# AUTOMATING DATA SCIENCE

***For Domain Experts***

FINAL YEAR PROJECT REPORT

***Submitted by***


| | |
|---|---|
| BL.EN.U4CSE16014 | ARVIND SUDHEER |
| BL.EN.U4CSE16106 | RAMSHANKAR YADHUNATH |
| BL.EN.U4CSE16112 | SRIVENKATA SRIKANTH |


***in partial fulfillment for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

IN

COMPUTER SCIENCE AND ENGINEERING



AMRITA SCHOOL OF ENGINEERING, BANGALORE

AMRITA VISHWA VIDYAPEETHAM

**BANGALORE 560 035**


JUNE-2020

**AMRITA VISHWA VIDYAPEETHAM**

# AMRITA SCHOOL OF ENGINEERING, BANGALORE, 560035


## BONAFIDE CERTIFICATE

This is to certify that the project report entitled **"AUTOMATING DATA SCIENCE"** submitted by

| | |
|---|---|
| ARVIND SUDHEER | BL.EN.U4CSE16014 |
| RAMSHANKAR YADHUNATH | BL.EN.U4CSE16106 |
| SRIVENKATA SRIKANTH | BL.EN.U4CSE16126 |

in partial fulfillment of the requirements for the award of the **Degree Bachelor of Technology** in **COMPUTER SCIENCE ENGINEERING** is a bonafide record of the work carried out under my(our) guidance and supervision at Amrita School of Engineering, Bangalore.

Jyotsna C
Assistant Professor
Computer Science and Engineering

Dr. Amudha J
Chairperson
Computer Science and Engineering

This project report was evaluated by us on ……...(Date)

EXAMINER 1 EXAMINER 2

# ACKNOWLEDGEMENT

Our humble pranams at the holy feet of **AMMA** who has been a driving force and a source of inspiration for all educational activities and extracurricular activities at Amrita School of Engineering,Bengaluru(ASE-B).

We would like to express our sincere gratitude to our Director **Vishwamrita Chaitanya Swamiji** for the successful completion of the project. We would like to extend our sincere thanks to **Dr. Nagaraja S. R, Principal**. We are highly indebted to **Dr. Amudha J., Chairperson,** Department of Computer Science and Engineering, for motivating us towards the successful completion of the project.

We would like to thank our mentor, **Mrs Jytosna C** , whose consistent guidance and motivation has helped us throughout this amazing learning experience. She has indeed played a crucial role in our college academic life and filled it with positivity and optimism and we are indebted to her for this and are hoping that her best wishes will support us and push each one of us to greater heights in our career.

We also extend our gratitude to students from ASE-B who volunteered to test our final tool and special thanks to Goku Mohandas, from Made with ML who reviewed our tool and provided some very important feedback.

We would also like to thank our **evaluation panel** for providing us with ideas to improve our project and helping us to keep the project on track. Finally, we would like to thank the Department of **Computer Science and Engineering,** Amrita School of Engineering, Bengaluru for the constant support.

# ABSTRACT

Data Science has transitioned itself into a ubiquitous discipline in the last decade or so and has been a focal point of discussion in several business organizations as well as research-oriented academic institutions. Data Science can also be credited with being the most far-reaching sub-field of Computer Science and Engineering, given its impact in areas such as the social sciences that are not necessarily connected to the computing discipline. However, there is a significant dearth of technically hardened data scientists across the world and especially in fields that are not directly related to CSE. Many of these practitioners might have preliminary knowledge in dealing with data, but many a time, such rudimentary understanding might not help them in their task. Also, data cleaning is a very important as well as time-consuming process even for seasoned data practitioners and has been widely recognized as a process where human intervention can't be replaced with automation. In this project, we attempt to fix these issues by proposing and implementing a framework that would help automate core data science operations of cleaning, visualizations, pre-processing, feature selection, modelling and reporting.

# TABLE OF CONTENTS


ACKNOWLEDGMENT i

ABSTRACT ii

TABLE OF CONTENTS iii

LIST OF FIGURES vi

1. INTRODUCTION 1
    1.1. Data Science 1
    1.2. Data Science as a Multifarious Discipline 2
    1.3. What are the existing challenges in Data Science 3
        1.3.1. Lack of Technical Expertise 4
        1.3.2. Data Quality 4
        1.3.3. Delay in generating Preliminary Results 4
    1.4. Motivation 5
        1.4.1. Beginners 5
        1.4.2. Non Data-Technicals 6
        1.4.3. Data-Technicals 6
    1.5. Problem Statement 6
    1.6. Project Limitations 7

2. LITERATURE SURVEY 9
    2.1. Data Cleaning 9
    2.2. Data Preprocessing 10
    2.3. Data Visualization 10
    2.4. Automated Machine Learning 11
    2.5. What do these words lack in 11

3. REQUIREMENT SPECIFICATION 13
    3.1. Intended Audience 13
    3.2. Product Scope 13
    3.3. Overall Description 14
        3.3.1. Product Perspective 14
        3.3.2. Product Functions 14
        3.3.3. User Classes 15
        3.3.4. Operating Environment 15
        3.3.5. Design and Implementation Constraints 16

3.3.6. User Documentation 16
3.3.7. Assumptions and Dependencies 16

4. PROJECT THEORY 18
4.1. General Terms 18
4.2. The importance of domain knowledge in Data Science 19
4.3. Data Cleaning 20
4.3.1. The difference between data cleaning and preprocessing 22
4.3.2. Common data cleaning steps in tabular data 23
4.4. Introducing a novel metric to assess the quality of tabular data 26
4.4.1. Important components in TDCS computation 26
4.4.2. Formula 27
4.4.3. Some more specifics about TDCS 27
4.5. Data Preprocessing 28
4.5.1. Type of features in a dataset 28
4.5.2. Preprocessing techniques used 30
4.6. Data Visualization 35
4.7. User-interaction Based Systems 35
4.8. Deep Learning Automation 37
4.8.1. Supervised Learning 38
4.8.2. Important parameters 38
4.8.3. Choosing the number of epochs 39
4.8.4. Regularization 40

