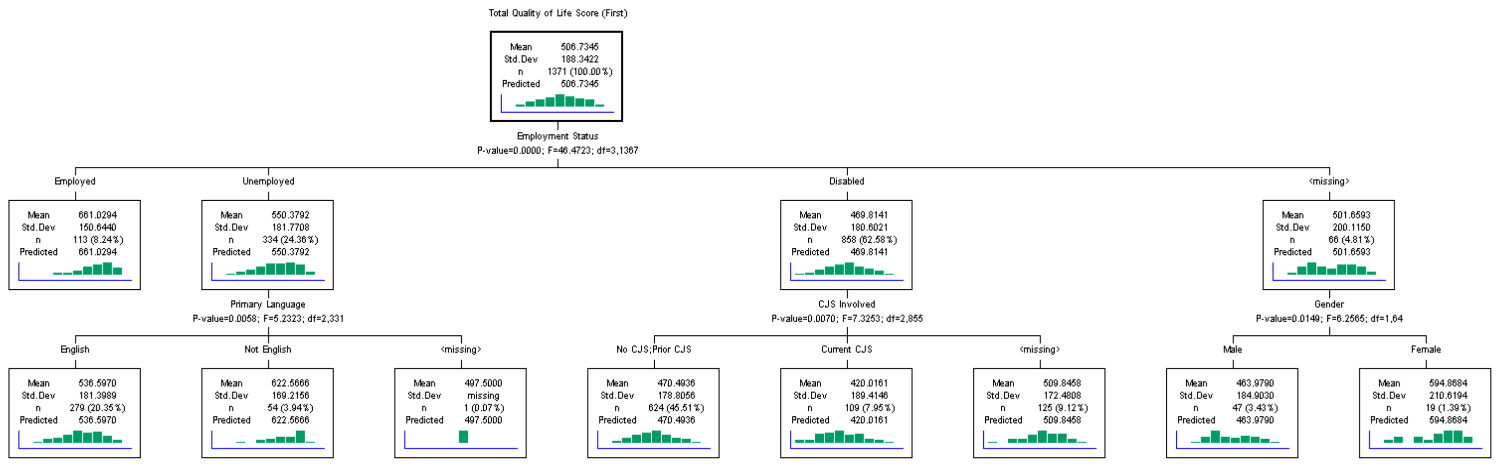
CHAID - Chi-squared automatic interaction detector

- CHAID, older than CART, uses chi-square statistical test to limit tree growth
- Splitting stops when purity improvement is not statistically significant

Shmueli, G., Bruce, P., Stephens, M. and Patel, N. (2016) Data Mining for Business Analytics: Concepts, Techniques, and Applications with JMP Pro, Wiley, USA https://www.wiley.com/en-us/Data+Mining+for+Business+Analytics%3A+Concepts%2C+Techniques%2C+and+Applications+with+JMP+Pro-p-9781118877524

CHAID - Chi-squared automatic interaction detector

- CHAID is a non-binary decision tree.
- The decision or split made at each node is still based on a single variable, but can result in multiple branches.
- The split search algorithm is designed for categorical variables.



Classification Trees: CART versus CHAID

- At each split, the CHAID algorithm looks for the predictor variable that if split, most "explains" the category response variable. In order to decide whether to create a particular split based on this variable, the CHAID algorithm tests a hypothesis regarding dependence between the split variable and the categorical response (using the chi-squared test for independence). Using a pre-specified significance level, if the test shows that the split variable and the response are independent, the algorithm stops the tree growth. Otherwise, the split is created, and the next best split is searched. In contrast, the CART algorithm decides on a split based on the amount of homogeneity within class that is achieved by the split. The split is reconsidered based on considerations of over-fitting.

- CHAID is most useful for analysis, whereas CART is more suitable for prediction. In other words, CHAID should be used when the goal is to describe or understand the relationship between a response variable and a set of explanatory variables, whereas CART is better suited for creating a model that has high prediction accuracy of new cases.

Limiting Tree Size

JMP uses a combination of limiting tree growth and pruning the tree after it has grown

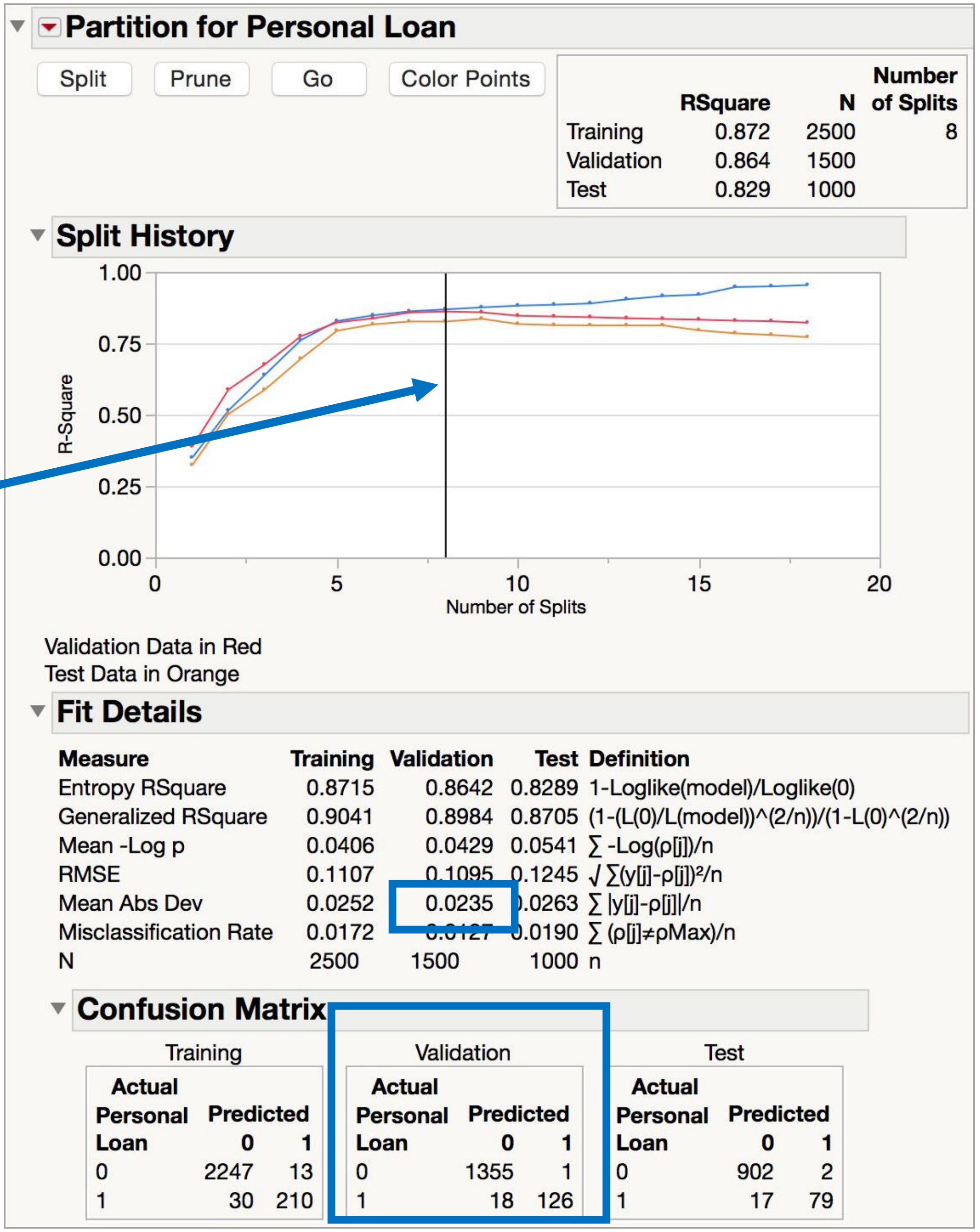
- **Minimum Split Size**: Controls the minimum number of records in terminal nodes
- **Validation**: The tree is grown, and pruned back to maximize the RSquare on the validation data

When validation is used, the “Go” option automates tree growth and pruning

The tree with the maximum Validation Rsquare has 8 splits

The tree is grown to 18 splits, and is pruned back to 8 splits

Validation error rate and confusion matrix for the final tree (cutoff for classification = 0.50)



Partition for Personal Loan

	RSquare	N	Number of Splits
Training	0.872	2500	8
Validation	0.864	1500	
Test	0.829	1000	

Split History

Validation Data in Red
Test Data in Orange

Fit Details

Measure	Training	Validation	Test	Definition
Entropy RSquare	0.8715	0.8642	0.8289	1-Loglike(model)/Loglike(0)
Generalized RSquare	0.9041	0.8984	0.8705	(1-(L(0)/L(model))^(2/n))/(1-L(0)^(2/n))
Mean -Log p	0.0406	0.0429	0.0541	∑ -Log(ρ[j])/n
RMSE	0.1107	0.1095	0.1245	√ ∑(y[j]-ρ[j])²/n
Mean Abs Dev	0.0252	0.0235	0.0263	∑ |y[j]-ρ[j]|/n
Misclassification Rate	0.0172	0.0127	0.0190	∑ (ρ[j]≠ρMax)/n
N	2500	1500	1000	n

Confusion Matrix

Training

Actual Personal Loan	Predicted 0	Predicted 1
0	2247	13
1	30	210

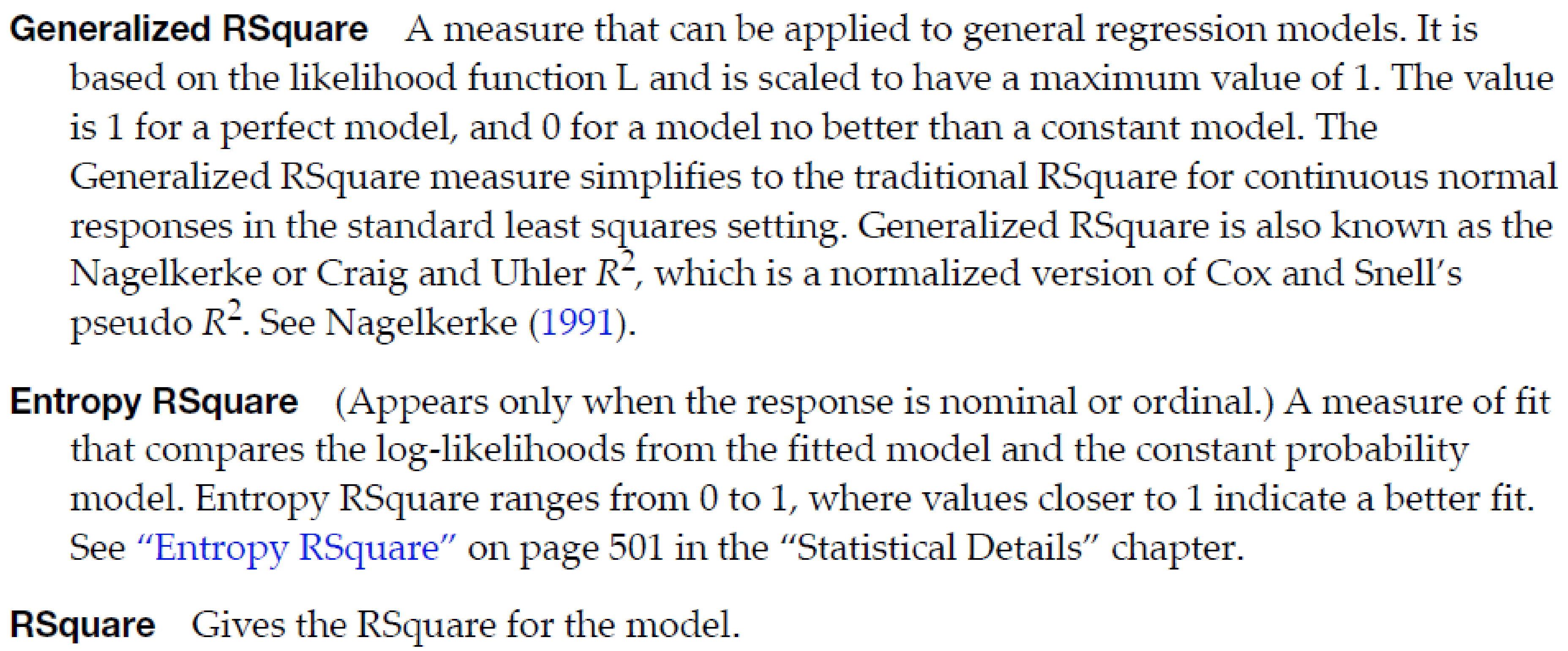
Validation

Actual Personal Loan	Predicted 0	Predicted 1
0	1355	1
1	18	126

Test

Actual Personal Loan	Predicted 0	Predicted 1
0	902	2
1	17	79

Validation

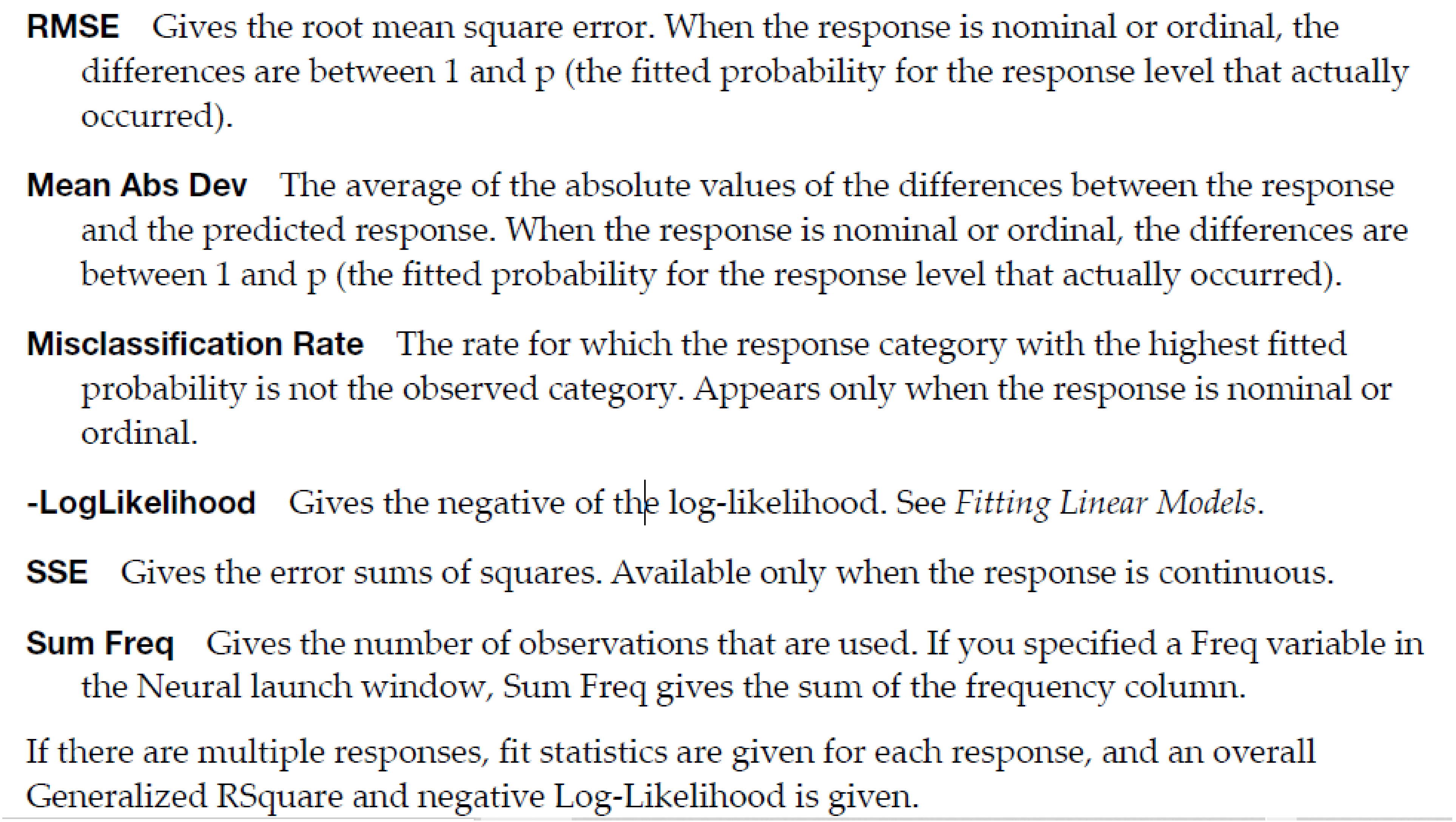


Generalized RSquare A measure that can be applied to general regression models. It is based on the likelihood function L and is scaled to have a maximum value of 1. The value is 1 for a perfect model, and 0 for a model no better than a constant model. The Generalized RSquare measure simplifies to the traditional RSquare for continuous normal responses in the standard least squares setting. Generalized RSquare is also known as the Nagelkerke or Craig and Uhler R^2, which is a normalized version of Cox and Snell's pseudo R^2. See Nagelkerke (1991).

Entropy RSquare (Appears only when the response is nominal or ordinal.) A measure of fit that compares the log-likelihoods from the fitted model and the constant probability model. Entropy RSquare ranges from 0 to 1, where values closer to 1 indicate a better fit. See "Entropy RSquare" on page 501 in the "Statistical Details" chapter.

RSquare Gives the RSquare for the model.

Validation



RMSE Gives the root mean square error. When the response is nominal or ordinal, the differences are between 1 and p (the fitted probability for the response level that actually occurred).

Mean Abs Dev The average of the absolute values of the differences between the response and the predicted response. When the response is nominal or ordinal, the differences are between 1 and p (the fitted probability for the response level that actually occurred).

Misclassification Rate The rate for which the response category with the highest fitted probability is not the observed category. Appears only when the response is nominal or ordinal.

-LogLikelihood Gives the negative of the log-likelihood. See *Fitting Linear Models*.

SSE Gives the error sums of squares. Available only when the response is continuous.

Sum Freq Gives the number of observations that are used. If you specified a Freq variable in the Neural launch window, Sum Freq gives the sum of the frequency column.

