**TY B.Tech. (CSE) – II [ 2021-22]**

**Software Engineering Tools Lab**

**PRN/ Roll No: 2019BTECS00018**

**Full name: Rushikesh Kedarnath Patange**

**Batch: T1**

**Assignment No. 1**

**Module 1- Introduction to OSS**

**Q1. Weka is a GUI workbench that empowers data wranglers to assemble machine learning pipelines, train models, and run predictions without having to write code.**

**Using Weka tool perform below tasks such as data preprocessing, data classification (use any appropriate ML algorithm) and data visualization efficiently on given dataset.**

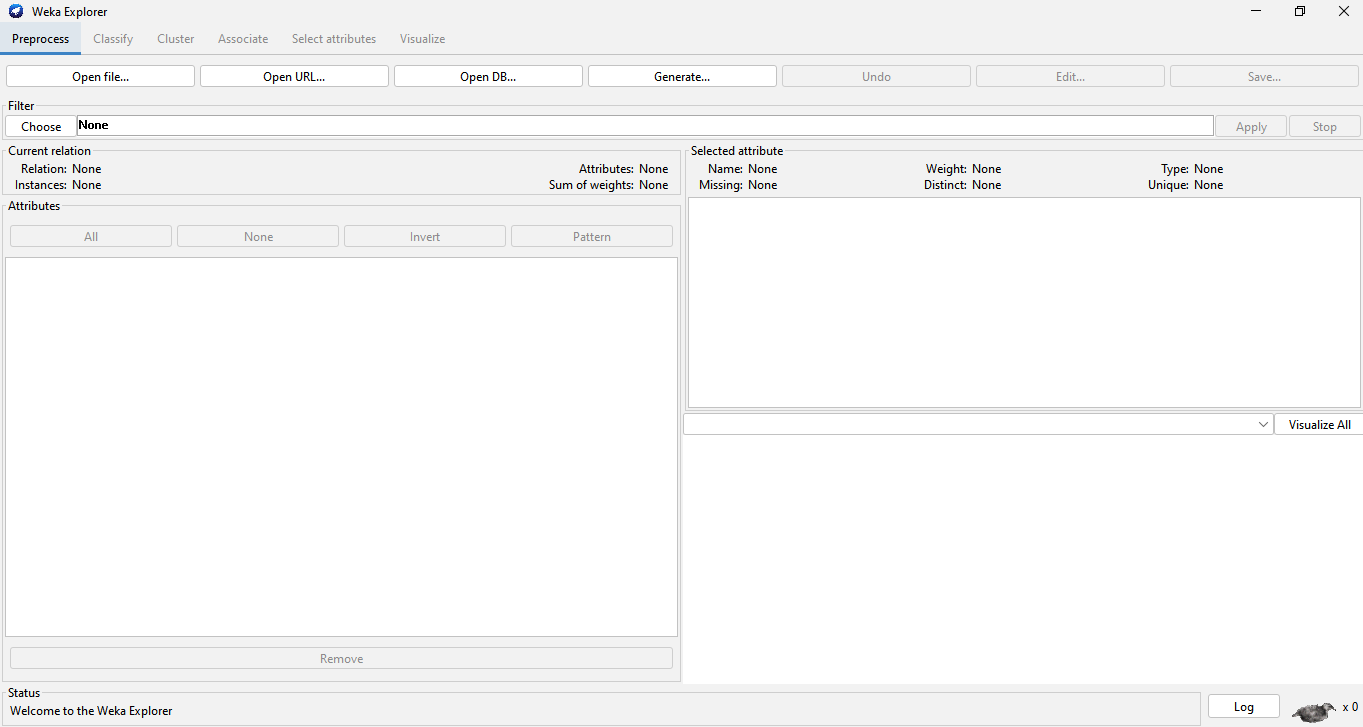
**Use the Iris dataset given-**

[**https://drive.google.com/file/d/1A3Fxsfzm6BSfhFZGDrjI47RTe45bSgYP/view**](https://drive.google.com/file/d/1A3Fxsfzm6BSfhFZGDrjI47RTe45bSgYP/view)

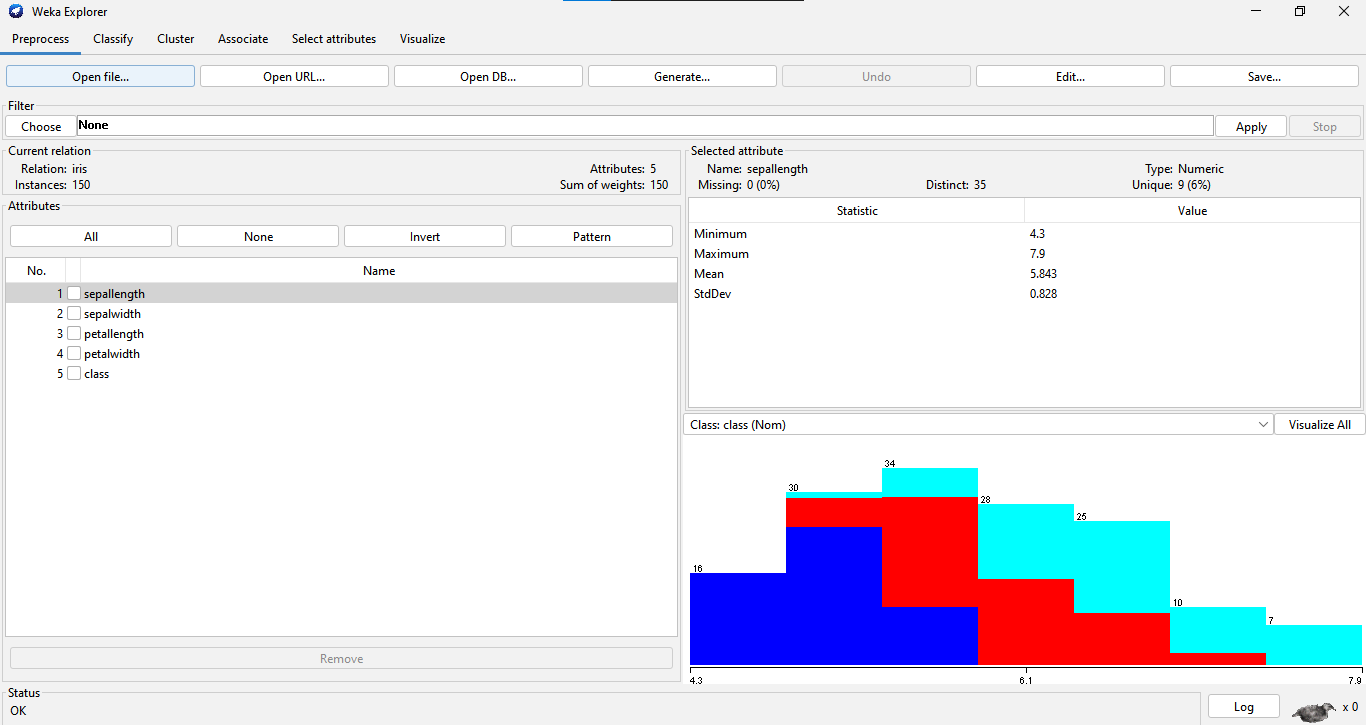
**Note-provide screen shots for every task**

