Veasualizer – Data Visualization and Machine Learning Tool

A Major Project Report Submitted
In partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering

By

Peddi Yuvaraj	**18N31A05H2**
M V Lakshman	**18N31A05D1**
SK Anwar	**17N31S05L4**

Under the esteemed guidance of

Mrs. P Honey Diana
Assistant Professor

Department of Computer Science and Engineering

Malla Reddy College of Engineering & Technology
(Autonomous Institution- UGC, Govt. of India)
(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA &NAAC with 'A' Grade)
Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100
website: www.mrcet.ac.in

2018-2022

Malla Reddy College of Engineering & Technology

(Autonomous Institution- UGC, Govt. of India)
(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA &NAAC with 'A' Grade)

Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100
website: www.mrcet.ac.in

CERTIFICATE

This is to certify that this is the bonafide record of the project entitled "**Veasualizer – Data Visualisation and Machine Learning Tool**", submitted by P Yuvaraj(18N31A05H2), M Lakshman(18N31A05D1), SK Anwar(17N31A05L4) of B. Tech in the partial fulfilment of the requirements for the degree of Bachelor of Technology in Computer Science and Engineering, Department of CSE during the year 2021-2022. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Internal Guide	**Head of the Department**
Mrs. P Honey Diana	**Dr. S. Shanthi**
Assistant Professor	**Professor**

External Examiner

DECLARATION

We hereby declare that the project titled "**Veasualizer – Data Visualisation and Machine Learning Tool**" submitted to Malla Reddy College of Engineering and Technology (UGC Autonomous), affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a result of original research carried-out in this thesis. It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

Name of Student(s): P Yuvaraj, M V Lakshman, SK Anwar

Hall Ticket Number(s): 18N31A05H2, 18N31A05D1, 17N31A05L4

Degree: Bachelor of Technology in Computer Science and Engineering

Department: Computer Science and Engineering

Title of the project: Veasualizer – Data Visualisation and Machine Learning Tool

Date:

ACKNOWLEDGEMENT

We feel ourselves honoured and privileged to place our warm salutation to our college Malla Reddy College of Engineering and Technology (UGC Autonomous) and our Director **Dr. V. S. K Reddy** and our Principal **Dr. S. Srinivasa Rao**, who gave us the opportunity to have experience in engineering and profound technical knowledge.

We express our heartiest thanks to our Head of the Department **Dr. S. Shanthi** for encouraging us in every aspect of our project and helping us realize our full potential.

We would like to thank our internal guide **Mrs. P Honey Diana** and project coordinator **Dr. V. Chandrashekhar** for their regular guidance and constant encouragement. We are extremely grateful to his valuable suggestions and unflinching co-operation throughout project work.

We would like to thank our class in charge **Mrs. D. Radha** who despite being busy with her duties took time to guide and keep us on the correct path.

We would also like to thank all the supporting staff of the Department of CSE and all other departments who have been helpful directly or indirectly in making our project a success.

We are extremely grateful to our parents for their blessings and prayers for the completion of our project that gave us strength to do our project.

With regards and gratitude

P Yuvaraj - 18N31A05H2

M V Lakshman - 18N31A05D1

SK Anwar - 17N31A05L4

ABSTRACT

Veasualizer is a modern data visualization and a “ML” tool. Using Veasualizer a user with zero knowledge can make beautiful visualizations: Graphs and plots. Not just that, Veasualizer has features using which users can Train, Test and Draw predictions from data by just uploading their dataset to Veasualizer. After uploading dataset, User will have freedom to choose different algorithms suitable for their data and can even choose different parameters for training the dataset on which accuracy of the model depends on. At the end user will get an Exploratory Data Analysis Report (EDA) for their dataset.

Our main objective of Veasualize is to make stuff, which is complex, look easy and useful to everybody. Just by some clicks and uploading datasets, Users will be able to plot different plots like: Scatter plot, Bar graphs, Bubble plots, Box plots, Dot plots, Pie charts and more. And ML algorithms include Regression and Classification algorithms like: Linear, Polynomial, Logistic etc.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Data visualization is a graphic representation that expresses the significance of data. It reveals insights and patterns that are not immediately visible in the raw data. It is an art through which information, numbers, and measurements can be made more understandable. The primary goal of data visualization is to communicate information clearly and effectively through graphical means. It does not mean that data visualization needs to look boring to be functional or extremely sophisticated to look beautiful. To convey ideas effectively, both aesthetic form and functionality need to go hand in hand, providing insights into a rather sparse and complex data set by communicating its key-aspects more intuitively. The main goal of data visualization is to communicate information clearly and effectively through graphical means. It doesn't mean that data visualization needs to look boring to be functional or extremely sophisticated to look beautiful. To convey ideas effectively, both aesthetic form and functionality need to go hand in hand, providing insights into a rather sparse and complex data set by communicating its key-aspects in a more intuitive way. "Data is the new oil" may be a cliche, but it is true. Like oil, data in its raw, unrefined form is worthless. To unlock its value, data needs to be refined, analysed and understood. More and more organizations are seeing potential in their data connections.

2. <u>LITERATURE SURVEY</u>

“Data Visualizations: A Literature Review and Opportunities for Technical and Professional Communication”

Abstract - This paper discusses the need for an integrative literature review on data visualizations. The paper analyses 25 studies across disciplines. The findings suggest there is little agreement on the best way to visualize complex data for lay audiences, but some emerging effective practices are being develop. Pictographs, icon arrays, and bar charts seem to hold promise for comprehension by users, and visualizations need to be kept as simple as possible with attention to integrating other design features such as headings and legends. The review ends with five specific research areas in which technical and professional communicators should focus their attention on empirical studies that examine: interactive displays, merge attention and comprehension, look at numeracy and risk and finally, cross health and medical subjects.

Technical and professional communicators have always been concerned with how information is delivered to potential readers. Complex information from technical subjects, like medicine, science, and the environment, creates additional challenges. Because our lives are filled with a continual flow of information, sometimes the best way to communicate is through visuals. But how can we leverage data visualizations to communicate complex ideas to non-expert audiences? Data visualization means using visuals to present large amounts of data according to certain parameters or categories such as the collection of data into charts, graphs, scatter plots or other standard visualization types.

“Systematic literature review of machine learning methods used in the analysis of real-world data for decision making”

Machine learning is a broad term encompassing a number of methods that allow the investigator to learn from the data. These methods may permit large real-world databases to be more rapidly translated to applications to inform patient-provider decision making.

A wide variety of approaches, algorithms, statistical software, and validation strategies were employed in the application of machine learning methods to inform patient-provider decision making. There is a need to ensure that multiple machine learning approaches are used, the model selection strategy is clearly defined, and both internal and external validation are necessary to be sure that decisions for patient care are being made with the highest quality evidence. Future work should routinely employ ensemble methods incorporating multiple machine learning algorithms.

3. SYSTEM ANALYSIS

3.1 HARWARE AND SOFTWARE REQUIREMENTS

3.1.1 HARDWARE REQUIREMENTS

Processor:	32-bit or 64-bit
Hard Disk	**:** 450MB
RAM	**:** 4GB or above

3.1.2 SOFTWARE REQUIREMENTS

Programming Language	**:** Python 3.7 and above
IDE	**:** Spyder, Visual Studio Code, or any Python IDE
OS	**:** Windows 7 and above

3.2 SOFTWARE REQUIREMENT SPECIFICATION:

A Software Requirements Specification (SRS) – a requirements specification for a software system is a complete description of the behaviour of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or 4 design constraints). System requirements specification: A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analysing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Projects are subject to three sorts of requirements:

- Business requirements describe in business terms what must be delivered or accomplished to provide value.
- Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
- Process requirements describe activities performed by the developing organization. For instance, process requirements could specify. Preliminary investigation examines project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and
- Economic feasibility for adding new modules and debugging old running systems. All system is feasible if they are unlimited resources and infinite time

3.3 EXISTING SYSTEM

We tried to find a project or tool like our project but, there are popular tools like Power BI, Tableau, which can only visualize user's dataset that too, these tools compress a lot of complex features and make simple task of data visualization tough. And there are no tools which can take pre-processed data as input from user and use ML algorithms to forecast or draw predictions from it (Train - Test - Predict). Veasualizer, our proposed system is a simple tool which has all the problems explained above, full filled.

3.3.1 Tableau

Tableau is considered as one of the best Business Intelligence and data visualization tools and has managed to top the charts quite a few times since its launch. The most important quality of this tool is that it makes organizing, managing, visualizing and understanding data extremely easy for its users.

3.3.2 Power BI

Microsoft Power BI is a suite that is a collection of business intelligence tools such as software services, apps and data connectors. It is a cloud-based platform used to consolidate data from varied sources into a single data set. These data sets are used for data visualization, evaluation, and analysis by making sharable reports, dashboards, and apps. Microsoft offers three types of Power BI platforms i.e., Power BI Desktop (a desktop application), Power BI Service (SaaS i.e., Software as a Service) and Power BI Mobile (for iOS and Android devices).

3.4 Proposed System

The whole idea of this project is based on its user base, that is people who are going to use it. Our main objective of Veasualize is to make stuff, which is complex, look easy and useful to everybody. Just by some clicks and uploading datasets, Users will be able to plot different plots like: Scatter plot, Bar graphs, Bubble plots, Box plots, Dot plots, Pie charts and more and ML algorithms include Regression and Classification algorithms like: Linear, Polynomial, Logistic etc

Veasualizer, our proposed system is a simple, user friendly and powerful tool. Veasualizer is a software that can generate plots and graphs from the dataset uploaded by user, and these can be downloaded in jpg or png format. Apart from visualizing user's dataset, Here using this tool users can also draw predictions. In our project (proposed system), we are implementing: 1. Bar Graph, Histogram, Scatterplot, Area plot, Pie plot and more 2. ML algorithms include, 2.1. Regression: Linear, Logistic and Polynomial 2.2. Classification: Decision Tree, Random Forest and SVM

4. SYSTEM STUDY

4.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are:

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement as only minimal or null changes are required for implementing this system.

SOCIAL FEASIIBLITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

5. TECHNOLOGIES USED

5.1 PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. Python, the reference implementation of Python, is open-source software and has a community-based development model, as do nearly all its variant implementations. Python is managed by the non-profit Python Software Foundation. Python features a dynamic type of system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

5.1.1 INTERACTIVE MODE PROGRAMMING

Invoking the interpreter without passing a script file as a parameter brings up the following prompt

```
− $ python Python 2.4.3 (#1, Nov 11, 2010, 13:34:43)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Type the following text at the Python prompt and press the Enter –

