A Clustering Visualization Query Language

Ana Paula Sodré[1], Luis Eduardo Mochenski Floriano[1], Aurora Ramirez Pozo[1] e Carmem S. Hara[1]

[1]Departamento de Informática – Universidade Federal do Paraná
Caixa Postal 19.081 – 81.531-980 – Curitiba – PR

{apas19, lemf19, aurora, carmem}@inf.ufpr.br

Abstract. *In recent years, machine learning techniques have been used in several areas and applications. However, the application of such techniques still relies on professional experts, with ability to execute the required sequence of tasks. In the database area, SQL has democratized the use of database management systems (DBMS) for the end user through a simple declarative language, which resembles natural language. In this paper, Clustering Visualization Query Language (CVQL) is proposed, which extends SQL to execute clustering and visualization tasks. Clustering is a technique used to divide data into groups that share similar features. The visualization of these groupings in different graphical forms is an essential functionality for the end user to analyze the results. CVQL was implemented using mySQL DBMS and scikit-learn library. To present the system, we consider an 8 lines query example on a real database of Covid-19 cases in the USA. The execution of the same sequence os tasks would require 140 lines of code in Python, which shows the usefulness of CVQL.*

Resumo. *A utilização de técnicas de aprendizado de máquina popularizou-se nos últimos anos nas mais diversas áreas e aplicações. No entanto, a aplicação de tais técnicas ainda depende de um profissional especializado, que execute a sequência de tarefas necessárias para sua execução. Na área de banco de dados, o SQL democratizou a utilização de sistemas gerenciadores de bancos de dados (SGBD) para o usuário final através de uma linguagem declarativa simples, que se assemelha à linguagem natural. Neste artigo, é proposta a linguagem Clustering Visualization Query Language (CVQL), que estende o SQL para a realização de agrupamentos e visualização destes resultados. Agrupamento é uma técnica utilizada para dividir os dados em grupos que compartilham características similares. A visualização destes agrupamentos em diferentes formas é uma funcionalidade essencial para o usuário final analisar os grupos gerados. O CVQL foi implementado utilizando o SGBD mySQL e a biblioteca scikit-learn. Para apresentar o sistema, é adotado um exemplo de consulta com apenas 8 linhas sobre uma base de dados real de casos de Covid-19 nos EUA. Para o processamento da sequência de tarefas executadas pela consulta seriam necessárias 140 linhas de código na linguagem Python, que demonstra a utilidade do sistema.*

1. Introdução

Atualmente, o aprendizado de máquina é utilizado nas mais diversas áreas e aplicações. Nos serviços públicos, ele pode ser aplicado para prever a quantidade de incêndios ou a

incidência de uma doença em uma determinada cidade; no comércio, ele pode ser útil para identificar padrões de consumo, bem como auxiliar na escolha da localidade de centros de distribuição para minimizar custos de transporte. A aplicação de técnicas de aprendizado de máquina requer a execução de uma sequência de tarefas, que envolvem diversas ferramentas computacionais. Esta sequência de tarefas é denominada de ML Pipeline [Makrynioti and Vassalos 2020].

De acordo com [Zuccarelli 2020], estes pipelines são modulares e envolvem uma sequência de tarefas, dentre as quais estão: a coleta de dados, agregação, pre-processamento (limpeza, criação de atributos, normalização entre outras), aplicação dos algoritmos de aprendizado de máquina e avaliação dos seus resultados. O processo não é simples, principalmente para o usuário final leigo na área de informática. Existem diversas propostas para facilitar a utilização de ferramentas e bibliotecas, como a combinação de diferentes algoritmos para geração de um único resultado [Baptista de Almeida et al. 2016] e o aprendizado de máquina automatizado (AutoML), que auxilia na seleção de modelos e definição dos seus parâmetros de entrada. O auto-WEKA [Kotthoff et al. 2016] e auto-sklearn [Feurer et al. 2019] são exemplos de ferramentas de AutoML. No entanto, a preparação dos dados para dar de entrada para o Auto-WEKA ainda é responsabilidade do usuário. Já a utilização do auto-sklearn assemelha-se à escrita de um script.

