SYSTEMS AND METHODS FOR PREDICTIVE QUERY IMPLEMENTATION AND USAGE IN A MULTI-TENANT DATABASE SYSTEM

Applicants: Beau David Cronin, Oakland, CA (US); Cap Christian Petschulat, San Francisco, CA (US); Eric Michael Jonas, San Francisco, CA (US); Fritz Obermeyer, San Francisco, CA (US)

Inventors: Beau David Cronin, Oakland, CA (US); Cap Christian Petschulat, San Francisco, CA (US); Eric Michael Jonas, San Francisco, CA (US); Fritz Obermeyer, San Francisco, CA (US)

Assignee: SALESFORCE.COM, INC., San Francisco, CA (US)

ABSTRACT

Disclosed herein are systems and methods for predictive query implementation and usage in a multi-tenant database system including means for implementing predictive population of null values with confidence scoring, means for predictive scoring and reporting of business opportunities with probability to close scoring, and other related embodiments.
FIG. 4

Processor Peripheral Device

FIG 4 - 400 PROCESSOR PERIPHERAL DEVICE

Network Interface Card (NIC) Secondary Memory Machine-Accessible Storage Medium Software

Network

Peripherals Device

Alphanumeric Input Device Cursor Control Device

User Interface

Integrated Speaker

Main Memory

Stored Indices

Analysis Engine

PreQL API

Network Interface Card (NIC)

BUS

Secondary Memory Machine-Accessible Storage Medium Software
Define a space of possible outcomes for the hypothesis space. Leverage prior knowledge.

Gather information from available sources.

FIG. 6
Random Walk Method

FIG. 7
<table>
<thead>
<tr>
<th>801 - Features/Characteristics as columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>walks</td>
</tr>
<tr>
<td>flies</td>
</tr>
<tr>
<td>lizards</td>
</tr>
<tr>
<td>snakes</td>
</tr>
<tr>
<td>alligators</td>
</tr>
<tr>
<td>crocodiles</td>
</tr>
<tr>
<td>birds</td>
</tr>
<tr>
<td>fish</td>
</tr>
<tr>
<td>mammals</td>
</tr>
<tr>
<td>reptiles</td>
</tr>
<tr>
<td>amphibians</td>
</tr>
<tr>
<td>insects</td>
</tr>
<tr>
<td>spiders</td>
</tr>
<tr>
<td>plants</td>
</tr>
<tr>
<td>trees</td>
</tr>
<tr>
<td>grasses</td>
</tr>
<tr>
<td>flowers</td>
</tr>
<tr>
<td>algae</td>
</tr>
<tr>
<td>fungi</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>802 - Entities (e.g., mammals) as rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>antelope</td>
</tr>
<tr>
<td>bat</td>
</tr>
<tr>
<td>beaver</td>
</tr>
<tr>
<td>blue whale</td>
</tr>
<tr>
<td>buffalo</td>
</tr>
<tr>
<td>chihuahua</td>
</tr>
<tr>
<td>chimpanzee</td>
</tr>
<tr>
<td>collie</td>
</tr>
<tr>
<td>cow</td>
</tr>
<tr>
<td>damania</td>
</tr>
<tr>
<td>dalmatian</td>
</tr>
<tr>
<td>dolphin</td>
</tr>
<tr>
<td>elephant</td>
</tr>
<tr>
<td>fox</td>
</tr>
<tr>
<td>grizzly</td>
</tr>
<tr>
<td>hamster</td>
</tr>
<tr>
<td>hippopotamus</td>
</tr>
</tbody>
</table>

FIG. 8

803 unobserved cell
804 observed cell
There are lots of partitions, even for small $n$.

<table>
<thead>
<tr>
<th>#Objects</th>
<th>#Partitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
</tr>
<tr>
<td>6</td>
<td>203</td>
</tr>
<tr>
<td>7</td>
<td>877</td>
</tr>
<tr>
<td>8</td>
<td>4140</td>
</tr>
<tr>
<td>9</td>
<td>21147</td>
</tr>
<tr>
<td>10</td>
<td>115975</td>
</tr>
<tr>
<td>11</td>
<td>678570</td>
</tr>
<tr>
<td>12</td>
<td>4213597</td>
</tr>
<tr>
<td>13</td>
<td>27644437</td>
</tr>
<tr>
<td>14</td>
<td>190899322</td>
</tr>
<tr>
<td>15</td>
<td>1382958545</td>
</tr>
<tr>
<td>16</td>
<td>10480142147</td>
</tr>
<tr>
<td>17</td>
<td>82864869804</td>
</tr>
</tbody>
</table>

**FIG. 11**

The Bell Numbers

Logarithmic scatterplot of $A000110(n)$

1103 Quantity of web pages presently indexed by Google.
Receiving a dataset in a tabular form, the dataset having a plurality of rows and a plurality of columns.

Processing the dataset to generate indices representing probabilistic relationships between the rows and the columns of the dataset.

Storing the indices in a database.

Exposing an Application Programming Interface (API) to query the indices in the database.

Receiving a request for a predictive query or a latent structure query against the indices in the database.

Querying the database for a result based on the request via the API.

Returning the result responsive to the request.

End
Exposing an interface to client devices operating remotely from the host organization, wherein the interface is accessible by the client devices via a public Internet.

Executing a predictive database at the host organization as an on-demand cloud based service for one or more subscribers.

Authenticating one of the client devices by verifying the client device is associated with one of the subscribers and based further on authentication credentials for the respective subscriber.

Receiving a request from the authenticated subscriber via the interface.

Executing a predictive query or a latent structure query against indices of the predictive database generated from a dataset of columns and rows on behalf of the authenticated subscriber, the indices representing probabilistic relationships between the rows and the columns of the dataset.

Returning a predictive record set to the authenticated subscriber responsive to the request.
1301: Input (Column ID)

Output: How often does each other column appear with it in the same view?

1302

FIG. 13A

Related
Generating indices from a dataset of columns and rows, the indices representing probabilistic relationships between the rows and the columns of the dataset.

Storing the indices within a database of the host organization.

Exposing the database of the host organization via a request interface.

Receiving, at the request interface, a query for the database specifying a RELATED command term and a specified column as a parameter for the RELATED command term.

Querying the database using the RELATED command term and passing the specified column to generate a predictive record set.

Returning the predictive record set responsive to the query, the predictive record set having a plurality of elements therein, each of the returned elements including a column identifier and a confidence indicator for the specified column passed with the RELATED command term, wherein the confidence indicator indicates whether a latent relationship exists between the specified column passed with the RELATED command and the column identifier returned for the respective element.
Output: Which rows most often appear together as a group in the same categories, in the view containing the input column?

FIG. 14A
Group
FIG. 14C

Start

Generating indices from a dataset of columns and rows, the indices representing probabilistic relationships between the rows and the columns of the dataset.

Storing the indices within a database of the host organization.

Exposing the database of the host organization via a request interface.

Receiving, at the request interface, a query for the database specifying a GROUP command term and a specified column as a parameter for the GROUP command term.

Querying the database using the GROUP command term and passing the specified column to generate a predictive record set.

Returning the predictive record set responsive to the query, the predictive record set having a plurality of groups specified therein, each of the returned groups of the predictive record set including a group of one or more rows of the dataset.

End
1501: Input (Row ID and Column ID)

Output: How often does each row appear in the same category as the input row, in the view containing the input column?

FIG. 15A

Similar
FIG. 15C

1. Generating indices from a dataset of columns and rows, the indices representing probabilistic relationships between the rows and the columns of the dataset.

2. Storing the indices within a database of the host organization.

3. Exposing the database of the host organization via a request interface.

4. Receiving, at the request interface, a query for the database specifying a SIMILAR command term, a specified row as a parameter for the SIMILAR command term, and a specified column as a parameter for the SIMILAR command term.

5. Querying the database using the SIMILAR command term and passing the specified row and the specified column to generate a predictive record set.

6. Returning the predictive record set responsive to the query, the predictive record set having a plurality of elements therein, each of the returned elements of the predictive record set including (i) a row identifier which corresponds to a row of the dataset assessed to be similar, according to a latent structure, to the specified row passed with the SIMILAR command term based on the specified column and (ii) a confidence indicator which indicates a likelihood of a latent relationship between the specified row passed with the SIMILAR command and the row identifier returned for the respective element.

End
Classification / Regression

Query

Predictive Samples

Fix a subset of the columns 1601: Input

PREDICT a single target column 1602: Output

FIG. 16A

Predict
Fill-in-the-blanks

Query: [□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □]
Predictive Samples:

Fix a subset of the columns 1611: Input
PREDICT the remaining columns 1612: Output

FIG. 16B
Predict
Synthetic Data Generation

Query

Predictive Samples

Fix none of the columns ~ 1621: Input

PREDICT all of the columns ~ 1622: Output

FIG. 16C
FIG. 16E

Start

Generating indices from a dataset of columns and rows, the indices representing probabilistic relationships between the rows and the columns of the dataset.

Storing the indices within a database of the host organization.

Exposing the database of the host organization via a request interface.

Receiving, at the request interface, a query for the database specifying at least (i) a PREDICT command term, (ii) one or more specified columns to be predicted, and (iii) one or more column name=value pairs specifying column names to be fixed and the values by which to fix them.

Querying the database using the PREDICT command term and passing the one or more specified columns to be predicted and the one or more column name=value pairs to generate a representation of a joint conditional distribution of the one or more specified columns to be predicted fixed according to the column name=value pairs using the indices stored in the database.

End
Start

Generating indices from a dataset of columns and rows, the indices representing probabilistic relationships between the rows and the columns of the dataset.

Storing the indices within a database of the host organization.

Exposing the database of the host organization via a request interface.

Receiving, at the request interface, a query for the database specifying a PREDICT command term and one or more specified columns to be passed with the PREDICT command term.

Querying the database using the PREDICT command term and the one or more specified columns to generate output, wherein the output comprises generated synthetic data having one or more entity rows with predicted values for every column of the dataset using the indices stored in the database.

End
<table>
<thead>
<tr>
<th>Category</th>
<th>Alkalinity</th>
<th>Phenols</th>
<th>Alcohol</th>
<th>Color</th>
<th>Malic acid</th>
<th>OK200</th>
<th>Proanthocyanine</th>
<th>Ash</th>
<th>Proline</th>
<th>Magnesium</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>18.5</td>
<td>1.45</td>
<td>12.0</td>
<td></td>
<td>1.30</td>
<td>1.00</td>
<td>4.56</td>
<td>1.98</td>
<td>1.09</td>
<td>2.34</td>
</tr>
<tr>
<td>3</td>
<td>18.5</td>
<td>13.5</td>
<td>3.50</td>
<td>1.60</td>
<td>0.00</td>
<td>1.00</td>
<td>4.56</td>
<td>1.98</td>
<td>1.09</td>
<td>2.34</td>
</tr>
<tr>
<td>2</td>
<td>18.5</td>
<td>4.00</td>
<td>1.17</td>
<td>3.42</td>
<td>2.00</td>
<td>1.94</td>
<td>1.62</td>
<td>1.62</td>
<td>2.26</td>
<td>1.62</td>
</tr>
<tr>
<td>1</td>
<td>13.2</td>
<td>2.50</td>
<td>2.45</td>
<td></td>
<td>1.62</td>
<td>7.35</td>
<td>3.27</td>
<td>14.2</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>1</td>
<td>13.2</td>
<td>2.50</td>
<td>2.45</td>
<td></td>
<td>1.62</td>
<td>7.35</td>
<td>3.27</td>
<td>14.2</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>1</td>
<td>17.1</td>
<td>3.00</td>
<td>12.8</td>
<td>6.00</td>
<td>1.52</td>
<td>2.05</td>
<td>2.56</td>
<td>1.50</td>
<td>1.18</td>
<td>1.18</td>
</tr>
<tr>
<td>1</td>
<td>2.10</td>
<td>14.0</td>
<td></td>
<td></td>
<td>2.05</td>
<td>2.10</td>
<td>5.10</td>
<td>1.50</td>
<td>1.18</td>
<td>1.18</td>
</tr>
<tr>
<td>3</td>
<td>24.0</td>
<td>12.8</td>
<td>4.32</td>
<td>1.30</td>
<td>1.52</td>
<td>2.05</td>
<td>2.56</td>
<td>1.50</td>
<td>1.18</td>
<td>1.18</td>
</tr>
<tr>
<td>1</td>
<td>12.4</td>
<td>1.93</td>
<td>4.50</td>
<td>1.25</td>
<td>2.02</td>
<td>1.04</td>
<td>1.04</td>
<td></td>
<td></td>
<td>1.04</td>
</tr>
<tr>
<td>3</td>
<td>21.0</td>
<td>2.30</td>
<td>12.4</td>
<td>7.56</td>
<td>1.83</td>
<td>1.04</td>
<td>1.04</td>
<td></td>
<td></td>
<td>1.04</td>
</tr>
<tr>
<td>2</td>
<td>22.0</td>
<td>3.00</td>
<td>1.01</td>
<td>1.78</td>
<td>2.42</td>
<td>2.42</td>
<td>2.42</td>
<td></td>
<td></td>
<td>2.42</td>
</tr>
<tr>
<td>2</td>
<td>16.0</td>
<td>2.10</td>
<td>13.8</td>
<td>5.00</td>
<td>2.45</td>
<td>1.30</td>
<td>1.30</td>
<td></td>
<td></td>
<td>1.30</td>
</tr>
<tr>
<td>2</td>
<td>24.0</td>
<td>1.10</td>
<td>12.8</td>
<td>2.50</td>
<td>1.04</td>
<td>0.50</td>
<td>2.40</td>
<td></td>
<td></td>
<td>2.40</td>
</tr>
<tr>
<td>2</td>
<td>1.50</td>
<td>14.6</td>
<td>2.00</td>
<td>2.83</td>
<td>1.10</td>
<td>1.98</td>
<td>1.98</td>
<td></td>
<td></td>
<td>1.98</td>
</tr>
<tr>
<td>14.5</td>
<td>3.00</td>
<td>13.2</td>
<td>2.17</td>
<td>2.30</td>
<td>1.81</td>
<td>1.00</td>
<td>2.43</td>
<td></td>
<td></td>
<td>2.43</td>
</tr>
<tr>
<td>1</td>
<td>7.90</td>
<td>3.43</td>
<td>2.16</td>
<td>2.50</td>
<td>1.40</td>
<td>113</td>
<td>113</td>
<td></td>
<td></td>
<td>113</td>
</tr>
<tr>
<td>1</td>
<td>18.1</td>
<td>3.80</td>
<td>1.97</td>
<td>2.64</td>
<td>1.09</td>
<td>1.00</td>
<td>1.00</td>
<td></td>
<td></td>
<td>1.00</td>
</tr>
</tbody>
</table>