```
>>> print "Hello, Python!"
```

If you are running a new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!"); However, in Python version 2.4.3, this produces the following result –

Hello, Python!

5.1.2 **SCRIPT MODE PROGRAMMING**

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file –

Live Demo

print "Hello, Python!"

We assume that you have a Python interpreter set in the PATH variable. Now, try to run this program as follows − $ python test.py

This produces the following result –

Hello, Python!

5.1.3 PYTHON IDENTIFIERS

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores, and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python. Here are naming conventions for Python identifiers –

Class names start with an uppercase letter. All other identifiers start with a lowercase letter. Starting an identifier with a single leading underscore indicates that the identifier is private. Starting an identifier with two leading underscores indicates a strongly private identifier. If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

5.1.4 RESERVED WORDS

The following list shows the Python keywords. These are reserved words, and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and exec not

assert finally or

break for pass

class from print

continue global raise

def if return

del import try

elif in while

else is with

except lambda yield

5.1.5 STANDARD DATA TYPES

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types –

- ➢ Numbers
- ➢ String
- ➢ List
- ➢ Tuple
- ➢ Dictionary

5.1.6 PYTHON NUMBERS

Number data types store numeric values. Number objects are created when you assign a value to them

5.1.7 PYTHON STRINGS

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

5.1.8 PYTHON LISTS

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are like arrays in C. One difference between them is that all the items belonging to a list can be of different data types.

The values stored in a list can be accessed using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

5.1.9 PYTHON TUPLES

A tuple is another sequence data type that is like the list. A tuple consists of several values separated by commas. Unlike lists, however, tuples are enclosed within parentheses. The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as read-only lists.

5.1.10 PYTHON DICTIONARIES

Python's dictionaries are a kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object. Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

5.1.11 PYTHON LIBRARIES

- NumPy: What is NumPy? NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant.
- Pandas: Pandas is a Python library used for working with data sets.
  It has functions for analysing, cleaning, exploring, and manipulating data.
- Streamlit: Streamlit lets you turn data scripts into sharable web apps in minutes, not weeks, It’s all python, open-source, and free! And once you’ve created an app, you can use cloud platform to deploy, manage and share your app.

5.2 MACHINE LEARNING

There are several excellent e-books to study Machine Learning competition entries. However, many will skip some of the explanation on how the solution is developed as these e-books are developed by experts for experts. The objective of this book is to follow a step-by-step workflow, explaining each step and rationale for every decision we take during solution development.

How Machine Learning Beat the Odds It's the classical problem; predict the outcome of a binary event. In laymen terms this means, it either occurred or did not occur. For example, you won or did not win, you passed the test or did not pass the test, you were accepted or not accepted, and you get the point. A common business application is churn or customer retention. Another popular use case is healthcare's mortality rate or survival analysis. Binary events create an interesting dynamic, because we know statistically, a random guess should achieve a 50%

accuracy rate, without creating one single algorithm or writing one single line of code. However, just like autocorrect spell-check technology, sometimes we humans can be too smart for our own good and underperform a coin flip.

5.2.1 A Machine Learning Framework

The competition solution workflow goes through seven stages described as

1. Question or problem definition.
2. Acquire training and testing data.
3. Wrangle, prepare, and cleanse the data.
4. Analyse, identify patterns, and explore the data.
5. Model, predict and solve the problem.
6. Visualize, report, and present the problem-solving steps and final solution.
7. Supply or submit the results.

The workflow indicates general sequence of how each stage may follow the other. However, there are use cases with exceptions.

• We may combine multiple workflow stages. We may analyse by visualizing data.

• Perform a stage earlier than indicated. We may analyse data before and after wrangling. • Perform a stage multiple times in our workflow. Visualize stage may be used multiple times.

• Drop a stage altogether. We may not need supply stage to productize, or service enable our dataset for a competition.

1. **Define the Problem:**

   If data science, big data, machine learning, predictive analytics, business intelligence, or any other buzzword is the solution, then what is the problem? As the saying goes, don't put the cart before the horse. Problems before requirements, requirements before solutions, solutions before design, and design before technology. Too often we are quick to jump on the new shiny technology, tool, or algorithm before determining the actual problem we are trying to solve.

2. **Gather the Data:**

   John Naisbitt wrote in his 1984 (yes, 1984) book Megatrends, we are “drowning in data, yet staving for knowledge." So, chances are, the dataset(s) already exist somewhere, in some format. It may be external or internal, structured, or unstructured, static or streamed, objective or subjective, etc. As the saying goes, you don't have to reinvent the wheel, you just must know where to find it. In the next step, we will worry about transforming "dirty data" to "clean data."

3. **Prepare Data for Consumption:**

   This step is often referred to as data wrangling, a required process to turn “wild” data into “manageable” data. Data wrangling includes implementing data architectures for storage and processing, developing data governance standards for quality and control, data extraction (i.e., ETL and web scraping), and data cleaning to identify aberrant, missing, or outlier data points.

4. **Perform Exploratory Analysis:**

   Anybody who has ever worked with data knows garbage-in, garbage-out (GIGO). Therefore, it is important to deploy descriptive and graphical statistics to look for potential problems, patterns, classifications, correlations, and comparisons in the dataset. In addition, data categorization (i.e., qualitative vs quantitative) is also important to understand and select the correct 10 hypothesis test or data model.

5. **Model Data:**

   Like descriptive and inferential statistics, data modelling can either summarize the data or predict future outcomes. Your dataset and expected results, will determine the algorithms available for use. It's important to remember, algorithms are tools and not magical wands or silver bullets. You must still be the master craft (wo)man that knows how-to select the right tool for the job. An analogy would be asking someone to hand you a Philip screwdriver, and they hand you a flathead screwdriver or worst a hammer. At best, it shows a complete lack of understanding. At worst, it makes completing the project impossible. The same is true in data modelling. The wrong model can lead to poor performance at best and the wrong conclusion (that’s used as actionable intelligence) at worst.

6. **Validate and Implement Data Model:**

   After you've trained your model based on a subset of your data, it's time to test your model. This helps ensure you haven't over fit your model or made it so specific to the selected subset, that it does not accurately fit another subset from the same dataset. In this step we determine if our model over fit, generalize, or under fit our dataset.

7. **Optimize and Strategize:**

   This is the "bionic man" step, where you iterate back through the process to make it better...stronger...faster than it was before. As a machine learning engineer or data scientist, your strategy should be to outsource developer operations and application plumbing, so you have more time to focus on recommendations and design. Once you're able to package your ideas, this becomes your “currency exchange" rate.

5.2.2 Question and problem definition

Define the problem to solve or questions to ask while providing the datasets for training your data science model and testing the model results against a test dataset. The question or problem definition for competition is described
We may also want to develop some early understanding about the domain of our problem.

Classifying: We may want to classify or categorize our samples. We may also want to understand the implications or correlation of different classes with our solution goal. **Correlating**: One can approach the problem based on available features within the training dataset. Which features within the dataset contribute significantly to our solution goal? Statistically speaking is there a correlation among a feature and solution goal? As the feature values change does the solution state change as well, and vice-versa? This can be tested both for numerical and categorical features in the given dataset. We may also want to determine correlation among features other than survival for subsequent goals and workflow stages. Correlating certain features may help in creating, completing, or correcting features.
Converting: For modelling stage, one needs to prepare the data. Depending on the choice of model algorithm one may require all features to be converted to numerical equivalent values. So, for instance converting text categorical values to numeric values. Completing: Data preparation may also require estimating any missing values within a feature. Model algorithms may work best when there are no missing values.
Correcting: We may also analyse the given training dataset for errors or possibly inaccurate values within features and try to correct these values or exclude the samples containing the errors. One way to do this is to detect any outliers among our samples or features. We may also completely discard a feature if it is not contributing to the analysis or may significantly skew the results.
Creating: Can we create new features based on an existing feature or a set of features, such that the new feature follows the correlation, conversion, completion of goals? **Charting**: How to select the right visualization plots and charts depending on the nature of the data and the solution goals?

Corrections: Corrections did after working on data

5.2.3 Best practices

• Performing feature correlation analysis early in the project.
• Using multiple plots instead of overlays for readability.

Acquire data

The Python Pandas package helps us work with our datasets. We start by acquiring the training and testing datasets into Pandas Data-Frame. We also combine these datasets to run certain operations on both datasets together.

Analyse by describing data

Pandas also help describing the datasets answering following questions early in our project.

Import Libraries

The following code is written in Python 3.x. Libraries provide pre-written functionality to perform necessary tasks. The idea is why write ten lines of code, when you can write one line.

Load Data Modelling Libraries

We will use the popular scikit-learn library to develop our machine learning algorithms. In sklearn, algorithms are called Estimators and implemented in their own classes.
For data visualization, we will use the matplotlib and seaborn library. Below are common classes to load.

Meet and Greet Data

This is the meet and greet step. Get to know your data by first name and learn a little bit about it. What does it look like (data type and values), what makes it tick (independent/feature variables(s)), what are its goals in life (dependent/target variable(s)). 12 To begin this step, we first import our data. Next, we use the info() and sample() function, to get a quick and dirty overview of variable data types (i.e. qualitative vs quantitative) The add column name which is dependent variable is our outcome or dependent variable. All other variables are potential predictor or independent variables. It's important to note, more predictor variables do not make a better model, but the right variables.