**Create a report which will illustrate the details of tasks performed (for e.g to perform preprocessing of data provide details of navigation and selection of appropriate parameters)**

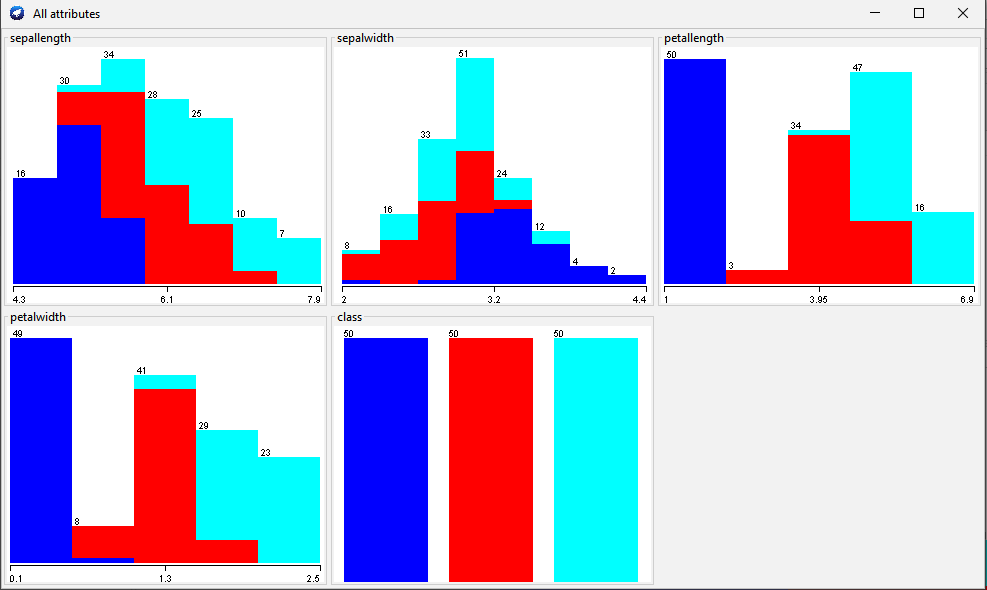
**Weka:**

****

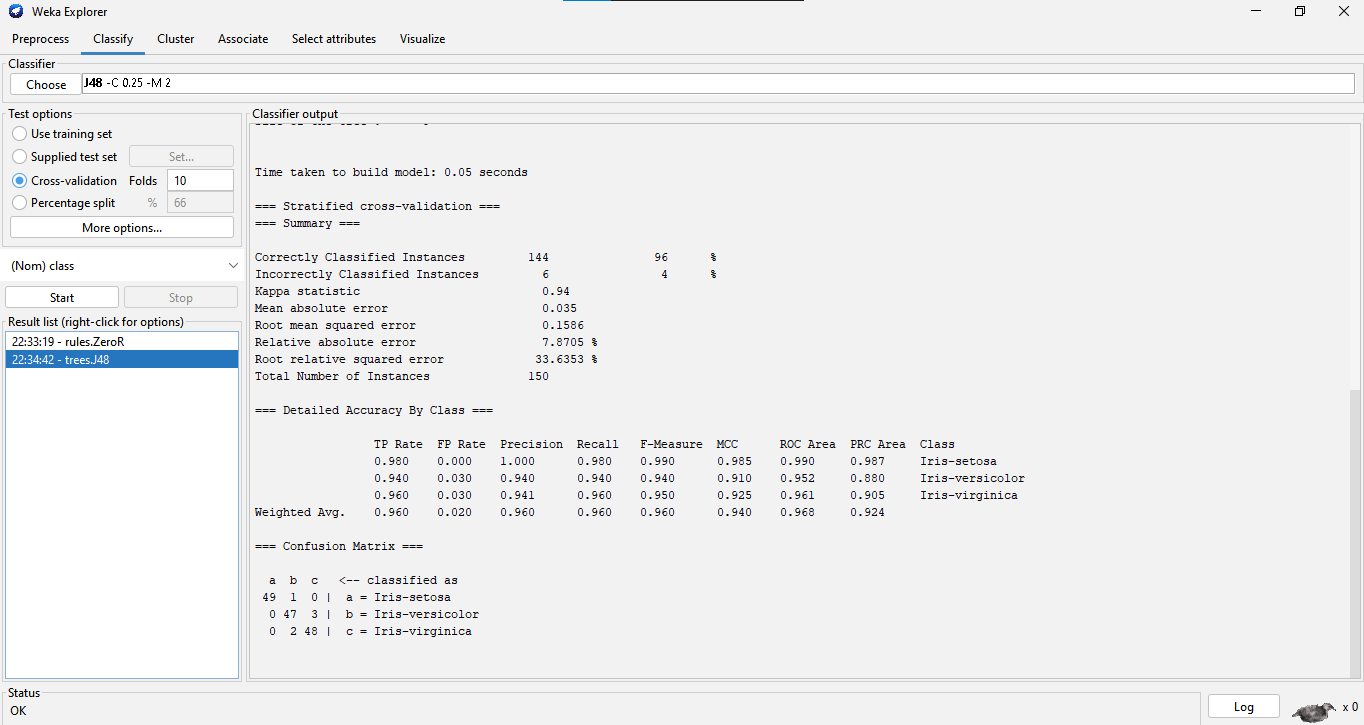
**Progress:**

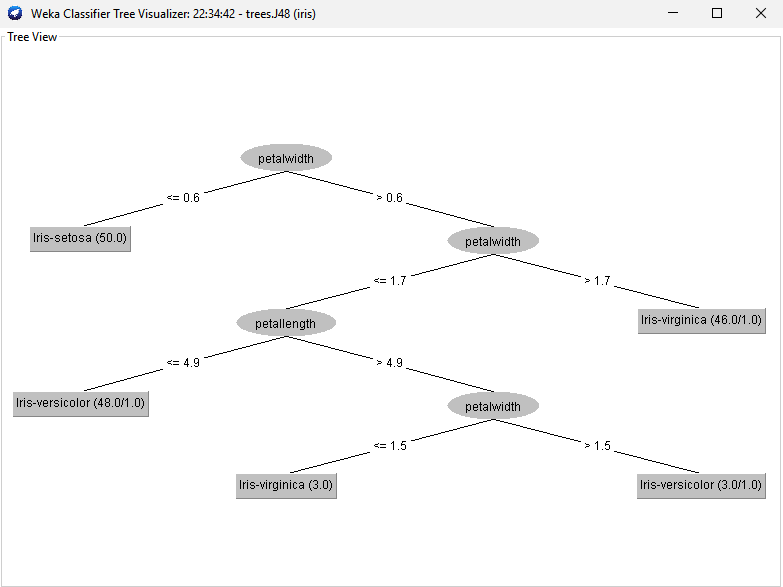
****

**VisulizeAllAttributes:**

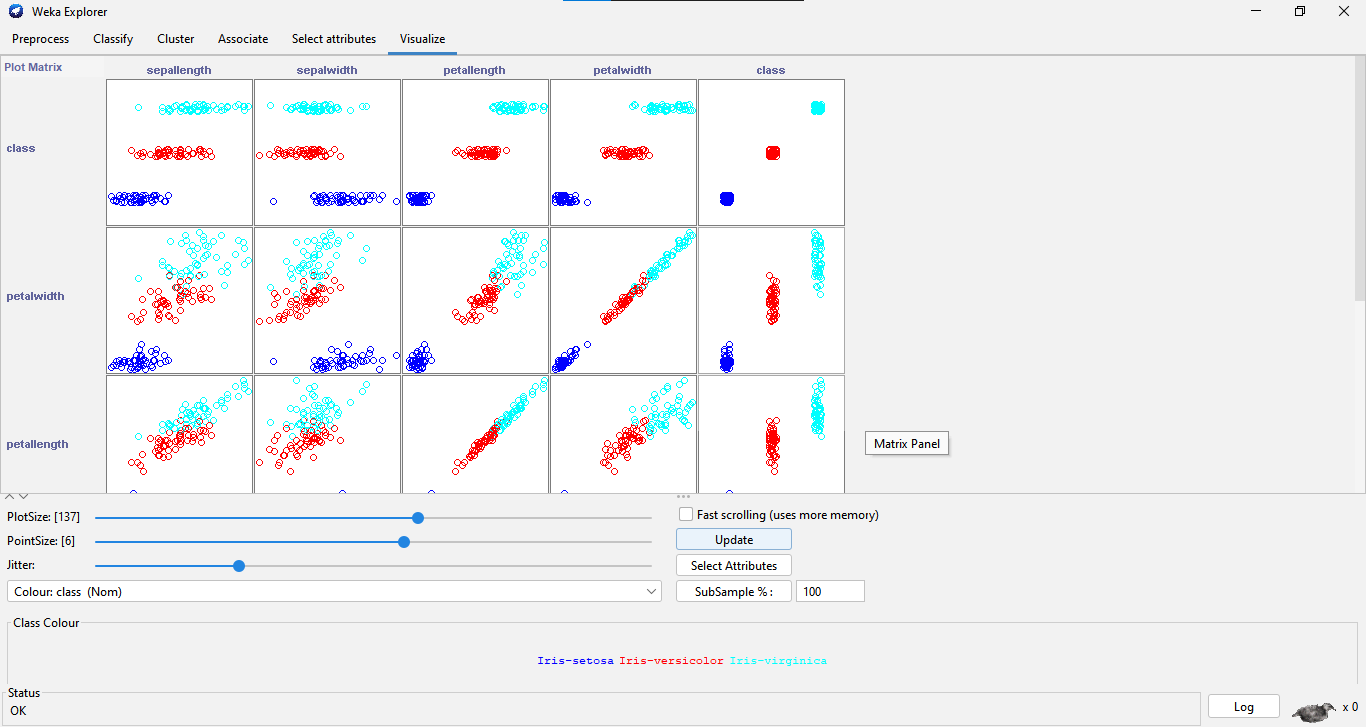
