Rapid Data Exploration and Visual Data Mining on Relational Data

Gartheeban Ganeshapillai, Joel Brooks, John Guttag
Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology
32 Vassar Street
Cambridge, MA 02139
{garthee, brooksjd, guttag}@mit.edu

ABSTRACT

Exploring and analyzing a large amount of data is becoming increasingly common, and human involvement in the process is often required. The advantage of visual data mining is that it combines the flexibility, creativity, and general knowledge of a human with brute computational power. In this paper, we describe a novel system, a visual data mining framework, called GNoT, that supports interactive knowledge discovery by interconnecting state of the art tools for visualization, relational database management, and machine learning. The system essentially provides the glue connecting these kinds of components, and thus will be able to "ride the wave" of improvements in each of these areas. We demonstrate the tool's utility with a case study on a real-world application.

1. INTRODUCTION

Data mining is the process of extracting useful information from large data sets. Historically, research on data mining has emphasized some combination of machine learning, statistics, and database systems. In recent years, however, data visualization has come to play an ever more important role as the field of visual data mining has grown. Studies suggest visual data mining can be faster and more intuitive than traditional data mining [9]. In this paper, we describe a novel system, called GNoT, that combines data visualization and data analysis tools. It supports a style of interactive discovery in which a user follows the iterative process depicted in Figure 1. This approach to data mining has been shown to be highly effective on moderately sized data sets [9], and we believe will become even more useful in the era of "big data."

GNoT is a visual data mining framework that supports this style of interactive knowledge discovery by interconnecting state of the art tools for visualization, relational database management, and machine learning. For example, the dramatic increase in the size of the data that are mined brings a renewed focus on the database management, and in our work we attempt to benefit from the latest advancements in database systems.

By using an existing relational database management system (RDBMS), GNoT simplifies the process of storing, and accessing the data. The process of filtering, projecting, and formatting data can be done efficiently using a conventional query language (SQL). Similarly, the visualization phase is well-served by the incorporation of a high-level versatile visualization library [4] and its extensions, and the data analytics phase by the incorporation of several external libraries well suited for this purpose. The system essentially provides the glue connecting these kinds of components, and should therefore be able to "ride the wave" of improvements in each of these areas.

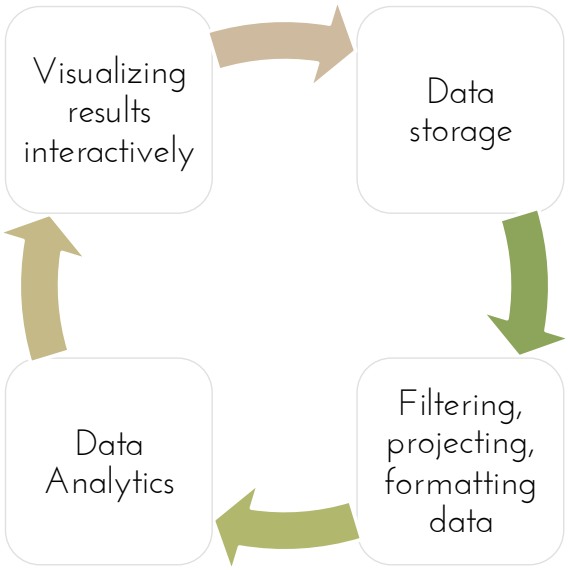


Figure 1: Pipeline of visual data exploration.

GNoT is written in Python (backend) and JavaScript (front-end). It is run as a web server, and visualizations are rendered in the clients' browsers. GNoT makes use of many JavaScript libraries that offer rich user-interactive dynamic visualizations. GNoT supports any PostgresSQL based RDBMS in the backend, and thus makes use of the recent developments on fast, parallel database systems based on a column store architecture such as Greenplum[1] and Vertica [10]. It can also interface to cloud based distributed database systems such as Redshift [15]. The platform is modular, comes with a large set of Visualization types readily available, and new types of visualization libraries can be easily added.

GNoT enables rapid visualization and visual data mining (Figure 2). With GNoT, using an existing module requires neither programming nor software engineering expertise, and extending a module typically involves integrating with a JavaScript library such as D3.

[1]http://www.gopivotal.com/

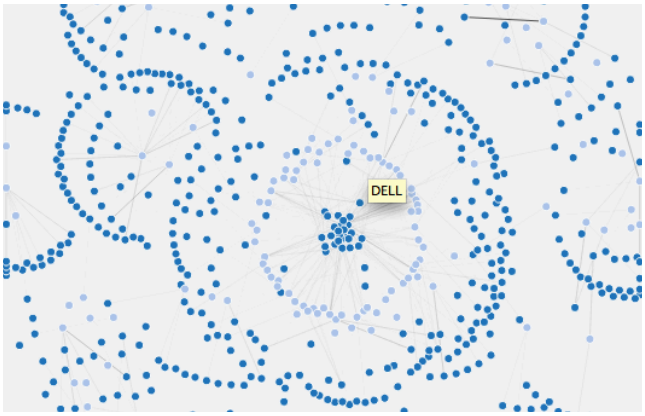


Figure 2: Using GNoT, we are able to quickly explore twitter data on financial news, and try to understand the patterns between the twitter users and the stocks covered by them. Query `|MODULE:explore_graph TABLE:finance.twitter_feed SOURCE:author_id TARGET:stock LIMIT:1000 FIELD: count(*)as n ORDERBY:n desc|` **generates the graph. First, we observe that a set of stocks are exclusively covered by a set of author_ids (twitter users). By clicking on a target node (light blue) we see the node's name (DELL). We can also observe that technology stocks form a cluster that is covered by a group of twitter users who rarely cover anything else.**

We make the following contributions in this work:

- A framework that elegantly interconnects tools from database systems, visualization tools, and machine learning libraries to offer a fluid experience in visual data mining. Our framework is modular, scalable, extensible, and makes use of state of the art tools to perform each of the subtasks.
- A fully functional implementation of the framework built using a column-store database, a set of core modules offering a large subset of the D3 visualization types, and a set of core machine learning modules. It is available at `http://github.com/garthee/gnot`.
- A detailed case study illustrating how the system can be used on a real problem. As part of this case study we introduce a set of modules offering analytics and visualization on geolocation data coupled with other types of data such as time series.

In the rest of the paper, we provide direct links to the demo pages (identified by ➥) whenever they are available.

The remainder of the paper is organized as follows. We first discuss related works. We then discuss the design rationale and the design of the framework. Then we go through a case study of using GNoT in a real-world data exploration application. We conclude with a short discussion and summary of the system.

