US 20160071017A1

(19) **United States**

(12) **Patent Application Publication**

Adjaoute

(10) **Pub. No.: US 2016/0071017 A1**

(43) **Pub. Date: Mar. 10, 2016**

(54) **METHOD OF OPERATING ARTIFICIAL INTELLIGENCE MACHINES TO IMPROVE PREDICTIVE MODEL TRAINING AND PERFORMANCE**

(71) Applicant: **Brighterion, Inc.**, San Francisco, CA (US)

(72) Inventor: **Akli Adjaoute**, Mill Valley, CA (US)

(73) Assignee: **BRIGHTERION, INC.**, San Francisco, CA (US)

(21) Appl. No.: **14/941,586**

(22) Filed: **Nov. 14, 2015**

Related U.S. Application Data

(63) Continuation-in-part of application No. 14/815,848, filed on Jul. 31, 2015, which is a continuation-in-part of application No. 14/514,381, filed on Oct. 15, 2014, now abandoned, Continuation-in-part of application No. 14/521,667, filed on Oct. 23, 2014, Continuation-in-part of application No. 14/815,934, filed on Jul. 31, 2015.

Publication Classification

(51) **Int. Cl.**

G06N 5/04 (2006.01)

G06N 5/02 (2006.01)

(52) **U.S. Cl.**

CPC . ***G06N 5/048*** (2013.01); ***G06N 5/02*** (2013.01)

(57) **ABSTRACT**

A method of improving the training and performance of predictive models. A first method of operating an artificial intelligence machine produces predictive model language documents describing improved predictive models that generate better business decisions from raw data record inputs. A second method of operating an artificial intelligence machine including processors for predictive model algorithms produces and outputs better business decisions from raw data record inputs. Both methods enrich the raw data records their processors are fed by deleting data fields with data values that have little benefit in decision making, and that derive and add new data fields from information sources then available that do benefit in the decision making of the artificial intelligence machine through improved accuracies of prediction.

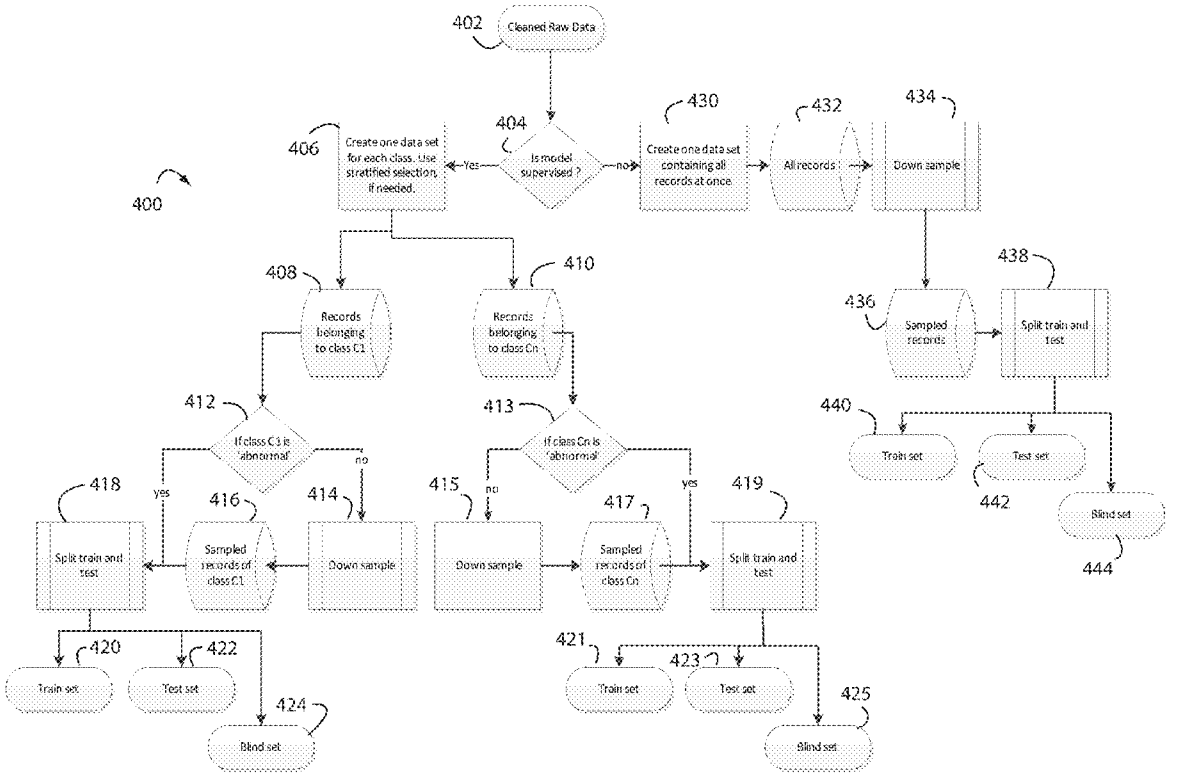


Fig. 1

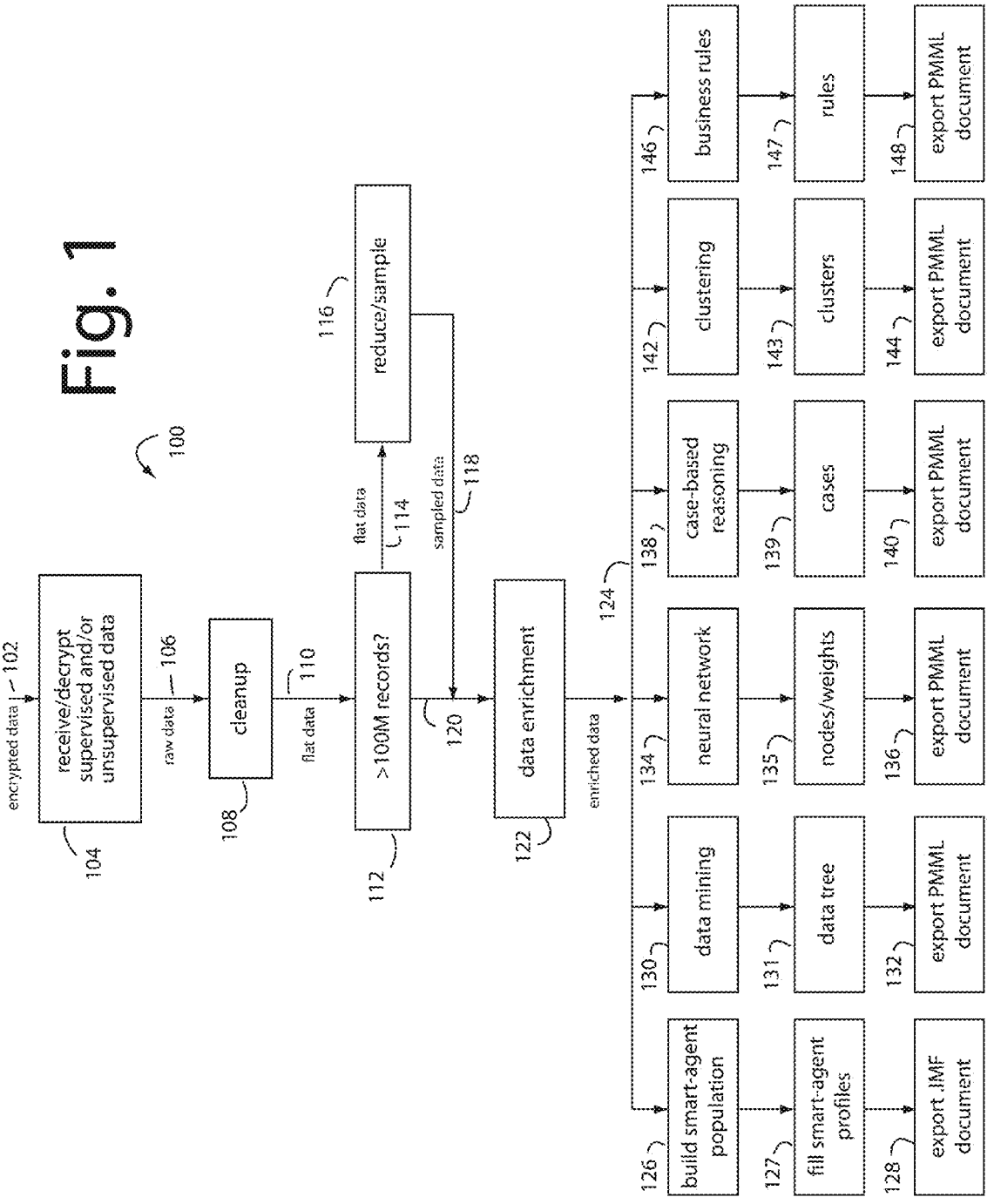
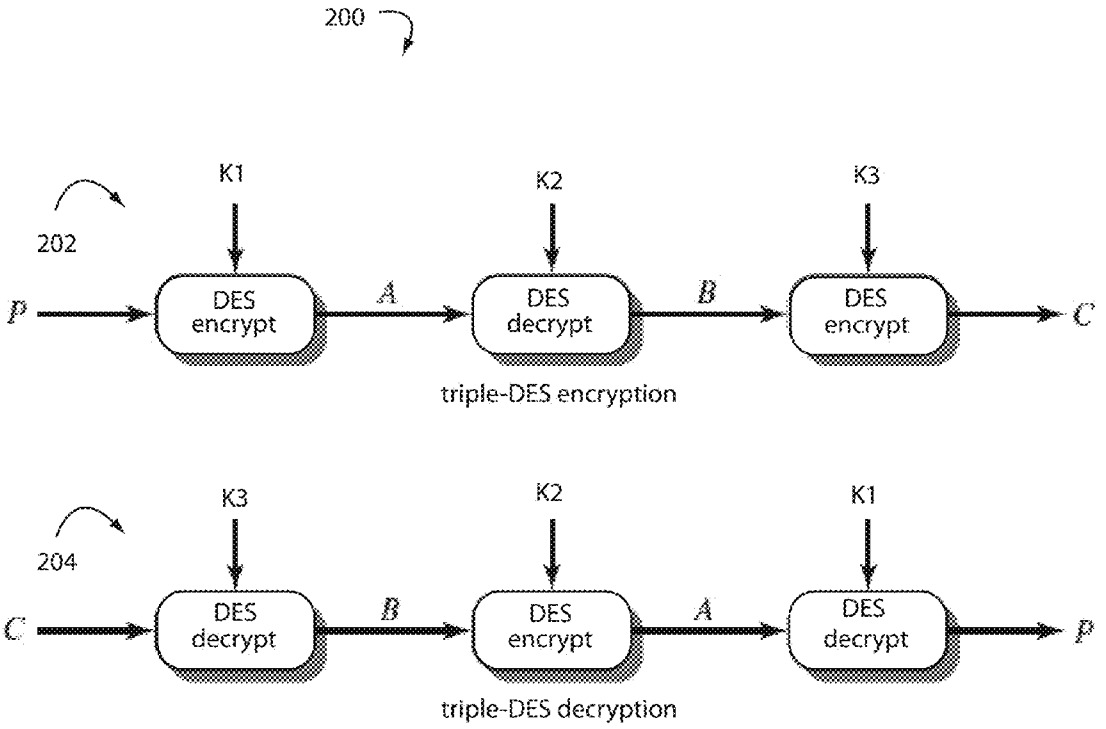

100
encrypted data
102
receive/decrypt supervised and/or unsupervised data
104
raw data
106
cleanup
108
flat data
110
>100M records?
112
flat data
114
reduce/sample
116
sampled data
118
120
data enrichment
122
enriched data
124
build smart-agent population
126
data mining
130
neural network
134
case-based reasoning
138
clustering
142
business rules
146
fill smart-agent profiles
127
data tree
131
nodes/weights
135
cases
139
clusters
143
rules
147
export JMF document
128
export PMML document
132
export PMML document
136
export PMML document
140
export PMML document
144
export PMML document
148

Fig. 2


200
202
K1
K2
K3
P
DES encrypt
A
DES decrypt
B
DES encrypt
C
triple-DES encryption
204
K3
K2
K1
C
DES decrypt
B
DES encrypt
A
DES decrypt
P
triple-DES decryption

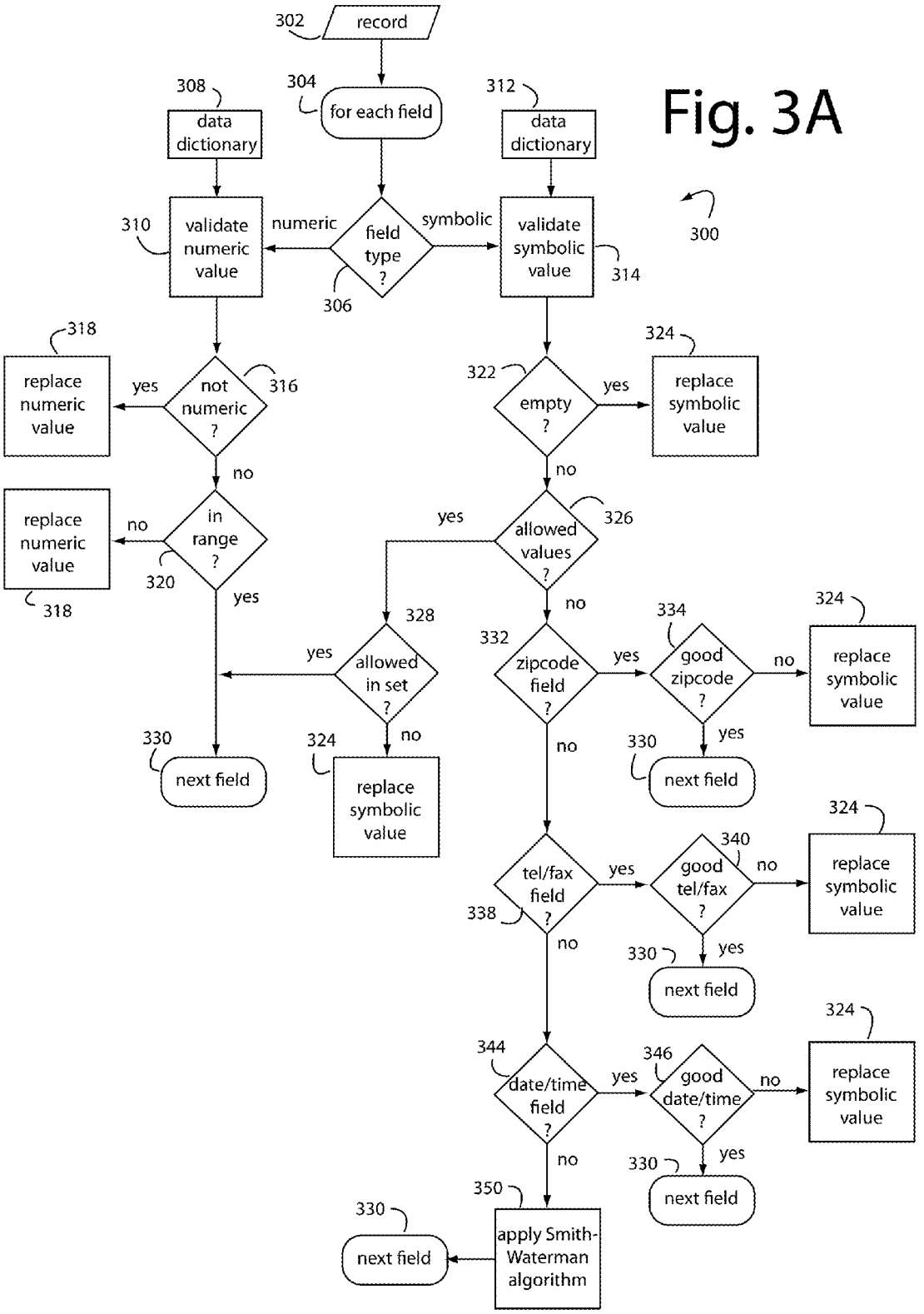
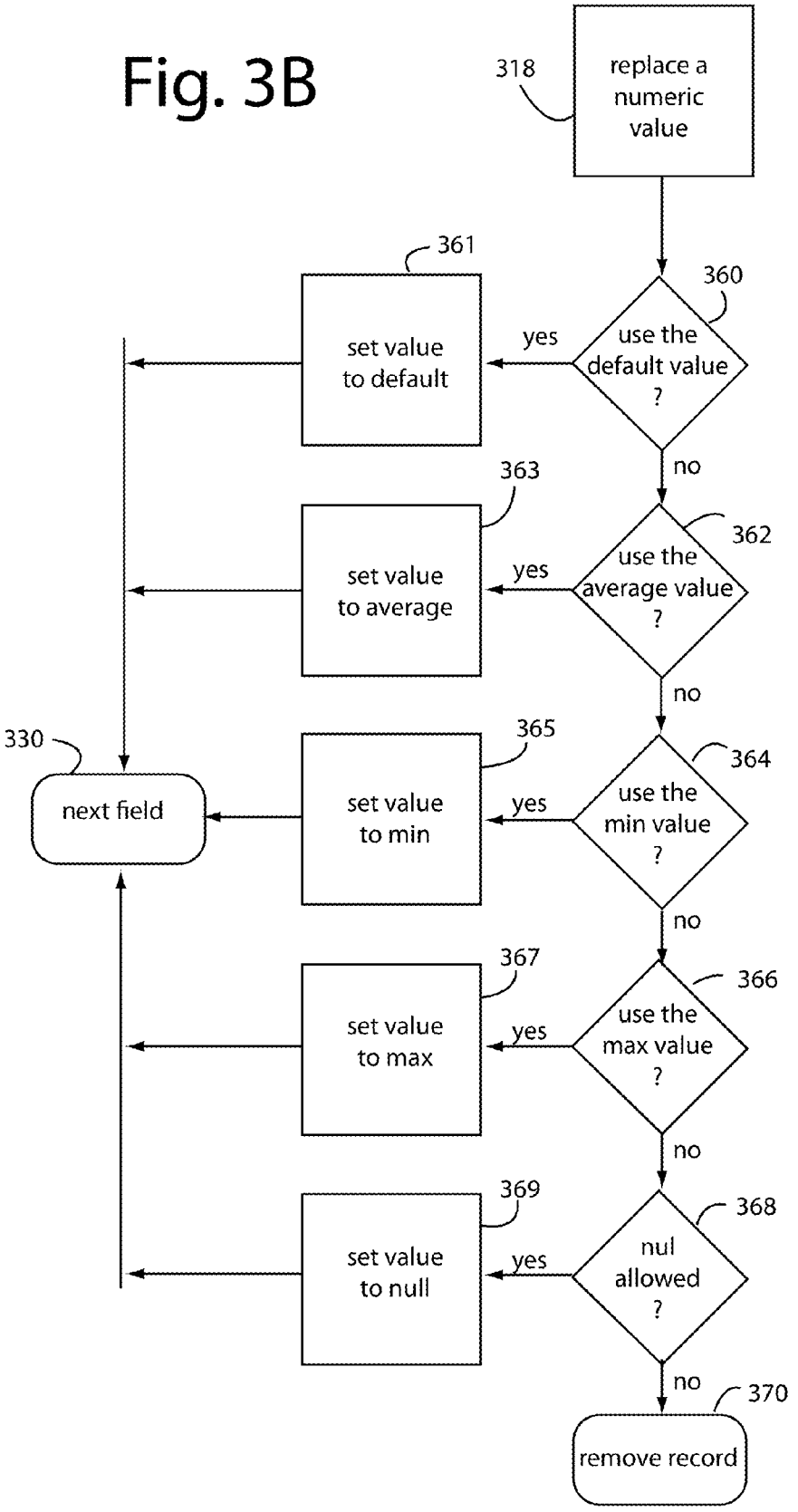

302
record
304
for each field
308
data dictionary
312
data dictionary
310
validate numeric value
numeric
field type ?
306
symbolic
validate symbolic value
314
300
318
replace numeric value
yes
not numeric ?
316
322
empty ?
yes
324
replace symbolic value
no
in range ?
320
no
yes
allowed values ?
326
yes
328
allowed in set ?
yes
no
332
zipcode field ?
yes
334
good zipcode ?
no
yes
330
next field
tel/fax field ?
338
yes
340
good tel/fax ?
no
yes
344
date/time field ?
yes
346
good date/time ?
no
yes
350
apply Smith-Waterman algorithm

Fig. 3A

Fig. 3B


318
replace a numeric value
360
use the default value ?
yes
361
set value to default
no
362
use the average value ?
yes
363
set value to average
no
364
use the min value ?
yes
365
set value to min
no
366
use the max value ?
yes
367
set value to max
no
368
nul allowed ?
yes
369
set value to null
no
370
remove record
330
next field

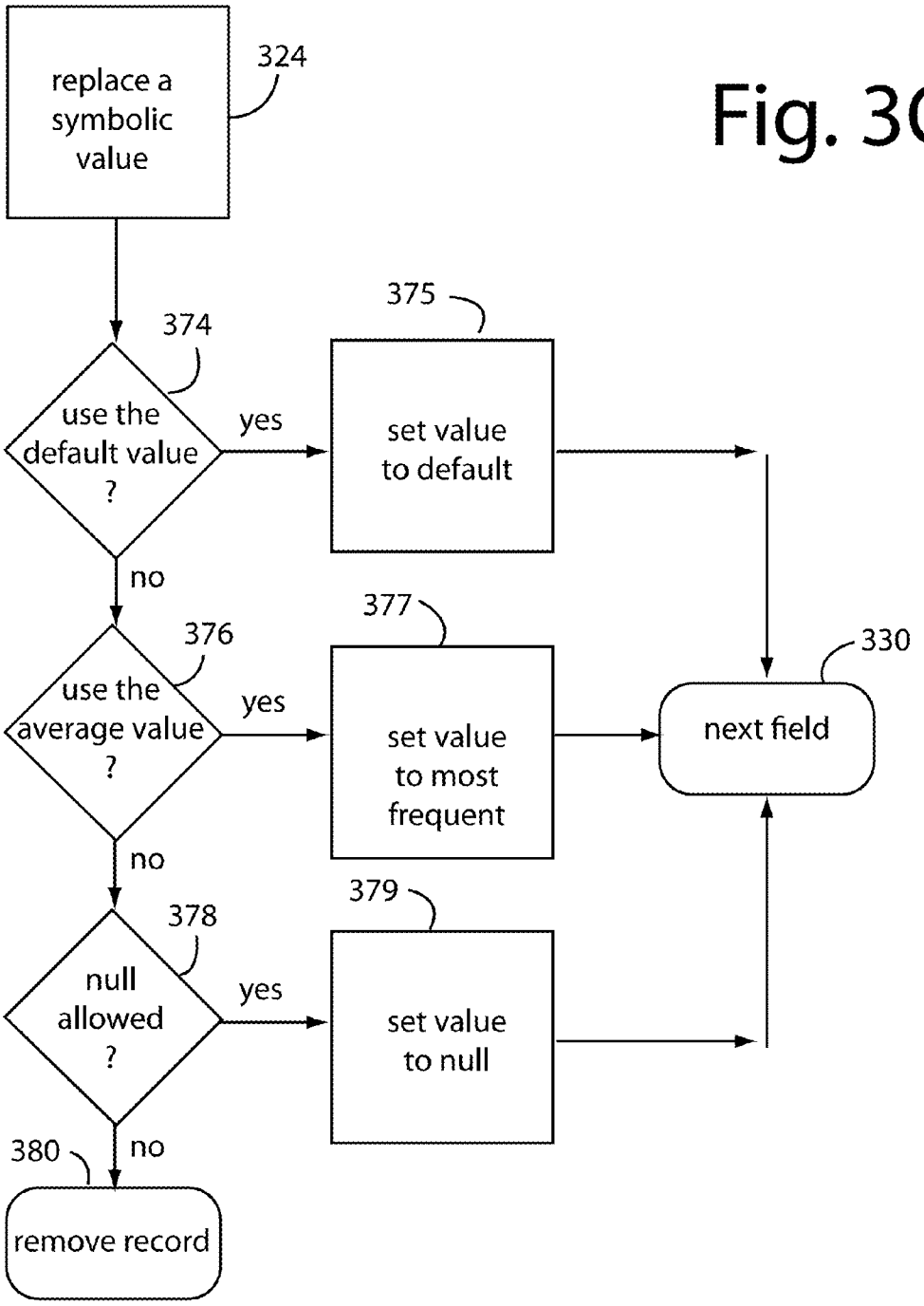

replace a symbolic value
324
use the default value ?
374
yes
set value to default
375
no
use the average value ?
376
yes
set value to most frequent
377
no
null allowed ?
378
yes
set value to null
379
no
remove record
380
next field
330

