School of Information Technology and Engineering at the ADA University

School of Engineering and Applied Science at the George Washington University

MACHINE LEARNING REPOSITORY

A Thesis
Presented to the Graduate Program of Computer Science and Data Analytics
of the School of Information Technology and Engineering
ADA University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Computer Science and Data Analytics
ADA University

By
Gunay Maharramli

April, 2024

THESIS ACCEPTANCE

This Thesis by: Gunay Maharramli
Entitled: *Machine Learning Repository*

has been approved as meeting the requirement for the Degree of Master of Science in Computer Science and Data Analytics of the School of Information Technology and Engineering, ADA University.

Approved:

(Adviser)	(Date)
(Program Director)	(Date)
(Dean)	(Date)

ABSTRACT

The thesis proposes a design and practical implementation of a database system that is made for holding and managing the descriptive data derived from various repositories. The driver of this project is the goal of giving researchers, data scientists, and enthusiasts a place to access, probe, which is to help analyze, data spread across different repositories. The main purpose is building up a powerful and easy to be used database solution which will be able to keep in memory a wide range of data descriptions and also provide efficient storing, retrieving and querying functions. For the implementing of the project, Mongodb as a NoSQL database was chosen because of following reasons. MongoDB provides a schema-free framework to be able to keep documents with various data types which do not belong to a predefined schema. This is necessary due to the fact that the structures of the dataset schemas are quite different in the data repositories. On the top of that, the flexibility and high performance of MongoDB render it a reliable solution for handling big volumes of data as well as to enable concurrent access for many users. The UCI World Machine Learning Repository, Kaggle and OpenML are the main data repositories in this database. As a initial phase, 100 documents exist in the MongoDB database. Other than that, MongoDB makes provision for lots of querying mechanisms such as Mongo Shell, MongoDB Compass, graphical user interface (GUI), which are meant to enhance the users' choice of interacting with the database depending on the preference, skill and expertise. Additionally, new GUI for querying datasets information was created using flask in the python code, and GUI improves the overall usability and accessibility of the database system by giving users an easy-to-use interface to search, filter, and view dataset descriptions. Broadly, the database system designed and described in this paper is a convenient asset for the data science community

Table of Contents

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Explanation
NoSQL	Not Only SQL
JSON	JavaScript Object Notation
BSON	Binary-encoded Serialization of JSON-like documents
ACID	Atomicity, Consistency, Isolation, Durability
BASE	Predictive Model Markup Language Basically Available, Soft State, Eventual Consistency
RDBMS	Relational Database Management System

1. INTRODUCTION

The quality and availability of datasets are essential for making and evaluating algorithms and models in the area of current data science and machine learning research. The need for efficient, scalable data management solutions is increasing with the rapid growth in volume as well as complexity of data. This study intends to fulfil this requirement by establishing a MongoDB database that is designed specifically for collecting and organizing information about machine learning datasets. MongoDB, with its known distributed design and document-based data structure, provides a robust foundation for efficiently dealing with the different and ever-changing requirements of managing machine learning datasets.

Among the NoSQL database management systems, MongoDB is one of the most flexible, scalable, and user-friendly. It operates with document-oriented data model. This is what makes it so easy to integrate with the current application development frameworks as the information gets stored in the documents in BSON format both internally and over the network. It looks similar to JSON (JavaScript Object Notation). As MongoDB is schema-free and can handle larger workloads than traditional relational databases, it is the most suitable for accomplishing key tasks. MongoDB, due to the fact that it has rich capabilities and powerful tools, is preferred for many things such as this project. The key purpose of this project is the development of a multipurpose database that will contain information, such as names, descriptions, subject areas, features, links etc via storage, while keeping high levels of flexibility, integrity and accessibility. With the use of MongoDB robust querying and indexing features users will be able to easily search filter retrieve desired information based on many criteria which makes it well suited for easy use within machine learning workflow processes. Dataset accuracy, consistency, and long-term reliability will also be looked at by researching data acquisition and online processes like validation and versioning. This study intends to proof the effectiveness and feasibility of the MongoDB based database solution as a one-size-fits-all solution for machine learning users, researchers, and data enthusiasts through experimenting with various parameters as well as comparisons with real-life use cases, and general datasets. As part of this research, the focus will be given to find out the real potential of working with MongoDB and designing a machine learning compatible database system with it. MongoDB presents such remarkable assets as data query

efficiency, scalability, and flexibility. Using the mentioned database, I plan to develop a robust solution designed to solve the specific challenge of managing machine learning data. Through rigorous testing and study, I want to demonstrate how MongoDB is of great help to machine learning tasks such as data collecting, organizing, and retrieval for they are highly beneficial to the process. Looking at the last stage, this initiative tries to enhance performance, reproducibility, and creativity among the machine learning community by introducing this central repository in which all the relevant information about machine learning datasets are collected. It will therefore foster a freely data sharing culture and develop the field in an expedited way.

2. PROBLEM STATEMENT

To provide solutions for numerous significant problems confronting professionals and experts in the realm of machine learning and data science, a combined database is being formed. It will gather dataset metadata from diverse origins like Kaggle, UCI Machine Learning Repository as well other specific or general data repositories. The thesis purpose is to tackle the current fragmentation and absence of uniformity between datasets with their accompanying metadata that are spread out across numerous platforms.

In projects of machine learning, this scattering is greatly limiting the success of searching, evaluating and using datasets. The situation created by data repositories where many significant datasets are not being used to their maximum capacity has been generated because those who might use them often do not know they exist or cannot easily get detailed information about these resources. In deciding if a dataset suits a particular machine learning project, access to thorough information or metadata about the dataset's subject area, features, instances, variables and more is very important. Without a central repository keeping all this metadata information, the search for an ideal dataset across multiple websites is a tedious task. As a result, projects build up the progress more slowly and and makes it more difficult to continue improving machine learning models. Next, the comparison of the datasets with different data source platforms becomes difficult as there are no standards in the presentation of the dataset metadata. In order to decide which dataset is appropriate, the experts and researchers, most of the time, need to

analyze the datasets metadata a process which is not efficient and error prone. Disorganized and dynamic character of the machine learning projects which may require datasets from a large number of items or worse from poorly defined features will lead to such inefficiency.

A unified database with a single queryable and standardized data metadata is the aim brought forward to tackle the inconsistency and scattered level of metadata dispersion of various datasets. Due to the property of specific search through application of various metadata criteria, the dataset reviewing and discovering process will be significantly accelerated for users using this database. This, in turn, can lead to find and search for datasets otherwise hard to identify or assess, as well as accelerate the startup and development of machine learning initiatives, and also encourage better utilization of the available data archive. The most importantly, with the database the most of datasets will be gathered in one place to make the decision-making for professionals and researchers better informed as offered the view of all datasets, hence it becomes easier for them to differentiate and choose those that go well with their needs.