FIG. 17A
1701 GUI

1703 Known Value

1708 Predicted Value

<table>
<thead>
<tr>
<th>Category</th>
<th>Alkalinity</th>
<th>Phenols</th>
<th>Alcohol</th>
<th>Color</th>
<th>Malic acid</th>
<th>OK200 _00315</th>
<th>Flavonoids</th>
<th>Proanthocyanin</th>
<th>Ash</th>
<th>Prolin</th>
<th>Magnesium</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>18.6</td>
<td>1.46</td>
<td>12.0</td>
<td>3.00</td>
<td>1.35</td>
<td>1.35</td>
<td>1.58</td>
<td>1.00</td>
<td>564</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>18.5</td>
<td>2.01</td>
<td>13.5</td>
<td>3.50</td>
<td>1.60</td>
<td>0.600</td>
<td>0.005</td>
<td>643</td>
<td>187</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>18.5</td>
<td>1.66</td>
<td>13.0</td>
<td>3.50</td>
<td>1.60</td>
<td>0.600</td>
<td>0.005</td>
<td>643</td>
<td>187</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12.2</td>
<td>1.05</td>
<td>1.30</td>
<td>0.520</td>
<td>1.62</td>
<td>2.28</td>
<td>647</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3.27</td>
<td>14.2</td>
<td>0.75</td>
<td>1.78</td>
<td>2.65</td>
<td>2.60</td>
<td>1.52</td>
<td>2.43</td>
<td>1400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3.27</td>
<td>4.97</td>
<td>2.50</td>
<td>2.45</td>
<td>2.06</td>
<td>1.62</td>
<td>735</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>17.1</td>
<td>3.60</td>
<td>12.9</td>
<td>6.00</td>
<td>1.52</td>
<td>2.58</td>
<td>118</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2.10</td>
<td>14.0</td>
<td>5.62</td>
<td>2.07</td>
<td>2.05</td>
<td>1165</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>24.9</td>
<td>1.58</td>
<td>12.8</td>
<td>4.32</td>
<td>1.30</td>
<td>1.73</td>
<td>0.771</td>
<td>560</td>
<td>101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>12.4</td>
<td>1.93</td>
<td>4.50</td>
<td>1.25</td>
<td>2.02</td>
<td>1.16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>21.0</td>
<td>2.30</td>
<td>12.4</td>
<td>7.56</td>
<td>1.63</td>
<td>1.04</td>
<td>0.620</td>
<td>663</td>
<td>56.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>22.0</td>
<td>3.09</td>
<td>1.01</td>
<td>1.79</td>
<td>2.44</td>
<td>336</td>
<td>90.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12.5</td>
<td>2.12</td>
<td>1.67</td>
<td>2.93</td>
<td>300</td>
<td>120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>18.0</td>
<td>2.10</td>
<td>13.7</td>
<td>5.00</td>
<td>2.46</td>
<td>1.38</td>
<td>690</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>24.9</td>
<td>1.10</td>
<td>12.6</td>
<td>2.39</td>
<td>1.72</td>
<td>1.04</td>
<td>0.500</td>
<td>248</td>
<td>96.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1.50</td>
<td>14.0</td>
<td>2.99</td>
<td>2.83</td>
<td>1.10</td>
<td>1.56</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>16.5</td>
<td>13.2</td>
<td>5.90</td>
<td>2.17</td>
<td>2.50</td>
<td>1.00</td>
<td>2.43</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>13.8</td>
<td>7.30</td>
<td>3.43</td>
<td>3.04</td>
<td>2.16</td>
<td>2.50</td>
<td>1400</td>
<td>113</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>18.1</td>
<td>3.69</td>
<td>12.3</td>
<td>8.90</td>
<td>1.07</td>
<td>2.64</td>
<td>3.00</td>
<td>1.00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIG. 17B
<table>
<thead>
<tr>
<th>Category</th>
<th>Alkalinity</th>
<th>Phenols</th>
<th>Alcohol</th>
<th>Color</th>
<th>Malic acid</th>
<th>OK200_00315</th>
<th>Flavanoids</th>
<th>Proanthocyanidine</th>
<th>Ash</th>
<th>Prolene</th>
<th>Magnesium</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>18.6</td>
<td>1.45</td>
<td>12.6</td>
<td>3.00</td>
<td>2.11</td>
<td>2.41</td>
<td>1.35</td>
<td>1.38</td>
<td>1.03</td>
<td>564</td>
<td>94.8</td>
</tr>
<tr>
<td>1</td>
<td>16.6</td>
<td>2.01</td>
<td>13.4</td>
<td>1.68</td>
<td>1.30</td>
<td>1.69</td>
<td>4.65</td>
<td>1.06</td>
<td>2.22</td>
<td>902</td>
<td>101</td>
</tr>
<tr>
<td>3</td>
<td>18.6</td>
<td>1.66</td>
<td>13.5</td>
<td>7.50</td>
<td>9.69</td>
<td>1.60</td>
<td>0.600</td>
<td>0.006</td>
<td>2.49</td>
<td>643</td>
<td>96.6</td>
</tr>
<tr>
<td>2</td>
<td>18.6</td>
<td>2.11</td>
<td>12.2</td>
<td>4.00</td>
<td>1.17</td>
<td>3.42</td>
<td>2.09</td>
<td>1.34</td>
<td>1.92</td>
<td>513</td>
<td>93.8</td>
</tr>
<tr>
<td>3</td>
<td>21.7</td>
<td>1.62</td>
<td>12.2</td>
<td>10.5</td>
<td>2.00</td>
<td>1.50</td>
<td>0.820</td>
<td>1.52</td>
<td>2.26</td>
<td>647</td>
<td>101</td>
</tr>
</tbody>
</table>

1701 GUI

1703 Known Value

1708 Predicted Value

**FIG. 17C**
Receiving a tabular dataset from a user as input, the tabular dataset having data values organized as columns and rows.

Identifying a plurality of null values within the tabular dataset, the null values being dispersed across multiple rows and multiple columns of the tabular dataset.

Generating indices from the tabular dataset of columns and rows, the indices representing probabilistic relationships between the rows and the columns of the tabular dataset.

Displaying the tabular dataset as output to the user, the displayed output including the data values depicted as known values and the null values depicted as unknown values.

Receiving input from the user to populate at least a portion of the unknown values within the displayed tabular dataset with predicted values.

Querying the indices for the predicted values.

Displaying the predicted values as updated output to the user.

End
Feature Move 1801

Entity Move 1802

View 1 1810
View 2 1811
View 3 1812

Existing 1820

or

New View 1821

or

New View 1821

Feature Being Moved 1805

Category 1 1825

Entity Being Moved 1806

Category 2 1826

Category 3 1827

New View 4 1813

FIG. 18
Feature and Entity Moves Method
### Opportunity Detail

- **Opportunity Name**: Kent Pharma - $240k Capsule/Wrap
- **Amount**: $240,000.00
- **Close Date**: 1/31/2013
- **Stage**: Proposal/Price Quote
- **Probability (%)**: 75%
- **Opportunity Score**: 42

#### Description

Description: Potential big win
Next Step: Quote sent out, waiting for response

#### Opportunity Team

<table>
<thead>
<tr>
<th>Action</th>
<th>Team Member</th>
<th>Member Role</th>
<th>Split Type</th>
<th>Split %</th>
<th>Split Amount</th>
<th>Split Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit/Del</td>
<td>Scott</td>
<td>Account Manager</td>
<td>Revenue</td>
<td>100</td>
<td>$240,000.00</td>
<td></td>
</tr>
<tr>
<td>Edit/Del</td>
<td>Amy</td>
<td>Partner</td>
<td></td>
<td>0</td>
<td>$0.00</td>
<td></td>
</tr>
</tbody>
</table>

**FIG. 19C**
Receiving input from a user device specifying a dataset of sales data for a customer organization, wherein the sales data specifies a plurality of sales opportunities.

Generating indices from rows and columns of the dataset, the indices representing probabilistic relationships between the rows and the columns of the dataset.

Storing the indices in a queryable database within the host organization.

Selecting one or more of the plurality of sales opportunities specified within the sales data.

Querying the indices for a win or lose predictive result for each of the selected one or more sales opportunities.

Displaying the win or lose predictive result for each of the selected one or more sales opportunities to the user device as output.
FIG. 20B
Waterfall Chart using predictive data

Start Stage 1 Stage 2 Stage 3 Stage 4 Stage 5 Stage 6 Stage 7 Stage 8 End

Opportunity Count

2005

2
1
1
3
1
1
1
4
FIG. 20C
Defaults after adding historical field filters

Filters
Show
Date Field
History
Add

- Historical Field Filter
  e.g., Amount (historical) greater than 1,000,000

- Field Filter
  e.g., Account Name equals Acme

- Filter Logic
  e.g., Filter 1 AND (Filter 2 or Filter 3)

- Cross Filter
  e.g., Accounts with or without Opportunities

- Row Limit
  e.g., Top 5 Accounts by Annual Revenue

- Help me choose...
Defaults for Added Custom Filter

<table>
<thead>
<tr>
<th>Filters</th>
<th>Add ▼</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show</td>
<td>All opportunities ▼</td>
</tr>
<tr>
<td>Date Field</td>
<td>Close Date ▼</td>
</tr>
<tr>
<td>Range</td>
<td>Close Date ▼</td>
</tr>
<tr>
<td>From</td>
<td>▫ To ▫</td>
</tr>
<tr>
<td>History</td>
<td>Monthly ▼</td>
</tr>
<tr>
<td>Range</td>
<td>Monthly ▼</td>
</tr>
<tr>
<td>From 11/1/2012</td>
<td>▫ To ▫</td>
</tr>
<tr>
<td>Amount (Historical) ▼</td>
<td>▼</td>
</tr>
<tr>
<td>Any Selected History ▼</td>
<td>equals ▼</td>
</tr>
<tr>
<td>▼ Amount ▼</td>
<td>Ok ▼</td>
</tr>
<tr>
<td>▼ Field ▼</td>
<td>Cancel ▼</td>
</tr>
</tbody>
</table>

- When in "Field" mode, only current values appear in the picker.

FIG. 20E
Storing records in a database, wherein updates to the records are recorded into a historical trending data object to maintain historical values for the records when the records are updated in the database.

Receiving input from a user device specifying data to be displayed at the user device.

Receiving historical filter input from the user device.

Querying the records stored in the database for the data to be displayed.

Querying the historical trending data object for the historical values of the data to be displayed.

Comparing the data to be displayed with the historical values of the data to be displayed to determine one or more changed values corresponding to the data to be displayed.

Displaying a change value indication GUI to the user device displaying at least the data to be displayed and a changed value indication based on the one or more changed values determined via the comparing.
2107 - Sales prediction 80% accurate/confident for > 50% of opportunities.

FIG. 21A
Completeness vs. Accuracy
FIG. 21B
Opportunity Confidence Breakdown

2014 Highly likely to close a sale
2013 Highly unlikely to close a sale
2012 - Probability of Sale

# Opportunities

0.0 0.2 0.4 0.6 0.8 1.0
FIG. 21C
Opportunity Win Prediction

ROC - 10k

ROC curve (area = 0.93)
Predictive Currency Conditioned on "IS WON = True|False" Field

2206 "IS WON True|False" by source.

2207 - Sales Lead Sources

2211 50/50 chance to win sale

2212 - Less likely to win sale

2213 - Highly likely to win sale

FIG. 22A
Predictive Relationships for Opportunity Scoring
Opportunity Type Conditioned on "IS WON = True|False" Field

2223 - More likely to win sale

2221 "IS WON True|False" by type.

2222 - Sales leads types

2224 - Less likely to win sale

FIG. 22B
Predictive Relationships for Opportunity Scoring
"IS WON True|False" by currency.

Predicted Currency Conditioned on "IS WON = True|False" Field

2232 - Sales lead by Currency

2236 Sales win more likely in Japan

2237 Sales win less likely in the EU

FIG. 22C
Predictive Relationships for Opportunity Scoring
SYSTEMS AND METHODS FOR PREDICTIVE QUERY IMPLEMENTATION AND USAGE IN A MULTI-TENANT DATABASE SYSTEM

CLAIM OF PRIORITY

This application is related to, and claims priority to, the provisional utility application entitled "SYSTEMS AND METHODS FOR PREDICTIVE QUERY IMPLEMENTATION AND USAGE IN A MULTI-TENANT DATABASE SYSTEM," filed on Mar. 13, 2013, having an application number of 61/780,503 and attorney docket No. 8956P119Z (526P10V), the entire contents of which are incorporated herein by reference.

COPYRIGHT NOTICE

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

TECHNICAL FIELD

Embodiments of the invention relate generally to the field of computing, and more particularly, to systems and methods for predictive query implementation and usage in a multi-tenant database system including means for implementing predictive population of null values with confidence scoring, means for predictive scoring and reporting of business opportunities with probability to close scoring, and other related embodiments.

BRIEF DESCRIPTION OF THE DRAWINGS

Embodiments are illustrated by way of example, and not by way of limitation, and will be more fully understood with reference to the following detailed description when considered in connection with the figures in which:

FIG. 1 depicts an exemplary architecture in accordance with described embodiments;
FIG. 2 illustrates a block diagram of an example of an environment in which an on-demand database service might be used;
FIG. 3 illustrates a block diagram of an embodiment of elements of FIG. 2 and various possible interconnections between these elements;
FIG. 4 illustrates a diagrammatic representation of a machine in the exemplary form of a computer system, in accordance with one embodiment;
FIG. 5A depicts a tablet computing device and a hand-held smartphone each having a circuitry integrated therein as described in accordance with the embodiments;
FIG. 5B is a block diagram of an embodiment of tablet computing device, a smart phone, or other mobile device in which touchscreen interface connectors are used;
FIG. 6 is a simplified flow for probabilistic modeling;
FIG. 7 illustrates an exemplary landscape upon which a random walk may be performed;
FIG. 8 depicts an exemplary tabular dataset;
FIG. 9 depicts means for deriving motivation or causal relationships between observed data;
FIG. 10A depicts an exemplary cross-categorization in still further detail;
FIG. 10B depicts an assessment of convergence, showing inferred versus ground truth;
FIG. 11 depicts a chart and graph of the Bell number series;
FIG. 12A depicts an exemplary cross categorization of a small tabular dataset;
FIG. 12B depicts an exemplary architecture having implemented data upload, processing, and predictive query API exposure in accordance with described embodiments;
FIG. 12C is a flow diagram illustrating a method for implementing data upload, processing, and predictive query API exposure in accordance with disclosed embodiments;
FIG. 12D depicts an exemplary architecture having implemented predictive query interface as a cloud service in accordance with described embodiments;
FIG. 12E is a flow diagram illustrating a method for implementing predictive query interface as a cloud service in accordance with disclosed embodiments;
FIG. 13A illustrates usage of the RELATED command term in accordance with the described embodiments;
FIG. 13B depicts an exemplary architecture in accordance with described embodiments;
FIG. 13C is a flow diagram illustrating a method in accordance with disclosed embodiments;
FIG. 14A illustrates usage of the GROUP command term in accordance with the described embodiments;
FIG. 14B depicts an exemplary architecture in accordance with described embodiments;
FIG. 14C is a flow diagram illustrating a method in accordance with disclosed embodiments;
FIG. 15A illustrates usage of the SIMILAR command term in accordance with the described embodiments;
FIG. 15B depicts an exemplary architecture in accordance with described embodiments;
FIG. 15C is a flow diagram illustrating a method in accordance with disclosed embodiments;
FIG. 16A illustrates usage of the PREDICT command term in accordance with the described embodiments;
FIG. 16B illustrates usage of the PREDICT command term in accordance with the described embodiments;
FIG. 16C illustrates usage of the PREDICT command term in accordance with the described embodiments;
FIG. 16D depicts an exemplary architecture in accordance with described embodiments;
FIG. 16E is a flow diagram illustrating a method in accordance with disclosed embodiments;
FIG. 16F depicts an exemplary architecture in accordance with described embodiments;
FIG. 16G is a flow diagram illustrating a method in accordance with disclosed embodiments;
FIG. 17A depicts a Graphical User Interface (GUI) to display and manipulate a tabular dataset having missing values by exploiting a PREDICT command term;
FIG. 17B depicts another view of the Graphical User Interface;
FIG. 17C depicts another view of the Graphical User Interface;
FIG. 17D depicts an exemplary architecture in accordance with described embodiments;
FIG. 17E is a flow diagram illustrating a method in accordance with disclosed embodiments;
FIG. 18 depicts feature moves and entity moves within indices generated from analysis of tabular datasets;

FIG. 19A depicts a specialized GUI to query using historical dates;

FIG. 19B depicts an additional view of a specialized GUI to query using historical dates;

FIG. 19C depicts another view of a specialized GUI to configure predictive queries;

FIG. 19D depicts an exemplary architecture in accordance with described embodiments;

FIG. 19E is a flow diagram illustrating a method in accordance with disclosed embodiments;

FIG. 20A depicts a pipeline change report in accordance with described embodiments;

FIG. 20B depicts a waterfall chart using predictive data in accordance with described embodiments;

FIG. 20C depicts an interface with defaults after adding a first historical field;

FIG. 20D depicts in additional detail an interface with defaults for an added custom filter;

FIG. 20E depicts another interface with defaults for an added custom filter;

FIG. 20F depicts an exemplary architecture in accordance with described embodiments;

FIG. 20G is a flow diagram illustrating a method in accordance with disclosed embodiments;

FIG. 21A provides a chart depicting prediction completeness versus accuracy;

FIG. 21B provides a chart depicting an opportunity confidence breakdown;

FIG. 21C provides a chart depicting an opportunity win prediction;

FIG. 22A provides a chart depicting predictive relationships for opportunity scoring;

FIG. 22B provides another chart depicting predictive relationships for opportunity scoring;

FIG. 22C provides another chart depicting predictive relationships for opportunity scoring.