Fig. 3C

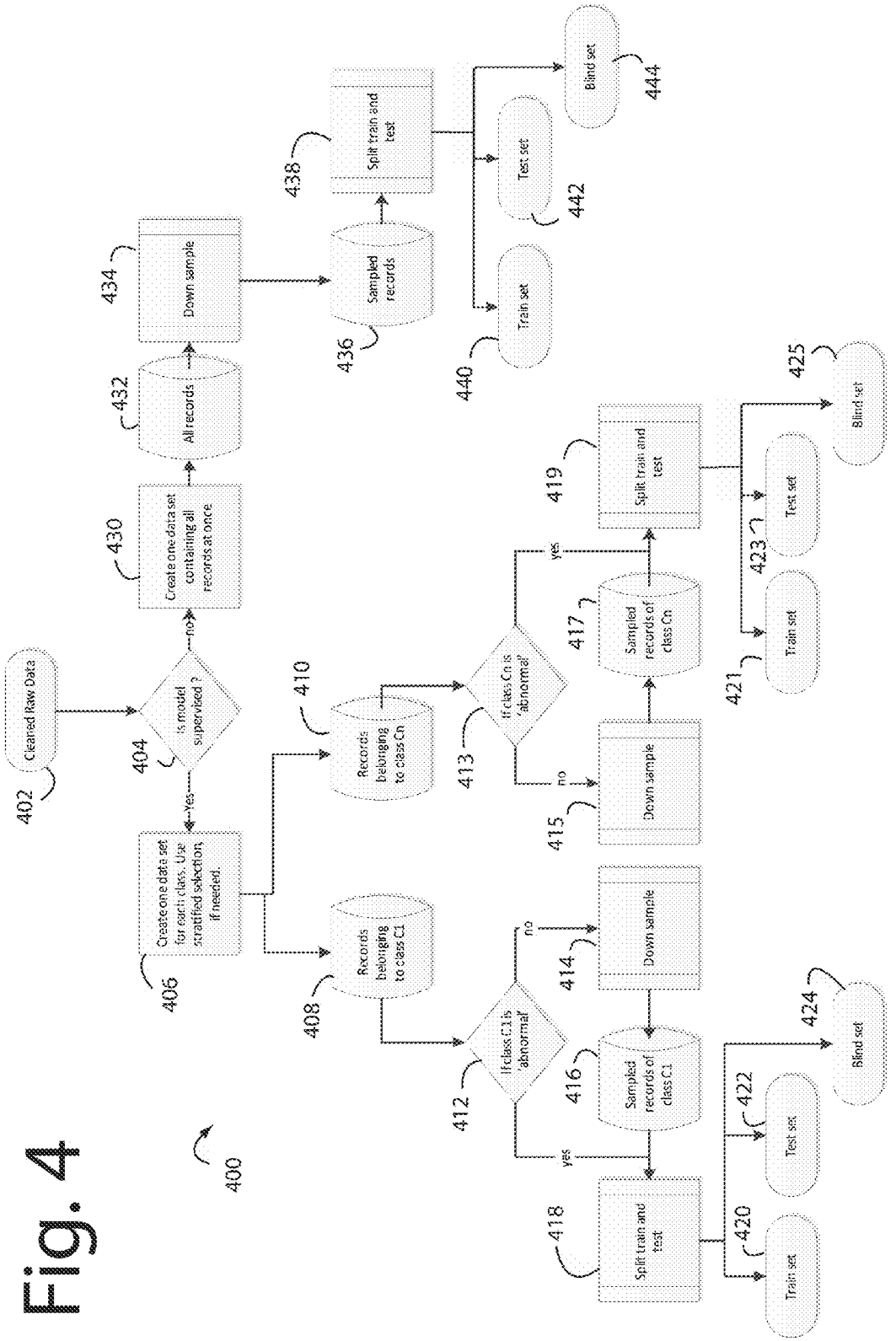

Cleaned Raw Data
402
Is model supervised ?
404
yes
no
Create one data set for each class. Use stratified selection if needed.
406
Records belonging to class C1
408
Records belonging to class Cn
410
If class C1 is 'abnormal'
412
If class Cn is 'abnormal'
413
Down sample
414
Down sample
415
Sampled records of class C1
416
Sampled records of class Cn
417
Split train and test
418
Split train and test
419
Train set
420
Test set
422
Blind set
424
Train set
421
Test set
423
Blind set
425
Create one data set containing all records at once
430
All records
432
Down sample
434
Sampled records
436
Split train and test
438
Train set
440
Test set
442
Blind set
444
400

Fig. 4

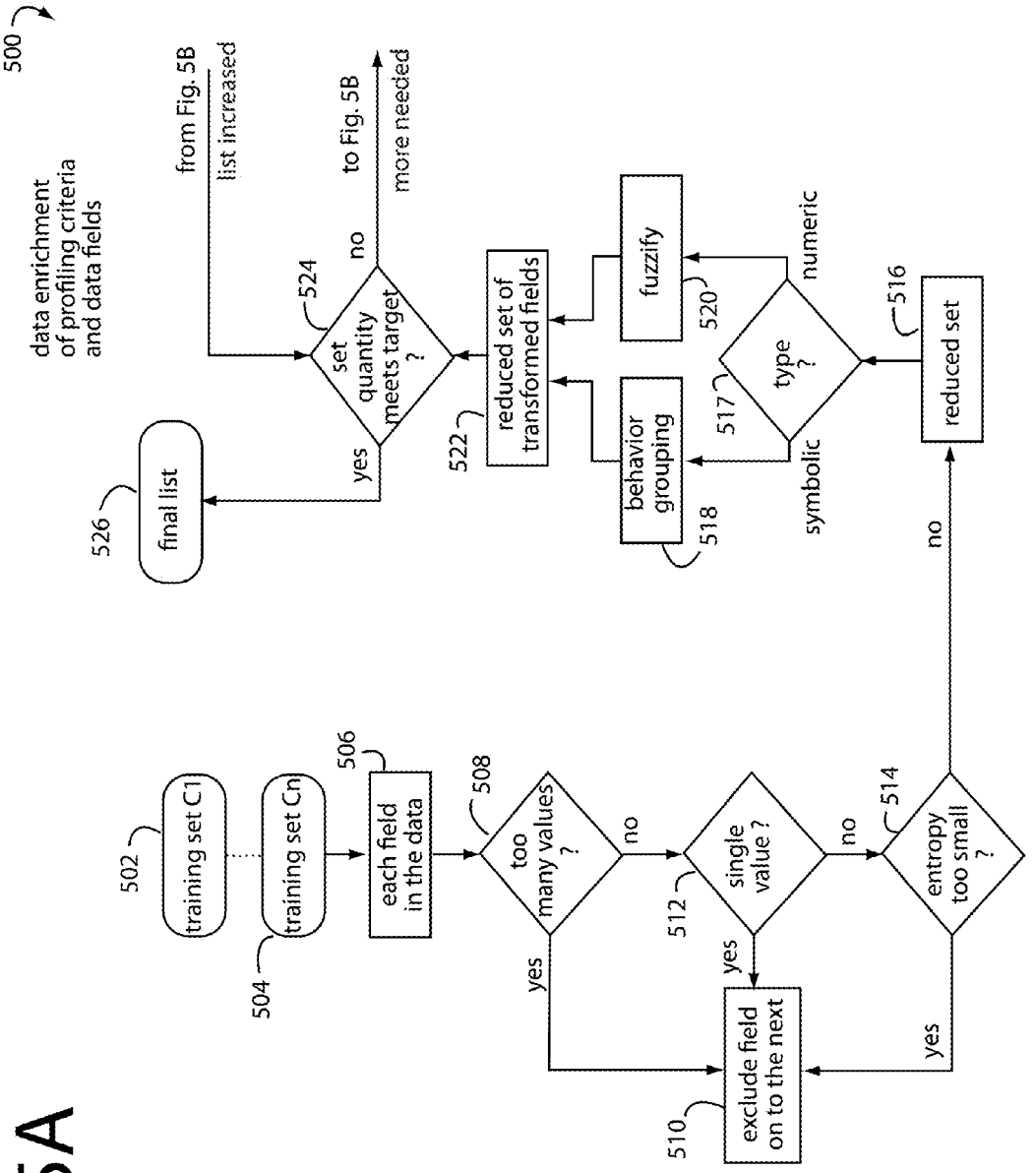

500
data enrichment of profiling criteria and data fields
from Fig. 5B
list increased
to Fig. 5B
more needed
524
set quantity meets target ?
yes
no
526
final list
522
reduced set of transformed fields
fuzzify
behavior grouping
518
517
type ?
symbolic
numeric
520
516
reduced set
502
training set C1
504
training set Cn
506
each field in the data
508
too many values ?
yes
no
512
single value ?
yes
no
514
entropy too small ?
yes
no
510
exclude field on to the next

Fig. 5A

Fig. 5B

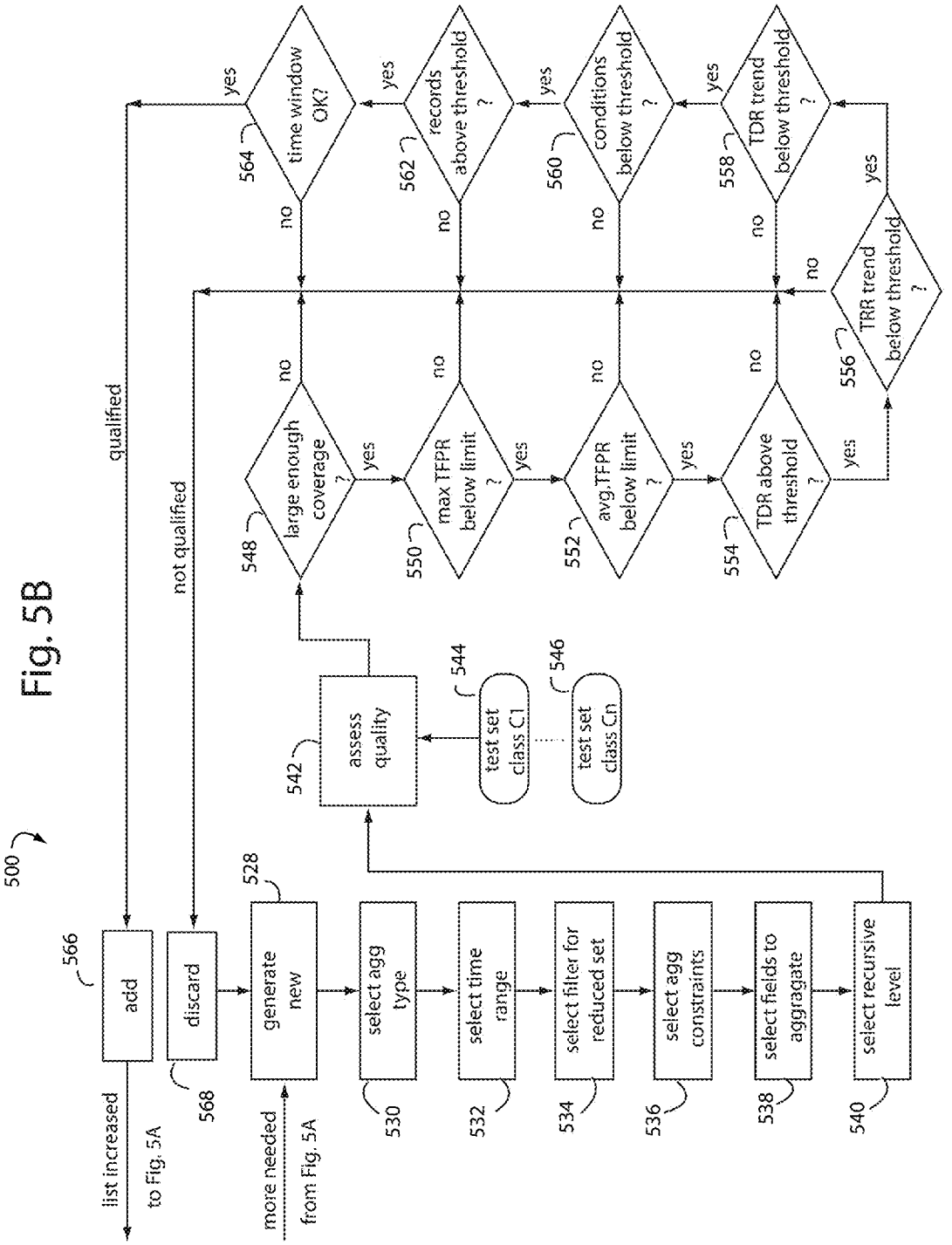
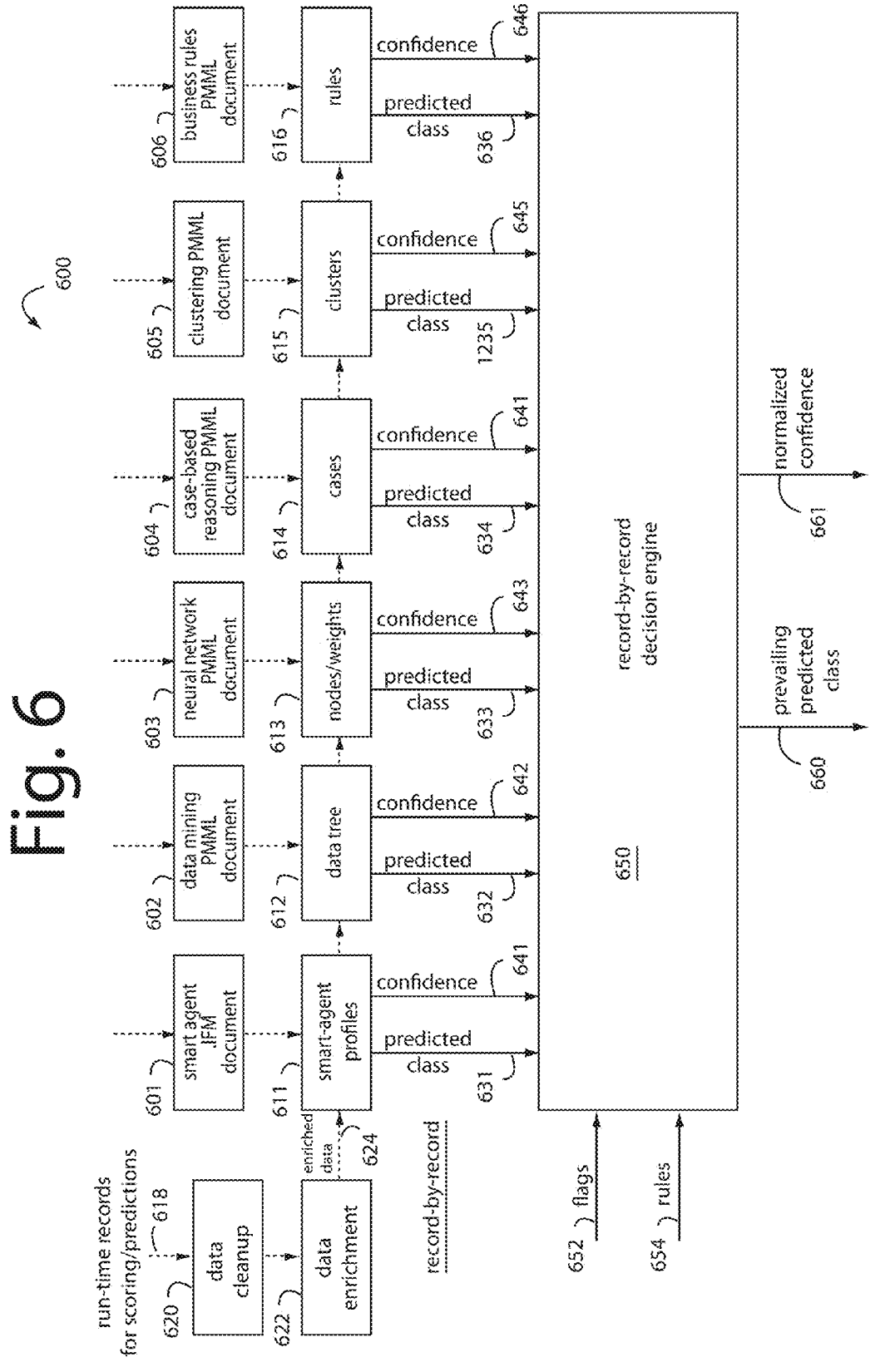

500
566
add
discard
568
528
generate new
more needed
from Fig. 5A
list increased
to Fig. 5A
qualified
not qualified
542
assess quality
544
test set class C1
546
test set class Cn
548
large enough coverage ?
550
max TFPR below limit ?
552
avg.TFPR below limit ?
554
TDR above threshold ?
556
TRR trend below threshold ?
558
TDR trend below threshold ?
560
conditions below threshold ?
562
records above threshold ?
564
time window OK?
530
select agg type
532
select time range
534
select filter for reduced set
536
select agg constraints
538
select fields to aggragate
540
select recursive level
yes
no

Fig. 6


600
run-time records for scoring/predictions
618
620
data cleanup
622
data enrichment
enriched data
624
record-by-record
601
smart agent JFM document
611
smart-agent profiles
602
data mining PMML document
612
data tree
603
neural network PMML document
613
nodes/weights
604
case-based reasoning PMML document
614
cases
605
clustering PMML document
615
clusters
606
business rules PMML document
616
rules
confidence
predicted class
631
632
633
634
636
641
642
643
645
646
1235
650
record-by-record decision engine
652
flags
654
rules
660
prevailing predicted class
661
normalized confidence

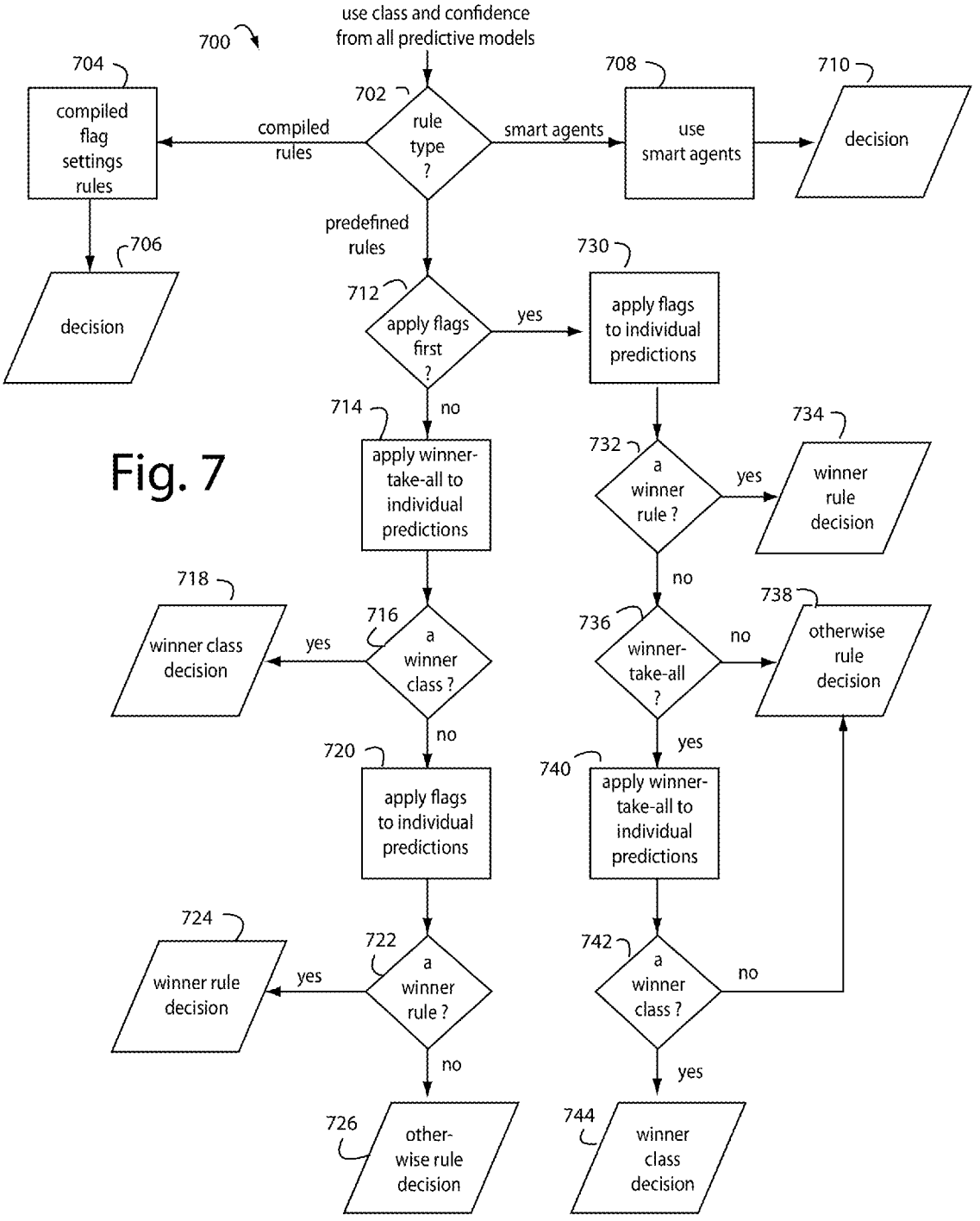

700
use class and confidence from all predictive models
704
compiled flag settings rules
compiled rules
702
rule type ?
smart agents
708
use smart agents
710
decision
706
decision
predefined rules
712
apply flags first ?
yes
730
apply flags to individual predictions
714
no
apply winner-take-all to individual predictions
732
a winner rule ?
yes
734
winner rule decision
no
718
winner class decision
716
a winner class ?
yes
736
winner-take-all ?
no
738
otherwise rule decision
no
720
apply flags to individual predictions
yes
740
apply winner-take-all to individual predictions
724
winner rule decision
722
a winner rule ?
yes
742
a winner class ?
no
no
yes
726
other-wise rule decision
744
winner class decision

Fig. 7

Fig. 8

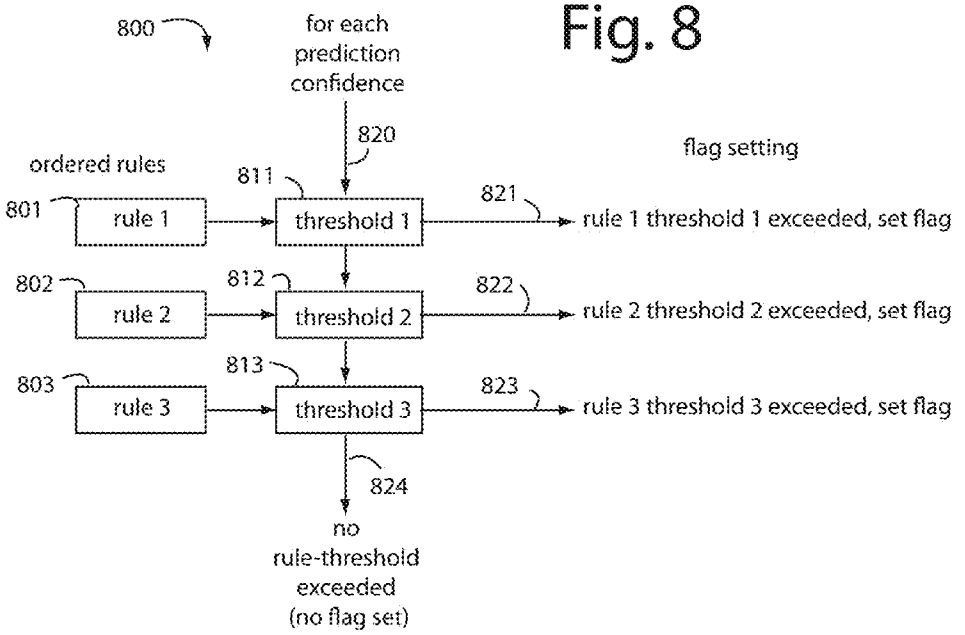

800
for each prediction confidence
820
ordered rules
flag setting
811
801
rule 1
threshold 1
821
rule 1 threshold 1 exceeded, set flag
802
812
rule 2
threshold 2
822
rule 2 threshold 2 exceeded, set flag
803
813
rule 3
threshold 3
823
rule 3 threshold 3 exceeded, set flag
824
no rule-threshold exceeded (no flag set)

Fig. 9

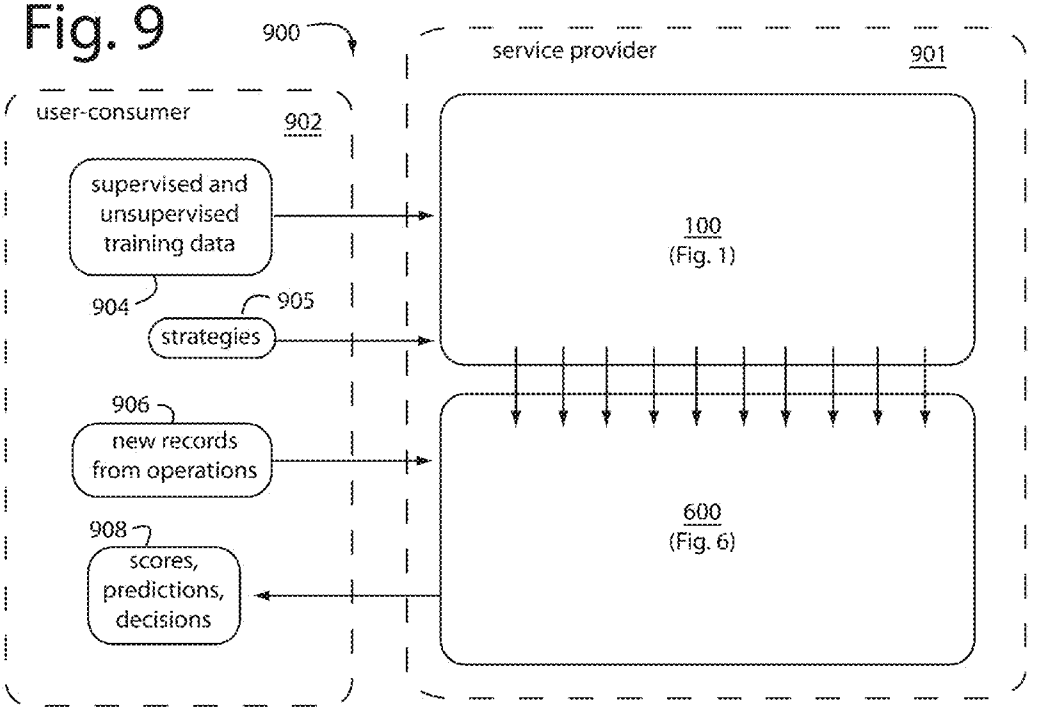

900
service provider
901
user-consumer
902
supervised and unsupervised training data
904
905
strategies
100
(Fig. 1)
906
new records from operations
600
(Fig. 6)
908
scores, predictions, decisions