2.1 SIGNIFICANCE OF THE STUDY

The proposed paper seeks to resolve the core problems with datasets, be they those of identification and evaluation, in order to improve research and development in the fields of machine learning and data science in respect to efficiency, accessibility, and quality. The main goal in this project is to lessen the usual amount of time and energy required for locating datasets by gathering metadata from various repositories into one searchable database.

The study also encourages interdisciplinary studies by creating a system that compiles datasets from different areas. Such an approach opens up the possibility of finding new discoveries in the intersection of fields like medicine, computer science, business and social sciences among other places while at the same time making it easier to locate and combine sets from unrelated areas. Standardizing metadata is one way to make comparison and evaluation of datasets less complex thereby enabling researchers or developers know what kind of data will work best for a given project. This close

matching ensures more dependable and faster machine learning models. Also, this database allows for imaginative reuse of old datasets which could bring out hidden potentials in previously ignored information; thus saving resources as well as fostering innovative thinking and resourcefulness.

This thorough matching of data with project requirements is likely to result in more dependable and effective machine learning models. The database also helps in the imaginative reuse of preexisting datasets, possibly revealing hidden value within previously ignored data. This could improve resource effectiveness and encourage an environment that promotes innovation and creativity.

2.2 LIMITATIONS OF STUDY

Despite the fact that the work has important contributions to the field of machine learning research and dataset location and evaluation, it has a set of limitations as well. The Metadata and the way repositories display information are often different among data repositories, which is the most serious problem. The dimensions in which the Kaggle, OpenML and UCI Machine Learning Repository provide the dataset information are different [6][7][8]. Sharing the common database is problematic because of differences resulting from different websites and data collection. Select the category of data while looking into bottom at the key characteristics of the data being showcased to a customer is the very basis of the process of deciding which data is the useful. Nevertheless, all mandatory information may not always be registered in a consistent manner or may not be consistently shown from one platform to another, resulting in data lacunae that can be standardized and gathered.

A very large part of manual work exhibited the solution of that issue. Pre originate code wrote to extract metadata has been run in the early stages when this process has been piloted. Nevertheless, due to complex structure of web sites these data have to be often added or modified manually to make sure that collected metadata is really accurate and the final product is targeted towards the right audience.

As a next step, distillation of the data includes assessment on which datasets to add to the database, specification of the particular field of metadata to collect, and how to cater the missing or incomplete data. Both fundamental for assuring the database quality as well as productiveness complicated however, selective curation process does have some weaknesses. The manual records database is not considered flexible and updated as it requires more time and efforts since the I have to manually input the records. Appropriately, it may lead to a postponement of the adding of fresh datasets to the database or an update of the current metadata data. This cause some delay in the addition of new datasets to the database or the updating of current metadata, that would reduce the tool's overall relevance and timeliness.

On the one hand, things like the selection, categorization, and the metadata of the datasets as a whole become so subjective when we use manual procedures. Despite the care taken to achieve objectivity and accuracy, judgment calls will still differ and later feed to disparate ill-representations of the subset provided. It is worth mentioning that the Human curation method works great with the data from the sources that has good documentations and the structure as compared to the unstructured data that might not have been documented. In the case that the difficulty of obtaining or understanding their content is beyond what the database can handle, then these datasets will be left out of the system. All these problems explicitly prove the hardness of one single database to collect, update and set it current, as it requires permanent endeavors to look for new data and improve techniques of curation.

Since the project can face resource issues, for instance, such as shortage of processing power, development period, and experience. Such constraints might even result in a lack of the advanced features in the database system and may also delay the development process and make it more difficult for the programming team to handle the technical problems. Shortage of computer resources might cause scaling problems, there might appear some performance bottlenecks, and, at the same time, tight of development time may force the deployment of some features over the others, which are more sophisticated. Although outdated database design, implementation, or maintenance knowledge can equally stand as a blockage to the project to benefit from optimal solutions or problem-solving difficult technical parameters. Besides streamlining the development processes

and maximizing the use of the information available, there should be a strategic planning for the resource utilization, at the same time being careful while allocating them.

3. RESEARCH APPROACH OR METHODOLOGY

Finding a dataset that suits well the machine learning field in which you might be working is usually quite complicated and winding. It was exactly this challenge, that motivated me to come up with the idea of not only finishing the project, but actually turning it into a thesis project. Obtaining a dataset which facilitate my research is somehow a hard job and I got this information through my personal experience and talking to others. Actually, searching through a set of different sources including datasets on OpenML, Kaggle, and the UCI Machine Learning Repository as well as various layouts and their unique search platforms may be quite a challenge and will entail sore investigation process. What is more, this unsystematic and sporadic way of finding data will lead to a wasted time as well as will result in missing a chance to stumble upon a dataset.

The main aim of thesis is to make it simpler for machine learning experts and students to find the datasets for their research projects. With this in mind, I created a centralized database that compiles metadata on different data sets collected on various platforms to achieve this task. The name of the dataset, a short definition, its variables, the kind of features, categories to which it belongs, subject area, associated tasks, characteristics, the number of features and instances and whether any values are missing are all included in this metadata. This data is also provided for those datasets that have been labeled with keywords to allow more advanced searches. The database links are available in every database record. So, the users can get their data instantaneously without taking extra steps. Besides, I also introduced variable information and other relevant pieces of data when possible because I have come to realize that some datasets provide additional useful information that can help a specific project. This extra data layer is meant to help users understand better what each dataset can bring to their work by giving them the chance to pick the most useful dataset.

3.1 DATABASE SELECTION

Choosing an appropriate database plays a key role when you want to create a database for your machine learning (ML) datasets collection. After that, I conducted research of NoSQL databases and traditional Relational Database Management Systems (RDBMS) so there are some differences between them.

For data modelling and querying, approaches taken by RDBMS and NoSQL databases differ from each other. For RDBMS, data may be standardized in order to avoid redundancy and guarantee the data can be stored in the most efficient way. The complex joins in querying frequently require the selection of data from multiple databases. Applications that require the exact adherence to schema and data integrity can be seen to conform better with the inherent structure of relational databases. Unlike relational databases, NoSQL databases accept denormalized and flexible schema designs, thus making reading and writing it faster because relevant data in one group. It all depends on the data model itself as queries vary and so do each type of NoSQL databases. On the contrary to some databases which have SQL-like query languages, other databases make use of APIs or query languages that are particular to their data model.