2. RELATED WORK

We realize that "one size doesn't fit all". GNoT is neither an all-in-one framework nor is it optimized for a single specialized visualization task. Instead, it provides the glue connecting various specialized tools to perform rapid visualization on relational data. This enables us to easily provide the latest machine learning methods to the visual data mining process while making use of the advancements in RDBMS and visualization libraries. Whenever possible, GNoT offloads the underlying tasks to these specialized tools.

2.1 Languages and low-level tools

Many languages provide rich visualization capabilities [6]. For example, Matlab offers a large set of static plots and Matplotlib offers similar capability with Python. There are also similar tools such as Weka [8], GNU plot and ggplot that support visualization within the framework of a language. Because these tools are set within the framework of a programming language, they readily offer integration with machine learning algorithms built in those languages (e.g., Matlab, Weka and R).

There also exists a set of libraries specifically targeting visualization tasks including low-level graphics libraries such as Processing[2] and Raphael[3] [4]. While they offer great flexibility in how visualizations appear, they can be challenging to use for complex visualization tasks. Generally the visualizations generated by these tools lack dynamic controls and user interactions.

2.2 Database systems

There have been significant advancements in relational database systems [14], e.g., column stores, distributed database systems, and map-reduce. Systems like Greenplum and Vertica are providing state of the art database techniques for relational data. Amazon's Redshift offers a distributed database system in the cloud [15]. However, these database systems do not offer any visualization capabilities. Further, they provide only limited support for complex analytics.

Tools such as VQE [7], Visage [12], Tioga-2 [1], and Snaptogether[11] were among the first to provide visualization environments that directly support interactive data exploration on relational data. However they offer only basic set of visualizations such as simple graphs, and are far behind contemporary graphics libraries such as D3 in keeping up with the latest advancements in visualization techniques. Further, they also do not offer any support for complex analytics. Recently, graphics libraries and visualization systems have taken their place.

2.3 Graphics Libraries

There have been many libraries developed in the last few years targeting specific graphical functionality. Flot, icharts, Exhibit, jQuery Visualize, Google Charts, and CartoDB are among those that offer specific types of in-browser visualizations. D3 stands out as a highly versatile generic browser-based visualization library [4]. It provides a close mapping between the data and the desired result, and a greater flexibility in achieving the latter. D3 also offers high-level capability by including a collection of helper modules that sit on top of the kernel library to offer rich visualizations with minimal effort. This capability has been further extended by other D3 based libraries such as Crossfilter[4], Rickshaw[5], and

[2]http://www.processing.org/
[3]http://raphaeljs.com/
[4]http://square.github.io/crossfilter/
[5]http://code.shutterstock.com/rickshaw/

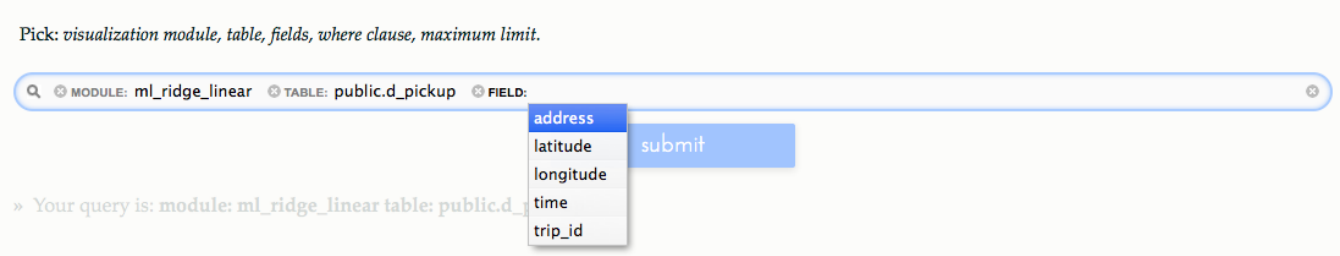


Figure 3: Frontend UI: A query input box with autocomplete and hotkeys makes it easy for users to construct a query specifying table, field, and other options.

NVD3[6]. These libraries often accept the data in a delimited file (CSV or TSV) or as a JSON file.

Although GNoT is not tied to any particular front-end library, the current set of modules depend on D3 or its extensions for the front-end, and acts as middleman to feed the relational data and the results of the analytics performed on the relational data to these libraries in the format that they expect.

2.4 Visualization Systems

There are frameworks supporting the full range of the data mining pipeline. Ranging from IBM Many Eyes [16] to Improvise [17] and Polaris [13] (and its commercial implementation Tableau), they offer vertically integrated systems with a hierarchy of visualization components, tools for analytics, and data storage systems. Users of Improvise and Polaris can use existing components, subclass and extend an existing components, and add new components.

GNoT takes a similar approach, but trades tight integration for efficiency and rapid deployment. With GNoT, using an existing module requires neither programming nor software engineering expertise, and extending a module typically involves integrating a JavaScript library such as D3. Further, instead of providing a vertically integrated monolithic solution, GNoT interconnects state of the art tools for visualization, relational database management, and machine learning.

Recently, cloud based solutions for visual data mining are becoming popular, e.g., Google fusion tables, IBM's Many Eyes, Google Public Data Explorer, and Wolfram Alpha. While they differ in the offerings and the richness of the set of features, these systems typically allow the user to upload a dataset and build a set of visualizations from the data. However, they rarely offer any support to perform complex analytics on the data, and are ill suited for large datasets.

3. DESIGN RATIONALE

GNoT's primary objective is to offer rapid interactive exploration and complex analytics on large relational databases while allowing the user to benefit from the latest developments in visualization techniques, database systems, and machine learning methods for analytics.

To effectively support the stated objective, our framework must meet the following demands:

- **Ease of use:**
  In order to perform the exploration rapidly and interactively, analysts need to be able to create visualizations with relative ease. People working with data are accustomed to SQL queries and tabular data. A SQL like query with a front-end UI (Figure 3) that offers autocomplete to guide the user with input selections allows us to achieve this.

- **Easy entry and flexibility:**
  It should be easy for new users to execute simple task, while at the same time allowing the flexibility for more advance users to integrate modules needed for more advanced tasks. Instead of creating a new graphics language or protocol, we simply extend the SQL syntax. Also because the current set of modules make use of D3, they simply follow D3's protocols.