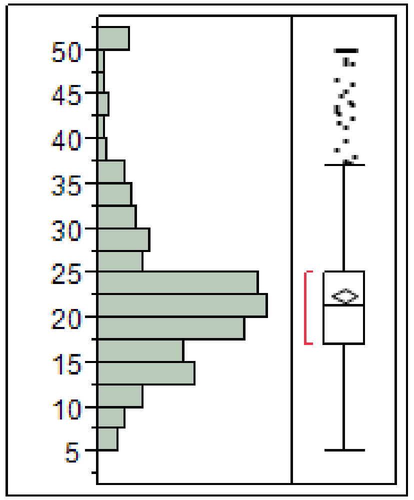
If there are multiple responses, fit statistics are given for each response, and an overall Generalized RSquare and negative Log-Likelihood is given.

Regression Trees for Prediction

- Used with continuous outcome variable
- Procedure like classification tree
- Many splits attempted, choose the one that maximizes the difference between subgroup means
- Difference measured as the sum of squared deviations
- Prediction is the **average** of the numerical target variable (rather than a probability)

Regression Trees

mvalue



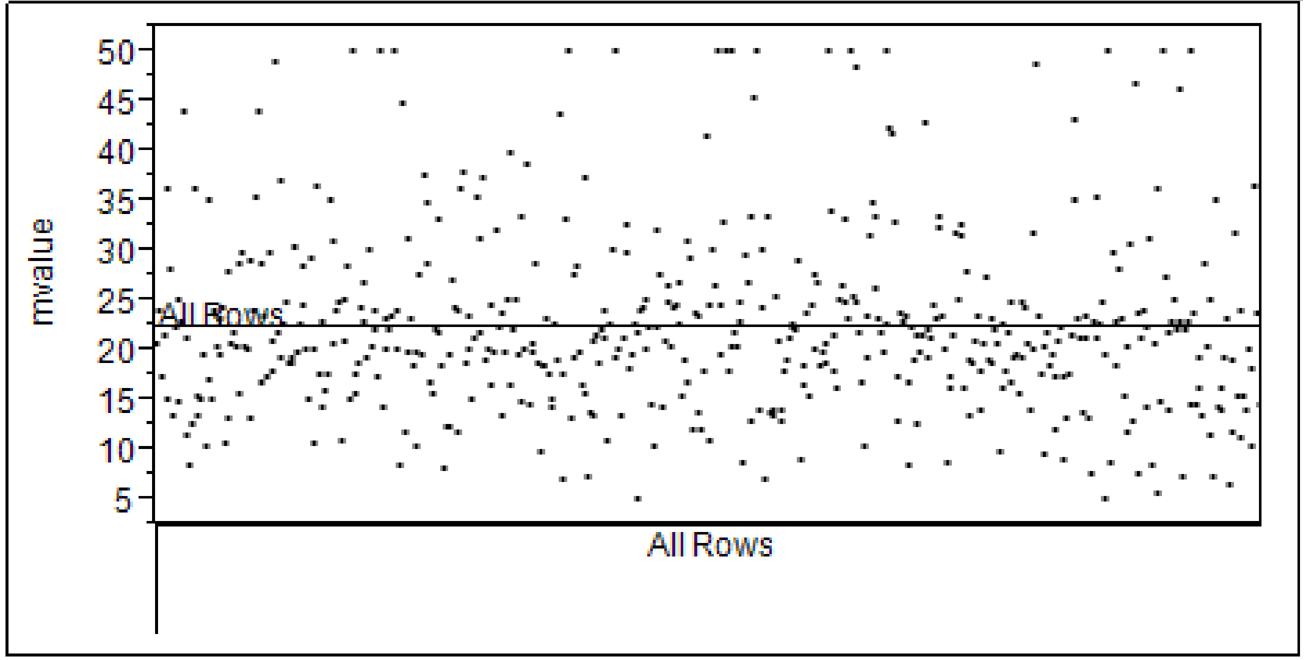
Quantiles

100.0%	maximum	50
99.5%		50
97.5%		50
90.0%		34.9
75.0%	quartile	25
50.0%	median	21.2
25.0%	quartile	16.95
10.0%		12.7
2.5%		8.235
0.5%		5.321
0.0%	minimum	5

Summary Statistics

Mean	22.532806
Std Dev	9.1971041
Std Err Mean	0.4088611
Upper 95% Mean	23.336085
Lower 95% Mean	21.729528
N	506

Partition for mvalue

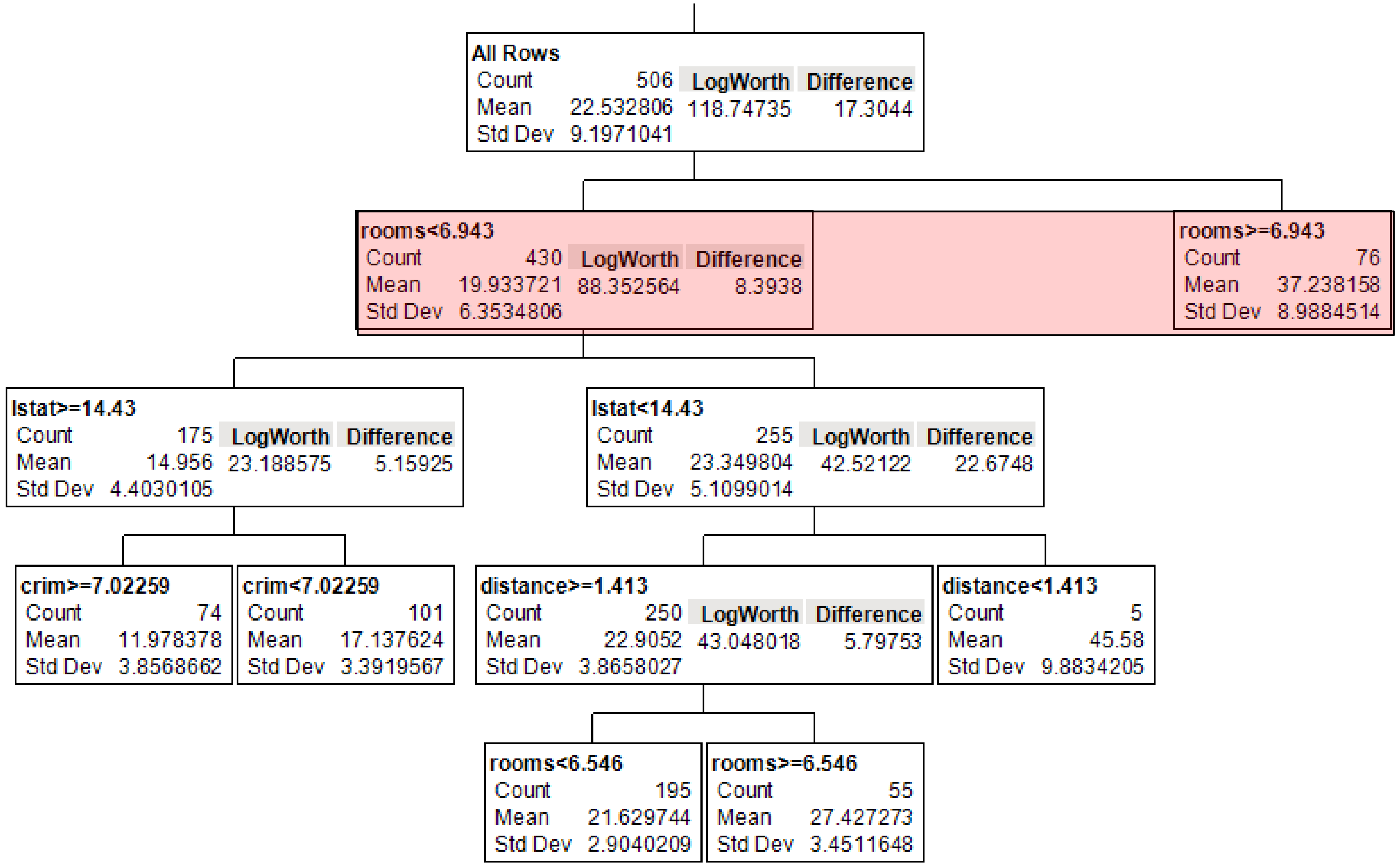


RSquare	RMSE	N	Number of Splits	AICc
0.000	.	506	0	0

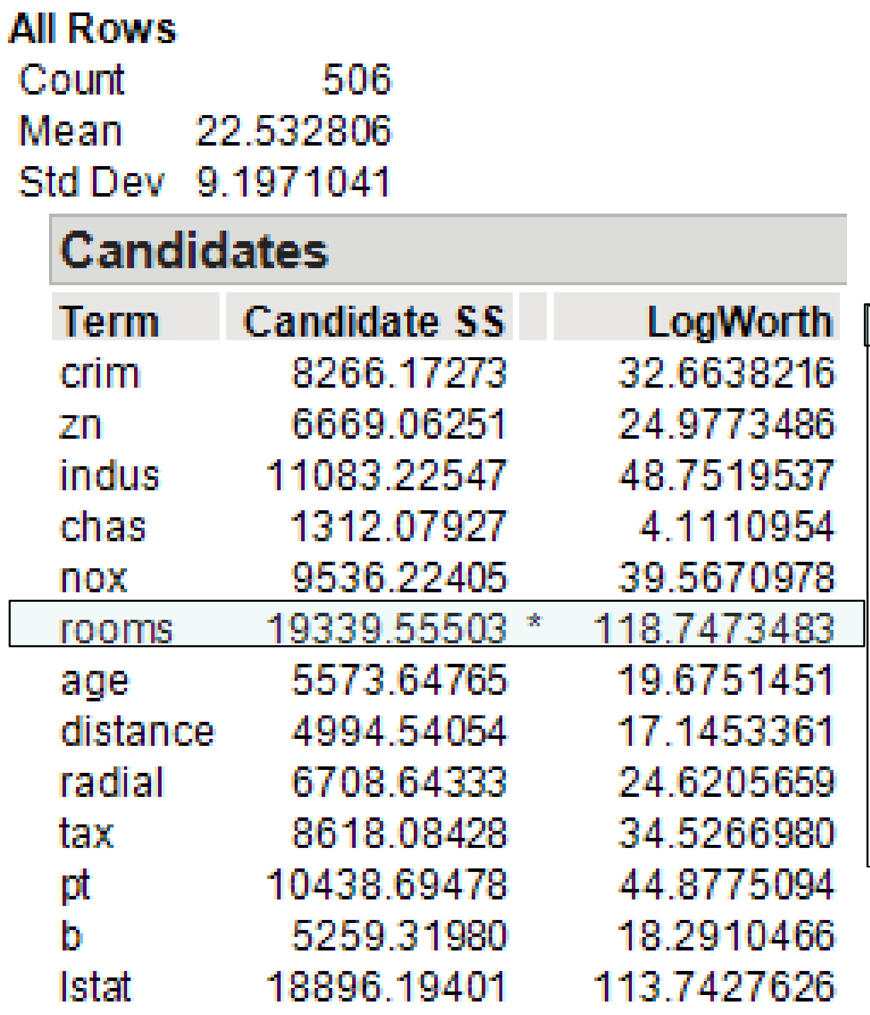
All Rows

Count	506
Mean	22.532806
Std Dev	9.1971041

Regression Trees



Boston Housing Data

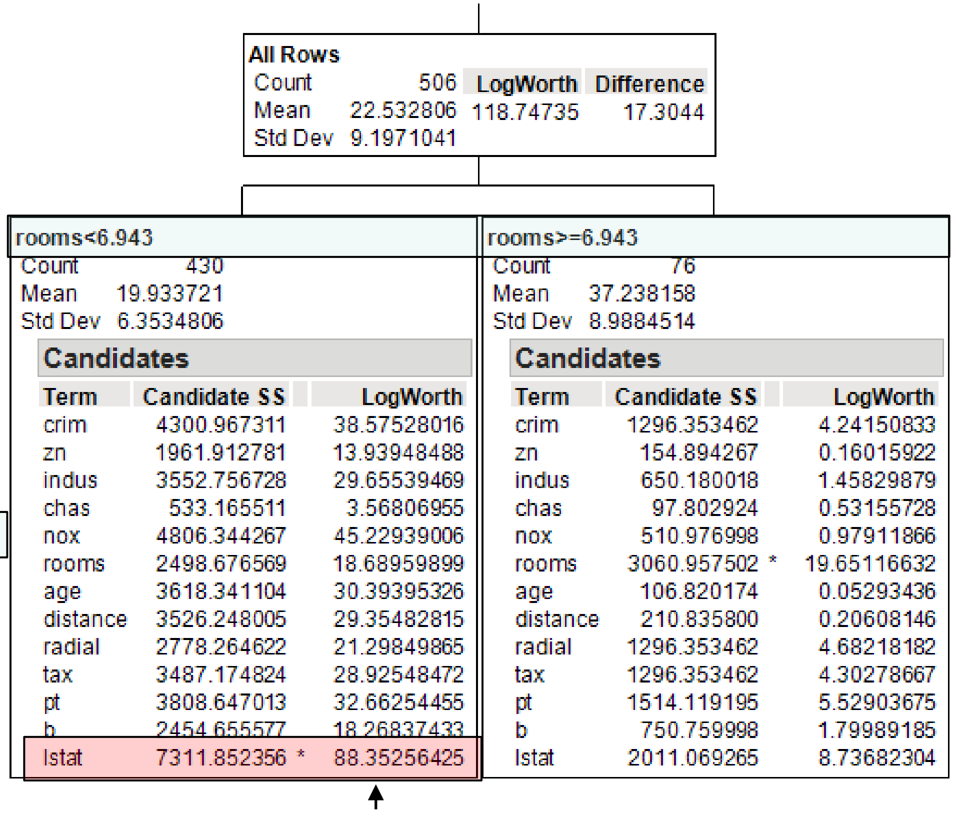


All Rows

Count	506
Mean	22.532806
Std Dev	9.1971041

Candidates

Term	Candidate SS		LogWorth
crim	8266.17273		32.6638216
zn	6669.06251		24.9773486
indus	11083.22547		48.7519537
chas	1312.07927		4.1110954
nox	9536.22405		39.5670978
rooms	19339.55503	*	118.7473483
age	5573.64765		19.6751451
distance	4994.54054		17.1453361
radial	6708.64333		24.6205659
tax	8618.08428		34.5266980
pt	10438.69478		44.8775094
b	5259.31980		18.2910466
lstat	18896.19401		113.7427626



All Rows

		LogWorth	Difference
Count	506		
Mean	22.532806	118.74735	17.3044
Std Dev	9.1971041		

rooms<6.943

Count	430
Mean	19.933721
Std Dev	6.3534806

Candidates

Term	Candidate SS		LogWorth
crim	4300.967311		38.57528016
zn	1961.912781		13.93948488
indus	3552.756728		29.65539469
chas	533.165511		3.56806955
nox	4806.344267		45.22939006
rooms	2498.676569		18.68959899
age	3618.341104		30.39395326
distance	3526.248005		29.35482815
radial	2778.264622		21.29849865
tax	3487.174824		28.92548472
pt	3808.647013		32.66254455
b	2454.655577		18.26837433
lstat	7311.852356	*	88.35256425

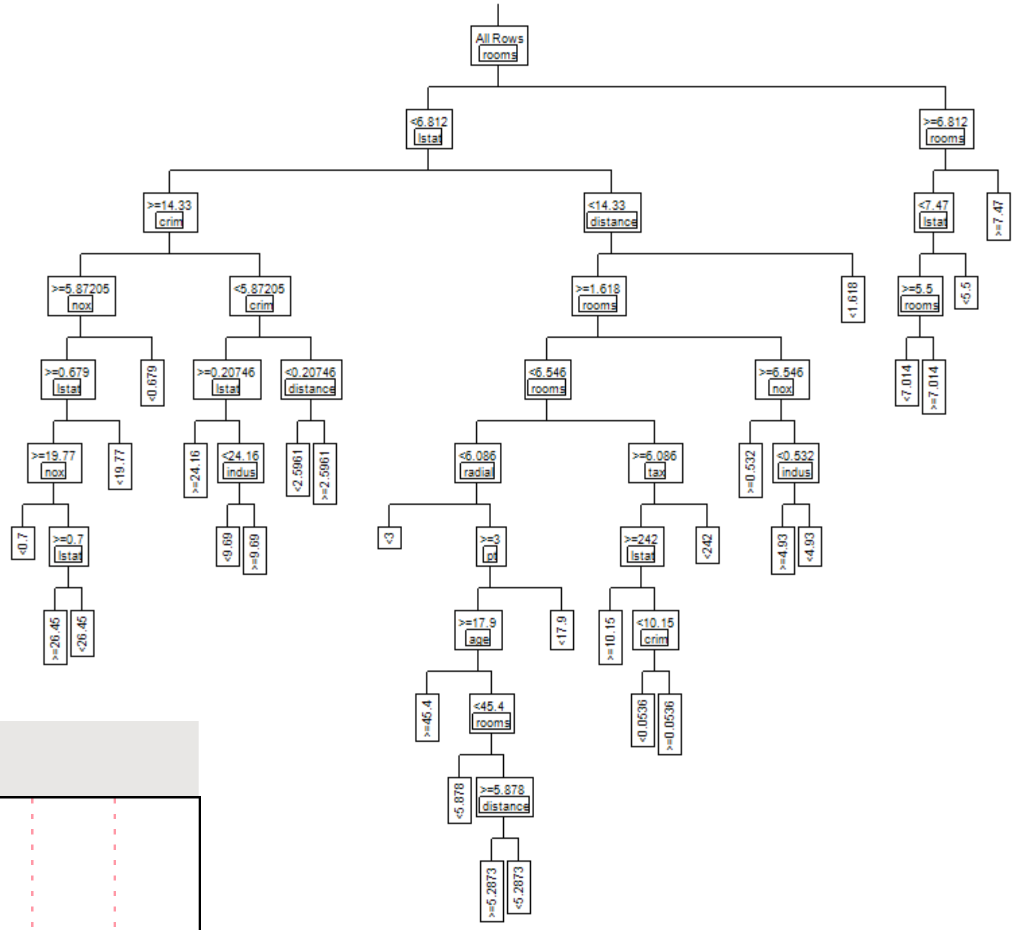
rooms>=6.943

Count	76
Mean	37.238158
Std Dev	8.9884514

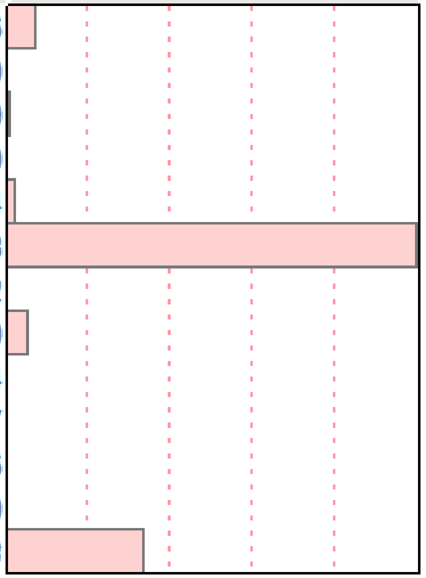
Candidates

Term	Candidate SS		LogWorth
crim	1296.353462		4.24150833
zn	154.894267		0.16015922
indus	650.180018		1.45829879
chas	97.802924		0.53155728
nox	510.976998		0.97911866
rooms	3060.957502	*	19.65116632
age	106.820174		0.05293436
distance	210.835800		0.20608146
radial	1296.353462		4.68218182
tax	1296.353462		4.30278667
pt	1514.119195		5.52903675
b	750.759998		1.79989185
lstat	2011.069265		8.73682304