Moreover, RDBMS and NoSQL are implemented with contrasting approaches to application builds. Vertical scaling is an ubiqitous RDBMS technique, where additional physical (hard) resources are updated in execution of increased load. Therefore in this case, vertical scaling will not be the right choice as it will require to choose the horizontal scalability. Rather than in comparison with SQL databases, NoSQL is intended to be horizontally scalable, which means that data can be distributed across multiple nodes while more of them can be added automatically if a need for more capacity arises. Hereby this level renders a NoSQL database good for web applications, IoT, or places where we completely do not know the amount of volumetrics and deal with a big number of episodic users at the same time.

RDBMSs is well known for the following features: their strong consistency, the organized schema, and the strong Query Language (SQL). However, this schema based on the the traditional relational databases is not very suitable to the dataset metadata

which contains lots of variability and unstructured data. The datasets that are collected by machine learning from different sources are mostly not having a common standard of representing data. Now if you keep to the different formats and descriptors, the appropriate RDBMS scheme would not be the best option.

Another difference between RDBMS and NoSQL databases is structure. Data in relational database management systems is stored in the tables where it keeps arrangement of data structure including entities of records. Each table consists of rows and columns, row describes a record and column represents field in that record. Foreign key constraints identify the table relationships which maintain referential integrity and also make it possible to link joins being used to carry out searches that are much more complicated across numerous tables. On the other hand, data in NoSQL databases such as MongoDB is stored in collections, collection is a equivalent of the table in RDBMS. Collections has documents that can have different structures depending on keys and values, and these collections display entry of the table.

Another difference between RDBMS and NoSQL databases lies in their transaction models ACID and BASE accordingly.

- **ACID:** ACID, which is an acronym for atomicity, consistency, isolation and durability. These ACID properties make certain that all database operations in a transaction (group of them together) maintain the database's valid state even if unexpected errors occur. Atomicity makes sure that all commands within a transaction are handled as one unit, and they either all succeed or fail together [1]. This is significant because if there is an unwanted event such as a crash or power cut, we can know for certain about the database's condition. The transaction would have finished correctly if it wasn't affected by any issue; otherwise, it might be rolled back if any part of the transaction did not work as expected. Consistency is vital as transactions only proceed provided all database pipeline rules have been met and there are no other inconsistencies in the process. This will be consistency in every rule and constraint, and every triggering clause. In the case data finds itself in the state of being unauthorized, then the transaction will just end up failing. Isolation ensures that all the transactions are taking place in independent areas [4]. This leads to the possibility of multiple transactions being performed on that block due to

the fact that they are not interfering with one another. Durability ensures that changes made to the database (transactions) that are successfully committed will survive permanently, even in the case of system failures. This ensures that the data within the database will not be corrupted by service outages, crashes, other cases of failure.

- **BASE:** It stands for Basically Available, Soft State, Eventually Consistent. Instead of requiring consistency, BASE-modelled NoSQL databases assure data availability by distributing and replicating it throughout the database cluster. Because of the absence of instant consistency, the data values may fluctuate with time. The BASE approach departs from the idea of a database that requires its own consistency, transferring that obligation to developers. The fact that BASE does not need instant consistency does not imply that it never achieves it. However, until it does, data readings are still possible [4].

The most importantly, NoSQL databases such as MongoDB are the better option since they are known its schema flexibility. This is one of the major advantages when dealing with an unstructured kind of data. With this flexibility, data information is not forced to conform to fixed schema while being saved. Leading NoSQL database MongoDB specifically had in mind the flexibility, and this can't be overemphasized regarding data sets metadata that varies broadly among databases. The feature that uses BSON, which is a language that very much resembles JSON and has dynamic schemas to facilitate quick insertion into the database of various types of data, provides such flexibility. The fact that MongoDB has strong indexing functionality ensures efficient gathering of data and search through the multitudes of data set attributes, which is the key when the users browse through the attributes to find the one that suits their experimental needs most.

Also, this often results in the output of the datasets is lacking in proper documentation and clarifications of the database. It is realized that users might have to struggle a lot in case they cannot locate the exact dataset that might serve in their work because of the succinct nature of data. We stand guided to carry out a detailed examination of the present data and build the metadata by aligning datasets to MongoDB. There are several ways of improving this, for instance, by adding tag words that can help users in searching for the entries, or by providing more descriptors and details that can help users in zeroing

into what they desire. Then, users may proceed with more elaborate searches narrowing their possibilities within the fields of more information using a wide range of information fields. To add to this, MongoDB is scalable, which makes it possible for the system to perform excellently well even as the datasets grow without the speed acting as a limitation. The high availability and robustness in MongoDB (the same the scalability) are important components of the system and grant the service for dataset metadata.

Apart from that, the aggregation structure of MongoDB facilitates the execution of in-depth analytical queries on the metadata of the datasets. This is beneficial to users who might have to do sophisticated searching to find data sets. MongoDB is the right choice for the project, for it can also perform text search, provide query capabilities, and handle the high volume of reads and writes. Researchers and practitioners can get a full experience with machine learning datasets that is user-friendly because our use of MongoDB features.

3.2 MONGODB DATABASE

MongoDB is an open source NoSQL database management program. NoSQL (Not only SQL) is used as an alternative to traditional relational databases. NoSQL databases are quite useful for working with large sets of distributed data. MongoDB is a tool that can manage document-oriented information, store or retrieve information.

3.3 HISTORY OF MONGODB DATABASE

In February 2009, MongoDB was developed and made readily available. In 2013, 10gen was renamed to MongoDB Inc. in order to fit their company's activities better, i.e. involvement with the MongoDB database. It is hosted by mongoDB.inc, a corporation that uses the server-side public license (SSPL). As the company went public with MongoDB Inc in 2017, the business has been able to operate and expand its services, with one of the services introduced being MongoDB Atlas - a cloud-based database [5]. The concept of providing database services through the cloud is not new, with other NoSQL database providers having already started similar offerings. Atlas runs on top-tier public

clouds such as AWS (Amazon Web Services), Microsoft Azure and Google Cloud Platform - these are environments that give high performance for managing data needs efficiently (Amazon Web Services; Microsoft Azure; Google Cloud Platform). MongoDB also made an addition to this by launching a platform called Stitch for application development on their MongoDB Atlas. They have intentions of expanding it further into databases that are located within premises or on-site (Amazon Web Services; Microsoft Azure; MongoDB Inc.). Also, in order to support numerous drivers for various programming languages, including C, C#, C++,.Net, Java, Swift, Python, PHP, and others, the business designed MongoDB. Many businesses that store large amounts of data, including Nokia, eBay, Aadhar, Shutterfly, EA, and others, use MongoDB.

The database's name "MongoDB" derived from the word "humongous" which means enormous. and serves the purpose of storing and processing of data in huge amounts. The MongoDB solution was invented by Dwight Merriman and Eliot Horowitz who were co-workers at DoubleClick as well and later took by the Google [12]. During their job egaligas with huge datasets spread across several data centers and they found that the relational databases didn't meet the performance and scalability criteria they required. They had made the decision to construct their own back-end database which eventually was named MongoDB. Owing to its deep flexibility that allows you to model data in a unique way, MongoDB has become a developer favorite. The topic also has a wide range of features like indexing, replication , sharding and aggregation etc.