Fig. 10

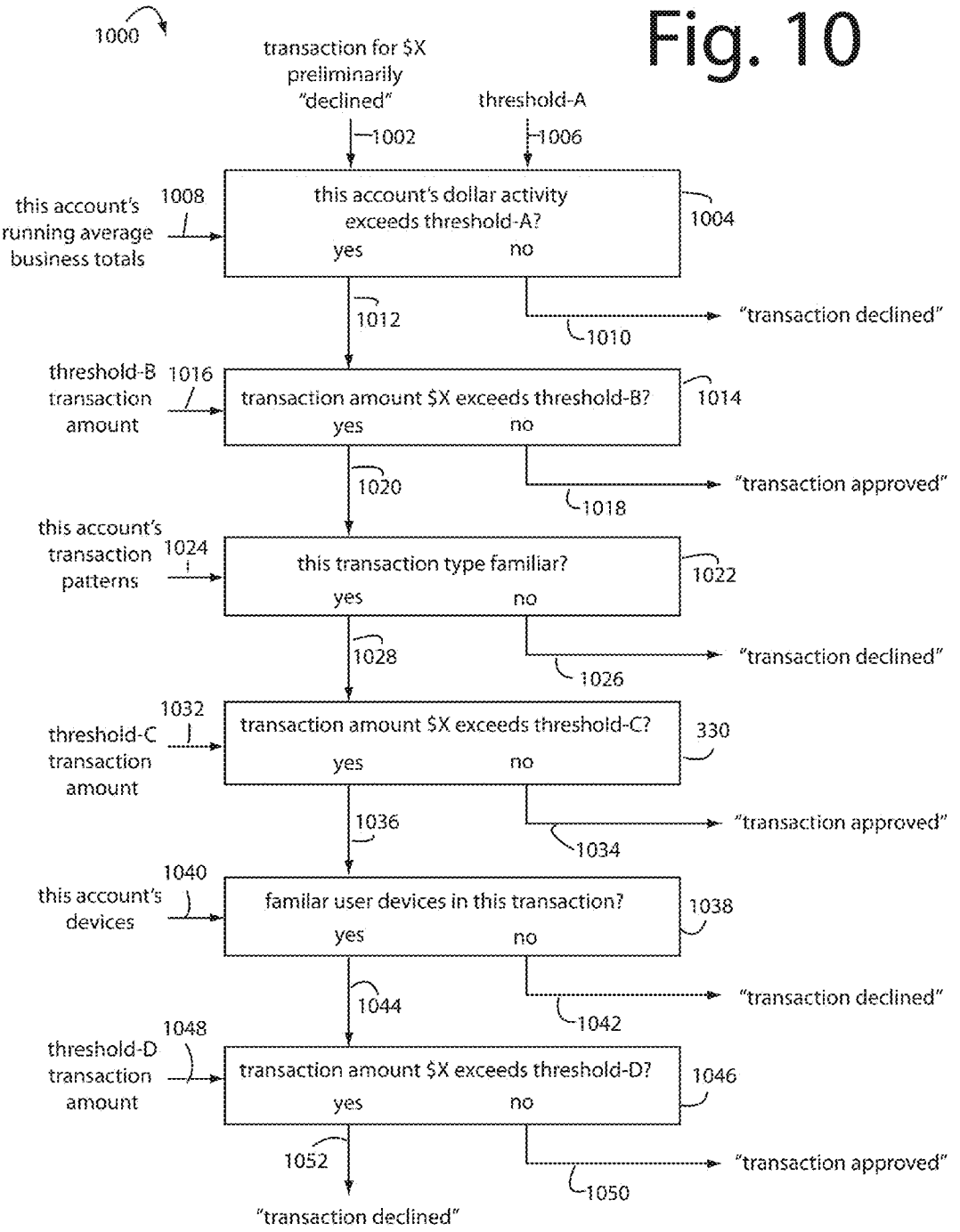
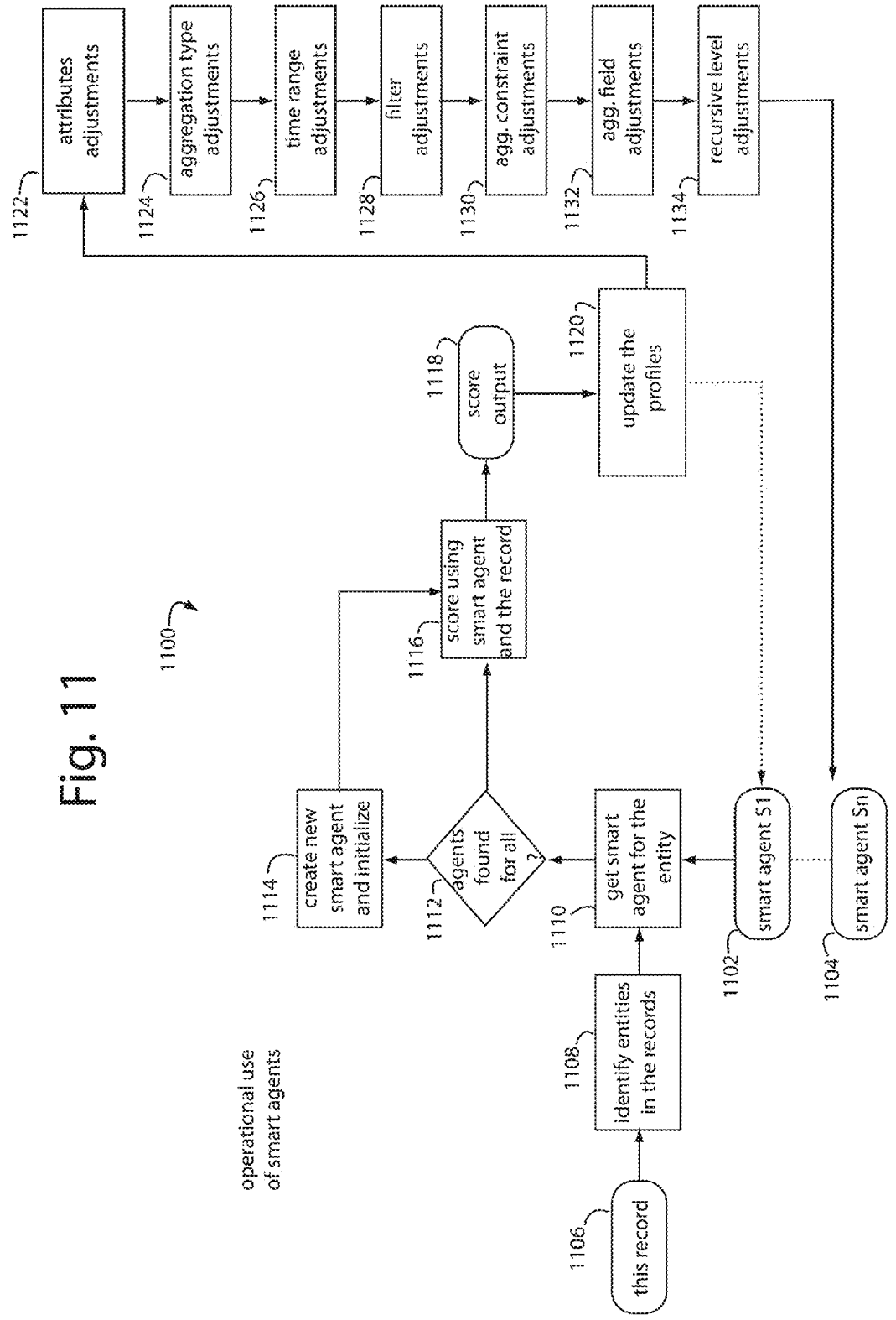

1000
transaction for $X preliminarily "declined"
1002
threshold-A
1006
this account's running average business totals
1008
this account's dollar activity exceeds threshold-A?
yes
no
1004
1012
"transaction declined"
1010
threshold-B transaction amount
1016
transaction amount $X exceeds threshold-B?
yes
no
1014
1020
"transaction approved"
1018
this account's transaction patterns
1024
this transaction type familiar?
yes
no
1022
1028
"transaction declined"
1026
threshold-C transaction amount
1032
transaction amount $X exceeds threshold-C?
yes
no
330
1036
"transaction approved"
1034
this account's devices
1040
familar user devices in this transaction?
yes
no
1038
1044
"transaction declined"
1042
threshold-D transaction amount
1048
transaction amount $X exceeds threshold-D?
yes
no
1046
1052
"transaction approved"
1050
"transaction declined"

Fig. 11


1100
operational use of smart agents
this record
1106
identify entities in the records
1108
get smart agent for the entity
1110
agents found for all ?
1112
create new smart agent and initialize
1114
score using smart agent and the record
1116
score output
1118
update the profiles
1120
smart agent S1
1102
smart agent Sn
1104
attributes adjustments
1122
aggregation type adjustments
1124
time range adjustments
1126
filter adjustments
1128
agg. constraint adjustments
1130
agg. field adjustments
1132
recursive level adjustments
1134

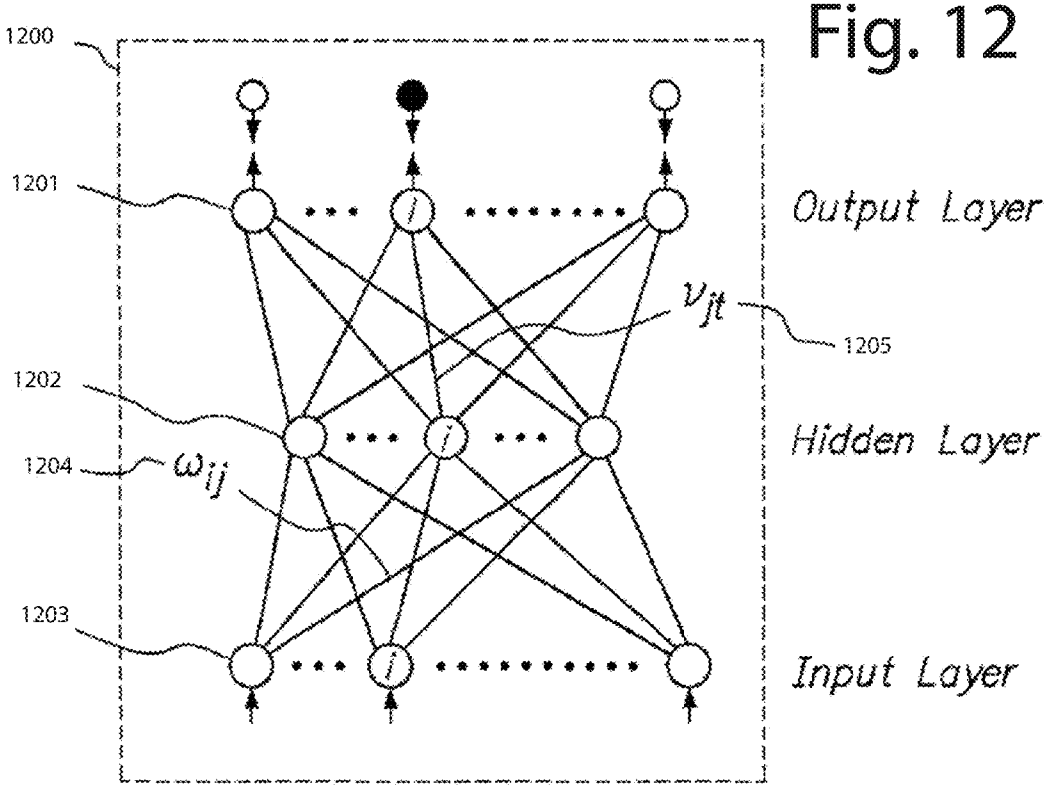

1200
1201
Output Layer
v_{jt}
1205
1202
Hidden Layer
1204
ω_{ij}
1203
Input Layer

Fig. 12

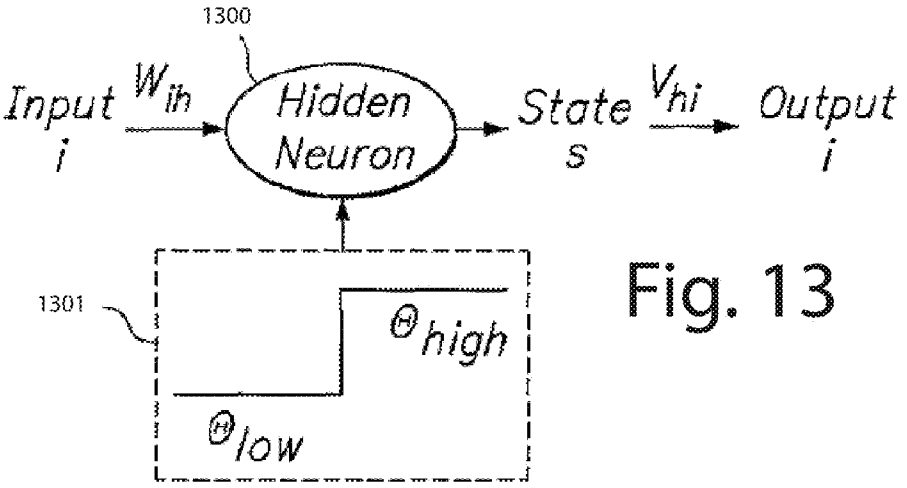

1300
Input i
W_{ih}
Hidden Neuron
State s
V_{hi}
Output i
1301
Θ_{high}
Θ_{low}

Fig. 13

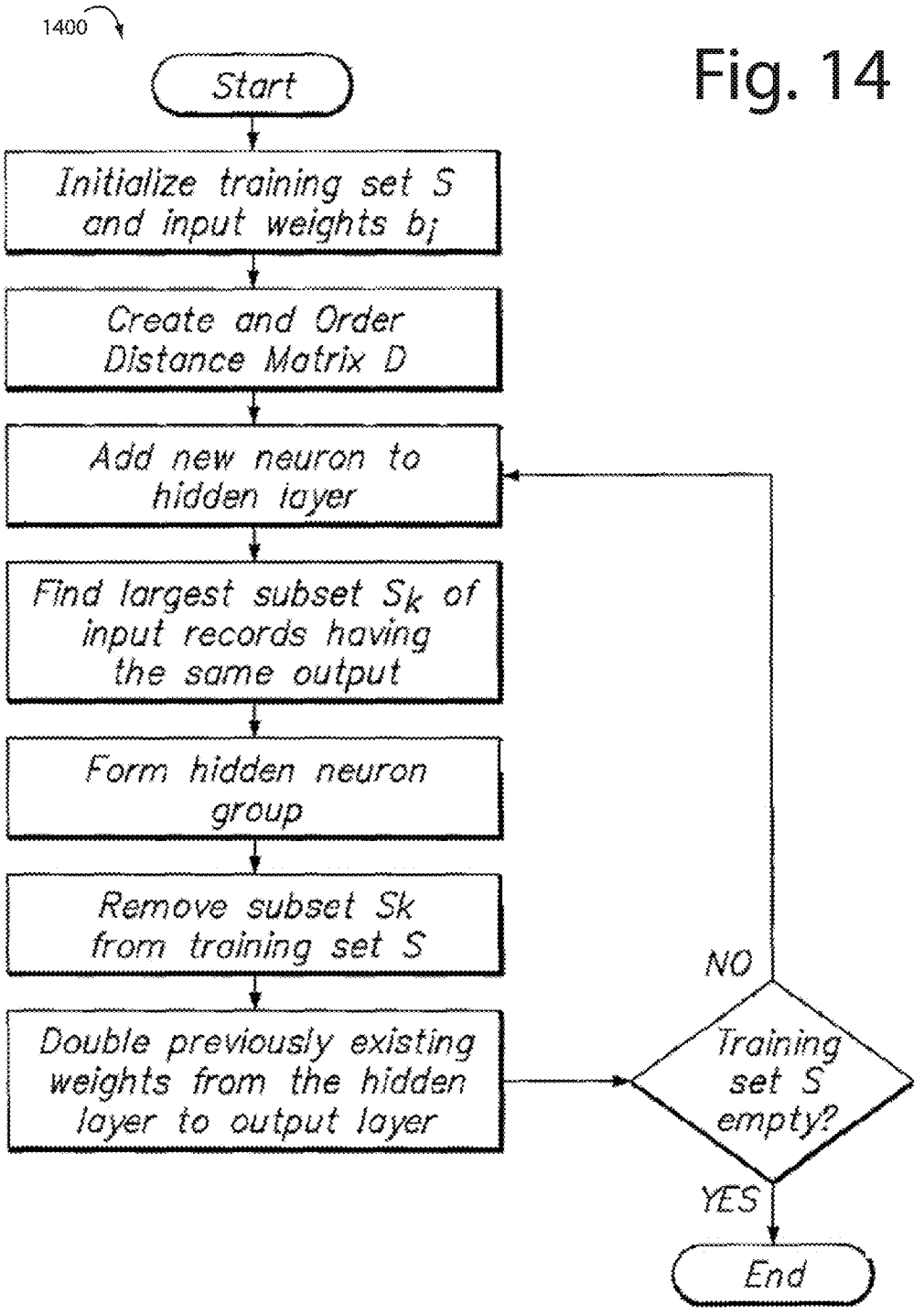

1400
Start
Initialize training set S and input weights b_j
Create and Order Distance Matrix D
Add new neuron to hidden layer
Find largest subset S_k of input records having the same output
Form hidden neuron group
Remove subset Sk from training set S
Double previously existing weights from the hidden layer to output layer
Training set S empty?
NO
YES
End

Fig. 14

Fig. 15

1500

Euclidean	$d(X_i,X_j)=\sqrt{\sum_{k=1}^{M}(X_{i,k}-X_{j,k})^2}$
Manhattan	$d(X_i,X_j)=\sum_{k=1}^{M}\lvert X_{i,k}-X_{j,k}\rvert$
Normalized Euclidean	$d(X_i,X_j)=\sqrt{\frac{1}{M}\sum_{k=1}^{M}\left(\frac{X_{i,k}-X_{j,k}}{max_k-min_k}\right)^2}$
Normalized Manhattan	$d(X_i,X_j)=\frac{1}{M}\sum_{k=1}^{M}\left\lvert\frac{X_{i,k}-X_{j,k}}{max_k-min_k}\right\rvert$
Weighted-Euclidean	$d(X_i,X_j)=\sqrt{\sum_{k=1}^{M}b_i*(X_{i,k}-X_{j,k})^2}$

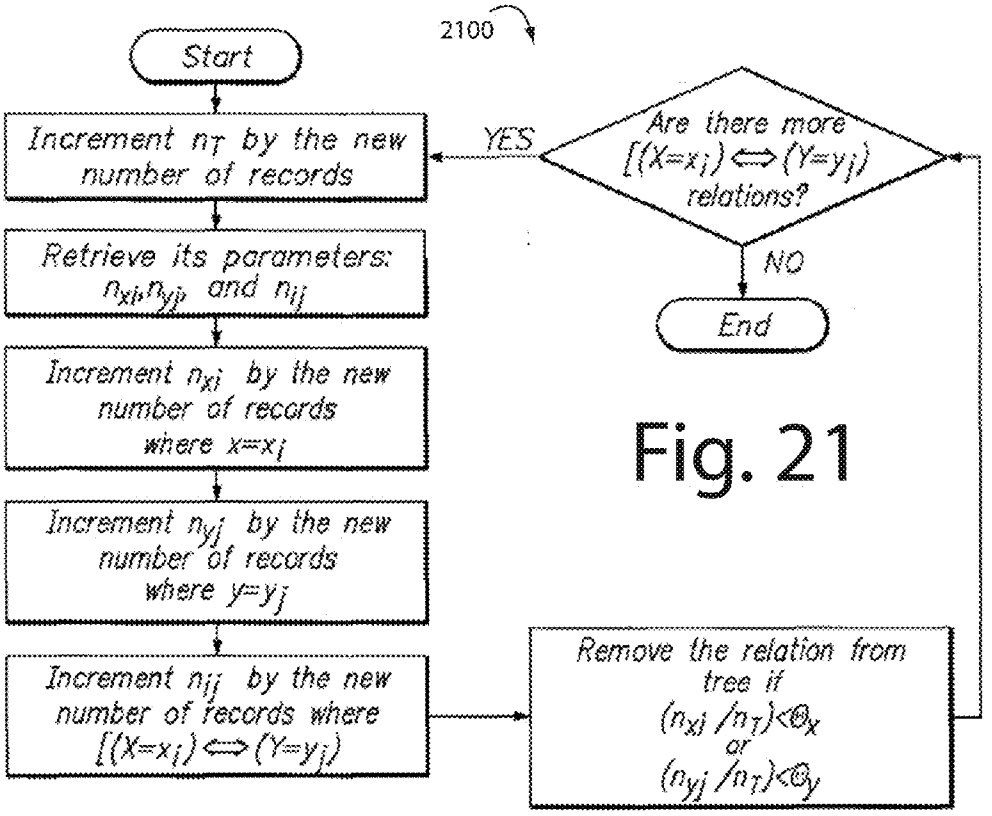
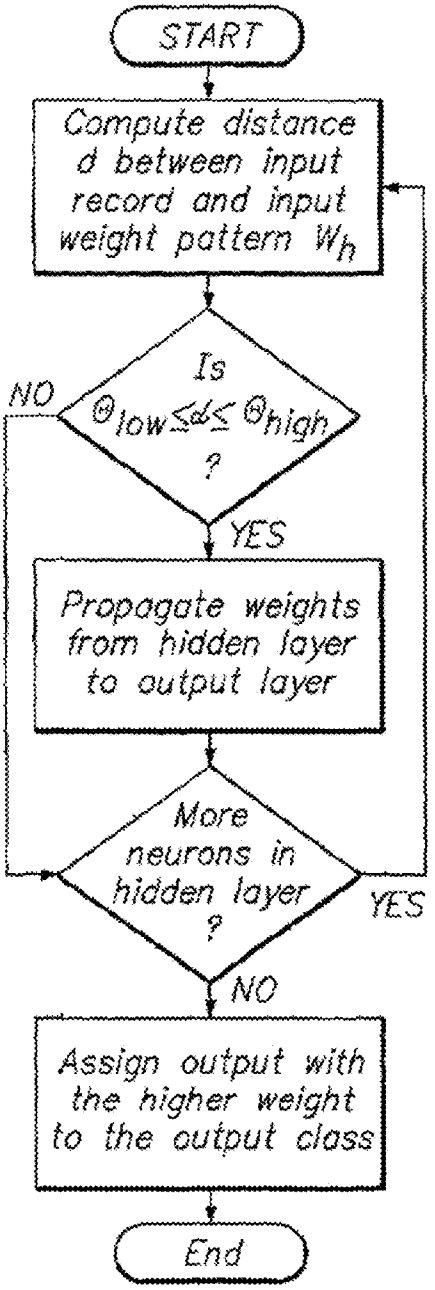


Fig. 21

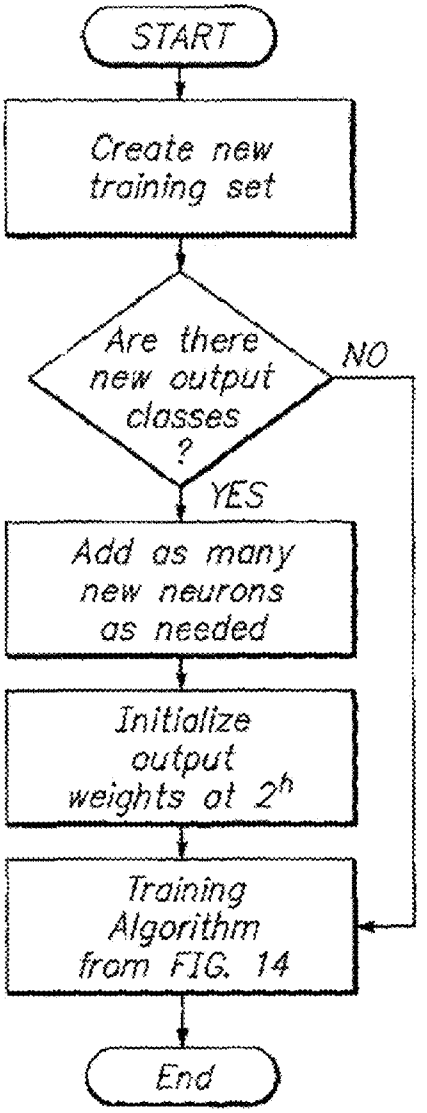
1600

Fig. 16



1700

Fig. 17



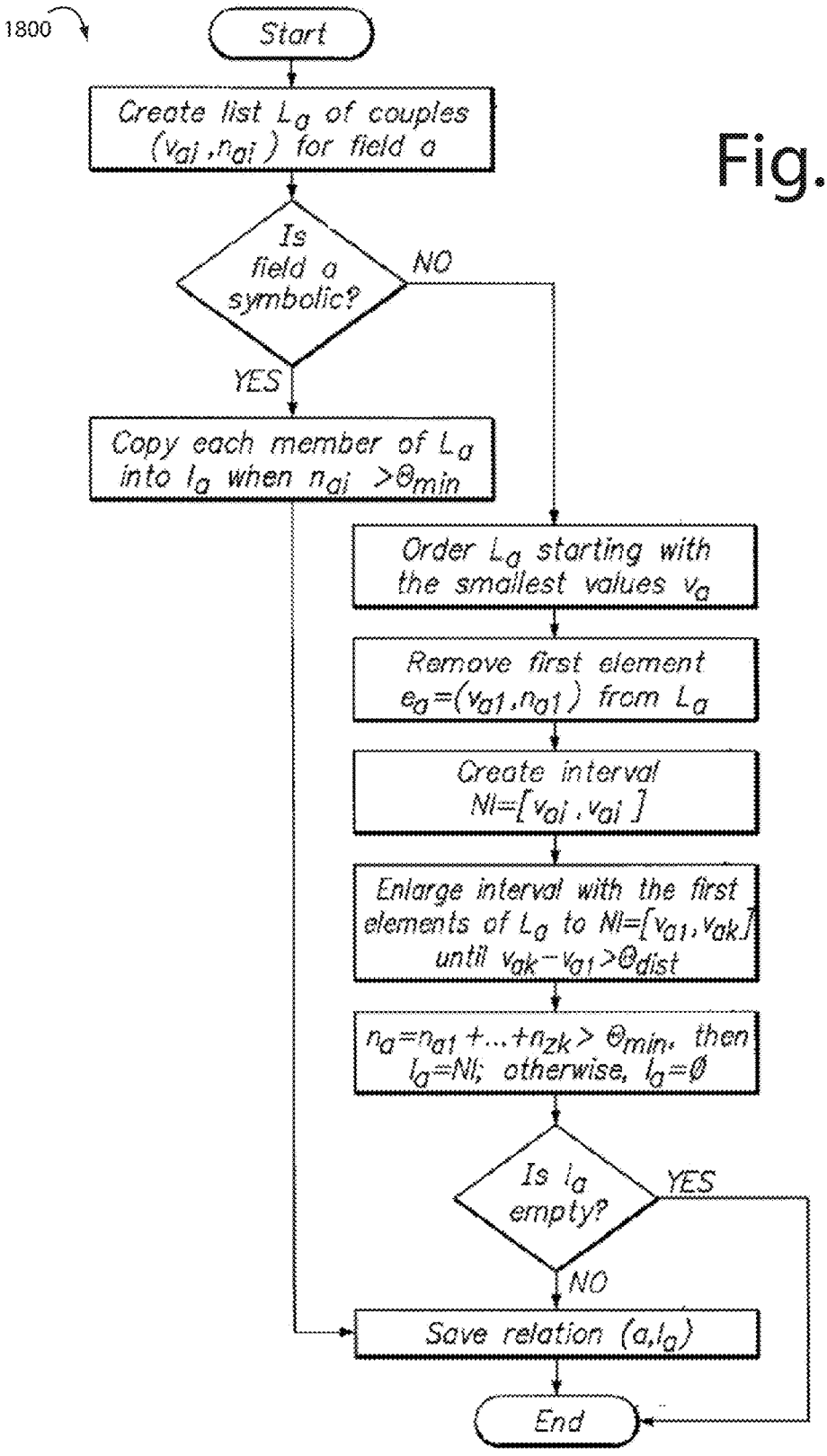
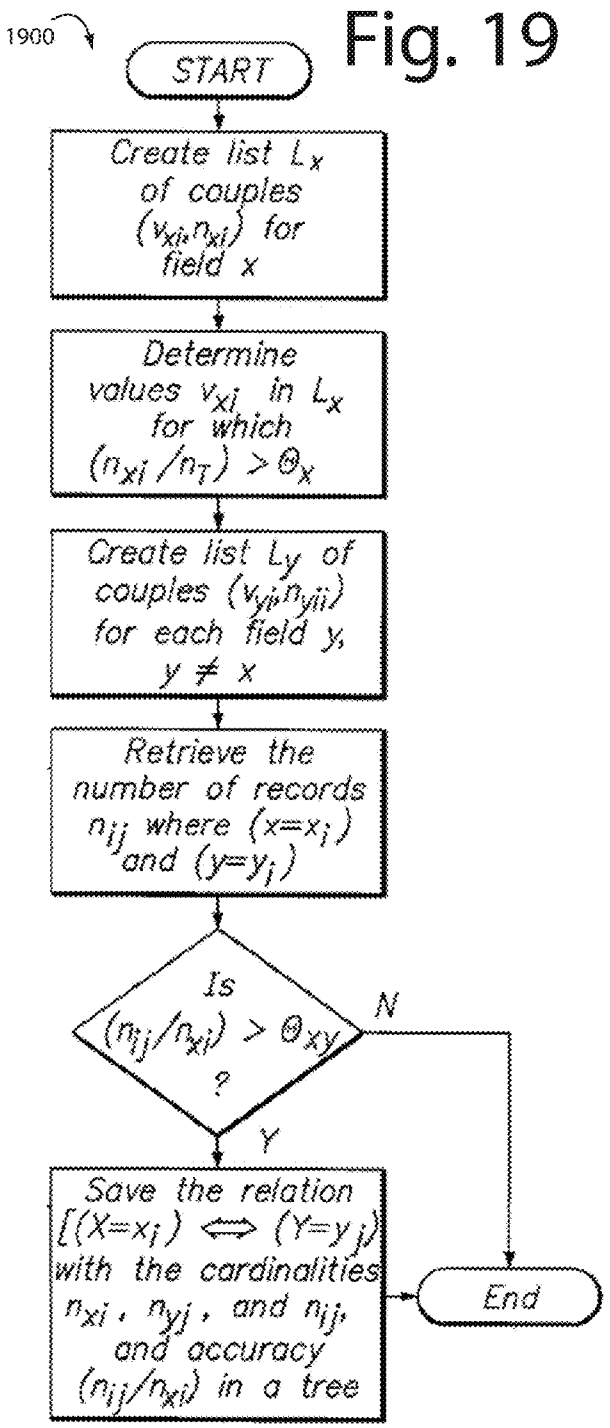