Este artigo propõe uma abordagem alternativa para facilitar o uso de aprendizado de máquina para o usuário final, baseada em um linguagem declarativa de alto nível. Assim como o SQL facilitou o acesso dos usuários às bases de dados, uma linguagem declarativa para o aprendizado de máquina também pode torná-lo mais acessível. A adoção de linguagens declarativas neste contexto não é uma ideia nova. Ela foi explorada no contexto de bases de dados indutivas [Imielinski and Mannila 1996], que permitem realizar consultas sobre padrões existentes em uma base de dados e na geração de regras de associação [Meo et al. 1996]. Atualmente, com o grande interesse nas áreas de ciência de dados e aprendizado de máquina, as abordagens que facilitam sua utilização tem seu interesse renovado [Makrynioti and Vassalos 2020].

Neste artigo é explorada a ideia de associar uma consulta declarativa a um pipeline pré-definido de tarefas associadas à análise de agrupamentos. Uma das formas de avaliar e interpretar os agrupamentos gerados é através da visualização. Dessa forma, a linguagem proposta, chamada de CVQL (*Cluster Visualization Query Language*), considera a visualização como uma das tarefas do pipeline.

O CVQL é baseado em um extensão da linguagem SQL e inspirada na linguagem SCCQL [Adam et al. 2013]. No entanto, a SCCQL foi proposta para realizar apenas uma tarefa, que é o agrupamento. O foco do CVQL é na definição de um pipeline de tarefas. Neste artigo, o foco é em apenas três: obtenção dos dados de uma base de dados, agrupamento e visualização. Um exemplo de consulta na linguagem CVQL é apresentado na Figura 1. Ele executa uma consulta na base de dados (cláusula `FROM`) que utiliza duas tabelas: `covid`, que contém o total de mortes pela covid-19 em cada mês e em cada estado nos EUA; e `states` que contém informações sobre cada estado americano, como sua população (`pop`). Para cada estado, a consulta retorna: seu nome (`State`), sua população (`Population`), o total de mortes no estado (`Total`) e seu respectivo porcentual com relação à população total (`Percentage`). A partir deste resultado é

```
CLUSTER Population, Percentage
FROM (SELECT c.Name as State, s.Pop as Population,
      SUM(c.COVID_19_Deaths) as Total,
      SUM(c.COVID_19_Deaths)/s.Pop*100 as Percentage
      FROM covid c, states s
      WHERE c.Name = s.Name
      GROUP BY c.Name, s.Pop)
VISUALIZE Population, Percentage AS Scatter
```

Figura 1. Exemplo de consulta na linguagem CVQL

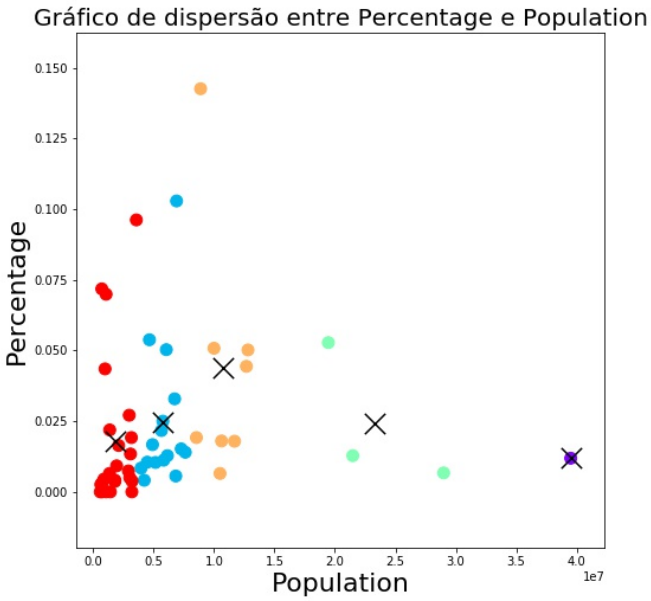

Figura 2. Gráfico de dispersão resultante da consulta na Figura 1

realizado o agrupamento sobre os atributos `Population` e `Percentage` utilizando o algoritmo k-Means, com 5 grupos. Por fim, é gerado um gráfico de dispersão (`Scatter`) sobre estes mesmos atributos. O resultado é apresentado na Figura 2, no qual cada ponto representa um estado, cores distintas são utilizadas para cada grupo e os centróides são representados com ”x”.