BACKGROUND

The subject matter discussed in the background section should not be assumed to be prior art merely as a result of its mention in the background section. Similarly, a problem mentioned in the background section or associated with the subject matter of the background section should not be assumed to have been previously recognized in the prior art. The subject matter in the background section merely represents different approaches, which in and of themselves may also correspond to embodiments of the claimed inventions.

Client organizations with datasets in their databases can benefit from predictive analysis. Unfortunately, there is no low cost and scalable solution in the marketplace today. Instead, client organizations must hire technical experts to develop customized mathematical constructs and predictive models which are very expensive. Consequently, client organizations without vast financial means are simply priced out of the market and thus do not have access to predictive analysis capabilities for their datasets.

The present state of the art may therefore benefit from methods, systems, and apparatuses for predictive query implementation and usage in a multi-tenant database system as described herein.

DETAILED DESCRIPTION

Users wanting to perform predictive analytics and data mining against their datasets must normally hire technical experts and explain the problem they wish to solve and then turn their data over to the hired experts to apply specialized mathematical constructs in an attempt to solve the problem.

By analogy, many years ago when you designed a computer system it was necessary to also figure out how to put data on a physical disk. Now programmers do not concern themselves with such issues. Similarly, it is highly desirable to utilize a server and sophisticated database technology to perform data analytics for ordinary users without having to hire specialized experts. By doing so, resources could be freed up to focus on other problems.

Some machine learning capabilities exist today. For instance, present capabilities can answer questions such as: “Is this person going to buy product x?” But such simplistic technology is not sufficient for helping people to solve more complex problems. For instance, Kaiser Healthcare corporation with vast financial resources may be able to hire experts from KXEN to develop customized analytics to solve a Kaiser specific problem based on Kaiser’s database, but a small company by contrast simply cannot afford to utilize KXEN’s services as the cost far outweighs a small company’s financial resources to do so. Thus, our exemplary small company would be forced to simply forget solving the problem at hand.

Consider KXEN’s own value proposition from their homepage which states: “As a business analyst, you don’t want to worry about complicated math or which algorithm to use. You need a model that is going to predict possible business outcomes. Is this customer likely to churn? Will they respond to a cross-sell or up-sell offer? . . . We’ll help you quickly get to the right algorithm for your business problem with a model built for accuracy and optimal results.”

If a small company lacks the financial resources to hire a company such as KXEN and lacks the technical know how to develop the “complicated math or [select] which algorithm to use,” then such company must go without.

Further still, the services offered today by technical experts in the field of analytics and predictive modeling provide solutions that are customized to the particular dataset of the customer. They do not offer capabilities that may be used by non-experts in an agnostic manner that is not anchored to a particular underlying dataset.

Veritable offers a predictive database and additional commands and verbs so that a non-expert user can query the predictive database with inquiries such as: “predict revenue from users where age is greater than 35.”

Further still, companies that hire analytics and predictive modeling are given a solution at adheres to the present data in their database, but do not adapt to changes in the data or the database structure over time. Thus, a large company may hire KXEN and math experts come in and stare at the data and build models, and so forth, and the models do work, but when the nature of the data changes over time within the database and layout of the data or the database’s structure changes over time, as is normal and common for businesses, then the models stop working as they were customized for the particular data and database structure at a given point in time.

Because Veritable provides a predictive database that is not anchored to any particular underlying dataset, it remains useful as data and data structures change over time. For instance, data analysis performed by the Veritable core
may simply be re-applied to a changed dataset. There is no need to re-hire experts or re-tool the models.

[0075] With respect to salesforce.com specifically, the company offers cloud services to clients, organizations, and end users, and behind those cloud services is a multi-tenant database system which permits users to have customized data, customized field types, and so forth. The underlying data and data structures are customized by the client organizing for their own particular needs. Verifiable may nevertheless be utilized on these varying datasets and data structures because it is not anchored to a particular underlying database scheme, structure, or content.

[0076] Customer organizations further benefit from the low cost of access. For instance, the cloud service provider may elect to provide the capability as part of an overall service offering at no additional cost, or may elect to provide the additional capabilities for an additional service fee. Regardless, because the Verifiable capabilities are systematically integrated into the cloud service's computing architecture and do not require experts to custom tailor a solution to each particular client organizations' dataset and structure, the scalability brings massive cost savings, thus enabling our exemplary small company with limited financial resources to go from a 0% capability because they cannot afford to hire technical experts from KXEN to, for instance, a 95% accuracy capability using Verifiable. Even if a large company with sufficient financial resources could feasibly hire KXEN to develop customized mathematical constructs and models, they would need to evaluate the ROI of hiring KXEN which may be able to get, for instance, 97% accuracy through customization but at high cost, versus using the turn-key access to the low cost cloud computing service which yields the exemplary 95% accuracy.

[0077] Regardless of the decision for a large company with financial means, a small company which would otherwise not have access to predictive analytic capabilities can benefit greatly as their capability for predictive analysis accuracy goes from 0% (e.g., mere guessing) to the exemplary 95% using the scalable architecture provided by Verifiable.

[0078] In the following description, numerous specific details are set forth such as examples of specific systems, languages, components, etc., in order to provide a thorough understanding of the various embodiments. It will be apparent, however, to one skilled in the art that these specific details need not be employed to practice the embodiments disclosed herein. In other instances, well known materials or methods have not been described in detail in order to avoid unnecessarily obscuring the disclosed embodiments.

[0079] In addition to various hardware components depicted in the figures and described herein, embodiments further include various operations which are described below. The operations described in accordance with such embodiments may be performed by hardware components or may be embodied in machine-executable instructions, which may be used to cause a general-purpose or special-purpose processor programmed with the instructions to perform the operations. Alternatively, the operations may be performed by a combination of hardware and software.

[0080] Embodiments also relate to an apparatus for performing the operations disclosed herein. This apparatus may be specially constructed for the required purposes, or it may be a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions, each coupled to a computer system bus.

[0081] The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these systems will appear as set forth in the description below. In addition, embodiments are not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the embodiments as described herein.

[0082] Embodiments may be provided as a computer program product, or software, that may include a machine-readable medium having stored thereon instructions, which may be used to program a computer system (or other electronic devices) to perform a process according to the disclosed embodiments. A machine-readable medium includes any mechanism for storing or transmitting information in a form readable by a machine (e.g., a computer). For example, a machine-readable (e.g., computer-readable) medium includes a machine (e.g., a computer) readable storage medium (e.g., read only memory ("ROM"), random access memory ("RAM"), magnetic disk storage media, optical storage media, flash memory devices, etc.), a machine (e.g., computer) readable transmission medium (electrical, optical, acoustical), etc.

[0083] Any of the disclosed embodiments may be used alone or together with one another in any combination. Although various embodiments may have been partially motivated by deficiencies with conventional techniques and approaches, some of which are described or alluded to within the specification, the embodiments need not necessarily address or solve any of these deficiencies, but rather, may address only some of the deficiencies, address none of the deficiencies, or be directed toward different deficiencies and problems where are not directly discussed.

[0084] In one embodiment, means for predictive query implementation and usage in a multi-tenant database system execute an application in a computing device, a computing system, or a computing architecture, in which the application is enabled to communicate with a remote computing device over a public Internet, such as remote clients, thus establishing a cloud-based computing service in which the clients utilize the functionality of the remote application which implements the predictive query and usage capabilities.

[0085] Model-based clustering techniques, including inference in Dirichlet process mixture models, have difficulty when different dimensions are best explained by very different clusterings. Based on MCMC inference in a novel non-parametric Bayesian model, methods automatically discover the number of independent nonparametric Bayesian models needed to explain the data, using a separate Dirichlet process mixture model for each group in an inferred partition of the dimensions. Unlike a DP mixture, the disclosed model is exchangeable over both the rows of a heterogeneous data array (the samples) and the columns (new dimensions), and
can model any dataset as the number of samples and dimensions both go to infinity. Efficiency and robustness is improved through use of algorithms described herein which in certain instances require no preprocessing to identify vertical causal structure in provided raw datasets.

Clustering techniques are widely used in data analysis for problems of segmentation in industry, exploratory analysis in science, and as a preprocessing step to improve performance of further processing in distributed computing and in data compression. However, as datasets grow larger and noisier, the assumption that a single clustering or distribution over clusterings can account for all the variability in the observations becomes less realistic if not wholly infeasible.

From a machine learning perspective, this is an unsupervised version of the feature selection problem: different subsets of measurements should, in general, induce different natural clusterings of the data. From a cognitive science and artificial intelligence perspective, this issue is reflected in work that seeks multiple representations of data instead of a single monolithic representation.

As a limiting case, a robust clustering method should be able to ignore an infinite number of uniformly random or perfectly deterministic measurements. The assumption that a single nonparametric model must explain all the dimensions is partly responsible for the accuracy issues Dirichlet process mixtures often encounter in high dimensional settings. DP mixture based classifiers via class conditional density estimation highlight the problem. For instance, while a discriminative classifier can assign low weight to noisy or deterministic and therefore irrelevant dimensions, a generative model must explain them. If there are enough irrelevancies, it ignores the dimensions relevant to classification in the process. Combined with slow MCMC convergence, these difficulties have inhibited the use of nonparametric Bayesian methods in many applications.

To overcome these limitations, cross-categorization is utilized, which is an unsupervised learning technique for clustering based on MCMC inference in a novel nested nonparametric Bayesian model. This model can be viewed as a Dirichlet process mixture, over the dimensions or columns, of Dirichlet process mixture models over sampled data points or rows. Conditioned on a partition of the dimensions, our model reduces to an independent product of DP mixtures, but the partition of the dimensions, and therefore the number and domain of independent nonparametric Bayesian models, is also inferred from the data.

Standard feature selection results in the case where the partition of dimensions has only two groups. The described model utilizes MCMC because both model selection and deterministic approximations seem intractable due to the combinatorial explosion of latent variables, with changing numbers of latent variables as the partition of the dimensions changes.

The hypothesis space captured by the described model is super-exponentially larger than that of a Dirichlet process mixture, with a very different structure than a Hierarchical Dirichlet Process. A generative process, viewed as a model for heterogeneous data arrays with N rows, D columns of fixed type and values missing at random, can be described as follows:

1. For each dimension d ∈ D:
   a. Generate hyperparameters λ_d from an appropriate hyper-prior.
   b. Generate the model assignment z_d for dimension d from a Chinese restaurant process with hyperparameter α (with α from a vague hyperprior).
2. For each group g in the dimension partition {z_d}:
   a. For each sampled datapoint (or row) r, generate a cluster assignment z_r from a Chinese restaurant process with hyperparameter c_r (with c_r from a vague hyperprior).
   b. For each cluster c in the row partition for this group of dimensions {z_r}:
      i. For each dimension d, generate component model parameters θ_{d,c} from an appropriate prior and
         ii. For each data cell x_{(r,c)} in this component z_{r,c} for (r,c), generate its value from an appropriate likelihood and θ_{d,c}.

The model encodes a very different inductive bias than the IBP, discovering independent systems of categories over heterogeneous data vectors, as opposed to features that are typically additively combined. It is also instructive to contrast the asymptotic capacity of our model with that of a Dirichlet process mixture. The DP mixture has arbitrarily large asymptotic capacity as the number of samples goes to infinity. Put differently, it can model any distribution over finite dimensional vectors given enough data. However, if the number of dimensions (or features) is taken to infinity, it no longer asymptotically consistent: if we generate a sequence of datasets by sampling the first K_s dimensions from a mixture and then append K_s > K_s dimensions that are constant valued (e.g. the price of tea in China), it will eventually be forced to model only those dimensions, ignoring the statistical structure in the first K_s. In contrast, the model has asymptotic capacity both in terms of the number of samples and the number of dimensions, and is infinitely exchangeable with respect to both quantities.

As a consequence, it is self-consistent over the subset of variables measured, and can thus enjoy considerable robustness in the face of noisy, missing, and irrelevant measurements or confounding statistical signals. This should be especially helpful in demographic settings and in high-throughput biology, where noisy, or coherently co-varying but orthogonal, measurements are the norm, and each data vector arises from multiple, independent generative processes in the world.

The algorithm builds upon a general-purpose MCMC algorithm for probabilistic programs and specializes three of the kernels. It scales linearly per iteration in the number of rows and columns and includes inference over all hyperparameters.

FIG. 10B depicts an assessment of convergence, showing inferred versus ground truth. With reference to FIG. 10B, an assessment of convergence, showing inferred versus ground truth joint score for 1000 MCMC runs (200 iterations each) with varying dataset sizes (up to 512 by 512, requiring 1-10 minutes each) and true dimension groups. A strong majority of points fall near the ground truth dashed line, indicating reasonable convergence; perfect linearity is not expected, partly due to posterior uncertainty.

A preliminary comparison of the learning curves for cross-categorization and one-versus-all SVMs on synthetic 5-class classification, averaged over datasets generated from 10 dimensional Bernoulli mixtures.
[0103] The detailed mechanisms by which the latent variables introduced by this method above those in a regular DP mixture improve mixing performance. Massively parallel implementations exploit the conditional independencies in the described model. Because the described method is essentially parameter free (e.g., with improper uniform hyperpriors), robust to noisy and/or irrelevant measurements generated by multiple interacting causes, and supports arbitrarily sparsely observed, heterogeneous data, it may be broadly applicable in exploratory data analysis. Additionally, the performance of our MCMC algorithm suggests that the described approach to nesting latent variable models in a Dirichlet process over dimensions may be applied to generate robust, rapidly converging, cross-cutting variants of a wide variety of nonparametric Bayesian techniques.

[0104] A co-assignment matrix for dimensions, where:

\[
C_i \rightarrow \text{Pr}(z_i = \cdot | z_j)
\]

[0105] That is, the probability that dimensions i and j share a common cause and therefore are modeled by the same Dirichlet process mixture. Labels show the consensus dimension groups (probability > 0.75). These reflect attributes that share a common cause and thus co-vary, while the remainder of the matrix captures correlations between these discovered causes, for instance, mammals rarely have feathers or fly, ungulates are not predators, and so forth. Each dimension group picks out a different cross-cutting categorization of the rows (e.g., vertebrates, birds, canines, . . . , not shown).