1800
Start
Create list L_a of couples (v_{aj}, n_{aj}) for field a
Is field a symbolic?
NO
YES
Copy each member of L_a into l_a when $n_{aj} > \Theta_{min}$
Order L_a starting with the smallest values v_a
Remove first element $e_a=(v_{a1}, n_{a1})$ from L_a
Create interval $NI=[v_{a1}, v_{a1}]$
Enlarge interval with the first elements of L_a to $NI=[v_{a1}, v_{ak}]$ until $v_{ak}-v_{a1}>\Theta_{dist}$
$n_a=n_{a1}+\ldots+n_{zk}>\Theta_{min}$, then $l_a=NI$; otherwise, $l_a=\emptyset$
Is l_a empty?
YES
NO
Save relation (a, l_a)
End

Fig. 18

Fig. 19


1900
START
Create list L_x of couples (v_{xi}, n_{xi}) for field x
Determine values v_{xi} in L_x for which $(n_{xi}/n_T) > \Theta_x$
Create list L_y of couples (v_{yi}, n_{yii}) for each field y, $y \neq x$
Retrieve the number of records n_{ij} where $(x=x_i)$ and $(y=y_j)$
Is $(n_{ij}/n_{xi}) > \Theta_{xy}$?
N
Y
Save the relation $[(X=x_i) \Leftrightarrow (Y=y_j)]$ with the cardinalities n_{xi}, n_{yj}, and n_{ij}, and accuracy (n_{ij}/n_{xi}) in a tree
End

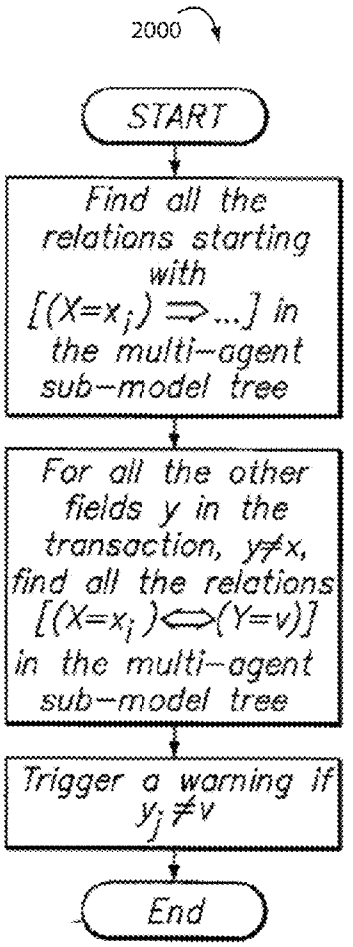

2000
START
Find all the relations starting with $[(X=x_i) \Rightarrow ...]$ in the multi-agent sub-model tree
For all the other fields y in the transaction, $y \neq x$, find all the relations $[(X=x_i) \Leftrightarrow (Y=v)]$ in the multi-agent sub-model tree
Trigger a warning if $y_j \neq v$
End

Fig. 20

Fig. 22

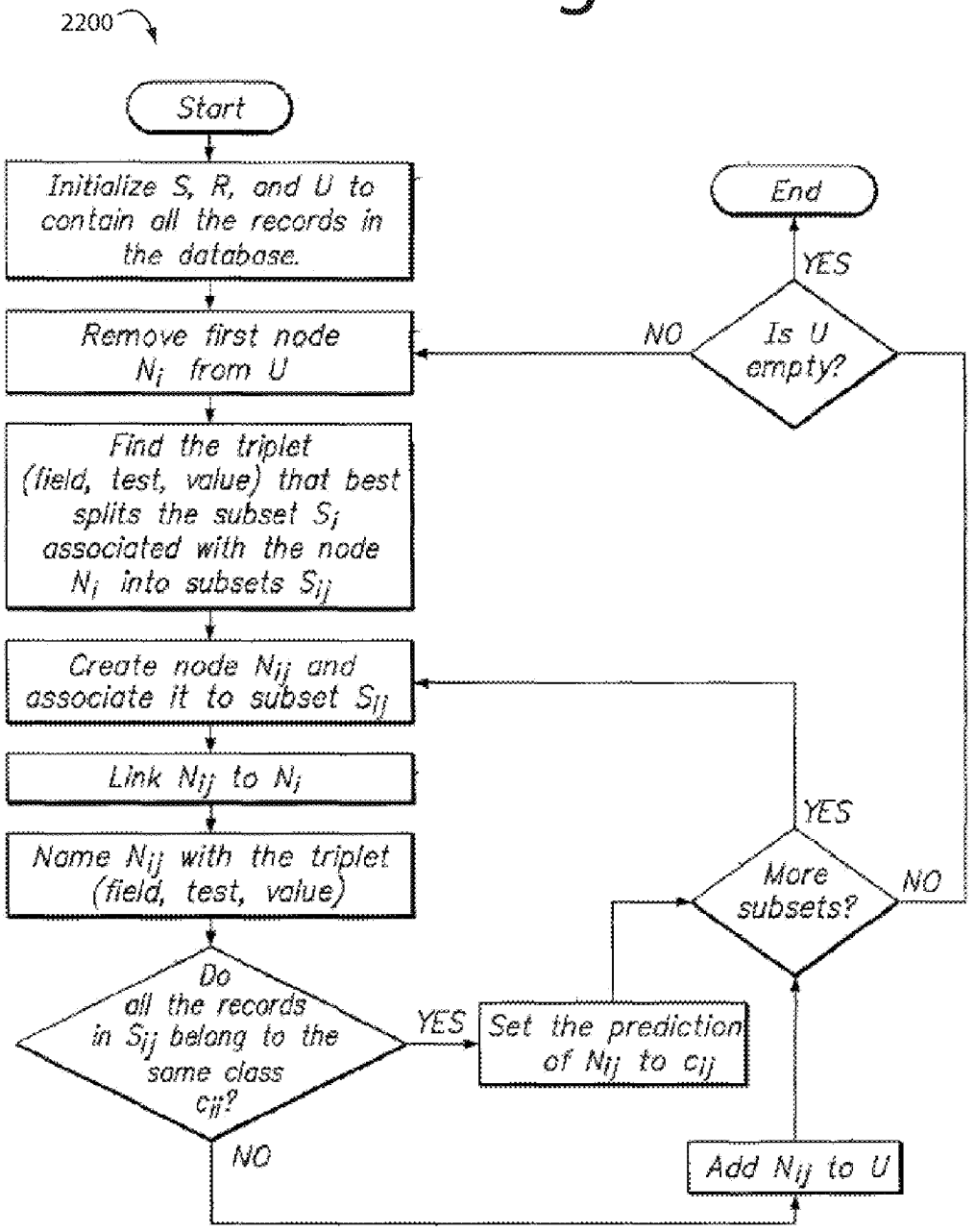

2200
Start
Initialize S, R, and U to contain all the records in the database.
Remove first node N_i from U
Find the triplet (field, test, value) that best splits the subset S_i associated with the node N_i into subsets S_{ij}
Create node N_{ij} and associate it to subset S_{ij}
Link N_{ij} to N_i
Name N_{ij} with the triplet (field, test, value)
Do all the records in S_{ij} belong to the same class c_{ij}?
YES
Set the prediction of N_{ij} to c_{ij}
NO
Add N_{ij} to U
More subsets?
YES
NO
Is U empty?
NO
YES
End

Fig. 23

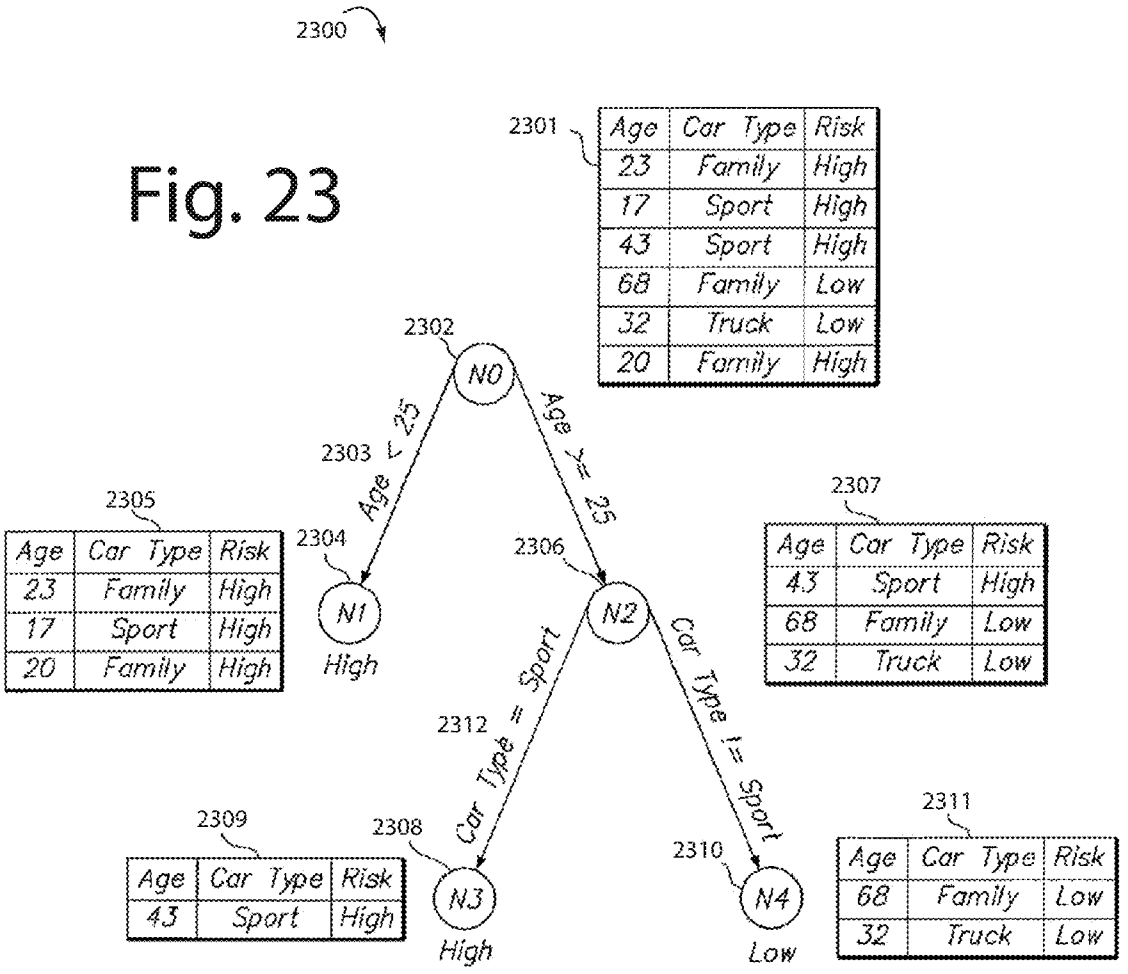

2300
2301
Age | Car Type | Risk
23 | Family | High
17 | Sport | High
43 | Sport | High
68 | Family | Low
32 | Truck | Low
20 | Family | High
2302
N0
2303
Age < 25
Age >= 25
2305
Age | Car Type | Risk
23 | Family | High
17 | Sport | High
20 | Family | High
2304
N1
High
2306
N2
2307
Age | Car Type | Risk
43 | Sport | High
68 | Family | Low
32 | Truck | Low
2312
Car Type = Sport
Car Type != Sport
2309
Age | Car Type | Risk
43 | Sport | High
2308
N3
High
2310
N4
Low
2311
Age | Car Type | Risk
68 | Family | Low
32 | Truck | Low

Fig. 24

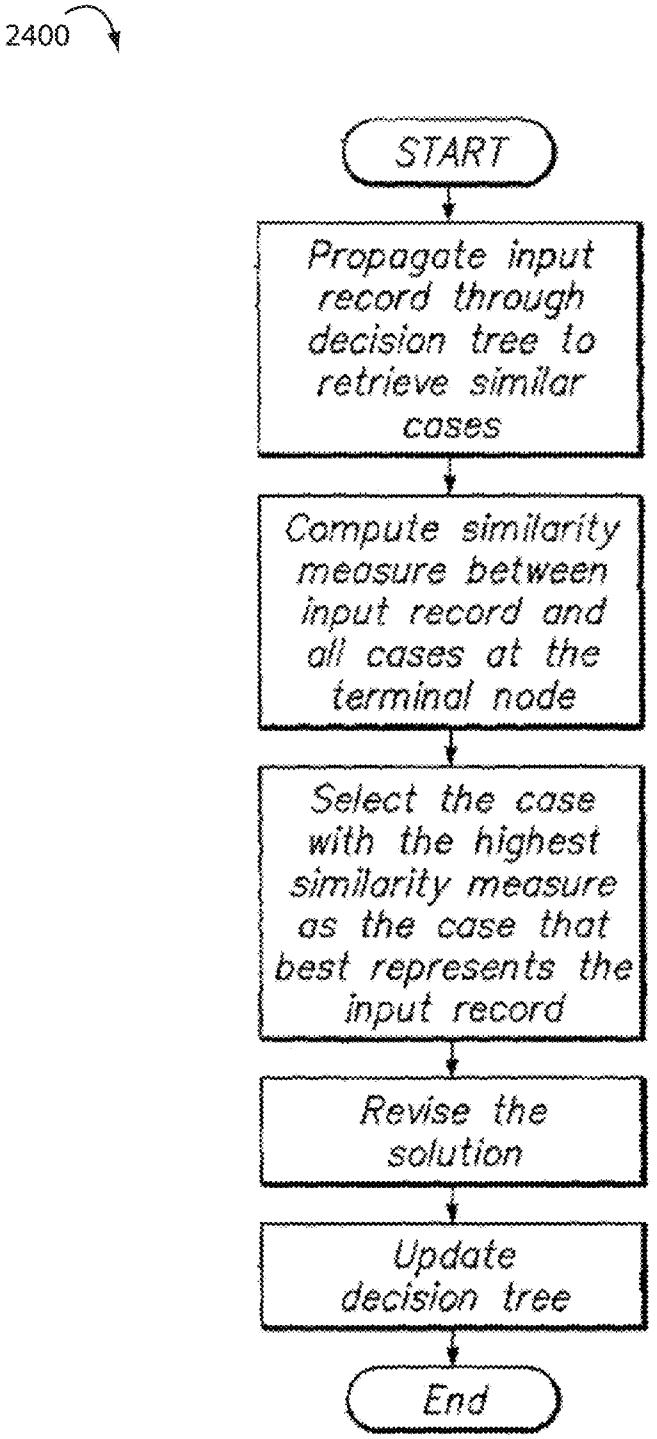

2400
START
Propagate input record through decision tree to retrieve similar cases
Compute similarity measure between input record and all cases at the terminal node
Select the case with the highest similarity measure as the case that best represents the input record
Revise the solution
Update decision tree
End

Fig. 25

2500

Global Similarity Measure	Expression
City–block	$\frac{1}{\rho}\sum_{i=4}^{\rho} sim_i(V_{1i}, V_{2i})$
Weighted city–block	$\frac{1}{\rho}\sum_{i=4}^{\rho} w_i * sim_i(V_{1i}, V_{2i})$
Euclidean	$\frac{1}{\rho}\sqrt{\sum_{i=4}^{\rho} sim_i(V_{1i}, V_{2i})^2}$
Minkowski	$\frac{1}{\rho}\sqrt[r]{\sum_{i=1}^{\rho} sim_i(V_{1i}, V_{2i})^r}$
Weighted Minkowski	$\sqrt[r]{\sum_{i=1}^{\rho} w_i * sim_i(V_{1i}, V_{2i})^r}$
Weighted maximum	$max_i \; w_i * sim_i(V_{1i}, V_{2i})$

Fig. 26

2600

Local Similarity Measures	Field Type	Field Valuation
$\begin{cases} 0, & \text{if } V_1 \cap V_2 = \emptyset \\ 1, & \text{otherwise} \end{cases}$	Nominal	Single, multiple
$\frac{Card(V_1 \cup V_2) - Card(V_1 \cap V_2)}{Card(V_1 \cup V_2)}$	Nominal	Multiple
$\frac{Card(V_1 \cup V_2) - Card(V_1 \cap V_2)}{Min(V_1 \cup V_2)}$	Nominal	Multiple
$\frac{Card(V_1 \cup V_2) - Card(V_1 \cap V_2)}{Max(V_1 \cup V_2)}$	Nominal	Multiple
$\frac{Card(V_1 \cup V_2) - Card(V_1 \cap V_2)}{Card(0)}$	Nominal	Multiple
$\frac{ec(min(V_1^-, V_2^-), max(V_1^+, V_2^+)) - Card(V_1 \cap V_2)}{Card(0)}$	Ordinal, Numeric	Multiple
$\frac{\lvert V_1 - V_2 \rvert}{ec(0)}$	Numeric	Single
$\frac{\lvert V_{1c} - V_{2c} \rvert}{ec(0)}$	Numeric	Multiple
$\frac{ec(min(V_1^-, V_2^-), max(V_1^+, V_2^+)) - ec(V_1 \cap V_2)}{ec(0)}$	Numeric	Multiple
$\frac{ec(V_1 \cup V_2) - ec(V_1 \cap V_2)}{ec(V_1 \cup V_2)}$	Numeric	Multiple
$\frac{ec(V_1 \cup V_2) - ec(V_1 \cap V_2)}{min(ecV_1, ecV_2)}$	Numeric	Multiple
$\frac{ec(V_1 \cup V_2) - ec(V_1 \cap V_2)}{max(ecV_1, ecV_2)}$	Numeric	Multiple
$\frac{2 * h(V_1 \cup V_2) - h(V_1) - h(V_2)}{2 * h_{max}}$	Taxonomic	Multiple
$\frac{h(\text{node that unit } V_1 \text{ \& } V_2)}{\text{total height of } h}$	Taxonomic	Single

Fig. 27

2700

IF	(certain specified condition occur in the data record)
THEN	(take the appropriate action)

2800

Fig. 28

$$tall(x) = \begin{cases} 0, & \text{if } height(x) < 5 \text{ ft.,} \\ (height(x) - 5 \text{ ft.})/2 \text{ ft.}, & \text{if } 5 \text{ ft.} <= height(x) <= 7 \text{ ft.,} \\ 1, & \text{if } height(x) > 7 \text{ ft.} \end{cases}$$

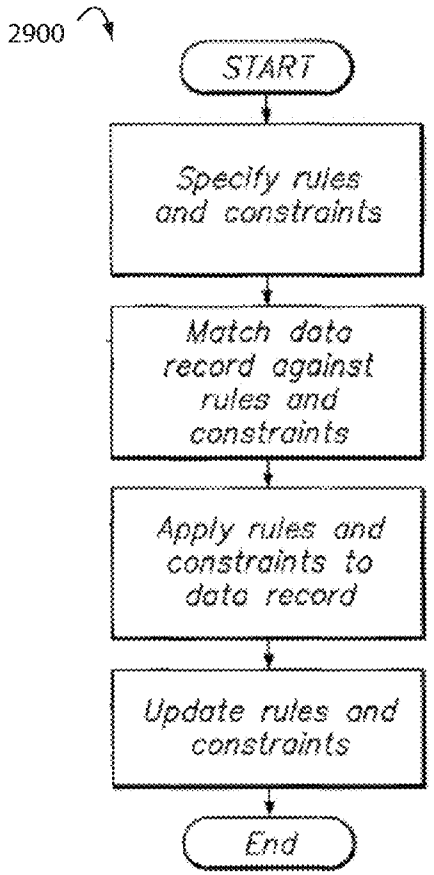
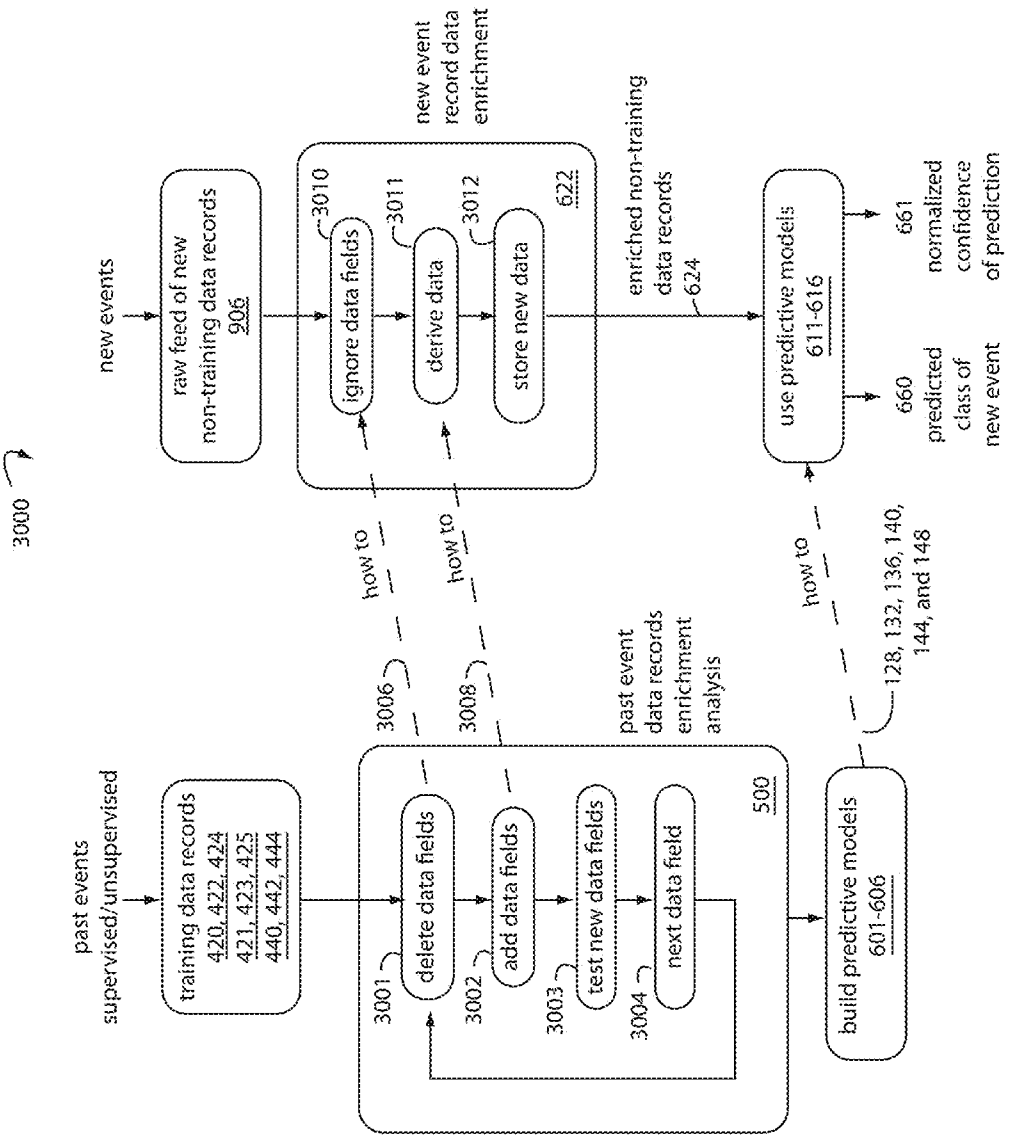


Fig. 29

Fig. 30


3000
past events
supervised/unsupervised
training data records
420, 422, 424
421, 423, 425
440, 442, 444
3001
delete data fields
3002
add data fields
3003
test new data fields
3004
next data field
500
past event
data records
enrichment
analysis
3006
how to
3008
how to
build predictive models
601-606
how to
128, 132, 136, 140,
144, and 148
new events
raw feed of new
non-training data records
906
3010
ignore data fields
3011
derive data
3012
store new data
622
new event
record data
enrichment
enriched non-training
data records
624
use predictive models
611-616
660
predicted
class of
new event
661
normalized
confidence
of prediction

METHOD OF OPERATING ARTIFICIAL INTELLIGENCE MACHINES TO IMPROVE PREDICTIVE MODEL TRAINING AND PERFORMANCE

BACKGROUND OF THE INVENTION

[0001] 1. Field of the Invention

[0002] The present invention relates to ARTIFICIAL INTELLIGENCE MACHINES and more specifically to methods of improving the training and performance of predictive models these include by enriching the data records to produce better decisions.