Which features are categorical?

These values classify the samples into sets of similar samples. Within categorical features are the values nominal, ordinal, ratio, or interval based? Among other things this helps us select the appropriate plots for visualization.

Which features are numerical?

Which features are numerical? These values change from sample to sample. Within numerical features are the values discrete, continuous or time series based? Among other things this helps us select the appropriate plots for visualization.

Which features are mixed data types?

Numeric and Alphanumeric data within the same feature.

Which features may contain errors or typos?

This is harder to review for a large dataset, however reviewing a few samples from a smaller dataset may just tell us outright, which features may require correcting. • Name feature may contain errors or typos as there are several ways used to describe a name including titles, round brackets, and quotes used for alternative or short names.

Which features contain blank, null, or empty values?

These will require correcting. • Some features contain several null values in that order for the training dataset. • Some features are incomplete in case of test dataset. What are the data types for various features? Information will provide dtypes for different features, available data types are float, int, object and bool.

What is the distribution of numerical feature values across the samples?

This helps us determine, among other early insights, how representative is the training dataset of the actual problem domain. And the available numerical features.

What is the distribution of categorical features?

This helps us determine, among other early insights, how representative is the training dataset of the actual problem domain. And the available categorical features.

5.2.4 Wrangle data

• We have collected several assumptions and decisions regarding our datasets and solution requirements. So far, we did not have to change a single feature or value to arrive at these. Let us now execute our decisions and assumptions for correcting, creating, and completing goals.

Developer Documentation:

• pandas.DataFrame

• pandas.DataFrame.info

• pandas.DataFrame.describe

• Indexing and Selecting Data

• pandas.isnull

• pandas.DataFrame.sum

• pandas.DataFrame.mode

• pandas.DataFrame.copy

• pandas.DataFrame.fillna

• pandas.DataFrame.drop

• pandas.Series.value_counts

• pandas.DataFrame.loc

Correlating:

We want to know how well does each feature correlate with Survival. We want to do this early in our project and match these quick correlations with modeled correlations later in the project.

Completing:

There are null values or missing data in the features. Missing values can be bad, because some algorithms don't know how-to handle null values and will fail. While others, like decision trees, can handle null values. Thus, it's important to fix before we start modelling because we will compare several models. There are two common methods, either delete the record or populate the missing value using a reasonable input. It is not recommended to delete the record, especially a large percentage of records, unless it truly represents an incomplete record. Instead, it's best to impute missing values. A basic methodology for qualitative data is imputing using mode. A basic methodology for quantitative data is imputing using mean, median, or mean and randomized standard deviation. There are more complex methodologies, however before deploying; it should be compared to the base model to determine if complexity truly adds value. For this dataset, Subsequent model iterations may modify this decision to determine if it improves the model's accuracy.

1. A simple way is to generate random numbers between mean and standard deviation.
2. More accurate way of guessing missing values is to use other correlated features. Combine methods 1 and 2. So instead of guessing age values based on median, use random numbers between mean and standard deviation, based on sets of categorical and numerical features

Correcting:

Reviewing the data, there does not appear to be any aberrant or non-acceptable data inputs. In addition, we see we may have potential outliers in features. However, since they are reasonable values, we will wait until after we complete our exploratory analysis to determine if we should include or exclude from the dataset. However, we want to use caution when we modify data from its original value, because it may be necessary to create an accurate model.

• This is a good starting goal to execute. By dropping features, we are dealing with fewer data points. Speeds up our notebook and eases the analysis.

• Note that where applicable we perform operations on both training and testing datasets together to stay consistent.

1. Feature may be dropped from our analysis as it contains high ratio of duplicates and there may not be a correlation between Features.

2. Feature may be dropped as it is highly incomplete or contains many null values both in training and test dataset.

3. FeatureID may be dropped from training dataset as.

4. Some features are relatively non-standard, may not contribute directly to dependent feature,

so maybe dropped.

Creating: Feature engineering is when we use existing features to create new features to determine if they provide new signals to predict our outcome. For this dataset, we will create a feature to determine if it played a role in.

Classifying: We may also add to our assumptions based on the problem description noted earlier.

Convert Formats: We will convert categorical data to dummy variables for mathematical analysis.

Developer Documentation:

• Categorical Encoding

• Sklearn LabelEncoder

• Sklearn OneHotEncoder

• Pandas Categorical dtype

• pandas.get_dummies

There are no date or currency formats, but data type formats. Our categorical data imported as objects, which makes it difficult for mathematical calculations. For this dataset, we will convert object data types to categorical dummy variables**.**

Analyse by pivoting features

To confirm some of our observations and assumptions, we can quickly analyse our feature correlations by pivoting features against each other. We can only do so at this stage for features which do not have any empty values. It also makes sense doing so only for features which are categorical, ordinal, or discrete type.

Perform Exploratory Analysis with Statistics

Now that our data is cleaned, we will explore our data with descriptive and graphical statistics to describe and summarize our variables. In this stage, you will find yourself classifying features and determining their correlation with the target variable and each other.

Now we can continue confirming some of our assumptions using visualizations for analysing the data.

Pearson Correlation Heat map

let us generate some correlation plots of the features to see how related one feature is to the next. To do so, we will utilize the Seaborn plotting package which allows us to plot heatmaps very conveniently as follows

Take away from the Plots

One thing that that the Pearson Correlation plot can tell us is that there are not too many features strongly correlated with one another. This is good from a point of view of feeding these features into your learning model because this means that there isn't much redundant or superfluous data in our training set and we are happy that each feature carries with it some unique information. Here are two most correlated features.

Pairplots

Finally let us generate some pairplots to observe the distribution of data from one feature to the other. Once again, we use Seaborn to help us.

Correlating numerical features

Let us start by understanding correlations between numerical features and our solution goal? A histogram chart is useful for analysing continuous numerical variables like Age where banding or ranges will help identify useful patterns. The histogram can indicate distribution of samples using automatically defined bins or equally ranged bands. This helps us answer questions relating to specific bands Note that x-axis in histogram visualizations represents the count of samples or passengers.

Decisions

This simple analysis confirms our assumptions as decisions for subsequent workflow stages.

Correlating numerical and ordinal features

We can combine multiple features for identifying correlations using a single plot. This can be done with numerical and categorical features which have numeric values.

Correlating categorical features

Now we can correlate categorical features with our solution goal.

Correlating categorical and numerical features

We may also want to correlate categorical features (with non-numeric values) and numeric features

Split Training and Testing Data

As mentioned previously, the test file provided is really validation data for competition submission. So, we will use sklearn function to split the training data in two datasets; 75/25 split. This is important, so we don't over fit our model. Meaning, the algorithm is so specific to a given subset, it cannot accurately generalize another subset, from the same dataset. It's important our algorithm has not seen the subset we will use to test, so it doesn't "cheat" by memorizing the answers. We will use sklearn's train_test_split function. In later sections we will also use sklearn's cross validation functions, that splits our dataset into train and test for data modeling comparison.

5.2.5 Model Data:

Data Science is a multi-disciplinary field between mathematics (i.e., statistics, linear algebra, calculus, etc.), computer science (i.e., programming languages, computer systems, etc.) and business management (i.e., communication, subject-matter knowledge, etc.). Most data scientist comes from one of the three fields, so they tend to lean towards that discipline. However, data science is like a three-legged stool, with no one leg being more important than the other. So, this step will require advanced knowledge in mathematics

Machine Learning (ML), as the name suggest, is teaching the machine how-to think and not what to think. While this topic and big data has been around for decades, it is becoming more popular than ever because the barrier to entry is lower, for businesses and professionals alike. This is both good and bad. It’s good because these algorithms are now accessible to more people that can solve more problems in the real-world First, you must understand, that the purpose of machine learning is to solve human problems.

Machine learning can be categorized as: supervised learning, unsupervised learning, and reinforced learning.

Supervised learning is where you train the model by presenting it a training dataset that includes the correct answer.

Unsupervised learning is where you train the model using a training dataset that does not include the correct answer.

And reinforced learning is a hybrid of the previous two, where the model is not given the correct answer immediately, but later after a sequence of events to reinforce learning. We are doing supervised machine learning, because we are training our algorithm by presenting it with a set of features and their corresponding target. We then hope to present it a new subset from the same dataset and have similar results in prediction accuracy.

There are many machine learning algorithms, however they can be reduced to four categories: classification, regression, clustering, or dimensionality reduction, depending on your target variable and data modeling goals. We can generalize that a continuous target variable requires a regression algorithm, and a discrete target variable requires a classification algorithm. One side note, logistic regression, while it has regression in the name, is really a classification algorithm. We will use cross validation and scoring metrics, discussed in later sections, to rank and compare our algorithms' performance.

Machine Learning Selection:

• Sklearn Estimator Overview