3.4 DATABASE EXPLANATION

A feature common to most NoSQL databases is their extreme flexibility: MongoDB. In a single document, you may deal with and save several sorts of data. Comparatively speaking to relational databases, MongoDB can hold significantly more data. JSON is the document storage format used by the database system, which gives MongoDB these fundamental features. When storing and transmitting data objects, readable text is used in JSON (JavaScript Object Notation), an open standard file and data interchange format. Utilized in numerous electronic data interchanges, it is a standard data format. Despite having beginnings in JavaScript, it is now available for usage with a wide range of

computer languages because the codes supporting JSON data, which is saved in files ending in.json [10]. MongoDB stores large amounts of data in collections. Groups of related documents in a database are kept in collections. Documents inside the same collection may have diverse structures because collections, unlike tables in relational databases, do not impose a schema on the documents they include. We store a variety of data kinds and modify the data model as needed to accommodate changing application requirements because of this flexibility. Documents within the collections are comparable to the rows in a relational database system table. MongoDB documents consist of keys, which are the fundamental data unit in the database. Documents can contain several distinct fields and data types. Adding or deleting new and current fields allows one to change the structure of data that has already been saved in a document under the NoSQL MongoDB. The same document can also be utilized with different data types and this makes MongoDB's structure flexible.

Schema migrations are made easier and scalability and flexibility increase when dynamic data modification is permitted as data structure may be changed on the fly. Also, on the other side, MongoDB's schema-less architecture allows to keep data with different structures in the same collection what makes it easier to work with diverse kinds of data. As it is a flexible database, MongoDB can deal with application complexity for today's highly shiftable data requirements. The fact that MongoDB is able to manage arrays and nested documents is also one of its significant features. It allows for the proper representation of hierarchical relationships and provides a naturally readable format to the data in this context [2].

As a part of my thesis project, I designed and create database called "repository" , and one collection that is named "mlrep". Mainly, in the collection descriptors of each dataset from various repositories are stored. Items in this collection is called a document and documents have many fields that describes a dataset itself. For the initial phase 100 repositories' descriptives available in the collection. Depending on the websites, the fields are different; however, for each one there are some common fields exist. The fields are following:

- **Name**: Unique name for every dataset taken from website

- **Description**: Some descriptive and additional information to quickly understand dataset
- **HasMissingValues**: Field that shows if the missing values exist in the dataset or not
- **Link**: Direct Link to the original source for each dataset
- **Keywords**: Search terms that describes dataset, it will help users to find datasets by specific topics
- **Subject_Area:** Relevant field of study that dataset is about such as health and medicine, business, computer science or etc.
- **Variables:** Information that was taken from csv files of each dataset, it will give additional information of each feature.

The fields that were give above are the same for all three repositories. In the following figures the document's overall structure and additional fields are described.

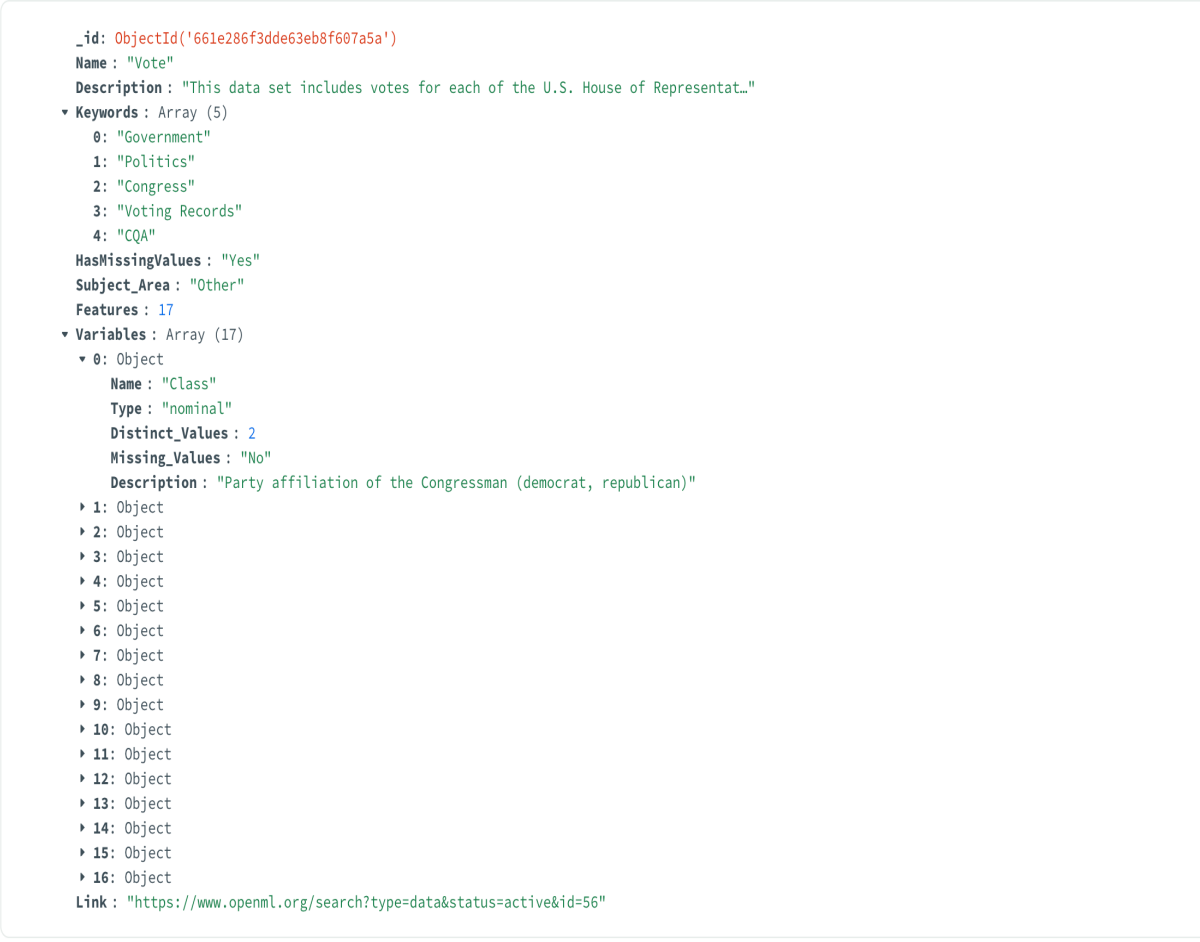

Figure 1. Document for OpenML Repository in the MLPREP Collection

In the OpenML repository, the only one additional field exist.