[0106] An example may include data for 4273 hospitals by 74 variables, including quality scores and various spending measurements. The data is analyzed (~1 hour for convergence) with no preprocessing or missing data imputation.

[0107] Each box contains one consensus dimension group and the number of categories according to that group. In accordance with custom statistical analyses, no causal dependence between quality of care, hospital capacity, and spending is found, though each kind of measurement results in a different clustering of the hospitals. Also recovered is the cost structure of modern hospitals (e.g., increased long-term care causes increased ambulance costs, likely due to an increase in at-home mishaps). Standard clustering methods miss most of this type cross-cutting structure.

[0108] Veritable and associated Veritable APIs make use of a predictive database that finds the causes behind data and uses these causes to predict and explain the future in a highly automated fashion heretofore unavailable, thus allowing any developer to carry out scientific inquiries against a dataset without requiring custom programming and consultation with mathematicians and other such experts.

[0109] Veritable works by searching through the massive hypothesis space of all possible relationships present in a dataset, using an advanced Bayesian machine learning algorithm. The described Veritable technologies offer developers: state of the art inference performance and predictive accuracy on a very wide range of real-world datasets, with no manual parameter tuning whatsoever; scalability to very large datasets, including very high-dimensional data with hundreds of thousands or millions of columns; completely flexible predictions (e.g., predict the value of any subset of columns, given values for any other subset) without any retraining or adjustment; and quantification of the uncertainty associated with its predictions, since the system is built around a fully Bayesian probability model.

[0110] Described applications built on top of Veritable range from predicting heart disease, to understanding health care expenditures, to assessing business opportunities and scoring a likelihood to successfully “close” such business opportunities (e.g., to successfully commensurate a sale, contract, etc.).

[0111] Consider the problems with real-world data. There are different kinds of data mixed together. Data is entered and maintained by people with other things on their minds and real work to get done. Real-world data contains errors and blanks, such as null values in places where populated values are appropriate. Users of the data may be measuring the wrong thing, or may be measuring the same thing in ten different ways. And in many organizations, no one knows precisely what data is contained in the database. Perhaps there was never a DBA (Data Base Administrator) for the organization, or the DBA left, and ten years of sedimentary layers of data has since built up. All of these are very realistic and common problems with “real-world” data found in production databases for various organizations, in contrast to pristine and small datasets that may be found in a laboratory or test setting.

[0112] A system is needed that can make sense of data as it exists in real businesses and does not require an pristine dataset or conform to typical platonic ideals of what data should look like. A system is needed that can be queried for many different questions about many different variables, in real time. A system is needed which is capable of getting at the hidden structures in such data, that is, which variables matter and what are the segments or groups within the data. At the same time, the system must be trustworthy, that is, it can’t lie to the users by providing erroneous relationships and predictions. Such a system shouldn’t reveal things that aren’t true and shouldn’t report host patterns may exist in a first dataset, but won’t hold up overall. Such desirable characteristics are exceedingly difficult to attain with customized statistical analysis and customized predictive modeling, and wholly unheard of in automated systems.

[0113] According to the described embodiments, the resulting database appears to its users much like a traditional database. But instead of selecting columns from existing rows, users may issue predictive query requests via a structured query language. Such a structured language, rather than SQL may be referred to the PreQL, or PreQL Query Language (“PreQL”). PreQL is not to be confused with PQL which is short for the “Program Query Language.”

[0114] PreQL is thus used to issue queries against the database to predict values. Such a PreQL query offers the same flexibility as SQL-style queries. When exploring structure, users may issue PreQL queries seeking notions of similarity that are hidden or latent in the overall data without advanced knowledge of what those similarities may be. When used in a multi-tenant database system against a massive cloud based database and its dataset, such features are potentially transformative in the computing arts.

[0115] According to certain embodiments, Veritable utilizes a specially customized probabilistic model based upon foundational CrossCat modeling. CrossCat is a good start but could nevertheless be improved. For instance, it was not possible to run equations with CrossCut, which is solved via the core of Veritable which uses a particular engine implementation enabling such equation execution. Additionally, prior models matched data with the model to understand hidden structure, like building a probabilistic index, but was so com-
plex that its users literally required advanced mathematics and probability theory understanding simply to implement the models for any given dataset, rendering mere mortals incapable of realistically using such models. Verifiable implementations described herein provide a service which includes distributed processing, job scheduling, persistence, checkpointing, and a user-friendly API or front-end interface which accepts users’ questions and queries via the PreQL query structure. Other specialized front-end GUIs and interfaces are additionally described to solve for particular use cases on behalf of users and provide other simple interfaces to complex problems of probability.

[0116] What is probability? There are many perspectives, but probability may be described as a statement, by an observer, about a degree of belief in an event, past, present, or future. Timing doesn’t matter

[0117] What is uncertainty? An observer, as noted above, doesn’t know for sure whether an event will occur, notwithstanding the degree of belief in such an event having occurred, occurring, or to occur in the future.

[0118] Probabilities are assigned relative to knowledge or information context. Different observers can have different knowledge, and assign different probabilities to same event, or assign different probabilities even when both observers have the same knowledge. Probability, as used herein, is a number between “0” (zero) and “1” (one), in which 0 means the event is sure to not occur on one extreme of a continuum and where 1 means the event is sure to occur on the other extreme of the same continuum. Both extremes are uninteresting because they represent a complete absence of uncertainty.

[0119] A probability ties belief to one event. A probability distribution ties beliefs to every possible event, or at least, every event we want to consider. Choosing the outcome space is an important modeling decision. Summed over all outcomes in space is total probability which must be a total of “1,” that is to say, one of the outcomes must occur. Probability distributions are convenient mathematical forms that help summarize the system’s beliefs in the various probabilities, but choosing a standard distribution is a modeling choice in which all models are wrong, but some are useful.

[0120] Consider for example, a Poisson distribution which is a good model when some event can occur 0 or more times in a span of time. The outcome space is the number of times the event occurs. The Poisson distribution has a single parameter, which is the rate—the average number of times. Its mathematical form has some nice properties: Defined for all the non-negative integers $\sum_{i=0}$.

[0121] Many examples exist besides the Poisson distribution. Each such standard distribution encompasses a certain set of assumptions, such as a particular outcome space, a particular way of assigning probabilities to outcomes, etc. If you work with them, you’ll start to understand why some are nice and some are frustrating if not outright evil.

[0122] Verifiable utilizes distributions which move beyond the standard distributions with specially customized modeling thus allowing for a more complex outcome space and further allowing for more complex ways of assigning probabilities to outcomes. Depicted at slide 8b above is what’s called a mixture distribution combining a bunch of simpler distributions to form a more complex one. A mixture of Gaussians to model any distribution may be employed, while still assigning probabilities to outcomes, yielding a more involved mathematical relationship.

[0123] With more complex outcome spaces, a Mondrian process defines a distribution on k-dimensional trees, providing means for dividing up a square or a cube. The outcome space is all possible trees and resulting divisions look like the famous painting. The outcome space is more structured than what is offered by the standard distributions. CrossCut does not use the Mondrian process, but it does use a structured outcome space. Verifiable utilizes the Mondrian process in select embodiments.

[0124] At a high level, probability theory is a generalization of logic. Just like computers can use logic to reason deductively, probability lets computer reason inductively, generalize, categorize, etc. Probability gives us a way to combine different sources of information in a systematic manner, that is, utilizing automated computer implemented functionality, even when that information is vague, or uncertain, or ambiguous.

[0125] FIG. 6 depicts a simplified flow for probabilistic modeling. Modeling is a series of choices and assumptions. For instance, it is possible to trade off fidelity and detail with tractability. Assumptions define an outcome space. Such an outcome space may be considered hypotheses, and in the modeling view, one of these possible hypotheses typically occurs. This is the hidden structure, and it is this hidden structure that generates the data. The hidden structure and the resulting generated data may be considered the generative view. For learning or inference we observe the data and infer the hidden structure. Inference the likelihood of the data given the hidden structure may include certain modeling assumptions (“prior”), as well as data observed (“likelihood”), from which a combination of prior and likelihood may be utilized to draw conclusions (“posterior”).

[0126] Such assumptions don’t just give us a hypothesis space, they also give us a way of assigning probabilities to them, yielding a probability distribution on hypotheses, given the actual data observed.

[0127] There can be a great many hypotheses and finding the best ones to explain the data is not a straightforward or obvious proposition.

[0128] Modeling makes assumptions and using the assumptions defines a hypothesis space. Probabilities are assigned to the hypotheses given data observed and then inference is used to figure out which of those explanatory hypotheses is the best or are plausible.

[0129] Many approaches exist and experts in the field do not agree on how to select the best hypothesis. In simple cases, we can use math to solve the equations directly. Optimization methods are popular such as hill climbing and its relatives. Verifiable may use any such approaches, but according to certain described embodiments, Monte Carlo methods are specifically utilized in which a random walk is taken through the space of hypotheses. Random doesn’t mean stupid, of course. In fact, efficiently navigating these huge spaces is one of the innovations utilized to improve the path taken by the random walk.

[0130] FIG. 7 illustrates an exemplary landscape upon which a random walk may be performed. Consider the above exemplary random walk in which each axis is one dimension in the hypothesis space. Real spaces can have very many dimensions. Height of surface is probability of the hidden variables, given data and modeling assumptions. Exploration starts by taking a random step somewhere, anywhere, and if the step is higher then it is kept, but if the step is lower, then it is kept sometimes and other times it is not, electing to stay put instead. The result is seemingly magic, in which it is
guaranteed to explore the space in proportion to the true probability values. Over the long run two peaks result in this example whereas simple hill climbing will get caught. Such an approach thus explores the whole space. Other innovations include added intelligence about jumps and exploring one or many dimensions at a time.

[0131] According to exemplary models, tabular data is provided such that rows are individual entities and columns contain a certain piece of information about the respective entity (e.g., field). Different kinds of columns in one table are acceptable.

[0132] FIG. 8 depicts an exemplary tabular dataset. In the exemplary table rows are mammals and columns are variables that describe the mammals. Most are Boolean but some are categorical.

[0133] FIG. 9 depicts means for deriving motivation or causal relationships between observed data. Consider the realistic problems with “real-world” data described in prior slides in which it is proffered that real-world data is not pristine. For instance, there may be data which simply doesn’t matter. For instance, Some columns may not matter or certain columns may carry redundant information. A system is needed which utilizes a model that can understand the predictive relationships between all the columns. That is, some columns are predictively related and should get grouped together whereas others are not predictively related, and should be grouped separately. We call these groups of columns “views.” Within each view, the rows are grouped into categories.

[0134] FIG. 10A depicts an exemplary cross categorization in still further detail.

[0135] Utilizing cross-categorization, columns/features are grouped into views and rows/entities group into categories. View 3/Category 3 from slide 19 above is expanded to the right having 8 features and 4 entities. The highlighted rows are in different categories in View 3 but are in the same category in another view. Zoom in again, and it is seen that this category contains the actual data points corresponding to the cell values in the table for just the columns in this view, and just the rows in this category.

[0136] A single cross-categorization is a particular way of slicing and dicing the table. First by column and then by row. It’s a particular kind of process to yield a desired structured space. Utilizing concepts discussed above with respect to probability distributions, a probability is then assigned to each cross-categorization. More complex cross-categorizations yielding more views and more categories are less probable in and of themselves and typically are warranted only when the data really supports them.

[0137] FIG. 11 depicts a chart and graph of the Bell number series. A series called the Bell numbers defines the number of partitions for n labeled objects which as can be seen from the above graph on the right, grows really, really fast. A handful of objects are exemplified in the chart on the left. Plotted on the right is a plot through 200 resulting in $e^{+250}$ or a number with 250 zeros. Now consider the massive datasets available in a cloud computing multitenant database system which could easily result in datasets of interest with thousands of columns and millions of rows. Thus, such datasets will not merely result in the Bell numbers depicted above, but rather, potentially the Bell’s “squared,” placing us firmly into the land of infeasible numbers.

[0138] These numbers are so massive that it may be helpful to consider the following context. The number in red, that is, the horizontal line near the very bottom of the plot on the right, is the total number of web pages. Thus, Google only needs to search through the 17th bell number or so. The space is so unimaginably massive that it is not possible to explore it exhaustively. Moreover, because it’s not smooth or concave, you can’t just climb the hill either.

[0139] So where does the data come into cross-categorizations? Views pick out some of the columns and categories pick out some of the rows. Each column contains a single kind of data such as vertical strip within a category contains typed data such as numerical, categorical, etc. Now, the basic standardized distributions may be utilized more effectively. In certain embodiments, each collection of points is modeled with a single simple distribution. Basic distributions are pre-selected which work well for each data type and each selected basic distribution is only responsible for explaining a small subset of the actual data, for which it is particularly useful. Then using the mixture distribution discussed above, the basic distributions are combined such that a bunch of simple distributions are used to make a more complex one. The structure of the cross-categorization is used to chop up the data table into a bunch of pieces and each piece is modeled using the simple distribution selected of the data type, yielding a big mixture distribution of the data.

[0140] FIG. 12A depicts an exemplary cross categorization of a small tabular dataset.

[0141] So what does this look like? Go back to our mammals example providing a sample cross-categorization having two views. The view on the right has the habitat and feeding style columns and the rows are divided into four categories land mammals (Persian cat through Zebra), sea predators (dolphin through walrus), baleen whales (blue whale and humpback), and the outlier (e.g., both land and water living; we do not suggest that mammal bears have gills). The view on the left has another division in which the primates are grouped together, large mammals are grouped, grazers are grouped, and then a couple of oddities at the bottom (bat and seal). Even with a small dataset it is easy to imagine different ways of dividing the data up. But data is ambiguous. There is no perfect or obviously right division. For all the groupings that seemingly fit correctly, certain groupings may seem awkward or poor fitting. The systematic process of applying various models and assumptions makes tradeoffs and compromises, which is why even experts cannot agree on a single approach. Nevertheless, the means described herein permits use of a variety of available models such that these tradeoffs and compromises may be exploited to further benefit the system.

[0142] Results are thus not limited to a single cross-categorization. Instead, a collection of them are utilized and such a collection when used together tells us about the hidden structure of the data. For instance, if they’re all the same, then there was no ambiguity in the data, but such a result doesn’t occur with real-world data, despite being a theoretical possibility. Conversely, if they’re all completely different, that means we couldn’t find any structure in the data, which sometimes happens, and requires some additional post-processing to get at the uncertainty, such as feeding in additional noise. Typically, however, something in between occurs, and some interesting hidden structure is revealed from the data.

[0143] The specially customized cross-categorization implementation represents the core of Veritable. This core is not directly exposed to the users who interface via APIs,
PreQL, and specialized utility GUIs and interfaces, but such users nevertheless benefit from the functionality which drives these other capabilities.

The Veritable core utilizes Monte Carlo methods for certain embodiments.

FIG. 12B depicts an exemplary architecture having implemented data upload, processing, and predictive query API exposure in accordance with described embodiments.

First, you need to get data into the system so API calls are provided to upload data into the system. A row is the basic unit. An API call for "Analyze" kicks off a learning pass applying the specially customized CrossCat model for the uploaded data. It's also possible to specify an existing dataset, or to define a sub-set of data from a larger dataset.

Cross-categorizations are found in the data that are most plausible explanations of the data at hand. Though such functions happen in the background out of the eyes of the user, such functionality is computationally intensive and is thus, well suited for a distributed computing structure provided by a cloud based multi-tenant database system technology.

FIG. 12C is a flow diagram illustrating a method for implementing data upload, processing, and predictive query API exposure in accordance with disclosed embodiments.