O restante do artigo está organizado da seguinte forma. A Seção 2 apresenta trabalhos correlatos que adotam abordagens declarativas para aprendizado de máquina. A Seção 3 detalha o CVQL e sua implementação é descrita na Seção 4. Conclusões e trabalhos futuros são apresentados na Seção 5.

2. Trabalhos Relacionados

Em [Makrynioti and Vassalos 2020] é apresentado um estudo sobre diversos sistemas que possuem uma linguagem declarativa para executar tarefas associadas à análise de dados, tais como integração com uma base de dados, processamento paralelo e técnicas

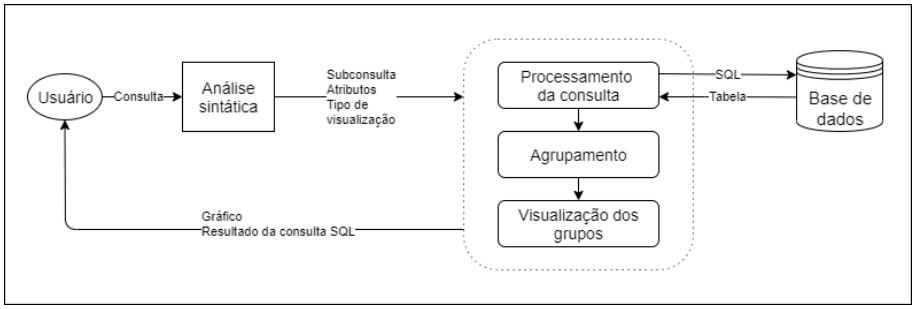

Figura 3. Arquitetura do sistema CVQL

de otimização. Já uma classificação dentre as abordagens de aprendizado de máquina declarativas é proposta em [Boehm et al. 2016]. Elas são classificadas em três categorias, dependendo da porção fixa do processo: tarefas, algoritmos e plano de execução. A maioria das propostas existentes tem como objetivo obter um maior desempenho nos processos de aprendizado de máquina e são classificadas como de algoritmo fixo. Dois exemplos são o SystemML [Boehm et al. 2016] e Stratosphere [Markl 2014], que permitem o processamento paralelo de tarefas de análise de dados baseado em Spark e Hadoop, respectivamente.

Algumas propostas baseadas em SQL também podem ser classificadas como de algoritmo fixo. Isso se deve ao objetivo para o qual foram propostos. O SciDB [Rivers 2017], por exemplo, é uma base de dados vetorial, proposta para analisar grandes volumes de dados, e o SimSQL [Cai et al. 2013], foi proposto para o ML Bayesiano. O CVQL, no entanto, tem como objetivo fazer o agrupamento de dados utilizando uma *composição* de ferramentas existentes e portanto pode ser classificada como de tarefa fixa. O objetivo também é distinto. Ele se propõe a dar suporte a uma linguagem simples para executar um processo que exigiria do usuário uma familiaridade maior com a programação. Neste sentido, o CVQL é similar ao SCCQL [Adam et al. 2013]. O CVQL não dá suporte a alguns parâmetros de agrupamento do SCCQL, tais como *must link* e *cannot link*. No entanto, ao contrário do SCCQL, o CVQL inclui em seu pipeline a tarefa de visualização dos agrupamentos, o que facilita a análise dos resultados.

3. CVQL: Uma Linguagem de Consulta para Visualização de Agrupamentos

Nesta seção é apresentado o CVQL. A Figura 3 mostra a sua arquitetura, que é composta por quatro módulos: a Análise sintática, o Processamento da consulta, Agrupamento e Visualização dos grupos. A análise sintática é responsável por determinar se a consulta submetida é válida e por extrair os elementos necessários para a execução do pipeline de tarefas. Estes elementos são: a subconsulta SQL, que é utilizada pelo módulo de processamento da consulta; os atributos sobre os quais o agrupamento é realizado; e o tipo de visualização e seus atributos, que são dados de entrada para o módulo de visualização. Nas próximas seções os módulos do sistema são detalhados.