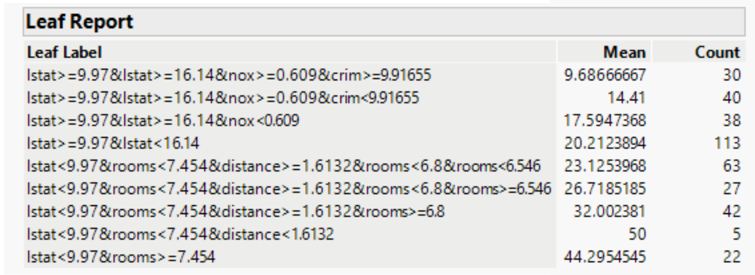
$-\log_{10}(p\text{-value})$



Term	Number of Splits	SS
crim	3	924.046
zn	0	0.000
indus	2	86.329
chas	0	0.000
nox	3	265.824
rooms	6	12285.998
age	1	50.482
distance	3	642.560
radial	1	39.724
tax	1	68.077
pt	1	40.686
b	0	0.000
lstat	6	4131.903

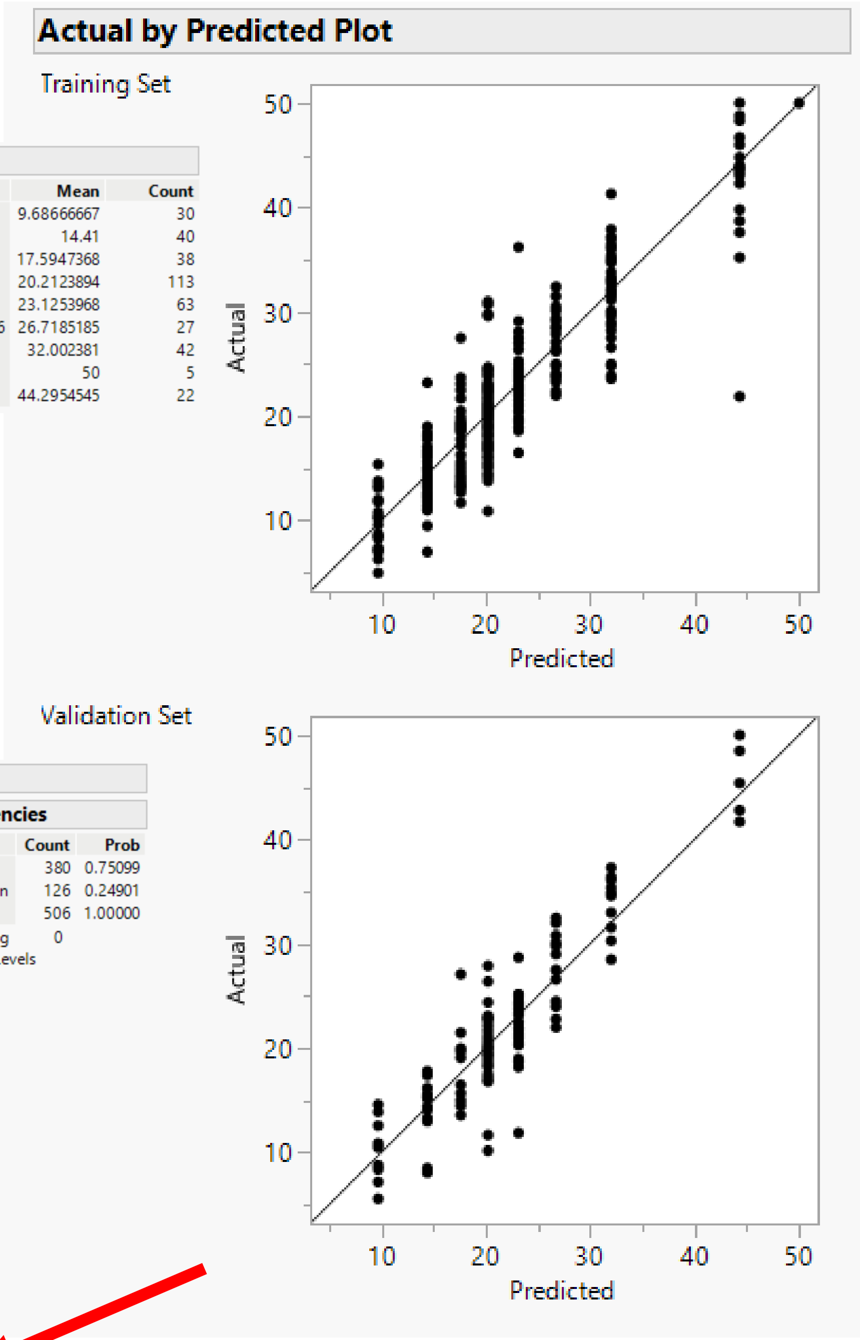


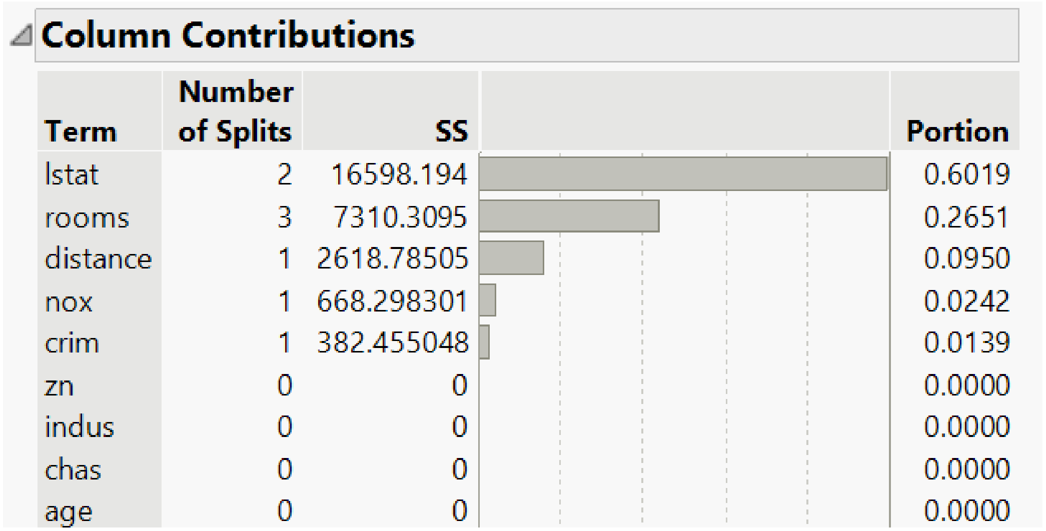
1. CRIM - per capita crime rate by town
2. ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS - proportion of non-retail business acres per town.
4. CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
5. NOX - nitric oxides concentration (parts per 10 million)
6. RM - average number of rooms per dwelling
7. AGE - proportion of owner-occupied units built prior to 1940
8. DIS - weighted distances to five Boston employment centres
9. RAD - index of accessibility to radial highways
10. TAX - full-value property-tax rate per $10,000
11. PTRATIO - pupil-teacher ratio by town
12. B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
13. LSTAT - % lower status of the population
14. MEDV - Median value of owner-occupied homes in $1000's



Leaf Report

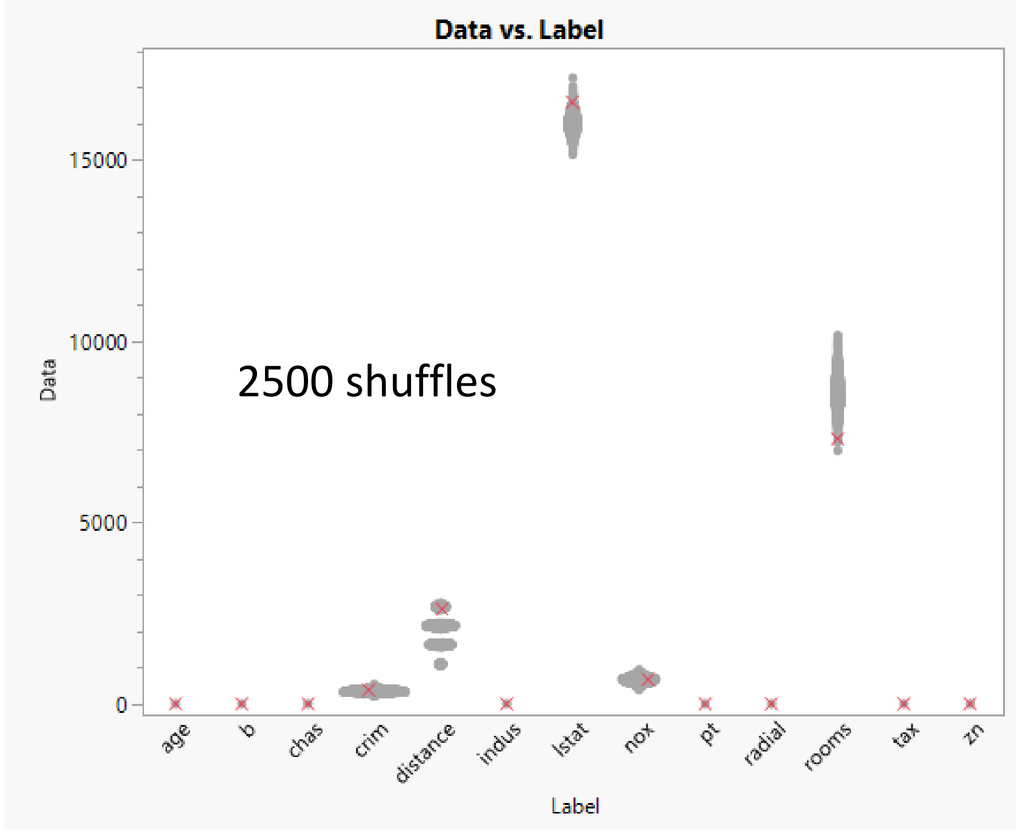
Leaf Label	Mean	Count
lstat>=9.97&lstat>=16.14&nox>=0.609&crim>=9.91655	9.68666667	30
lstat>=9.97&lstat>=16.14&nox>=0.609&crim<9.91655	14.41	40
lstat>=9.97&lstat>=16.14&nox<0.609	17.5947368	38
lstat>=9.97&lstat<16.14	20.2123894	113
lstat<9.97&rooms<7.454&distance>=1.6132&rooms<6.8&rooms<6.546	23.1253968	63
lstat<9.97&rooms<7.454&distance>=1.6132&rooms<6.8&rooms>=6.546	26.7185185	27
lstat<9.97&rooms<7.454&distance>=1.6132&rooms>=6.8	32.002381	42
lstat<9.97&rooms<7.454&distance<1.6132	50	5
lstat<9.97&rooms>=7.454	44.2954545	22

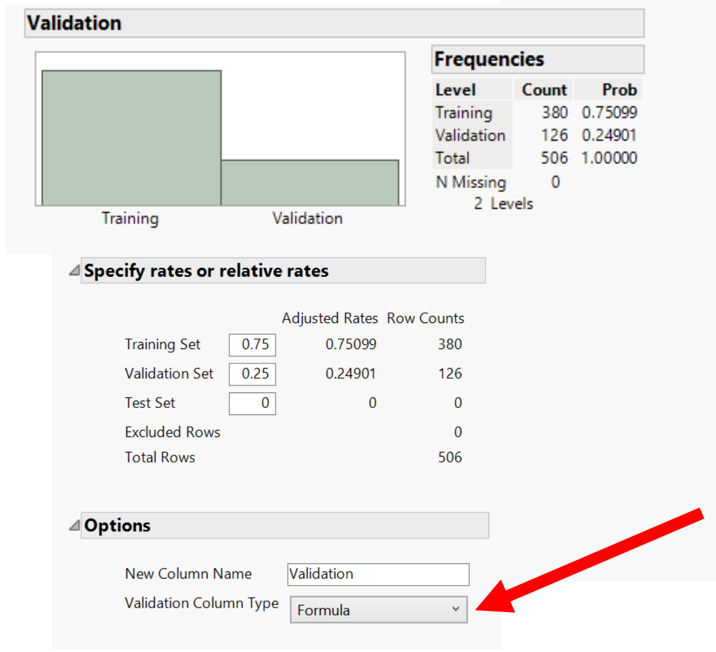




Column Contributions

Term	Number of Splits	SS	Portion
lstat	2	16598.194	0.6019
rooms	3	7310.3095	0.2651
distance	1	2618.78505	0.0950
nox	1	668.298301	0.0242
crim	1	382.455048	0.0139
zn	0	0	0.0000
indus	0	0	0.0000
chas	0	0	0.0000
age	0	0	0.0000





Validation

Frequencies

Level	Count	Prob
Training	380	0.75099
Validation	126	0.24901
Total	506	1.00000
N Missing	0	
2 Levels		

Specify rates or relative rates

		Adjusted Rates	Row Counts
Training Set	0.75	0.75099	380
Validation Set	0.25	0.24901	126
Test Set	0	0	0
Excluded Rows			0
Total Rows			506

Options

New Column Name: Validation

Validation Column Type: Formula

Advantages of Trees

- Easy to use, understand
- Produce rules that are easy to interpret & implement
- Variable selection & reduction is automatic
- Do not require the assumptions of statistical models
- Can work without extensive handling of missing data (this is an option in the Partition dialog in JMP)

Disadvantages of Trees

- May not perform well where there is structure in the data that is not well captured by horizontal or vertical splits
- Since the process deals with one variable at a time, no way to capture interactions between variables

Shmueli, G., Bruce, P., Stephens, M. and Patel, N. (2016) Data Mining for Business Analytics: Concepts, Techniques, and Applications with JMP Pro, Wiley, USA https://www.wiley.com/en-us/Data+Mining+for+Business+Analytics%3A+Concepts%2C+Techniques%2C+and+Applications+with+JMP+Pro-p-9781118877524

Improving Trees

- Single trees may not have good predictive ability.
- Results from multiple trees can be combined to improve performance
- The resulting model is an “ensemble” model
- Two multi-tree approaches in JMP Pro:
  - **Bootstrap Forests** (a variant of Random Forests)
  - **Boosted Trees**

Shmueli, G., Bruce, P., Stephens, M. and Patel, N. (2016) Data Mining for Business Analytics: Concepts, Techniques, and Applications with JMP Pro, Wiley, USA https://www.wiley.com/en-us/Data+Mining+for+Business+Analytics%3A+Concepts%2C+Techniques%2C+and+Applications+with+JMP+Pro-p-9781118877524

Ensemble Tree Methods

- Bootstrap Forests (Random Forrest)

Grow many trees to bootstrapped versions of the training data and average them

- Boosted Trees (Boosting)

Repeatedly grow shallow trees to the residuals and build up an additive model consisting of a sum of trees

Shmueli, G., Bruce, P., Stephens, M. and Patel, N. (2016) Data Mining for Business Analytics: Concepts, Techniques, and Applications with JMP Pro, Wiley, USA https://www.wiley.com/en-us/Data+Mining+for+Business+Analytics%3A+Concepts%2C+Techniques%2C+and+Applications+with+JMP+Pro-p-9781118877524

Ensemble Tree Methods

Bootstrap Forests

1. A random sample is drawn with replacement from the data set (bootstrapping)
2. Predictors are randomly drawn from the candidate list of predictors
3. A small tree is fit (a “weak learner”)
4. The process is repeated
5. The final model is the average of all of the trees, producing a “Bootstrap aggregated” (or “bagged) model

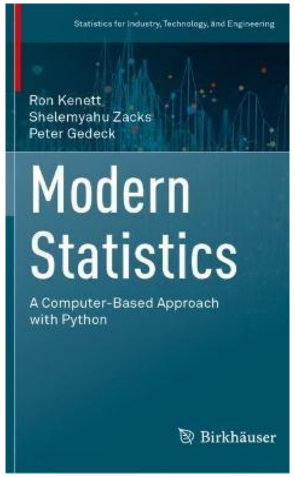
Shmueli, G., Bruce, P., Stephens, M. and Patel, N. (2016) Data Mining for Business Analytics: Concepts, Techniques, and Applications with JMP Pro, Wiley, USA https://www.wiley.com/en-us/Data+Mining+for+Business+Analytics%3A+Concepts%2C+Techniques%2C+and+Applications+with+JMP+Pro-p-9781118877524

Ensemble Tree Methods

Boosted Trees

1. A simple (small) tree is fit to the data with a random sample of the predictors
2. The scaled residuals from this tree are calculated
3. A new simple tree is fit to these scaled residuals with another random sample of predictors
4. This process continues
5. The final boosted model is the sum of the models for the individual trees

Shmueli, G., Bruce, P., Stephens, M. and Patel, N. (2016) Data Mining for Business Analytics: Concepts, Techniques, and Applications with JMP Pro, Wiley, USA https://www.wiley.com/en-us/Data+Mining+for+Business+Analytics%3A+Concepts%2C+Techniques%2C+and+Applications+with+JMP+Pro-p-9781118877524