****

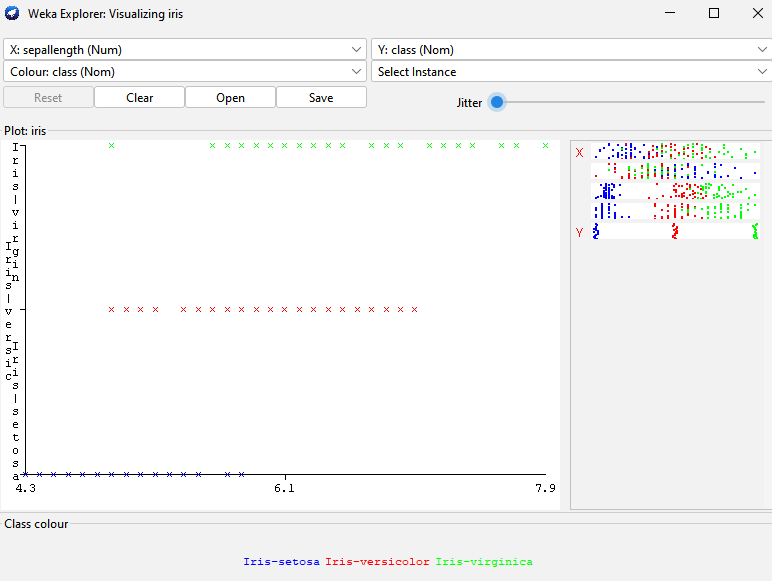
**Classification:**

****

****

**Visulization:**

****

****

**Q2. Orange is an easy-to-use data visualization tool with a large toolkit. In spite of being a GUI-based beginner-friendly tool, you mustn’t mistake it for a light-weight one. It can do statistical distributions and box plots as well as decision trees, hierarchical clustering and linear projections.