FIG. 12D depicts an exemplary architecture having implemented predictive query interface as a cloud service in accordance with described embodiments.

FIG. 12E is a flow diagram illustrating a method for implementing predictive query interface as a cloud service in accordance with disclosed embodiments.

FIG. 13A illustrates usage of the RELATED command term in accordance with the described embodiments.

Using PreQL, specialized queries are thus made feasible. For instance, we can ask: for a given column, what are the other columns that are predictively related to it? In terms of the cross-categorizations, we tabulate how often each of the other columns appears in the same view as the input column, thus revealing what matters and what doesn’t matter.

FIG. 13B depicts an exemplary architecture in accordance with described embodiments.

FIG. 13C is a flow diagram illustrating a method in accordance with disclosed embodiments.

FIG. 14A illustrates usage of the GROUP command term in accordance with the described embodiments.

FIG. 14B depicts an exemplary architecture in accordance with described embodiments.

FIG. 14C is a flow diagram illustrating a method in accordance with disclosed embodiments.

Using PreQL, we can ask what rows “go together.” Such a feature can be conceptualized as clustering, except that there’s more than one way to cluster. Consider the mammals example in which we additionally input a column via the PreQL query. The groups that are returned are in the context of that column.

FIG. 15A illustrates usage of the SIMILAR command term in accordance with the described embodiments.

FIG. 15B depicts an exemplary architecture in accordance with described embodiments.

FIG. 15C is a flow diagram illustrating a method in accordance with disclosed embodiments.

Using PreQL, we can ask which rows are most similar to a given row. Rows can be similar in one context but dissimilar in another. For instance, killer whales and blue whales are a lot alike in some respects, but very different in others. The input column disambiguates.

FIG. 16A illustrates usage of the PREDICT command term in accordance with the described embodiments.

Using PreQL, we can predict, that is, ask the system to render a prediction. With the cross-categorizations a prediction request is treated as a new row and we assign that row to categories in each cross-categorization. Then using the basic standardized distributions for each category the values we want to predict are predicted. Unlike conventional predictive analytics, the system provides for flexible predictive queries thus allowing the user of the PreQL query to specify as many or as few columns as they desire and thus allowing the system to predict as many or as few as the user wants.

For instance, consider classification or regression in which all but one of the columns are used to predict a single target column. Veritable’s core, via the APIs, can predict using a single target column or can predict using a few target columns at the user’s discretion. For instance, a user can query the system asking: will an opportunity close AND at what amount? Such capabilities do not exist using conventional means.

FIG. 16B illustrates usage of the PREDICT command term in accordance with the described embodiments.

FIG. 16C illustrates usage of the PREDICT command term in accordance with the described embodiments. At the extreme, predict can be utilized to predict a row without fixing anything, thus asking Veritable to make up a row that isn’t actually in the underlying source data, but could be nevertheless, resulting in what may be considered a synthetic row. Such a row will exhibit all of the structure and predictive relationships as in the real data. Such a capability may enable a user to test a dataset that’s realistic, but not radioactive, without having to manually enter or guess at what such data may look like.

FIG. 16D depicts an exemplary architecture in accordance with described embodiments.

FIG. 16E is a flow diagram illustrating a method in accordance with disclosed embodiments.

FIG. 16F depicts an exemplary architecture in accordance with described embodiments.

FIG. 16G is a flow diagram illustrating a method in accordance with disclosed embodiments.

Alternatively, the user can take an incomplete row and predict all the missing values to fill in the blanks. At the extreme, the user can begin a table with many missing values and render a table where all of the blanks were filled in. Specialized tools for this particular use case are discussed below in which functionality allows the user to trade off confidence for more or less data, such that more data (or all the data) can be populated with degrading confidence or only some data is populated, above a given confidence, and so forth. A specialized GUI is additionally provided and described for this particular use case. Such a GUI calls the predict query via PreQL via an API on behalf of the user, but fundamentally exercises Veritable’s core.

FIG. 17A depicts a Graphical User Interface (GUI) to display and manipulate a tabular dataset having missing values by exploiting a PREDICT command term.

Here the table is provided as being 61% filled. No values are predicted, but the user may simply move the slider to increase the data fill for the missing values, causing the GUI’s functionality to utilize the predict function on behalf of the user.
FIG. 17B depicts another view of the Graphical User Interface.

Here the table is provided as being 73% filled. Some but not all values are predicted. Not depicted here is the confidence threshold which is hidden from the user. Alternative interfaces allow the user to specify such a threshold.

FIG. 17C depicts another view of the Graphical User Interface.

Here the table is provided as being 100% filled. All values are predicted, but it may be necessary to degrade the confidence somewhat to attain 100% fill. Though such a fill may nevertheless be feasible at acceptable levels of confidence. In each of the instances, the grey scale values show the original data and the blue values depict the predicted values which do not actually exist in the underlying table. In certain embodiments, the chosen fill level, selected by the user via the slider bar, can be “saved” to the original or a copy of the table, thus resulting in the predictive values provided being saved or input to the cell locations. Meta data can be used to recognize later that such values were predicted and not sourced.

FIG. 17D is an exemplary architecture in accordance with described embodiments.

FIG. 17E is a flow diagram illustrating a method in accordance with disclosed embodiments.

Other specialized GUIs and API tools include business opportunity scoring, next best offer, etc. These and other are described in additional detail below.

When making predictions, it is helpful to additionally let the users know whether they can trust the result. That is, how confident is the result and is the system literally capable of saying: “I do not know.” With such a system, the result may come back and tell the user if the answer is 1 or between 1 and 10 or between infinity and infinity.

With probabilities, the system can advise the user that it is 90% confident that the answer given is real, accurate, and correct, or the system may alternatively return a result indicating that it simply lacks sufficient data, and thus, there is not enough known to render a prediction. For example, the Veritable core may be used to a complete data set as if it was real by filling in the missing but predicted information into a spreadsheet, from which the completed data may be used to draw conclusions, as is depicted in the above slides 33, 34, and 35. The control slider is feasible because when you complete income, for example, what is actually returned to the GUI functionality making the call is the respective persons’ income distribution.

By using such a GUI interface or such a concept in general, the user is given control over accuracy and confidence. In such a way, the user can manipulate how much data is to be filled in and to what extent the confidence level applies. What Veritable does behind the scenes is to take a table of data with a bunch of typed columns, and then the depicted Perceptible GUI interface at slides 33, 34, and 35 asks for a prediction for every single cell. Then for each cell that is missing, the Perceptible GUI gets a distribution in return from the API call to the Veritable core for the individual cell. Then when the slider is manipulated by a user, functionality for the slider looks at the distributions and looks at their variances and then gives the estimates. Thus, for any given cell having a predicted result in place of the missing null value, having seen “a” and “b” and “c” it then returns a value for that column.

Starting with nothing more than raw data in a tabular form, such as data on paper, in a spreadsheet, or in tables of a relational database, an API call first made to upload or insert the data into the predictive database upon which Veritable operates and then an API call is made to the Veritable core to analyze the data. Upon insert, the data it looks just like all other data. But once uploaded and the analyze operation is initiated, a probabilistic model is executed against the data. So the Veritable core starts to look at the ways that the rows and the columns can interact with each other and start to build the various relationships and causations. A generated statistical index figures out how and which columns are related to another. Veritable goes through and says, for instance, these particular columns are likely to share a causal origin. The difficult problem is that Veritable must perform this analysis using real world realities rather than pristine and perfect datasets. With data that exists in the real world, some columns are junk, some columns are duplicates, some columns are heterogeneous, some columns are noisy with only sparse data, and Veritable’s core functionality implementing the statistical index must be able to pull the appropriate relationships and causations out despite having to perform its analysis operations against real-world data.

There will also be no one right answer, as there are uncertainties. So Veritable does not just build up one statistical model but rather, Veritable builds multiple statistical indices as a distribution or an ensemble of statistical indices. Veritable performs its analysis by searching through a large space for all of the ways that the data provided can possibly interact.

The distribution of indices results in a model that is stored and is queryable by ProQL structured queries via Veritable’s APIs. What Veritable figures out via the analysis operations is first how the columns group together and then how the various rows group together. The analysis thus discovers the hidden structure in the data to provide a reduced representation of a table that explains how rows and columns may be related such that they can be queried via ProQL.

ProQL structured queries allow access to the queryable model and its ensemble of indices through specialized calls, including: “RELATED,” “SIMILAR,” “GROUP,” AND “PREDICT,” each of which are introduced above at slides 28 through 32.

Beginning with PREDICT, calling an appropriate API for the PREDICT functionality enables users to predict any chosen sub-set of data predict any column/value. It is not required that an entire dataset be utilized to predict only a single value, as is typical with custom implemented models.

Using PREDICT, the user provides or fixes the value of any column and then the PREDICT API accepts the fixed values and those you want to predict. The functionality then queries the Veritable core asking: “Given a row that has these values fixed, as provided by the user, then what would the distribution be?” For instance, the functionality could fix all but one column in the dataset and then predict the last one, as is done with customized models. But the PREDICT functionality is far more flexible, and thus, the user can change the column to be predicted at a whim and custom implemented models simply lack this functionality as they lack the customized mathematical constructs to predict for such unforeseen columns or inquiries. That is to say, absent a particular function be pre-programmed, the models simply cannot perform this kind of varying query, for instance, for a user exploring data making multiple distinct queries or simply changing the
column or columns to be predicted as business needs and the underlying data and data structures of the client organization change over time. [0192] Perhaps also the user does not know all the columns to fix. For instance, perhaps the dataset knows a few things about one user but lots about another user. For instance, an ecommerce site may know little about a non-registered passerby user but knows lots of information about a registered user with a rich purchase history. In such an example, the PREDICT functionality permits fixing or filling in only the stuff that is known without having to require all data for all users, as some of the data is known to be missing. In such a way, the PREDICT functionality can still predict missing data elements with what is actually known.  

[0193] Another capability using the PREDICT functionality is to specify or fix all the data in a dataset that is known, that is, non-null, and then fill in everything else. In such a way, a user can say that what is known in the dataset is known, but much data is understood to be missing, but render predictions for the data nevertheless. The PREDICT operation would thus increase the population of predicted data for missing or null-values by accepting decreasing confidence, until the all or a specified population percentage of data is reached, much like the Perceptible GUI and slider examples described above.  

[0194] Another functionality using PREDICT is to fill in an empty set. So maybe data is wholly missing, and then you start generating data that represents new rows and the new data in those rows represents plausible data, albeit synthetic data.  

[0195] In other embodiments, PREDICT can be used to populate data elements that are not known but should be present or may be present, yet are not filled in within the data set, thus allowing the PREDICT functionality to populate such data elements.  

[0196] Another example is to use PREDICT to attain a certainty or uncertainty for any element and to display or return the range of plausible values or the element.  

[0197] Next is the RELATED functionality. Given a table with columns or variables in it, Veritable’s analysis behind the scenes divides the columns or variables into groups and because of the distributions there is more than one way to divide these columns or variables up. Take height for example. Giving the height column to an API call for RELATED, a user can query: “How confident can I be about the probability of the relationship existing in all the other columns with the height column so specified?” Then what is returned from the Veritable core for height is a confidence for every other column in the dataset which was not specified. So for example, the RELATED functionality may return confidence to the height column, “Weight–1.0” meaning that Veritable, according to the dataset, is extremely confident that there is a relationship between weight and height. Such a result is somewhat intuitive and expected. But other results may be less intuitive and thus provide interesting results for exploration and additional investigation. Continuing with the “height” example for the specified column to a RELATED API call, Veritable may return “Age:0.8″ meaning that Veritable is quite sure, but not perfectly certain, due to, for instance, noisy data which precludes an absolute positive result. Perhaps also returned for the specified “height” column is “hair color=0.1″ meaning there is realistically no correlation whatsoever between a person’s height and their hair color, according to the dataset utilized. Thus, the RELATED functionality permits a user to query for what matters for a given column, such as the height column, and the functionality returns all the columns with a scoring of how related the columns are to the specified column, based on their probability.  

[0198] Next is the SIMILAR functionality. Like the RELATED functionality, an API call to Veritable for SIMILAR accepts a row and then returns what other rows are most similar to the row specified. Like the RELATED examples, the SIMILAR functionality returns the probability that a row specified and any respective returned row actually exhibits similarity. For instance, rather than specifying column, you specify “Fred” as a row in the dataset. Then you ask using the SIMILAR functionality, for “Fred,” what rows are scored based on probability to be the most like “Fred.” The API call can return all rows scored from the dataset or return only rows above or below a specified threshold. For instance, perhaps rows above 0.8 are the most interesting or the rows below 0.2 are most interesting, or both, or a range. Regardless, SIMILAR scores every row for the specified row and returns the rows and the score based on probability according to the user’s constraints or the constraints of an implementing GUI, if any such constraints are given. Because the Veritable system figures out these relationships using its own analysis, there is more than way to evaluate for this inquiry. Thus, user must provide to an API call for SIMILAR the specified row to find and additionally the COLUMN which provides how you the user constructing the PreQl query or API call actually cares about the data. Thus, the API call requires both row and column to be fixed. In such a way, providing, specifying, or fixing the column variable provides disambiguation information to Veritable and the column indication tells the Veritable core where to enter the index. Otherwise there would be too many possible ways to score the returned rows as the Veritable core could not disambiguate how the caller cares about the information for which a similarity is sought.  

[0199] Next is the GROUP functionality. Sometimes rows tend to group up on noisy elements in the dataset when the Veritable core applies its analysis; yet these elements may result in groupings that are not actually important. We know that each column will appear in exactly one of the groups as a view and so Veritable permits using that column to identify the particular “view” that will be utilized. The GROUP functionality therefore implements a row-centric operation like the SIMILAR functionality, but in contrast to an API call for SIMILAR where you must give a row and the SIMILAR call returns back a list of other rows and a score based on their probabilities of being related, with the GROUP functionality, the API call requires no row to be given or fixed whatsoever. Only a column is thus provided when making a call to the GROUP functionality.  

[0200] Calling the GROUP functionality with a specified or fixed column causes the functionality to return the groupings of the ROWS that seem to be related or correlated in some way based on Veritable’s analysis.  

[0201] In such a way, use of the PreQl structure queries permits programmatic queries into the predictive database in a manner similar to a programmer making SQL queries into a relational database. Rather than a “select” statement in the query the term is replaced with the “predict” or “similar” or “related” or “group” statements. For instance, an exemplary PreQl statement may read as follows: “PREDICT IS_WON, DOLLAR_AMOUNT FROM OPPORTUNITY WHERE STAGE=’QUOTE’. ” So in this example, “QUOTE” is the
fixed column, "FROM" is the dataset from which an opportunity is to be predicted, the "PREDICT" term is the call into the appropriate function, "IS_WON" is the value to be predicted, that is to say, the functionality is to predict whether a given opportunity is likely or unlikely to be won where the "IS_WON" may have completed data for some rows but be missing for other rows due to, for example, pending or speculative opportunities, etc. "DOLLAR_AMOUNT" is the fixed value.

[0202] In certain embodiments, the above query is implemented via a specialized GUI interface which accepts inputs from a user via the GUI interface and constructs, calls, and returns data via the PREDICT functionality on behalf of the user without requiring the user actually write or even be aware of the underlying PreQL structure query made to the Veritable core.

[0203] There are additionally provide Veritable use case implementation embodiments and customized GUIs. According to a first embodiment, a specialized GUI implementation enables users to filter on a historical value by comparing a historical value versus a current value in a multi-tenant system. Filtering for historical data using a GUI’s field option wherein the GUI displays current fields related to historical fields as is depicted.