[0003] 2. Background

[0004] Machine learning can use various techniques such as supervised learning, unsupervised learning and Reinforcement learning. In supervised learning the learner is supplied with labeled training instances (set of examples), where both the input and the correct output are given. For example, historical stock prices are used to guesses future prices. Each example used for training is labeled with the value of interest—in this case the stock price. A supervised learning algorithm learns from the labeled values using information such as the day of the week, the season, the company's financial data, the industry, etc. After the algorithm has found the best pattern it can, it uses that pattern to make predictions.

[0005] In unsupervised learning, data points have no labels associated with them. Instead, the goal of unsupervised learning is to identify and explore regularities and dependencies in data, e.g., the structure of the underlying data distributions. The quality of a structure is measured by a cost function which is usually minimized to infer optimal parameters characterizing the hidden structure in the data. Reliable and robust inference requires a guarantee that the extracted structures are typical for the data source, e.g., similar structures have to be extracted from a second sample set of the same data source.

[0006] Reinforcement learning maps situations to actions to maximize a scalar reward or reinforcement signal. The learner does not need to be directly told which actions to take, but instead must discover which actions yield the best rewards by trial and error. An action may affect not only the immediate reward, but also the next situation, and consequently all subsequent rewards. Trial and error search, and delayed reward, are two important distinguishing characteristics of reinforcement learning.

[0007] Supervised learning algorithms use a known dataset to thereafter make predictions. The dataset training includes input data that produces response values. Supervised learning algorithms are used to build predictive models for new responses to new data. The larger the training datasets, the better will be the prediction models. Supervised learning includes classifications in which the data must be separated into classes, and regression for continuous-response. Common classification algorithms include support vector machines (SVM), neural networks, Naïve Bayes classifier and decision trees. Common regression algorithms include linear regression, nonlinear regression, generalized linear models, decision trees, and neural networks.

SUMMARY OF THE INVENTION

[0008] Briefly, method embodiments of the present invention improve the training and performance of predictive models included in artificial intelligence machines. A first method of operating an artificial intelligence machine produces predictive model language documents describing improved predictive models that generate better business decisions from raw data record inputs. A second method of operating an artificial intelligence machine including processors for predictive model algorithms produces and outputs better business decisions from raw data record inputs. Both methods enrich the raw data records their processors are fed by deleting data fields with data values that have little benefit in decision making, and that derive and add new data fields from information sources then available that do benefit in the decision making of the artificial intelligence machine through improved accuracies of prediction.

[0009] The above and still further objects, features, and advantages of the present invention will become apparent upon consideration of the following detailed description of specific embodiments thereof, especially when taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] FIG. **1** is a flowchart of a method embodiment of the present invention that provides user-service consumers with data science as-a-service operating on artificial intelligence machines;

[0011] FIG. **2** is a flowchart diagram of an algorithm for triple data encryption standard encryption and decryption as used in the method of FIG. **1**;

[0012] FIG. **3**A is a flowchart diagram of an algorithm for data cleanup as used in the method of FIG. **1**;

[0013] FIG. **3**B is a flowchart diagram of an algorithm for replacing a numeric value as used in the method of FIG. **3**A;

[0014] FIG. **3**C is a flowchart diagram of an algorithm for replacing a symbolic value as used in the method of FIG. **3**A;

[0015] FIG. **4** is a flowchart diagram of an algorithm for building training sets, test sets, and blind sets, and further for down sampling if needed and as used in the method of FIG. **1**;

[0016] FIG. **5**A is a flowchart diagram of an algorithm for a first part of the data enrichment as used in the method of FIG. **1**;

[0017] FIG. **5**B is a flowchart diagram of an algorithm for a second part of the data enrichment as used in the method of FIG. **1** and where more derived fields are needed to suit quality targets;

[0018] FIG. **6** is a flowchart diagram of a method of using the PMML Documents of FIG. **1** with an algorithm for the run-time operation of parallel predictive model technologies in artificial intelligence machines;

[0019] FIG. **7** is a flowchart diagram of an algorithm for the decision engine of FIG. **6**;

[0020] FIG. **8** is a flowchart diagram of an algorithm for using ordered rules and thresholds to decide amongst prediction classes;

[0021] FIG. **9** is a flowchart diagram of a method that combines the methods of FIGS. **1-8** and their algorithms to artificial intelligence machines that provide an on-line service for scoring, predictions, and decisions to user-service consumers requiring data science and artificial intelligence services without their being required to invest in and maintain specialized equipment and software;

[0022] FIG. **10** is a flowchart diagram illustrating an artificial intelligence machine apparatus for executing an algorithm for reconsideration of an otherwise final adverse decision, for example, in a payment authorization system a

transaction request for a particular amount $X has already been preliminarily "declined" according to some other decision model;

[0023] FIG. **11** is a flowchart diagram of an algorithm for the operational use of smart agents in artificial intelligence machines;

[0024] FIGS. **12-29** provide greater detail regarding the construction and functioning of algorithms that are employed in FIGS. **1-11**;

[0025] FIG. **12** is a schematic diagram of a neural network architecture used in a model;

[0026] FIG. **13** is a diagram of a single neuron in a neural network used in a model;

[0027] FIG. **14** is a flowchart of an algorithm for training a neural network;

[0028] FIG. **15** is an example illustrating a table of distance measures that is used in a neural network training process;

[0029] FIG. **16** is a flowchart of an algorithm for propagating an input record through a neural network;

[0030] FIG. **17** is a flowchart of an algorithm for updating a training process of a neural network;

[0031] FIG. **18** is a flowchart of an algorithm for creating intervals of normal values for a field in a training table;

[0032] FIG. **19** is a flowchart of an algorithm for determining dependencies between each field in a training table;

[0033] FIG. **20** is a flowchart of an algorithm for verifying dependencies between fields in an input record;

[0034] FIG. **21** is a flowchart of an algorithm for updating a smart-agent technology;

[0035] FIG. **22** is a flowchart of an algorithm for generating a data mining technology to create a decision tree based on similar records in a training table;

[0036] FIG. **23** is an example illustrating a decision tree for a database maintained by an insurance company to predict a risk of an insurance contract based on a type of a car and a age of its driver;

[0037] FIG. **24** is a flowchart of an algorithm for generating a case-based reasoning technology to find a case in a database that best resembles a new transaction;

[0038] FIG. **25** is an example illustrating a table of global similarity measures used by a case-based reasoning technology;

[0039] FIG. **26** is an example illustrating a table of local similarity measures used by a case-based reasoning technology;

[0040] FIG. **27** is an example illustrating a rule for use with a rule-based reasoning technology;

[0041] FIG. **28** is an example illustrating a fuzzy rule to specify if a person is tall;

[0042] FIG. **29** is a flowchart of an algorithm for applying rule-based reasoning, fuzzy logic, and constraint programming to assess the normality/abnormality of and classify a transaction assess an activity; and

[0043] FIG. **30** is a flowchart diagram of an algorithm executed by an apparatus needed to implement a method embodiment of the present invention for improving predictive model training and performance by data enrichment of transaction records.

DETAILED DESCRIPTION OF THE INVENTION

[0044] Computer-implemented method embodiments of the present invention provide an artificial intelligence and machine-learning service that is delivered on-demand to user-service consumers, their clients, and other users through network servers. The methods are typically implemented with special algorithms executed by computer apparatus and delivered to non-transitory storage mediums to the providers and user-service consumers who then sell or use the service themselves.

[0045] Users in occasional or even regular need of artificial intelligence and machine learning Prediction Technologies can get the essential data-science services required on the Cloud from an appropriate provider, instead of installing specialized hardware and maintaining their own software. Users are thereby freed from needing to operate and manage complex software and hardware. The intermediaries manage user access to their particular applications, including quality, security, availability, and performance.

[0046] FIG. **1** represents a predictive model learning method **100** that provides artificial intelligence and machine learning as-a-service by generating predictive models from service-consumer-supplied training data input records. A computer file **102** previously hashed or encrypted by a triple-DES algorithm, or similar protection. It also possible to send a non-encrypted filed through an encrypted channel. Users of the platform would upload their data through SSL/TLS from a browser or from a command line interface (SCP or SFTP). This is then received by a network server from a service consumer needing predictive models. Such encode the supervised and/or unsupervised data of the service consumer that are essential for use in later steps as training inputs. The records **102** received represent an encryption of individual supervised and/or unsupervised records each comprising a predefined plurality of predefined data fields that communicate data values, and structured and unstructured text. Such text often represents that found in webpages, blogs, automated news feeds, etc., and very often such contains errors and inconsistencies.

[0047] Structured text has an easily digested form and unstructured text does not. Text mining can use a simple bag-of-words model, such as how many times does each word occur. Or complex approaches that pull the context from language structures, e.g., the metadata of a post on Twitter where the unstructured data is the text of the post.

[0048] These records **102** are decrypted in a step **104** with an apparatus for executing a decoding algorithm, e.g., a standard triple-DES device that uses three keys. An example is illustrated in FIG. **2**. A series of results are transformed into a set of non-transitory, raw-data records **106** that are collectively stored in a machine-readable storage mechanism.

[0049] A step **108** cleans up and improves the integrity of the data stored in the raw-data records **106** with an apparatus for executing a data integrity analysis algorithm. An example is illustrated in FIGS. **3**A, **3**B, and **3**C. Step **108** compares and corrects any data values in each data field according to user-service consumer preferences like min, max, average, null, and default, and a predefined data dictionary of valid data values. Step **108** discerns the context of the structured and unstructured text with an apparatus for executing a contextual dictionary algorithm. Step **108** transforms each result into a set of flat-data records **110** that are collectively stored in a machine-readable storage mechanism.

[0050] Method **108** improves the training of predictive models by converting and transforming a variety of inconsistent and incoherent supervised and unsupervised training data for predictive models received by a network server as electronic data files, and storing that in a computer data storage mechanism. It then transforms these into another single, error-free, uniformly formatted record file in computer data

storage with an apparatus for executing a data integrity analysis algorithm that harmonizes a range of supervised and unsupervised training data into flat-data records in which every field of every record file is modified to be coherent and well-populated with information.

[0051] The data values in each data field in the inconsistent and incoherent supervised and unsupervised training data are compared and corrected according to a user-service consumer preference and a predefined data dictionary of valid data values. An apparatus for executing an algorithm substitutes data values in the data fields of incoming supervised and unsupervised training data with at least one value representing a minimum, a maximum, a null, an average, and a default.

[0052] The context of any text included in the inconsistent and incoherent supervised and unsupervised training data is discerned, recognized, detected, and discriminated with an apparatus for executing a contextual dictionary algorithm that employs a thesaurus of alternative contexts of ambiguous words for find a common context denominator, and to then record the context determined into the computer data storage mechanism for later access by a predictive model.

[0053] Further details regarding data clean-up are provided below in connection with FIGS. **3**A, **3**B, and **3**C. Data cleaning herein deals with detecting and removing errors and inconsistencies from data in order to improve the quality of data. Data quality problems are present in single data collections, such as files and databases, or multiple data sources. For example,

Single-Source Data

[0054]

	level	data errors
attribute	illegal values	birth date = 30.13.70
record	violated attribute dependencies	age = 32, birth date = 12.02.76
record type	uniqueness violation	name = "john smith", SSN = "123456"); name = "peter miller", SSN = "123456")
source	referential integrity violation	
attribute	missing values	phone = 9999-999999
	misspellings	city = "SO"
	abbreviations	Occupation = "database programmer."
	embedded values	name = "j. smith 12.02.70 new York"
	misfielded values	city = "USA"
record	violated attribute dependencies	city = "mill valley", zip = 765662
record type	word transpositions	name1 = "j. smith", name2 = "miller p."
	duplicated records	name = "john smith", . . .); name = "j. smith", . . .)
	contradicting records	name = "john smith", birth date = 12.02.76); name = "john smith", birth date = 12.12.76)
source	wrong references	employee = (name = "john smith", dept. no = 17)
problems	metadata	examples/heuristics
illegal values	cardinality	e.g., cardinality (gender) 2 indicates problem
	max, min	max, min should not be outside of permissible range
	variance, deviation	variance, deviation of statistical values should not be higher than threshold

-continued

	level	data errors
misspellings	attribute values	sorting on values often brings misspelled values next to correct values
missing values	null values	percentage/number of null values
	attribute values + default values	presence of default value may indicate real value is missing
varying value representations	attribute values	comparing attribute value set of a column of one table against that of a column of another table
duplicates	cardinality + uniqueness	attribute cardinality = # rows should hold
	attribute values	sorting values by number of occurrences; more than 1 occurrence indicates duplicates

[0055] In a step **112**, a test is made to see if a number of records **114** in the set of flat-data records **110** exceeds a predefined threshold, e.g., about one hundred million. The particular cutoff number to use is inexact and is empirically determined by what produces the best commercial efficiencies.

[0056] But if the number of records **114** is too large, a step **116** then samples a portion of the set of flat-data records **110**. An example is illustrated in FIG. **4**. Step **116** stores a set of samples **118** in a machine-readable storage mechanism for use in the remaining steps. Step **116** consequently employs an apparatus for executing a special sampling algorithm that limits the number of records that must be processed by the remaining steps, but at the same time preserves important training data. The details are described herein in connection with FIG. **4**.

[0057] A modeling data **120** is given a new, amplified texture by a step **122** for enhancing, enriching, and concentrating the sampled or unsampled data stored in the flat-data records with an apparatus for executing a data enrichment algorithm. An example apparatus is illustrated in FIG. **4**, which outputs training sets **420**, **421**, and **440**; and test sets **422**, **423**, and **442**; and blind sets **424**, **425**, and **444** derived from either the flat data **110** or sampled data **118**. Such step **122** removes data that may exist in particular data fields that is less important to building predictive models. Entire data fields themselves are removed here that are predetermined to be unavailing to building good predictive models that follow.

[0058] Step **122** calculates and combines any data it has into new data fields that are predetermined to be more important to building such predictive models. It converts text with an apparatus for executing a context mining algorithm, as suggested by FIG. **6**. Even more details of this are suggested in my U.S. patent application Ser. No. 14/613,383, filed Feb. 4, 2015, and titled, ARTIFICIAL INTELLIGENCE FOR CONTEXT CLASSIFIER. Step **122** then transforms a plurality of results from the execution of these algorithms into a set of enriched-data records **124** that are collectively stored in a machine-readable storage mechanism.

[0059] A step **126** uses the set of enriched-data records **124** to build a plurality of smart-agent predictive models for each entity represented. Step **126** employs an apparatus for executing a smart-agent building algorithm. The details of this are shown in FIG. **6**. Further related information is included in my U.S. Pat. No. 7,089,592 B2, issued Aug. 8, 2006, titled, SYSTEMS AND METHODS FOR DYNAMIC DETECTION AND PREVENTION OF ELECTRONIC FRAUD, which is incorporated herein by reference. (Herein, Adjaoute

'592.) Special attention should be placed on FIGS. **11-30** and the descriptions of smart-agents in connection with FIG. **21** and the smart-agent technology in Columns **16-18**.

Unsupervised Learning of Normal and Abnormal Behavior

[0060] Each field or attribute in a data record is represented by a corresponding smart-agent. Each smart-agent representing a field will build what-is-normal (normality) and what-is-abnormal (abnormality) metrics regarding other smart-agents.

[0061] Apparatus for creating smart-agents is supervised or unsupervised. When supervised, an expert provides information about each domain. Each numeric field is characterized by a list of intervals of normal values, and each symbolic field is characterized by a list of normal values. It is possible for a field to have only one interval. If there are no intervals for an attribute, the system apparatus can skip testing the validity of its values, e.g., when an event occurs.

[0062] As an example, a doctor (expert) can give the temperature of the human body as within an interval [35° C.: 41° C.], and the hair colors can be {black, blond, red}.

1) For each field "a" of a Table:
   i) Retrieve all the distinct values and their cardinalities and create a list "La" of couples (vai, nai);
   ii) Analyze the intermediate list "La" to create the list of intervals of normal values Ia with this method:
      (a) If "a" is a symbolic attribute, copy each member of "La" into Ia when nai is superior to a threshold Θ_{min};
      (b) If "a" is a numeric attribute:
         1. Order the list "La" starting with the smallest values "a";
         2. While La is not empty;
            i. Remove the first element ea= (val, na1) of "La"
            ii. Create an interval with this element: I' = [val, val]
            iii. While it is possible, enlarge this interval with the first elements of "La" and remove them from "La": I' = [val, vak]. The loop stops before the size of the interval vak-val becomes greater than a threshold Θ_{dist}.
      (c) given: na' = na1 + ... + nak
      (d) If na' is superior to a threshold Θmin, Ia = I' otherwise, ia = Ø;
   iii) If Ia is not empty, save the relation (a , Ia).

[0063] An unsupervised learning process uses the following algorithm:

[0064] Θ_{man} represents the minimum number of elements an interval must include. This means that an interval will only be take into account if it encapsulates enough values, so its values will be considered normal because frequent;
the system apparatus defines two parameters that is modified:

[0065] the maximum number of intervals for each attribute n_{max};

[0066] the minimum frequency of values in each interval f_{Imin};
Θ_{min} is computed with the following method:

$$\Theta_{min}=f_{Imin}*\text{number of records in the table.}$$

[0067] Θ_{dist} represents the maximum width of an interval. This prevents the system apparatus from regrouping some numeric values that are too disparate. For an attribute a, lets call mina the smallest value of a on the whole table and maxa the biggest one. Then:

$$\Theta_{dist}=(\max a-\min a)/n_{max}$$

For example, consider a numeric attribute of temperature with the following values:

75	80	85	72	69	72	83	64	81	71	65	75	68	70

The first step is to sort and group the values into "La": "La"={(64, 1) (65, 1) (68, 1) (69, 1) (70, 1) (71, 1) (72, 2) (75, 2) (80, 1) (81, 1) (83, 1) (85, 1)}
Then the system apparatus creates the intervals of normal values:

[0068] Consider f_{Imin}=10% and n_{max}=5 then Θ_{min}=1.4 and Θ_{dist}=(85−64)/5=4.2

[0069] Ia={[64, 68] [69, 72] [75] [80, 83]}
The interval [85, 85] was removed because its cardinality (1) is smaller than Θ_{min}.

[0070] When a new event occurs, the values of each field are verified with the intervals of the normal values it created, or that were fixed by an expert. It checks that at least one interval exists. If not, the field is not verified. If true, the value inside is tested against the intervals, otherwise a warning is generated for the field.

[0071] During creation, dependencies between two fields are expressed as follows:

[0072] When the field 1 is equal to the value v1, then the field 2 takes the value v2 in significant frequency p.
Example: when species is human the body_temperature is 37.2° C. with a 99.5% accuracy.

[0073] Given cT is the number of records in the whole database. For each attribute X in the table:
Retrieve the list of distinct values for X with the cardinality of each value:

[0074] Lx={(x1, cx1), . . . (xi, c_{xi}), . . . (xn, cxn)}
For each distinct value xi in the list:
Verify if the value is typical enough: (c_{xi}/cT)>Θx?

[0075] If true, for each attribute Y in the table, Y≠X
Retrieve the list of distinct values for Y with the cardinality of each value:

[0076] Ly={(y1, cy1), . . . (yj, c_{yj}), . . . (yn, cyn)}
For each value yj;

[0077] Retrieve the number of records c_{ij} where (X=xi) and (Y=yj). If the relation is significant, save it: if (c_{ij}/c_{xi})>Θxy then save the relation [(X=xi)⇒ (Y=yj)] with the cardinalities c_{xi}, c_{yj} and c_{ij}.

[0078] The accuracy of this relation is given by the quotient (c_{ij}/c_{xi})

[0079] Verify the coherence of all the relations: for each relation [(X=xi)⇒ (Y=yj)] (1)

[0080] Search if there is a relation [(Y=yj)⇒ (X=xk)] (2)

[0081] If xi≠xk remove both relations (1) and (2) from the model otherwise it will trigger a warning at each event since (1) and (2) cannot both be true.

[0082] To find all the dependencies, the system apparatus analyses a database with the following algorithm:

[0083] The default value for Θx is 1%: the system apparatus will only consider the significant value of each attribute.

[0084] The default value for Θxy is 85%: the system apparatus will only consider the significant relations found.

[0085] A relation is defined by: (Att_1=v_1)⇒ (Att_2=v_2) (eq).

[0086] All the relations are stored in a tree made with four levels of hash tables, e.g., to increase the speed of the system apparatus. A first level is a hash of the attribute's name (Att1 in eq); a second level is a hash for each attribute the values that imply some correlations (v1 in eq); a third level is a hash of

the names of the attributes with correlations (Att2 in eq) to the first attribute; a fourth and last level has values of the second attribute that are correlated (v2 in eq).

Each leaf represents a relation. At each leaf, the system apparatus stores the cardinalities c_{xi}, c_{yj} and c_{ij}. This will allow the system apparatus to incrementally update the relations during its lifetime. Also it gives:

[0087] the accuracy of a relation: c_{ij}/c_{xi};

[0088] the prevalence of a relation: c_{ij}/cT;

[0089] the expected predictability of a relation: c_{yj}/cT.

[0090] Consider an example with two attributes, A and B:

A	B
1	4
1	4
1	4
1	3
2	1
2	1
2	2
3	2
3	2
3	2

There are ten records: cT=10.

Consider all the possible relations:

Relation	C_{xi}	C_{yi}	C_{ij}	(c_{xi}/c_T)	Accuracy	
(A = 1) ⇒ (B = 4)	4	3	3	40%	75%	(1)
(A = 2) ⇒ (B = 1)	2	2	2	20%	100%	(2)
(A = 3) ⇒ (B = 2)	3	4	3	30%	100%	(3)
(B = 4) ⇒ (A = 1)	3	4	3	30%	100%	(4)
(B = 3) ⇒ (A = 1)	1	4	1	10%	100%	(5)
(B = 1) ⇒ (A = 2)	2	3	2	20%	100%	(6)
(B = 2) ⇒ (A = 3)	4	3	3	40%	75%	(7)

With the defaults values for Θx and Θxy, for each possible relation, the first test $(c_{xi}/c_T)>\Theta x$ is successful (since Θx=1%) but the relations (1) and (7) would be rejected (since Θxy=85%).

Then the system apparatus verifies the coherence of each remaining relation with an algorithm:

(A = 2) ⇒ (B = 1) is coherent with (B = 1) ⇒ (A = 2);
(A = 3) ⇒ (B = 2) is not coherent since there is no more relation
(B = 2) ⇒ . . . ;
(B = 4) ⇒ (A = 1) is not coherent since there is no more relation
(A = 1) ⇒ . . . ;
(B = 3) ⇒ (A = 1) is not coherent since there is no more relation
(A = 1) ⇒ . . . ;
(B = 1) ⇒ (A = 2) is coherent with (A = 2) ⇒ (B = 1).

The system apparatus classifies the normality/abnormality of each new event in real-time during live production and detection.

[0091] For each event couple attribute/value (X, xi):

Looking in the model for all the relations starting by [(X=xi) ⇒ . . .]

[0092] For all the other couple attribute/value (Y, Y_j), Y≠X, of the event:

[0093] Look in the model for a relation [(X=x_i) ⇒ (Y=v)];

[0094] If $y_j \neq v$ then trigger a warning "[(X=x_i) ⇒ (Y=y_j)] not respected".

Incremental Learning

[0095] The system apparatus incrementally learns with new events:

Increment cT by the number or records in the new table T.

For each relation [(X=xi)⇒ (Y=yj)] previously created:

[0096] Retrieve its parameters: c_{xi}, c_{yj} and c_{ij}

[0097] Increment c_{xi} by the number of records in T where X=x_i;

[0098] Increment c_{yj} by the number of records in T where Y=y_j;

[0099] Increment c_{ij} by the number of records in T where [(X=x_i)⇒ (Y=Y_j)];

[0100] Verify if the relation is still significant:

[0101] If $(c_{xi}/c_T)<\theta_x$, remove this relation;

If $(c_{ij}/c_{xi})<\Theta_{xy}$, remove this relation.

[0102] In FIG. **1**, a step **127** selects amongst a plurality of smart-agent predictive models and updates a corresponding particular smart-agent's real-time profile and long-term profile. Such profiles are stored in a machine-readable storage mechanism with the data from the enriched-data records **124**. Each corresponds to a transaction activity of a particular entity. Step **127** employs an apparatus for executing a smart-agent algorithm that compares a current transaction, activity, behavior to previously memorialized transactions, activities and profiles such as illustrated in FIG. **7**. Step **127** then transforms and stores a series of results as smart-agent predictive model in a markup language document in a machine-readable storage mechanism. Such smart-agent predictive model markup language documents are XML types and best communicated in a registered file extension format, ".IFM", marketed by Brighterion, Inc. (San Francisco, Calif.).

[0103] Steps **126** and **127** can both be implemented by the apparatus of FIG. **11** that executes algorithm **1100**.

[0104] A step **128** exports the .IFM-type smart-agent predictive model markup language documents to a user-service consumer, e.g., using an apparatus for executing a data-science-as-a-service algorithm from a network server, as illustrated in FIGS. **6** and **9**.