**

**a. Install orange**

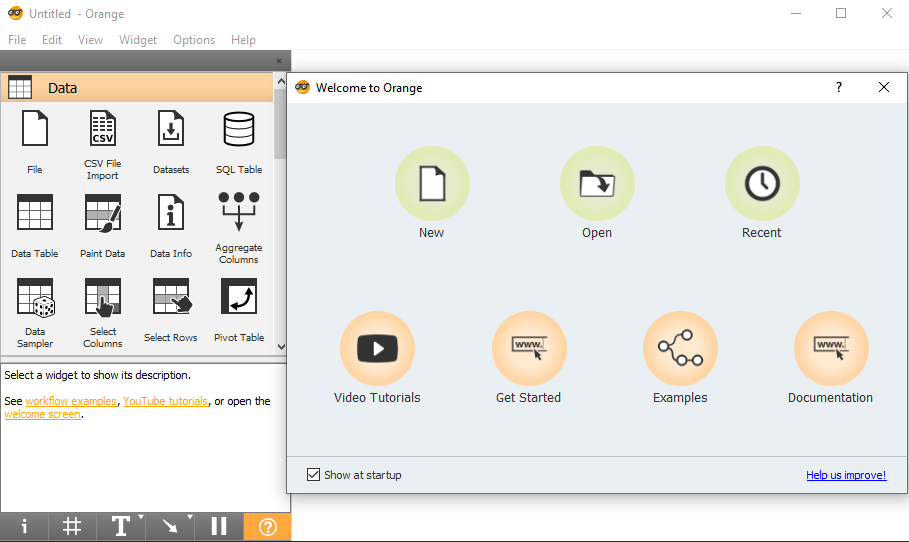
**b. Show data distribution**

**c. Show linear projection**

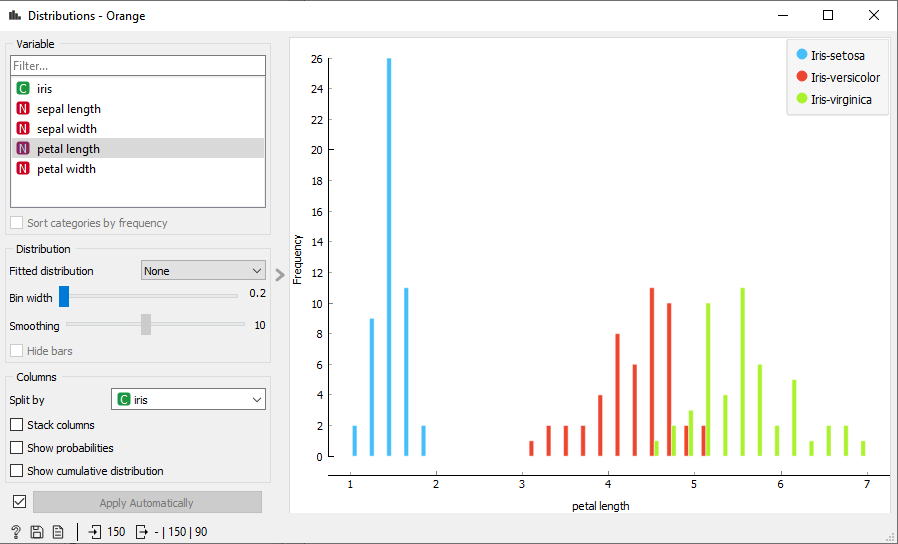
**d. Show FreeViz**

**Use dataset** [**https://drive.google.com/file/d/1m6sKI1Dap0XK6Bw1edUd5PohwpPwXnd9/view**](https://drive.google.com/file/d/1m6sKI1Dap0XK6Bw1edUd5PohwpPwXnd9/view)