[0204] FIG. 19A depicts a specialized GUI to query using historical dates.

[0205] Embodiments provide for the ability to filter historical data by comparing historical value versus a constant in a multi-tenant system. The embodiments utilize the Veritable core by calling the appropriate APIs to make queries on behalf of the GUI users. The GUI performs the query and then consumes the data which is then presented back to the end users via the interface. Consider for example, a sales person looking at the sales information in a particular data set. The interface can take the distributions provided by Veritable’s core and produce a visual indication for ranking the information according to a variety of customized solutions and use cases.

[0206] For instance, in a particular embodiment, systems and methods for determining the likelihood of an opportunity to close using only closed opportunities is provided.

[0207] Salesforce is an industry leading CRM application currently used by 125,000 enterprise customers. Customers see the value of storing the data in the Cloud. These customers appreciate a web based interface to view and act on their data, and these customers like to use report and dashboard mechanisms provided by the cloud based service. Presenting these various GUIs as tabs enables salespeople and other end users to explore their underlying dataset in a variety of ways to learn how their business is performing in real-time. These users also rely upon partners to extend the provided cloud based service capabilities through APIs.

[0208] A cloud based service that offers customers the opportunity to learn from the past and draw data driven insights is highly desirable as such functionality should help these customers make intelligent decisions about the future for their business based on their existing dataset.

[0209] The customized GUIs utilize Veritable’s core to implement predictive models which may vary per customer organization or be tailored to a particular organization needs via programmatic parameters and settings exposed to the customer organization to alter the configuration and operation of Veritable’s functionality.

[0210] For instance, a GUI may be provided to compute and assign an opportunity score based on probability for a given opportunity reflecting the likelihood of that opportunity to close as a win or loss. The data set to compute this score would consists of all the opportunities that have been closed (either won/loss) in a given period of time, such as 1, 2, or 3 years or a lifetime of an organization, etc. Additional data elements from the customer organization’s dataset may also be utilized, such as the account object as an input. Machine learning techniques implemented via Veritable’s core, such as SVN, Regression, Decision Trees, PGM, etc., are then used to build an appropriate model to render the opportunity score and then the GUI depictions the information to the end user via the interface.

[0211] Systems and methods for determining the likelihood of an opportunity to close using historical trending data is additionally disclosed. For instance, a historical selector for picking relative or absolute dates is described.

[0212] FIG. 19B depicts an additional view of a specialized GUI to query using historical dates.

[0213] With this example we enable users to look at how an opportunity has changed over time, independent of stage, etc. The user can additionally look at how that opportunity has matured from when created until when it was closed.

[0214] Systems and methods for determining the likelihood of an opportunity to close at a given stage using historical trending data. Where the above example operates independent of stage of the sales opportunity this example further focuses on the probability of closing at a given stage as a further limiting condition for the closure. Thus, customers are enabled to use the historical trending data to know exactly when the stage has changed and then additionally predict what factors were involved to move from stage 1 to 2, from stage 2 to 3 and so forth.

[0215] Systems and methods for determining the likelihood for an opportunity to close given social and marketing data is additionally disclosed. In this example, the dataset of the customer organization or whomever is utilizing the system is expanded on behalf of the end user beyond that which is specified and then that additionally information is utilized to further influence and educate the predictive models. For instance, certain embodiments pull information from an exemplary website such as “data.com,” and then the data is associated with each opportunity in the original data set to discover further relationships, causations, and hidden structure which can then be presented to the user. Other data sources are equally feasible, such as pulling data from social networking sites, search engines, data aggregation service providers, etc.

[0216] In one embodiment, social data is retrieved and a sentiment is provided to the end-user via the GUI to depict how the given product is viewed by others in a social context. Thus, a salesperson can look at the persons linked in profile and with information from data.com or other sources the salesperson can additionally be given sentiment analysis in terms of social context for the person that the salesperson is actually trying to sell to. For instance, has the target purchaser commented about other products or have they complained about any other products, etc. Each of these data points and others based may help influence the model employed by Veritable’s core to render a prediction.

[0217] Systems and methods for determining the likelihood for an opportunity to close given industry specific data is additionally disclosed. For instance, rather than using...
socially relevant data for social context of sentiment analysis, industry specific data can be retrieved and input to the predictive database upon which Veritable performs its analysis as described above, and from which further exploration can then be conducted by users of the database now having the industry specific data integrated therein.

[0218] According to other embodiments, datasets are explored beyond the boundaries of any particular customer organization having data within the multi-tenant database system. For instance, in certain embodiments, benchmark predictive scores are generated based on industry specific learning using cross-organizational data stored within the multi-tenant database system. For example, data mining may be performed against telecom specific customer datasets, given their authorization or license to do so. Such cross-organization data to render a much larger multi-tenant dataset can then be analyzed via Veritable and provide insights, relationships, causations, and additional hidden structure that may not be present within a single customer organizations’ dataset. For instance, if as a customer you are trying to close a $100k deal in the NY-NJ-Virginia tri-city area, the probability for that deal to close in 3 months may be, according to such analysis, 50% because past transactions have shown that it could take up to six months to close a $100k telecom deal in NY-NJ-Virginia tri-city area when viewed in the context of multiple customer organizations’ datasets. Many of the insights realized through such a process may be non-intuitive, yet capable of realization through application of the techniques described herein.

[0219] With industry specific data present within a given dataset it is possible to delve even deeper into the data and identify benchmark using such data for a variety of varying domains across multiple different industries. For instance, based on such data predictive analysis may review that, in a given region it takes six months to sell sugar in the mid west and it takes three months to sell laptops in the east coast, and so forth.

[0220] Then if a new opportunity arises and a vendor is trying to, for example, sell watches in California, the vendor can utilize such information to gain a better understanding of the particular regional market based on the predictions and confidence levels given.

[0221] Provided functionality can additionally predict information for a vertical sector as well as for the region. When mining a customer organization’s dataset a relationship may be discovered that where customers bought a, those customers also bought b. These kinds of matching relationships are useful, but can be further enhanced. For instance, using the predictive analysis of Veritable it is additionally possible to identify the set of factors that led to a particular opportunity score (e.g., a visualized presentation of such analysis).

[0222] FIG. 19C depicts another view of a specialized GUI to configure predictive queries.

[0223] Thus, the GUI presents a 42% opportunity at the user interface but when the user mouses over the opportunity score, the GUI then displays sub-detailed elements that make up that opportunity score. The GUI makes the necessary Veritable based API calls on behalf of the user such that an appropriate call is made to the predictive platform to pull the opportunity score and display that information to the user as well as the sub-detail relationships and causations considered relevant.

[0224] The GUI can additionally leverage the predict and analyze capabilities of Veritable which upon calling a predict function for a given opportunity will return data necessary to create a histogram for an opportunity. So not only can the user be given a score, but the user can additionally be given the factors and guidance on how to interpret the information provided and what to do with such information.

[0225] Moreover, as the end-users, such as salespeople, see the data and act upon it, a feedback loop is created through which further data is input into the predictive database upon which additional predictions and analysis are carried out in an adaptive manner. For example, as the Veritable core learns more about the data the underlying models may be refreshed on a monthly basis by re-performing the analysis of the dataset so as to re-calibrate the data using the new data obtained via the feedback loop.

[0226] Additionally disclosed are systems and methods to deliver a matrix report for historical data in a multi-tenant system. For instance, consider a summary view of a matrix report according to provided embodiments.

[0227] Systems and methods to deliver a matrix report for historical data in a multi-tenant system follow the established matrix report format familiar to salesforce.com customers but which is limited only to current data and cannot display historical data. With Veritable’s capabilities the historical data can additionally be provided via the matrix reports.

[0228] FIG. 19D depicts an exemplary architecture in accordance with described embodiments.

[0229] FIG. 19E is a flow diagram illustrating a method in accordance with disclosed embodiments.

[0230] FIG. 20A depicts a pipeline change report in accordance with described embodiments. For example, a user can request to be shown the open pipeline for the current month by stage.

[0231] In a summary view, users can see data in an aggregate fashion. Each stage may consist of multiple opportunities and each might be able to be duplicated because each might change according to the amount or according to the stage, etc. Thus, if a user is looking at the last four weeks, then one opportunity may change from $500 to $1500 and thus be duplicated.

[0232] The cloud computing architecture executes functionality which runs across all the data for all tenants. Thus, for any cases, leads, and opportunities, the database maintains a history object into which all of audit data is retained such that a full and rich history can later be provided to the user at their request to show the state of any event in the past, without corrupting the current state of the data. Thus, while the underlying data must be maintained in its correct state for the present moment, a user may nevertheless utilize the system to display the state of a particular opportunity as it stood last week, or as it transitioned through past quarter, and so forth.

[0233] All of the audit data from history objects for various categories of data is then aggregated into a historical trending entity which is a custom object that stores any kind of data. This object is then queried by the different historical report types across multiple tenants to retrieve the necessary audit trail data such that any event at any time in the past can be re-created for the sake of reporting, predictive analysis, and exploration. The historical audit data may additionally be subjected to the analysis capabilities of Veritable by including it within a historical dataset for the sake of providing further predictive capabilities.
The algorithms to provide historical reporting capabilities are applied across all the tenant data which is common within the historical trending data object and the interim opportunity history and lead history, etc.

Within the matrix report, the data can also be visualized using salesforce.com’s charting engine as depicted by the waterfall diagram. FIG. 20B depicts a waterfall chart using predictive data in accordance with described embodiments.

Systems and methods to deliver waterfall charts for historical data in a multi-tenant system are thus provided. For instance, on the x axis is the weekly snapshot for all the opportunities being worked. The amounts are changing up and down and then are also grouped by stages. The waterfall enables a user to look at two points in time and by defining opportunities between day one and day two. Alternatively, waterfall diagrams can be used to group all opportunities into different stages as in the example above which every opportunity is mapped according to its stage allowing a user to look into the past and understand what the timing is for these opportunities to actually come through to closure.

Historical data and the audit history saved to the historical trending data object are enabled through snapshots and field history. Using the historical trending data object the desired data can then be queried. The historical trending data object may be implemented as one table with indexes on the table from which any of the desired data can then be retrieved. The software modules implementing the various GUIs and use cases populate the depicted table using the opportunity data retrieved from the historical trending data object’s table.

Additionally disclosed are systems and methods to deliver waterfall chart for historical data in a multi-tenant system and historical trending in a multi-tenant environment. Systems and methods for using a historical selector for picking relative or absolute dates are additionally described.

These specialized implementations enable users to identify how the data has changed on a day to day basis or week to week basis or over a month to month basis, etc. The users can therefore see the data that is related to the user’s opportunities not just for the present time, but with this feature, the users can identify opportunities based on a specified time such as absolute time or relative time, so that they can see how the opportunity has changed over time. In this embodiment, time as a dimension is used to then provide a decision tree for the customers to pick either absolute date or a range of dates. Within the date customers can pick an absolute date, such as Jan. 1, 2013 or a relative date such as the first day of the current month or first day of the last month, etc.

This solves the problem of a sales manager or salesperson needing to see how the opportunity has changed today versus the first day of this month or last month. With this capability, the user can take a step back in time, thinking back where they were a week ago, or a month ago and identify the opportunity by creating a range of dates and displaying what opportunities were created during those dates.

Thus, a salesperson wanting such information may have had ten opportunities and on Feb. 1, 2013, the salesperson’s target buyer expresses interest in a quote, so the stage changes from prospecting to quotation. Another target buyer, however, says they want to buy immediately, so the state changes from quotation to sale/charge/close. The functionality therefore provides a back end which implements a decision tree with the various dates that are created. The result is that the functionality can give the salesperson a view of all the opportunities that are closing in the month of January, or February, or within a given range, etc.

The query for dates is unique because it is necessary to traverse the decision tree to get to the date the user picks and then enabling the user to additionally pick the number of snapshots, from which the finalized result set is determined, for instance, from Feb. 1 to Feb. 6, 2013.

Additionally described is the ability to filter historical data by comparing historical values versus current values in a multi-tenant system as is shown.

FIG. 20C depicts an interface with defaults after adding a first historical field.

Additionally enabled is the ability to filter historical data by comparing historical values versus a constant in a multi-tenant system, referred to as a historical selector. Based on the opportunity or report type, the customer has the ability to filter on historical data using a custom historical filter. The interface provides the ability for the customer to look at all of the filters on the left that they can use to restrict a value or a field, thus allowing customers to filter on historical column data for any given value. Thus, a customer may look at all of the open opportunities for a given month or filter the data set according to current column data rather than historical. Thus, for a given opportunity a user at the interface can fill out the amount, stage, close date, probability, forecast category, or other data elements and then as the salesperson speaks with the target buyer, the state is changed from prospecting to quoting, to negotiation based on the progress that is made with the target buyer, and eventually to a state of won/closed or lost, etc. So maybe the target buyer is trying to decrease the amount of the deal and the salesperson is trying to increase the amount. All of that data and state changes (e.g., a change in amount can be a state change within a given phase of the deal) and the information is stored in the historical opportunity object which provides the audit trail.

As the current data changes the data in the current tables change and thus, historical data is not accessible to the customer. But the audit trail is retained so it can be retrieved. For instance, the GUI enables a user to go back 12 months according to one embodiment. Such historical data and audit data may be processed with granularity of one day, and thus, a salesperson can go back in time and view how the data has changed overtime with within the data set with the daily granular reporting.

Thus, for any given opportunity object with all the object history and the full audit trail with daily granular data the historical trending entity object is used to allow the tool to pull the information about how these opportunities changed over time for the salesperson. Such metrics would be useful to other disciplines also, such as a service manager running a call center who gets 100 cases from sales agents wants to know how to close those calls, etc. Likewise, when running a marketing campaign, is it being spent in California, or Tokyo, etc., the campaign managers will want to know how to close the various leads an opportunities as well as peer back into history to see how events influenced the results of past opportunities.

Additional detail with respect to applying customized filters to historical data is further depicted, as follows:

FIG. 20D depicts additional detail an interface with defaults for an added custom filter.

FIG. 20E depicts another interface with defaults for an added custom filter.
[0252] FIG. 20F depicts an exemplary architecture in accordance with described embodiments.

[0253] FIG. 20G is a flow diagram illustrating a method in accordance with disclosed embodiments.

[0254] According to one embodiment, the historical trending entity object is implemented via a historical data schema in which history data is stored in a new table core: historical_entity_data as depicted at Table 1 below:

<table>
<thead>
<tr>
<th>column name</th>
<th>data type</th>
<th>nullable</th>
<th>notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>organization_id</td>
<td>char(15)</td>
<td>no</td>
<td>key prefix of historical data itself</td>
</tr>
<tr>
<td>key_prefix</td>
<td>char(3)</td>
<td>no</td>
<td>FK to the parent record</td>
</tr>
<tr>
<td>historical_entity_id</td>
<td>char(15)</td>
<td>no</td>
<td>generated key used to uniquely identify</td>
</tr>
<tr>
<td>parent_id</td>
<td>char(15)</td>
<td>no</td>
<td>transaction that changed the parent record. Main purpose is to reconcile multiple changes that may occur in one transaction (custom field versus standard field, for example may be written separately) and enable asynchronous fiber operations (if used).</td>
</tr>
<tr>
<td>transaction_id</td>
<td>char(15)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>division</td>
<td>number</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>currency_iso_code</td>
<td>char(15)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>deleted</td>
<td>char(15)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>row_version</td>
<td>number</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>standard_audit_fields</td>
<td>date</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>valid_from</td>
<td>date</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>valid_to</td>
<td>date</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>val0 ... val800</td>
<td>varchar(765)</td>
<td>yes</td>
<td>default to 3000/1/1 for current data; flex fields for storing historic values</td>
</tr>
</tbody>
</table>

[0255] Indices utilized in the above Table 1 include: organization_id, key_prefix, historical_entity_data_id. PK includes: organization_id, key_prefix, system_modstamp. Unique, find, and snapshot for given date and parent record: organization_id, key_prefix, parent_id, valid_to, valid_from. Indices organization_id, key_prefix, valid_to facilitate data clean up. Such a table is additionally counted against storage requirements according to certain embodiments. Usage may be capped at 100. Alternatively, when available slots are running low, old slots may be cleaned. Historical data management, row limits, and statistics may be optionally utilized. For new history the system assumes an average 20 byte per column and 60 effective columns (50 effective data columns+PK+audit fields) for the new history table, and thus, row size is 1200 bytes. For row estimates the system assumes that historical trending will have usage patterns similar to entity history. Since historical trending storage is charged to the customer's applicable resource limits, it is expected that usage will not be heavier than usage of entity history.