- **Technology reuse:**
  Technology reuse reduces the foot print of the framework, and allows the system to keep up to date with minimal effort. Our framework minimizes its footprint by bridging a RDBMS with front-end visualization libraries such as D3 and backend libraries such as Scikit to perform complex analytics. By reusing an ecosystem of related components, we offload a major fraction of the subtasks to specialized tools. Thus, we are able to keep up to date with the advancements in each of the subtasks.

- **Extensible and scalable:**
  In order to keep up to date, the system must be extensible. A modular approach that offloads most of the work to external libraries makes the system easily extensible. This approach allows us to replace one component by another to achieve greater functionality or scalability. For instance, in our framework a PostgresSQL based RDBMS is readily replaceable with Greenplum or Redshift. Similarly, the modular approach allows us to update the modules to use a different machine learning library with relative ease.

- **Performance:**
  Since the intent is to provide an interactive exploration tool, performance is critical. Therefore the system should have minimal overhead so that performance is governed by the speed of the individual components (e.g., the RDBMS or the machine learning component).

[6]http://nvd3.org/

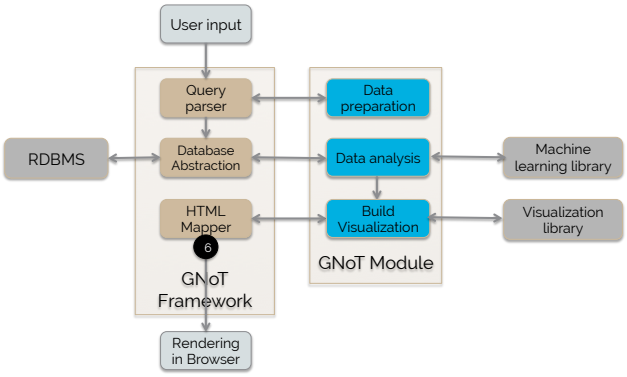
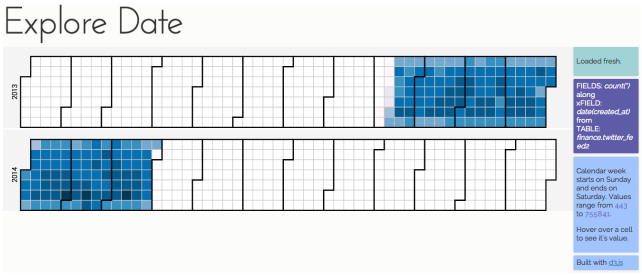
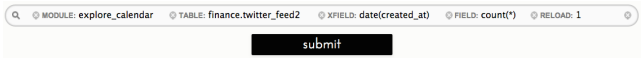


Figure 4: Architecture of GNoT



(a) Visualization Output



(b) User input

Figure 5: To produce a date-based heat map (a) of the data using Explore_calendar is as easy as submitting the query `|MODULE:explore_calendar TABLE:finance.twitter_feed2 XFIELD:date(created_at)FIELD:count(*)|` **from the frontend UI (b).**

4. GNOT

Figure 4 shows the architecture of GNoT. The visualizations are implemented as a series of modules. The framework exposes the functionalities of each module to the user, and mediates interconnection among the user, the RDBMS, and the module. In addition, the framework performs various maintenance tasks such as caching, parsing and validating user inputs, and mapping HTML outputs.

GNoT is a web server implemented on top of Werkzeug[7], and encompasses a query parser, database abstraction, and an output mapper. The database abstraction executes SQL queries generated by other modules in the system and returns the result. It also caches the query output in the file system so that queries are bypassed when the output exists in the file system. Users can overwrite the cache by opting for reload in the input query. The front end is a visual search UI built on top of Visualsearch[8]. The UI populates the options from a list of options specified by individual modules, and the table-field information from the database. The UI allows the user to quickly specify the inputs.

Visualizations are implemented by individual modules. Built-in modules cover the broad range of visualizations available in D3, and using them is as simple as selecting the visualization type in the query box. Figure 5 shows the use of such a module. Note that the input options make use of the SQL functions to format the fields.

Should users find the built-in modules insufficient, they can add additional modules. Adding a new module requires a backend file written in Python and a front end HTML/JS file to liaise with visualization libraries. The choice between these two approaches offers a tradeoff between simplicity and flexibility.

5. MACHINE LEARNING USING GNOT

A major thrust of this work is to allow users to interactively incorporate machine learning models into visual data mining tasks. GNoT has built in modules to support the three most commonly used machine learning tasks: classification (using an SVM), clustering (using k-means), and regression (using a Ridge regression). We use scikit-learn, a Python library to support the machine learning tasks in the built-in machine learning modules. However, the library can be switched for another with relative ease.

Let us walk through two of modules offering machine learning tasks: ML_SVM and ML_K-Means. We assume that the reader is familiar with using support vector machine (SVM) classifiers and the K-means clustering algorithm [5, 3].

5.1 ML_SVM

ML_SVM allows a user to apply a SVM classifier to the data. It learns a classifier separating the positives (+1) from negatives (-1) of a dependent variable using a set of features (independent variables) and validates its accuracy on a test set. ML_SVM allows user to easily construct a model, interactively tune the hyper parameters of the model, and visually analyze the fit of the model on the test set.