5. PROPOSED SYSTEM 41
5.1. Project Goals 41
5.2. Key Considerations 42
5.3. User Stories 42
5.4. System Block Diagram 43
5.4.1. Explanation of Data Flow through Architecture 43
5.4.2. Fundamental building blocks of the system 44
5.5. Dissecting DataSwissKnife 46
5.5.1. Create Structure in Host's System 46
5.5.2. Load Data 48
5.5.3. Clean Data 49
5.5.4. Preprocess Data 53
5.5.5. Visualize Data 55
5.5.6. Model Data 56
5.6. The devices algorithm for deep learning automation 57

5.6.1. Stepwise elucidation of the algorithm 57
5.6.2. Algorithmic implementation details 58
5.6.3. The neural network architectures used in the system 60

6. IMPLEMENTATION 64
6.1. Flow of Control 64
6.2. DataSwissKnife is built with OOP paradigms 65
6.3. Links to Repository 67

7. TESTING 68
7.1. Who the responders were 68
7.2. Did the respondents need a no-code tool for data science. 69
7.3. How likeable the idea was behind the tool. 70
7.4. How easy the tool was to use. 70
7.5. Time to use the tool 71
7.6.How significant do the respondents feel the tool will be in the future 71

8. CONCLUSION ,KEY TAKEAWAYS AND FUTURE SCOPE 73
8.1. Conclusion and Key Takeaways 73
8.2. Future Scope 74

9. REFERENCES 75

# LIST OF FIGURES

**Fig (1.1)** High level steps in Data Science 1

**Fig (1.2)** Google Trends data for "Data science" for the past decade. 3

**Fig (1.3)** Classification of Data Science Consumers 5

**Fig (3.1)** User Classes 15

**Fig (4.1)** Instances and Features in a Dataset 18

**Fig (4.2)** Dependence of Data Science on Domain Knowledge 19

**Fig (4.3)** Inconsistency Errors in data lead to bad insights 20

**Fig (4.4)** Correct bar plot 21

**Fig (4.5)** Lighter values signify "missing values" 21

**Fig (4.6)** Difference between data cleaning and data preprocessing 23

**Fig (4.7)** Dealing with Missing Values - A Block Diagram 26

**Fig (4.8)** Types of features in a tabular dataset 30

**Fig (4.9)** Effect of outlier removal 31

**Fig (4.10)** Standardization and Normalization 32

**Fig (4.11)** Label encoding 33

**Fig (4.12)** One-hot Encoding 34

**Fig (4.13)** Interval Encoding 35

**Fig (4.14)** Basic structure of User-Interaction based systems 36

**Fig (4.15)** Epoch-wise accuracy line graph for the titanic dataset 39

**Fig (5.1)** User Stories 42

**Fig (5.2)** System High-Level Architecture 43

**Fig (5.3)** 8 Building Blocks of the System 45

**Fig (5.4)** Project Structure in the Host's System 47

**Fig (5.5)** Choosing the location to place the project directory 48

**Fig (5.6)** Naming the project directory 48

**Fig (5.7)** Choosing the dataset from the local system 49

**Fig (5.8)** Output after loading and renaming the dataset; a preview is generated 49

**Fig (5.9)** The initiation of Data Cleaning 50

**Fig (5.10)** Missing value percentages in the data 50

**Fig (5.11)** Missing value percentages in the data 51

**Fig (5.12)** Performing Imputation to fill in the remaining missing values 51

**Fig (5.13)** Datatype consistency management 52

**Fig (5.14)** Prompts to enter the target 53

**Fig (5.15)** Interact with the user to choose feature types 54

**Fig (5.16)** Other preprocessing steps with the help of user input 55

**Fig (5.17)** Data Visualization 55

**Fig (5.18)** Auto-generated plots 56

**Fig (5.19)** Modelling Results 57

**Fig (5.20)** The devised deep learning automation algorithm 58

**Fig (5.21)** Logistic Regression model architecture 61

**Fig (5.22)** Shallow neural network architecture 62

**Fig (5.23)** Deep neural network architecture 63

**Fig (6.1)** Flow of control through DSK 64

**Fig (6.2)** Class Hierarchy in clean_data.py 66

**Fig (6.3)** Class Hierarchy in preprocess_data.py 66

**Fig (6.4)** Class Hierarchy in visualize_data.py 67

**Fig(7.1)** Number of students and working professionals 68

**Fig(7.2)** Exposure to data science on a scale of 1-10 69

**Fig(7.3)** Did users want a tool to perform preliminary operations? 69

**Fig(7.4)** How much users liked the idea behind the tool 70

**Fig(7.5)** How easy it was for the users to use the tool on a scale on 1-5. 71

**Fig(7.6)** Time taken to run the tool on the user's system 71

**Fig(7.7)** Significance in the future on a scale of 1-10 72

# 1. INTRODUCTION

## 1.1 Data Science

The consistent surge in data available with organizations in the contemporary world has been a major talking point over the past few years. This surge has led to a phenomenon where the need for generating valuable insights of data has far exceeded the ability of organizations to do so manually [1]. Such a phenomenon has engendered the emergence of a new discipline called Data Science. While there are various definitions of data science doing rounds in popular media, blog articles and academic program websites, the main underlying theme states that "Data Science is a study of the generalizable extraction of knowledge from data" [2]. However, a more comprehensive definition of the field suggests that data science is a concept to unify statistics, data analysis and machine learning among other related methods in order to understand actual phenomena [3].

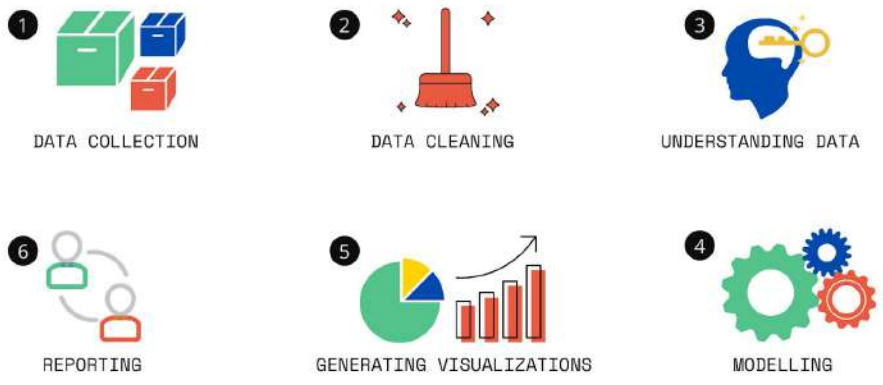



**Fig (1.1)** High level steps in Data Science. Inspired from [4]

Fig(1.1) provides a high-level graphical representation of the data science process. The process comprises of data collection(the task of aggregating data from multiple sources), data cleaning(the task of converting raw data into a form suitable for analysis), understanding data(the task of interpreting data sources and incorporating domain-level knowledge), modelling(the task of using data mining to build predictive models), generating visualizations(the task of summing up raw data in graphs for easier comprehension) and reporting(the task of projecting and describing finds to potential stakeholders).