- **Features**: The number of features in each dataset

Figure 2. Document for Kaggle Repository in the MLREP Collection

As shown from the Figure 2, two new fields were included.

- **Associated_Tasks**: Determines the machine learning tasks that datasets is appropriate for such as classification, regression, or clustering.
- **Characteristics:** Describes structure and complexity of dataset

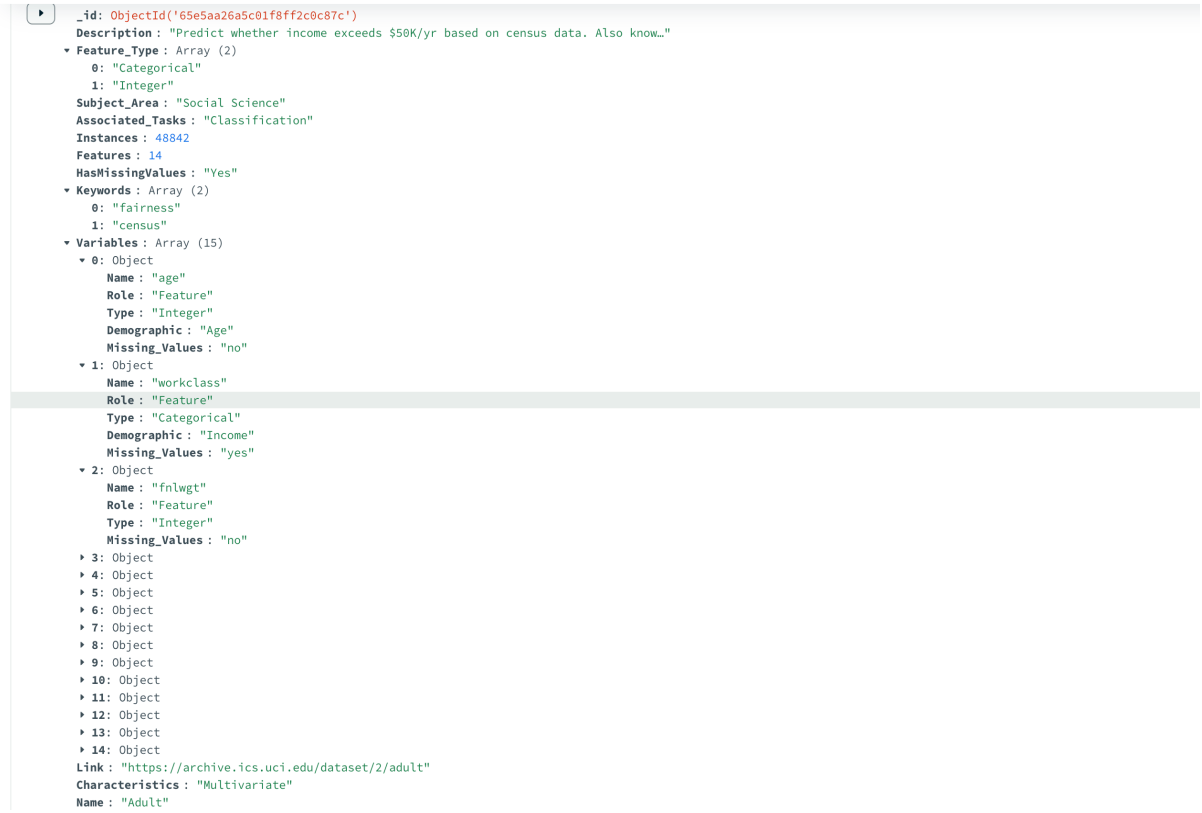

Figure 3. Document for UCI Machine Learning Repository in the MLREP Collection

UCI Machine Learning Repository has more information including fields that exist in OpenML and Kaggle.

- **Feature_Type:** Details about the kinds of features in the dataset, it can be categorical, real or integer.
- **Instances:** Number of data records

4. QUERYING DATA

I have taken a thorough approach to obtaining metadata from several sources, including Kaggle, OpenML and the UCI Machine Learning Repository, for my master's thesis, which examines machine learning database design. Having chosen MongoDB as my database management solution, I am aware of the inherent diversity in datasets' metadata between platforms and even within individual datasets. MongoDB's schema-less

architecture aligns seamlessly with the unstructured nature of the collected data, offering unparalleled flexibility in document storage and retrieval.

Each document in MongoDB database is basically a content which contains all the important things including name, description, link and keywords to establish the necessary framework for dataset categorization and exploration [3]. Next, including the attributes such as features, instances along with demonstration of tasks, topic areas and characteristics in metadata repository from UCI Machine Learning Repository will make it more meaningful by virtue of the related information. I have elevated the detail and the depth of the dataset descriptions by embedding into the new document structure not only these additional variables but also the necessary information that allows in-depth analysis of the data and therefore knowledgeable decision making.

The fact that MongoDB has no rigidly data structure scheme allows me easily from being limited by preestablished data structure patterns and will help me to easily handle the complexity and diversity of metadata representations from different sources. Such adaptability empowers tasks which follow like searching and analysis, at the same time it shortens the time used to load data. There is no problem with determining relevant information from given filters; People can easily do that through MongoDB's intuitive querying interface. This enhances the creation of grounds for the effective research and discovery inside the large repository.

An organized strategy is required to fetch the relevant data from MongoDB. The planning of queries is a must to achieve that. For users, MongoDB enables custom searches to fulfill any particular criteria because database's versatile query operators with precise data extraction become easier.Among the elements of MongoDB Querying, understanding the collection's class of document's is a key one. In the case of a document with field imbedded in field of nested data, where the documents are nested, one needs to use MongoDB's querying syntax properly to retrieve the data. The first thing we need is to use dot notation in order to deal with deeply nested documents like array and objects gracefully. MongoDB is capable of providing a detailed path using dot notation so that it can easily find the needed data within nested structures and store it in a cache.

Another feature that makes MongoDB appealing is its array operators which have high level of document-centric functionality for array queries. MongoDB's array operators provide list techniques which can be deployed to tackle tasks like finding out documents having arrays that match certain filter requirements or a specific value search within the array. MongoDB query operators, which are composed of regular expressions, logical operators as well as comparison operators, are a powerful resource that enables you to generate precise queries to solve your unique needs. Either imposing logical constraints, discovering patterns or working with numerical thresholds, MongoDB possesses an enormous set of operators meant to efficiently work with different querying needs. For retrieving data, three ways to query data and their examples will be shown.

4.1 DATA FROM MONGODB SHELL

The one way to query data is to use MongoDB Shell, The MongoDB Shell, mongosh, is a JavaScript and Node.js REPL environment that allows you to interact with a MongoDB setup locally or remotely. Use the MongoDB Shell to run queries and interact with data in your MongoDB database. The mongo shell is supplied with the MongoDB server installation. If it is previously installed the server, the mongo shell is placed in the same directory as the server binary. However, it can also be downloaded separately from MongoDB Server.