3.1. Análise Sintática

Com o objetivo de ser simples e intuitiva, a linguagem CVQL foi desenvolvida com base no comando `SELECT` do SQL. A Figura 4 apresenta uma parte da sua gramática.

```
<statement>  ::= "CLUSTER" <attributes>
                 "FROM" <SQLQuery>
                 "VISUALIZE" <columns> AS <plotType>

<plotType> ::= "Bars"
             | "Scatter"
```

Figura 4. Sintaxe da linguagem da consulta CVQL

A consulta é composta por três cláusulas: `CLUSTER`, `FROM` e `VISUALIZE`. Na cláusula `CLUSTER`, os `<attributes>` correspondem às colunas que serão usadas para o agrupamento. Diferentemente da instrução SELECT, é possível ignorar alguns atributos que não serão significativos para o agrupamento mas ainda tê-los no resultado da consulta.

Na cláusula `FROM`, o `<SQLQuery>` refere-se à consulta SQL que deve ser executada para obter a tabela dada como entrada para o algoritmo de agrupamento.

Já a cláusula `VISUALIZE` define em `<columns>` as colunas que serão usadas para a plotagem dos gráficos do agrupamento, enquanto o `<plotType>` refere-se à forma de visualização do resultado escolhida pelo usuário.

```
subquery select c.Name ...  group by c.Name,s.Pop
attrList Population, Percentage
plotList Population, Percentage
plotType Scatter
```

Figura 5. Saída da Análise Sintática

Um exemplo de consulta é apresentada na Figura 1. A execução da análise sintática tem como resultado a obtenção dos diversos elementos necessários para os módulos seguintes. A Figura 5 apresenta a saída da análise sintática para a consulta na Figura 1. Os elementos que compõem a saída são: a consulta SQL (`subquery`), os atributos sobre os quais é executado o agrupamento (`attrList`), os atributos utilizados na visualização (`plotList`) e o tipo de visualização (`plotType`).

3.2. Agrupamento

Atualmente, o CVQL utiliza o algoritmo de agrupamento K-Means, responsável por categorizar os dados em seus respectivos grupos. O algoritmo agrupa os dados dada sua semelhança com o centroide de cada grupo. Esta semelhança é medida pela distância entre o objeto e o centroide.

No entanto, existe um passo importante que este algoritmo não realiza de forma automática, que é determinar a quantidade de grupos K. Isso em geral fica a cargo do usuário. Porém, a dificuldade de obter este valor pode ser superado com a utilização de métodos para estimar a quantidade ideal de K, dado um conjunto de dados.

Embora existam diversos métodos para determinar o valor de K [Real 2016], no CVQL foi adotado o método do cotovelo (ou *Elbow Method*). Ele opera aplicando o K-Means com diferentes números de grupos, dentro de um intervalo. Para cada valor de K, o método determina a soma dos quadrados intra-agrupamentos (WCSS - *Within-Cluster*

Sum of Squares). A partir disso, é possível traçar um gráfico usando o intervalo de K e seus respectivos WCSS, que pode lembrar o formato de um cotovelo, como visto na Figura 6.

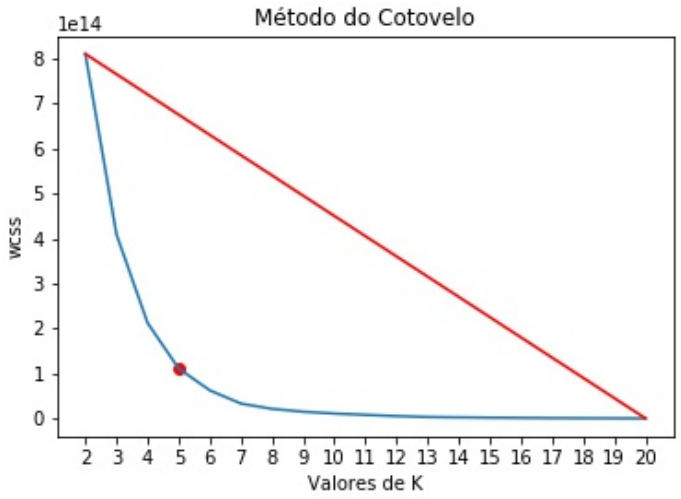

Figura 6. Gráfico do método do cotovelo

Finalmente, o ponto mais distante da reta, traçada pelo ponto inicial e final do gráfico, é tomado como valor otimizado de K. No gráfico da Figura 6, gerado a partir dos dados utilizados no exemplo, o valor de K escolhido foi 5. Isso permite que o usuário, em um primeiro momento, não tenha que se preocupar com esse tipo de análise.