## 1.2 Data Science as a Multifarious Discipline

Data science can't be pinpointed as one definitive field of study [5]. Software developers, ML engineers, Analysts, Data scientists are all job titles that resemble the work done in this broad domain. However, a wide range of job titles is not the only point of variance in data science. Even the academic backgrounds of the people working in this discipline can range from the qualitative disciplines to quantitative subjects. The applications of data science too are not constrained to just data-driven business decisions, but find significant prominence in all kinds of fields ranging from the social sciences to advanced computing techniques. A recent wave has also brought about a rise in the number of social good use-cases of data science.

Data science has also become a tool of competition amongst organizations operating within the same industry. Data has become the new battleground and those that understand this and work towards creating a more data-literate environment thrive, while others perish. It is the very understanding of this change in the nature of organization-level competition that has helped companies like Netflix, Amazon.com, Harrah's Entertainment etc. emerge as winners in the past [6].

The academic space too has been responding strongly to the industry's growing demand for data scientists with more and more data science programs being created at universities [7]. As depicted in fig(1.2), the popularity of the term "Data science" has been constantly on-the-rise over the past decade. This increase in popularity of the field has also helped institutions(both private and public) to trust in the lucrative prospects of creating data science programs.

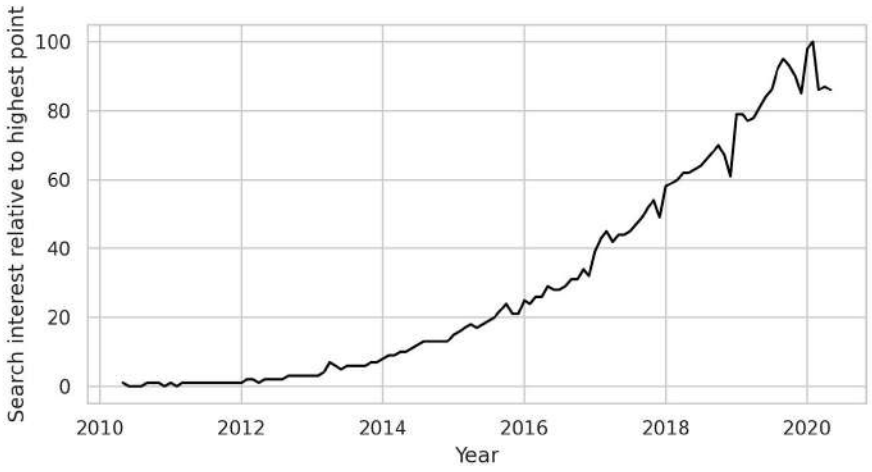



**Fig (1.2)** Google Trends data for "Data science" for the past decade. The y-axis represents the search interest that is relative to the highest point, which occurs in the year 2020.

## 1.3 What are the existing challenges in Data Science

While [8] discusses most of the challenges in depth, in this section, we attempt to describe three often overlooked cases when it comes to doing data science.

### 1.3.1 Lack of Technical Expertise

The use of mathematics, statistics and algorithms are essential needs for doing data science. This means that an individual who wishes to work with data will have to first

be proficient in these skills. Unfortunately, this might mean a delay in producing preliminary reports and basic predictive models.

### 1.3.2 Data Quality

Poor quality of data is a major concern in the data ecosystem. Works such as [9] produce effective evaluation frameworks for assessing data quality. However, such metrics and frameworks will not help if data with compromised quality is handed over to the teams analysing the same. Rectifying such data issues is called data cleaning and it is widely said that the average data scientist spends upto 80% of his time cleaning data [10]. Data cleaning often requires individuals to write code explicitly to perform cleaning operations. The need for writing code to clean data points back to the challenge of lack of technical expertise if it is to be performed by an individual without the necessary computational background.

### 1.3.3 Delay in generating Preliminary Results

It takes time to learn and apply data science concepts. This time space might be further extended if individuals lack the technical expertise to readily analyse data. The problem with this extended time space is that it delays the ability of a person to see how impactful data science can be for his or her work. Such a delay coupled with the steep learning curve involved(as depicted in the challenges above), will act as a deadly inhibition for smaller businesses and industry players to use data science in their tasks.

The delay can also be rather disturbing for data science beginners as well as experienced data practitioners. While beginners might find it difficult to write code

that performs cleaning operations, those with experience will find it rather cumbersome to clean up data every time.

## 1.4 Motivation

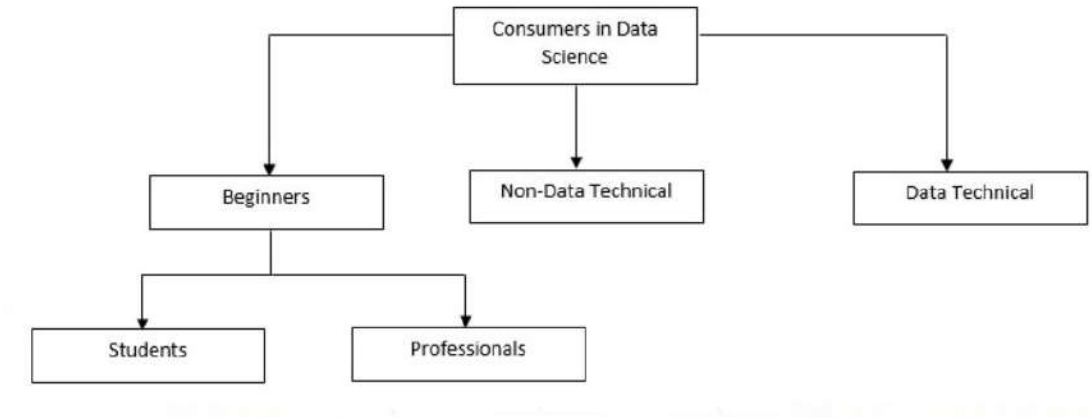



**Fig (1.3)** Classification of Data Science Consumers

In fig(1.3), we attempt to make a classification of the consumers of data science. The classification is as follows:

### 1.4.1 Beginners

**Who:** Individuals who are beginners in data science and are only interested in learning. They are not looking to solve a problem yet. Can be further classified into students and professionals looking to make a career transition.