For retrieving dataset's information, initially, we need to find connection string, the connection string can vary depending on the deployment that connects on. Example connection string can be shown as following:

mongodb://repository_user: 1iForM1ojPY8nmd27nnM @IP_Address:27017/?authSource=repository

Every user that connects a database should have user repository_user in this example, followed by long password, IP Address where database is imported, mongodb's port that 27017, and database that we want to connect. We are connecting repositoy database in this scenario.

As a next step, to connect database using mongosh following with connection string is needed.

Finally, you are able to import documents and run queries based on your needs. Example query is show in the below.

Query: db.mlrep.find({ "Keywords": "Business", "Link": /kaggle/, "Associated_Tasks": "Regression"})

Result:

```
repository> db.mlrep.find({ "Keywords": "Business", "Link": /kaggle/, "Associated_Tasks": "Regression"})
[
  {
    _id: ObjectId('661c43b7c64c23fbd7c41279'),
    Name: 'Amazon Sales Dataset',
    Description: "This dataset is having the data of 1K+ Amazon Product's Ratings and Reviews as per their details listed on the official website of Amazon",
    Keywords: [
      'Business',
      'Retail and Shopping',
      'Data Analytics',
      'Exploratory Data Analytics',
      'Ratings and Reviews'
    ],
    Variables: {
      product_id: 'Product ID',
      product_name: 'Name of the Product',
      category: 'Category of the Product',
      discounted_price: 'Discounted Price of the Product',
      actual_price: 'Actual Price of the Product',
      discount_percentage: 'Percentage of Discount for the Product',
      rating: 'Rating of the Product',
      rating_count: 'Number of people who voted for the Amazon rating',
      about_product: 'Description about the Product',
      user_id: 'ID of the user who wrote review for the Product',
      user_name: 'Name of the user who wrote review for the Product',
      review_id: 'ID of the user review',
      review_title: 'Short review',
      review_content: 'Long review',
      img_link: 'Image Link of the Product',
      product_link: 'Official Website Link of the Product'
    },
    Link: 'https://www.kaggle.com/datasets/karkavelrajaj/amazon-sales-dataset',
    Associated_Tasks: 'Regression',
    Characteristics: 'Multivariate',
    Subject_Area: 'Business'
  },
  {
    _id: ObjectId('661d8c608f19cf35b2d302bb'),
    Name: 'University Employee Salaries (2011 - Present)',
    Description: "Explore the salaries of higher education employees from Ohio's public universities dating back to 2011. This dataset offers insights into the earnings of various positions across multiple institutions. Please note
are not included in the reported salaries.",
    Keywords: [
      'Business',
      'Finance',
      'Universities and Colleges',
      'Economics',
      'United States',
      'Employment'
    ],
    Subject_Area: [
      'Economics',
      'Employment Studies',
      'Social Sciences',
      'Business'
    ],
    Associated_Tasks: [ 'Regression', 'Descriptive Analytics', 'Comparative Analysis' ],
    Characteristics: 'Tabular',
    Variables: {
      Name: 'Name of the employee.',
      School: 'Name of the university.',
      Job_Description: "Description of the employee's job role.",
      Department: 'Department of the employee.',
      Earnings: 'Gross earnings of the employee in USD.',
      Year: 'Year of the reported salary.'
    },
    Link: 'https://www.kaggle.com/datasets/asaniczka/university-employee-salaries-2011-present'
  }
]
repository>
```

Figure 4. Example Query 1 from MongoDB Shell

The query that was run in Mongo Shell basically says retrieve data that has Business as a keyword, has associated_tasks is Regression from Kaggle repository. As a result we got two datasets out of 100, one is Amazon Sales Dataset, and second one is University Employee Salaries.

Query: db.mlrep.find({ "Variables.Name": "sepal length", "Link": /uci/})

Result:

```
repository> db.mlrep.find({ "Variables.Name": "sepal length", "Link": /uci/})
[
  {
    _id: ObjectId('65e5a744a8c01f8ff2c8c87a'),
    Feature_Type: 'Real',
    Subject_Area: 'Biology',
    Associated_Tasks: 'Classification',
    Instances: 150,
    Features: 4,
    HasMissingValues: 'No',
    Variables: [
      {
        Name: 'sepal length',
        Role: 'Feature',
        Type: 'Continuous',
        Units: 'cm',
        Missing_Values: 'no'
      },
      {
        Name: 'sepal width',
        Role: 'Feature',
        Type: 'Continuous',
        Units: 'cm',
        Missing_Values: 'no'
      },
      {
        Name: 'petal length',
        Role: 'Feature',
        Type: 'Continuous',
        Units: 'cm',
        Missing_Values: 'no'
      },
      {
        Name: 'petal width',
        Role: 'Feature',
        Type: 'Continuous',
        Units: 'cm',
        Missing_Values: 'no'
      },
      {
        Name: 'class',
        Role: 'Target',
        Type: 'Categorical',
        Description: 'class of iris plant: Iris Setosa, Iris Versicolour, or Iris Virginica',
        Missing_Values: 'no'
      }
    ],
    Link: 'https://archive.ics.uci.edu/dataset/53/iris',
    Characteristics: 'Tabular',
    Keywords: 'Ecology',
    Description: 'A small classic dataset from Fisher, 1936. This is one of the earliest datasets used in the literature on classification methods and widely used in statistics and machine learning.  The data set contains 3 classes of 50 insta
es each, where each class refers to a type of iris plant.  One class is linearly separable from the other 2; the latter are not linearly separable from each other.',
    Name: 'Iris'
  }
]
```

Figure 5. Example Query 2 from Mongo Shell

This is another example query which shows results from UCI Machine Learning Repository, the difference is that because Variables field is an array, dot notation is used for retrieving appropriate document. Any type of queries can be written to get dataset's descriptors quickly and easily using Mongo Shell. In the below, I added some simple queries that are useful for querying data.

1) **db.mlrep.find({ "Feature_Type": "Categorical"})** - The query retrives documents that Feature Type is Categorical.
2) **db.mlrep.find({ "Subject_Area": "Health and Medicine", "Instances": { $gt: 50 })** – The query fetches documents which has subject area Health and Medicine, and amount of instances is greater than 50.
3) **db.mlrep.find({ "Characteristics": { $in: ["Real", "Multivariate"] })** – The query fetches documents which characteristics is Real or Multivariate.