[0105] In alternative method embodiments of the present invention, Method **100** further includes a step **130** for building a data mining predictive model (e.g. **612**, FIG. **6**) by applying the same data from the samples of the enriched-data records **124** as an input to an apparatus for generating a data mining algorithm. For example, as illustrated in FIG. **22**. A data-tree result **131** is transformed by a step **132** into a data-mining predictive model markup language document that is stored in a machine-readable storage mechanism. For example, as an industry standardized predictive model markup language (PMML) document. PMML is an XML-based file format developed by the Data Mining Group (dmg.org) to provide a way for applications to describe and exchange models produced by data mining and machine learning algorithms. It supports common models such as logistic regression and feed-forward neural networks. Further information related to data mining is included in Adjaoute '592. Special attention should be placed on FIGS. **11-30** and the descriptions of the data-mining technology in Columns **18-20**.

[0106] Method **100** further includes an alternative step **134** for building a neural network predictive model (e.g. **613**, FIG. **6**) by applying the same data from the samples of the enriched-data records **124** as an input to an apparatus for generating a neural network algorithm. For example, as illustrated in FIG. **12-17**. A nodes/weight result **135** is trans-

formed by a step **136** into a neural-network predictive model markup language document that is stored in a machine-readable storage mechanism. Further information related to neural networks is included in Adjaoute '592. Special attention should be placed on FIGS. **13-15** and the descriptions of the neural network technology in Columns **14-16**.

[0107] Method **100** further includes an alternative step **138** for building a case-based-reasoning predictive model (e.g. **614**, FIG. **6**) by applying the same data from the samples of the enriched-data records **124** as an input to an apparatus for generating a cased-based reasoning algorithm. As suggested by the algorithm of FIG. **25-26**. A cases result **139** is transformed into a case-based-reasoning predictive model markup language document **140** that is stored in a machine-readable storage mechanism. Further information related to case-based-reasoning is included in Adjaoute '592. Special attention should be placed on FIGS. **24-25** and the descriptions of the case-based-reasoning technology in Columns **20-21**.

[0108] Method **100** further includes an alternative step **142** for building a clustering predictive model (e.g. **615**, FIG. **6**) by applying the same data from the samples of the enriched-data records **124** as an input to an apparatus for generating a clustering algorithm. A clusters result **143** is transformed by a step **144** into a clustering predictive model markup language document that is stored in a machine-readable storage mechanism.

[0109] Clustering here involves the unsupervised classification of observations, data items, feature vectors, and other patterns into groups. In supervised learning, a collection of labeled patterns are used to determine class descriptions which, in turn, can then be used to label the new pattern. In the case of unsupervised clustering, the challenge is in grouping a given collection of unlabeled patterns into meaningful clusters.

[0110] Typical pattern clustering algorithms involve the following steps:

[0111] (1) Pattern representation: extraction and/or selection;

[0112] (2) Pattern proximity measure appropriate to the data domain;

[0113] (3) Clustering, and

[0114] (4) Assessment of the outputs.

Feature selection algorithms identify the most effective subsets of the original features to use in clustering. Feature extraction makes transformations of the input features into new relevant features. Either one or both of these techniques is used to obtain an appropriate set of features to use in clustering. Pattern representation refers to the number of classes and available patterns to the clustering algorithm. Pattern proximity is measured by a distance function defined on pairs of patterns.

[0115] A clustering is a partition of data into exclusive groups or fuzzy clustering. Using Fuzzy Logic, A fuzzy clustering method assigns degrees of membership in several clusters to each input pattern. Both similarity measures and dissimilarity measures are used here in creating clusters.

[0116] Method **100** further includes an alternative step **146** for building a business rules predictive model (e.g. **616**, FIG. **6**) by applying the same data from the samples of the enriched-data records **124** as an input to an apparatus for generating a business rules algorithm. As suggested by the algorithm of FIG. **27-29**. A rules result **147** is transformed by a step **148** into a business rules predictive model markup language document that is stored in a machine-readable storage mechanism. Further information related to rule-based-reasoning is included in Adjaoute '592. Special attention should be placed on FIG. **27** and the descriptions of the rule-based-reasoning technology in Columns **20-21**.

[0117] Each of Documents **128**, **132**, **136**, **140**, **144**, and **146** is a tangible machine-readable transformation of a trained model and can be sold, transported, installed, used, adapted, maintained, and modified by a user-service consumer or provider.

[0118] FIG. **2** represents an apparatus **200** for executing an encryption algorithm **202** and a matching decoding algorithm **204**, e.g., a standard triple-DES device that uses two keys. The Data Encryption Standard (DES) is a widely understood and once predominant symmetric-key algorithm for the encryption of electronic data. DES is the archetypal block cipher—an algorithm that takes data and transforms it through a series of complicated operations into another cipher text bit string of the same length. In the case of DES, the block size is 64 bits. DES also uses a key to customize the transformation, so that decryption can supposedly only be performed by those who know the particular key used to encrypt. The key ostensibly consists of 64 bits; however, only 56 of these are actually used by the algorithm. Eight bits are used solely for checking parity, and are thereafter discarded. Hence the effective key length is 56 bits.

[0119] Triple DES (3DES) is a common name in cryptography for the Triple Data Encryption Algorithm (TDEA or Triple DEA) symmetric-key block cipher, which applies the Data Encryption Standard (DES) cipher algorithm three times to each data block. The original DES cipher's key size of 56-bits was generally sufficient when that algorithm was designed, but the availability of increasing computational power made brute-force attacks feasible. Triple DES provides a relatively simple method of increasing the key size of DES to protect against such attacks, without the need to design a completely new block cipher algorithm.

[0120] In FIG. **2**, algorithms **202** and **204** transform data in separate records in storage memory back and forth between private data (P) and triple encrypted data (C).

[0121] FIGS. **3**A, **3**B, and **3**C represent an algorithm **300** for cleaning up the raw data **106** in stored data records, field-by-field, record-by-record. What is meant by "cleaning up" is that inconsistent, missing, and illegal data in each field are removed or reconstituted. Some types of fields are very restricted in what is legal or allowed. A record **302** is fetched from the raw data **304** and for each field **306** a test **306** sees if the data value reported is numeric or symbolic. If numeric, a data dictionary **308** is used by a step **310** to see if such data value is listed as valid. If symbolic, another data dictionary **312** is used by a step **314** to see if such data value is listed as valid.

[0122] For numeric data values, a test **316** is used to branch if not numeric to a step **318** that replaces the numeric value. FIG. **3**B illustrates such in greater detail. A test **320** is used to check if the numeric value is within an acceptable range. If not, step **318** is used to replace the numeric value.

[0123] For symbolic data values, a test **322** is used to branch if not numeric to a step **324** that replaces the symbolic value. FIG. **3**C illustrates such in greater detail. A test **326** is used to check if the symbolic value is an allowable one. If yes, a step **328** checks if the value is allowed in a set. If yes, then a return **330** proceeds to the next field. If no, step **324** replaces the symbolic value.

[0124] If in step **326** the symbolic value in the field is not an allowed value, a step **332** asks if the present field is a zip code field. If yes, a step **334** asks if it's a valid zip code. If yes, the processing moves on to the next field with step **330**. Otherwise, it calls on step **324** to replace the symbolic value.

[0125] If in step **332** the field is not an allowed value a zip code field, then a step **338** asks if the field is reserved for telephone and fax numbers. If yes, a step **340** asks if it's a valid telephone and fax number. If yes, the processing moves on to the next field with step **330**. Otherwise, it calls on step **324** to replace the symbolic value.

[0126] If in step **338** the field is not a field reserved for telephone and fax numbers, then a step **344** asks if the present field is reserved for dates and time. If yes, a step **346** asks if it's a date or time. If yes, the processing moves on to the next field with step **330**. Otherwise, it calls on step **324** to replace the symbolic value.

[0127] If in step **344** the field is not a field reserved for dates and time, then a step **350** applies a Smith-Waterman algorithm to the data value. The Smith-Waterman algorithm does a local-sequence alignment. It's used to determine if there are any similar regions between two strings or sequences. For example, to recognize "Avenue" as being the same as "Ave."; and "St." as the same as "Street"; and "Mr." as the same as "Mister". A consistent, coherent terminology is then enforceable in each data field without data loss. The Smith-Waterman algorithm compares segments of all possible lengths and optimizes the similarity measure without looking at the total sequence. Then the processing moves on to a next field with step **330**.

[0128] FIG. **3**B represents what happens inside step **318**, replace numeric value. The numeric value to use as a replacement depends on any flags or preferences that were set to use a default, the average, a minimum, a maximum, or a null. A step **360** tests if user preferences were set to use a default value. If yes, then a step **361** sets a default value and returns to do a next field in step **330**. A step **362** tests if user preferences were set to use an average value. If yes, then a step **361** sets an average value and returns to do the next field in step **330**. A step **364** tests if user preferences were set to use a minimum value. If yes, then a step **361** sets a minimum value and returns to do the next field in step **330**. A step **366** tests if user preferences were set to use a maximum value. If yes, then a step **361** sets a maximum value and returns to do the next field in step **330**. A step **368** tests if user preferences were set to use a null value. If yes, then a step **361** sets a null value and returns to do the next field in step **330**. Otherwise, a step **370** removes the record and moves on to the next record.

[0129] FIG. **3**C represents what happens inside step **324**, replace symbolic value. The symbolic value to use as a replacement depends on if flags were set to use a default, the average, or null. A step **374** tests if user preferences were set to use a default value. If yes, then a step **375** sets a default value and returns to do the next field in step **330**. A step **376** tests if user preferences were set to use an average value. If yes, then a step **377** sets an average value and returns to do the next field in step **330**. A step **378** tests if user preferences were set to use a null value. If yes, then a step **379** sets a null value and returns to do the next field in step **330**. Otherwise, a step **380** removes the record and moves on to a next record.

[0130] FIG. **4** represents the apparatus for executing sampling algorithm **116**. A sampling algorithm **400** takes cleaned, raw-data **402** and asks in step **404** if method embodiments of the present invention data are supervised. If so, a step **406** creates one data set "C**1**" **408** and a "Cn" **410** for each class. Stratified selection is used if needed. Each application carries its own class set, e.g., stocks portfolio managers use buy-sell-hold classes; loans managers use loan interest rate classes; risk assessment managers use fraud-no_fraud-suspicious classes; marketing managers use product-category-to-suggest classes; and, cybersecurity uses normal_behavior-abnormal_behavior classes. Other classes are possible and useful. For all classes, a step **412** and **413** asks if the class is abnormal (e.g., uncharacteristic). If not, a step **414** and **415** down-sample and produce sampled records of the class **416** and **417**. Then a step **418** and **419** splits the remaining data into separate training sets **420** and **421**, separate test sets **422** and **423**, and separate blind sets **424** and **425**.

[0131] If in step **404** method embodiments of the present invention data was determined to be unsupervised, a step **430** creates one data set with all the records and stores them in a memory device **432**. A step **434** down-samples all of them and stores those in a memory device **436**. Then a step **438** splits the remaining data into separate a training set **440**, a separate test set **442**, and a separate blind set **444**.

[0132] Later applications described herein also require data cleanup and data enrichment, but they do not require the split training sets produced by sampling algorithm **400**. Instead they process new incoming records that are cleaned and enriched to make a prediction, a score, or a decision, record one at a time.

[0133] FIGS. **5**A and **5**B together represent an apparatus **500** with at least one processor for executing a specialized data enrichment algorithm that works both to enrich the profiling criteria for smart-agents and to enrich the data fields for all the other general predictive models. They all are intended to work together in parallel with the smart-agents in operational use.

[0134] In FIG. **5**A, a plurality of training sets, herein **502** and **502**, for each class C**1** . . . Cn are input for each data field of a record in a step **506**. Such supervised and unsupervised training sets correspond to training sets **420**, **421**, and **440** (FIG. **4**). More generally, flat data **110**, **120** and sampled data **118** (FIG. **1**). A step **508** asks if there are too many distinct data values, e.g., more than a threshold data value stored in memory. For example, data that is so random as to reveal no information and nothing systemic. If so, a step **510** excludes that field and thereby reduces the list of fields. Otherwise, a step **512** asks if there is a single data value. Again, if so such field is not too useful in later steps, and step **510** excludes that field as well. Otherwise, a step **514** asks if the Shannon entropy is too small, e.g., less than a threshold data value stored in memory. The Shannon entropy is calculable using a conventional formula:

$$H(X) = \sum_{i=1}^{n} p(x_i) I(x_i) = \sum_{i=1}^{n} p(x_i) \log_b \frac{1}{p(x_i)} = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i),$$

[0135] The entropy of a message is its amount of uncertainty. It increases when the message is closer to random, and decreases when it is less random. The idea here is that the less likely an event is, the more information it provides when it occurs. If the Shannon entropy is too small, step **510** excludes that field. Otherwise, a step **516** reduces the number of fields in the set of fields carried forward as those that actually provide useful information.

[0136] A step **517** asks if the field type under inspection at that instant is symbolic or numeric. If symbolic, a step **518** provides AI behavior grouping. For example, colors or the names of boys. Otherwise, a step **520** does a numeric fuzzification in which a numeric value is turned into a membership of one or more fuzzy sets. Then a step **522** produces a reduced set of transformed fields. A step **524** asks if the number of criteria or data fields remaining meets a predefined target number. The target number represents a judgment of the optimum spectrum of profiling criteria data fields that will be needed to produce high performance smart-agents and good predictive models.

[0137] If yes, a step **526** outputs a final list of profiling criteria and data fields needed by the smart-agent steps **126** and **127** in FIG. **1** and all the other predictive model steps **130**, **131**, **134**, **135**, **138**, **139**, **142**, **143**, **146**, and **147**.

[0138] If not, the later steps in Method **100** need richer data to work with than is on-hand at the moment. The enrichment provided represents the most distinctive advantage that embodiments of the present invention have over conventional methods and systems. A step **528** (FIG. **5**B) begins a process to generate additional profiling criteria and newly derived data fields. A step **530** chooses an aggregation type. A step **532** chooses a time range for a newly derived field or profiling criteria. A step **534** chooses a filter. A step **536** chooses constraints. A step **538** chooses the fields to aggregate. A step **540** chooses a recursive level.

[0139] A step **542** assesses the quality of the newly derived field by importing test set classes C**1** . . . Cn **544** and **546**. It assesses the profiling criteria and data field quality for large enough coverage in a step **548**, the maximum transaction/event false positive rate (TFPR) below a limit in a step **550**, the average TFPR below a limit in a step **552**, transaction/event detection rate (TDR) above a threshold in a step **554**, the transaction/event review rate (TRR) trend below a threshold in a step **556**, the number of conditions below a threshold in a step **560**, the number of records is above a threshold in a step **562**, and the time window is optimal a step **564**.

[0140] If the newly derived profiling criteria or data field has been qualified, a step **566** adds it to the list. Otherwise, the newly derive profiling criteria or data field is discarded in a step **568** and returns to step **528** to try a new iteration with updated parameters.

[0141] Thresholds and limits are stored in computer storage memory mechanisms as modifiable digital data values that are non-transitory. Thresholds are predetermined and is "tuned" later to optimize overall operational performance. For example, by manipulating the data values stored in a computer memory storage mechanism through an administrator's console dashboard. Thresholds are digitally compared to incoming data, or newly derived data using conventional devices.

Using the Data Science

[0142] Once the predictive model technologies have been individually trained by both supervised and unsupervised data and then packaged into a PMML Document, one or more of them can be put to work in parallel render a risk or a decision score for each new record presented to them. At a minimum, only the smart-agent predictive model technology will be employed by a user-consumer. But when more than one predictive model technology is added in to leverage their respective synergies, a decision engine algorithm is needed to single out which predicted class produced in parallel by several predictive model technologies would be the best to rely on.

[0143] FIG. **6** is a flowchart diagram of a method **600** for using the PMML Documents (**128**, **132**, **136**, **140**, **144**, and **148**) of FIG. **1** with an algorithm for the run-time operation of parallel predictive model technologies.

[0144] Method **600** depends on an apparatus to execute an algorithm to use the predictive technologies produced by method **100** (FIG. **1**) and exported as PMML Documents. Method **600** can provide a substantial commercial advantage in a real-time, record-by-record application by a business. One or more PMML Documents **601-606** are imported and put to work in parallel as predictive model technologies **611-616** to simultaneously predict a class and its confidence in that class for each new record in a raw data record input **618** that are presented to them.

[0145] It is important that these records receive a data-cleanup **620** and a data-enrichment, as were described for steps **108** and **122** in FIG. **1**. A resulting enriched data **624** with newly derived fields in the records is then passed in parallel for simultaneous consideration and evaluation by all the predictive model technologies **611-616** present. Each will transform its inputs into a predicted class **631-636** and a confidence **641-646** stored in a computer memory storage mechanism.

[0146] A record-by-record decision engine **650** inputs user strategies in the form of flag settings **652** and rules **654** to decision on which to output as a prevailing predicted class output **660** and to compute a normalized confidence output **661**. Such record-by-record decision engine **650** is detailed here next in FIG. **7**.

Typical examples of prevailing predicted classes **660**:

FIELD OF APPLICATION	OUTPUT CLASSES
stocks use class	buy , buy, sell, hold, etc.
loans use class	provide a loan with an interest , or not
risk use class	fraud, no fraud, suspicious
marketing use class	category of product to suggest
cybersecurity use class	normal behavior, abnormal, etc.

[0147] Method **600** works with at least two of the predictive models from steps **128**, **132**, **136**, **140**, **144**, and **148** (of FIG. **1**). The predictive models each simultaneously produce a score and a score-confidence level in parallel sets, all from a particular record in a plurality of enriched-data records. These combine into a single result to return to a user-service consumer as a decision.

[0148] Further information related to combining models is included in Adjaoute '592. Special attention should be placed on FIG. **30** and the description in Column **22** on combining the technologies. There, the neural network, smart-agent, data mining, and case-based reasoning technologies all come together to produce a final decision, such as if a particular electronic transaction is fraudulent, in a different application, if there is network intrusion.

[0149] FIG. **7** is a flowchart diagram of an apparatus with an algorithm **700** for the decision engine **650** of FIG. **6**. Algorithm **700** chooses which predicted class **631-636**, or a composite of them, should be output as prevailing predicted class **660**. Switches or flag settings **652** are used to control the decision outcome and are fixed by the user-service consumer in operating their business based on the data science embod-

ied in Documents **601-606**. Rules **654** too can include business rules like, "always follow the smart agent's predicted class if its confidence exceeds 90%."

[0150] A step **702** inspects the rule type then in force. Compiled flag settings rules are fuzzy rules (business rules) developed with fuzzy logic. Fuzzy rules are used to merge the predicted classes from all the predictive models and technologies **631-636** and decide on one final prediction, herein, prevailing predicted class **660**. Rules **654** are either manually written by analytical engineers, or they are automatically generated when analyzing the enriched training data **124** (FIG. **1**) in steps **126**, **130**, **134**, **138**, **142**, and **146**.

[0151] If in step **702** it is decided to follow "compiled rules", then a step **704** invokes the compiled flag settings rules and returns with a corresponding decision **706** for output as prevailing predicted class **660**.

[0152] If in step **702** it is decided to follow "smart agents", then a step **708** invokes the smart agents and returns with a corresponding decision **710** for output as prevailing predicted class **660**.

[0153] If in step **702** it is decided to follow "predefined rules", then a step **712** asks if the flag settings should be applied first. If not, a step **714** applies a winner-take-all test to all the individual predicted classes **631-636** (FIG. **6**). A step tests if one particular class wins. If yes, a step **718** outputs that winner class for output as prevailing predicted class **660**.

[0154] If not in step **716**, a step **720** applies the flag settings to the individual predicted classes **631-636** (FIG. **6**). Then a step **722** asks there is a winner rule. If yes, a step **724** outputs that winner rule decision for output as prevailing predicted class **660**. Otherwise, a step **726** outputs an "otherwise" rule decision for output as prevailing predicted class **660**.

[0155] If in step **712** flag setting are to be applied first, a step **730** applies the flags to the individual predicted classes **631-636** (FIG. **6**). Then a step **732** asks if there is a winner rule. If yes, then a step **734** outputs that winner rule decision for output as prevailing predicted class **660**. Otherwise, a step **736** asks if the decision should be winner-take-all. If no, a step **738** outputs an "otherwise" rule decision for output as prevailing predicted class **660**.

[0156] If in step **736** it should be winner-take-all, a step **740** applies winner-take-all to each of the individual predicted classes **631-636** (FIG. **6**). Then a step **742** asks if there is now a winner class. If not, step **738** outputs an "otherwise" rule decision for output as prevailing predicted class **660**. Otherwise, a step **744** outputs a winning class decision for output as prevailing predicted class **660**.

[0157] Compiled flag settings rules in step **704** are fuzzy rules, e.g., business rules with fuzzy logic. Such fuzzy rules are targeted to merge the predictions **631-636** into one final prediction **660**. Such rules are either written by analytical engineers or are generated automatically by analyses of the training data.

[0158] When applying flag settings to the individual predictions, as in step **730**, an algorithm for a set of ordered rules that indicate how to handle predictions output by each prediction technology. FIG. **8** illustrates this further.

[0159] FIG. **8** shows flag settings **800** as a set of ordered rules **801-803** that indicate how to handle each technology prediction **631-636** (FIG. **6**). For each technology **611-616**, there is at least one rule **801-803** that provides a corresponding threshold **811-813**. Each are then compared to prediction confidences **641-646**.

[0160] When a corresponding incoming confidence **820** is higher or equal to a given threshold **811-813** provided by a rule **801-803**, the technology **611-616** associated with rule **801-803** is declared "winner" and its class and confidence are used as the final prediction. When none of the technologies **611-616** win, an "otherwise rule" determines what to do. In this case, a clause indicates how to classify the transaction (fraud/not-fraud) and it sets the confidence to zero.

Consider the following example:

Flags Settings			Predictions		
Prediction Type	Prediction Technology	Threshold	Class	Prediction Technology	Confidence
All	Smart-agents	0.75	Fraud	Smart-agents	0.7
All	Data Mining	0.7		Data Mining	0.8
. . .	. . .	. . .	, , ,	. . .	. .

A first rule, e.g., **801**, looks at a smart-agent confidence (e.g., **641**) of 0.7, but that is below a given corresponding threshold (e.g., **811**) of 0.75 so inspection continues.

[0161] A second rule (e.g., **802**) looks at a data mining confidence (e.g., **642**) of 0.8 which is above a given threshold (e.g., **812**) of 0.7. Inspection stops here and decision engine **650** uses the Data Mining prediction (e.g., **632**) to define the final prediction (e.g., **660**). Thus it is decided in this example that the incoming transaction is fraudulent with a confidence of 0.8.

[0162] It is possible to define rules that apply only to specific kinds of predictions. For example, a higher threshold is associated with predictions of fraud, as opposed to prediction classes of non-frauds.

[0163] A winner-take-all technique groups the individual predictions **631-636** by their prediction output classes. Each Prediction Technology is assigned its own weight, one used when it predicts a fraudulent transaction, another used when it predicts a valid transaction. All similar predictions are grouped together by summing their weighted confidence. The sum of the weighted confidences is divided by the sum of the weights used in order to obtain a final confidence between 0.0 and 1.0.

For example:

Weights			Predictions		
Prediction Technology	Weight-Fraud	Weight-Valid	Class	Prediction Technology	Confidence
Smart-agents	2	2	Fraud	Smart-agents	0.7
Data Mining	1	1	Fraud	Data Mining	0.8
Case Based Reasoning	2	2	Valid	Cases Based Reasoning	0.4

Here in the Example, two prediction technologies (e.g., **611** and **612**) are predicting (e.g., **631** and **632**) a "fraud" class for the transaction. So their cumulated weighted confidence here is computed as: 2*0.7+1*0.8 which is 2.2, and stored in computer memory. Only case-based-reasoning (e.g., **614**) predicts (e.g., class **634**) a "valid" transaction, so its weighted confidence here is computed as: 1*0.4, and is also stored in computer memory for comparison later.

[0164] Since the first computed value of 2.2 is greater than the second computed value of 0.4, this particular transaction in this example is decided to belong to the "fraud" class. The confidence is then normalized for output by dividing it by the sum of the weights that where associated with the fraud (2 and 1). So the final confidence (e.g., **661**) is computed by 2.2/(2+1) giving: 0.73.

[0165] Some models **611-616** may have been trained to output more than just two binary classes. A fuzzification can provide more than two slots, e.g., for buy/sell/hold, or declined/suspect/approved. It may help to group classes by type of prediction (fraud or not-fraud).

For example:

Weights			Predictions				
Prediction Technology	Weight-Fraud	Weight-Valid	Class	Prediction Technology	Confidence	Classes Value	Classes Type
Smart-agents	2	2	00	Smart-agents	0.6	00	Fraud
Data Mining	1	1	01	Data Mining	0.5	01	Fraud
Cases Based Reasoning	2	2	G	Cases Based Reasoning	0.7	G	Valid

[0166] In a first example, similar classes are grouped together. So fraud=2*0.6+1*0.5=1.7, and valid=2*0.7=1.4. The transaction in this example is marked as fraudulent.

[0167] In a second example, all the classes are distinct, with the following equation: 2*0.6 "00"+1*0.5 "01"+2*0.7 "G" so the winner is the class "G" and the transaction is marked as valid in this example.

[0168] Embodiments of the present invention integrate the constituent opinions of the technologies and make a single prediction class. How they integrate the constituent predictions **631-636** depend on a user-service consumers' selections of which technologies to favor and how to favor, and such selections are made prior to training the technologies, e.g., through a model training interface.

[0169] A default selection includes the results of the neural network technology, the smart-agent technology, the data mining technology, and the case-based reasoning technology. Alternatively, the user-service consumer may decide to use any combination of technologies, or to select an expert mode with four additional technologies: (1) rule-based reasoning technology; (2) fuzzy logic technology; (3) genetic algorithms technology; and (4) constraint programming technology.

[0170] One strategy that could be defined by a user-service consumer-consumer assigns one vote to each predictive technology **611-616**. A final decision **660** then stems from a majority decision reached by equal votes by the technologies within decision engine **650**.

[0171] Another strategy definable by a user-service consumer-consumer assigns priority values to each one of technologies **611-616** with higher priorities that more heavily determine the final decision, e.g., that a transaction is fraudulent and another technology with a lower priority determines that the transaction is not fraudulent, then method embodiments of the present invention use the priority values to discriminate between the results of the two technologies and determine that the transaction is indeed fraudulent.