[0256] Sampling of production data revealed recent grow in row count for entity history is ~2.5 B (billion) rows/year. Since historical trending will store a single row for any number of changed fields, an additional factor of 0.78 can be applied. Since historical trending will only allow 4 standard and at most 5 custom objects, additional factor of 0.87 can be used to only include top standard and custom objects contributing to entity history. With additional factor of 0.7 to only include UE and EE organizations, the expected row count for historical trending is 1.2 B row/year in the worst case scenario.

[0257] Historical data may be stored by default for 2 years and the size of the table is expected to stay around 2.4 B rows. Custom value columns are to be handled by custom indexes similar to custom objects. To prevent unintentional abuse of the system, for example, by using automated scripts, each organization will have a history row limit for each object. Such a limit could be between approximately 1 and 5 million rows per object which is sufficient to cover storage of current data as well as history data based on analyzed usage patterns of production data with only very few organizations (3-5) occasionally having so many objects that they would hit the configurable limit. The customized table can be custom indexed to help query performance.

[0258] High level use cases for such historical based data in a dataset to be analyzed and subjected to Verifiable's predictive analysis include: Propensity to Buy and Lead Scoring for sales representatives and marketing users. For instance, sales users often get leads from multiple sources (marketing, external, sales prospecting etc.) and often times, in any given quarter, they have more leads to follow up than they have time. Sales representatives often need guidance with key questions such as: which leads have the highest propensity to buy, what is the likelihood of a sale, what is the potential revenue impact if this lead is converted to an opportunity, what is the estimated sale cycle based on historical observations and if this lead is converted to an opportunity, what is the lead score for each of their leads in their pipeline so that sales representatives can discover the high potential sales leads in their territories, and so forth. Sales representatives may seek to determine the top ten products each account will likely buy based on the predictive analysis and the deal sizes if they successfully close, the length of the deal cycle based on the historical trends of similar accounts, and so forth. When sales representatives act on these recommendations, they can broaden their pipeline and increase their chance to meet or exceed quota, thus improving sales productivity, business processes, prospecting, and lead qualification.
Additional use cases for such historical based data may further include: likelihood to close/win and opportunity scoring. For instance, sales representatives and sales managers may benefit from such data as they often have many few deals in their current pipeline and must juggle where to apply their time and attention in any month/quarter. As these sales professionals approach the end of the sales period, the pressure to meet their quote is of significant importance. Opportunity scoring can assist with ranking the opportunities in the pipeline based on the probability of such deals to close, thus improving the overall effectiveness of these sales professionals.

Data sources may include such data as: Comments, sales activities logged, standard field numbers for activities (e.g., events, log a call, tasks etc.), C-level customer contacts, decision maker contacts, close dates, standard field numbers for times the close date has pushed, opportunity competitors, standard field opportunities, competitive assessments, executive sponsorship, standard field sales team versus custom field sales team as well as the members of the respective teams, chatter feed and social network data for the individuals involved, executive sponsor involved in a deal, DSRs (Deal Support Requests), and other custom fields.

Historical based data can be useful to Veritable’s predictive capabilities for generating metrics such as Next Likelihood Purchase (NLP) and opportunity whitespace for sales reps and sales managers. For instance, a sales rep or sales manager responsible for achieving quarterly sales targets will undoubtedly be interested in: which types of customers are buying which products, which prospects most resemble existing customers; are the right products being offered to the right customer at the right price; what more can we sell to my customer to increase the deal size, and so forth. Looking at historical data of things that similar customers have purchased to uncover selling trends, and using such metrics yields valuable insights to make predictions about what customers will buy next, thus improving sales productivity and business processes.

Another capability provided to end users is to provide customer references on behalf of sales professionals and other interested parties. When sales professionals require customer references for potential new business leads they often spend significant time searching through and piecing together such information from CRM sources such as custom applications, intranet sites, or reference data captured in their databases. However, the Veritable core and associated use case GUIs can provide key information to these sales professionals. For instance, the application can provide data including that is grouped according to industry, geography, size, similar product footprint, and so forth, as well as provide in one place what reference assets are available for those customer references, such as customer success stories, videos, best practices, which reference customers are available to chat with a potential buyer, customer reference information grouped according to the contact person’s role, such as CIO, VP of sales, etc., which reference customers have been over utilized and thus may not be good candidate references at this time, who are the sales representatives or account representatives for those reference customers at the present time or at any time in the past, who is available internal to an organization to reach or make contact with the reference customer, and so forth. This type of information is normally present in database systems but is not organized in such a way and is extremely labor intensive to retrieve, however, Veritable’s analysis can identify such relationships and hidden structure in the data which may then be retrieved and displayed by specialized GUI interfaces for end-users. Additionally, the functionality can identify the most ideal or the best possible reference customer among many based on a predictive analysis which builds the details of a reference customer into a probability to win/close the opportunity, which is data wholly unavailable from conventional systems.

In other embodiments, filter elements are provided to the user to narrow or limit the search according to desired criteria, such as industry, geography, deal size, products in play, etc. Such functionality thus aids sales professionals with improving sales productivity and business processes.

According to other embodiments, functionality is provided to predict forecast adjustments on behalf of sales professionals. For instance, businesses commonly have a system of sales forecasting as part of their critical management strategy. Yet, such forecasts are, by nature, inexact. The difficulty is knowing which direction such forecasts are wrong and then turning that understanding into an improved picture of how the business is doing. Veritable’s functionality can improve such forecasting using existing data of a customer organization including existing forecasting data. For instance, applying Veritable’s analysis to past forecast to the business can aid in trending and with improving existing forecasts in the future which have yet to be realized. Sales managers are often asked to provide their judgment or adjustment on forecasting data for their respective sales representatives which requires such sales managers to aggregate their respective sales representatives’ individual forecasts. This is a labor intensive process which tends to induce error. Sales managers are intimately familiar with their representatives’ deals and they spend time reviewing them on a periodic basis as part of a pipeline assessment. Improved forecasting results can aid such managers with rendering improved judgments and assessments as well as help with automating the aggregating function which is often carried out manually or using inefficient tools, such as spreadsheets, etc.

In such an embodiment, Veritable’s analysis function mines past forecast trends by the sales representatives for relationships and causations such as forecast versus quota versus actual for the past eight quarters or other time period, and then provides a recommended judgment and/or adjustment that should be applied to the current forecast. By leveraging the analytical assessment at various levels of the forecast hierarchy, organizations can reduce the variance between individual sales representative’s stipulated quotas, forecasts, and actuals, over a period of time, thereby narrowing deltas between forecast and realized sales via improved forecast accuracy.

Additional functionality enables use case GUI interfaces to render a likelihood to renew an opportunity or probability of retention for an opportunity by providing a retention score. Such functionality is helpful to sales professionals as such metrics can influence where a salesperson’s time and resources are best spent so as to maximize revenue.

FIG. 21A provides a chart depicting prediction completeness versus accuracy.

FIG. 21B provides a chart depicting an opportunity confidence breakdown.

FIG. 21C provides a chart depicting an opportunity win prediction.

FIG. 22A provides a chart depicting predictive relationships for opportunity scoring.
FIG. 22B provides another chart depicting predictive relationships for opportunity scoring.

FIG. 22C provides another chart depicting predictive relationships for opportunity scoring.

Unbounded Categorical Data types model categorical columns where new values that are not found in the dataset can show up. For example, most opportunities will be replacing one of a handful of common existing systems, such as an Oracle implementation, but a new opportunity might be replacing a new system which has not been seen in the data ever before.

FIG. 1 depicts an alternative exemplary architectural overview of the environment in which embodiments may operate. In particular, there are depicted multiple customer organizations 305A, 305B, and 305C. Obviously, there may be many more customer organizations than those depicted. In the depicted embodiment, each of the customer organizations 305A-C includes at least one client device 306A, 306B, and 306C. A user may be associated with such a client device, and may further initiate requests to the host organization 310 which is connected with the various customer organizations 305A-C and client devices 306A-C via network 325 (e.g., such as via the public Internet), thus establishing a relationship between the cloud based services provider and the customer organizations.

The client devices 306A-C each individually transmit request packets 316 to the remote host organization 310 via the network 325. The host organization 310 may responsively send response packets 315 to the originating customer organization to be received via the respective client devices 306A-C. Such interactions thus establish the communications necessary to transmit and receive information in fulfillment of the described embodiments on behalf of each of the customer organizations and the host organization 310 providing the cloud based computing services including access to the Veritable functionality described.

Within host organization 310 is a request interface 375 which receives the packet requests 315 and other requests from the client devices 306A-C and facilitates the return of response packets 316. Further depicted is a PreQL query interface 380 which operates to query the predictive database 350 in fulfillment of such request packets from the client devices 306A-C, for instance, issuing API calls for PreQL structure query terms such as “PREDICT,” “RELATED,” “SIMILAR,” and “GROUP.” Also available are the API calls for “UPLOAD” and “ANALYZE,” so as to upload new data sets or define datasets to the predictive database 350 and trigger the Veritable core 390 to instantiate analysis of such data. Server side application 385 may operate cooperatively with the various client devices 306A-C. Veritable core 390 includes the necessary functionality to implement the embodiments described herein.

FIG. 2 illustrates a block diagram of an example of an environment 210 in which an on-demand database service might be used. Environment 210 may include user systems 212, network 214, system 216, processor system 217, application platform 218, network interface 220, tenant data storage 222, system data storage 224, program code 226, and process space 228. In other embodiments, environment 210 may not have all of the components listed and/or may have other elements instead of, or in addition to, those listed above.

Environment 210 is an environment in which an on-demand database service exists. User system 212 may be any machine or system that is used by a user to access a database user system. For example, any of user systems 212 can be a handheld computing device, a mobile phone, a laptop computer, a work station, and/or a network of computing devices. As illustrated in FIG. 2 (and in more detail in FIG. 3) user systems 212 might interact via a network 214 with an on-demand database service, which is system 216.

An on-demand database service, such as system 216, is a database system that is made available to outside users that do not need to necessarily be concerned with building and/or maintaining the database system, but instead may be available for their use when the users need the database system (e.g., on the demand of the users). Some on-demand database services may store information from one or more tenants stored into tables of a common database image to form a multi-tenant database system (MTS). Accordingly, “on-demand database service 216” and “system 216” is used interchangeably herein. A database image may include one or more database objects. A relational database management system (RDMS) or the equivalent may execute storage and retrieval of information against the database object(s). Application platform 218 may be a framework that allows the applications of system 216 to run, such as the hardware and/or software, e.g., the operating system. In an embodiment, on-demand database service 216 may include an application platform 218 that enables creation, managing and executing one or more applications developed by the provider of the on-demand database service, users accessing the on-demand database service via user systems 212, or third party application developers accessing the on-demand database service via user systems 212.

The users of user systems 212 may differ in their respective capacities, and the capacity of a particular user system 212 might be entirely determined by permissions (permission levels) for the current user. For example, where a salesperson is using a particular user system 212 to interact with system 216, that user system has the capacities allotted to that salesperson. However, while an administrator is using that user system to interact with system 216, that user system has the capacities allotted to that administrator. In systems with a hierarchical role model, users at one permission level may have access to applications, data, and database information accessible by a lower permission level user, but may not have access to certain applications, database information, and data accessible by a user at a higher permission level. Thus, different users will have different capabilities with regard to accessing and modifying application and database information, depending on a user’s security or permission level.

Network 214 is any network or combination of networks of devices that communicate with one another. For example, network 214 can be any one or any combination of a LAN (local area network), WAN (wide area network), telephone network, wireless network, point-to-point network, star network, token ring network, hub network, or other appropriate configuration. As the most common type of computer network in current use is a TCP/IP (Transfer Control Protocol and Internet Protocol) network, such as the global internetwork of networks often referred to as the “Internet” with a capital “I,” that network will be used in many of the examples herein. However, it is understood that the networks that the claimed embodiments may utilize are not so limited, although TCP/IP is a frequently implemented protocol.

User systems 212 might communicate with system 216 using TCP/IP and, at a higher network level, use other common Internet protocols to communicate, such as HTTP,
FTP, AFS, WAP, etc. In an example where HTTP is used, user system 212 might include an HTTP client commonly referred to as a “browser” for sending and receiving HTTP messages to and from an HTTP server at system 216. Such an HTTP server might be implemented as the sole network interface between system 216 and network 214, but other techniques might be used as well or instead. In some implementations, the interface between system 216 and network 214 includes load sharing functionality, such as round-robin HTTP request distributor to balance loads and distribute incoming HTTP requests evenly over a plurality of servers. At least as for the users that are accessing that server, each of the plurality of servers has access to the MTS’ data; however, other alternative configurations may be used instead.

[0283] In one embodiment, system 216, shown in FIG. 2, implements a web-based customer relationship management (CRM) system. For example, in one embodiment, system 216 includes application servers configured to implement and execute CRM software applications as well as provide related data, code, forms, web pages and other information to and from user systems 212 and to store to, and retrieve from, a database system related data, objects, and Web page content. In a multi-tenant system, data for multiple tenants may be stored in the same physical database object, however, tenant data is typically arranged so that data of one tenant is logically separate from that of other tenants so that one tenant does not have access to another tenant’s data, unless such data is expressly shared. In certain embodiments, system 216 implements applications other than, or in addition to, a CRM application. For example, system 216 may provide tenant access to multiple hosted (standard and custom) applications, including a CRM application. User or third party developer applications, which may or may not include CRM, may be supported by the application platform 218, which manages creation, storage of the applications into one or more database objects and executing of the applications in a virtual machine in the process space of the system 216.

[0284] One arrangement for elements of system 216 is shown in FIG. 2, including a network interface 220, application platform 218, tenant data storage 222 for tenant data 223, system data storage 224 for system data 225 accessible to system 216 and possibly multiple tenants, program code 226 for implementing various functions of system 216, and a process space 228 for executing MTS system processes and tenant-specific processes, such as running applications as part of an application hosting service. Additional processes that may execute on system 216 include database indexing processes.