3.3. Visualização dos Grupos

Como resultado do algoritmo de K-Means, a tabela dada de entrada é estendida com uma nova coluna, chamada de `cluster`, que tem valores no intervalo [0, K-1], que indicam a qual dos K grupos cada linha pertence. Um exemplo da tabela estendida para a consulta na Figura 1 é apresentada na Figura 7.

	State	Population	Total	Percentage	cluster
0	Alabama	4903185	821	0.0167	1
1	Alaska	731545	0	0.0000	4
2	Arizona	7278717	1103	0.0152	1
3	Arkansas	3017804	150	0.0050	4
4	California	39512223	4697	0.0119	0

Figura 7. Exemplo de saída como tabela

Ao aplicar o algoritmo, os dados são separados em grupos, mas ainda há questões a serem respondidas como: porque tal dado pertence a um grupo, ou então, porque tais grupos estão tão próximos. Esta análise pode ser facilitada por diferentes visualizações destes resultados na forma de gráficos.

Em seu estado atual de desenvolvimento, existem dois tipos de visualização no CVQL: gráfico de dispersão, que mostra um ponto na posição de cada dado com a cor de

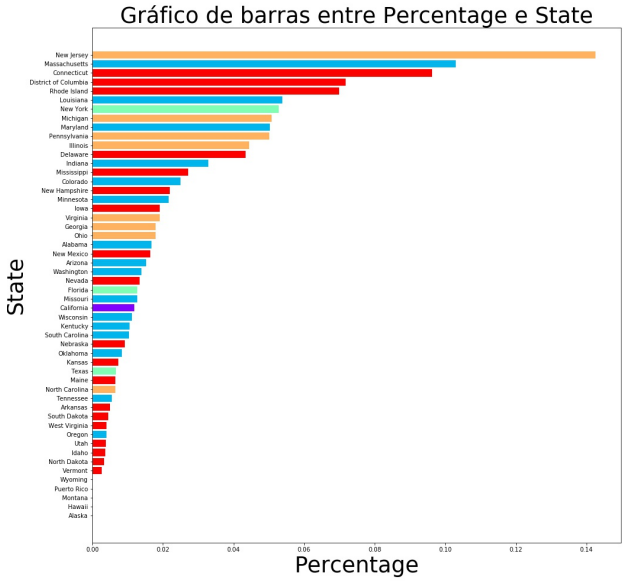

Figura 8. Exemplo de saída como gráfico de barras entre Porcentual de mortes por COVID-19 e estados

seu grupo, e o centro de cada agrupamento marcado com um X; e gráfico de barras, que mostra cada dado com a cor de seu grupo.

O gráfico de dispersão na Figura 2, por exemplo, mostra que o porcentual de mortos pela COVID-19 tende a ser maior quanto maior a sua população até o limite de 1.5 milhões de habitantes e que não há uma tendência clara para os estados com populações maiores. A saída na forma de um gráfico de barras para a mesma consulta é apresentada na Figura 8. Ela apresenta os nomes dos estados em um eixo e o porcentual de mortes no outro, com os agrupamentos representados em cores distintas. A Figura 9 apresenta um gráfico de barras semelhante, sobre a população total. Estes gráficos mostram que de fato os agrupamentos foram realizados baseados na população total, uma vez que todas as barras com a mesma cor na Figura 9 estão agrupadas. Além disso, a Figura 8 mostra que do grupo representado em verde, que corresponde ao segundo grupo de estados mais populosos, o estado do Texas se destaca por um baixo porcental de mortes, enquanto os estados de Connecticut, Columbia e Rhode Island destacam-se dentre os menos populosos, com um grande porcentual de mortes. Este exemplo mostra como a visualização dos resultados, integrados ao processo de agrupamento pode facilitar a análise dos seus resultados.