**Main Problem:** They need to know too much of theory before being able to start real analysis or develop models.

### 1.4.2 Non-Data-technicals

**Who:** Individuals who do not have a background in technical knowledge for data science, but are eagerly looking forward to using it for improving their businesses or for other purposes. They have the necessary domain knowledge they require.
**Main Problem:** They do not have the expertise to write code and neither do they have a lot of time to spare for learning how to write code.

### 1.4.3 Data-technicals

**Who:** Individuals with complete in-depth understanding of data science. They might be working professionals or working in academia. They know how to write code and are technically sound.
**Main Problem:** Data cleaning, generating preliminary baseline models, creating basic reports etc. becomes time-consuming and rather cumbersome. They will appreciate it if provided with a mechanism to automate these tasks(if not completely, at least partially).

Based on the challenges specified in 1.3 and the classification made above, the onus is on the creation of a mechanism that will help beginners and data-non-technicals analyse their data faster and build their preliminary models quicker. There is also a need to reduce the effort needed for an expert to clean raw data.

## 1.5 Problem Statement

“To build software that will aid anybody who is familiar with necessary domain knowledge, do preliminary data science”. The gravity of the problem statement at

hand calls for a new approach to data science. If data science has to be made simpler and easier for quickly generating essential results(without having to write code), full or partial control has to be transferred from the hands of the user to the system itself. In other words, the system has to be automated. Therefore, in this project we have made an attempt at setting the foundations for a system that will work in automated fashion to help serve the problem statement.

While automated machine learning has seen some work in the past, automating data science is a new idea and since it is so, the extent of prior work performed in this field is minimal. Hence, much of this project's theory will be based on fresh principles backed by past examples. The software that has been developed is prototypical and will be scaled to a full product in the future.

## 1.6 Project Limitations

A couple of limitations the project suffers from currently:

1. Lack of prior groundwork and research
2. Currently works only for tabular data
3. Modelling only considers supervised classification problems
4. Automation is not a silver bullet and can face the following issues:
    a. Users might mutate/change their data into formats that make it less suitable for analysis if they do not understand and apply their domain-level knowledge well
    b. Final performance(in terms for evaluation metrics) might not be as high as the output of a human data scientist

c. Not all data can be cleaned and processed in an “objective” sense, some require the subjectivity of human thought

# 2. LITERATURE SURVEY

The amount of prior work performed in this project's domain is minimal and hence, publications that directly relate to the domain are little. However, the project's theory is built on several concepts of data science derived from the literature discussed in this section.

## 2.1 Data Cleaning

The lack of good quality data is a major pain point in data analysis. Dasu et. al [10] describe how 80% of an average data scientist's time on a data science project is spent on data cleaning and processing. Research such as the one conducted by Strong et. al [11], suggests that data quality metrics find meaning only when they are able to gauge how good data is at the customer's end. This is a very strong point that places the responsibility of data quality on those responsible for data generation and maintenance. However, once the data has been passed further down the organization pipeline to the data analysts, such metrics see their abilities being limited. In the case of understanding the common problems associated with data, Rahm et. al [12] created a broad classification that divides data-related problems into schema-level and instance-level(data entry issues). Tierney et. al [13] in their work describe the principles of dealing with missing values which are interesting components of unclean datasets. Their work is in fact inspired from the work of Wickham, Hadley [14] in which the author provides an in-depth discussion on the principles of tidy data, common issues with data and methods to manipulate messy data into its tidy form.

## 2.2 Data Preprocessing

Data preprocessing [15] is an important step in data science and often also includes data cleaning. Ramirez-Gallego et. al [16] have described data preprocessing as a method to make raw data fit for further analysis. Famili et. al [17] states that the two fundamental objectives for using data preprocessing techniques is to solve data-related issues and to also learn more about the nature of data. The authors also classify preprocessing methods into data transformation methods, information gathering methods and generation of new information methods. Another classification of preprocessing methods in Garcia et. al [18] divides data preprocessing techniques into methods for dealing with imperfect data, imbalanced data preprocessing and data reduction techniques. The work in [16] also provides an outlook of different data reduction techniques such as dimensionality reduction, instance reduction and feature space simplification.

## 2.3 Data Visualization

The ability to convey important information with the help of graphs and charts have been an indispensable part of data science since the beginning [19]. Cleveland, William S [20] argues the potential for quantities to go unnoticed if appropriate visualization techniques are not applied. He also states how good visualizations can help understand the data at hand in better sense. Tukey, John W. [21] was amongst the first formal publications on data analysis in 1977 and advocated the importance of using Exploratory Data Analysis as a part of the conventional analysis procedure to understand data better and find hidden patterns within it. Aparicio et. al [22] also suggests that data visualization is less of an independent field and more of a coalition

between several fields such as psychology, computer science and multimedia designing; where each plays its own part in creating a visually appealing and statistically relevant visualization. Segel et. al [23] projects the importance of using narrative visualization as a means of storytelling with data.

## 2.3 Automated Machine Learning

Automated Machine Learning(AutoML) is the automation of machine learning procedures applied to real-world use cases [24]. Thornton et. al [25] created the Auto-WEKA package in this work and laid out what was amongst the very first in-depth discussions on AutoML. The package was tested over 21 datasets and consisted of 27 base, 10 meta and 2 ensemble classifiers. It treated AutoML problems as CASH(Combined Algorithm Selection and Hyperparameter Optimization Problem). Feurer et. al [26] extended [25] to create Auto-Sklearn which apart from viewing AutoML as a CASH problem, also used meta-learning to predict the correct classifier for a new dataset based on what had worked in the past.