**Orange**

****

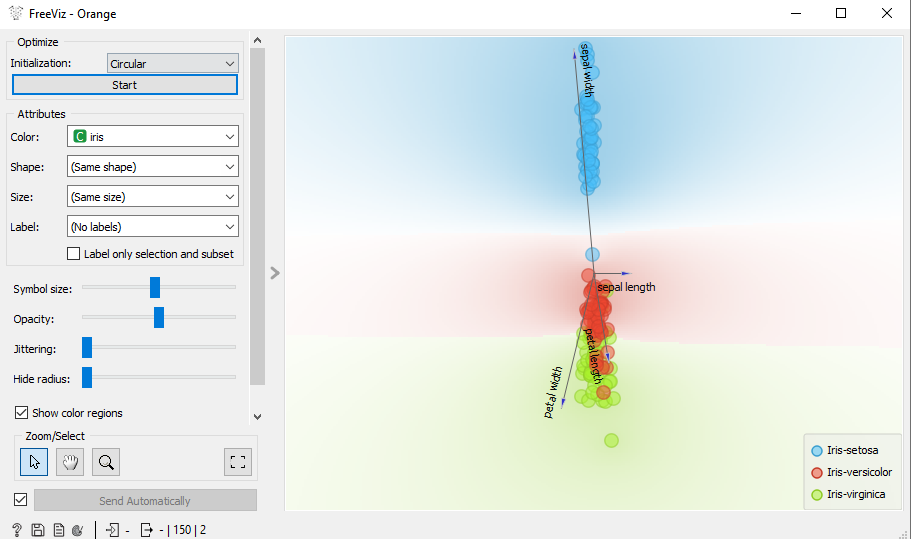
**Distributions**

****

**Linear Projection:**

****

**Freeviz:**

****

**Q3. Difference between Open-Source and Proprietary Software**

**Control of Open-Source and Proprietary Software**

The idea alone that developers and programmers are allowed to examine and modify the source code as deemed necessary shouts aloud control. More control means more flexibility, which means non-programmers can also benefit from the open collaboration. Proprietary software, on the contrary, restricts control only to the owner of the software.

**Security of Open-Source and Proprietary Software**

Because anyone with the required knowledge can add or modify additional features to the program’s source code to make it work better, it allows better sustainability of the software as in discrepancies in the software can be rectified and corrected repeatedly. As developers can work without any restrictions, it allows them to rectify errors that might have missed by the original developers or publishers.

**Driver Support of Open-Source and Proprietary Software**

Open-source software packages often have missing drivers which is natural when you have an open community of users with access to every single line of code. The software may include code modified by one or more individuals, each subject to different terms and conditions. The lack of formal support or sometimes use of generic drivers can put the project at risk. Proprietary software means closed group support which means better performance.

**Usability of Open-Source and Proprietary Software**

Unlike open-source projects, proprietary ones are typically designed keeping in mind a limited group of end users with limited skills. They target a small knit circle of end users unlike projects accomplished within open-source communities. Users outside the programming community won’t even look at the source code let alone modify it.

**Opacity of Open-Source and Proprietary Software**

The viewing restrictions barred the end users from modifying the code let alone debugging it effectively with no control over possible workarounds. The internal structure of proprietary software is strictly closed-access meaning they lack transparency which makes it virtually impossible for users to even suggest modifications or optimizations to the software. Open source, on the other hand, promotes open collaboration which means lesser bugs and faster bug fixes with fewer complexities.

**Q4. Using Anaconda Python create Histogram, Scatter plot and Bar plot for the dataset given below.**

**Dataset-**

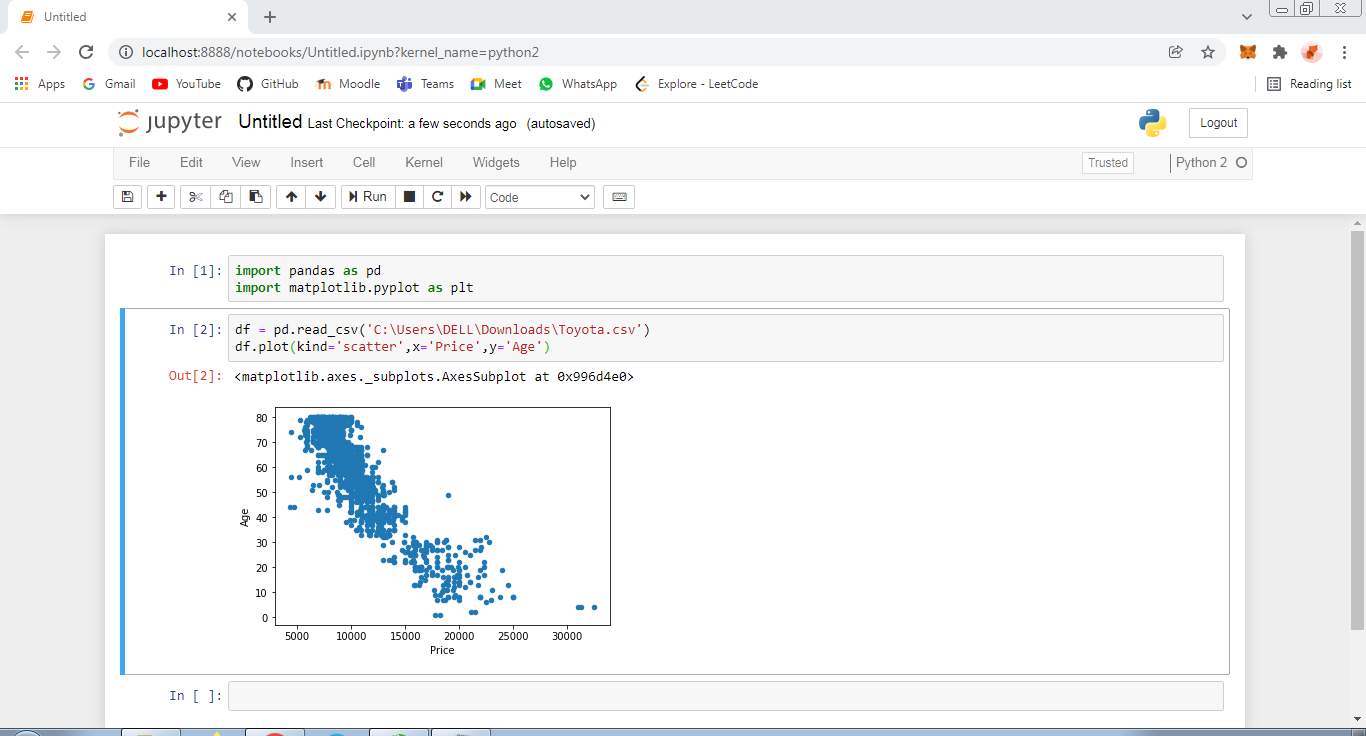
[**https://drive.google.com/file/d/1i11BZFe8Xj9kNq7eeE9KOa\_Iz1KhEdXJ/view**](https://drive.google.com/file/d/1i11BZFe8Xj9kNq7eeE9KOa_Iz1KhEdXJ/view)