4. Desenvolvimento do Sistema

O CVQL foi implementado utilizando diversas ferramentas. O sistema gerenciador de banco de dados (SGBD) utilizado foi o mySQL. Para o analisador sintático foram escolhidas as ferramentas Flex, para gerar um programa em C que realiza a análise léxica, e o Bison, que gera um programa, também em C, para a análise sintática. O resultado da análise sintática é a extração dos diversos elementos da consulta (`subquery`, `attrList`, `plotList`, `plotType`), como ilustrado na Figura 5. Após a Análise

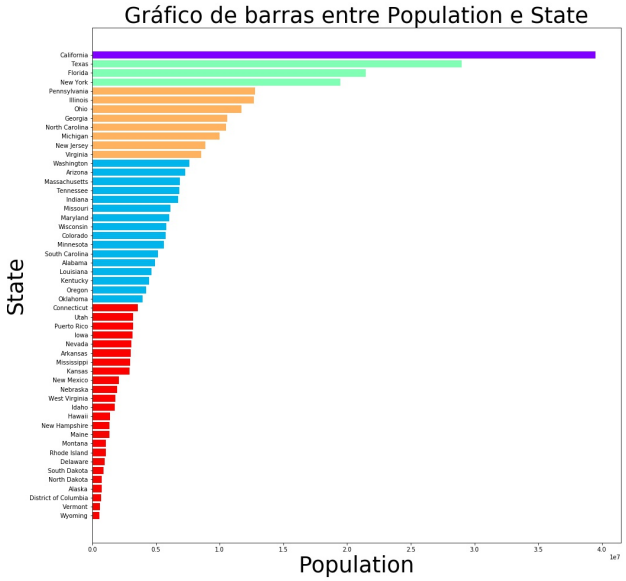

Figura 9. Exemplo de saída como gráfico de barras entre população e estados

Sintática, foi usada a linguagem de programação Python, escolhida pela disponibilidade de uma grande quantidade de bibliotecas. O programa então recebe dados do SQL, e executa uma sequência de funções para gerar o resultado desejado. Foi implementada uma função para cada módulo da arquitetura do CVQL, que recebe como parâmetro os elementos extraídos da consulta. Dessa forma, o pipeline de tarefas executados para a consulta é composto pelas seguintes tarefas:

1. **Processamento da consulta.** A função recebe como parâmetro a `subquery`, estabelece a conexão com o SGBD, submete a consulta e obtém como resultado uma tabela.
2. **Transformação.** A função recebe a tabela e a converte para um Dataframe para poder utilizar as bibliotecas Pandas e Numpy, na linguagem Python.
3. **Agrupamento.** Os dados `attrList` são usados para selecionar as colunas do Dataframe para o agrupamento, fornecendo-as como entrada para o K-Means, da biblioteca scikit-learn. O resultado é um conjunto de grupos, dos quais são obtidos os seus centros, a partir de um vetor que contem o grupo de cada linha da tabela. Esse vetor é transformado em uma nova coluna do Dataframe (`cluster`).
4. **Visualização.** O `plotList` é usado para selecionar as colunas da visualização e `plotType` para selecionar o seu tipo. Baseado nisso o programa seleciona a função de visualização conforme especificado. A biblioteca *matplotlib* foi utilizada para gerar o gráfico e salvá-lo como um arquivo JPEG.

Vale ressaltar que o sistema foi implementado de forma modular. Dessa forma, ele pode ser estendido facilmente para incluir novas funcionalidades, tais como: novas formas de visualização, entrada de parâmetros de entrada e novos algoritmos de agrupamento.

4.1. Casos de Covid-19 nos EUA

Os dados sobre Covid-19 utilizados nos exemplos ao longo do artigo foram obtidos do Centro de Controle e Prevenção de Doenças (CDC) dos Estados Unidos, que disponibiliza a tabela *Provisional COVID-19 Death Counts by Sex, Age, and State*[1]. Ela conta com 13 colunas e 1416 linhas, com dados a respeito do número de mortes por COVID-19 por estado americano, incluindo atributos como sexo, intervalo de idades e relação com outras doenças respiratórias, como influenza e pneumonia. Os dados sobre a população dos estados americanos foram obtidos da página do censo americano[2].