[0285] Several elements in the system shown in FIG. 2 include conventional, well-known elements that are explained only briefly here. For example, each user system 212 may include a desktop personal computer, workstation, laptop, PDA, cell phone, or any wireless access protocol (WAP) enabled device or any other computing device capable of interfacing directly or indirectly to the Internet or other network connection. User system 212 typically runs an HTTP client, e.g., a browsing program, such as Microsoft’s Internet Explorer browser, a Mozilla or Firefox browser, an Opera, or a WAP-enabled browser in the case of a smartphone, tablet, PDA or other wireless device, or the like, allowing a user (e.g., subscriber of the multi-tenant database system) of user system 212 to access, process and view information, pages and applications available to it from system 216 over network 214. Each user system 212 also typically includes one or more user interface devices, such as a keyboard, a mouse, trackball, touch pad, track screen, pen or the like, for interacting with a graphical user interface (GUI) provided by the browser on a display (e.g., a monitor screen, LCD display, etc.) in conjunction with pages, forms, applications and other information provided by system 216 or other systems or servers. For example, the user interface device can be used to access data and applications hosted by system 216, and to perform searches on stored data, and otherwise allow a user to interact with various GUI pages that are presented to a user. As discussed above, embodiments are suitable for use with the Internet, which refers to a specific global internetwork of networks. However, it is understood that other networks can be used instead of the Internet, such as an intranet, an extranet, a virtual private network (VPN), a non-TCP/IP based network, any LAN or WAN or the like.

[0286] According to one embodiment, each user system 212 and all of its components are operator configurable using applications, such as a browser, including computer code run using a central processing unit such as an Intel Pentium® processor or the like. Similarly, system 216 (and additional instances of an MTS, where more than one is present) and all of their components might be operator configurable using application(s) including computer code to run using a central processing unit such as processor system 217, which may include an Intel Pentium® processor or the like, and/or multiple processor units.

[0287] According to one embodiment, each system 216 is configured to provide web pages, forms, applications, data and media content to user (client) systems 212 to support the access by user systems 212 as tenants of system 216. As such, system 216 provides security mechanisms to keep each tenant’s data separate unless the data is shared. If more than one MTS is used, they may be located in close proximity to one another (e.g., in a server farm located in a single building or campus), or they may be distributed at locations remote from one another (e.g., one or more servers located in city A and one or more servers located in city B). As used herein, each MTS may include one or more logically and/or physically connected servers distributed locally or across one or more geographic locations. Additionally, the term “server” is meant to include a computer system, including processing hardware and process space(s), and an associated storage system and database application (e.g., OODBMS or RDBMS) as is well known in the art. It is understood that “server system” and “server” are often used interchangeably herein. Similarly, the database object described herein can be implemented as single databases, a distributed database, a collection of distributed databases, a database with redundant online or offline backups or other redundancies, etc., and might include a distributed database or storage network and associated processing intelligence.

[0288] FIG. 3 illustrates a block diagram of an embodiment of elements of FIG. 2 and various possible interconnections between these elements. FIG. 3 also illustrates environment 210. However, in FIG. 3, the elements of system 216 and various interconnections in an embodiment are further illustrated. FIG. 3 shows that user system 212 may include a processor system 212A, memory system 212B, input system 212C, and output system 212D. FIG. 3 shows network 214 and system 216. FIG. 3 also shows that system 216 may include tenant data storage 222, tenant data 223, system data storage 224, system data 225. User Interface (UI) 330, Application Program Interface (API) 332 (e.g., a PL/SQL or JSON
API), PL/SQL 334, save routines 336, application setup mechanism 338, applications servers 300, system process space 302, tenant process spaces 304, tenant management process space 310, user storage 312, and application metadata 316. In other embodiments, environment 210 may not have the same elements as those listed above and/or may have other elements instead of, or in addition to, those listed above.

User system 212, network 214, system 216, tenant data storage 222, and system data storage 224 were discussed above in FIG. 2. As shown by FIG. 3, system 216 may include a network interface 220 (of FIG. 2) implemented as a set of HTTP application servers 300, an application platform 218, tenant data storage 222, and system data storage 224. Also shown is system process space 302, including individual tenant process spaces 304 and a tenant management process space 310. Each application server 300 may be configured to tenant data storage 222 and the tenant data 223 therein, and system data storage 224 and the system data 225 therein to serve requests of user systems 212. The tenant data 223 might be divided into individual tenant storage areas 312, which can be either a physical arrangement and/or a logical arrangement of data. Within each tenant storage area 312, user storage 314 and application metadata 316 might be similarly allocated for each user. For example, a copy of a user's most recently used (MRU) items might be stored to user storage 314. Similarly, a copy of MRU items for an entire organization that is a tenant might be stored to tenant storage area 312. A UI 330 provides a user interface and an API 332 (e.g., a PreQL or JSON API) provides an application programmer interface to system 216 resident processes to users and/or developers at user systems 212. The tenant data and the system data may be stored in various databases, such as one or more Oracle™ databases.

Application platform 218 includes an application setup mechanism 338 that supports application developers’ creation and management of applications, which may be saved as metadata into tenant data storage 222 by save routines 336 for execution by subscribers as one or more tenant process spaces 304 managed by tenant management process space 310 for example. Invocations to such applications may be coded using PL/SQL 334 that provides a programming language style interface extension to API 332 (e.g., a PreQL or JSON API). Invocations to applications may be detected by one or more system processes, which manages retrieving application metadata 316 for the subscriber making the invocation and executing the metadata as an application in a virtual machine.

Each application server 300 may be communicably coupled to database systems, e.g., having access to system data 225 and tenant data 223, via a different network connection. For example, one application server 300, might be coupled via the network 214 (e.g., the Internet), another application server 300, might be coupled via a direct network link, and another application server 300, might be coupled by yet a different network connection. Transfer Control Protocol and Internet Protocol (TCP/IPv4) are typical protocols for communicating between application servers 300 and the database system. However, it will be apparent to one skilled in the art that other transport protocols may be used to optimize the system depending on the network interconnect used.

In certain embodiments, each application server 300 is configured to handle requests for any user associated with any organization that is a tenant. Because it is desirable to be able to add and remove application servers from the server pool at any time for any reason, there is preferably no server affinity for a user and/or organization to a specific application server 300. In one embodiment, therefore, an interprocess system implementing a load balancing function (e.g., an F5 Big-IP load balancer) is communicably coupled between the application servers 300 and the user systems 212 to distribute requests to the application servers 300. In one embodiment, the load balancer uses a least connections algorithm to route user requests to the application servers 300. Other examples of load balancing algorithms, such as round robin and observed response time, also can be used. For example, in certain embodiments, three consecutive requests from the same user may hit three different application servers 300, and three requests from different users may hit the same application server 300. In this manner, system 216 is multi-tenant, in which system 216 handles storage of, and access to, different objects, data and applications across disparate users and organizations.

As an example of storage, one tenant might be a company that employs a sales force where each salesperson uses system 216 to manage their sales process. Thus, a user might maintain contact data, leads data, customer follow-up data, performance data, goals and progress data, etc., all applicable to that user’s personal sales process (e.g., in tenant data storage 222). In an example of an MTS arrangement, since all of the data and the applications to access, view, modify, report, transmit, calculate, etc., can be maintained and accessed by a user system having nothing more than network access, the user can manage his or her sales efforts and cycles from any of many different user systems. For example, if a salesperson is visiting a customer and the customer has Internet access in their lobby, the salesperson can obtain critical updates as to that customer while waiting for the customer to arrive in the lobby.

While each user’s data might be separate from other users’ data regardless of the employers of each user, some data might be organization-wide data shared or accessible by a plurality of users or all of the users for a given organization that is a tenant. Thus, there might be some data structures managed by system 216 that are allocated at the tenant level, while other data structures might be managed at the user level. Because an MTS might support multiple tenants including possible competitors, the MTS may have security protocols that keep data, applications, and application use separate. Also, because many tenants may opt for access to an MTS rather than maintain their own system, redundancy, up-time, and backup are additional functions that may be implemented in the MTS. In addition to user-specific data and tenant specific data, system 216 might also maintain system level data usable by multiple tenants or other data. Such system level data might include industry reports, news, postings, and the like that are sharable among tenants.

In certain embodiments, user systems 212 (which may be client systems) communicate with application servers 300 to request and update system-level and tenant-level data from system 216 that may require sending one or more queries to tenant data storage 222 and/or system data storage 224. System 216 (e.g., an application server 300 in system 216) automatically generates one or more SQL statements or PreQL statements (e.g., one or more SQL or PreQL queries respectively) that are designed to access the desired information. System data storage 224 may generate query plans to access the requested data from the database.
Each database can generally be viewed as a collection of objects, such as a set of logical tables, containing data fitted into predefined categories. A "table" is one representation of a data object, and may be used herein to simplify the conceptual description of objects and custom objects as described herein. It is understood that "table" and "object" may be used interchangeably herein. Each table generally contains one or more data categories logically arranged as columns or fields in a viewable schema. Each row or record of a table contains an instance of data for each category defined by the fields. For example, a CRM database may include a table that describes a customer with fields for basic contact information such as name, address, phone number, fax number, etc. Another table might describe a purchase order, including fields for information such as customer, product, sale price, date, etc. In some multi-tenant database systems, standard entity tables might be provided for use by all tenants. For CRM database applications, such standard entities might include tables for Account, Contact, Lead, and Opportunity data, each containing pre-defined fields. It is understood that the word "entity" may also be used interchangeably herein with "object" and "table."

In some multi-tenant database systems, tenants may be allowed to create and store custom objects, or they may be allowed to customize standard entities or objects, for example by creating custom fields for standard objects, including custom index fields. In certain embodiments, for example, all custom entity data rows are stored in a single multi-tenant physical table, which may contain multiple logical tables per organization. It is transparent to customers that their multiple "tables" are in fact stored in one large table or that their data may be stored in the same table as the data of other customers.

Fig. 4 illustrates a diagrammatic representation of a machine 400 in the exemplary form of a computer system, in accordance with one embodiment, within which a set of instructions, for causing the machine/computer system 400 to perform any one or more of the methodologies discussed herein, may be executed. In alternative embodiments, the machine may be connected (e.g., networked) to other machines in a Local Area Network (LAN), an intranet, or the public Internet. The machine may operate in the capacity of a server or a client machine in a client-server network environment, as a peer machine in a peer-to-peer (or distributed) network environment, as a server or series of servers within an on-demand service environment. Certain embodiments of the machine may be in the form of a personal computer (PC), a tablet PC, a set-top box (STB), a Personal Digital Assistant (PDA), a cellular telephone, a web appliance, a server, a network router, switch or bridge, computing system, or any machine capable of executing a set of instructions (sequential or otherwise) that specify actions to be taken by that machine. Further, while only a single machine is illustrated, the term "machine" shall also be taken to include any collection of machines (e.g., computers) that individually or jointly execute a set (or multiple sets) of instructions to perform any one or more of the methodologies discussed herein.

The exemplary computer system 400 includes a processor 402, a main memory 404 (e.g., read-only memory (ROM), flash memory, dynamic random access memory (DRAM) such as synchronous DRAM (SDRAM) or Rambus DRAM (RDRAM), etc., static memory such as flash memory, static random access memory (SRAM), volatile but high-data rate RAM, etc.), and a secondary memory 418 (e.g., a persistent storage device including hard disk drives and a persistent database and/or a multi-tenant database implementation), which communicate with each other via a bus 430. Main memory 404 includes stored indices 424, an analysis engine 423, and a PreQL API 425. Main memory 404 and its sub-emolements are operable in conjunction with processing logic 426 and processor 402 to perform the methodologies discussed herein. The computer system 400 may additionally or alternatively embody the server side elements as described above.

Processor 402 represents one or more general-purpose processing devices such as a microprocessor, central processing unit, or like. More particularly, the processor 402 may be a complex instruction set computing (CISC) microprocessor, reduced instruction set computing (RISC) microprocessor, very long instruction word (VLIW) microprocessor, processor implementing other instruction sets, or processors implementing a combination of instruction sets. Processor 402 may also be one or more special-purpose processing devices such as an application specific integrated circuit (ASIC), a field programmable gate array (FPGA), a digital signal processor (DSP), network processor, or like. Processor 402 is configured to execute the processing logic 426 for performing the operations and functionality which is discussed herein.

The computer system 400 may further include a network interface card 408. The computer system 400 also may include a user interface 410 (such as a video display unit, a liquid crystal display (LCD), or a cathode ray tube (CRT)), an alphanumeric input device 412 (e.g., a keyboard), a cursor control device 414 (e.g., a mouse), and a signal generation device 416 (e.g., an integrated speaker). The computer system 400 may further include peripheral device 436 (e.g., wireless or wired communication devices, memory devices, storage devices, audio processing devices, video processing devices, etc.).

The secondary memory 418 may include a non-transitory machine-readable or computer readable storage medium 431 on which is stored one or more sets of instructions (e.g., software 422) embodying any one or more of the methodologies or functions described herein. The software 422 may also reside, completely or at least partially, within the main memory 404 and/or within the processor 402 during execution thereof by the computer system 400, the main memory 404 and the processor 402 also constituting machine-readable storage media. The software 422 may further be transmitted or received over a network 420 via the network interface card 408.

Fig. 5A depicts a tablet computing device and a hand-held smartphone each having a circuitry integrated therein as described in accordance with the embodiments.

Fig. 5B is a block diagram of an embodiment of tablet computing device, a smart phone, or other mobile device in which touchscreen interface connectors are used.

While the subject matter disclosed herein has been described by way of example and in terms of the specific embodiments, it is to be understood that the claimed embodiments are not limited to the explicitly enumerated embodiments disclosed. To the contrary, the disclosure is intended to cover various modifications and similar arrangements as are apparent to those skilled in the art. Therefore, the scope of the appended claims are to be accorded the broadest interpretation so as to encompass all such modifications and similar arrangements. It is to be understood that the above description
is intended to be illustrative, and not restrictive. Many other embodiments will be apparent to those of skill in the art upon reading and understanding the above description. The scope of the disclosed subject matter is therefore to be determined in reference to the appended claims, along with the full scope of equivalents to which such claims are entitled.

What is claimed is:

1. A method comprising:
   receiving a dataset in tabular form of columns and rows;
   triggering Veritable core analysis of the dataset received;
   identifying hidden structure of the dataset via the Veritable core analysis, the hidden structure including one or more of relationships and causations in the data for which such relationships and causations are not pre-defined by the dataset;
   storing the analyzed dataset including the hidden structure having the one or more of relationships and causations as a queryable model.

2. The method of claim 1, further comprising:
   querying the stored queryable model for predictive analysis.

3. The method of claim 1, further comprising one or more of:
   issuing a ProQL structure query against the queryable model, the ProQL structure comprising one of:
   a PREDICT term;
   a RELATED term;
   a SIMILAR term; and
   a GROUP term.

4. The method of claim 1, further comprising:
   providing a graphical user interface to the Veritable core as a cloud based computing service.

5. The method of claim 1, further comprising:
   providing a perceptible GUI as a cloud based service, the perceptible GUI accepting as input a data source within the predictive database;
   presenting at the perceptible GUI, a table representing the data source within the predictive database, wherein the table has a plurality of non-null values and a plurality of null values;
   providing a graphical slider mechanism at the perceptible GUI, wherein the graphical slider mechanism is manipulatable at a client device to increase and decrease a percentage of predictive fill for the null values of the table;
   populating null values of the table at the graphical perceptible GUI responsive to the graphical slider mechanism registering an increase in value to populate by the client device.

6. The method of claim 5, wherein populating the null values comprises:
   for every null value cell element in the data, retrieving a distribution via Veritable API calls from the Veritable core;
   correlating a percentage fill value registered by the graphical slider mechanism to a necessary confidence threshold to reach the requested percentage fill; and
   populating null values of the table at the graphical perceptible GUI until the percentage fill value is reached by selecting cell elements for which the corresponding distribution has a confidence in excess of the confidence threshold.

7. The method of claim 6, further comprising:
   receiving a 100% fill value request from the graphical slider mechanism;
   populating all null values of the table at the graphical perceptible GUI by degrading required confidence until a predicted result is available for every null value of the table.