Statistics for Industry, Technology, and Engineering

Ron Kenett
Shelemyahu Zacks
Peter Gedeck

Modern Statistics

A Computer-Based Approach with Python

Birkhäuser

Chapter 7 Modern Analytic Methods: Part I

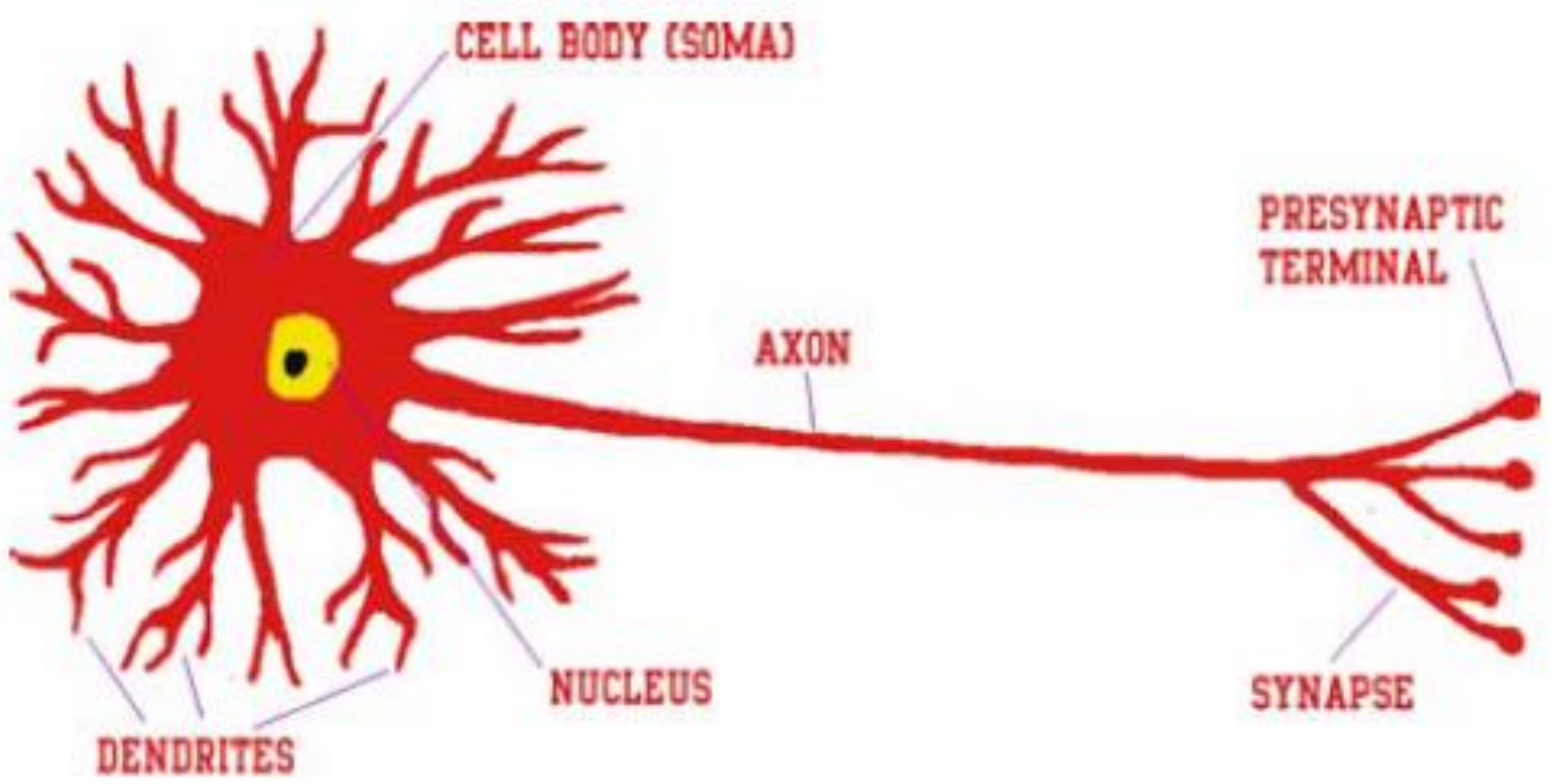
7.8 Neural Networks

A neural network is composed of a set of computational units, called neurons, connected together through weighted connections. Neurons are organized in layers so that every neuron in a layer is exclusively connected to the neurons of the preceding layer and the subsequent layer. Every neuron, also called a node, represents an autonomous computational unit and receives inputs as a series of signals that dictate its activation. Following activation, every neuron produces an output signal. All the input signals reach the neuron simultaneously, so the neuron receives more than one input signal, but it produces only one output signal. Every input signal is associated with a connection weight. The weight determines the relative importance the input signal can have in producing the final impulse transmitted by

Neural networks: Basic Idea

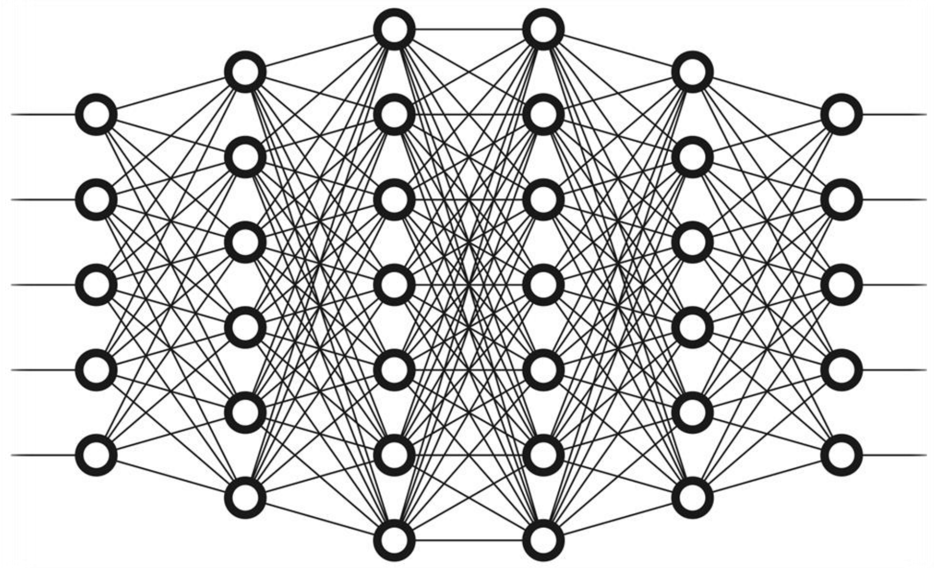
- Combine input information in a complex and flexible neural net "model"
- Model "coefficients" are continually tweaked in an iterative process
- The network's interim performance in classification and prediction informs successive tweaks

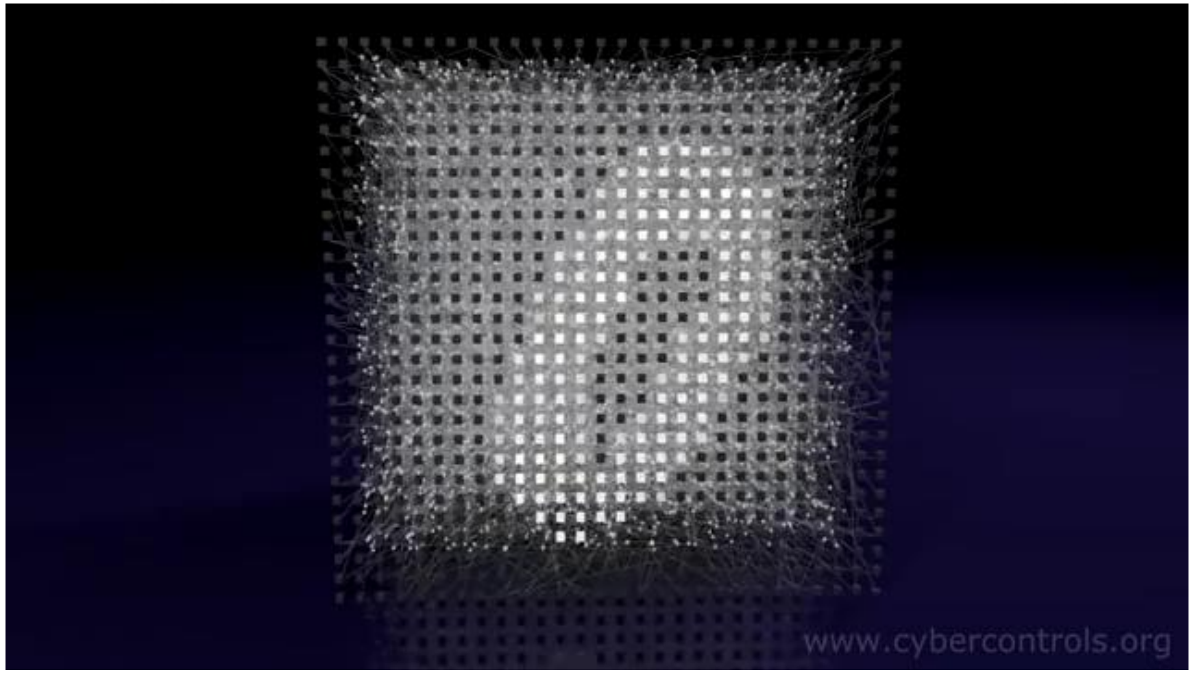
Neural Networks

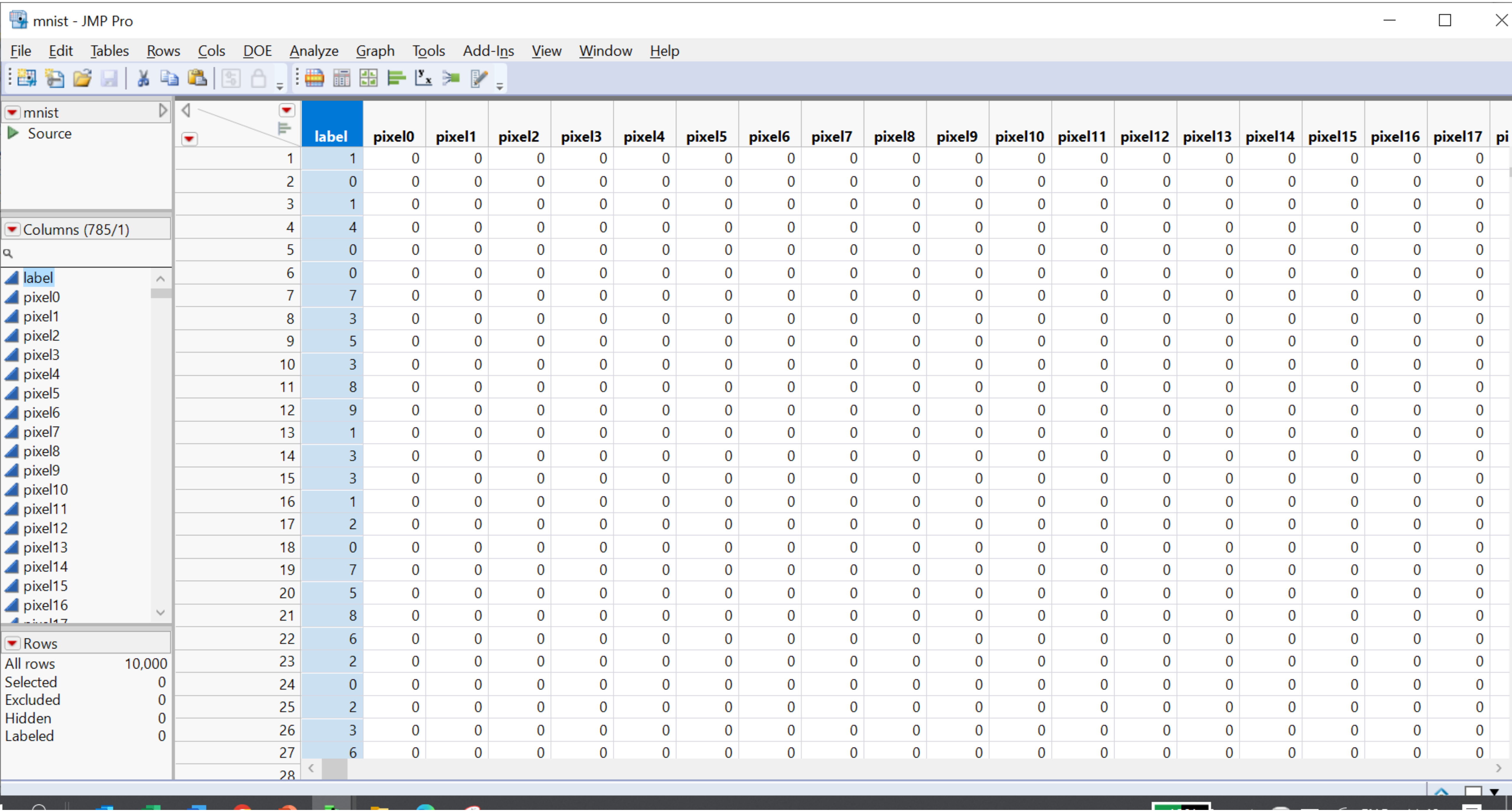


MNIST

Modified **N**ational **I**nstitute of **S**tandards and **T**echnology







mnist - JMP Pro

File Edit Tables Rows Cols DOE Analyze Graph Tools Add-Ins View Window Help

mnist

Source

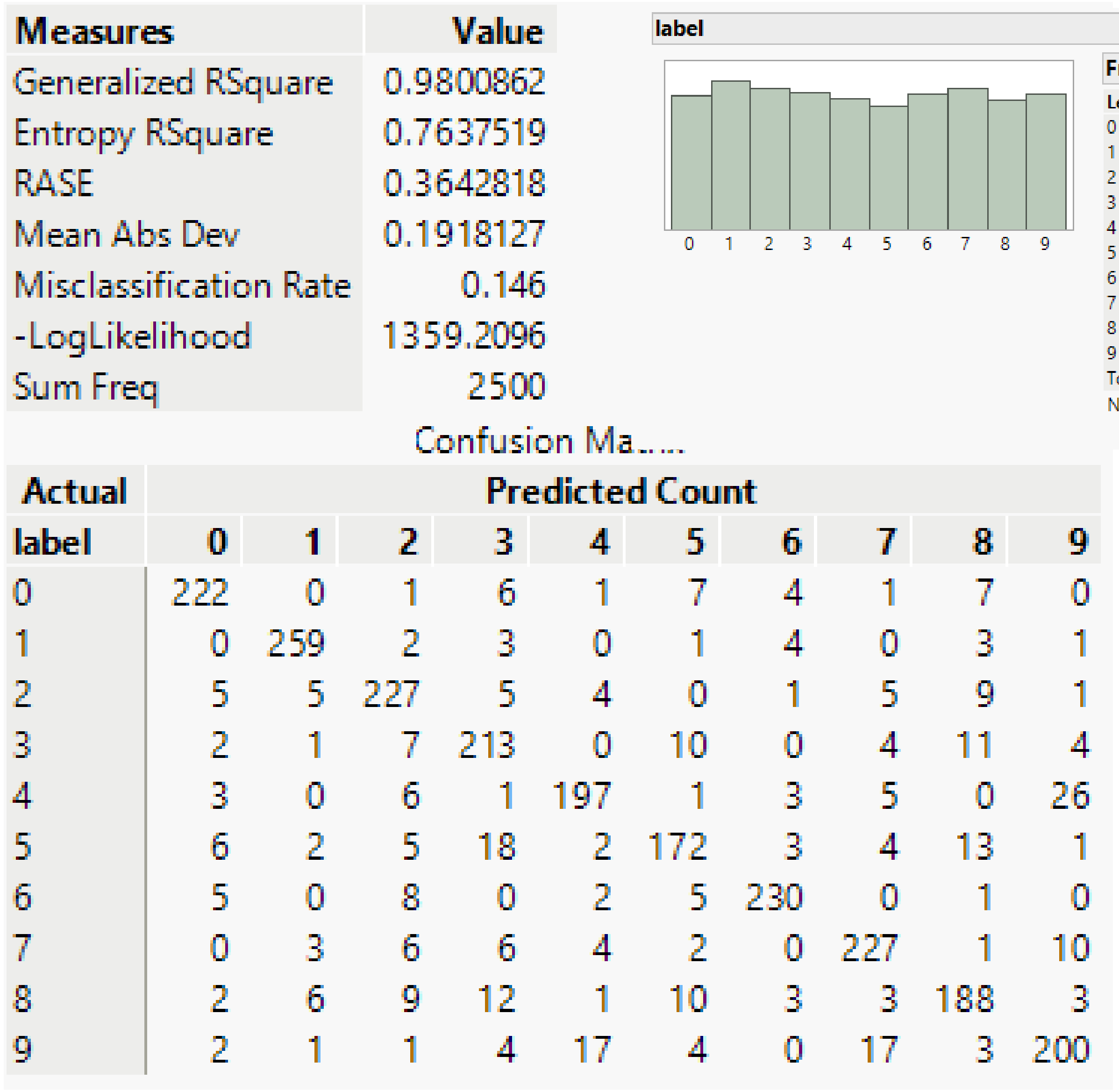
Columns (785/1)

label
pixel0
pixel1
pixel2
pixel3
pixel4
pixel5
pixel6
pixel7
pixel8
pixel9
pixel10
pixel11
pixel12
pixel13
pixel14
pixel15
pixel16

Rows

All rows	10,000
Selected	0
Excluded	0
Hidden	0
Labeled	0

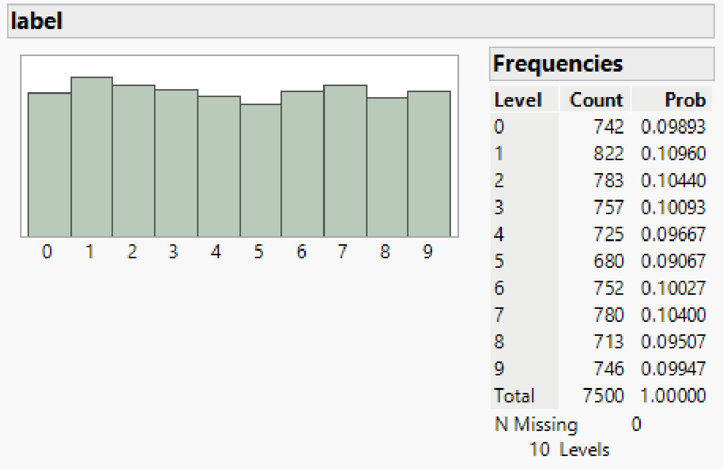
	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	pixel11	pixel12	pixel13	pixel14	pixel15	pixel16	pixel17	pi
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
8	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
9	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
10	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
11	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
12	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
13	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
14	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
15	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
16	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
17	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
19	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
20	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
21	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
22	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
23	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
25	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
26	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
27	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
28																				



Measures	Value
Generalized RSquare	0.9800862
Entropy RSquare	0.7637519
RASE	0.3642818
Mean Abs Dev	0.1918127
Misclassification Rate	0.146
-LogLikelihood	1359.2096
Sum Freq	2500

Confusion Ma.....