O objetivo de facilitar o uso de aprendizado de máquina para o usuário final pode ser observado comparando a consulta na linguagem de alto nível do CVQL com a quantidade de linhas de código em Python necessários para obter o mesmo resultado. Para realizar a mesma sequência de tarefas, o programa teria aproximadamente 140 linhas de código para um único tipo de visualização, sendo necessárias 10 linhas adicionais para novas visualizações, como o gráfico de barras.

5. Conclusão

Este artigo apresentou o CVQL, que tem como objetivo facilitar a utilização de técnicas de aprendizado de máquina para o usuário final. Ele adota uma linguagem declarativa simples, que é uma extensão da linguagem de consultas para bancos de dados SQL e, em sua versão atual, dá suporte a dois tipos de visualização: gráfico de dispersão e gráfico de barras.

Como trabalho futuro, é planejado estender a linguagem e o sistema com outras tarefas, tais como: pré-processamento de dados, tratamento de dados categóricos e interpolação de dados para o tratamento de dados nulos; utilização de outros algoritmos de agrupamento; suporte a outros tipos de visualização; e definição de parâmetros pelo usuário para ajustar tanto a visualização como o algoritmo de agrupamento. É importante ressaltar que algumas extensões listadas são simples de ser implementadas, uma vez que o sistema foi desenvolvido tendo em vista adições de funcionalidades futuras.

Referências

Adam, A., Blockeel, H., Govers, S., and Aertsen, A. (2013). Sccql : A constraint-based clustering system. *Lecture Notes in Computer Science - Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*, 8190:681–684.

Baptista de Almeida, J. L., Sakata, T. C., and Faceli, K. (2016). Asaclu: Selecting diverse and relevant clusters. In *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 474–479.

Boehm, M., Evfimievski, A. V., Pansare, N., and Reinwald, B. (2016). Declarative machine learning - a classification of basic properties and types. *arXiv*, 1605.05826.

Cai, Z., Vagena, Z., Perez, L., Arumugam, S., Haas, P. J., and Jermaine, C. (2013). Simulation of database-valued markov chains using SimSQL. In *Proc. of the International Conference on Management of Data (SIGMOD)*, pages 637–648.

[1] `https://data.cdc.gov/NCHS/Provisional-COVID-19-Death-Counts-by-Sex-Age-and-S/9bhg-hcku/data`

[2] `https://www.census.gov/data/datasets/time-series/demo/popest/2010s-national-detail.html`

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M., and Hutter, F. (2019). Auto-sklearn: efficient and robust automated machine learning. *Automated Machine Learning*, pages 113–134.

Imielinski, T. and Mannila, H. (1996). A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64.

Kotthoff, L., Thornton, C., Hoos, H. H., Hutter, F., and Leyton-Brown, K. (2016). Auto-WEKA 2.0: Automatic model selectionand hyperparameter optimization in WEKA. *ournal of Machine Learning Research*, 17:1–5.

Makrynioti, N. and Vassalos, V. (2020). Declarative data analytics: a survey. *IEEE Transactions of Knowledge and Data Engineering (TKDE) (to appear)*.

Markl, V. (2014). Breaking the chains: On declarative data analysis and data independence in the big data era. *PVLDB*, 7(13):1730–1733.

Meo, R., Psaila, G., and Ceri, S. (1996). A new sql-like operator for mining association rules. In *Proc. of the 22nd International Conference on Very Large Data Bases (VLDB)*, page 122–133.

Real, E. M. (2016). Estimating the number of clusters based on sequential clustering algorithms. In *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 229–234.

Rivers, J. (2017). Scidb: An array-native computational database for heterogeneous, multi-dimensional data sets. In *Proc. of the 2017 IEEE International Conference on Big Data (Big Data)*, pages 3206–3210.

Zuccarelli, E. (2020). Developing machine learning pipelines. *Towards Data Science*.