• Sklearn Estimator Detail

• Choosing Estimator Mind Map Now we identified that our solution is supervised learning classification algorithm. We can narrow our list of choices. Machine Learning Classification Algorithms:

• Ensemble Methods

• Generalized Linear Models (GLM)

• Naive Bayes

• Nearest Neighbours

• Support Vector Machines (SVM)

• Decision Trees

• Discriminant Analysis

5.2.6 How to Choose a Machine Learning Algorithm (MLA)

When it comes to data modeling, the beginner's question is always, "what is the best machine learning algorithm?" There is no super algorithm that works best in all situations, for all datasets. So the best approach is to try multiple MLAs, tune them, and compare them for your specific scenario.

So, with all this information, where is a beginner to start? I recommend starting with Trees, Bagging, Random Forests, and Boosting. They are basically different implementations of a decision tree, which is the easiest concept to learn and understand. They are also easier to tune, discussed in the next section, than something like SVC.

Evaluate Model Performance

Here the model performance was done with both random forest and gradient boosting with individually for yield and cultivation prediction.

Determine a Baseline Accuracy

Before we decide how-to make our model better, let's determine if our model is even worth keeping. So, think of it like a coin flip problem. If you have a fair coin and you guessed heads or tail, then you have a 50-50 chance of guessing correct. So, let's set 50% as the worst model performance; because anything lower than that, then why do I need you when I can just flip a coin?

How-to Create Your Own Model

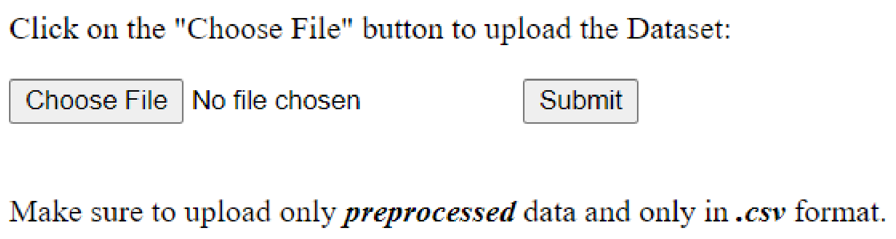
Our accuracy is increasing, but can we do better? Are there any signals in our data? To illustrate this, we're going to build our own decision tree model, because it is the easiest to conceptualize and requires simple addition and multiplication calculations. When creating a decision tree, you want to ask questions that segment your target response what it is here our target variable is numerical and working on regression algorithms.

6 MODULE DESCRIPTION

If we divide this whole project into different modules, The following will be the divided modules:

- Uploading Dataset (Taking Input)
- Choose Operation to be performed
  1. Data Visualization
  2. ML
- Choose Parameters
- Generate Results
- Analyse and Download Results
- Generate and Download EDA

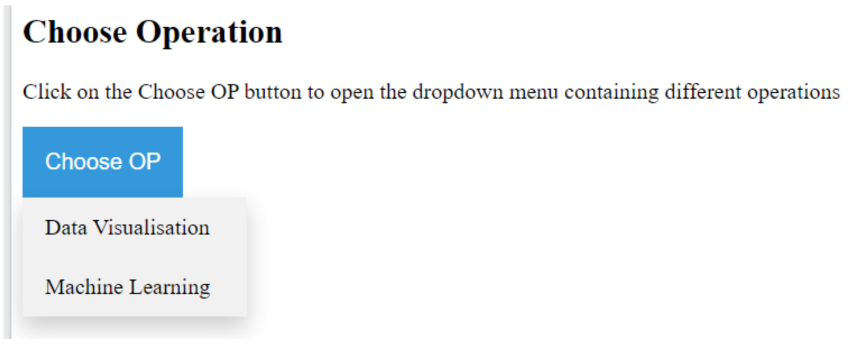
Uploading Dataset:


Click on the "Choose File" button to upload the Dataset:

Make sure to upload only ***preprocessed*** data and only in ***.csv*** format.

This UI will look like above figure, where user can upload his pre-processed dataset into our application in csv format by clicking “Choose File” button and “Submit” after successfully uploading.

Choose Operation:


Choose Operation

Click on the Choose OP button to open the dropdown menu containing different operations

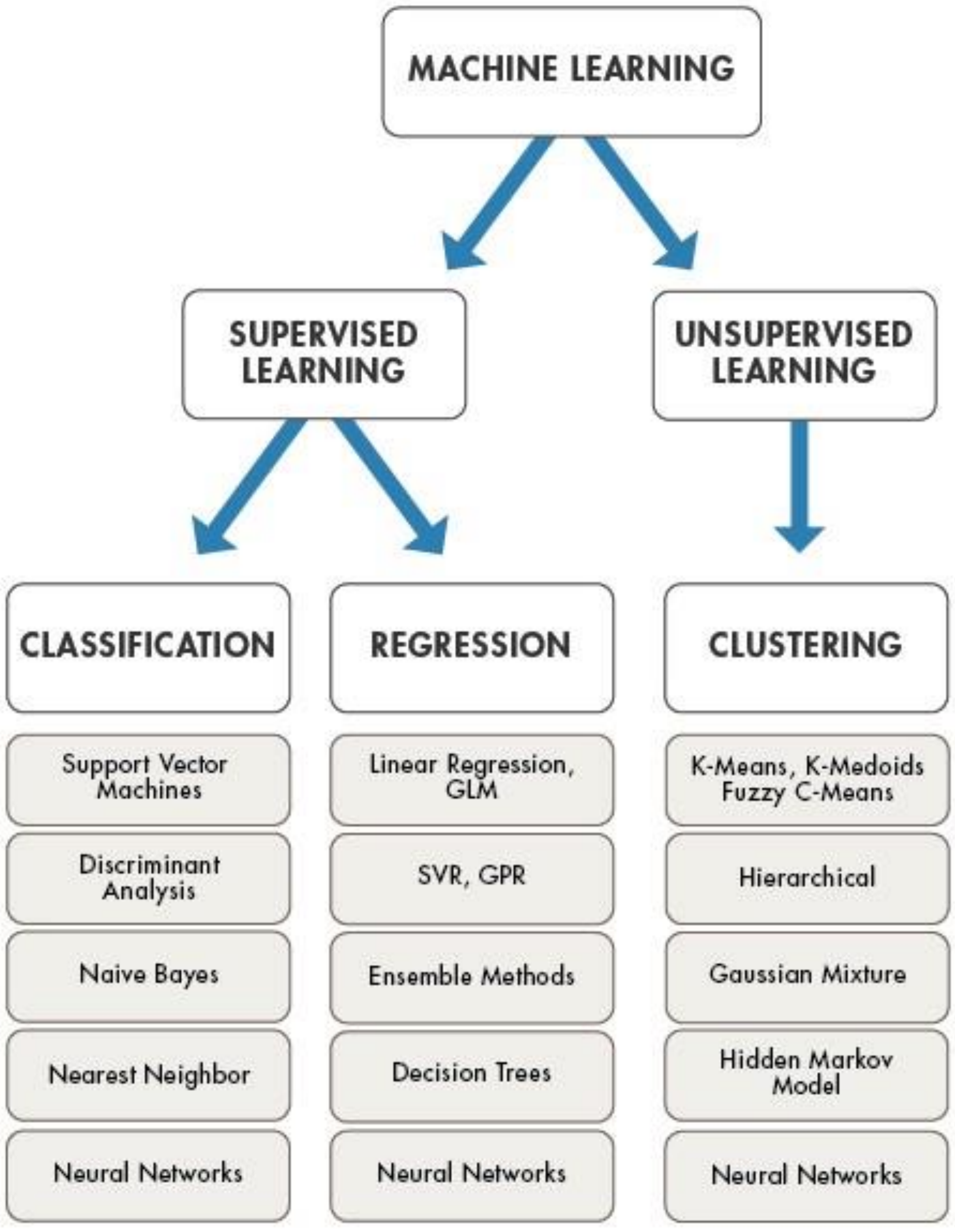
Here user after uploading dataset, will have to choose operation to perform on the dataset. Clicking on dropdown will populate two operations Data visualization and Machine Learning, from which one should be chosen.

Choose Parameters:

If Operation chosen is Data Visualization, then, User have to choose the graphs and plots that he wants his dataset to be visualized in.

Bar Graph
Histogram
Scatter Plot
Area Plot
25%
Pie Plot

If ML is chosen then, type of algorithm, algorithm and parameters that should be given to algorithm chosen are to be done in this module.



Now that, User’s work is done. User uploaded the data and chose what results user is expecting.

Generate and Download Results:

Here Select the plots and curves needed and click on “Classify” button. This will generate results in the right-side space. All the plots, model building, model accuracy and more will be shown to user and can be downloaded.

Results

Graphs, plots that user selected and Accuracy, Precision and Recall of the model build will be in results module

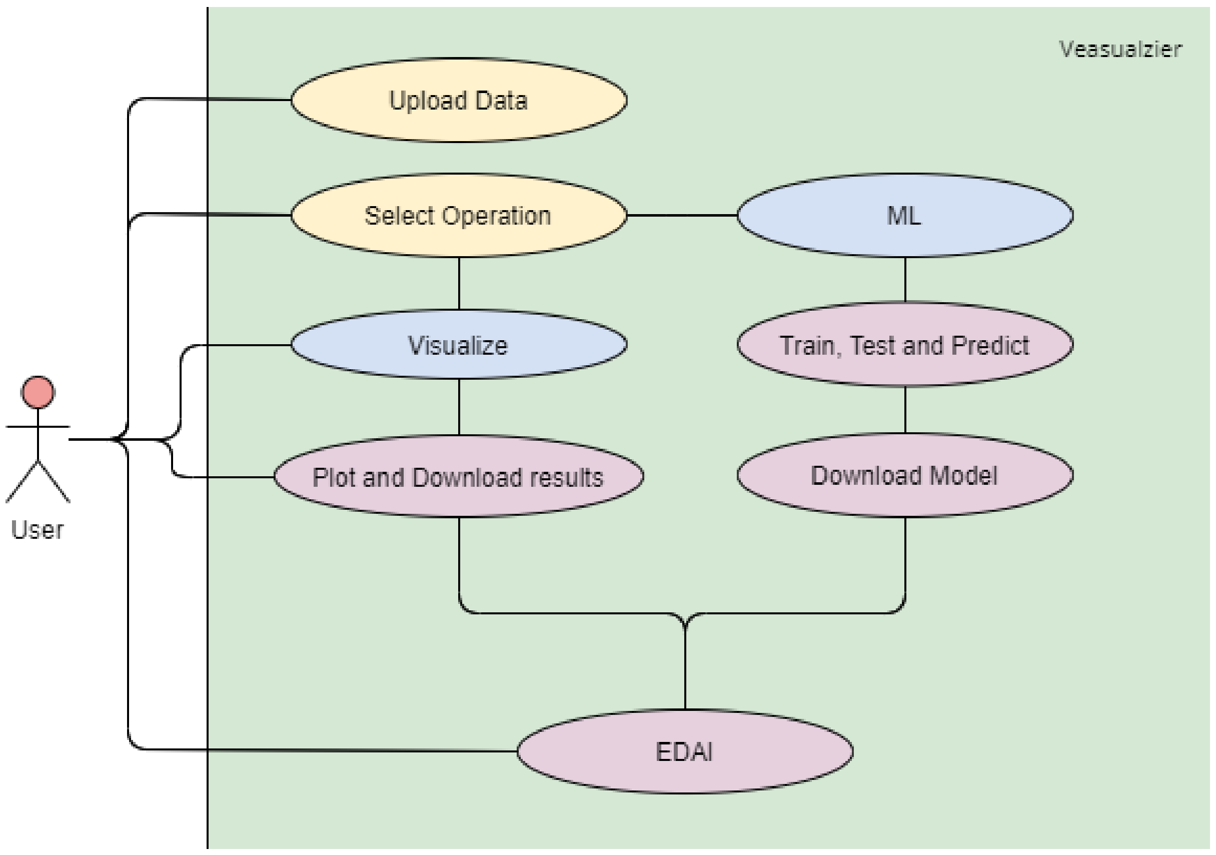
EDA (Exploratory Data Analysis Report):

Exploratory Data Analysis refers to the critical process of performing initial investigations on data to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with help of summary statistics and graphical representations. The primary goal of EDA is to maximize the analyst’s insight into dataset and into the underlying structure of the dataset, while providing all the specific items that an analyst would want to extract from a dataset.
So here using our tool user can directly download an EDA report consisting of all the critical things mentioned above.

This is a modular way of describing steps involved in implementation and execution of this project.

7 SYSTEM DESIGN AND UML DIAGRAMS

Architecture Diagram:



System design transitions from a user-oriented document to programmers or database personnel. The design is a solution, how to approach the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Designing goes through logical and physical stages of development, logical design reviews the present physical system, preparing input and output specification, details of implementation plan and prepare a logical design walkthrough.

7.1 SOFTWARE DESIGN:

In designing the software following principles are followed:

1) **Modularity and partitioning**: Software is designed such that each system should consist of a hierarchy of modules and serve to partition into separate functions.

2) **Coupling:** Modules should have little dependence on other modules of a system.

3) **Cohesion:** Modules should be carried out in a single processing function.

4) **Shared use:** Avoid duplication by allowing a single module to be called by others that need the function it provides.

7.2 UNIFIED MODELING LANGUAGE (UML) :

Unified modelling is a standard language for specifying, visualizing, constructing and documenting the system and its components is a graphical language which provides a vocabulary and set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. It is used to understand, design, configure and control information about the systems.

Depending on the development culture, some of these artifacts are treated formally than others. Such artifacts are not only the deliverables of a project; they are also critical in controlling, measuring, and communicating about a system during its development and after its deployment.

The UML addresses the documentation of a system's architecture and all of its details. The UML also provides a language for expressing requirements and for tests. Finally, the UML provides a language for modelling the activities of project planning and release management.

7.2.1 BUILDING BLOCKS OF UML:

The vocabulary of the UML encompasses three kinds of building blocks:

✓ Things.

✓ Relationships.

✓ Diagrams.

7.2.1.1 Things in the UML:

Things are the abstractions that are first-class citizens in a model; relationships tie these things together; diagrams group interesting collections of things.

There are four kinds of things in the UML:

✓ Structural things.

✓ Behavioural things.

✓ Grouping things.

✓ Notational things.

1. **Structural things** are the nouns of UML models. The structural things used in the project design are:

✓ First, a **class** is a description of a set of objects that share the same attributes, operations, relationships and semantics.

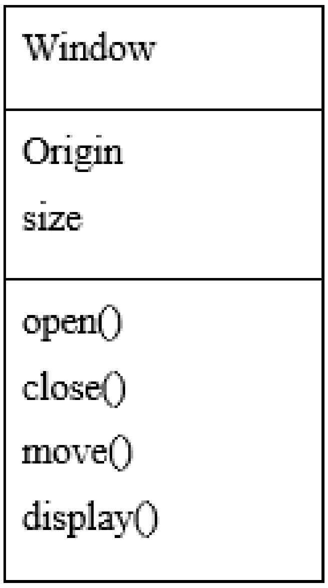


Fig 6.2.1.1.1 Classes

✓ Second, a **use case** is a description of a set of sequence of actions that a system performs that yields an observable result of value to a particular actor.

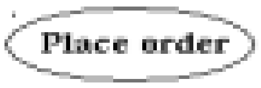


Fig 6.2.1.1.2 Use Cases

✓ Third, a node is a physical element that exists at runtime and represents a computational resource, generally having at least some memory and often processing capability.

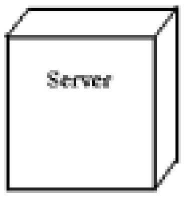


Fig 6.2.1.1.3 Nodes

2. **Behavioural things** are the dynamic parts of UML models. The behavioural thing used is:

✓ Interaction: An interaction is a behaviour that comprises a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose. An interaction involves a number of other elements, including messages, action sequences (the behaviour invoked by a message, and links (the connection between objects).

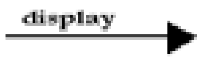


Fig 6.2.1.1.4 Messages

7.2.1.2 Relationships in the UML:

There are four kinds of relationships in the UML:

✓ Dependency.

✓ Association.

✓ Generalization.

✓ Realization.

✓ A **dependency** is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing (the dependent thing).

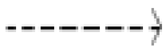


Fig 7.2.1.2.1 Dependencies

✓ An **association** is a structural relationship that describes a set of links, a link being a connection among objects. Aggregation is a special kind of association, representing a structural relationship between a whole and its parts.

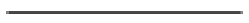


Fig 7.2.1.2.2 Association

✓ A **generalization** is a specialization/ generalization relationship in which objects of the specialized element (the child) are substitutable for objects of the generalized element(the parent).

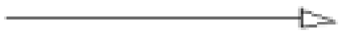


Fig 7.2.1.2.3 Generalization

✓ A **realization** is a semantic relationship between classifiers, where one classifier specifies a contract that another classifier guarantees to carry out.

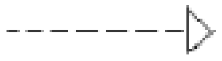


Fig 7.2.1.2.4 Realization

7.3 UML DIAGRAMS

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML comprises two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software systems, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.
The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

- USE CASE DIAGRAM
- CLASS DIAGRAM
- STATE DIAGRAM
- COLLABORATION DIAGRAM
- ACTIVITY DIAGRAM

7.3.1. USE CASE DIAGRAM:

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

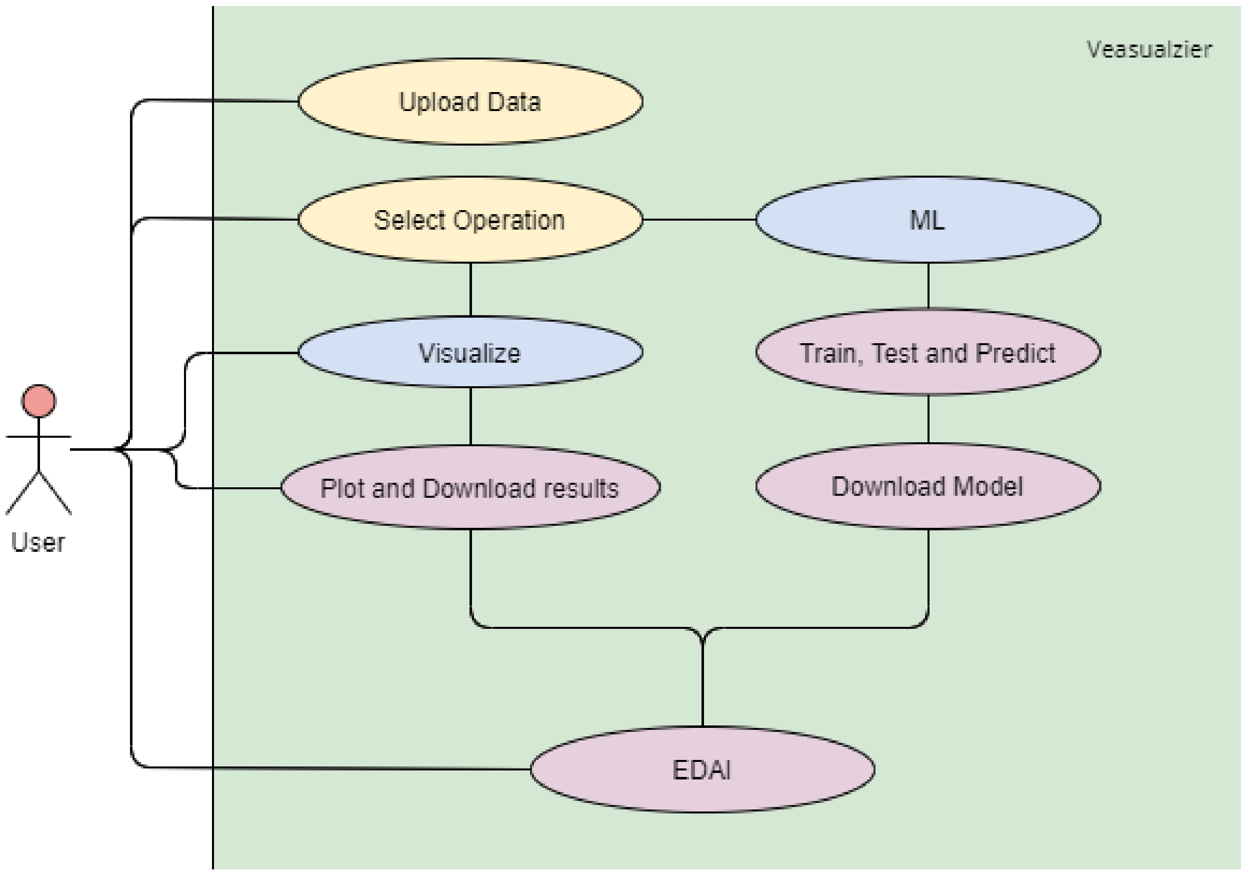


FIG 7.2.2.1 USE CASE DIAGRAM

7.3.2. ACTIVITY DIAGRAM

The process flows in the system are captured in the activity diagram. Like a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions

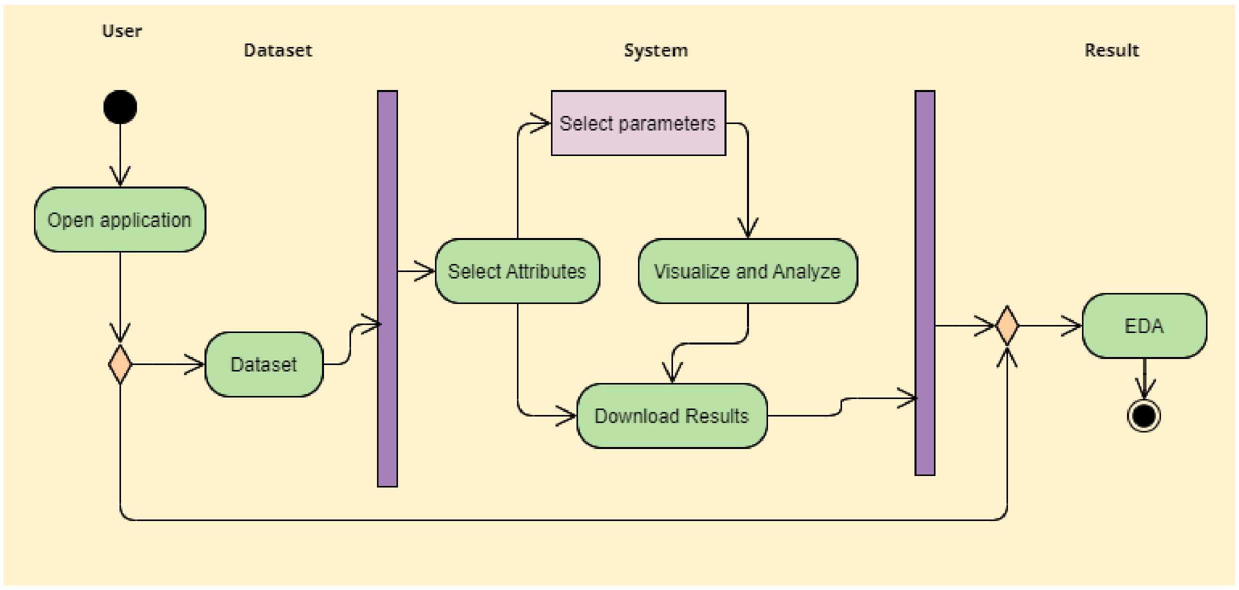


FIG 7.2.2.2 ACTIVITY DIAGRAM

7.3.3 STATE DIAGRAM

A state diagram, as the name suggests, represents the different states that objects in the system undergo during their life cycle.

● Objects in the system change states in response to events.

● In addition to this, a state diagram also captures the transition of the object's state from an initial state to a final state in response to events affecting the system.

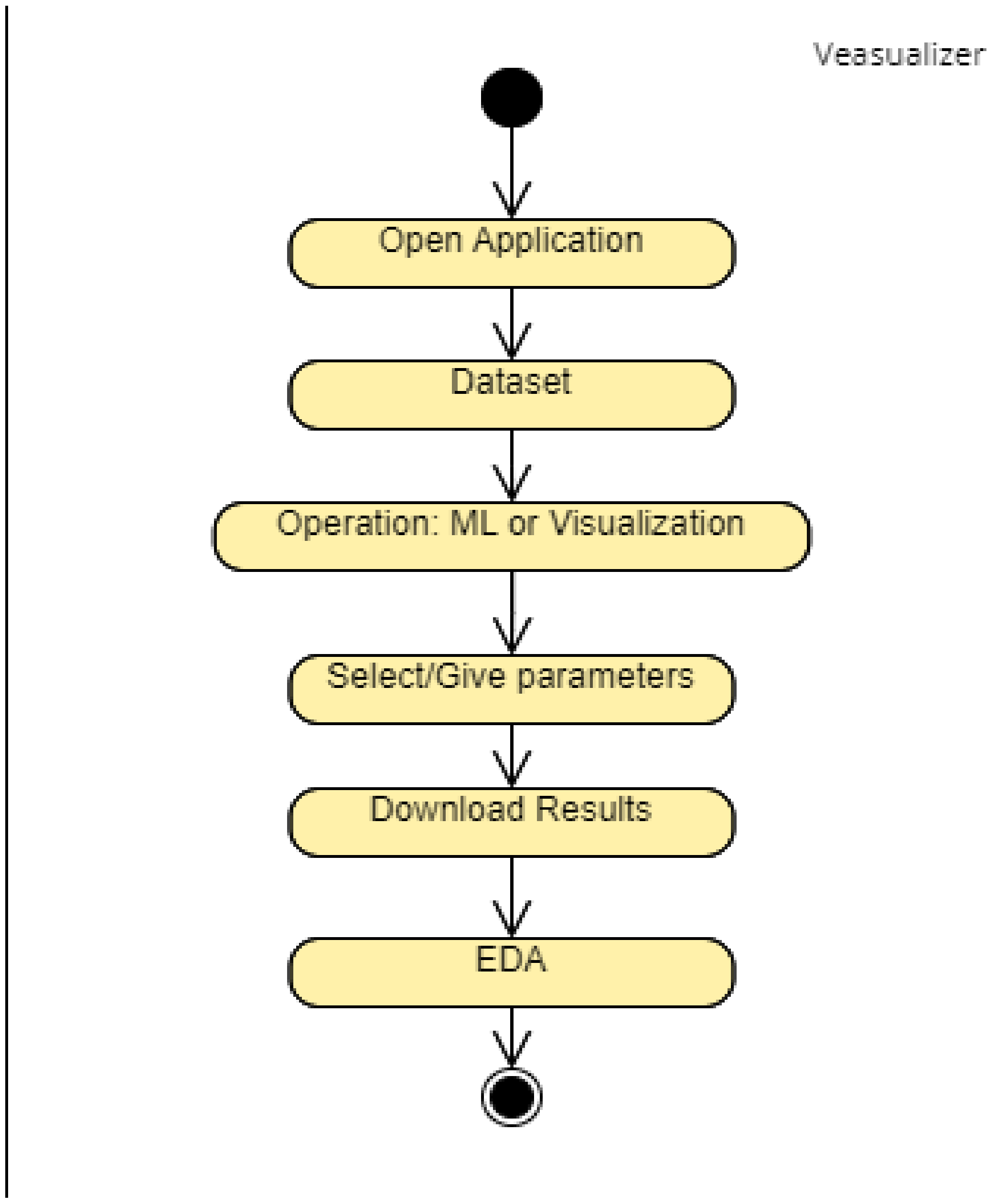


FIG 7.2.2.3 STATE DIAGRAM

7.3.4. CLASS DIAGRAM

The class diagram is used to refine the use case diagram and define a detailed design of the system.

- The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes.
- The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities

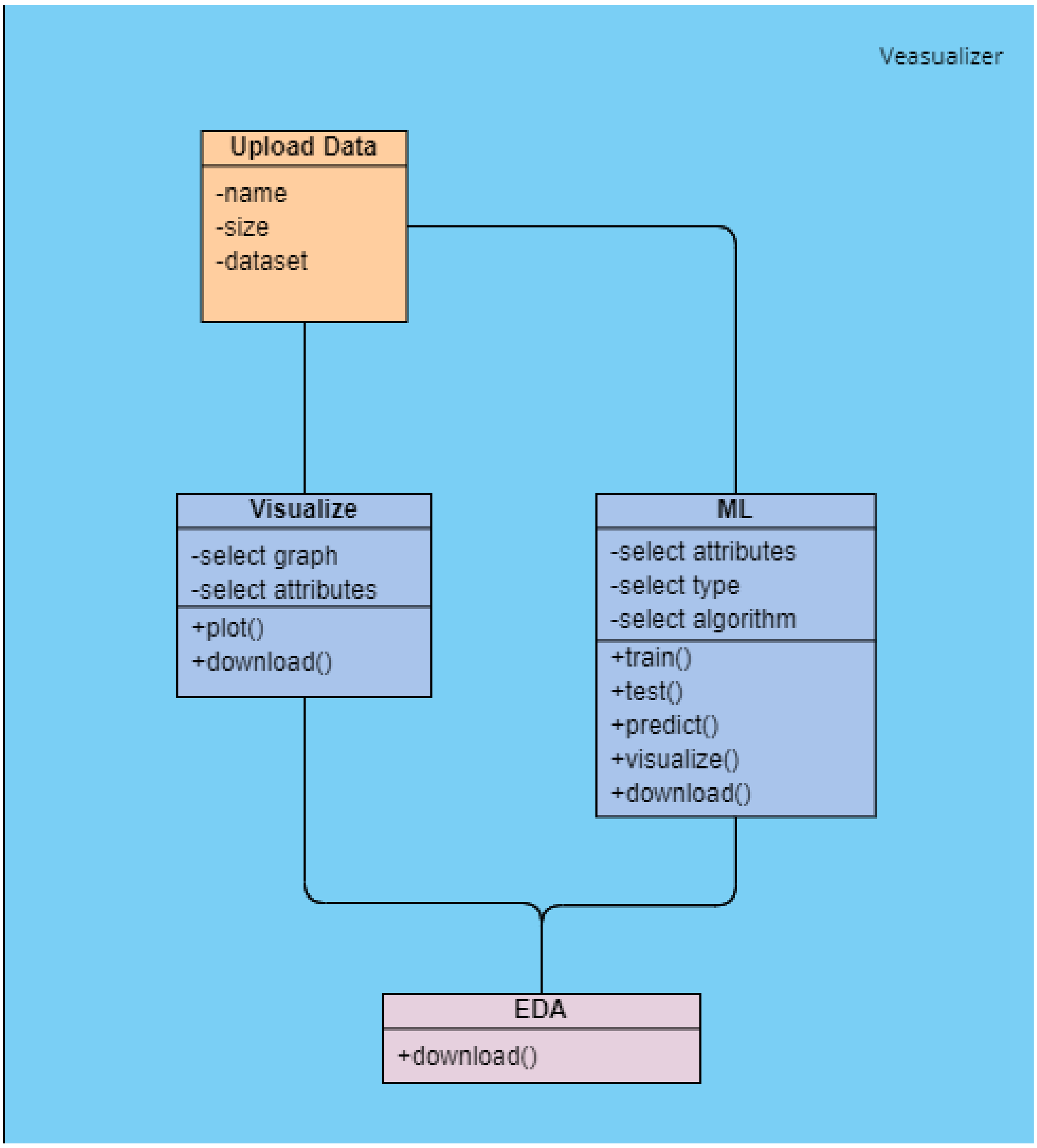


FIG 7.2.2.4 CLASS DIAGRAM

7.3.5 COLLABORATION DIAGRAM

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

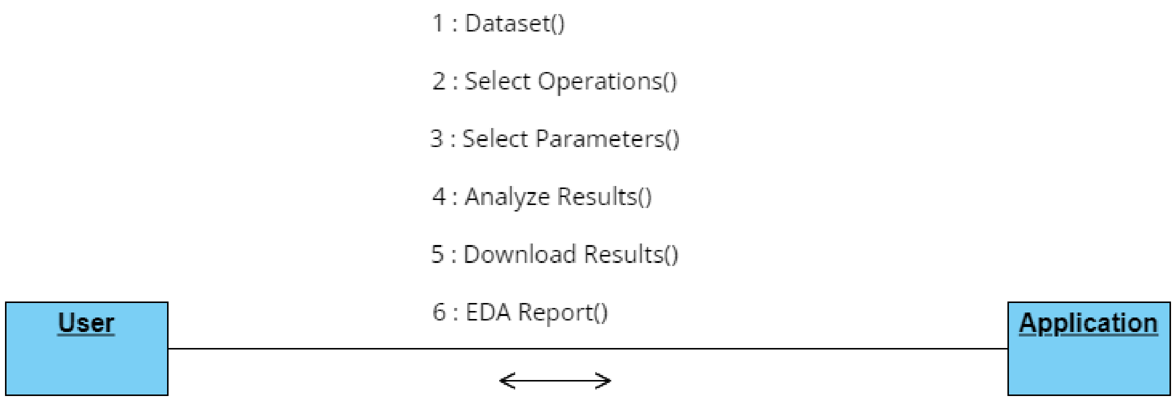


FIG 7.2.2.5 COLLAB DIAGRAM

8 IMPLEMENTATION

8.1 Software Environment

Why Python:

- Python is a high level, structured, open-source programming language that can be used for a wide variety of programming tasks.
- Python within itself is an interpreted programming language that is automatically compiled into bytecode before execution.
- It is also a dynamically typed language that includes (but does not require one to use) object-oriented features.
- NASA has used Python for its software systems and has adopted it as the standard scripting language for its Integrated Planning System.
- Python is also extensively used by Google to implement many components of its Web Crawler and Search Engine & Yahoo! for managing its discussion groups.

Streamlit:

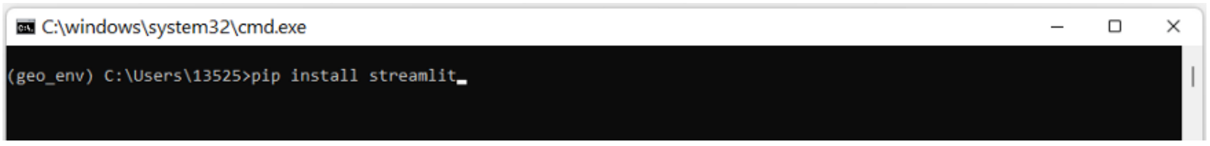
The fastest way to build and share data apps.

Streamlit lets you turn data scripts into sharable web apps in minutes, not weeks. It’s all Python, open-source, and free! And once you’ve created an app you can use our cloud platform to deploy, manage, and share your app!

Installation:

```
pip install streamlit
streamlit hello
```

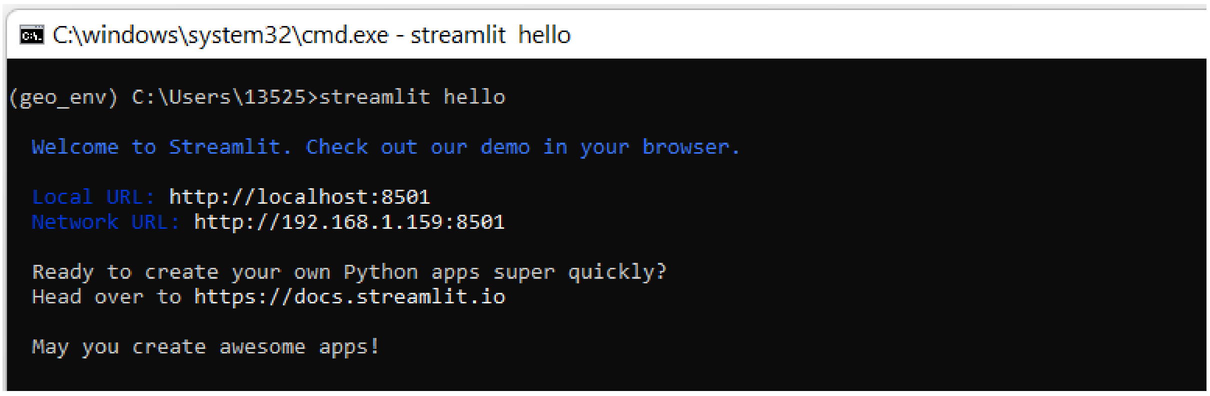
Streamlit can also be installed in a virtual environment on Windows, Mac, and Linux.