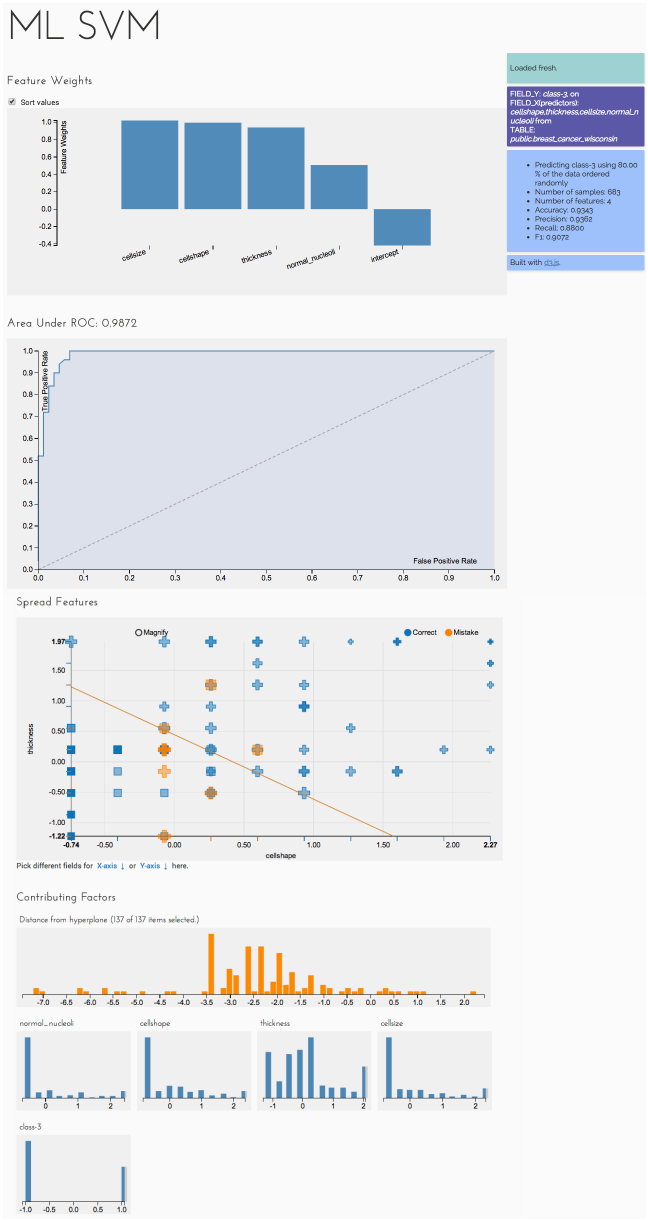
In the user's query, the first field is the dependent variable and the rest of the fields are the independent variables (i.e., the feature vector). The user also specifies the ratio of the data used for training the model and a regularization parameter for the SVM. The module also supports pre-processing and pre-transformation on the independent variables. For instance, the user can use builtin modules to normalize the independent variables by applying whitened PCA or a Z-score transformation, and then transform the data to a quadratic scale to better capture the distribution of the independent variables.

GNoT allows the user to visually investigate the characteristics of the data and the model produced by the SVM. Using the automatically produced visualization, the user can analyze the model fit on the test data by brushing and selecting a value range for each independent variable. The user can then visually examine the corresponding change in the distribution of the other independent variables and the model fit.

In ML_SVM ➥, we demonstrate the application of ML_SVM module of GNoT on the Wisconsin breast cancer dataset [2]. Figure 6 shows the resulting page. The side bar provides the summary of the results. Even with this basic model, we achieve an accuracy of 97% when predicting 20% of the samples by learning the model on the rest. The visualiza-

[7]http//http://werkzeug.pocoo.org/
[8]http://documentcloud.github.io/visualsearch/

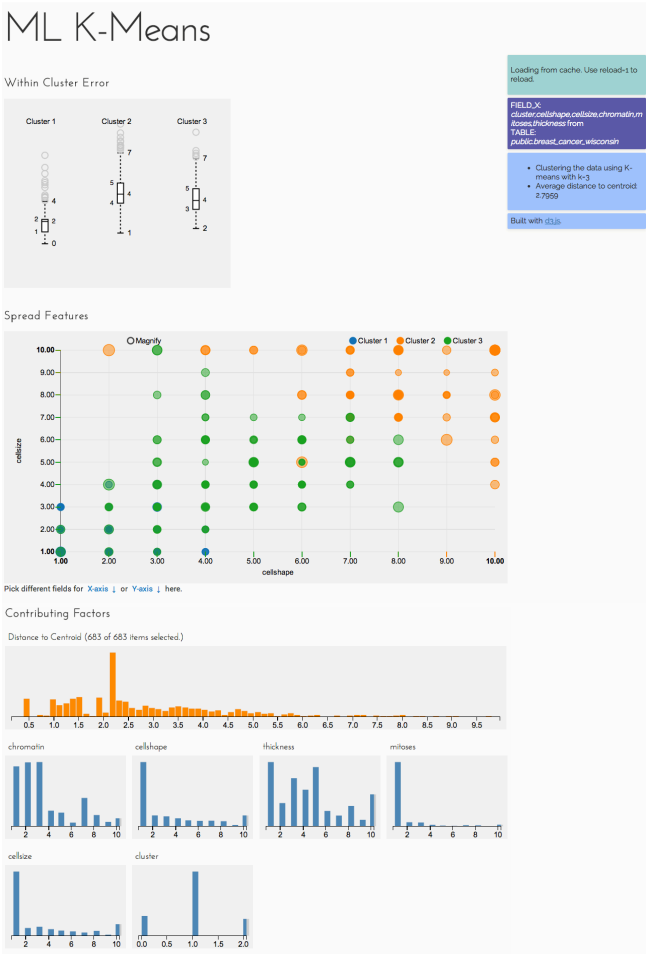
(a) User input



(b) Visualization Output

Figure 6: (a) Query `|MODULE:ml_svm_linear TABLE: public.breast_cancer_wisconsin FIELD:class-3 FIELD :cellshape FIELD:thickness FIELD:cellsize FIELD :normal_nucleoli RATIO:0.8 PRE_PROCESS:Z-Score REGULARIZER:10 |` **produces the visualization. (b) The visualization gives the summary of the results, feature weights of the model, and receiver operating characteristic curve (to show the accuracy of the model on the test data). It also allows the user to interactively examine the fit of the model on the test data by brushing and selecting a value range on each of the independent variable, and corresponding distributions on other variables.**

(a) User input



(b) Visualization Output

Figure 7: (a) Query `|MODULE:ml_kmeans TABLE: public.breast_cancer_wisconsin FIELD:cellshape FIELD:cellsize FIELD:chromatin FIELD:mitoses FIELD :thickness K:3 |` **produces the visualization. (b) The visualization gives the summary of the model, cluster spreads, and the distribution of the clusters against any two fields. It also allows the user to interactively examine the fit of the model by brushing and selecting a value range on each of the fields, and corresponding distributions on other fields.**

tions allow the users to understand the characteristics of the model. The bar chart shows the weights of four features and the intercept (from the linear fit). The next graph shows the accuracy of the model on the test data with the receiver operating characteristic curve and the area under the curve. The scatter plot allows the user to visualize the spread of the samples from the test set on any two dimensions (the space of any two features) and the projection of the separating hyperplane on those two dimensions. The color and the shape of the samples indicate whether they were correctly classified, and the correct labels of the samples respectively. The visualizations on this page are connected to each other. The cross filter allows the user to brush and select a value range of an independent variable. This will update the distributions of the rest of the independent variables and the feature spread. Using cross filter, the user can see the contributing factors for large errors (by filtering by large positive distances from hyperplane): smaller cell sizes stand out as the most difficult to predict.