**a. Scatter plot- Scatter plot of Price Vs Age**

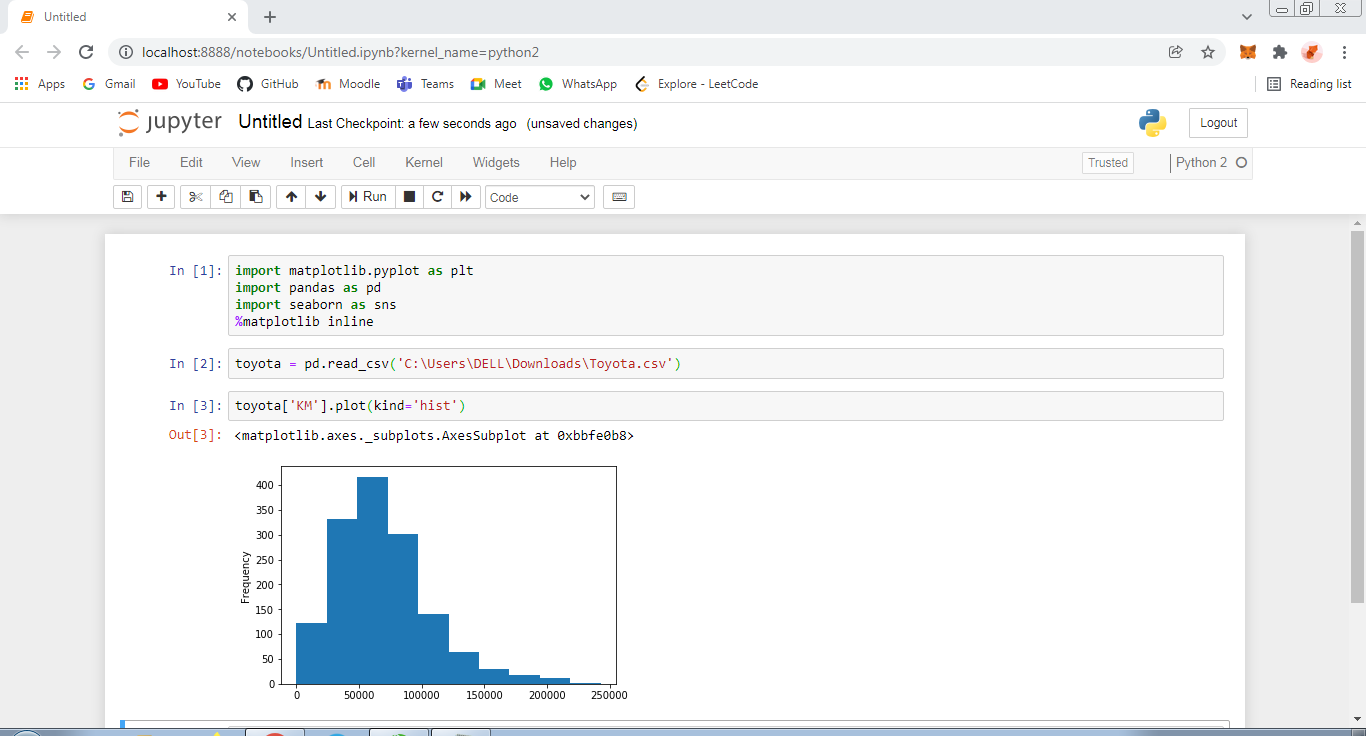
**b. Histogram- for Kilometer and CC**

**c. Bar plot- Bar plot for different fuel types**

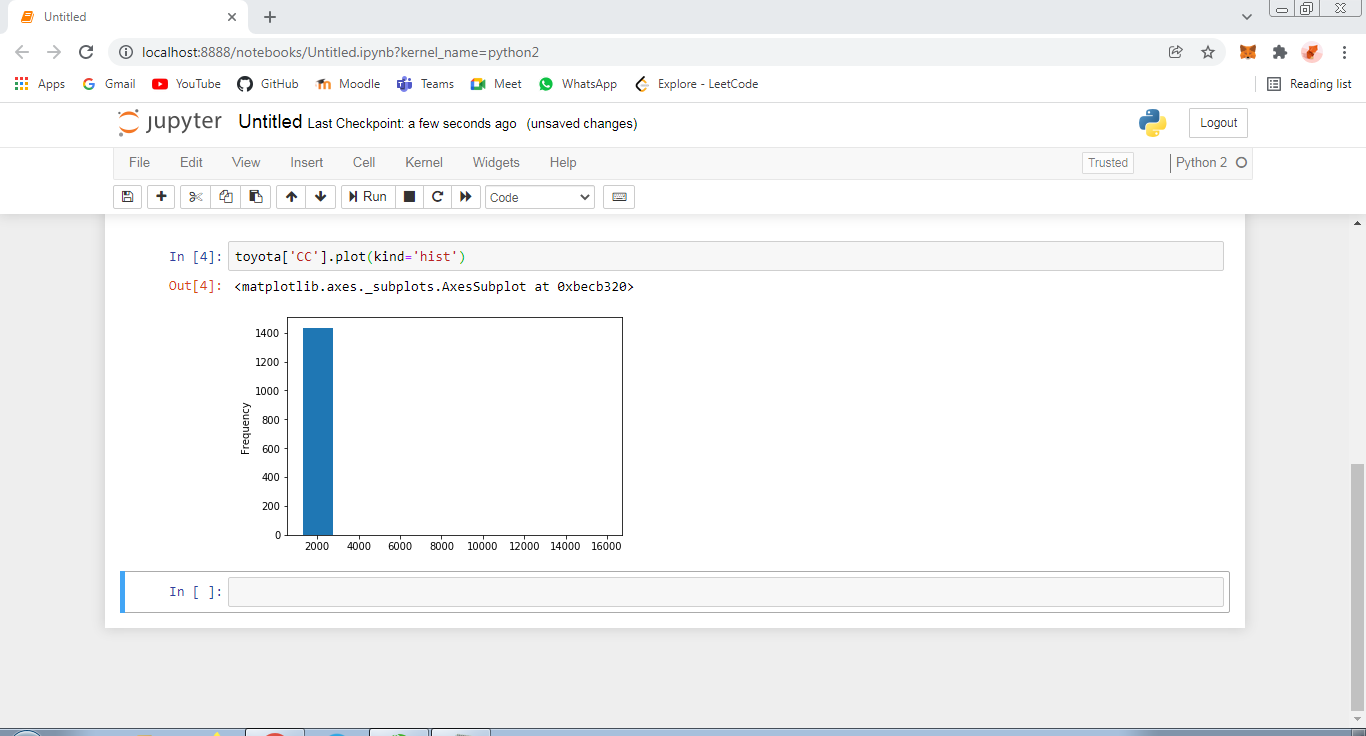
**Scatter Plot – Price v/s Age**

****

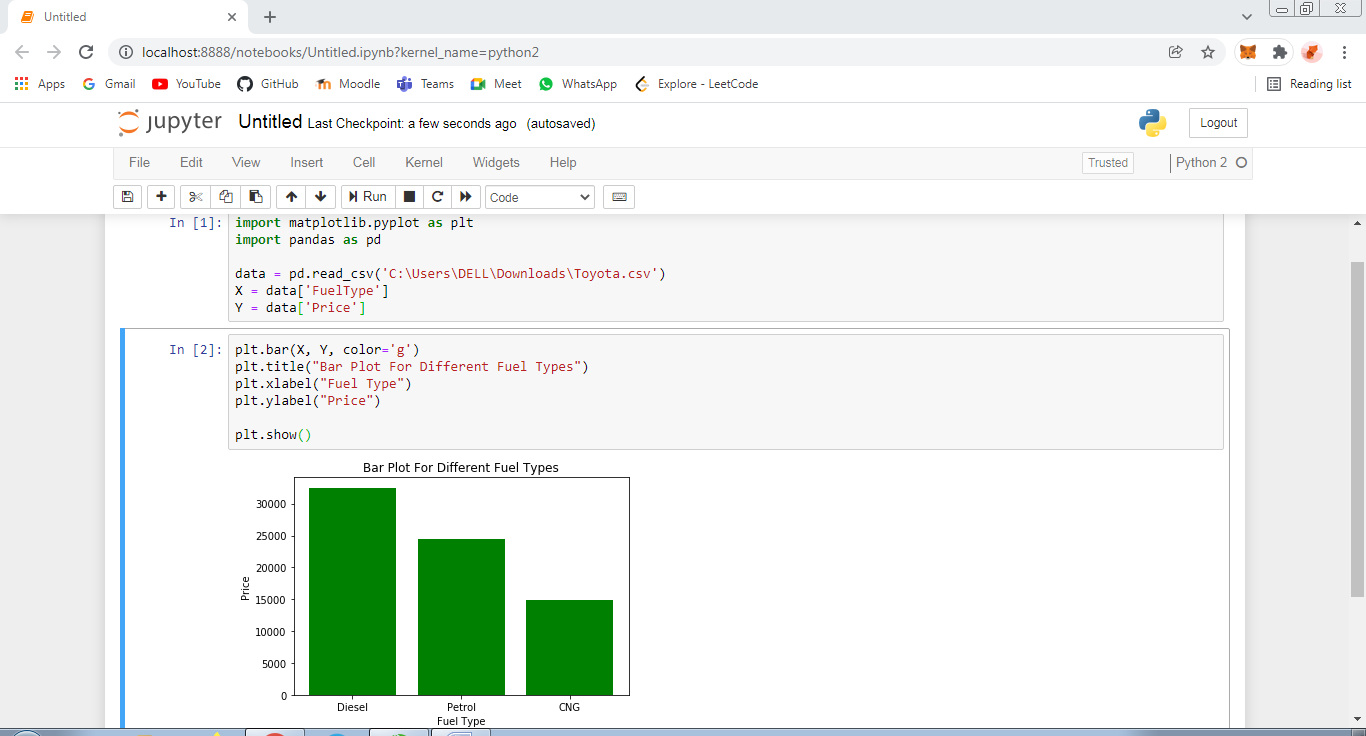
**Histogram – Kilometer**

****

**Histogram – CC**

****

**Bar Plot – Fuel Types**

****

**Q5. Enlist some examples along with its purpose and properties of FOSS and proprietary software w.r.t database.**

**Free and Open Source Software:**

**PostgreSQL**

If you’re from the PHP land (WordPress, Magento, Drupal, etc.), then PostgreSQL will sound foreign to you. However, this relational database software has been around since 1997 and is the top choice in communities like Ruby, Python, Go, etc.



In fact, many developers eventually “graduate” to PostgreSQL for the features it offers or simply for stability. It’s hard to convince someone in a short write-up like this but think of PostgreSQL as a thoughtfully engineered product that never lets you down.

There are many good SQL clients available to connect to the PostgreSQL database for administration and development.