Additionally, James Max Kanter and Kalyan Veeramachaneni developed a data science machine that delivers predictive models for given raw data. They have devised an algorithm, termed as “Deep feature synthesis” that synthesises features for a given dataset for better predictive modelling [57]. However, this work is not keen on preprocessing the data thoroughly and has not incorporated deep learning.

## 2.4 What these works lack

The above research that has been discussed does not majorly include papers that lack detail. However, there is a dearth of research that focuses on combining these multiple aspects of data science with automation to create systems that will help

end-users do data science without having to necessarily cope with sophisticated details. Samulowitz et. al [27] through their research has provided finds closest to the idea of our project and their work coins a very specific term for automating data science with the help of user interaction - Cognitive automation of data science.

In this project, we strive to provide a more foundational theoretical basis for automating data science and democratize the field so that anybody would be able to begin working on it even if they lack the technical background to do so.

# 3. REQUIREMENT SPECIFICATION

This chapter will aim to provide clear and lucid explanations on the different requirements needed and the reasons behind why it is so. The requirement specification will be written with a user-centric outlook as the idea of the project is to develop a tool that will finally be used and evaluated by the data science community. The contents also specify resources that the team will need while developing the product.

## 3.1 Intended Audience

This chapter is primarily intended for the team that is developing the product i.e the students who are submitting this report. Secondary audience include the evaluation panel, students from the institution who would like to build on this project or contribute to it at a later stage and to anybody who wishes to be a part of the core development team of our product as a developer, tester or a documentation maintainer.

## 3.2 Product Scope

The product being developed will serve as an efficient, easy-to-use tool that can best help individuals with domain expertise, but lack coding skills to quickly perform preliminary analysis to their data. Through the product, quick data cleaning could be performed and baseline models can be created. Other objectives of the product also include providing a basic interface to data science enthusiasts of any skill level to quickly generate preliminary results with the data they possess. The product is not

limited to a closed-development environment, rather it will be built keeping in mind the possibilities of making it open source so that it could help the community in one way or the other.

## 3.3 Overall Description

### 3.3.1 Product Perspective

The product developed is a new, self-contained product based on ideas that have seen limited or almost no implementation in contemporary data science. However, the product is not meant to supplant to traditional data science processes. Instead, it is most efficient when used as a tool for preliminary data science operations.

### 3.3.2 Product Functions

The main functions of the product can be grouped into 3 main categories:

- Automated Data Preparation
    - Data Cleaning
    - Data Pre-processing
- Data Visualization and Modelling
    - Basic EDA
    - Preliminary Modelling
- Maintain Project
    - Create a hierarchical structure for users' data projects in their local system
    - Generate reports

### 3.3.3 User Classes

There are 3 main users for whom this software has been intended. Fig(3.1) describes one line user stories for each of these categories of user to summarise the different needs the product should cater to.


**Fig (3.1)** User Classes

### 3.3.4 Operating Environment

The product is intended to be platform independent and should be able to run on any operating system. However, the user's system needs to have both pip and python3 installed. The hardware requirements for the product will depend on the size of the dataset the user will be using the product on.

It is highly advised that users install and run the product within a python virtual environment in order to prevent unexpected version changes in the other libraries that are installed with the product. Further information on virtual environments can be found at https://realpython.com/python-virtual-environments-a-primer/

### 3.3.5 Design and Implementation Constraints

The main constraints surrounding the development of the product are:

1. Limited availability of time considering the size of the project
2. Lack of preceding ground to build the project on
3. Lack of the team's expertise in building complete end-to-end systems

However, the team assumes complete responsibility of developing and maintaining the product, including the responsibilities of understanding, analysing and fixing issues that might occur with the product in the future.

### 3.3.6 User Documentation

Documentation is the core of any open source project and the product developed shall also be provided with ample documentation. Documentation will be provided as a comprehensive README.md file on the github repository that will contain product scope, installation guidelines, usage examples and contact details of the creators. A complete usage manual will be created and will be hosted on Readthedocs.io for providing in-depth coverage on how to use the product.

### 3.3.7 Assumptions and Dependencies

The working of the tool built is subject to the following assumptions that have been made by the team:

1. The user has pip and python3(version $\geq$ 3.7) installed in their host systems
2. The user is aware of the risks posed if running th tools outside of a virtual environment, hence uses a virtual environment

3. The user knows basic command line interface commands

Though these assumptions are realistic and not too complex for a user to accomplish on his/her side, the team recognizes that the project needs to be as simple as possible for a user to be comfortable with it. Hence, the team will try to make the product very simple to use, at least within the first 2-3 versions of the product; if not the first.

All dependencies needed for running the product will be installed automatically via the requirements.txt file that keeps track of the required dependencies. The requirements.txt file is generated using the pipreqs python library.

# 4. PROJECT THEORY

The focus of this chapter is on discussing the theoretical foundations on which the proposed system has been built.

## 4.1 General Terms

1. **Data:** Characteristics or information, usually numerical, that are collected through observation [28].
2. **Dataset:** A collection of data [29]. In the scope of this project, a dataset is a tabular collection of data i.e data that can be represented in the form of tables.
3. **Feature:** An individual measurable property or characteristic of a phenomenon being observed [30].
4. **Instance:** A single object of the world from which a model will be learned, or on which a model will be used(i.e prediction) [31].
5. **Domain Knowledge**: The knowledge of a specific, specialized discipline or field, in contrast to general knowledge [32].