```
C:\windows\system32\cmd.exe

(geo_env) C:\Users\13525>pip install streamlit
```

Type ‘Streamlit hello’ in the command line. If the install is successful, you will see the welcome message shown in your terminal and Streamlit’s Hello app appear in your web browser.



C:\windows\system32\cmd.exe - streamlit hello

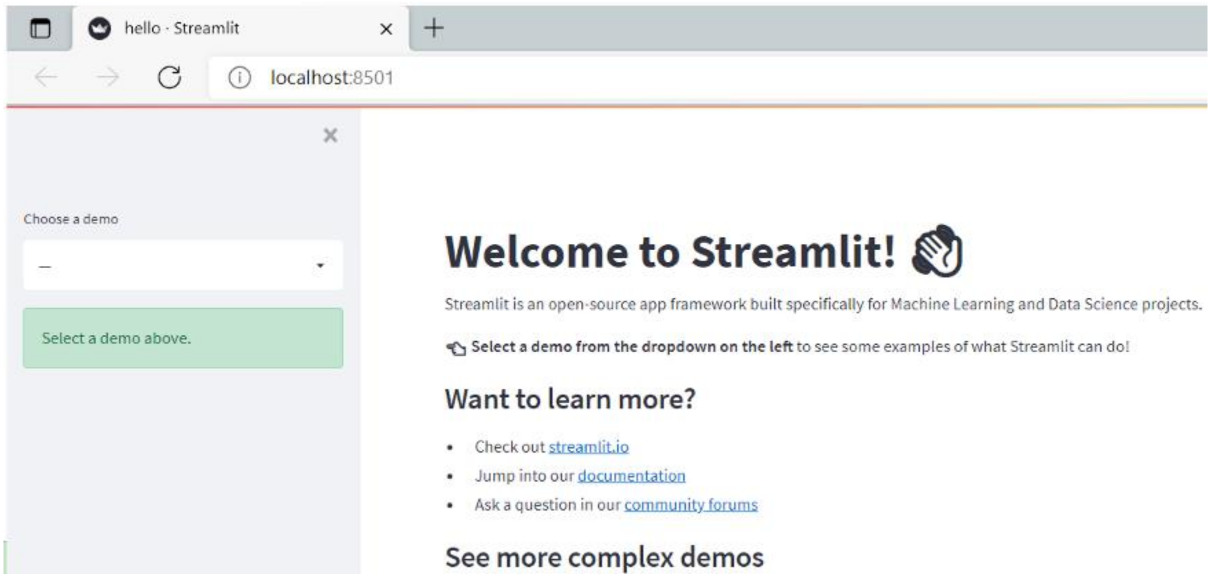
```
(geo_env) C:\Users\13525>streamlit hello

  Welcome to Streamlit. Check out our demo in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.1.159:8501

  Ready to create your own Python apps super quickly?
  Head over to https://docs.streamlit.io

  May you create awesome apps!
```



Welcome to Streamlit!

Streamlit is an open-source app framework built specifically for Machine Learning and Data Science projects.

Select a demo from the dropdown on the left to see some examples of what Streamlit can do!

Want to learn more?

- Check out streamlit.io
- Jump into our documentation
- Ask a question in our community forums

See more complex demos

Streamlit vs. Flask

Streamlit is a data dashboarding tool, while Flask is a web framework. Serving pages to users is an important but small component of data dashboards. Flask doesn't have any data visualization, manipulation, or analytical capabilities (though since it's a general Python library, it can work well with other libraries that perform these tasks). Streamlit is an all-in-one tool that encompasses web serving as well as data analysis.

- Use Streamlit if you want a structured data dashboard with many of the components you'll need already included. Use Streamlit if you want to build a data dashboard with common components and don't want to reinvent the wheel.
- Use Flask if you want to build a highly customized solution from the ground up and you have the engineering capacity.

8.2 Coding:

Import required packages:

```
import streamlit as st
import pandas as pd
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import plot_confusion_matrix, plot_roc_curve, plot_precision_recall_curve
from sklearn.metrics import precision_score, recall_score
```

Code to build side bar:

```
def main():
    st.title("Binary Classification Web App")
    st.sidebar.title("Binary Classification Web App")
    st.markdown("Are your mushrooms edible or poisonous? ")
    st.sidebar.markdown("Are your mushrooms edible or poisonous? ")
    def load_data():
        data = pd.read_csv("mushrooms.csv")
        labelencoder=LabelEncoder()
        for col in data.columns:
            data[col] = labelencoder.fit_transform(data[col])
        return data
    def split(df):
        y = df.type
        x = df.drop(columns=['type'])
        x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)
        return x_train, x_test, y_train, y_test
```

Code for selecting graphs:

```
def plot_metrics(metrics_list):
    if 'Confusion Matrix' in metrics_list:
        st.subheader("Confusion Matrix")
        plot_confusion_matrix(model, x_test, y_test, display_labels=class_names)
        st.pyplot()
    if 'ROC Curve' in metrics_list:
        st.subheader("ROC Curve")
        plot_roc_curve(model, x_test, y_test)
        st.pyplot()
    if 'Precision-Recall Curve' in metrics_list:
        st.subheader('Precision-Recall Curve')
        plot_precision_recall_curve(model, x_test, y_test)
st.pyplot()
```

Code for selecting Classifier:

```
if classifier == 'Support Vector Machine (SVM)':
    st.sidebar.subheader("Model Hyperparameters")
    #choose parameters
    C = st.sidebar.number_input("C (Regularization parameter)", 0.01, 10.0,
step=0.01, key='C_SVM')
    kernel = st.sidebar.radio("Kernel", ("rbf", "linear"), key='kernel')
    gamma = st.sidebar.radio("Gamma (Kernel Coefficient)", ("scale", "auto"),
key='gamma')
    metrics = st.sidebar.multiselect("What metrics to plot?", ('Confusion Matrix',
'ROC Curve', 'Precision-Recall Curve'))
    if st.sidebar.button("Classify", key='classify'):
        st.subheader("Support Vector Machine (SVM) Results")
        model = SVC(C=C, kernel=kernel, gamma=gamma)
        model.fit(x_train, y_train)
        accuracy = model.score(x_test, y_test)
        y_pred = model.predict(x_test)
```

```
            st.write("Accuracy: ", accuracy.round(2))
            st.write("Precision: ", precision_score(y_test, y_pred,
labels=class_names).round(2))
            st.write("Recall: ", recall_score(y_test, y_pred, labels=class_names).round(2))
            plot_metrics(metrics)

    if classifier == 'Logistic Regression':
        st.sidebar.subheader("Model Hyperparameters")
        C = st.sidebar.number_input("C (Regularization parameter)", 0.01, 10.0,
step=0.01, key='C_LR')
        max_iter = st.sidebar.slider("Maximum number of iterations", 100, 500,
key='max_iter')
        metrics = st.sidebar.multiselect("What metrics to plot?", ('Confusion Matrix',
'ROC Curve', 'Precision-Recall Curve'))
        if st.sidebar.button("Classify", key='classify'):
            st.subheader("Logistic Regression Results")
            model = LogisticRegression(C=C, penalty='l2', max_iter=max_iter)
            model.fit(x_train, y_train)
            accuracy = model.score(x_test, y_test)
            y_pred = model.predict(x_test)
            st.write("Accuracy: ", accuracy.round(2))
            st.write("Precision: ", precision_score(y_test, y_pred,
labels=class_names).round(2))
            st.write("Recall: ", recall_score(y_test, y_pred, labels=class_names).round(2))
            plot_metrics(metrics)

    if classifier == 'Random Forest':
        st.sidebar.subheader("Model Hyperparameters")
        n_estimators = st.sidebar.number_input("The number of trees in the forest", 100,
5000, step=10, key='n_estimators')
        max_depth = st.sidebar.number_input("The maximum depth of the tree", 1, 20,
step=1, key='n_estimators')
        bootstrap = st.sidebar.radio("Bootstrap samples when building trees", ('True',
'False'), key='bootstrap')
```

```
        metrics = st.sidebar.multiselect("What metrics to plot?", ('Confusion Matrix',
'ROC Curve', 'Precision-Recall Curve'))
        if st.sidebar.button("Classify", key='classify'):
            st.subheader("Random Forest Results")
            model = RandomForestClassifier(n_estimators=n_estimators,
max_depth=max_depth, bootstrap=bootstrap, n_jobs=-1)
            model.fit(x_train, y_train)
            accuracy = model.score(x_test, y_test)
            y_pred = model.predict(x_test)
            st.write("Accuracy: ", accuracy.round(2))
            st.write("Precision: ", precision_score(y_test, y_pred,
labels=class_names).round(2))
            st.write("Recall: ", recall_score(y_test, y_pred, labels=class_names).round(2))
plot_metrics(metrics)
```

Code for generating EDA:

```
import pandas as pd
from pandas_profiling import ProfileReport
df = pd.read_csv('covid_19_data.csv')
# Generating a report
profile = ProfileReport(df)
profile.to_file(output_file='covid_19_analysis1.html')
# Generating a minimal report
profile = ProfileReport(df, minimal=True)
profile.to_file(output_file='covid_19_analysis_minimal.html')
```

9 TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

9.1 TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	:	identified classes of valid input must be accepted.
Invalid Input	:	identified classes of invalid input must be rejected.
Functions	:	identified functions must be exercised.
Output	:	identified classes of application outputs must be exercised.
Systems/Procedures	:	interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure, and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document.

It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually, and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results

All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results:

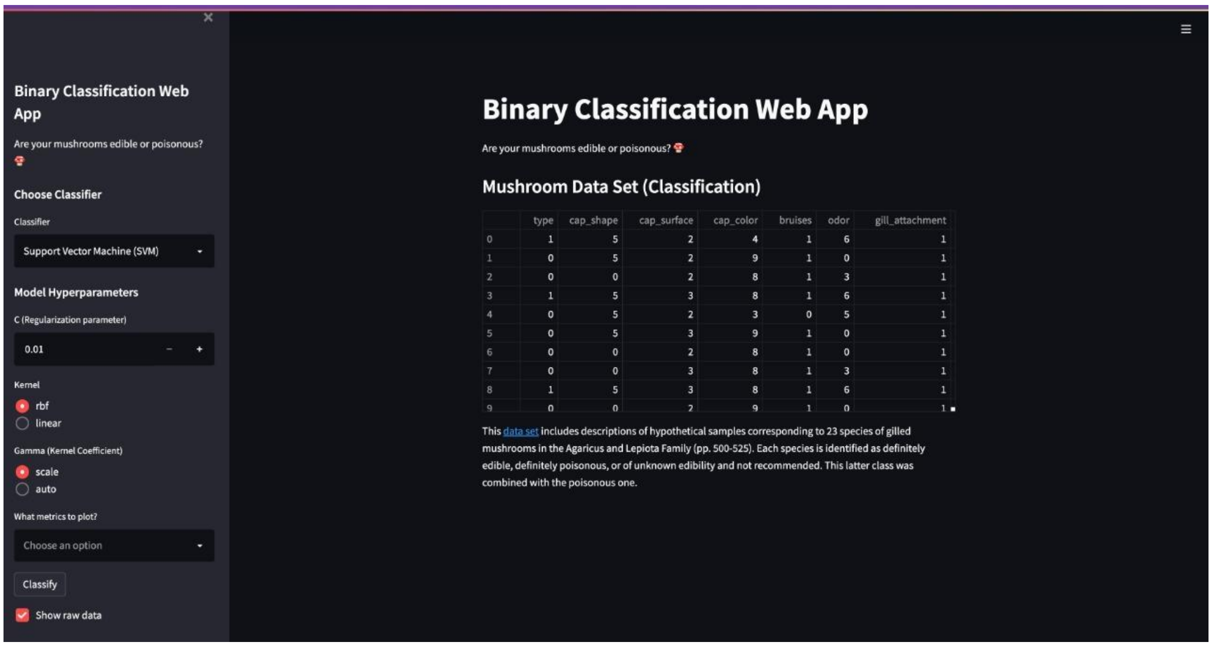
All the test cases mentioned above passed successfully. No defects encountered.

9.2 TEST CASES

TEST CASE NUMBER	SCENARIO	VALUE	EXPECTED RESULT	TEST RESULT
1.	Not uploading Dataset	No Dataset	Please Re-upload Dataset	PASS
2.	Uploading invalid dataset	Wrong Dataset	Invalid Dataset, please check and Re-upload Dataset.	PASS
3.	Not selecting Graphs	No graphs selected	Please select graphs to continue	PASS
4.	Not selecting Algorithm	No Algorithm selected	Please select algorithm to continue	PASS
5.	Not uploading Dataset for EDA	No Dataset	Please upload Dataset for generating EDA	PASS

10 OUTPUT SCREENS

The Overall User Interface (Before and after selecting relevant options):



Binary Classification Web App

Are your mushrooms edible or poisonous? 🍄

Choose Classifier

Classifier: Support Vector Machine (SVM)

Model Hyperparameters

C (Regularization parameter): 0.01

Kernel: [✓ rbf] linear

Gamma (Kernel Coefficient): [✓ scale] auto

What metrics to plot? Choose an option

Classify

✓ Show raw data

Binary Classification Web App

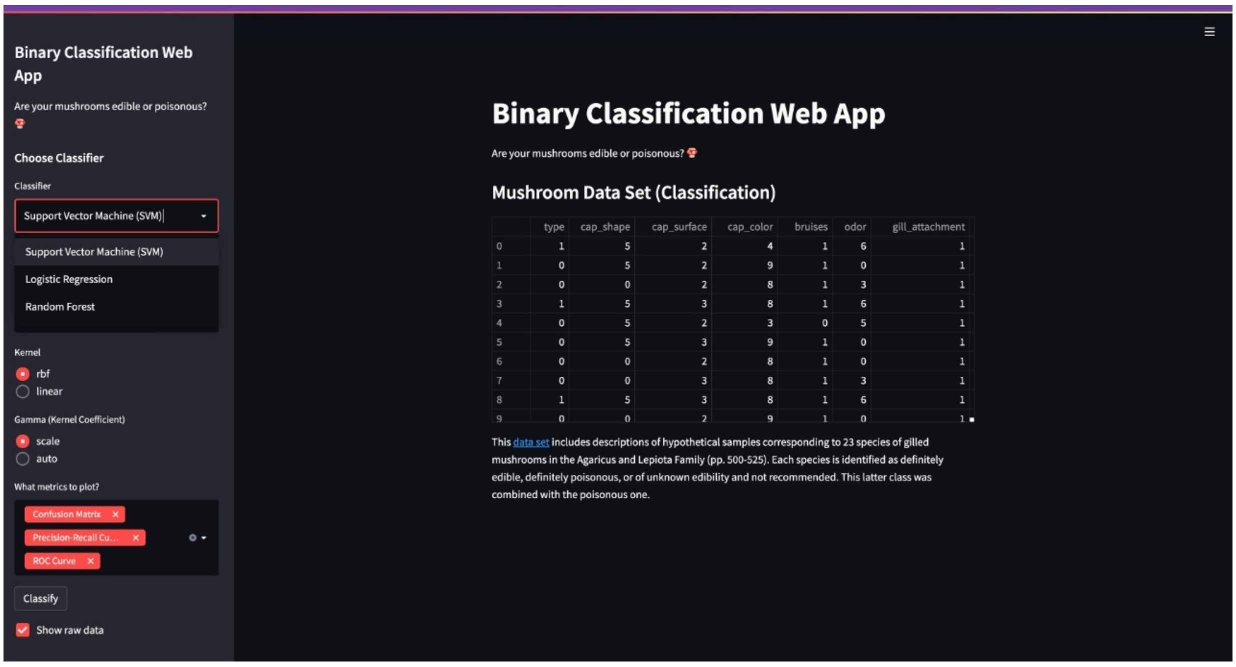
Are your mushrooms edible or poisonous? 🍄

Mushroom Data Set (Classification)

	type	cap_shape	cap_surface	cap_color	bruises	odor	gill_attachment
0	1	5	2	4	1	6	1
1	0	5	2	9	1	0	1
2	0	0	2	8	1	3	1
3	1	5	3	8	1	6	1
4	0	5	2	3	0	5	1
5	0	5	3	9	1	0	1
6	0	0	2	8	1	0	1
7	0	0	3	8	1	3	1
8	1	5	3	8	1	6	1
9	0	0	2	9	1	0	1

This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family (pp. 500-525). Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one.

Figure: 10.1



Binary Classification Web App

Are your mushrooms edible or poisonous? 🍄

Choose Classifier

Classifier: Support Vector Machine (SVM)

- Support Vector Machine (SVM)
- Logistic Regression
- Random Forest

Kernel: [✓ rbf] linear

Gamma (Kernel Coefficient): [✓ scale] auto

What metrics to plot? Confusion Matrix, Precision-Recall Cu..., ROC Curve

Classify

✓ Show raw data

Binary Classification Web App

Are your mushrooms edible or poisonous? 🍄

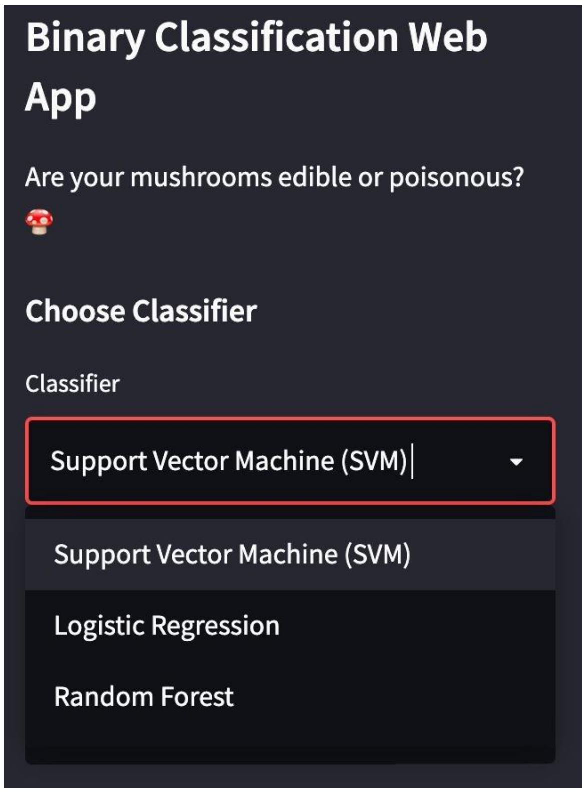
Mushroom Data Set (Classification)

	type	cap_shape	cap_surface	cap_color	bruises	odor	gill_attachment
0	1	5	2	4	1	6	1
1	0	5	2	9	1	0	1
2	0	0	2	8	1	3	1
3	1	5	3	8	1	6	1
4	0	5	2	3	0	5	1
5	0	5	3	9	1	0	1
6	0	0	2	8	1	0	1
7	0	0	3	8	1	3	1
8	1	5	3	8	1	6	1
9	0	0	2	9	1	0	1

This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family (pp. 500-525). Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one.

Figure: 10.2

UI for choosing ML Algorithm: Figure: 10.3

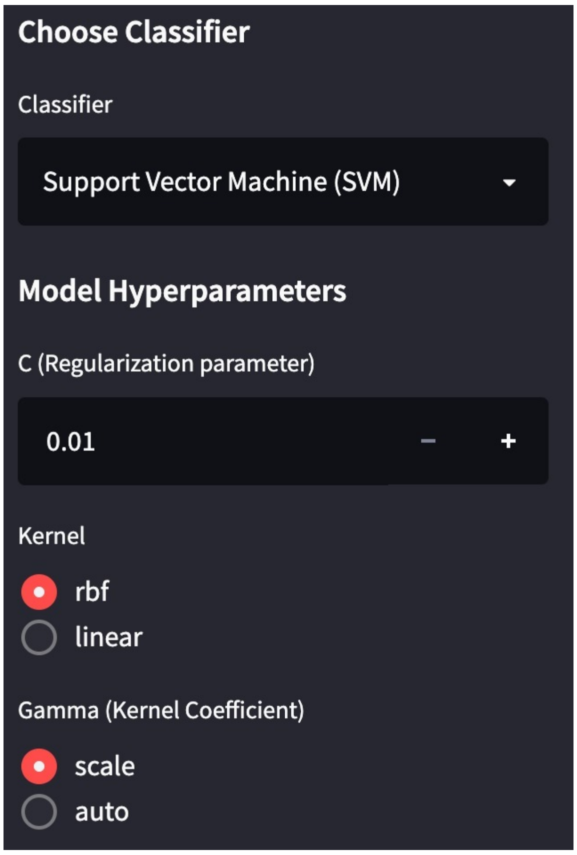

Binary Classification Web App

Are your mushrooms edible or poisonous? 🍄

Choose Classifier

Classifier: Support Vector Machine (SVM)

If SVM is selected: Figure: 10.4


Choose Classifier

Classifier: Support Vector Machine (SVM)

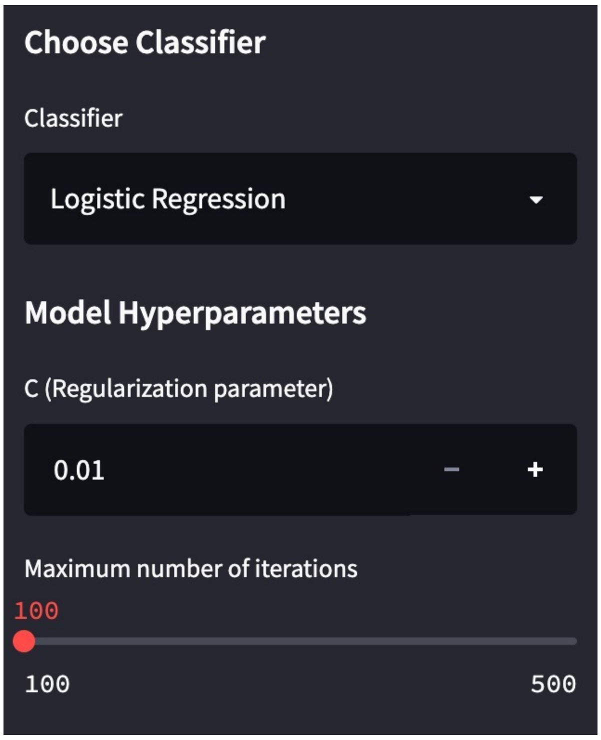
Model Hyperparameters

C (Regularization parameter): 0.01

Kernel: [✓ rbf] linear

Gamma (Kernel Coefficient): [✓ scale] auto

If Logistic Regression is selected: Figure: 10.5



Choose Classifier

Classifier

Logistic Regression

Model Hyperparameters

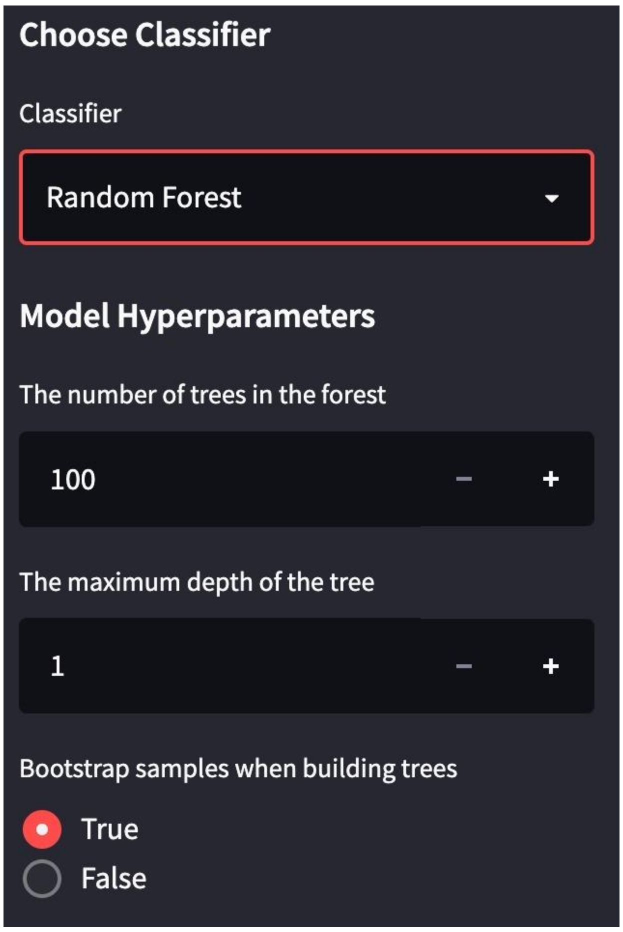
C (Regularization parameter)

0.01

Maximum number of iterations

100

If Random Forest is selected: Figure: 10.6



Choose Classifier

Classifier

Random Forest

Model Hyperparameters

The number of trees in the forest

100

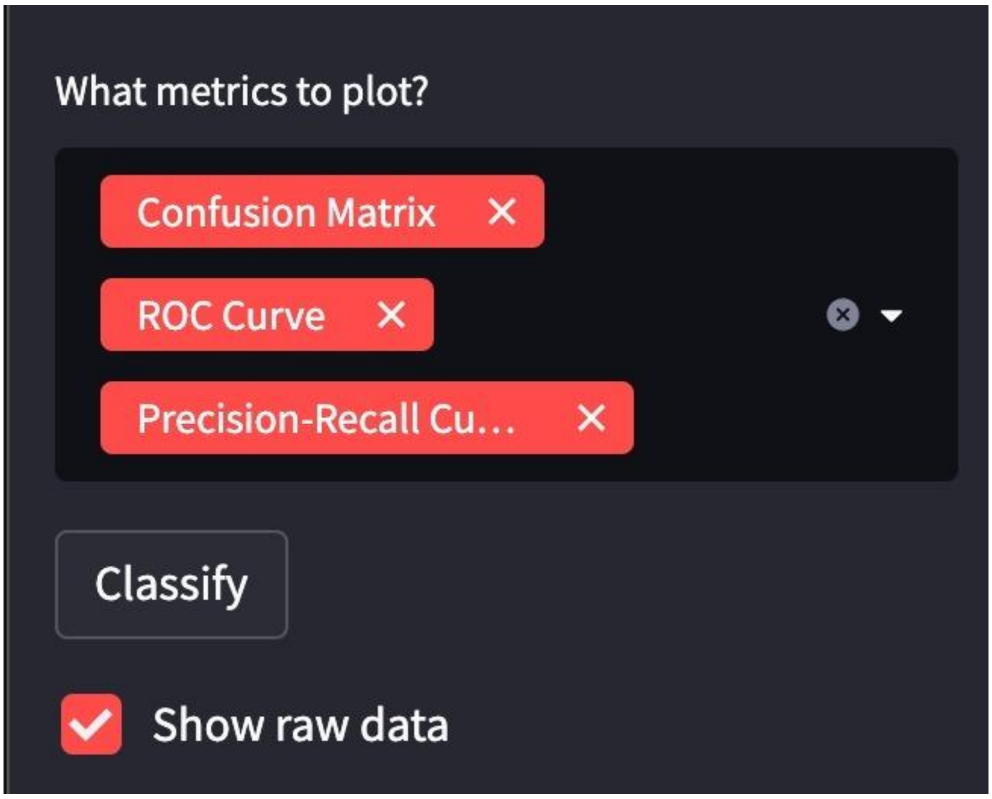
The maximum depth of the tree

1

Bootstrap samples when building trees

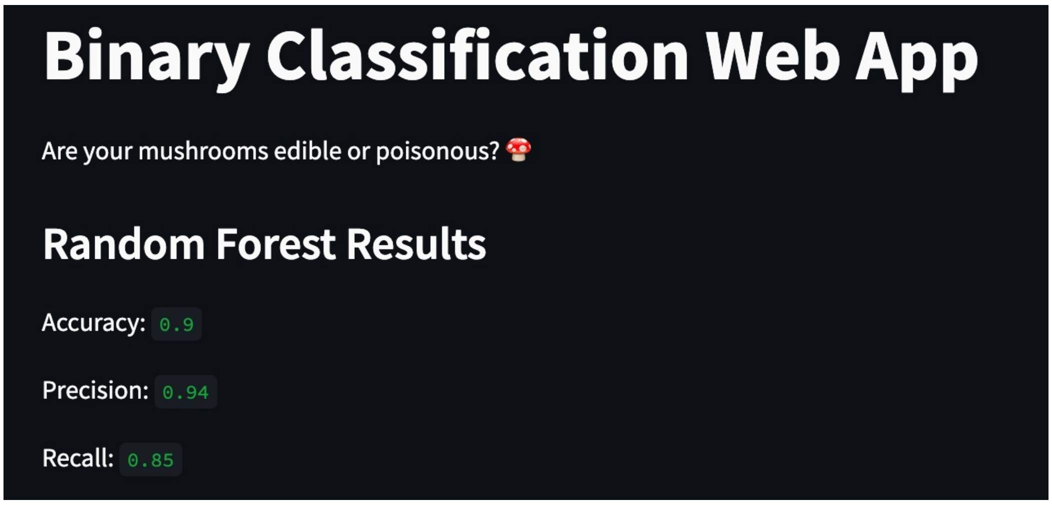
[✓ True] False

UI for Choosing Metrics: Figure: 10.7



10.8 Result Screens:

For Random Forest: Figure: 10.8.1



Binary Classification Web App

Are your mushrooms edible or poisonous? 🍄