Actual	Predicted Count									
label	0	1	2	3	4	5	6	7	8	9
0	222	0	1	6	1	7	4	1	7	0
1	0	259	2	3	0	1	4	0	3	1
2	5	5	227	5	4	0	1	5	9	1
3	2	1	7	213	0	10	0	4	11	4
4	3	0	6	1	197	1	3	5	0	26
5	6	2	5	18	2	172	3	4	13	1
6	5	0	8	0	2	5	230	0	1	0
7	0	3	6	6	4	2	0	227	1	10
8	2	6	9	12	1	10	3	3	188	3
9	2	1	1	4	17	4	0	17	3	200

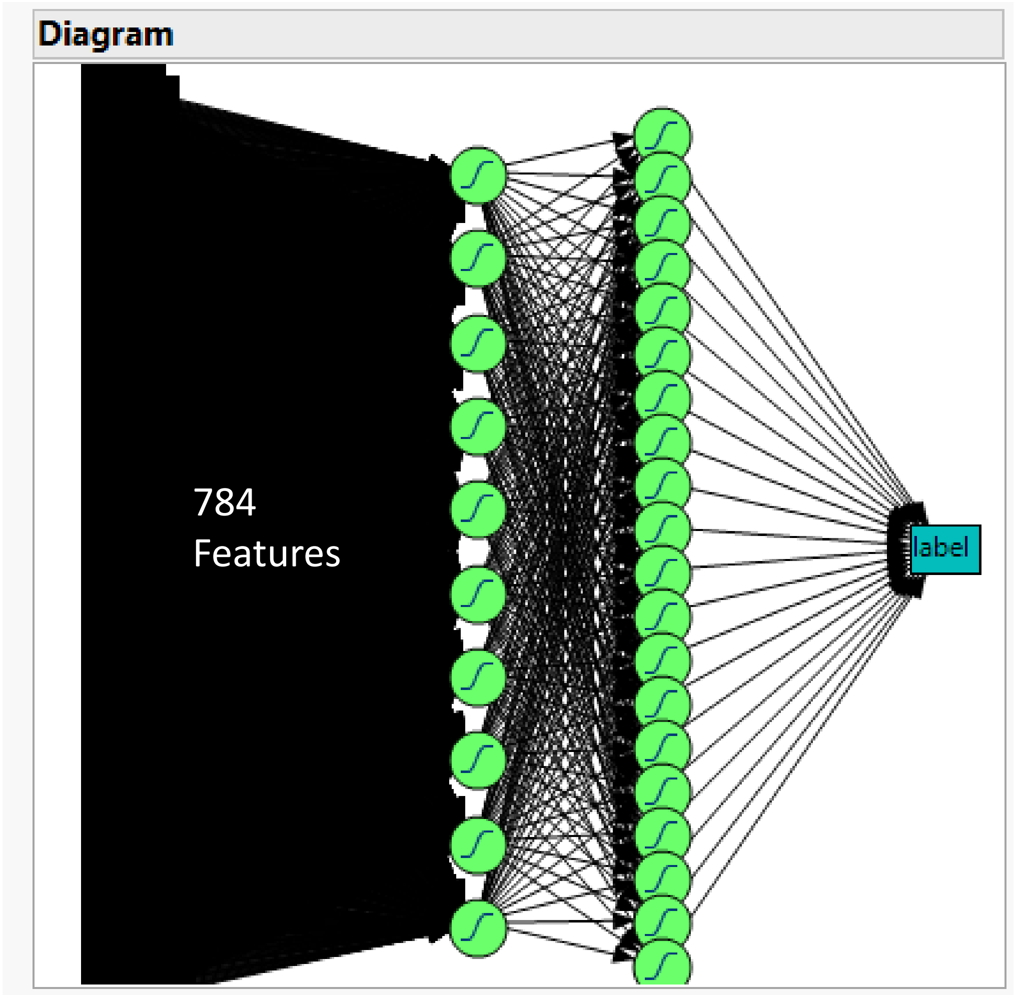


label

Frequencies

Level	Count	Prob
0	742	0.09893
1	822	0.10960
2	783	0.10440
3	757	0.10093
4	725	0.09667
5	680	0.09067
6	752	0.10027
7	780	0.10400
8	713	0.09507
9	746	0.09947
Total	7500	1.00000

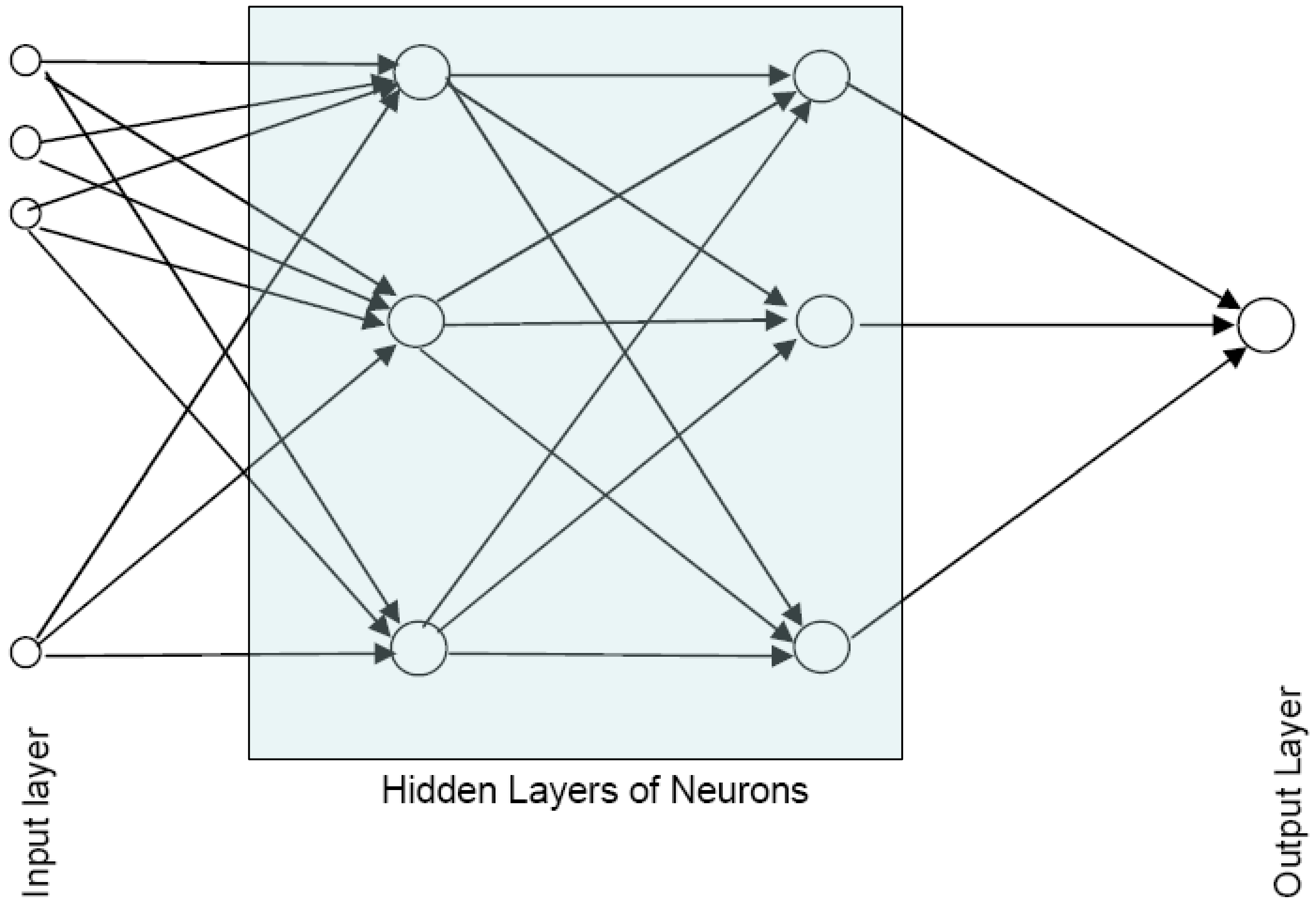
N Missing 0
10 Levels



Network Structure

- Multiple layers
  - Input layer (raw observations)
  - Hidden layers
  - Output layer
- Nodes
- Weights (like coefficients, subject to iterative adjustment)
- Bias values (also like coefficients, but not subject to iterative adjustment)

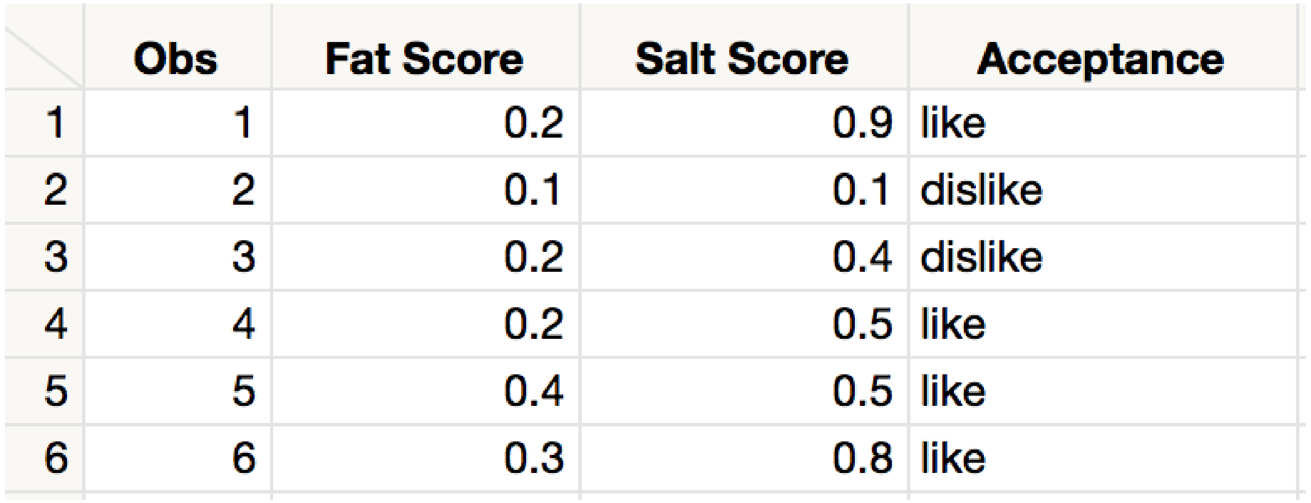
Schematic Diagram



Tiny Example

Tiny Dataset.jmp

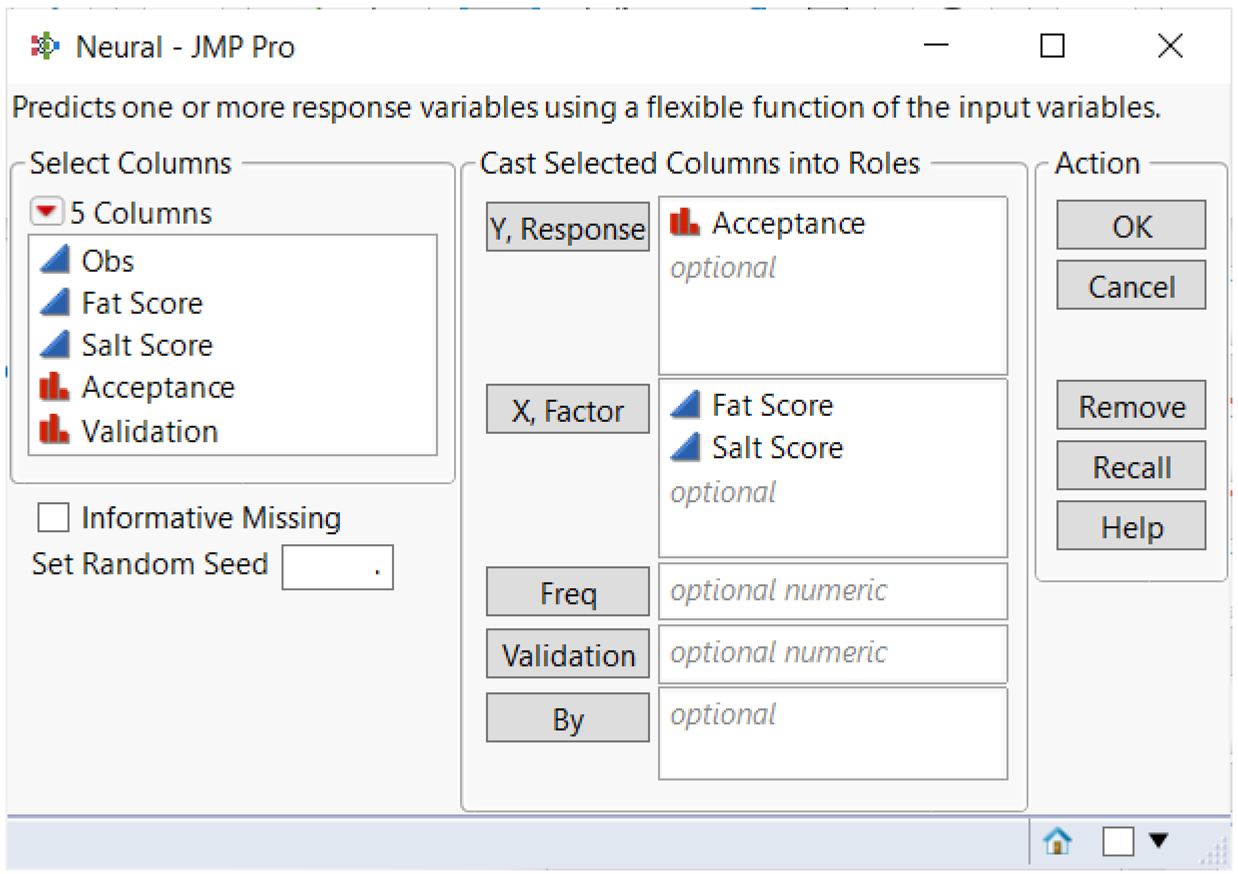
- Using fat and salt content to predict consumer acceptance of cheese



	Obs	Fat Score	Salt Score	Acceptance
1	1	0.2	0.9	like
2	2	0.1	0.1	dislike
3	3	0.2	0.4	dislike
4	4	0.2	0.5	like
5	5	0.4	0.5	like
6	6	0.3	0.8	like

Shmueli, G., Bruce, P., Stephens, M. and Patel, N. (2016) Data Mining for Business Analytics: Concepts, Techniques, and Applications with JMP Pro, Wiley, USA https://www.wiley.com/en-us/Data+Mining+for+Business+Analytics%3A+Concepts%2C+Techniques%2C+and+Applications+with+JMP+Pro-p-9781118877524

Tiny Example Neural Network



Neural - JMP Pro

Predicts one or more response variables using a flexible function of the input variables.

Select Columns

5 Columns

- Obs
- Fat Score
- Salt Score
- Acceptance
- Validation

☐ Informative Missing

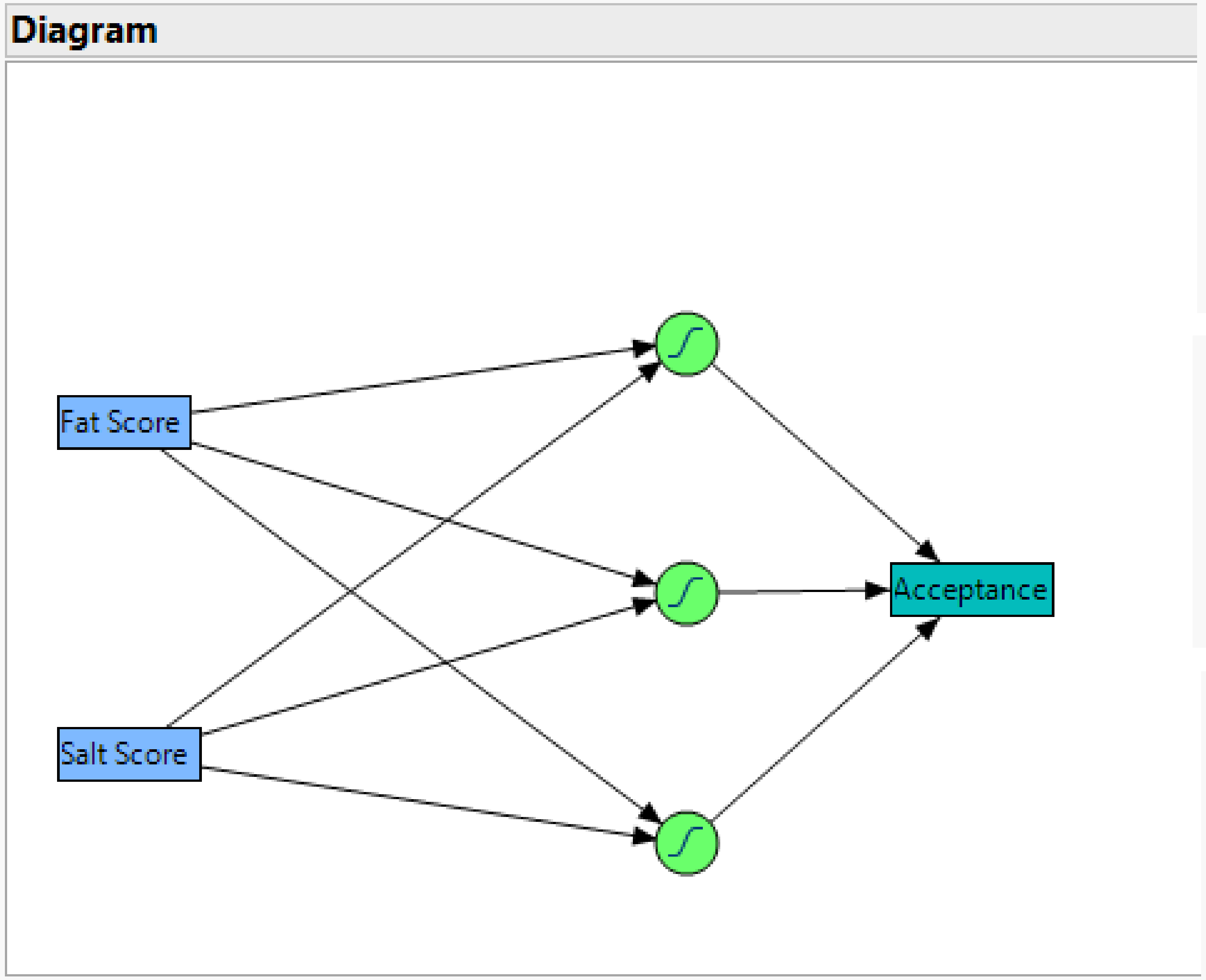
Set Random Seed .