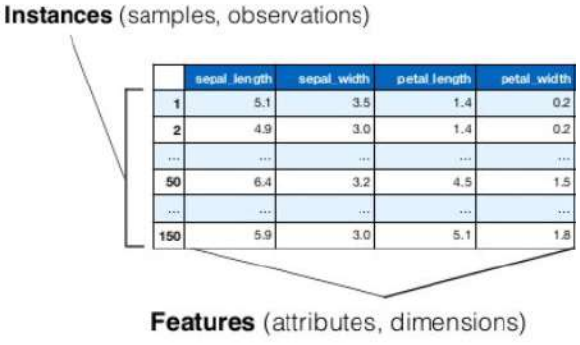



**Fig (4.1)** Instances and Features in a Dataset

## 4.2 The Importance of Domain Knowledge in Data Science

Data science is an interdisciplinary field [3] and also a field that can be applied to a wide range of use cases [6]. A few notable examples would be HR analytics, medical data science, product recommenders etc. While each of these use cases differ in terms of data, outputs, stakeholders, customer-base etc., they all are connected by the application of data science in them. The distinction between these use cases is brought about by the relevant domain knowledge that is associated with each of them. As in section 4.1, domain knowledge relates to very specific knowledge in a given discipline. A domain expert is one who is educated or experienced in the domain, understands the meaning of data generated in the domain(be informed about how the features are related, what values are legal, what factors indicate that the data is unclean etc.), can assess the quality of the data generated and can communicate effectively with final end users(who mostly might be unaware of all technical detail) [33].

As shown in fig(4.2), the amount of sophistication that can be achieved in the field of data science is directly dependent on the level of domain knowledge one has at their disposal.


**Fig (4.2)** Dependence of Data Science on Domain Knowledge [6]

Hence, having necessary domain knowledge can help conduct reasonably effective data science.

## 4.3 Data Cleaning

Raw data is often unclean. Here, unclean refers to a state of the data when a part of it or all of it does not find use in the subsequent analysis as these left out parts are not in a format that can aid unbiased, clear, trustworthy and robust analysis. Data cleaning, being amongst the first half of tasks performed(usually second) as per most data science methodologies including KDD and OSEMN, is a step that will provide better results when it is performed by one who understands the data well i.e knows what should be and should not be there in the data. Such a person is the domain expert (as in section 4.2). For example in fig(4.3.a), we have a subset of data from an animal shelter. Here, the same reptile species of the spectacled cobra has been entered with 3 separate spellings. If a bar plot was plotted here to identify the most occuring reptile species at the shelter, a plot that takes this unclean data into consideration will identify the king cobra as the most common reptile at the shelter.

| | common name | count |
|---|---|---|
| 0 | spectacled cobra | 2 |
| 1 | spectacle cobra | 4 |
| 2 | Spectacled cobra | 1 |
| 3 | python | 1 |
| 4 | king cobra | 5 |

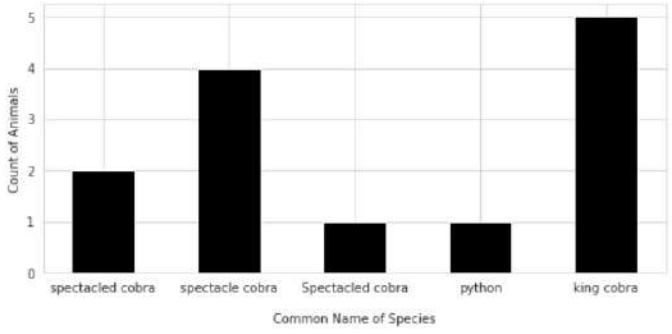



**Fig (4.3)** Inconsistency Errors in data lead to bad insights. a)Unclean data b)Bar plot generated from unclean data

However, a domain expert(this example is a simplification of the real world issue at animal shelters) will easily be able to understand that the first 3 entries represent the same reptile. Once this has been rectified and cleaned, the new bar plot will look like this.

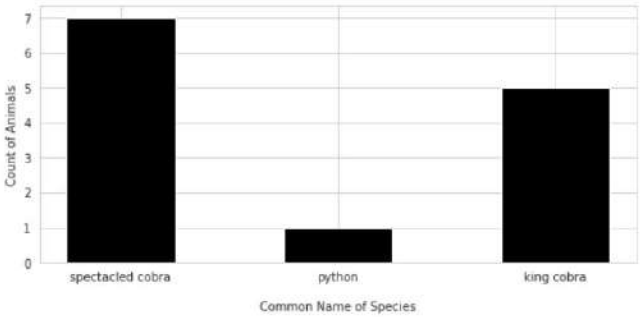



**Fig (4.4)** Correct bar plot

Fig(4.4) shows that the spectacled cobra is the shelter's most common species and not the king cobra.

Missing Values [13] pose another very important data cleaning problem. Missing values describe values that are not present in the dataset. The absence of these values might be for a variety of reasons, most of which can only be explained and understood by a domain expert. Fig (4.5) describes the missing value heatmap of the famous Titanic dataset [50].


**Fig (4.5)** Lighter values signify "missing values"

From fig(4.5), it can be noticed that the features of “Cabin” and “Age” have loads of missing values, especially “Cabin”.

There are several other examples that can be provided for discussing data cleaning and its need, but we shall limit to only the above two examples for now.

### 4.3.1 The difference between data cleaning and data preprocessing

Data cleaning and data preprocessing are often confused terms. The main reason for this uncertainty is the reason that data cleaning is often treated as a part of data preprocessing in academic literature, but the two terms are used interchangeably in the applied world of data science. However, to maintain continuity and consistency with the usage of the two terms in this project, we have decided to define both data cleaning and data preprocessing as disjoint entities or separate steps in the data science process. The diagram for the same is displayed in fig(4.6).

“Data cleaning refers to the process of converting raw data into a form more suitable for analysis such that this new form will aid in unbiased, clear, trustworthy and robust analysis”

“Data preprocessing refers to the process of converting cleaned data(obtained after the data cleaning process) into a form more suitable for modelling tasks such as machine learning”

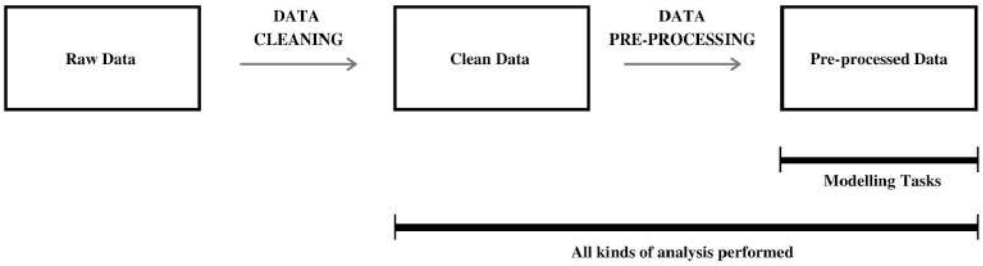



**Fig (4.6)** Difference between data cleaning and data preprocessing

### 4.3.2 Common data cleaning steps in tabular data

Some very common data cleaning steps undertaken in the context of tabular data are as follows:

#### 4.3.2.a) Renaming features

**Issue:** Feature names that are too long, have capital case letters, have spaces

**Negative effect:** Becomes difficult to maintain long, difficult column names throughout the analysis

**Reasonable operations:**

- Convert to lower case
- Replace spaces with _
- Explicitly rename all features (Can be cumbersome if there are too many features)

#### 4.3.2.b) Removing constant features

**Issue:** Features with a constant value across all instances are called constant

**Negative effect:** Constant features can't help learn any distinguishing point between two instances

**Reasonable operation:** Remove such features

**WARNING:** Sometimes, such features might be important takeaways in a data analytics report. For example, a constant value of October throughout the "Month of Crime" feature in a crime dataset for 2019 in the city of Bangalore can indicate that all crimes in the year 2019 in Bangalore took place only in October.

#### 4.3.2.c) Removing empty features and empty instances

**Issue:** Improper data importing from spreadsheets might cause the appearance of empty features and instances

**Negative effect:** A wastage of space

**Reasonable operation:** Remove such features and instances

#### 4.3.2.d) Round features having float values

**Issue:** Inconsistent decimal points across values in numeric features

**Negative effect:** Non-uniform precision

**Reasonable operation:** Round of all values to a uniform value, eg: to 3.

#### 4.3.2.e) Remove duplicate instances

**Issue:** Duplicated instance entries

**Negative effect:** A wastage of space

**Reasonable operation:** Remove such instances

#### 4.3.2.f) Remove stray whitespace

**Issue:** Stray whitespaces in values

**Negative effect:** Inconsistencies will occur; eg. "life boat" and "lifeboat" are different.

**Reasonable operation:** Replace whitespace with '_'

#### 4.3.2.g) Maintain data type consistency

**Issue:** Features might be of data types that they are not originally supposed to be of. For example, a "Money" feature should be numeric for easy calculation. But, if it's values are encoded with symbols like "$", then the feature will be stored as non-numeric.

**Negative effect:** Inhibits easy analysis

**Reasonable operation:** Remove non-numeric symbols in features meant to be numeric

#### 4.3.2.h) Dealing with spelling inconsistencies

**Issue:** Misspelt values make one single value look like multiple values

**Negative effect:** Inconsistencies like the one mentioned in section 4.3

**Reasonable operation:** Rectify spelling errors, either manually or with computing techniques

#### 4.3.2.i) Dealing with Missing Data

**Issue:** Missing values in data

**Negative effect:** Missing values in data cause several issues like incomplete analysis, errors while modelling, mistaken insights etc.

**Reasonable operations:** The operations mentioned in [34] are comprehensive techniques in dealing with missing data. In the context of this project, we limit ourselves to a smaller subset of methods to deal with missing values (as shown in fig(4.7)).

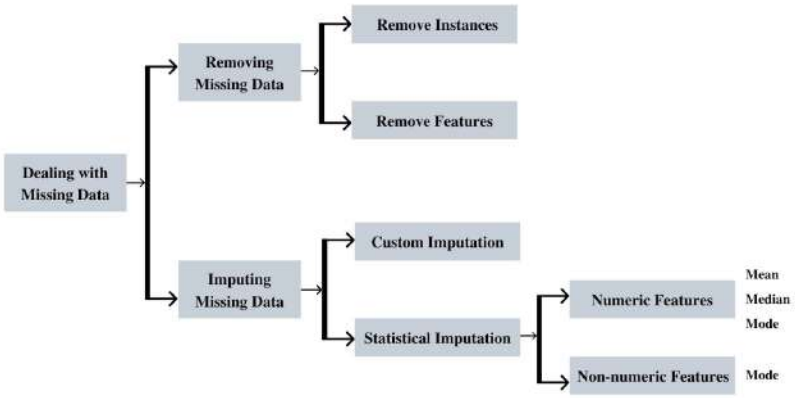



**Fig (4.7)** Dealing with Missing Values - A Block Diagram

## 4.4 Introducing a novel metric to assess the quality of tabular data

This section introduces a new metric that is aimed at quantifying the quality of tabular data. It has been named "Tabular Data Cleanliness Score" and will be abbreviated as TDCS in all further discussions.

### 4.4.1 Important components in the TDCS computation

- **Messy Value:** Any value in a particular cell in a given dataset that is
  - Missing(NA)
  - Misspelt words
  - Multiple values combined as one(eg: apple, mango)
- **Feature Mess(FM)**: A value determined by ($Num.\ of\ Messy\ Values\ /\ Num.\ of\ Instances\ in\ the\ Feature$)
- **Whole Instance Mess(WIM)**: Takes into account the ratio of whole instances in a dataset that are messy. These include:
  - Num. of duplicate instances(Make sure to exclude the original instance)
  - Num. of gibberish instances(Pretty common in surveys)
    - Instances that the user does not recognize as valid data points
    - Cases where the headers of the dataset form an instance

    - Num. of instances where the instance-wise missing value percentage > Threshold(To be decided by the data examiner)
- **Whole Feature Mess(WFM)**: Takes into account the ratio of whole features in a dataset that are messy. These include:
    - Num. of features where the feature-wise missing value percentage > Threshold(To be decided by the data examiner)
    - Num. of features where the feature's data type is not suited to the interests of the user
    - Num. of features with duplicate column names(Keep the original)
    - Num. of features that are numerical and have high cardinality
    - Num. of features that have a separator like : or ; or , in their values [Usually used for multi-label problems]
- **Other Dataset Concerns(ODC) :** Concerned with other aspects of a good dataset like
    - Unique Identifier(At least one feature must exist with num. of unique values = num. of instances, else it is a concern)
    - Spaces in column names is flagged as a concern