5.2 ML_K-Means

ML_K-Means allows a user to apply a K-means clustering algorithm to the data. The module also supports preprocessing and pre-transformation on the field values. The user can analyze the model fit on the data by brushing and selecting a value range for each of the fields. The user can also then visually examine the corresponding change in the distribution of the other fields and the model fit.

In ML_K-Means ➥, we demonstrate the application of ML_K-Means on Wisconsin breast cancer dataset [2]. Using the resulting page (Figure 7) we can try different field ranges and their effects in the resulting cluster distributions.

6. USING GNOT: A CASE STUDY

We built GNoT to help with exploring new datasets in solving real-world applications. We now describe an example application to demonstrate its usage and capabilities.

In early 2014, MIT Big Data Initiative at CSAIL together with the City of Boston hosted a Big Data Challenge[9] to gain new insights into how people use all modes of transportation to travel in and around the downtown Boston area. We use the Boston taxi dataset from this challenge.

6.1 Exploring Boston taxi data

Below, we outline how one might use GNoT to explore this dataset, generate hypotheses, and validate them. We use the demonstration site setup at `http://ddmg1.csail.mit.edu:4999` for following expositions. Links to demo pages are identified by ➥ whenever they are available.

- We start with a visualization of the raw data using query `|MODULE:explore_raw TABLE:public.d_pickup2 FIELD:* LIMIT:10|`. From the output of the Explore_Raw module as available at Pickup: Raw ➥, we understand that there are 5 fields: trip_id, time, address, longitude, and latitude.

- Next, we used the Explore_Calendar module to see the distribution of the data over the time span with query `|MODULE:explore_calendar TABLE:public.d_pickup2 XFIELD:date(time)|`. From the output Pickup: Calendar ➥ (Figure 5), we can see that the data spans

[9]http://bigdatachallenge.csail.mit.edu

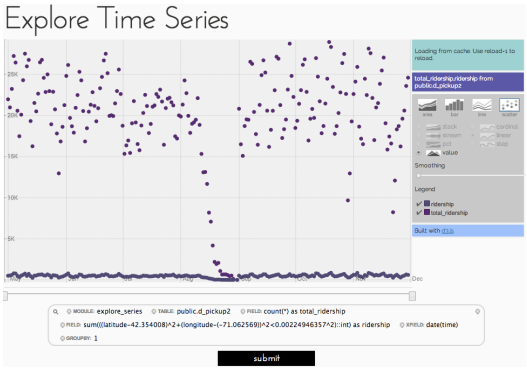


Figure 8: Time series module in GNoT: Comparing the ridership around a location with that of the whole city using GNoT.

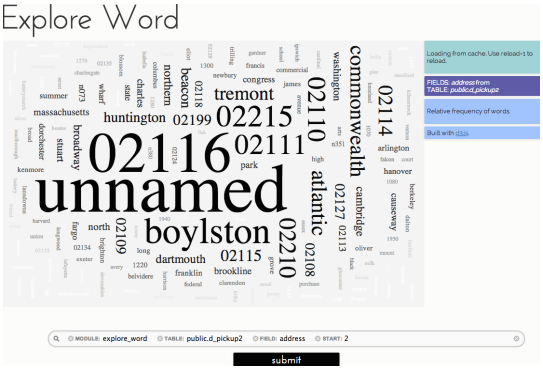


Figure 9: Word module in GNoT: Exploring the popular words in the pickup addresses of taxi rides using GNoT.

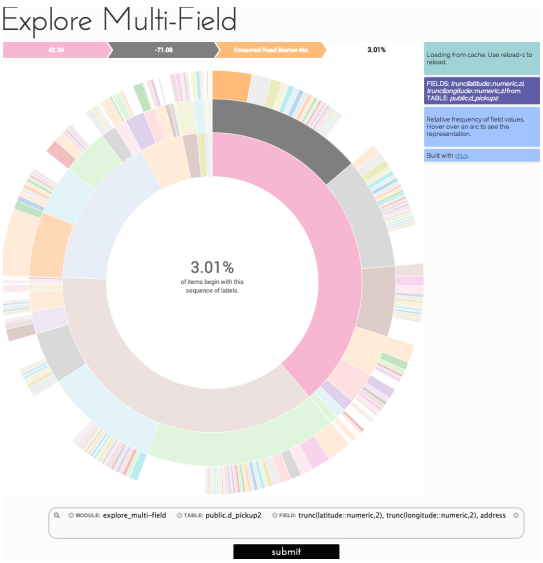


Figure 10: Multi-field module in GNoT: Visualizing the hierarchical split by various features.

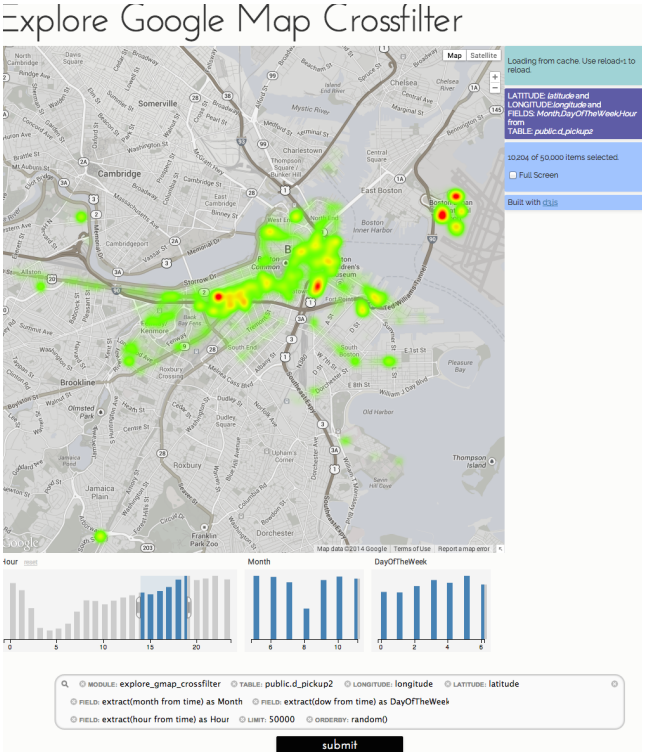


Figure 11: Google maps with crossfilter in GNoT: Visualizing the factors contributing to the ridership with GNoT.