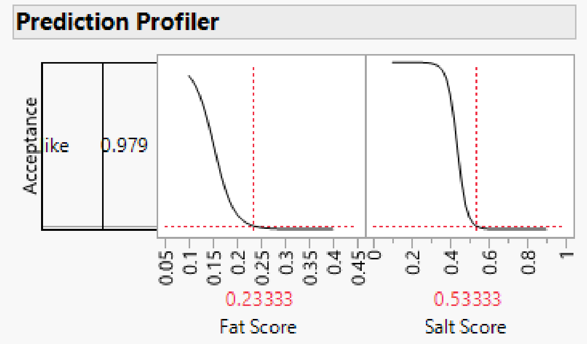
Cast Selected Columns into Roles

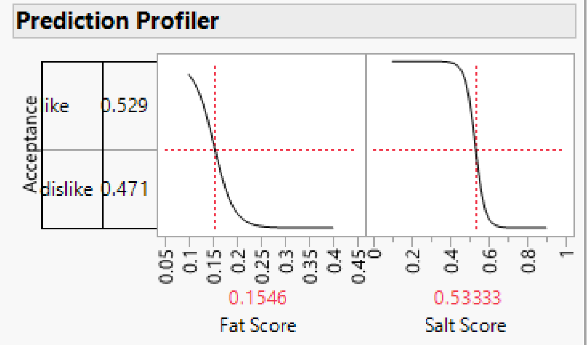
Y, Response	Acceptance *optional*
X, Factor	Fat Score Salt Score *optional*
Freq	*optional numeric*
Validation	*optional numeric*
By	*optional*

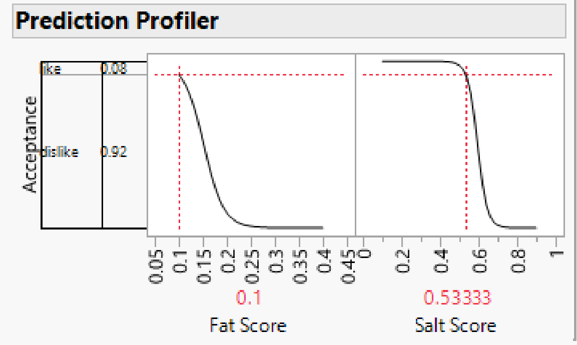
Action

Tiny Example Neural Network

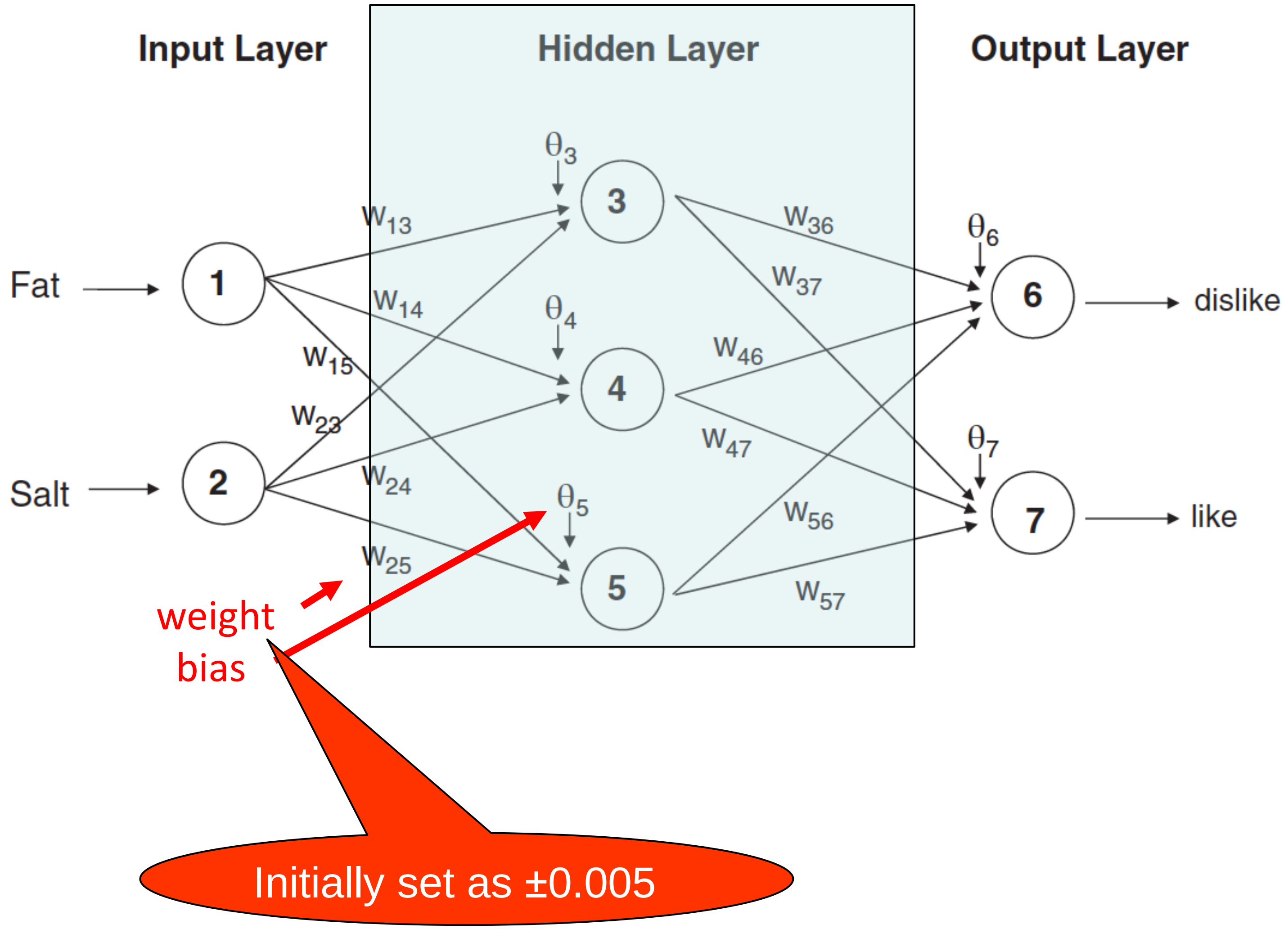








Tiny Example Neural Network



The Input Layer

For input layer, input = output

E.g., for record #1:

Fat input = output = 0.2

Salt input = output = 0.9

Output of input layer = input into hidden layer

The Hidden Layer

In this example, hidden layer has 3 nodes

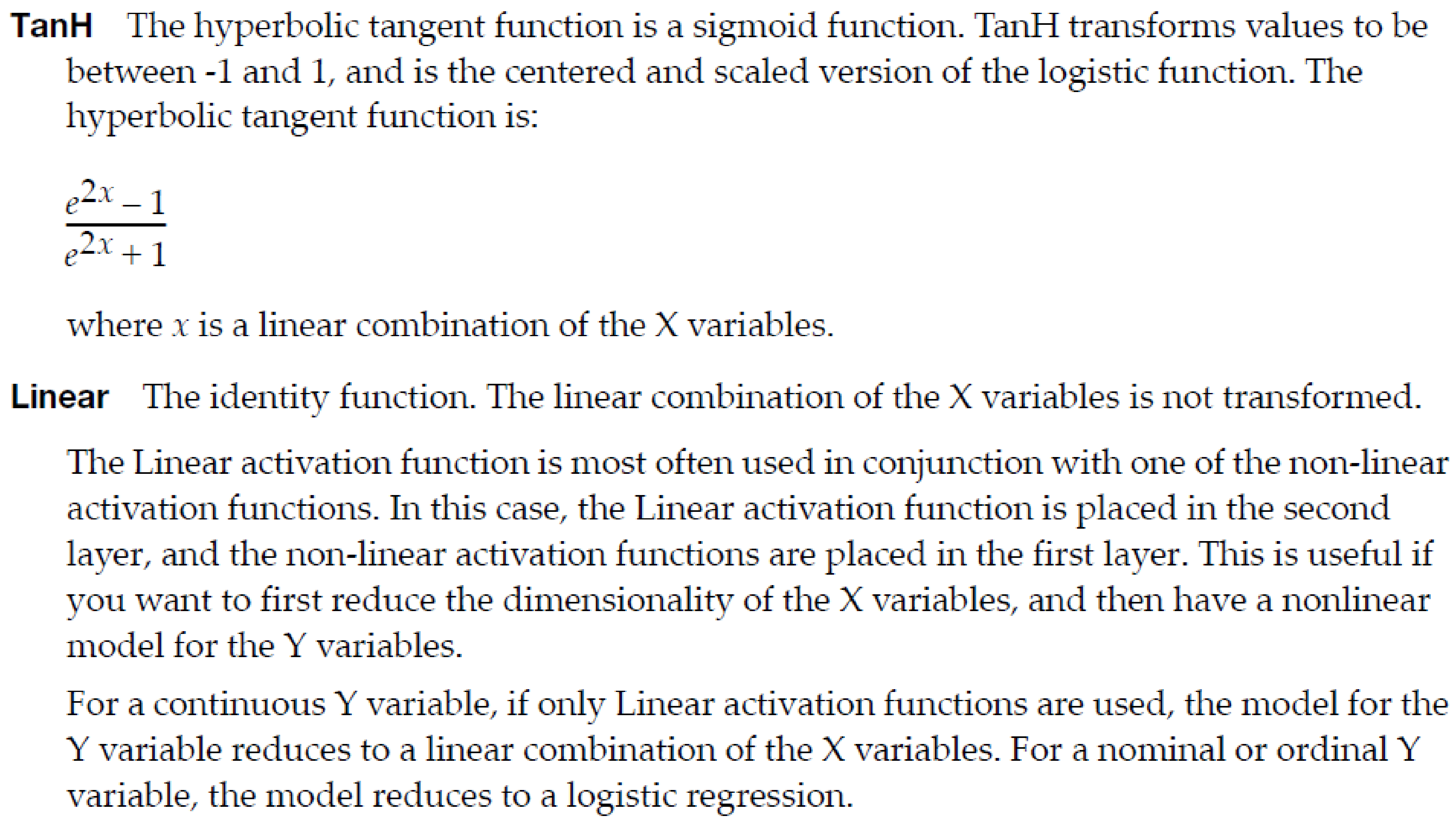
Each node receives as input the output of all input nodes

Output of each hidden node is a function of the weighted sum of inputs

$$output_j = g(\Theta_\mathrm{j} + \sum_{\mathrm{i=1}}^{\mathrm{p}} \mathrm{w_{ij}}\, \mathrm{x_i})$$

(The hidden layer function is also called an ”activation function”.)

The Hidden Layer



TanH The hyperbolic tangent function is a sigmoid function. TanH transforms values to be between -1 and 1, and is the centered and scaled version of the logistic function. The hyperbolic tangent function is:

$$\frac{e^{2x}-1}{e^{2x}+1}$$

where x is a linear combination of the X variables.

Linear The identity function. The linear combination of the X variables is not transformed.

The Linear activation function is most often used in conjunction with one of the non-linear activation functions. In this case, the Linear activation function is placed in the second layer, and the non-linear activation functions are placed in the first layer. This is useful if you want to first reduce the dimensionality of the X variables, and then have a nonlinear model for the Y variables.

For a continuous Y variable, if only Linear activation functions are used, the model for the Y variable reduces to a linear combination of the X variables. For a nominal or ordinal Y variable, the model reduces to a logistic regression.

The Hidden Layer

Gaussian The Gaussian function. Use this option for radial basis function behavior, or when the response surface is Gaussian (normal) in shape. The Gaussian function is:

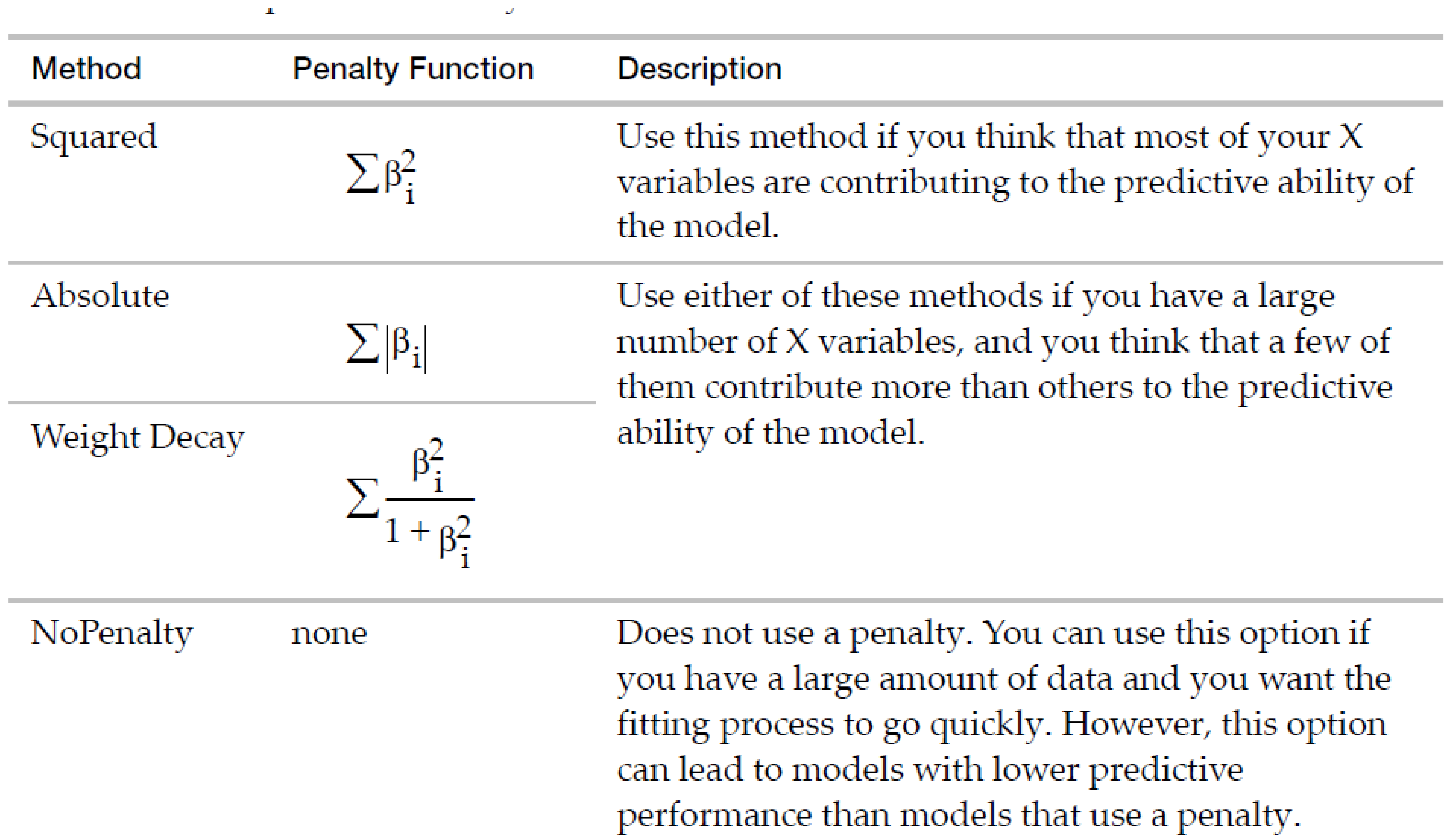
$$e^{-x^2}$$

where x is a linear combination of the X variables.

Use the Boosting panel in the Model Launch control panel to specify the number of component models and the learning rate. Use the Hidden Layer Structure panel in the Model Launch control panel to specify the structure of the base model.

The learning rate must be $0 < r \leq 1$. Learning rates close to 1 result in faster convergence on a final model, but also have a higher tendency to overfit data. Use learning rates close to 1 when a small Number of Models is specified.

Options



Method	Penalty Function	Description
Squared	$\sum \beta_i^2$	Use this method if you think that most of your X variables are contributing to the predictive ability of the model.
Absolute	$\sum \lvert\beta_i\rvert$	Use either of these methods if you have a large number of X variables, and you think that a few of them contribute more than others to the predictive ability of the model.
Weight Decay	$\sum \frac{\beta_i^2}{1+\beta_i^2}$	
NoPenalty	none	Does not use a penalty. You can use this option if you have a large amount of data and you want the fitting process to go quickly. However, this option can lead to models with lower predictive performance than models that use a penalty.