[0172] A further strategy definable by a user-service consumer-consumer specifies instead a set of meta-rules to help choose a final decision **660** for output. These all indicate an output prediction class and its confidence level as a percentage (0-1000, or 0-1.0) proportional to how confident the system apparatus is in the prediction.

[0173] FIG. **9** illustrates a method **900** of business decision making that requires the collaboration of two businesses, a service provider **901** and a user-consumer **902**. The two businesses communicate with one another via secure Internet between network servers. The many data records and data files passed between them are hashed or encrypted by a triple-DES algorithm, or similar protection. It also possible to send a non-encrypted filed through an encrypted channel. Users of the platform would upload their data through SSL/TLS from a browser or from a command line interface (SCP or SFTP).

[0174] The service-provider business **901** combines method **100** (FIG. **1**) and method **600** (FIG. **6**) and their constituent algorithms. It accepts supervised and unsupervised training data **904** and strategies **906** from the user-service consumer business **902**. Method **100** then processes such as described above with FIGS. **1-8** to produce a full set of fully trained predictive models that are passed to method **600**.

[0175] New records from operations **906** provided, e.g., in real-time as they occur, are passed after being transformed by encryption from the user-service consumer business **902** to the service provider business **901** and method **600**. An ongoing run of scores, predictions, and decisions **908** (produced by method **600** according to the predictive models of method **100** and the strategies **905** and training data **904**) are returned to user-service consumer business **902** after being transformed by encryption.

[0176] With some adjustment and reconfiguration, method **900** is trained for a wide range of uses, e.g., to classify fraud/no-fraud in payment transaction networks, to predict buy/sell/hold in stock trading, to detect malicious insider activity, and to call for preventative maintenance with machine and device failure predictions.

[0177] Referring again to FIG. **9**, another method of operating an artificial intelligence machine to improve their decisions from included predictive models begins by deleting with at least one processor a selected data field and any data values contained in the selected data field from each of a first series of data training records stored in a memory of the artificial intelligence machine to exclude each data field in the first series of data training records that has more than a threshold number of random data values, or that has only one repeating data value, or that has too small a Shannon entropy, and using an information gain to select the most useful data fields, and then transforming a surviving number of data fields in all the first series of data training records into a corresponding reduced-field series of data training records stored in the memory of the artificial intelligence machine.

[0178] A next step includes adding with the at least one processor a new derivative data field to all the reduced-field series of data training records stored in the memory and initializing each added new derivative data field with a new data value, and including an apparatus for executing an algorithm to either change real scaler numeric data values into fuzzy values, or if symbolic, to change a behavior group data value, and testing that a minimum number of data fields

survive, and if not, then to generate a new derivative data field and fix within each an aggregation type, a time range, a filter, a set of aggregation constraints, a set of data fields to aggregate, and a recursive level, and then assessing the quality of a newly derived data field by testing it with a test set of data, and then transforming the results into an enriched-field series of data training records stored in the memory of the artificial intelligence machine.

[0179] A next step includes verifying with the at least one processor that each predictive model if trained with the enriched-field series of data training records stored in the memory produces decisions having fewer errors than the same predictive model trained only with the first series of data training records.

[0180] A further step includes recording a data-enrichment descriptor into the memory to include an identity of selected data fields in a data training record format of the first series of data training records that were subsequently deleted, and which newly derived data fields were subsequently added, and how each newly derived data field was derived and from which information sources.

[0181] A next step includes causing the at least one processor of the artificial intelligence machine to start extracting decisions from a new series of data records of new events by receiving and storing the new series of data records in the memory of the artificial intelligence machine.

[0182] A further step includes causing the at least one processor to fetch the data-enrichment descriptor and use it to select which data fields to delete and then deleting all the data values included in the selected data fields from each of a new series of data records of new events. Each data field deleted matches a data field in the first series of data training records had more than a threshold number of random data values, or that had only one repeating data value, or that had too small a Shannon entropy.

[0183] A next step includes adding with the at least one processor a new derivative data field to each record of the new series of data records stored in the memory according to the data-enrichment descriptor, and initializing each added new derivative data field with a new data value stored in the memory. Each new derivative data field added matches a new derivative data field added to the enriched-field series of data training records in which real scaler numeric data values were changed into fuzzy values, or if symbolic, were changed into a behavior group data value stored in the memory, and were tested that a minimum number of data fields survive, and if not, then that generated a new derivative data field and fixed within each an aggregation type, a time range, a filter, a set of aggregation constraints, a set of data fields to aggregate, and a recursive level.

[0184] The method concludes by producing and outputting a series of predictive decisions with the at least one processor that operates at least one predictive model algorithm derived from one originally built and trained with records having a same record format described by the data-enrichment descriptor and stored in the memory of the artificial intelligence machine.

[0185] FIG. **10** represents an apparatus for executing an algorithm **1000** for reclassifying a decision **660** (FIG. **6**) for business profitability reasons. For example, when a payment card transaction for a particular transaction amount $X has already been preliminarily "declined" and included in a decision **1002** (and **660**, FIG. **6**) according to some other decision model. A test **1004** compares a dollar transaction "threshold amount-A" **1006** to a computation **1008** of the running average business a particular user has been doing with the account involved. The rational for doing this is that valuable customers who do more than an average amount (threshold-A **1006**) of business with their payment card should not be so easily or trivially declined. Some artificial intelligence deliberation and reconsideration is appropriate.

[0186] If, however test **1004** decides that the accountholder has not earned special processing, a "transaction declined" decision **1010** is issued as final (transaction-declined **110**). Such is then forwarded by a financial network to the merchant point-of-sale (POS).

[0187] But when test **1004** decides that the accountholder has earned special processing, a transaction-preliminarily-approved decision **1012** is carried forward to a test **1014**. A threshold-B transaction amount **1016** is compared to the transaction amount $X. Essentially, threshold-B transaction amount **1016** is set at a level that would relieve qualified accountholders of ever being denied a petty transaction, e.g., under $250, and yet not involve a great amount of risk should the "positive" scoring indication from the "other decision model" not prove much later to be "false". If the transaction amount $X is less than threshold-B transaction amount **1016**, a "transaction approved" decision **1018** is issued as final. Such is then forwarded by the financial network to the merchant CP/CNP, unattended terminal, ATM, online payments, etc.

[0188] If the transaction amount $X is more than threshold-B transaction amount **1016**, a transaction-preliminarily-approved decision **1020** is carried forward to a familiar transaction pattern test **1022**. An abstract **1024** of this account's transaction patterns is compared to the instant transaction. For example, if this accountholder seems to be a new parent with a new baby as evidenced in purchases of particular items, then all future purchases that could be associated are reasonably predictable. Or, in another example, if the accountholder seems to be on business in a foreign country as evidenced in purchases of particular items and travel arrangements, then all future purchases that could be reasonably associated are to be expected and scored as lower risk. And, in one more example, if the accountholder seems to be a professional gambler as evidenced in cash advances at casinos, purchases of specific things and arrangements, then these future purchases too could be reasonably associated are be expected and scored as lower risk.

[0189] So if the transaction type is not a familiar one, then a "transaction declined" decision **1026** is issued as final. Such is then forwarded by the financial network **106** to the merchant (CP and/or CNP) and/or unattended terminal/ATM. Otherwise; a transaction-preliminarily-approved decision **1028** is carried forward to a threshold-C test **1030**.

[0190] A threshold-C transaction amount **1032** is compared to the transaction amount $X. Essentially, threshold-C transaction amount **1032** is set at a level that would relieve qualified accountholders of being denied a moderate transaction, e.g., under $2500, and yet not involve a great amount of risk because the accountholder's transactional behavior is within their individual norms. If the transaction amount $X is less than threshold-C transaction amount **1032**, a "transaction approved" decision **1034** is issued as final (transaction-approved). Such is then forwarded by the financial network **106** to the merchant (CP and/or CNP) and/or unattended terminal/ATM.

[0191] If the transaction amount $X is more than threshold-C transaction amount **1032**, a transaction-preliminarily-approved decision **1036** is carried forward to a familiar user device recognition test **1038**. An abstract **1040** of this account's user devices is compared to those used in the instant transaction.

[0192] So if the user device is not recognizable as one employed by the accountholder, then a "transaction declined" decision **1042** is issued as final. Such is then forwarded by the financial network **106** to the merchant (CP and/or CNP) and/or unattended terminal/ATM. Otherwise; a transaction-preliminarily-approved decision **1044** is carried forward to a threshold-D test **1046**.

[0193] A threshold-D transaction amount **1048** is compared to the transaction amount $X. Basically, the threshold-D transaction amount **1048** is set at a higher level that would avoid denying substantial transactions to qualified accountholders, e.g., under $10,000, and yet not involve a great amount of risk because the accountholder's user devices are recognized and their instant transactional behavior is within their individual norms. If the transaction amount $X is less than threshold-D transaction amount **1032**, a "transaction approved" decision **1050** is issued as final. Such is then forwarded by the financial network **106** to the merchant (CP and/or CNP) and/or unattended terminal/ATM.

[0194] Otherwise, the transaction amount $X is just too large to override a denial if the other decision model decision **1002** was "positive", e.g., for fraud, or some other reason. In such case, a "transaction declined" decision **1052** is issued as final (transaction-declined **110**). Such is then forwarded by the financial network **106** to the merchant (CP and/or CNP) and/or unattended terminal/ATM.

[0195] In general, threshold-B **1016** is less than threshold-C **1032**, which in turn is less than threshold-D **1048**. It could be that tests **1022** and **1038** would serve profits better if swapped in FIG. **10**. Embodiments of the present invention would therefore include this variation as well. It would seem that threshold-A **1006** should be empirically derived and driven by business goals.

[0196] The further data processing required by technology **1000** occurs in real-time while merchant (CP and CNP, ATM and all unattended terminal) and users wait for approved/declined data messages to arrive through financial network. The consequence of this is that the abstracts for this-account's-running-average-totals **1008**, this account's-transaction-patterns **1024**, and this-account's-devices **1040** must all be accessible and on-hand very quickly. A simple look-up is preferred to having to compute the values. The smart agents and the behavioral profiles they maintain and that we've described in this Application and those we incorporate herein by reference are up to doing this job well. Conventional methods and apparatus may struggle to provide this information quickly enough.

[0197] FIG. **10** represents for the first time in machine learning an apparatus that allows a different threshold for each customer. It further enables different thresholds for the same customer based on the context, e.g., a Threshold-**1** while traveling, a Threshold-**2** while buying things familiar with his purchase history, a Threshold-**3** while in same area where they live, a Threshold-**4** during holidays, a Threshold-**5** for nights, a Threshold-**6** during business hours, etc.

[0198] FIG. **11** represents an algorithm that executes as smart-agent production apparatus **1100**, and is included in the build of smart-agents in steps **126** and **127** (FIG. **1**), or as step **611** (FIG. **6**) in operation. The results are either exported as an .IFM-type XML document in step **128**, or used locally as in method **600** (FIG. **6**). Step **126** (FIG. **1**) builds a population of smart-agents and their profiles that are represented in FIG. **11** as smart-agents S**1** **1102** and Sn **1104**. Step **127** (FIG. **1**) initialized that build. Such population can reach into the millions for large systems, e.g., those that handle payment transaction requests nationally and internationally for millions of cardholders (entities).

[0199] Each new record **1106** received, from training records **124**, or from data enrichment **622** in FIG. **6**, is inspected by a step **1108** that identifies the entity unique to the record that has caused to record to be generated. A step **1110** gets the corresponding smart-agent that matches this identification from the initial population of smart-agents **1102**, **1102** it received in step **128** (FIG. **1**). A step **1112** asks if any were not found. A step **1114** uses default profiles optimally defined for each entity, and to create and initialize smart-agents and profiles for entities that do not have a match in the initial population of smart-agents **1102**, **1102**. A step **1116** uses the matching smart-agent and profile to assess record **1106** and issues a score **1118**. A step **1120** updates the matching smart-agent profile with the new information in record **1106**.

[0200] A step **1122** dynamically creates/removes/updates and otherwise adjusts attributes in any matching smart-agent profile based on a content of records **1106**. A step **1124** adjusts an aggregation type (count, sum, distinct, ratio, average, minimum, maximum, standard deviation, . . .) in a matching smart-agent profile. A step **1126** adjusts a time range in a matching smart-agent profile. A step **1128** adjusts a filter based on a reduced set of transformed fields in a matching smart-agent profile. A step **1130** adjusts a multi-dimensional aggregation constraint in a matching smart-agent profile. A step **1132** adjusts an aggregation field, if needed, in the matching smart-agent profile. A step **1134** adjusts a recursive level in the matching smart-agent profile.

[0201] FIGS. **12-29** provide greater detail regarding the construction and functioning of algorithms that are employed in FIGS. **1-11**.

Neural Network Technology

[0202] FIG. **12** is a schematic diagram of the neural network architecture used in method embodiments of the present invention. Neural network **1200** consists of a set of processing elements or neurons that are logically arranged into three layers: (1) input layer **1201**; (2) output layer **1202**; and (3) hidden layer **1203**. The architecture of neural network **1200** is similar to a back propagation neural network, but its training, utilization, and learning algorithms are different. The neurons in input layer **1201** receive input fields from a training table. Each of the input fields are multiplied by a weight such as weight "Wij" **1204** a to obtain a state or output that is passed along another weighted connection with weights "Vjt" **1205** between neurons in hidden layer **1202** and output layer **1203**. The inputs to neurons in each layer come exclusively from output of neurons in a previous layer, and the output from these neurons propagate to the neurons in the following layers.

[0203] FIG. **13** is a diagram of a single neuron in the neural network used in method embodiments of the present invention. Neuron **1300** receives input "i" from a neuron in a previous layer. Input "i" is multiplied by a weight "Wih" and processed by neuron **1300** to produce state "s". State "s" is

then multiplied by weight "V_{hi}" to produce output "i" that is processed by neurons in the following layers. Neuron **1300** contains limiting thresholds **1301** that determine how an input is propagated to neurons in the following layers.

[0204] FIG. **14** is a flowchart of an algorithm **1400** for training neural networks with a single hidden layer that builds incrementally during a training process. The hidden layers may also grow in number later during any updates. Each training process computes a distance between all the records in a training table, and groups some of the records together. In a first step, a training set "S" and input weights "bi" are initialized. Training set "S" is initialized to contain all the records in the training table. Each field "i" in the training table is assigned a weight "bi" to indicate its importance. The input weights "bi" are selected by a client. A distance matrix D is created. Distance matrix D is a square and symmetric matrix of size N×N, where N is the total number of records in training set "S". Each element "Dij" in row "i" and column "j" of distance matrix D contains the distance between record "i" and record "j" in training set "S". The distance between two records in training set "S" is computed using a distance measure.

[0205] FIG. **15** illustrates a table of distance measures **1500** that is used in a neural network training process. Table **1500** lists distance measures that is used to compute the distance between two records Xi and Xj in training set "S". The default distance measure used in the training process is a Weighted-Euclidean distance measure that uses input weights "bi" to assign priority values to the fields in a training table.

[0206] In FIG. **14**, a distance matrix D is computed such that each element at row "i" and column "j" contains d(Xi, Xj) between records Xi and Xj in training set "S". Each row "i" of distance matrix D is then sorted so that it contains the distances of all the records in training set "S" ordered from the closest one to the farthest one.

[0207] A new neuron is added to the hidden layer of the neural network the largest subset "Sk" of input records having the same output is determined. Once the largest subset "Sk" is determined, the neuron group is formed at step **97**. The neuron group consists of two limiting thresholds, Blow and Θhigh, input weights "Wh", and output weights "Vh", such that Θlow=Dk, "j" and Θhigh=Dk, l, where "k" is the row in the sorted distance matrix D that contains the largest subset "Sk" of input records having the same output, "j" is the index of the first column in the subset "Sk" of row "k", and **1** is the index of the last column in the subset "Sk" of row "k". The input weights "Wh" are equal to the value of the input record in row "k" of the distance matrix D, and the output weights "Vh" are equal to zero except for the weight assigned between the created neuron in the hidden layer and the neuron in the output layer representing the output class value of any records belonging to subset "Sk". A subset "Sk" is removed from training set "S", and all the previously existing output weights "Vh" between the hidden layer and the output layer are doubled. Finally, the training set is checked to see if it still contains input records, and if so, the training process goes back. Otherwise, the training process is finished and the neural network is ready for use.

[0208] FIG. **16** is a flowchart of an algorithm **1600** for propagating an input record through a neural network. An input record is propagated through a network to predict if its output signifies a fraudulent transaction. A distance between the input record and the weight pattern "Wh" between the input layer and the hidden layer in the neural network is computed. The distance "d" is compared to the limiting thresholds low and high of the first neuron in the hidden layer. If the distance is between the limiting thresholds, then the weights "Wh" are added to the weights "Vh" between the hidden layer and the output layer of the neural network. If there are more neurons in the hidden layer, then the propagation algorithm goes back to repeat steps for the other neurons in the hidden layer. Finally, the predicted output class is determined according to the neuron at the output layer that has the higher weight.

[0209] FIG. **17** is a flowchart of an algorithm **1700** for updating the training process of a neural network. The training process is updated whenever a neural network needs to learn some new input record. Neural networks are updated automatically, as soon as data from a new record is evaluated by method embodiments of the present invention. Alternatively, the neural network may be updated offline.

[0210] A new training set for updating a neural network is created. The new training set contains all the new data records that were not utilized when first training the network using the training algorithm illustrated in FIG. **14**. The training set is checked to see if it contains any new output classes not found in the neural network. If there are no new output classes, the updating process proceeds with the training algorithm illustrated in FIG. **14**. If there are new output classes, then new neurons are added to the output layer of the neural network, so that each new output class has a corresponding neuron at the output layer. When the new neurons are added, the weights from these neurons to the existing neurons at the hidden layer of the neural network are initialized to zero. The weights from the hidden neurons to be created during the training algorithm are initialized as 2h, where "h" is the number of hidden neurons in the neural network prior to the insertion of each new hidden neuron. With this initialization, the training algorithm illustrated in FIG. **14** is started to form the updated neural network technology.

[0211] Evaluating if a given input record belongs to one class or other is done quickly and reliably with the training, propagation, and updating algorithms described.

Smart-Agent Technology

[0212] Smart-agent technology uses multiple smart-agents in unsupervised mode, e.g., to learn how to create profiles and clusters. Each field in a training table has its own smart-agent that cooperates with others to combine some partial pieces of knowledge they have about data for a given field, and validate the data being examined by another smart-agent. The smart-agents can identify unusual data and unexplained relationships. For example, by analyzing a healthcare database, the smart-agents would be able to identify unusual medical treatment combinations used to combat a certain disease, or to identify that a certain disease is only linked to children. The smart-agents would also be able to detect certain treatment combinations just by analyzing the database records with fields such as symptoms, geographic information of patients, medical procedures, and so on.

[0213] Smart-agent technology creates intervals of normal values for each one of the fields in a training table to evaluate if the values of the fields of a given electronic transaction are normal. And the technology determines any dependencies between each field in a training table to evaluate if the values of the fields of a given electronic transaction or record are coherent with the known field dependencies. Both goals can generate warnings.

[0214] FIG. **18** is a flowchart of an algorithm for creating intervals of normal values for a field in a training table. The algorithm illustrated in the flowchart is run for each field "a" in a training table. A list "La" of distinct couples ("vai", "nai") is created, where "vai" represents the i^{th} distinct value for field "a" and "nai" represents its cardinality, e.g., the number of times value "vai" appears in a training table. At step **119**, the field is determined to be symbolic or numeric. If the field is symbolic, each member of "La" is copied into a new list "Ia" whenever "nai" is superior to a threshold "Θmin" that represents the minimum number of elements a normal interval must include. "Θmin" is computed as "Θmin"=fmin*M, where M is the total number of records in a training table and fmin is a parameter specified by the user representing the minimum frequency of values in each normal interval. Finally, the relations (a, Ia) are saved in memory storage. Whenever a data record is to be evaluated by the smart-agent technology, the value of the field "a" in the data record is compared to the normal intervals created in "Ia" to determine if the value of the field "a" is outside the normal range of values for that given field.

[0215] If the field "a" is determined to be numeric, then the list "La" of distinct couples ("vai", nai) is ordered starting with the smallest value Va. At step **122**, the first element e=(val, nal) is removed from the list "La", and an interval NI=[val, val] is formed. At step **124**, the interval NI is enlarged to NI=[Val, vak] until Vak−Val>Θdist, where Θdist represents the maximum width of a normal interval. Θdist is computed as Θdist=(maxa−mina)/nmax, where nmax is a parameter specified by the user to denote the maximum number of intervals for each field in a training table. The values that are too dissimilar are not grouped together in the same interval.

[0216] The total cardinality "na" of all the values from "val" to "vak" is compared to "Θmin" to determine the final value of the list of normal intervals "Ia". If the list "Ia" is not empty, the relations (a, Ia) are saved. Whenever a data record is to be evaluated by the smart-agent technology, the value of the field "a" in the data record is compared to the normal intervals created in "Ia" to determine if the value of the field "a" is outside the normal range of values for that given field. If the value of the field "a" is outside the normal range of values for that given field, a warning is generated to indicate that the data record is likely fraudulent.

[0217] FIG. **19** is a flowchart of an algorithm **1900** for determining dependencies between each field in a training table. A list Lx of couples (vxi, nxi) is created for each field "x" in a training table. The values vxi in Lx for which (nxi/nT)>Θx are determined, where nT is the total number of records in a training table and Θx is a threshold value specified by the user. In a preferred embodiment, Θx has a default value of 1%. At step **132**, a list Ly of couples (vyi, nyi) for each field y, Y≠X, is created. The number of records nij where (x=xi) and (y=yj) are retrieved from a training table. If the relation is significant, that is if (nij/nxi)>Θxy, where Θxy is a threshold value specified by the user when the relation (X=xi) ⇔(Y=yj) is saved with the cardinalities nxi, nyj, and nij, and accuracy (nij/nxi). In a preferred embodiment, Θxy has a default value of 85%.

[0218] All the relations are saved in a tree made with four levels of hash tables to increase the speed of the smart-agent technology. The first level in the tree hashes the field name of the first field, the second level hashes the values for the first field implying some correlations with other fields, the third level hashes the field name with whom the first field has some correlations, and finally, the fourth level in the tree hashes the values of the second field that are correlated with the values of the first field. Each leaf of the tree represents a relation, and at each leaf, the cardinalities nxi, nyj, and nij are stored. This allows the smart-agent technology to be automatically updated and to determine the accuracy, prevalence, and the expected predictability of any given relation formed in a training table.

[0219] FIG. **20** is a flowchart of an algorithm **2000** for verifying the dependencies between the fields in an input record. For each field "x" in the input record corresponding to an electronic transaction, the relations starting with [(X=xi) ⇔ . . .] are found in the smart-agent technology tree. For all the other fields "y" in a transaction, the relations [(X=xi) ⇔(Y=v)] are found in the tree. A warning is triggered anytime Yj≠V. The warning indicates that the values of the fields in the input record are not coherent with the known field dependencies, which is often a characteristic of fraudulent transactions.

[0220] FIG. **21** is a flowchart of an algorithm **2100** for updating smart-agents. The total number of records nT in a training table is incremented by a new number of input records to be included in the update of the smart-agent technology. For the first relation (X=xi)⇔(Y=yj) previously created in the technology, the parameters nxi, nyj, and nij are retrieved, and, nxi, nyj, and nij are respectively incremented. The relation is verified to see if it is still significant for including it in a smart-agent tree. If the relation is not significant, then it is removed from the tree. Finally, a check is performed to see if there are more previously created relations (X=xi) ⇔*(Y=yj)] in the technology. If there are, then algorithm **2100** goes back and iterates until there are no more relations in the tree to be updated.

Data Mining Technology

[0221] FIG. **22** represents one way to implement a data mining algorithm as in steps **130-132** (FIG. **1**). More detail is incorporated herein by reference to Adjaoute '592, and especially that relating to its FIG. **22**. Here the data mining algorithm and the data tree of step **131** are highly advantaged by having been trained by the enriched data **124**. Such results in far superior training compared to conventional training with data like raw data **106**.

[0222] Data mining identifies several otherwise hidden data relationships, including: (1) associations, wherein one event is correlated to another event such as purchase of gourmet cooking books close to the holiday season; (2) sequences, wherein one event leads to another later event such as purchase of gourmet cooking books followed by the purchase of gourmet food ingredients; (3) classification, and, e.g., the recognition of patterns and a resulting new organization of data such as profiles of customers who make purchases of gourmet cooking books; (4) clustering, e.g., finding and visualizing groups of facts not previously known; and (5) forecasting, e.g., discovering patterns in the data that can lead to predictions about the future.

[0223] One goal of data mining technology is to create a decision tree based on records in a training database to facilitate and speed up the case-based reasoning technology. The case-based reasoning technology determines if a given input record associated with an electronic transaction is similar to any typical records encountered in a training table. Each record is referred to as a "case". If no similar cases are found,

a warning is issued to flag the input record. The data mining technology creates a decision tree as an indexing mechanism for the case-based reasoning technology. Data mining technology can also be used to automatically create and maintain business rules for a rule-based reasoning technology.

[0224] The decision tree is an "N-ary" tree, wherein each node contains a subset of similar records in a training database. (An N-ary tree is a tree in which each node has no more than N children.) In preferred embodiments, the decision tree is a binary tree. Each subset is split into two other subsets, based on the result of an intersection between the set of records in the subset and a test on a field. For symbolic fields, the test is if the values of the fields in the records in the subset are equal, and for numeric fields, the test is if the values of the fields in the records in the subset are smaller than a given value. Applying the test on a subset splits the subset in two others, depending on if they satisfy the test or not. The newly created subsets become the children of the subset they originated from in the tree. The data mining technology creates the subsets recursively until each subset that is a terminal node in the tree represents a unique output class.

[0225] FIG. **22** is a flowchart of an algorithm **2200** for generating the data mining technology to create a decision tree based on similar records in a training table. Sets "S", R, and U are initialized. Set "S" is a set that contains all the records in a training table, set R is the root of the decision tree, and set U is the set of nodes in the tree that are not terminal nodes. Both R and U are initialized to contain all the records in a training table. Next, a first node Ni (containing all the records in the training database) is removed from U. The triplet (field, test, value) that best splits the subset Si associated with the node Ni into two subsets is determined. The triplet that best splits the subset Si is the one that creates the smallest depth tree possible, that is, the triplet would either create one or two terminal nodes, or create two nodes that, when split, would result in a lower number of children nodes than other triplets. The triplet is determined by using an impurity function such as Entropy or the Gini index to find the information conveyed by each field value in the database. The field value that conveys the least degree of information contains the least uncertainty and determines the triplet to be used for splitting the subsets.