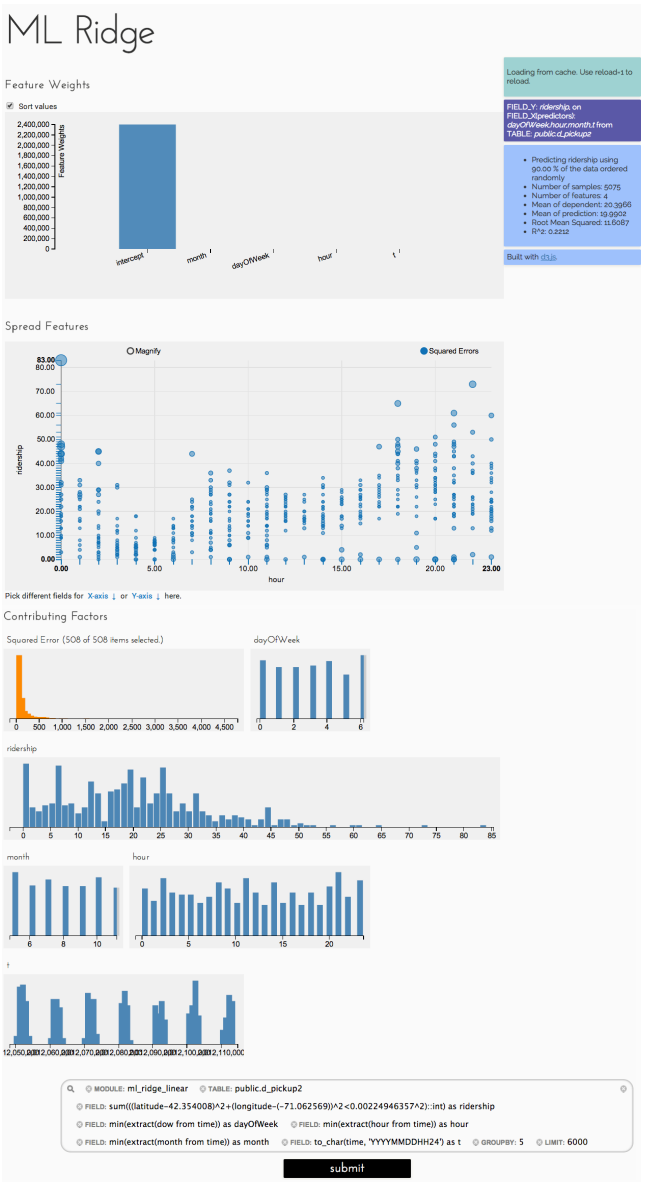


Figure 12: ML_Ridge_Linear module in GNoT: Visualizing the fit of ridge regression predicting the number of pickups in an hour window.

from May 1, 2012 to November 30, 2012, and that the data is missing for about two weeks during the latter part of August.

- We next used the Explore_Diff module to compare the ridership at specific locations to overall ridership across Boston with query `|MODULE:explore_diff TABLE:public.d_pickup2 XFIELD:date(time)FIELD:sum(((latitude-42.354008)^2+(longitude-(-71.062569))^2<0.00224946357^2)::int)as ridership FIELD:count(*) as total_ridership|`.
  In the resulting page Pickup: Diff ➥, we see that the ratio fluctuates hugely.

  We used Explore_Series to get a more detailed view. It is as easy as changing the module option in the query box. In the resulting page Pickup: Series ➥, we have many types of visualizations at our disposal (Figure 8).

- We used the Explore_Word module to see the popular words in the addresses with query `|MODULE:explore_word TABLE:public.d_pickup2 FIELD:address|`. It shows the words sized according to the number of occurrences: Pickup: Word ➥. When we viewed this visualization, the word "Boston" was dominant, not surprising given the data set.

  We therefore asked it to omit "Boston" with query `|MODULE:explore_word TABLE:public.d_pickup2 FIELD:address START:1|`, and got the visualization Pickup: Word2 ➥ (Figure 9). ("Unnamed road" is a road within Logan Airport.)

- We used Explore_Multi-Field to understand which latitude, longitude, and specific address with high ridership in query `|MODULE:explore_multi-field TABLE:public.d_pickup2 FIELD:trunc(latitude::numeric,2), trunc(longitude::numeric,2), address|`. Here, we truncated the coordinates to the second decimal point using SQL itself. The resulting visualization Pickup: Multi-field ➥ is seen in Figure 10. We can see latitude 42.34 has the highest with 34% of the ridership, out of which longitude -71.08 takes the highest ridership at 14%. Within this combination, "unnamed road Boston" (which is part of Boston Logan interna-

tional airport) takes the highest fraction. Altogether, it represents 3% of the total taxi ridership.

- However, making sense of the latitudes and longitudes is easier when they are plotted on a map. We used Explore_Gmap to view them in Google maps with query `|MODULE:explore_gmap TABLE:public.d_pickup2 LATITUDE:trunc(latitude::numeric,3) LONGITUDE:trunc(longitude::numeric,3) FIELD:count(*)|` and got Pickup: Gmap ➥.

  We also visualized the changes in the ridership against time with query `|MODULE:explore_gmap TABLE:public.d_pickup2 LATITUDE:trunc(latitude::numeric,3) LONGITUDE:trunc(longitude::numeric,3)XFIELD:date(time)FIELD:count(*)|` and got Pickup: Gmap2 ➥.

- In order to gain a deeper insight into the ridership based on hour of the day, month of the year, and day of the week, we used Explore_Gmap_Crossfilter in query `|MODULE:explore_gmap_cross_filter TABLE:public.d_pickup2 LATITUDE:latitude LONGITUDE:longitude FIELD:extract(month from time) as Month FIELD:extract(dow from time)as Day FIELD:extract(hour from time)as Hour LIMIT:50000 ORDERBY:random()|` and got Pickup: Gmap Crossfilter ➥ (Figure 11). Using the brushing feature, we can learn that between 10 PM and 12 AM taxi rides peak around Boston University, Boylston Street/Prudential Tower, and Terminal E at Boston Logan international airport.

6.1.1 *Machining Learning Models*

First, we used k-means to perform unsupervised clustering of the coordinates with query `|MODULE:ml_kmeans TABLE:public.d_pickup2 FIELD:latitude, longitude LIMIT:1000 K:5|`. Arbitrarily, we chose to partition the coordinates into 5 clusters Pickup: K-means ➥. We observe the segments formed by dividing the city into 5 explainable regions. We could also interactively explore different number of segments by adjusting k.