The Weights

The weights θ (theta) and *w* are typically initialized to random values in the range -0.05 to +0.05

- JMP uses random normal starting weights

Equivalent to a model with random prediction (in other words, no predictive value)

These initial weights are used in the first round of training

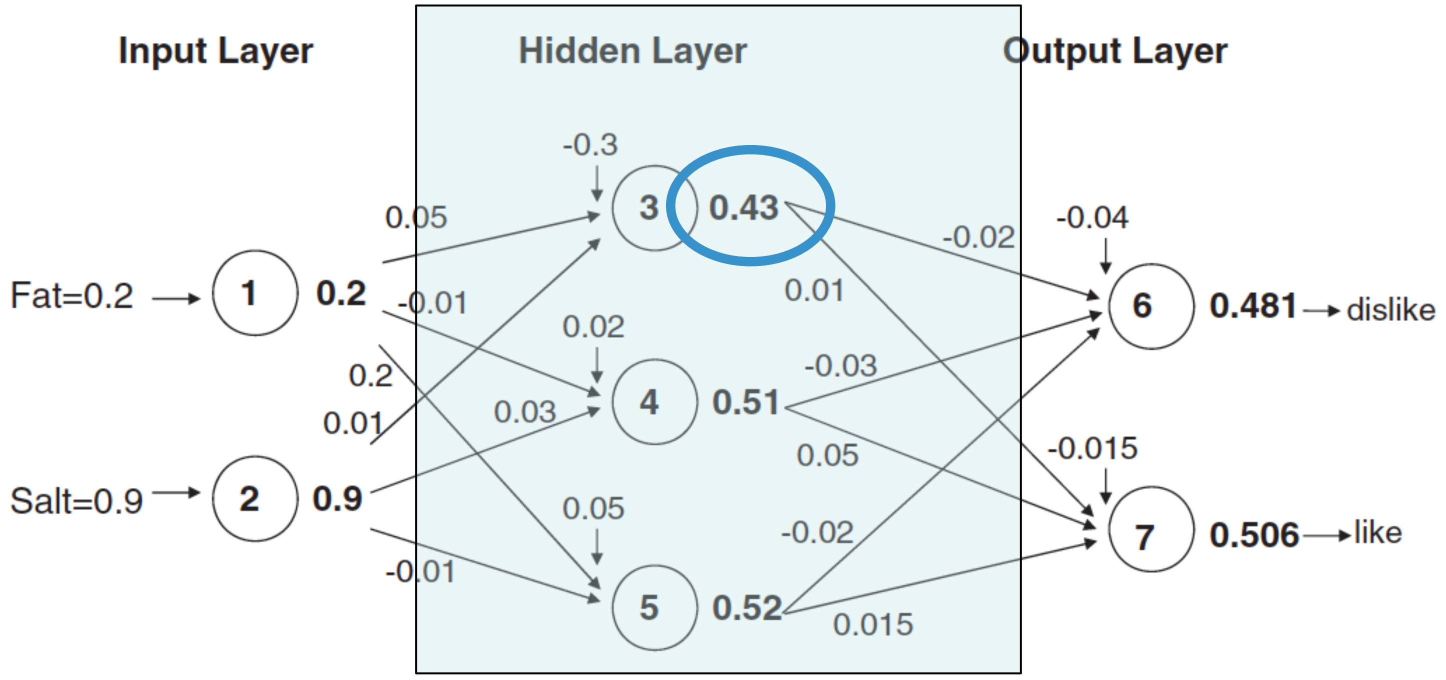
Output of Node 3, if *g* is a Logistic Function

$$output_j = g(\Theta_j + \sum_{i=1}^{p} w_{ij}\, x_i)$$

$$\text{Output}_j = g\left(\theta_j + \sum_{i=1}^{p} w_{ij} x_i\right) = \frac{1}{1 + e^{-(\theta_j + \sum_{i=1}^{p} w_{ij} x_i)}}.$$

$$output_3 = \frac{1}{1 + e^{-[-0.3+(0.05)(0.2)+(0.01)(0.9)]}} = \boxed{0.43}$$

Tiny Example Neural Weights



Shmueli, G., Bruce, P., Stephens, M. and Patel, N. (2016) Data Mining for Business Analytics: Concepts, Techniques, and Applications with JMP Pro, Wiley, USA https://www.wiley.com/en-us/Data+Mining+for+Business+Analytics%3A+Concepts%2C+Techniques%2C+and+Applications+with+JMP+Pro-p-9781118877524

Output Layer

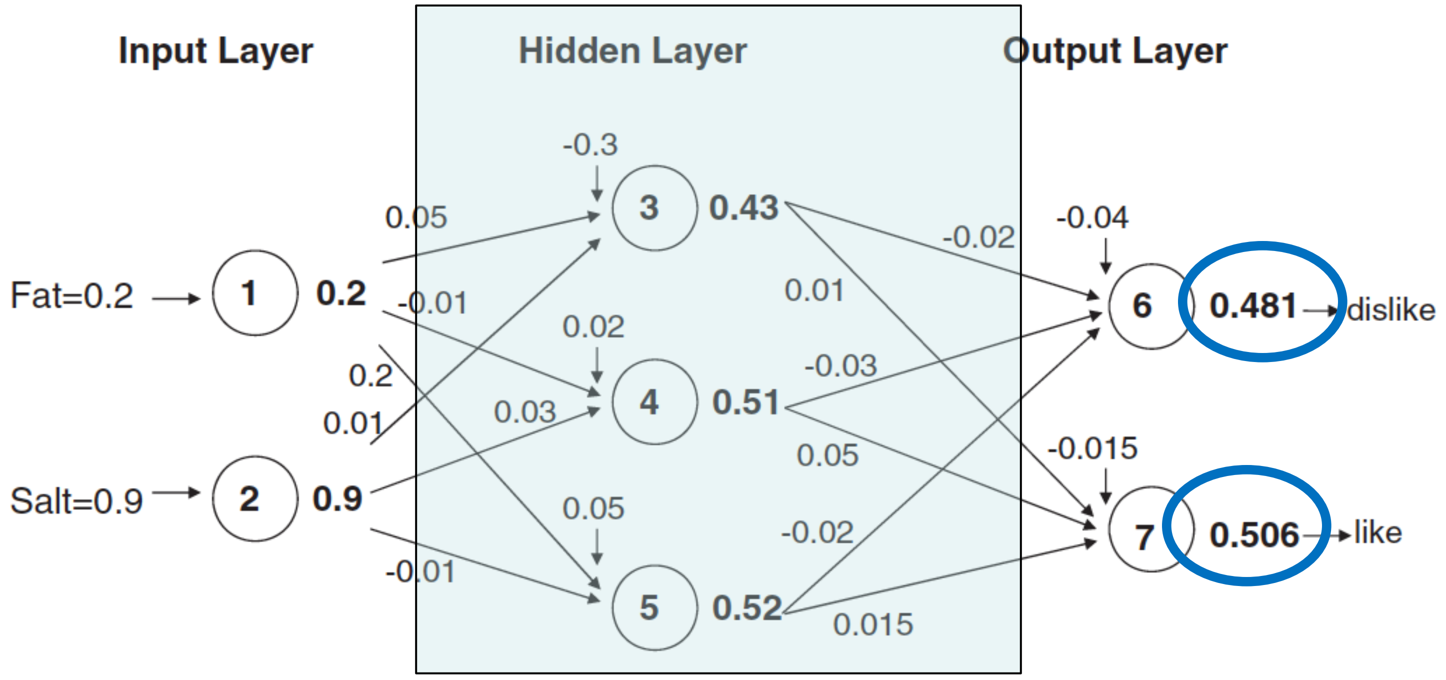
The output of the last hidden layer becomes input for the output layer

Uses same function as above, i.e. a function *g* of the weighted average

$$\text{Output}_6 = \frac{1}{1 + e^{-[-0.04+(-0.02)(0.43)+(-0.03)(0.51)+(0.015)(0.52)]}} = \bigcirc\!\!\!\!\!\!\!\!\!\!\quad 0.481$$

$$\text{Output}_7 = \frac{1}{1 + e^{-[-0.015+(0.01)(0.430)+(0.05)(0.507)+(0.015)(0.511)]}} = 0.506$$

Tiny Example Output Layer



Mapping the output to a classification

These values are normalized so they are propensities (which add up to 1.0).

$$\begin{aligned}
P(Y = \text{Dislike}) &= \text{Output}_6/(\text{Output}_6 + \text{Output}_7) \\
&= 0.481/(0.481 + 0.506) = 0.49 \\
P(Y = \text{Like}) &= 1 - P(Y = \text{Dislike}) \\
&= 0.506/(0.481 + 0.506) = 0.51
\end{aligned}$$

The default cutoff for classification is 0.5.

This first record would be classified as a Like.

Relation to Linear Regression

A net with a single output node and no hidden layers, where *g* is the identity function, takes the same form as a linear regression model

$$\hat{y} = \Theta + \sum_{i=1}^{p} w_i \, x_i$$

Initial Pass-Through Network

Goal: Find weights that yield best predictions

- The process we described above is repeated for all records
- At each record, compare prediction to actual
- Difference is the error for the output node
- Error is propagated back and distributed to all the hidden nodes and used to update their weights

Training the Model

Back Propagation of Error

- Output from output node k: $\hat{y}_k$
- Error associated with that node:

$$err_k = \boxed{\hat{y}_k(1-\hat{y}_k)}(y_k - \hat{y}_k)$$

Note: this is like ordinary error, multiplied by a correction factor

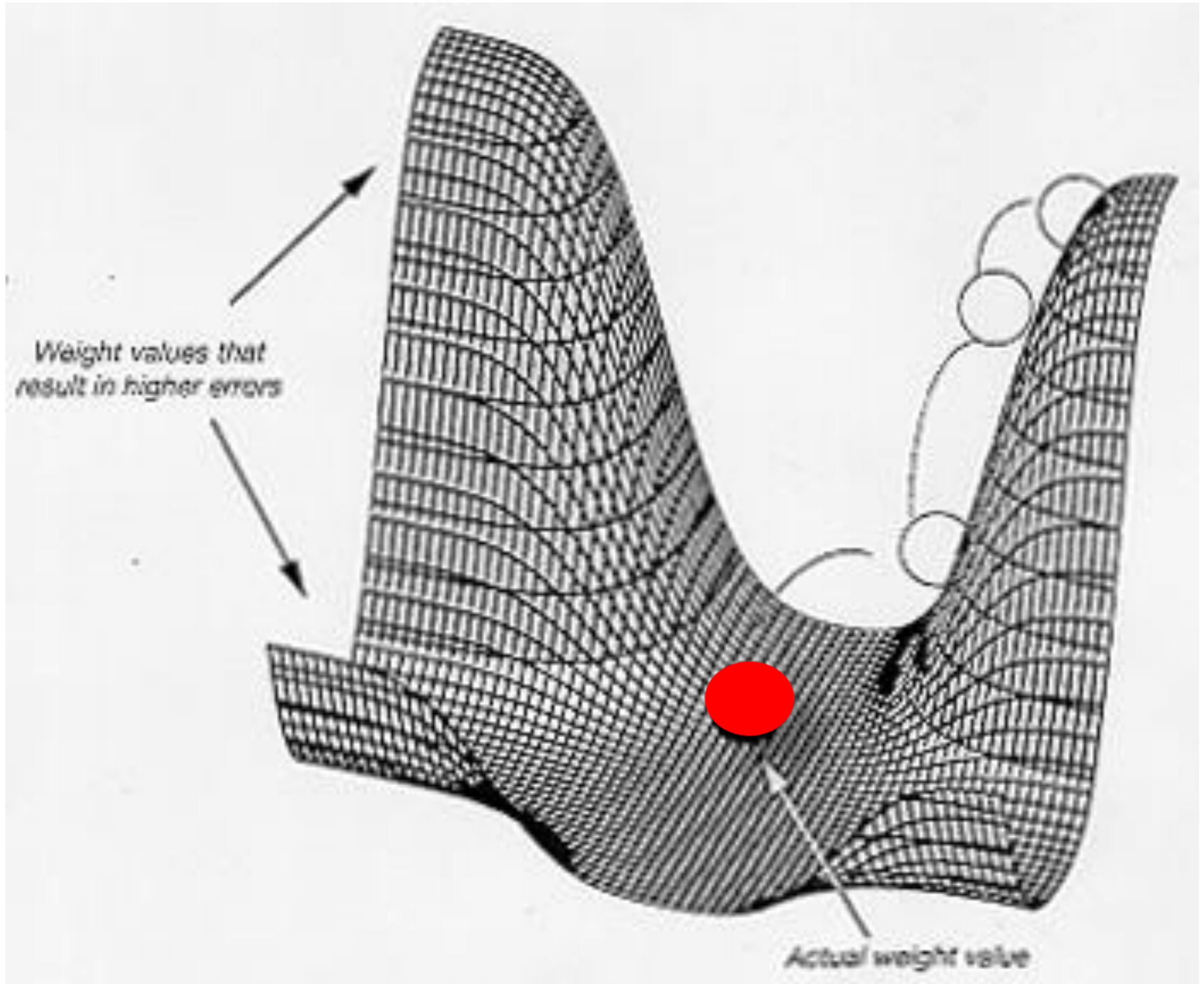
Error is Used to Update Weights

$$\theta_j^{new} = \theta_j^{old} + l(err_j)$$

$$\mathrm{w}_j^{new} = w_j^{old} + l(err_j)$$

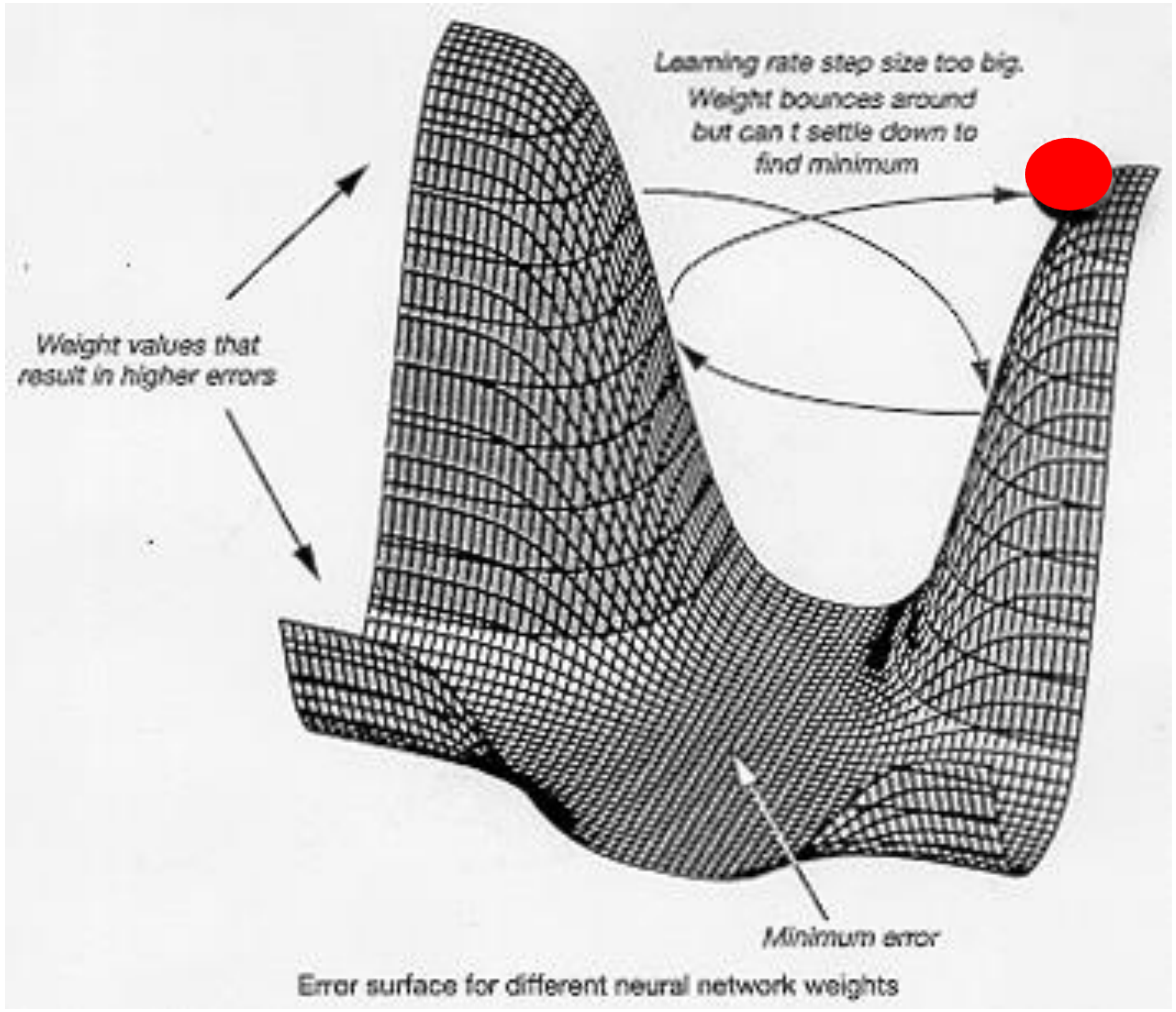
l = constant between 0 and 1, reflects the “learning rate” or “weight decay parameter”

Error is Used to Update Weights



Error surface for different neural network weights

Error is Used to Update Weights



Case Updating

- Weights are updated after each record is run through the network
- Completion of all records through the network is one *epoch* (also called *sweep* or *iteration*)
- After one epoch is completed, return to first record and repeat the process

Case Updating

In case updating, the weights are updated after each record is run through the network (called a *trial*). For example, if we used case updating in the tiny example, the weights would first be updated after running record 1 as follows: Using a learning rate of 0.5, the weights $\theta_7, w_{3,7}, w_{4,7}$, and $w_{5,7}$ are updated to

$$
\begin{aligned}
\theta_7 &= -0.015 + (0.5)(0.123) = 0.047 \\
w_{3,7} &= 0.01 + (0.5)(0.123) = 0.072 \\
w_{4,7} &= 0.05 + (0.5)(0.123) = 0.112 \\
w_{5,7} &= 0.015 + (0.5)(0.123) = 0.077
\end{aligned}
$$

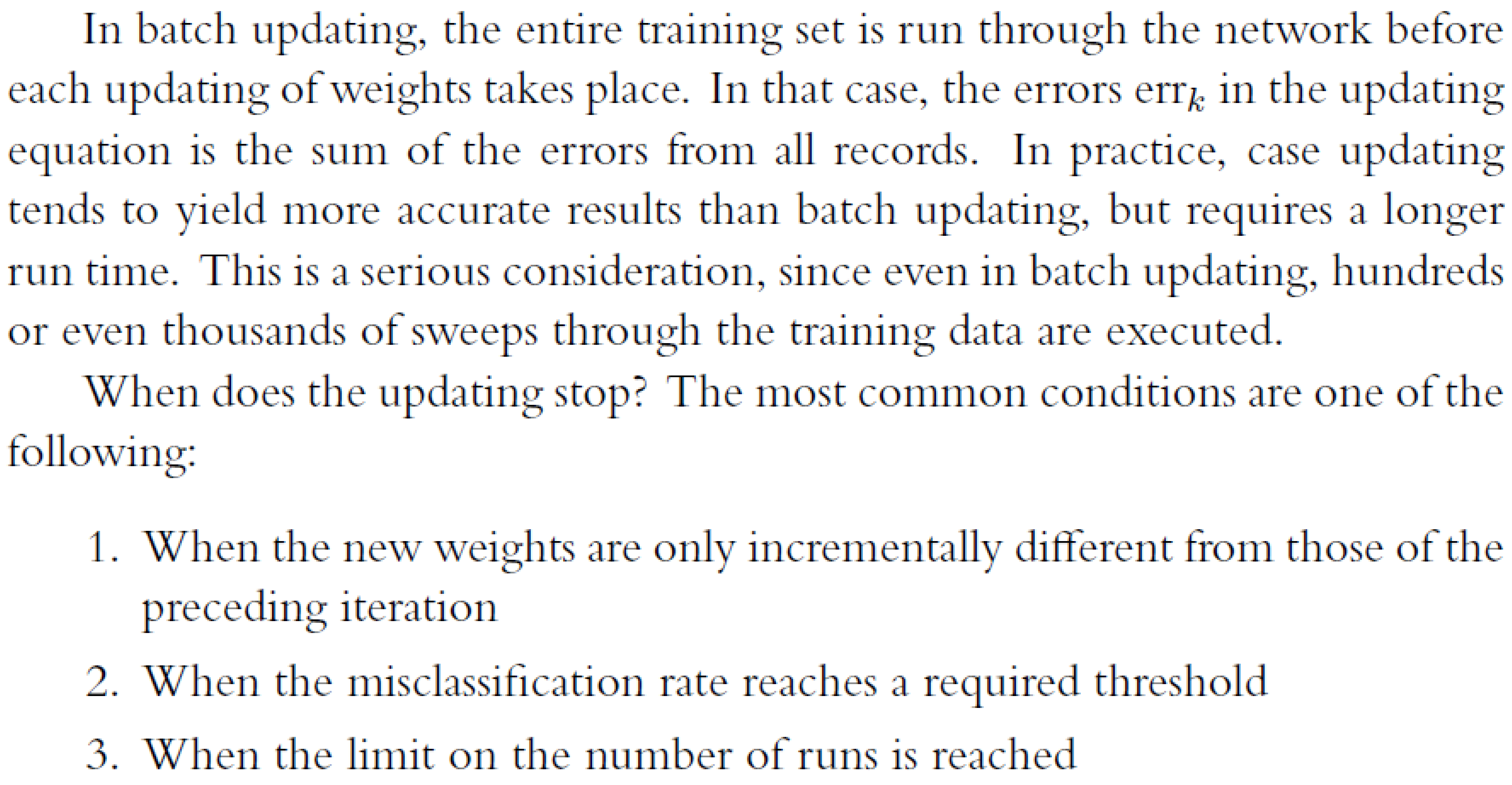
Similarly, we obtain updated weights $\theta_6 = 0.025$, $w_{3,6} = 0.045$, $w_{4,6} = 0.035$, and $w_{5,6} = 0.045$. These new weights are next updated after the second record is run through the network, the third, and so on, until all records are used. This is called one *epoch*, *sweep*, or *iteration* through the data. Typically, there are many iterations.