**Unique Features**

PostgreSQL has several fascinating features as compared to other relational databases (specifically, MySQL), such as:

* Built-in data types for Array, Range, UUID, Geolocation, etc.
* Native support for document storage (JSON-style), XML, and key-value storage (Hstore)
* Synchronous and asynchronous replication
* Scriptable in PL, Perl, Python, and more
* Full-text search

**2. MariaDB:**

It was created as a replacement for MySQL by the same person who developed MySQL.

Well, actually, after MySQL was taken over by Oracle in 2010 (by acquiring Sun Microsystems, which, incidentally, is also how Oracle came to control Java), the creator of MySQL started a new open source project called MariaDB.



**Unique features**

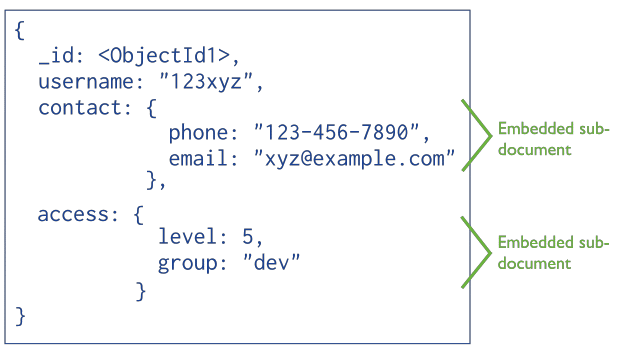
While MariaDB is essentially a clone of MySQL, it’s not strictly true. Ever since the introduction of the database, the differences between the two have been growing. As of writing, adopting MariaDB needs to be a well-thought-through decision on your part. That said, there are plenty of new things going on in MariaDB that may help you make this transition:

* Truly free and open: Since there’s no single corporate entity controlling MariaDB, you can be free of sudden predatory licensing and other worries.
* Several more options of storage engines for specialized needs: for instance, the Spider engine for distributed transactions; ColumnStore for massive data warehousing; the ColumnStore engine for parallel, distributed storage; and many, many more.
* Speed improvements over MySQL, especially due to the Aria storage engine for complex queries.
* Dynamic columns for different rows in a table.
* Better replication capabilities (for example, multi-source replication)
* Several JSON functions
* Virtual columns

**3. MongoDB**

MongoDB was the first non-relational database to make big waves in the tech industry and continues to dominate a fair share of attention.

Unlike relational databases, MongoDB is a “document database,” which stores data in chunks, with related data clumped together in the same chunk. This is best understood by imagining an aggregation of JSON structures like this:



Here, unlike a table-based structure, the user’s contact details and access levels reside inside the same object. Fetching the user object fetches the associated data automatically, and there’s no concept of a join.

**Unique features**

MongoDB has some serious (I almost want to write “kick-ass” to convey the impact, but it wouldn’t be proper on a public website, perhaps) features that have made several seasoned architects abandon the relational land forever:

* A flexible schema for specialized/unpredictable use cases.
* Ridiculously simple sharing and clustering. You just need to set up the configuration for a cluster and forget about it.
* Adding or removing a node from a cluster is drop-dead simple.
* Distributed transactional locks. This feature was missing in the earlier versions but was eventually introduced.
* It is optimized for very fast writes, making it highly suitable for analytics data as a caching system.

**4. SQLite**

Yes, I promised that we were done with relational databases, but SQLite is too cute to ignore.



SQLite is a lightweight C library that provided a relational database storage engine. Everything in this database lives in a single file (with a .sqlite extension) that you can put anywhere in your filesystem. And that’s all you need to use it! Yes, no “server” software to install and no service to connect to.

**Useful features**

Even though SQLite is a lightweight alternative to a database like MySQL, it packs quite a punch. Some of its shocking features are:

* Full support for transactions, with COMMIT, ROLLBACK, and BEGIN.
* Support for 32,000 columns per table
* JSON support
* 64-way JOIN support
* Subqueries, full-text search, etc.
* Maximum database size of 140 terabytes!
* Maximum row size of 1 gigabyte!
* 35% faster than file I/O

**Proprietary Software’s:**

* 1. **Amazon RDS**

Amazon RDS (Relational Database Service) is one of the best DBMS tools.  It has a dedicated secured connection, and it automatically backs up your data through an inbuilt feature. Furthermore, it can resize your entire database activities.

**Pros**

* Users can process the heavy workloads in a single database.
* You have to pay for the used resources.
* It can get you access to MySQL, Oracle, or Microsoft SQL databases.
* Point-in-recovery attracts programmers who want flexibility and scalable storage options.

**Cons**

* It has limited auto-scale options.
* Unavailability of access to physical server to check server logs.
  1. **Oracle RDBMS**

The latest version of the Oracle RDBMS tool encompasses larger databases, takes less space, is more secure, and quickly processes data. It is, in fact, one of the most effective object-relational DBMS tools.

You can run Oracle RDBMS on a PC or a high-powered supercomputer. The active designs, for instance, trigger a dedicated referential system within the software

**Pros**

* You can create partitions to have better administrative control over your data
* Smooth transactional process and data security

**Cons**

* The price range of the tool is discouraging for small businesses
* Client applications are often plugged-in automatically
  1. **Microsoft Access**

Microsoft Access works solely on Windows OS. It's a great tool to create custom databases in numerous formats as per your business needs. The affordable database management system is useful in the IT sector.

**Pros**

* Custom templates of Microsoft Access are ideal for adding web databases and simultaneously *track*ing, *report*ing, or *shar*ing with other users
* Users get to have full access to Office Suite Packages such as Word, Excel, and Outlook.

**Cons**

* License extension to integrate third-party tools can take time.
  1. **Informix**

Informix Dynamics Server is a commercial DBMS tool that is perfect for businesses that want parallel and multi-threaded processing of data. The scalability of the software makes it ideal, plus it has a great in-depth support option.

**Pros**

* It's Online Transactional Processing (OLTP) performance is better than others
* Flexibility to scale small systems to multi-layered distributed nodes
* It can merge relational data in a time series

**Cons**

* The top edition is expensive for small businesses.