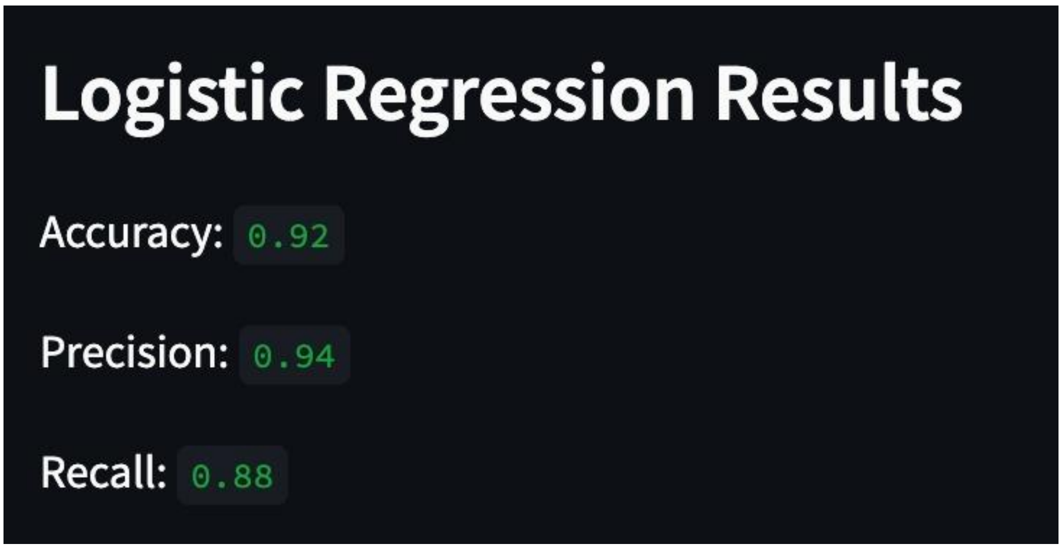
Random Forest Results

Accuracy: `0.9`

Precision: `0.94`

Recall: `0.85`

For Logistic Regression: Figure: 10.8.2



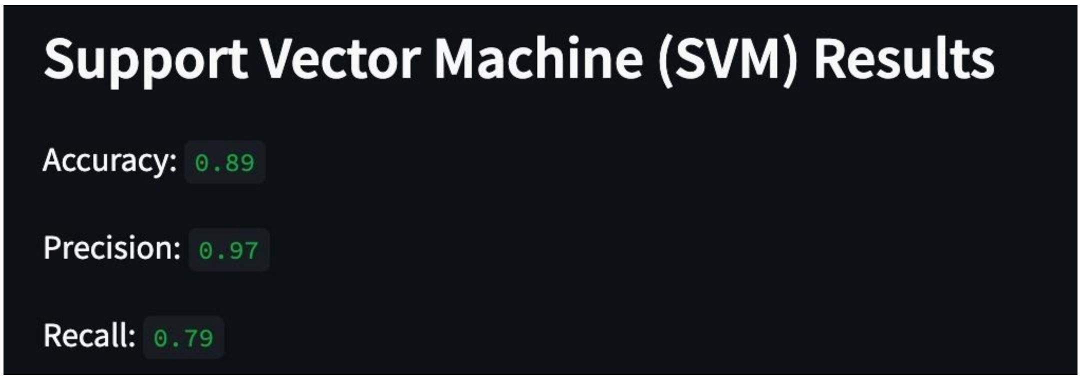
Logistic Regression Results

Accuracy: `0.92`

Precision: `0.94`

Recall: `0.88`

For Support Vector Machine: Figure: 10.8.3



Support Vector Machine (SVM) Results

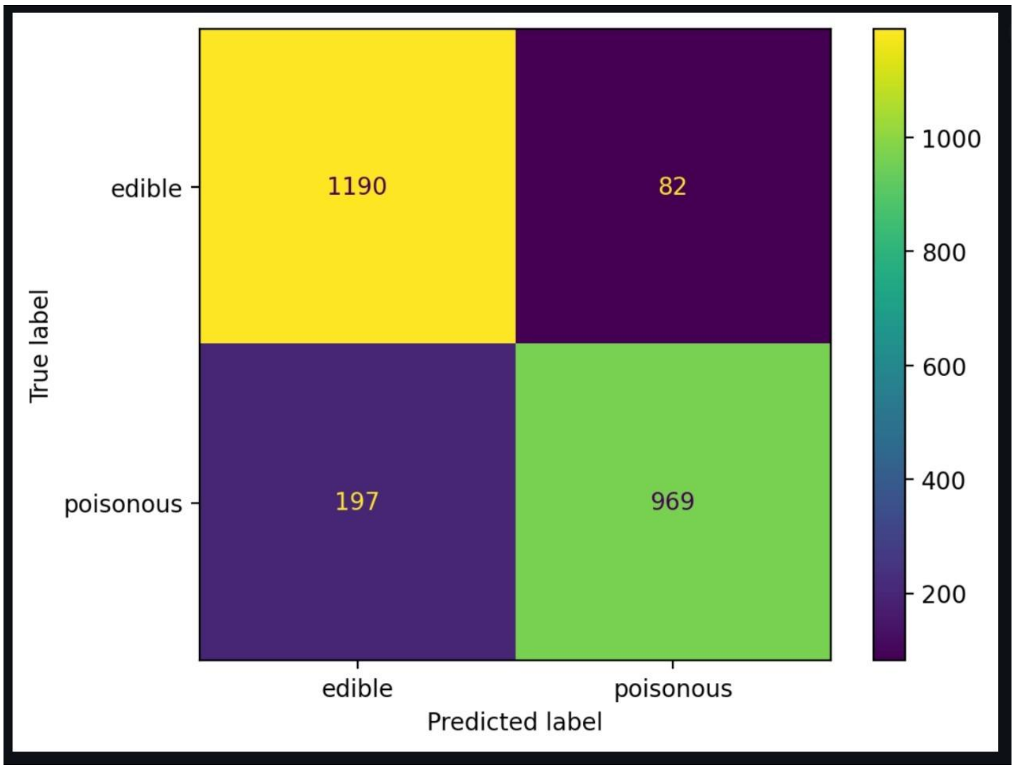
Accuracy: `0.89`

Precision: `0.97`

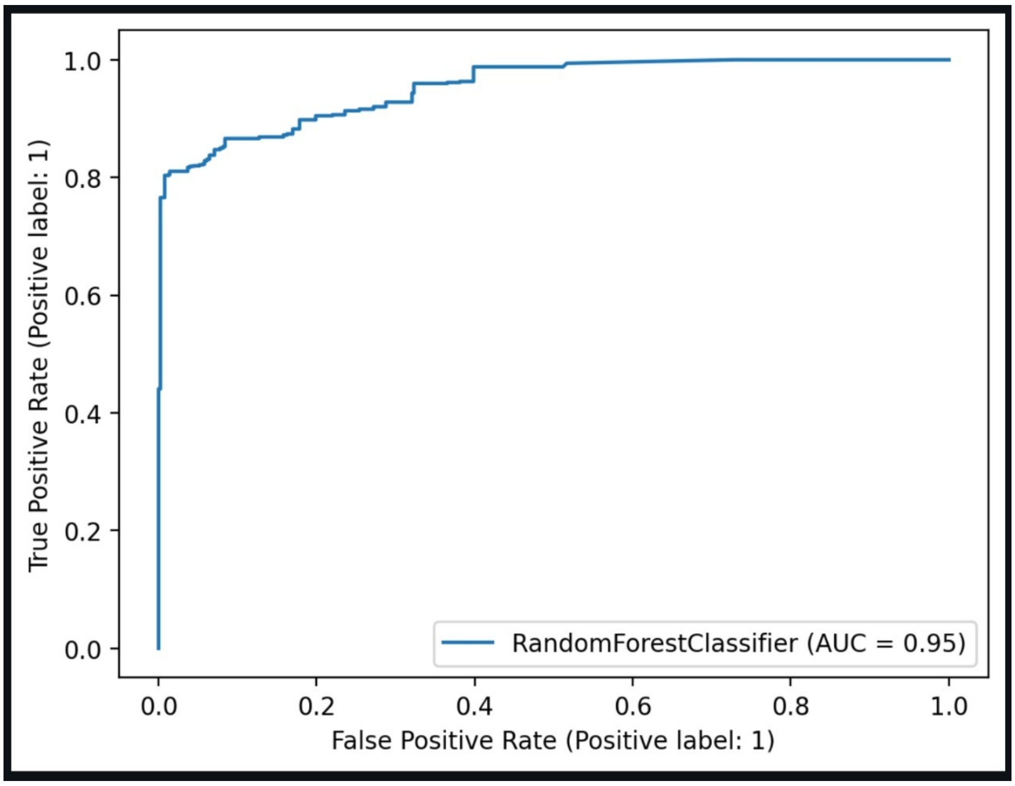
Recall: `0.79`

10.9 Metrics

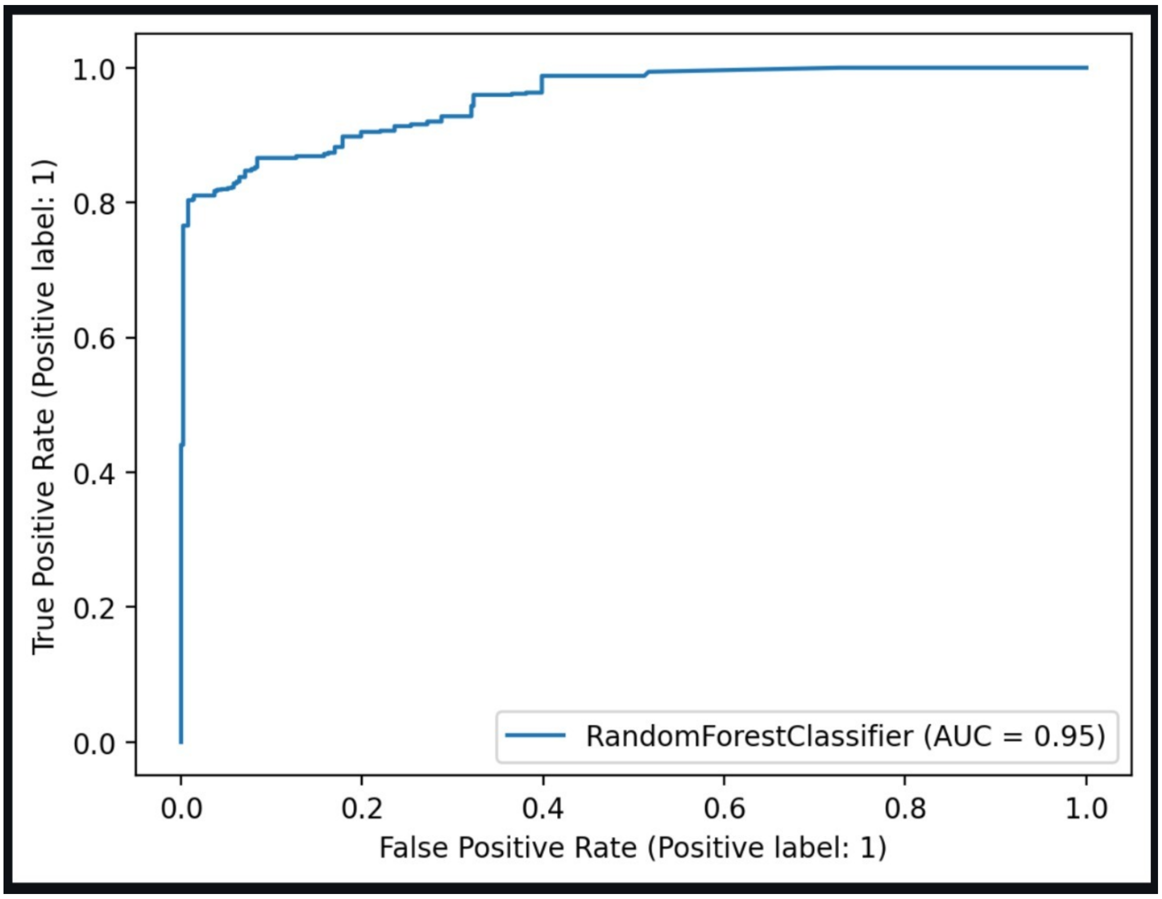
Confusion Matrix: Figure: 10.9.1



ROC Curve: Figure: 10.9.2

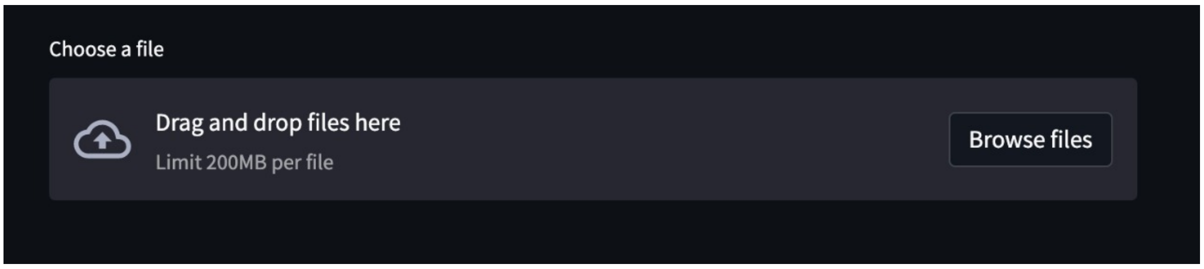


Precision Recall Curve: Figure: 10.9.3

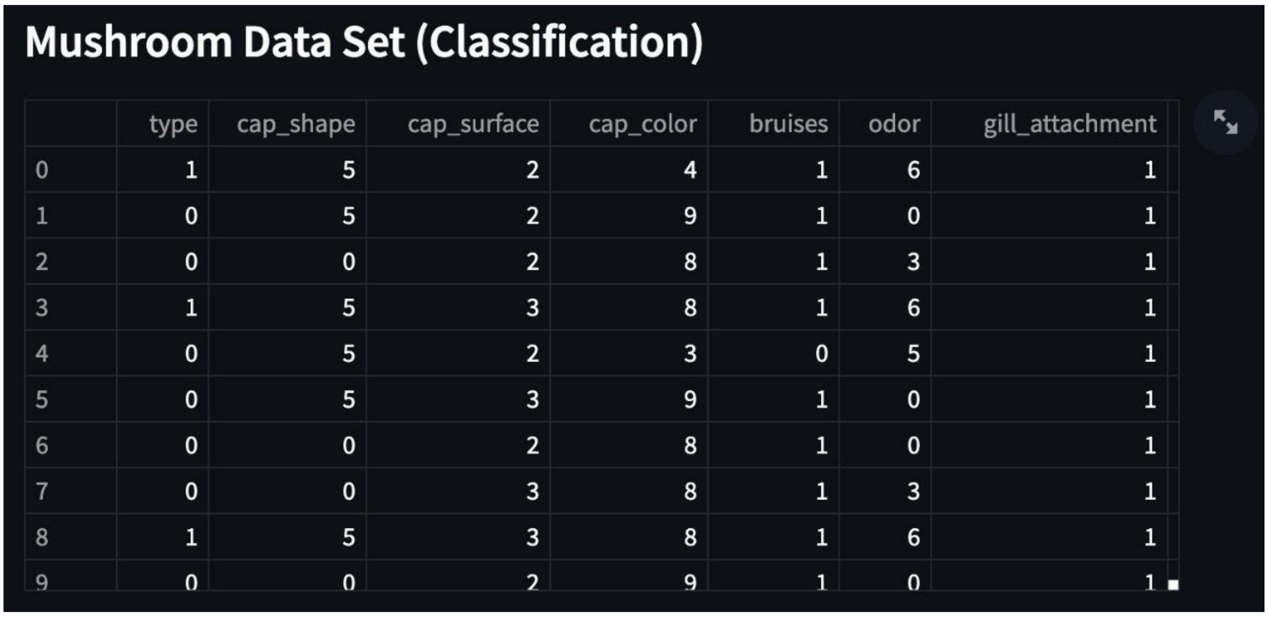


10.10 EDA Report Generation:

Upload Dataset: Figure: 10.10.1



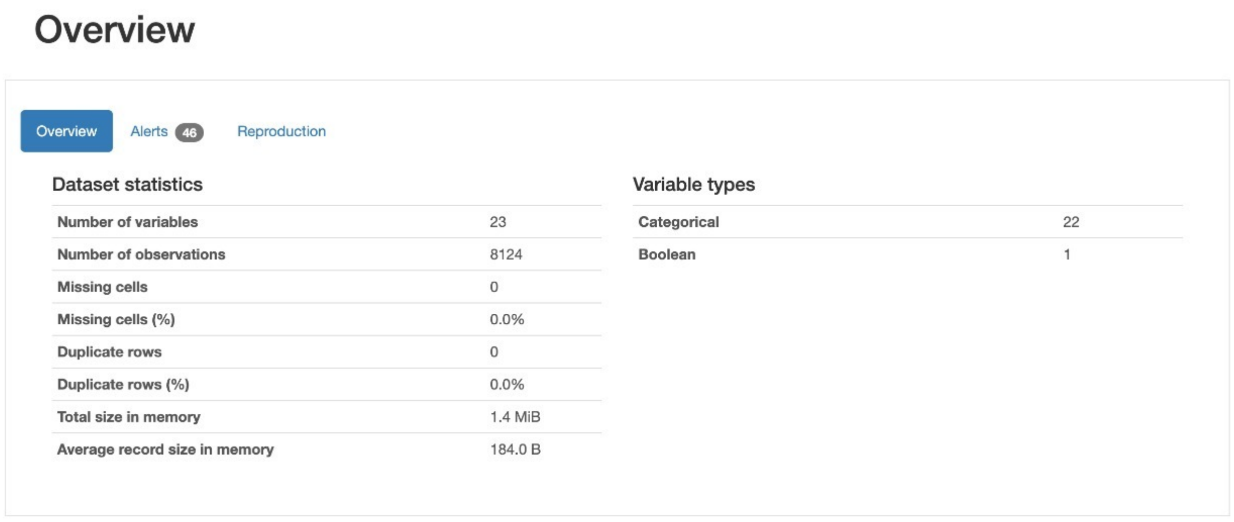
Show Dataset: Figure: 10.10.2



Mushroom Data Set (Classification)

	type	cap_shape	cap_surface	cap_color	bruises	odor	gill_attachment
0	1	5	2	4	1	6	1
1	0	5	2	9	1	0	1
2	0	0	2	8	1	3	1
3	1	5	3	8	1	6	1
4	0	5	2	3	0	5	1
5	0	5	3	9	1	0	1
6	0	0	2	8	1	0	1
7	0	0	3	8	1	3	1
8	1	5	3	8	1	6	1
9	0	0	2	9	1	0	1

Overview: Figure: 10.10.3



Overview

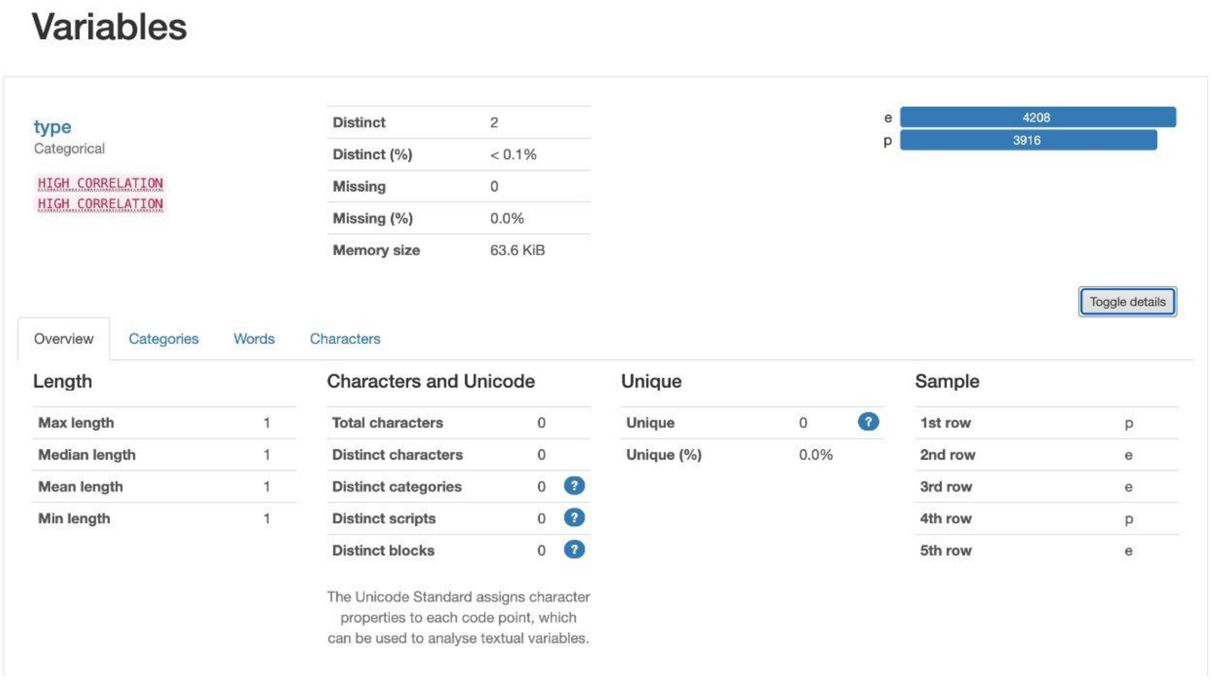
Dataset statistics

Number of variables	23
Number of observations	8124
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	1.4 MiB
Average record size in memory	184.0 B

Variable types

Categorical	22
Boolean	1

Variables: Figure: 10.10.4



Variables

type
Categorical

HIGH CORRELATION
HIGH CORRELATION

Distinct	2
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	63.6 KiB

e	4208
p	3916

Toggle details

Overview | Categories | Words | Characters

Length

Max length	1
Median length	1
Mean length	1
Min length	1

Characters and Unicode

Total characters	0
Distinct characters	0
Distinct categories	0
Distinct scripts	0
Distinct blocks	0

The Unicode Standard assigns character properties to each code point, which can be used to analyse textual variables.

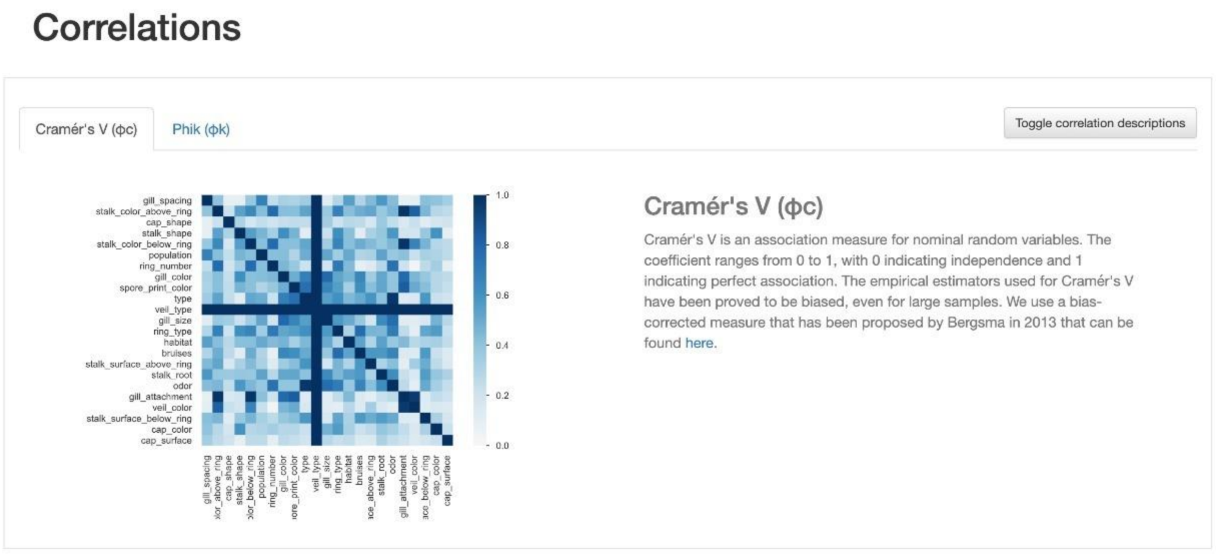
Unique

Unique	0
Unique (%)	0.0%

Sample

1st row	p
2nd row	e
3rd row	e
4th row	p
5th row	e

Correlations: Figure: 10.10.5



Correlations

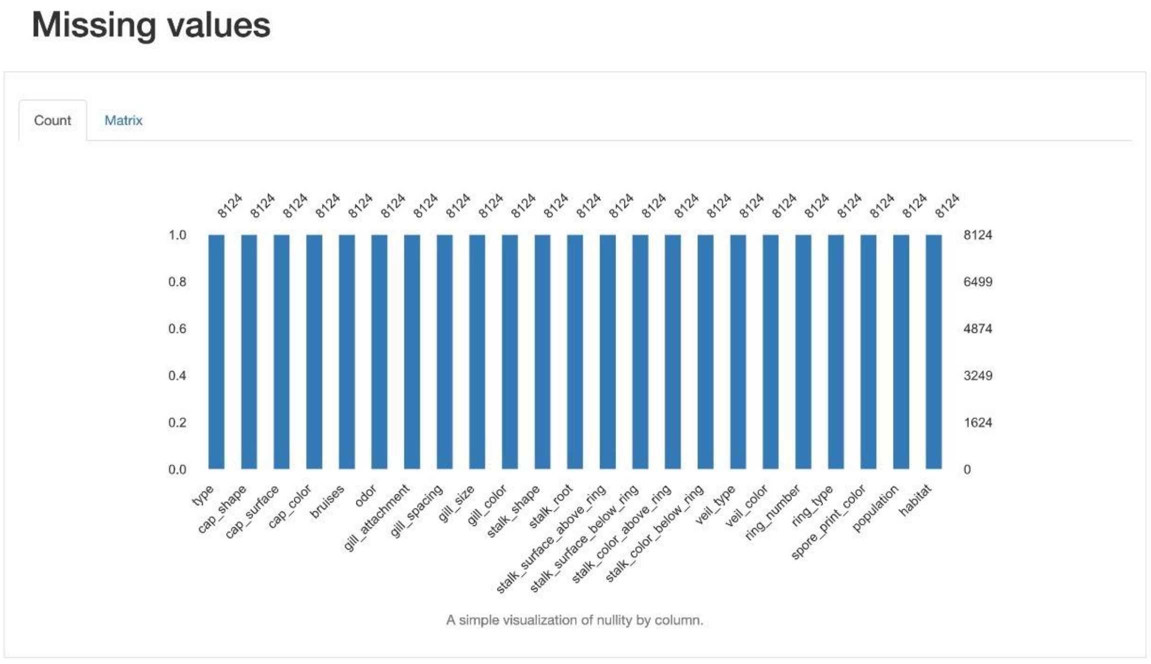
Cramér's V (φc) | Phik (φk)

Toggle correlation descriptions

Cramér's V (φc)

Cramér's V is an association measure for nominal random variables. The coefficient ranges from 0 to 1, with 0 indicating independence and 1 indicating perfect association. The empirical estimators used for Cramér's V have been proved to be biased, even for large samples. We use a bias-corrected measure that has been proposed by Bergsma in 2013 that can be found here.

Missing Values: Figure: 10.10.6



A simple visualization of nullity by column.

11 CONCLUSION

Veasualizer is a data visualization and ML tool and Using Veasualizer a user with zero knowledge can make beautiful visualizations: Graphs and plots. Not just that, Veasualizer has features using which users can Train, Test and Draw predictions from data by just uploading their dataset to Veasualizer.

The whole idea of this project is based on its user base, that is people who are going to use it. Our main objective of Veasualize is to make stuff, which is complex, look easy and useful to everybody. Just by some clicks and uploading datasets, Users will be able to plot different plots like: Scatter plot, Bar graphs, Bubble plots, Box plots, Dot plots, Pie charts and more and ML algorithms include Regression and Classification algorithms like: Linear, Polynomial, Logistic etc

The Exploratory Data Analysis report (EDA) is what makes this project unique, EDA includes Overview of the dataset uploaded, Correlation between variables, Missing values in the dataset and Statistical description of each variable.

12 REFERENCES

- Data Visualizations: A Literature Review and Opportunities for Technical and Professional Communication https://tek-ritr.com/wp-content/uploads/2017/09/data_visualization.pdf
- Systematic literature review of machine learning techniques https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-021-01403-2
- https://www.w3schools.com/python/pandas/pandas_intro.asp
- https://www.hindawi.com/journals/scn/2020/8830683/
- https://powerbi.microsoft.com/en-us/why-power-bi/
- https://github.com/streamlit/streamlit
- https://www.analyticsvidhya.com/blog/2021/06/build-web-app-instantly-for-machine-learning-using-streamlit/
- https://towardsdatascience.com/streamlit-hands-on-from-zero-to-your-first-awesome-web-app-2c28f9f4e214
- https://towardsdatascience.com/streamlit-hands-on-from-zero-to-your-first-awesome-web-app-2c28f9f4e214
- https://itnext.io/streamlit-kills-flask-1773c33fdc88
- https://blog.streamlit.io/snowflake-to-acquire-streamlit/