What if we add the day of the week as the third dimension? Since the range of the day of the week is not similar to the range of latitude and longitude (i.e., they have different units), we apply Z-score normalization before applying K-means in the query `|MODULE:ml_kmeans TABLE:public.d_pickup2 FIELD:latitude, longitude, extract(dow from time)PRE_PROCESS:Z-Score LIMIT:1000 K:5|`. Here, we see a totally different pattern emerging Pickup: K-means2 ➥.

Now, let's look at a problem of predicting the number of pickups within 250 meters of a location (e.g., (-71.057114, 42.343365)) within a given time window. Before building a complex model, we can use GNoT to visually explore the data in order to test the viability of the task and identify useful feature combination.

We used the ML_Ridge_Linear module to explore the accuracy achievable in predicting the ridership in an hour with only three features: day of the week, hour of the day, and month with query `|MODULE:ml_ridge_linear TABLE:public.d_pickup2 FIELD:sum(((latitude-42.354008)^2+(longitude-(-71.062569))^2<0.00224946357^2)::int)as ridership FIELD:min(extract(dow from time))as dayOfWeek FIELD:min(extract(hour from time))as hour FIELD:min(extract(month from time))as month FIELD:to_char(time, 'YYYYMMDDHH24')as t GROUPBY:5 LIMIT:6000|`.

We can see the resulting page Pickup: Ridge Linear ➥ in Figure 12. Even with this basic model, we achieve a coefficient of determination ($R^2 = 0.22$) when predicting 10% of the time windows by learning the model on the rest. Using GNoT, we can also try different feature combinations, feature transformations such as applying interactions and quadratic transformations on the features, and pre-processing such as applying Z-score or PCA transformation.

6.2 Custom solution with GNoT

The visualizations built with GNoT to assist with rapid visual data mining, called *Boston Rides* ➥[10], won the first prize in the competition. *Boston Rides* is a customized version of GNoT where the user's queries are hardcoded, and they are guided through a predefined exploration path.

Boston Rides allows the user to explore the data through 6 different visualization types: hotspots for pickups, daily and hourly variations of ridership (Figure 13), popular intra-city routes, factors contributing to variations in pickups, and machine learning generated models that predict the number of pickups in an hour window using various features.

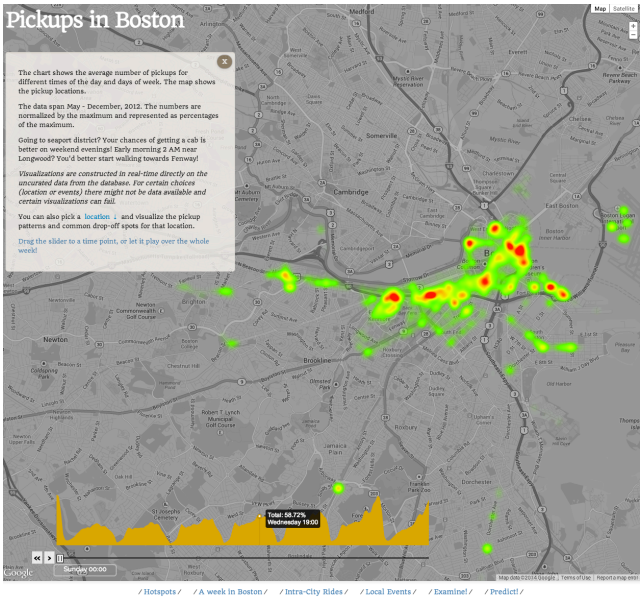


Figure 13: Daily and hourly variation in ridership as visualized in Boston rides with GNoT. The module makes use of D3 and the Google maps API to construct the visualization. The popup box in the figure contains the guide.

6.3 Discussion

One of the biggest advantages with GNoT is that it offers the choice between the ease of use and flexibility.

In the first part of the case study (Section 6.1), we used existing modules. This requires neither programming nor software engineering expertise, and allows us to rapidly explore the data with great ease. Each query requires very

[10] http://bostonrides.info

minimal input from the user, and the queries are intuitive as they follow the SQL style.

For the second part of the case study (Section 6.2), we demonstrated the use of customized modules. Customizing GNoT to build Boston Rides took less than 10 human-hours. Using a low-level tool or a language framework such as R would have been far more timing consuming and would have lacked the interactive features offered by GNoT. Using an integrated solution such as IBM's many eyes or Tableau would constrain the flexibility in making use of external libraries to offer complex machine learning capabilities. The resulting solution would still lack the latest interactive features offered by D3. Finally, building a system from the scratch by connecting individual components: an RDBMS (e.g., Vertica), machine learning library (Weka), and a graphics library (D3), would result in a solution similar to that of GNoT, but only after spending many more human hours.

7. SUMMARY

Visual data mining combines the flexibility, creativity, and general knowledge of a human with brute computational power. In this paper, we describe a novel system, GNoT, that supports interactive knowledge discovery by interconnecting state of the art tools for visualization, relational database management, and machine learning. The system provides the glue connecting these kinds of components, and thus is able to "ride the wave" of improvements in each of these areas.

8. REFERENCES

[1] A. Aiken, A. Woodruff, J. Chen, and M. Stonebraker. Tioga-2: A direct manipulation database visualization environment. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 208–208. IEEE Computer Society, 1996.

[2] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[3] C. M. Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[4] M. Bostock, V. Ogievetsky, and J. Heer. D3 Data-Driven Documents. *Visualization and Computer Graphics, IEEE Transactions on*, 2011.

[5] C. Cortes and V. Vapnik. Support vector machine. *Machine learning*, 20(3):273–297, 1995.

[6] L. Denuzière, A. Granicz, and A. Tayanovskyy. Visualizing data on the web. In *DDFP '13: Proceedings of the 2013 workshop on Data driven functional programming*, 2013.

[7] M. Derthick, J. Kolojejchick, and S. F. Roth. An interactive visualization environment for data exploration. In *Proceedings of the Knowledge Discovery in Databases*, pages 2–9. Press, 1997.

[8] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[9] D. A. Keim. Information visualization and visual data mining. *Visualization and Computer Graphics, IEEE Transactions on*, 8(1):1–8, 2002.

[10] A. Lamb, M. Fuller, R. Varadarajan, N. Tran, B. Vandier, L. Doshi, and C. Bear. The Vertica Analytic Database: C-Store 7 Years Later. *arXiv.org*, 2012.

[11] C. North and B. Shneiderman. Snap-together visualization: a user interface for coordinating visualizations via relational schemata. In *Proceedings of the working conference on Advanced visual interfaces*, pages 128–135. ACM, 2000.