4) **db.mlrep.find({ "Keywords": "Music", "Link": /kaggle/}) –** The query will give results where it has Music as a Keyword, and it is from Kaggle dataset.
5) **db.mlrep.find({ "Keywords": "education", "HasMissingValues": "No"}) –** The query fetches documents with keywords education and does not have any missing values.
6) **db.mlrep.find({ "Link": /openml /}) –** All documents from openml repository will be shown.
7) **db.mlrep.find({ "Link": /uci/})** - All documents from UCI Machine Learning Repository will be demonstrated.
8) **db.mlrep.find({ "Keywords": "computer", "Link": /kaggle/i, "Feature_Type": { $ne: "Categorical" } }) –** The documents that has keywords computer, feature type is categorical from kaggle will be shown.
9) **db.mlrep.find({"Keywords": { $regex: /business /i }}) –** The query retrieves all documents that has business in any keywords.
10) **db.mlrep.find({"Variables.Name": { $regex: /buying/i }, "Associated_Tasks": "Classification"}) –** In this query, the datasets that has variable name buying and associated task is classification will be returned.

4.2 DATA FROM MONGODB COMPASS

On the other hand, for the ones do not write the MongoDB queries and search for datasets just using key and values, MongoDB Compass tool is useful. MongoDB Compass is a poweful GUI that allows you to query, aggregate, and analyze MongoDB data visually [9]. Basically, the one should connect the deployment hosted on local machine or cloud, import data from CSV or JSON files into MongoDB database. Then, querying a data is quite simple, as an example, if you type **{}**, all data will be prompted. In the following images, the example queries and their results will be shown.

Query:

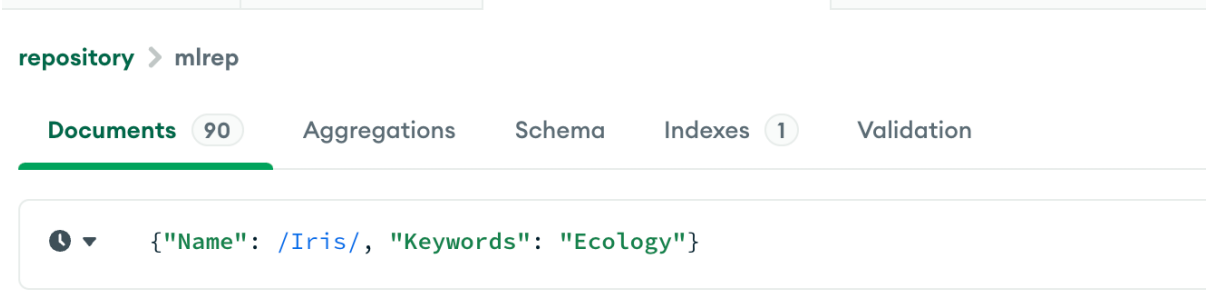

Figure 6. Sample Query in MongoDB Compass

Here, to specify equality conditions we need to use <field>:<value> expression. In the image shown above, we use regex with using / sign to get Names that has Iris in the whole word, and any keyword that matches Ecology. Two results will be demonstrated in the following queries.

Figure 7. Result from MongoDB Compass Tool

The result in the Figure 7 demonstrates dataset which is named Iris from UCI Machine Learning Repository.

Figure 8. Result from MongoDB Compass

In this figure, since we used regex, we got Name field that has a key “Iris Dataset Extended”, dataset was taken from Kaggle Repository.

4.3 DATA FROM MONGODB DOCUMENT SEARCH

Another way to get document is using MongoDB Document search which is Flask application that provides people with a simple user interface to query MongoDB database were written in python. Basically, it consists of main fields that exist in mlrep collection, and display the result based on search terms that user defines. In this document search, user is able to search based on Feature Type, HasMissing Values (Yes, No, not specified), Link (Openml, Kaggle and UCI Machine Learning Repository), Subject Area (user can mention any word, if there is any it will return results), Name (regex is applied for name), Features (users can searcg based on amount of features in the dataset), Keywords (regex is applied in this term also) and Characteristics (it can be Multivariate, Tabular and etc.). In the Figure 9, search terms for HasMissingValues (No in the case), Keywords (education) and Name that has UCI were demonstrated. Fields are changeable, and results depend on the keys that written in the search fields.