Batch Updating

- All records in the training set are fed to the network before updating takes place

- In this case, the error used for updating is the sum of all errors from all records

Batch Updating



In batch updating, the entire training set is run through the network before each updating of weights takes place. In that case, the errors err_k in the updating equation is the sum of the errors from all records. In practice, case updating tends to yield more accurate results than batch updating, but requires a longer run time. This is a serious consideration, since even in batch updating, hundreds or even thousands of sweeps through the training data are executed.

When does the updating stop? The most common conditions are one of the following:

1. When the new weights are only incrementally different from those of the preceding iteration
2. When the misclassification rate reaches a required threshold
3. When the limit on the number of runs is reached

Why It Works

- Big errors lead to big changes in weights
- Small errors leave weights relatively unchanged
- Over thousands of updates, a given weight keeps changing until the error associated with that weight is negligible, at which point weights change little

Common Criteria to Stop the Updating

- When weights change very little from one iteration to the next
- When the misclassification rate reaches a required threshold
- When a limit on runs is reached

Neural Model fitting in JMP

- JMP uses an algorithm that finds optimal values of weights and bias values that minimize a function of the combined errors (maximum likelihood)
- This approach produces similar results to back propagation, but
  - ➢Its generally much faster
  - ➢It can be used for both continuous and categorical responses

Neural Model fitting in JMP

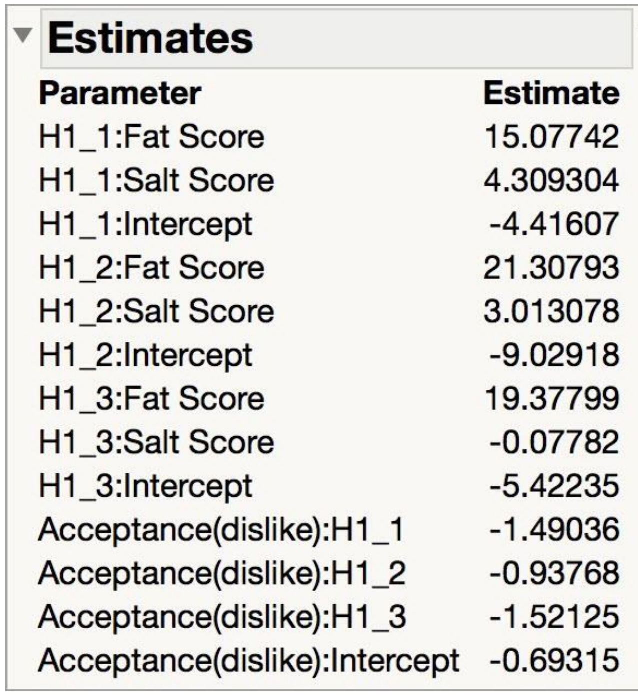
Neural models tend to overfit the data.

To avoid overfitting, JMP uses a **penalty parameter** and requires **crossvalidation**

The JMP Neural fitting process:

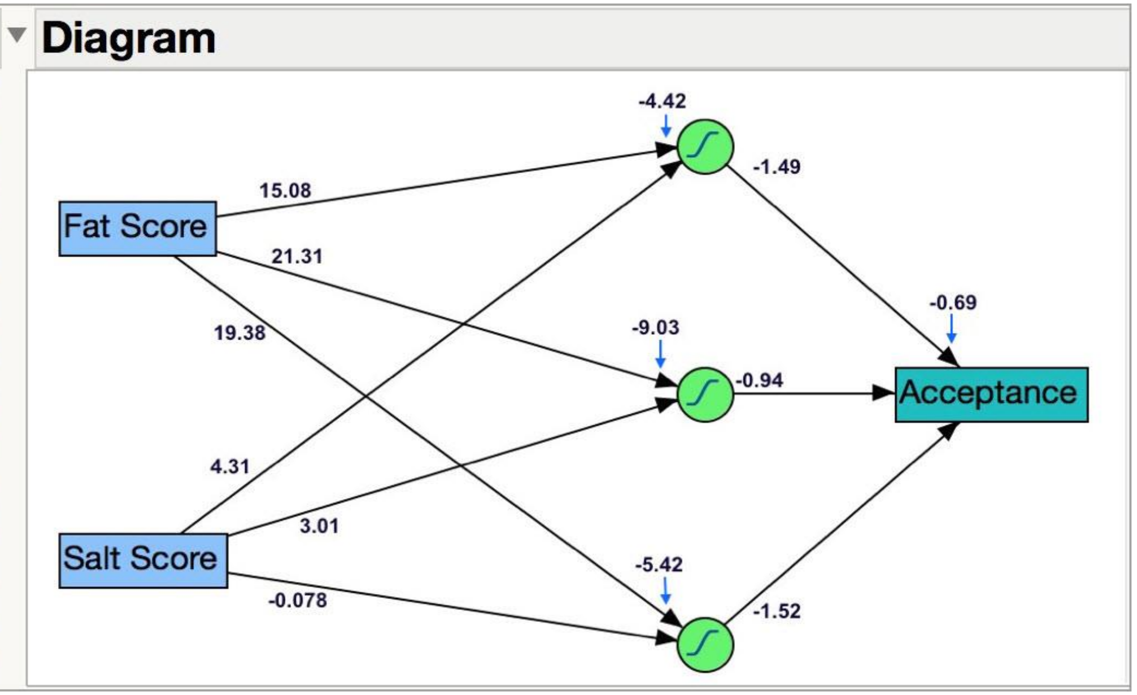
1. Set the penalty to 0
2. Use random normal weights for the starting values
3. Vary the penalty parameter
4. For each value of the penalty parameter, search for weights that minimize error
5. Select the model with the lowest crossvalidation error

Tiny Example: Final Weights

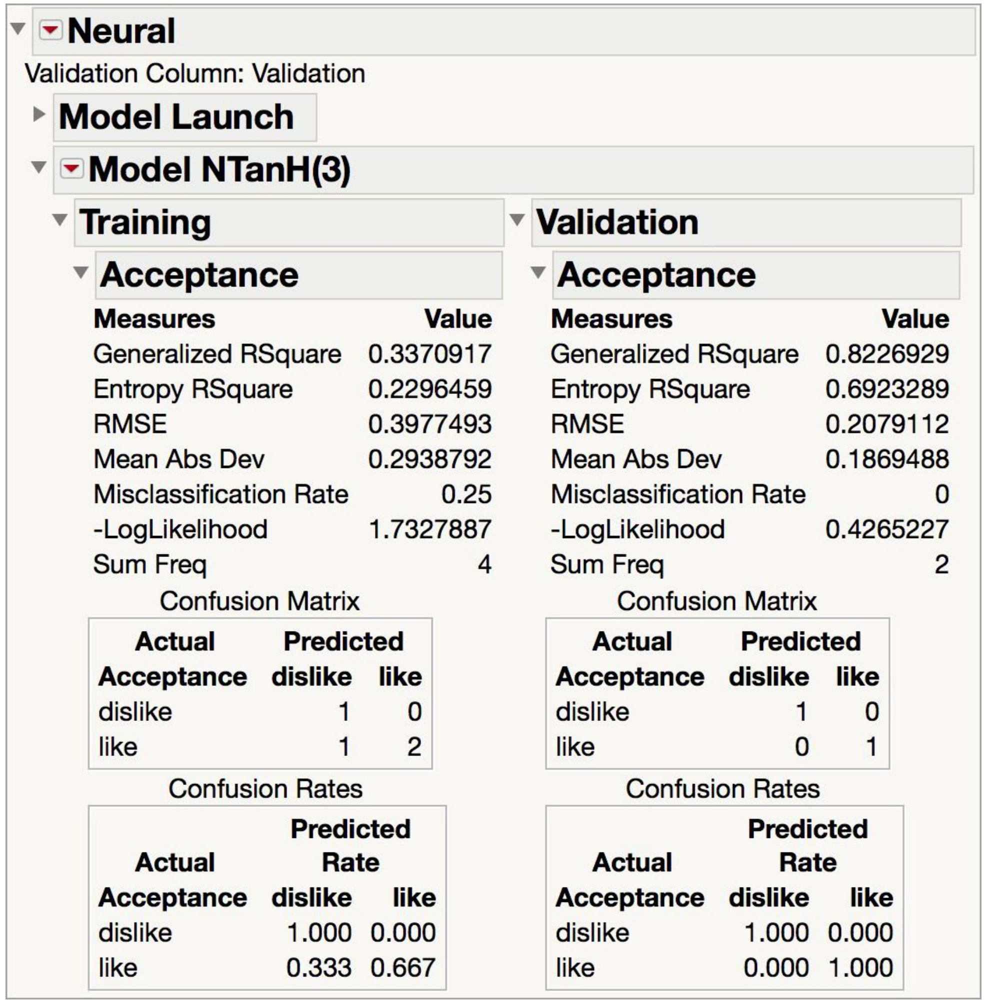


Estimates

Parameter	Estimate
H1_1:Fat Score	15.07742
H1_1:Salt Score	4.309304
H1_1:Intercept	-4.41607
H1_2:Fat Score	21.30793
H1_2:Salt Score	3.013078
H1_2:Intercept	-9.02918
H1_3:Fat Score	19.37799
H1_3:Salt Score	-0.07782
H1_3:Intercept	-5.42235
Acceptance(dislike):H1_1	-1.49036
Acceptance(dislike):H1_2	-0.93768
Acceptance(dislike):H1_3	-1.52125
Acceptance(dislike):Intercept	-0.69315



Tiny Example: Fit Statistics



Neural

Validation Column: Validation

Model Launch

Model NTanH(3)

Training

Acceptance

Measures	Value
Generalized RSquare	0.3370917
Entropy RSquare	0.2296459
RMSE	0.3977493
Mean Abs Dev	0.2938792
Misclassification Rate	0.25
-LogLikelihood	1.7327887
Sum Freq	4

Confusion Matrix

Actual	Predicted	
Acceptance	dislike	like
dislike	1	0
like	1	2

Confusion Rates

	Predicted Rate	
Actual Acceptance	dislike	like
dislike	1.000	0.000
like	0.333	0.667

Validation

Acceptance

Measures	Value
Generalized RSquare	0.8226929
Entropy RSquare	0.6923289
RMSE	0.2079112
Mean Abs Dev	0.1869488
Misclassification Rate	0
-LogLikelihood	0.4265227
Sum Freq	2

Confusion Matrix

Actual	Predicted	
Acceptance	dislike	like
dislike	1	0
like	0	1

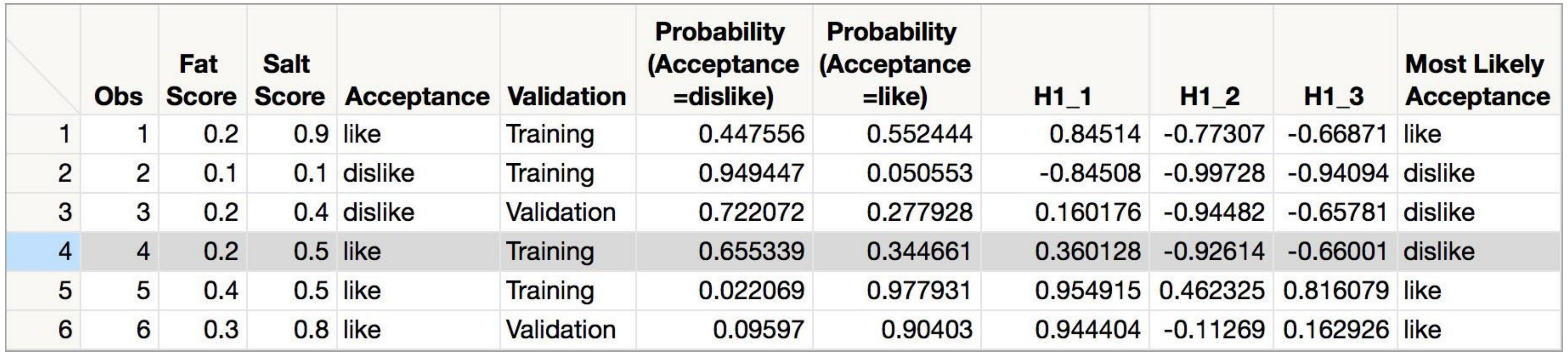
Confusion Rates

	Predicted Rate	
Actual Acceptance	dislike	like
dislike	1.000	0.000
like	0.000	1.000

Tiny Example: Classifications

Estimated propensities and classifications

- One record in the training set was misclassified
- Both records in the validation set were correctly classified



	Obs	Fat Score	Salt Score	Acceptance	Validation	Probability (Acceptance =dislike)	Probability (Acceptance =like)	H1_1	H1_2	H1_3	Most Likely Acceptance
1	1	0.2	0.9	like	Training	0.447556	0.552444	0.84514	-0.77307	-0.66871	like
2	2	0.1	0.1	dislike	Training	0.949447	0.050553	-0.84508	-0.99728	-0.94094	dislike
3	3	0.2	0.4	dislike	Validation	0.722072	0.277928	0.160176	-0.94482	-0.65781	dislike
4	4	0.2	0.5	like	Training	0.655339	0.344661	0.360128	-0.92614	-0.66001	dislike
5	5	0.4	0.5	like	Training	0.022069	0.977931	0.954915	0.462325	0.816079	like
6	6	0.3	0.8	like	Validation	0.09597	0.90403	0.944404	-0.11269	0.162926	like

Specify Network Architecture in JMP

Number of hidden layers

- Most popular – one hidden layer

Number of nodes in hidden layer(s)

- More nodes capture complexity, but increase chances of overfit

Hidden Layer Activation Functions

- Combinations of three functions (TanH, Linear and Gaussian) can be applied in the hidden layers to add model complexity

User Inputs

JMP Network Architecture, cont.

Number of tours

- How many times JMP restarts the model-fitting algorithm

“Learning Rate”

- Low values “downweight” the new information from errors at each iteration
- This slows learning, but reduces tendency to overfit to local structure

Advantages

- Good predictive ability
- Can capture complex relationships
- No need to specify a model

Disadvantages

- Considered a “black box” prediction machine, with no insight into relationships between predictors and outcome
- No variable-selection mechanism, so you have to exercise care in selecting variables
- Heavy computational requirements if there are many variables

Addressing Disadvantages in JMP

- Considered a “black box” prediction machine, with no insight into relationships between predictors and outcome
  - ➢**JMP Prediction Profiler allows you to explore the model**
- No variable-selection mechanism, so you have to exercise care in selecting variables
  - ➢**JMP Variable Importance can help identify most important variables**
- Heavy computational requirements if there are many variables
  - ➢**The JMP Neural algorithm is more efficient than back propagation**

A journey in random forests and penalized regression in an industrial classification problem: How to use models to sharpen your questions

Ron S. Kenett[1], Chris Gotwalt[2] and Jean Michel Poggi[3]

1 The KPA Group and the Samuel Neaman Institute, Technion, Israel, corresponding author: ron@kpa-group.com
2 JMP Division, SAS, Research Triangle, NC, USA
3 Laboratoire de Mathématiques, Université Paris-Saclay, Orsay, and Université Paris Cité, France

Abstract

The mathematician and population geneticist Sam Karlin considered that "The purpose of models is not to fit the data but to sharpen the question". Motivated by this, we describe here a journey through questions, models and data analysis, to reach specific goals. Specifically, we consider random forests, ridge regression, lasso and elastic nets in a case study of 63 sensors collected in functional testing of an electronic system. The paper lists a sequence of questions and how they were tackled by statistical analysis, sometimes unsuccessfully. Eventually we were able to provide a robust, parsimonious and effective model for predicting the system condition, using a small subset of the 63 sensors. In handling this problem, we combine several innovative methods and insights that can prove useful also in other contexts. The underlying objective is enhancing the awareness to the journey representing the modeling process. We conclude with an assessment of the information quality provided by the analysis.