[0226] A node Nij is created and associated to the first subset Sij formed. The node Nij is then linked to node Ni, and named with the triplet (field, test, value). Next, a check is performed to evaluate if all the records in subset Sij at node Nij belong to the same output class c_{ij}. If they do, then the prediction of node Nij is set to c_{ij}. If not, then node Nij is added to U. The algorithm then proceeds to check if there are still subsets Sij to be split in the tree, and if so, the algorithm goes back. When all subsets have been associated with nodes, the algorithm continues for the remaining nodes in U until U is determined to be empty.

[0227] FIG. **23** represents a decision tree **2300** in an example for a database **2301** maintained by an insurance company to predict a risk of an insurance contract based on a type of a car and an age of its driver. Database **2301** has three fields: (1) age, (2) car type, and (3) risk. The risk field is the output class that needs to be predicted for any new incoming data record. The age and the car type fields are used as inputs. The data mining technology builds a decision tree, e.g., one that can ease a search of cases in case-based reasoning to determine if an incoming transaction fits any profiles of similar cases existing in its database. The decision tree starts with a root node NO (**2302**). Once the data records in database **2301** are analyzed, a test **2303** is determined that best splits database **2301** into two nodes, a node N**1** (**2304**) with a subset **2305**, and a node N**2** (**2306**) with a subset **2307**. Node N**1** (**2304**) is a terminal node type, since all data records in subset **2305** have the same class output that indicates a high insurance risk for drivers that are younger than twenty-five.

[0228] The data mining technology then splits a node N**2** (**2306**) into two additional nodes, a node N**3** (**2308**) containing a subset **2309**, and a node N**4** (**2310**) containing a subset **2311**. Both nodes N**3** (**2308**) and N**4** (**2310**) were split from node N**2** (**2306**) based on a test **2312**, that checks if the car type is a sports car. As a result, nodes N**3** (**2308**) and N**4** (**2310**) are terminal nodes, with node N**3** (**2308**) signifying a high insurance risk and node N**4** (**2310**) representing a low insurance risk.

[0229] The decision tree formed by the data mining technology is preferably a depth two binary tree, significantly reducing the size of the search problem for the case-based reasoning technology. Instead of searching for similar cases to an incoming data record associated with an electronic transaction in the entire database, the case-based reasoning technology only has to use the predefined index specified by the decision tree.

Case-Based Reasoning Technology

[0230] The case-based reasoning technology stores past data records or cases to identify and classify a new case. It reasons by analogy and classification. Case-based reasoning technologies create a list of generic cases that best represent the cases in its training table. A typical case is generated by computing similarities between all the cases in its training table and selecting those cases that best represent distinct cases. Whenever a new case is presented in a record, a decision tree is to determine if any input record it has on file in its database is similar to something encountered in its training table.

[0231] FIG. **24** is a flowchart of an algorithm for generating a case-based reasoning technology used later to find a record in a database that best resembles an input record corresponding to a new transaction. An input record is propagated through a decision tree according to tests defined for each node in the tree until it reaches a terminal node. If an input record is not fully defined, that is, the input record does not contain values assigned to certain fields, and then the input record is propagated to a last node in a tree that satisfies all the tests. The cases retrieved from this node are all the cases belonging to the node's leaves.

[0232] A similarity measure is computed between the input record and each one of the cases retrieved. The similarity measure returns a value that indicates how close the input record is to a given case retrieved. The case with the highest similarity measure is then selected as the case that best represents the input record. The solution is revised by using a function specified by the user to modify any weights assigned to fields in the database. Finally, the input record is included in the training database and the decision tree is updated for learning new patterns.

[0233] FIG. **25** represents a table **2500** of global similarity measures useful by case-based reasoning technology. The table lists an example of six similarity measures that could be used in case-based reasoning to compute a similarity between cases. The Global Similarity Measure is a computation of the similarity between case values V_{1i} and V_{2i} and are based on

local similarity measures sim_i for each field y_i. The global similarity measures may also employ weights w_i for different fields.

[0234] FIG. **26** is an example table of Local Similarity Measures useful in case-based reasoning. Table **2600** lists fourteen different Local Similarity Measures that is used by the global similarity measures listed. The local similarity measures depend on the field type and valuation. The field type is: (1) symbolic or nominal; (2) ordinal, when the values are ordered; (3) taxonomic, when the values follow a hierarchy; and (4) numeric, which can take discrete or continuous values. The Local Similarity Measures are based on a number of parameters, including: (1) the values of a given field for two cases, V_1 and V_2; (2) the lower (V_1- and V_2-) and higher (V_1+ and V_2+) limits of V_1 and V_2; (3) the set of all values that is reached by the field; (4) the central points of V_1 and V_2, V1c and V2c; (5) the absolute value "ec" of a given interval; and (6) the height "h" of a level in a taxonomic descriptor.

Genetic Algorithms Technology

[0235] Genetic algorithms technologies include a library of genetic algorithms that incorporate biological evolution concepts to find if a class is true, e.g., a business transaction is fraudulent, there is network intrusion, etc. Genetic algorithms is used to analyze many data records and predictions generated by other predictive technologies and recommend its own efficient strategies for quickly reaching a decision.

Rule-Based Reasoning, Fuzzy Logic, and Constraint Programming Technologies

[0236] Rule-based reasoning, fuzzy logic, and constraint programming technologies include business rules, constraints, and fuzzy rules to determine the output class of a current data record, e.g., if an electronic transaction is fraudulent. Such business rules, constraints, and fuzzy rules are derived from past data records in a training database or created from predictable but unusual data records that may arise in the future. The business rules is automatically created by the data mining technology, or they is specified by a user. The fuzzy rules are derived from business rules, with constraints specified by a user that specify which combinations of values for fields in a database are allowed and which are not.

[0237] FIG. **27** represents a rule **2700** for use with the rule-based reasoning technology. Rule **2700** is an IF-THEN rule containing an antecedent and consequence. The antecedent uses tests or conditions on data records to analyze them. The consequence describes the actions to be taken if the data satisfies the tests. An example of rule **2700** that determines if a credit card transaction is fraudulent for a credit card belonging to a single user may include "IF (credit card user makes a purchase at 8 AM in New York City) and (credit card user makes a purchase at 8 AM in Atlanta) THEN (credit card number may have been stolen)". The use of the words "may have been" in the consequence sets a trigger that other rules need to be checked to determine if the credit card transaction is indeed fraudulent or not.

[0238] FIG. **28** represents a fuzzy rule **2800** to specify if a person is tall. Fuzzy rule **2800** uses fuzzy logic to handle the concept of partial truth, e.g., truth values between "completely true" and "completely false" for a person who may or may not be considered tall. Fuzzy rule **2800** contains a middle ground, in addition to the binary patterns of yes/no. Fuzzy rule **2800** derives here from an example rule such as

[0239] "IF height >6 ft., THEN person is tall".

Fuzzy logic derives fuzzy rules by "fuzzification" of the antecedents and "de-fuzzification" of the consequences of business rules.

[0240] FIG. **29** is a flowchart of an algorithm **2900** for applying rule-based reasoning, fuzzy logic, and constraint programming to determine if an electronic transaction is fraudulent. The rules and constraints are specified by a user-service consumer and/or derived by data mining technology. The data record associated with a current electronic transaction is matched against the rules and the constraints to determine which rules and constraints apply to the data. The data is tested against the rules and constraints to determine if the transaction is fraudulent. The rules and constraints are updated to reflect the new electronic transaction.

[0241] The present inventor, Dr. Akli Adjaoute and his Company, Brighterion, Inc. (San Francisco, Calif.), have been highly successful in developing fraud detection computer models and applications for banks, payment processors, and other financial institutions. In particular, these fraud detection computer models and applications are trained to follow and develop an understanding of the normal transaction behavior of single individual accountholders. Such training is sourced from multi-channel transaction training data or single-channel. Once trained, the fraud detection computer models and applications are highly effective when used in real-time transaction fraud detection that comes from the same channels used in training.

[0242] Some embodiments of the present invention train several single-channel fraud detection computer models and applications with corresponding different channel training data. The resulting, differently trained fraud detection computer models and applications are run several in parallel so each can view a mix of incoming real-time transaction message reports flowing in from broad diverse sources from their unique perspectives. One may compute a "hit" the others will miss, and that's the point.

[0243] If one differently trained fraud detection computer model and application produces a hit, it is considered herein a warning that the accountholder has been compromised or has gone rogue. The other differently trained fraud detection computer models and applications should be and are sensitized to expect fraudulent activity from this accountholder in the other payment transaction channels. Hits across all channels are added up and too many is reason to shut down all payment channels for the affected accountholder.

[0244] In general, a method of cross-channel financial fraud protection comprises training a variety of real-time, risk-scoring fraud model technologies with training data selected for each from a common transaction history. This then can specialize each member in the monitoring of a selected channel. After training, the heterogeneous real-time, risk-scoring fraud model technologies are arranged in parallel so that all receive the same mixed channel flow of real-time transaction data or authorization requests.

[0245] Parallel, diversity trained, real-time, risk-scoring fraud model technologies are hosted on a network server platform for real-time risk scoring of a mixed channel flow of real-time transaction data or authorization requests. Risk thresholds are directly updated for particular accountholders in every member of the parallel arrangement of diversity trained real-time, risk-scoring fraud model technologies when any one of them detects a suspicious or outright fraudulent transaction data or authorization request for the accountholder. So, a compromise, takeover, or suspicious

activity of an accountholder's account in any one channel is thereafter prevented from being employed to perpetrate a fraud in any of the other channels.

[0246] Such method of cross-channel financial fraud protection can further include building a population of real-time, long-term, and recursive profiles for each accountholder in each of the real-time, risk-scoring fraud model technologies. Then during real-time use, maintaining and updating the real-time, long-term, and recursive profiles for each accountholder in each and all of the real-time, risk-scoring fraud model technologies with newly arriving data.

[0247] If during real-time use a compromise, takeover, or suspicious activity of the accountholder's account in any one channel is detected, then updating the real-time, long-term, and recursive profiles for each accountholder in each and all of the other real-time, risk-scoring fraud model technologies to further include an elevated risk flag. The elevated risk flags are included in a final risk score calculation **728** for the current transaction or authorization request.

[0248] Fifteen-minute vectors are a way to cross pollenate risks calculated in one channel with the others. The 15-minute vectors can represent an amalgamation or fuzzification of transactions in all channels, or channel-by channel. Once a 15-minute vector has aged, it is shifted into a 100-minute vector, a one-hour vector, and a whole day vector by a simple shift register means. These vectors represent velocity counts that is very effective in catching fraud as it is occurring in real time.

[0249] In every case, embodiments of the present invention include adaptive learning that combines three learning techniques to evolve the artificial intelligence classifiers. First is the automatic creation of profiles, or smart-agents, from historical data, e.g., long-term profiling. The second is real-time learning, e.g., enrichment of the smart-agents based on real-time activities. The third is adaptive learning carried by incremental learning algorithms.

[0250] For example, two years of historical credit card transactions data needed over twenty seven terabytes of database storage. A smart-agent is created for each individual card in that data in a first learning step, e.g., long-term profiling. Each profile is created from the card's activities and transactions that took place over the two year period. Each profile for each smart-agent comprises knowledge extracted field-by-field, such as merchant category code (MCC), time, amount for an mcc over a period of time, recursive profiling, zip codes, type of merchant, monthly aggregation, activity during the week, weekend, holidays, Card not present (CNP) versus card present (CP), domestic versus cross-border, etc. this profile will highlights all the normal activities of the smart-agent (specific payment card).

[0251] Smart-agent technology learns specific behaviors of each cardholder and creates a smart-agent to follow the behavior of each cardholder. Because it learns from each activity of a cardholder, the smart-agent updates its profiles and makes effective changes at runtime. It is the only technology with an ability to identify and stop, in real-time, previously unknown fraud schemes. It has the highest detection rate and lowest false positives because it separately follows and learns the behaviors of each cardholder.

[0252] Smart-agents have a further advantage in data size reduction. Once, say twenty-seven terabytes of historical data is transformed into smart-agents, only 200-gigabytes is needed to represent twenty-seven million distinct smart-agents corresponding to all the distinct cardholders.

[0253] Incremental learning technologies are embedded in the machine algorithms and smart-agent technology to continually re-train from any false positives and negatives that occur along the way. Each corrects itself to avoid repeating the same classification errors. Data mining logic incrementally changes the decision trees by creating a new link or updating the existing links and weights. Neural networks update the weight matrix, and case based reasoning logic updates generic cases or creates new ones. Smart-agents update their profiles by adjusting the normal/abnormal thresholds, or by creating exceptions.

[0254] FIG. **30** represents a flowchart of an algorithm **3000** executed by an apparatus needed to implement a method embodiment of the present invention for improving predictive model training and performance by data enrichment of transaction records.

[0255] The data enrichment of transaction records is done first with supervised and unsupervised training data **124** (FIG. **1**) and training sets **420+422+424**, **421+423+425**, and **440+442+444** (FIG. **4**) during training to build predictive models **127**, **131**, **135**, **139**, **143**, and **147** (FIG. **1**), and **601-606** (FIG. **6**). These are ultimately deployed as predictive models **611-616** (FIG. **6**) for use in real time with a raw feed of new event, non-training data records **906** (FIG. **9**).

[0256] FIG. **30** shows on the left that method **500** (FIG. **5**) includes a step **3001** to delete some data fields not particularly useful, a step **3002** to add some data fields are helpful, a step **3003** to test that the data fields added in step **3002** do improve the final predictions, and a step **3004** to loop until all the original data fields are scrutinized.

[0257] In summary, embodiments of the present invention include a method **3000** of operating an artificial intelligence machine **100** to produce predictive model language documents **128**, **132**, **136**, **140**, **144**, and **148** describing improved predictive models that generate better business decisions **660**, **661** from raw data record inputs **618**. A first phase includes deleting **3001** with at least one processor a selected data field and any data values contained in the selected data field from each of a first series of data records (e.g., training sets **420+422+424**, **421+423+425**, and **440+442+444** [FIG. **4**]) stored in a memory of the artificial intelligence machine to exclude each data field in the first series of data records that has more than a threshold number of random data values, or that has only one repeating data value, or has too small a Shannon entropy, and then transforming a surviving number of data fields in all the first series of data records into a corresponding reduced-field series of data records stored in the memory of the artificial intelligence machine.

[0258] A next phase includes adding **3002** with the at least one processor a new derivative data field to all the reduced-field series of data records stored in the memory of the artificial intelligence machine and initializing each added new derivative data field with a new data value, and including an apparatus for executing an algorithm to either change real scaler numeric data values into fuzzy values, or if symbolic, to change a behavior group data value, and testing that a minimum number of data fields survive, and if not, then to generate a new derivative data field and fix within each an aggregation type, a time range, a filter, a set of aggregation constraints, a set of data fields to aggregate, and a recursive level, and then assessing the quality of a newly derived data field by testing it with a test set of data, and then transforming the results into an enriched-field series of data records stored in the memory of the artificial intelligence machine.

[0259] And a next phase includes verifying **3003** with the at least one processor that a predictive model trained with the enriched-field series of data records stored in the memory of the artificial intelligence machine produces more accurate predictions from the artificial intelligence machine having fewer errors than the same predictive model trained only with the first series of data records.

[0260] Another phase of the method includes verifying with the at least one processor that a predictive model **611-616** fed a non-training set of the enriched-field series of data records **906** stored in the memory of the artificial intelligence machine produces more accurate predictions **660**, **661** with fewer errors than the same predictive model fed with data records with unmodified data fields.

[0261] A still further phase of the method includes recording as a data-enrichment descriptor **3006** and **3008** into the memory of the artificial intelligence machine including the at least one processor an identity of any data fields in a data record format of the first series of data records that were subsequently deleted and can be ignored, and which newly derived data fields were subsequently added, and how each newly derived data field was derived and from which information sources.

[0262] Another phase includes passing along the data-enrichment descriptor with the at least one processor information stored in the memory of the artificial intelligence machine to an artificial intelligence machine including processors for predictive model algorithms to produce and output better business decisions from its own feed of new events as raw data record inputs stored in the memory of the artificial intelligence machine.

[0263] A method **622** (FIG. **6**) of operating an artificial intelligence machine including processors for predictive model algorithms that produces and that outputs better business decisions **660**, **661** from a new series of data records of new events as raw data record inputs **618** and **906**, includes a phase to recover with at least one processor a recording of a data-enrichment descriptor stored in a memory of an artificial intelligence machine including an identity **3006** of any data fields in a data record format of a series of data records that were subsequently deleted by an artificial intelligence machine including processors for predictive model building, and which of any newly derived data fields **3008** were subsequently added, and how each newly derived data field was derived and from which information sources. A next phase includes accepting a new series of data records **906** of new events with the artificial intelligence machine including at least one processor to receive and store records in the memory of the artificial intelligence machine. A next phase of the method **3000** includes ignoring or deleting **3010** with the at least one processor all data fields and all data values contained in the data fields from each of a new series of data records of new events, stored in the memory of the artificial intelligence machine, according to the data-enrichment descriptor **3006**. And in a next phase that includes adding **3011** with the at least one processor a new derivative data field to each record of the new series of data records stored in the memory of the artificial intelligence machine according to the data-enrichment descriptor **3008**, and initializing each added new derivative data field with a new data value stored in the memory of the artificial intelligence machine.

[0264] The method further includes producing and outputting a series of predictive decisions **660**, **661** with the at least one processor that operates at least one predictive model algorithm **611-616** derived from one originally built and trained with records (e.g., training sets **420+422+424**, **421+423+425**, and **440+442+444** [FIG. **4**]) having a same record format described by the data-enrichment descriptor and stored in the memory of the artificial intelligence machine.

[0265] The method excludes each data field stored in the memory of the artificial intelligence machine that has more than a threshold number of random data values, or that has only one repeating data value, or that has too small a Shannon entropy, and then transforming a surviving number of data fields into a corresponding reduced-field series of data records stored in the memory of the artificial intelligence machine.

[0266] The method adds a new derivative data field to a reduced-field series of data records stored in the memory of the artificial intelligence machine and initialize each added new derivative data field with a new data value, and to either change real scaler numeric data values into fuzzy values, or if symbolic, to change a behavior group data value stored in the memory of the artificial intelligence machine, and testing that a minimum number of data fields survive in that stored in the memory of the artificial intelligence machine, and if not, then to generate a new derivative data field and fix within each an aggregation type, a time range, a filter, a set of aggregation constraints, a set of data fields to aggregate, and a recursive level, and which the quality of each newly derived data field was test, and then transforming the results into an enriched-field series of data records stored in the memory of the artificial intelligence machine.

[0267] Although particular embodiments of the present invention have been described and illustrated, such is not intended to limit the invention. Modifications and changes will no doubt become apparent to those skilled in the art, and it is intended that the invention only be limited by the scope of the appended claims.

1. A method of operating an artificial intelligence machine to improve their decisions from included predictive models, comprising:

deleting with at least one processor a selected data field and any data values contained in the selected data field from each of a first series of data training records stored in a memory of the artificial intelligence machine to exclude each data field in the first series of data training records that has more than a threshold number of random data values, or that has only one repeating data value, or that has too small a Shannon entropy, and using any information gained to select the most useful data fields, and then transforming a surviving number of data fields in all the first series of data training records into a corresponding reduced-field series of data training records stored in the memory of the artificial intelligence machine;

adding with the at least one processor a new derivative data field to all the reduced-field series of data training records stored in the memory and initializing each added new derivative data field with a new data value, and including an apparatus for executing an algorithm to either change real scaler numeric data values into fuzzy values, or if symbolic, to change a behavior group data value, and testing that a minimum number of data fields survive, and if not, then to generate a new derivative data field and fix within each an aggregation type, a time range, a filter, a set of aggregation constraints, a set of data fields to aggregate, and a recursive level, and then assessing the quality of a newly derived data field by

testing it with a test set of data, and then transforming the results into an enriched-field series of data training records stored in the memory of the artificial intelligence machine;

verifying with the at least one processor that each predictive model if trained with the enriched-field series of data training records stored in the memory produces decisions having fewer errors than the same predictive model trained only with the first series of data training records;

recording a data-enrichment descriptor into the memory to include an identity of selected data fields in a data training record format of the first series of data training records that were subsequently deleted, and which newly derived data fields were subsequently added, and how each newly derived data field was derived and from which information sources;

causing the at least one processor of the artificial intelligence machine to start extracting decisions from a new series of data records of new events by receiving and storing the new series of data records in the memory of the artificial intelligence machine;

causing the at least one processor to fetch the data-enrichment descriptor and use it to select which data fields to delete and then deleting all the data values included in the selected data fields from each of a new series of data records of new events;

wherein, each data field deleted matches a data field in the first series of data training records had more than a threshold number of random data values, or that had only one repeating data value, or that had too small a Shannon entropy;

adding with the at least one processor a new derivative data field to each record of the new series of data records stored in the memory according to the data-enrichment descriptor, and initializing each added new derivative data field with a new data value stored in the memory;

wherein, each new derivative data field added matches a new derivative data field added to the enriched-field series of data training records in which real scaler numeric data values were changed into fuzzy values, or if symbolic, were changed into a behavior group data value stored in the memory, and were tested that a minimum number of data fields survive, and if not, then that generated a new derivative data field and fixed within each an aggregation type, a time range, a filter, a set of aggregation constraints, a set of data fields to aggregate, and a recursive level; and

producing and outputting a series of predictive decisions with the at least one processor that operates at least one predictive model algorithm derived from one originally built and trained with records having a same record format described by the data-enrichment descriptor and stored in the memory of the artificial intelligence machine.

2. A method of operating an artificial intelligence machine to produce predictive model language documents describing improved predictive models that generate better business decisions from raw data record inputs, comprising:

deleting with at least one processor a selected data field and any data values contained in the selected data field from each of a first series of data records stored in a memory of the artificial intelligence machine to exclude each data field in the first series of data records that has more than a threshold number of random data values, or that has only one repeating data value, or has too small a Shannon entropy, and then transforming a surviving number of data fields in all the first series of data records into a corresponding reduced-field series of data records stored in the memory of the artificial intelligence machine;

adding with the at least one processor a new derivative data field to all the reduced-field series of data records stored in the memory of the artificial intelligence machine and initializing each added new derivative data field with a new data value, and including an apparatus for executing an algorithm to either change real scaler numeric data values into fuzzy values, or if symbolic, to change a behavior group data value, and testing that a minimum number of data fields survive, and if not, then to generate a new derivative data field and fix within each an aggregation type, a time range, a filter, a set of aggregation constraints, a set of data fields to aggregate, and a recursive level, and then assessing the quality of a newly derived data field by testing it with a test set of data, and then transforming the results into an enriched-field series of data records stored in the memory of the artificial intelligence machine; and

verifying with the at least one processor that a predictive model trained with the enriched-field series of data records stored in the memory of the artificial intelligence machine produces more accurate predictions from the artificial intelligence machine having fewer errors than the same predictive model trained only with the first series of data records.

3. The method of claim **2**, further comprising:

verifying with the at least one processor that a predictive model supplied with a non-training set of the enriched-field series of data records stored in the memory of the artificial intelligence machine produces more accurate predictions with fewer errors than the same predictive model fed with data records with unmodified data fields.

4. The method of claim **2**, further comprising:

recording as a data-enrichment descriptor into the memory of the artificial intelligence machine including the at least one processor an identity of any data fields in a data record format of the first series of data records that were subsequently deleted, and which newly derived data fields were subsequently added, and how each newly derived data field was derived and from which information sources; and

passing along the data-enrichment descriptor with the at least one processor information stored in the memory of the artificial intelligence machine to an artificial intelligence machine including processors for predictive model algorithms to produce and output better business decisions from its own feed of new events as raw data record inputs stored in the memory of the artificial intelligence machine.

5. A method of operating an artificial intelligence machine including processors for predictive model algorithms that produces and that outputs better business decisions from a new series of data records of new events as raw data record inputs, comprising:

recovering with at least one processor a recording of a data-enrichment descriptor stored in a memory of the artificial intelligence machine including an identity of any data fields in a data record format of a series of data

records that were subsequently deleted by an artificial intelligence machine including processors for predictive model building, and which of any newly derived data fields were subsequently added, and how each newly derived data field was derived and from which information sources;

accepting a new series of data records of new events with the artificial intelligence machine including at least one processor to receive and store records in the memory of the artificial intelligence machine;

deleting with the at least one processor all data fields and all data values contained in the data fields from each of a new series of data records of new events, stored in the memory of the artificial intelligence machine, according to the data-enrichment descriptor;

adding with the at least one processor a new derivative data field to each record of the new series of data records stored in the memory of the artificial intelligence machine according to the data-enrichment descriptor, and initializing each added new derivative data field with a new data value stored in the memory of the artificial intelligence machine; and

producing and outputting a series of predictive decisions with the at least one processor that operates at least one predictive model algorithm derived from one originally built and trained with records having a same record format described by the data-enrichment descriptor and stored in the memory of the artificial intelligence machine.

6. The method of claim **5** which includes causing the at least one processor in the step of deleting to:

exclude each data field stored in the memory of the artificial intelligence machine that has more than a threshold number of random data values, or that has only one repeating data value, or that has too small a Shannon entropy, and then transforming a surviving number of data fields into a corresponding reduced-field series of data records stored in the memory of the artificial intelligence machine.

7. The method of claim **6** which includes causing the at least one processor in the step of adding to:

add a new derivative data field to a reduced-field series of data records stored in the memory of the artificial intelligence machine and initialize each added new derivative data field with a new data value, and to either change real scaler numeric data values into fuzzy values, or if symbolic, to change a behavior group data value stored in the memory of the artificial intelligence machine, and testing that a minimum number of data fields survive in that stored in the memory of the artificial intelligence machine, and if not, then to generate a new derivative data field and fix within each an aggregation type, a time range, a filter, a set of aggregation constraints, a set of data fields to aggregate, and a recursive level, and which the quality of each newly derived data field was test, and then transforming the results into an enriched-field series of data records stored in the memory of the artificial intelligence machine.

* * * * *