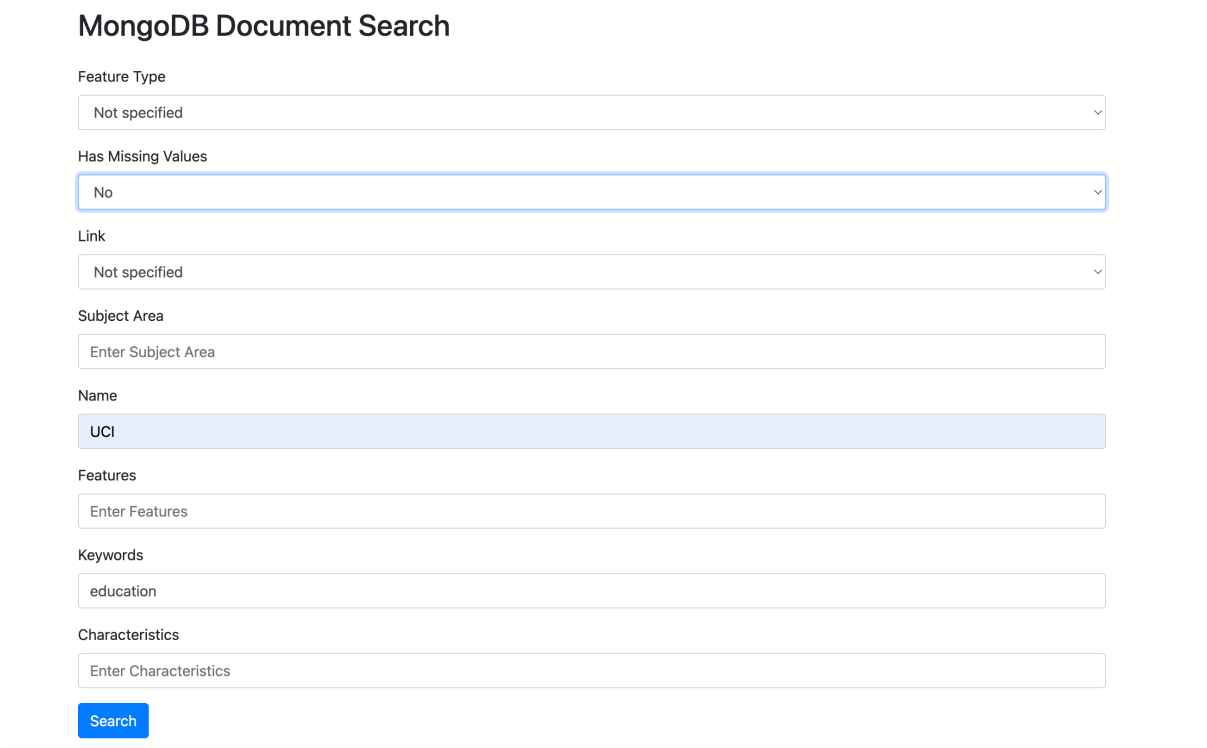

Figure 9. MongoDB Document Search

Result:

```
},
"Name": "UCI Student Performance Mat",
"Description": "This dataset approaches student achievement in secondary education of two Portuguese schools. Th
"Keywords": [
  "Education",
  "Student Performance",
  "Portuguese Schools",
  "Grades"
],
"Subject_Area": "Education",
"HasMissingValues": "No",
"Features": 33,
"Variables": [
  {
    "Name": "school",
    "Type": "string",
    "Distinct_Values": 2,
    "Missing_Values": "No",
    "Description": "Student's school ('GP' - Gabriel Pereira or 'MS' - Mousinho da Silveira)"
  },
  {
    "Name": "sex",
    "Type": "string",
    "Distinct_Values": 2,
    "Missing_Values": "No",
    "Description": "Student's sex ('F' - female or 'M' - male)"
  },
  {
    "Name": "age",
    "Type": "numeric",
    "Distinct_Values": 8,
    "Missing_Values": "No",
    "Description": "Student's age (from 15 to 22)"
  },
  {
    "Name": "address",
    "Type": "string",
    "Distinct_Values": 2,
    "Missing_Values": "No",
    "Description": "Student's home address type ('U' - urban or 'R' - rural)"
```


Figure 10. Result for MongoDB Document Search

CONCLUSION

In conclusion, this project involved immense work to get the dataset descriptions from the three powerful repositories including UCI Machine Learning Repository, OpenML and Kaggle. We decided to use MongoDB, which is schemaless database, since that way we could organize and manage datasets information with ease and rationality. Being able to manage dynamic schema turned out to be especially useful for this project: diverse repositories might store different forms of data. It can draw on document-oriented approach with enables one to work with datasets regardless of their attributes and formats without a preset schema.

The database MongoDB is very flexible, which makes it the best choice for this project. It fits our requirements that aims to provide users an interactive and quick method to query dataset descriptors easily. Clearly, MongoDB's appearance allows the current users to view or use repositories more easily regardless of what properties these data repositories offer. It does not require a fixed structure, so it can handle and process different kinds of data and structures that will be in future machine learning applications which are changing all time.

For the purpose of improving the effectiveness of queries, all different kinds of instruments are utilized. The Mongo shell makes highly dynamic command-line interface for pros to be easily used in the command line to perform queries directly. MongoDB Compass is a GUI that facilitates a visual desktop platform for data exploration and management based on the user's needs. To reach the end of the understanding, the special Flask application also has been doney written. Thus, the website, which was written with Python language, becomes like a platform that people can quickly and simply log in to the database. It facilitates the use of even those with the equivalent skills with lesser problems.

With this database system, the dataset descriptor is tightly associated with the ownership and access; in the meantime, it can serve as a guide for future data aggregation systems. By using MongoDB expandability and its flexibility in addition to the tools which can be used to get the needed data easily, this project presents the MongoDB solution as a top choice for effective management of various data sources. The harmony in the integration of data is now reachable depicting the boundless scale of information that may continually be added as the number of data repositories increases.

Following the thorough review of all the grounds, establishing such a project will signify a very important step in the data repository management. It represents the use of MongoDB NoSQL database for data mining methods of ML complex and multidimensional datasets of today. The infrastructure here drops the entry for using data down which accredits analysts, data scientists, and researcher to get valuable insights from the large volume of data simple.

FUTURE WORK

A perspective for the future is to further widen the scope by adding more information and increasing its accessibility. In the current conditions 100 datasets are in the database, and hence one of the main goals is increasing the database size by adding more new datasets from other repositories and data sources. This project will include also exploring other repositories that may hold unique or specific data sets, enlarging the database efficiency while maintaining the diversity of data. Simultaneously, resources will be expanded regarding a more comprehensive collection of data sets which combines datasets of different sources so as to present different variants of analytical options. In this manner the users that make different goals and fields of interest may find one place where the required data is always met.

Featuring the process of automating that can be used to include and analyze the datasets will be one of the largest steps towards the right path. At the beginning, I had to do some part of the project which involved analyzing the datasets and loading them into MongoDB database. While this way worked perfectly fine to start with, it demands less mannual labor to scale up, as well as more accuracy and efficiency. Besides removing the factor of inaccuracy of data and faster database content upgrade, automation will speed up the process of new dataset updating.

In addition to this, there can be an ongoing process of improving Flask applications that will deliver an intuitive experience and keep users engaged. The success of the database largely depends upon its interface and in order to enhance this aspect we aspire to decrease the challenges that limit entry into the platform, thereby lowering the threshold for a wider engagement and exploitation of the database resources for a larger base.

To conclude everything, it has reached to the point of making its own transformation which will include increasing the available information and changing the way it processes and shows data. The long-term product is a constantly revising, lean database which will show the way in the data management solutions arena and provide a top-quality material for data analysts, researchers and the others, who need a large amount of quality information.

REFERENCES

1. “ACID vs. BASE: Comparison of Database Transaction Models”. [Online]. Available: https://phoenixnap.com/kb/acid-vs-base
2. Banker, Kyle; Bakkum, Peter; Verch, Shaun; Garrett, Doug; Hawkins, Tim. "MongoDB in Action, 2nd Edition." Manning Publications, 2016.
3. Bradshaw, Shannon; Brazil, Eoin; Chodorow, Kristina. "MongoDB: The Definitive Guide, 3rd Edition." O'Reilly Media, 2019.
4. GC, Deepak, "A Critical Comparison of NOSQL Databases in the Context of Acid and Base" (2016). Culminating Projects in Information Assurance.
5. Gillis, A. S. & Botelho, B. MongoDB. TechTarget. Retrieved [2020], https://www.techtarget.com/searchdatamanagement/definition/MongoDB
6. UCI Machine Learning Repository. [Online]. Available: https://archive.ics.uci.edu/
7. Kaggle Repository. [Online]. Available: https://www.kaggle.com/datasets
8. OpenML Repository. [Online]. Available: https://www.openml.org/search?type=data&status=active
9. MongoDB Compass Documentation. [Online]. Available: https://www.mongodb.com/products/tools/compass
10. MongoDB Documentation. [Online]. Available: https://www.mongodb.com/docs/
11. Silberschatz, Abraham; Korth, Henry F.; Sudarshan, S. "Database System Concepts, 7th Edition." McGraw-Hill Education, 2020.
12. “ A Brief History of MongoDB”. [Online]. Available: https://www.tutorjoes.in/mongodb_tutorial/history_of_mongodb