[12] S. F. Roth, P. Lucas, J. A. Senn, C. C. Gomberg, M. B. Burks, P. J. Stroffolino, A. Kolojechick, and C. Dunmire. Visage: a user interface environment for exploring information. In *Information Visualization'96, Proceedings IEEE Symposium on*, pages 3–12. IEEE, 1996.

[13] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *Visualization and Computer Graphics, IEEE Transactions on*, 8(1):52–65, 2002.

[14] M. Stonebraker. Sql databases v. nosql databases. *Communications of the ACM*, 53(4):10–11, 2010.

[15] J. Varia and S. Mathew. Overview of amazon web services. *Amazon Web Services*, 2012.

[16] F. B. Viegas, M. Wattenberg, F. Van Ham, J. Kriss, and M. McKeon. Manyeyes: a site for visualization at internet scale. *Visualization and Computer Graphics, IEEE Transactions on*, 13(6):1121–1128, 2007.

[17] C. Weaver. Building highly-coordinated visualizations in improvise. *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 159–166, 2004.

APPENDIX

Table 1 lists the basic set of core modules. The table also details the fields supported by the modules. A few other fields are also optionally supported by many modules.

- orderBy: Order by field is useful in selecting a desired region when used together with limit and start. Time series (the modules that expect X Field) are always ordered by X Field.
- groupBy: When groupBy fields are used, values fields must be aggregates.

In addition to the required and optional inputs lists in the table, all modules support the following optional fields.

- where: Text entry used to filter data (e.g. *total* > 1000).
- limit: A numeric entry to limit the number of records retrieved (e.g. 1000). Typically each module assumes a reasonable limit when this option is not specified.
- start: A numeric entry representing the offset (e.g. 1000).
- reload: A binary switch instructing GNoT to ignore the cache.
- view: When fields are derived from a complex query, it is best specified inside as a view. Then, the supplied table is ignored and query is executed against this view.

Module	Description	Required Inputs	Optional Inputs
Raw	Outputs the raw data	Table T, Fields $f_1 \dots f_n$	groupBy fields $f_{a_1} \dots f_{a_r}$, orderBy fields $f_{b_1} \dots f_{b_s}$
Calendar	Displays date-based heatmap of data	Table T, X Field f_{date}, Value field f_{val} [1]	-
Field	Shows relative frequency of values within a field	Table T, Field f	-
Multi-Field	Shows relative frequency of values of multiple fields in a hierarchical manner	Table T, Fields $f_1 \dots f_n$	-
Series	Plots multiple time series. Can be visualized as an area, bar, line, or scatter plot with multiple options for combining series and smoothing plots. Can be annotated by a field in the same table	Table T, X Field f_x, Y Fields $f_{y1} \dots f_{yn}$	Annotation field f_a, orderBy fields $f_{b_1} \dots f_{b_s}$
Diff	Plots the difference of two fields in a time series	Table T, X Field f_x, Y Fields f_{y1}, f_{y2}	orderBy fields $f_{b_1} \dots f_{b_s}$
Word	Uses word cloud to display frequency of words in a collection of text	Table T, Text field f_t	-
Word Series	Plots the frequency of words over time.	Table T, X Field f_x, Text Field f_y	-
Graph	Undirected graph with distinguishable node types	Table T, Source Field f_s, Target Field f_t	orderBy fields $f_{b_1} \dots f_{b_s}$
Matrix	Visualization of a sortable matrix where cells represent values between entities. Optional field Linkgroup can be used to cluster nodes.	Table T, Source Field f_s, Target Field f_t, Value Field f_{val}	Linkgroup Field f_c, orderBy fields $f_{b_1} \dots f_{b_s}$
Digraph	Directed graph with node values	Table T, Source Field f_s, Target Field f_t, Value Field f_{val}	orderBy fields $f_{b_1} \dots f_{b_s}$
Scatter	Visualization of scatter plot on a two dimensional plane f_x, f_y. Two additional fields determine the radius (f_z) of the markers and grouping (class f_c).	Table T, Value fields f_x, f_y, f_z, f_c	groupBy fields $f_{a_1} \dots f_{a_r}$, orderBy fields $f_{b_1} \dots f_{b_s}$
Correlations	Creates a scatter plot matrix where each node is a plot of the values of one field against the values of another. Values can be filtered on all plots by selecting a region on a single plot. First field represents the sample classes.	Table T, Value fields $f_c, f_{v_1} \dots f_{v_n}$. Limit number of fields to 5	groupBy fields $f_{a_1} \dots f_{a_r}$, orderBy fields $f_{b_1} \dots f_{b_s}$
Bar	Simple bar graph	Table T, Fields f_x, f_y	-
Crossfilter	Plots distribution of values for each field. Values can be filtered on all plots by selecting a region on a single plot.	Table T, Value Fields $f_{v_1} \dots f_{v_n}$	groupBy fields $f_{a_1} \dots f_{a_r}$, orderBy fields $f_{b_1} \dots f_{b_s}$
gMaps	The distribution of locations is displayed in heatmap format with an interactive map display. If f_x is specified, then time play of the heatmap against f_x is provided.	Table T, Longitude field f_{lon}, Latitude field f_{lat}	X Field f_x, Value field f_1 [1]
gMaps Crossfilter	In addition to the heatmap, user is also given a histogram of each of the specified filtering fields $f_{f1} \dots f_{fn}$. The user can select ranges of values within these distributions to display on the map.	Table T, Longitude field f_{lon}, Latitude field f_{lat}, filtering fields $f_{f1} \dots f_{fn}$	
ML K-Means	Cluster into k clusters using the K-means clustering algorithm	Table T, Value fields $f_1 \dots f_n$, Number of clusters k	Pre-processing method (PCA, Whitened PCA, or Z-Score), pre-transform method (Quadratic, Purely quadratic, or Interaction), groupBy fields $f_{a_1} \dots f_{a_r}$, orderBy fields $f_{b_1} \dots f_{b_s}$
ML Ridge	Fraction r of X is used to train a linear regression model. The remaining $(1-r)$ fraction is then used as a test set.	Table T, Value fields fields $f_Y, f_{X_1} \dots f_{X_n}$, regularizer α, ratio r	
ML SVM	Similar to ML Ridge, but performs classification using SVM algorithm	Table T, Value fields fields $f_Y, f_{X_1} \dots f_{X_n}$,regularizer α, ratio r	

Table 1: List of core modules