### 4.4.2 Formula

$$TDCS = 1 - (\sum_{i=1}^{N_f} (FM)_i(weight)_i + (WIM) + (WFM) + (ODC)(0.1))$$

**…** Eq. (4.1)

$N_f$ is the number of features in the dataset

### 4.4.3 Some more specifics about TDCS

- TDCS of 1 indicates a **COMPLETELY CLEAN DATASET**
- The setting of the *weights* can seriously affect how small or big is TDCS. Therefore, it is best to keep these metrics as constants.
- FM, WIM and WFM are all ratios
- The *weights* decide the emphasis a user wants to give to each feature in the dataset. Certain features having messy values can be more dangerous than certain other features.
- All features in the dataset can be categorized as

- If Wrong, Innocuous(IWI)
- If Wrong, Harmfull(IWH)
- Only a domain expert will be able to categorize features as IWI or IWH perfectly
- By default, all features will be considered as IWH
- *weights* depend on what a given feature is
  - If its IWI, *weight* = 0.10
  - If its IWH, *weight* = 0.30

## 4.5 Data Preprocessing

Data preprocessing [15] has already been discussed in the perspective of contemporary data science research in section 2.2. The scope of our project currently deals with basic preprocessing techniques. Before discussing these techniques, the next section will provide a brief review on the different types of features that can exist in a dataset.

### 4.5.1 Types of features in a dataset

The notable Harvard psychologist Stanley Smith Stevens, was amongst the first to lay foundations for creating a classification for different kinds of data in his paper [35]. As per it, they were essentially 4 different types of data:

1. **Nominal Scale:** Numerals that can be used numbers or labels. These can have any character and can be used without much restrictions except for the fact that no two different data points should be represented by the same nominal representation. These values have no statistical meaning. Eg: Names of footballers

2. **Ordinal Scale:** Values in this scale have some kind of an implicit relative-order associated with them. It is not necessary that all the values have the same difference between them. Mean and standard deviation of such values have no meaning. Eg: A student's preference for a course on a 5-point scale
3. **Ratio Scale:** Values are numbers and all kinds of statistical definitions have meaning here. These values have a meaningful zero value. Eg: Marks obtained in an examination
4. **Interval Scale:** Values represented in this scale are interval measures that depict a range within which the original value can lie. The zero value of an interval scale does not exist. A mean can be a reasonable estimate of an interval. Eg: The temperature of a room for one hour

With these measures as foundations, several other scales such as those mentioned in [36]. In this project, we have focussed on the initial guidelines set by Stevens [35] and have formulated a classification of features which we feel is a reasonable categorization for contemporary data science.

**Main Categories:**

- **Numerical Features:** Features that have values that only consist of numbers
- **Non-numerical Features:** Features that have values that consist of characters other than just numbers

**Sub Categories:**

| Feature Type | Main Category |
|---|---|
| Ordinal | Numerical or Non-numerical |
| Nominal | Non-numerical |

| | |
|---|---|
| Ratio | Numerical |
| Interval | Non-numerical |

**NOTE:** Usually in datasets, intervals are represented with a separator. Eg: "45-55". Hence, interval features belong to the non-numerical type here.

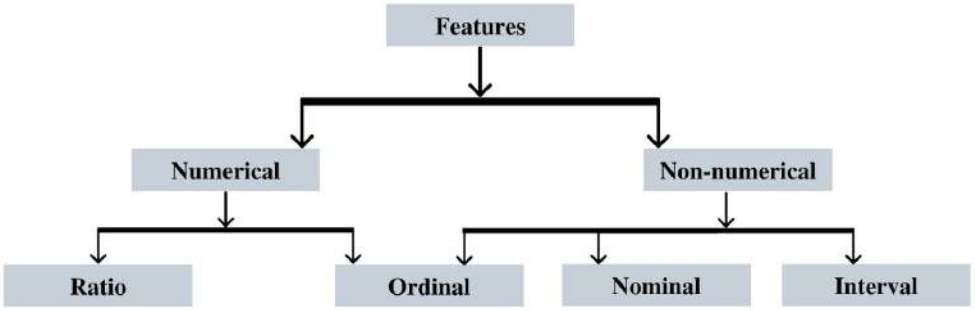



**Fig (4.8)** Types of features in a tabular dataset

### 4.5.2 Preprocessing techniques used

There are several kinds of preprocessing techniques [36] that are commonly used to deal with tabular data. However, the scope of this project deals with only a small subset of these techniques currently, and these methods are applied based on the type of features(section 4.5.1) that need these.

#### 4.5.2.a) Ratio Features

It is a useful first operation in certain use cases of data science to identify values that are too small or too large(with respect to the majority of values in the features) as outliers [37] and remove them.

In fig(4.9), $r$ is the Pearson Correlation Coefficient [39]. In basic terms, it defines how closely the two features (x-axis and y-axis) are related. A positive $r$ indicates that if one increases, the other will increase too(hence the upward sloping straight line).

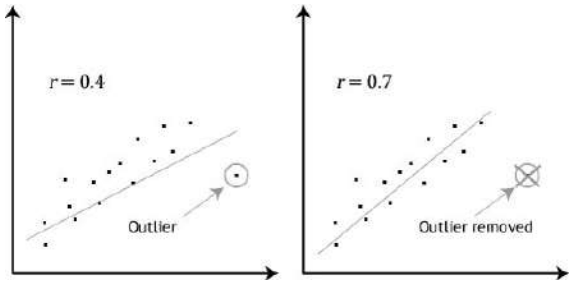



**Fig (4.9)** Effect of outlier removal [38]

In the graph to the left the value of $r$ is 0.4 as the existence of an outlier causes the line fitting the points to slightly deviate towards the outlier so as to accommodate it. However, in the graph on the right, this outlier has been removed and this causes the line to fit all the points in better fashion(i.e minimize sum of perpendicular distance between each point to the line) and also increases $r$ to 0.7.

Another very common problem with ratio features is that values tend to have a very wide range of distribution. For example, if a feature X has 100 values with a range of 10,000; it causes the model to not generalize well enough during testing as the range of 10,000 is too wide a range for it to make satisfactory predictions on. Fig(4.10) depicts the two most common used scaling techniques to bring data points within a fixed range. While several beginner level questions are aimed at when to use standardization and when to use normalization [40], the main point to consider here is that normalization deals with squashing values and mapping them into a corresponding value lying between 0 and 1. Standardization on the other hand

converts all values together such that they take shape of the gaussian distribution [41] with a mean of 0 and a standard deviation of 1 (Fig. (4.10)).

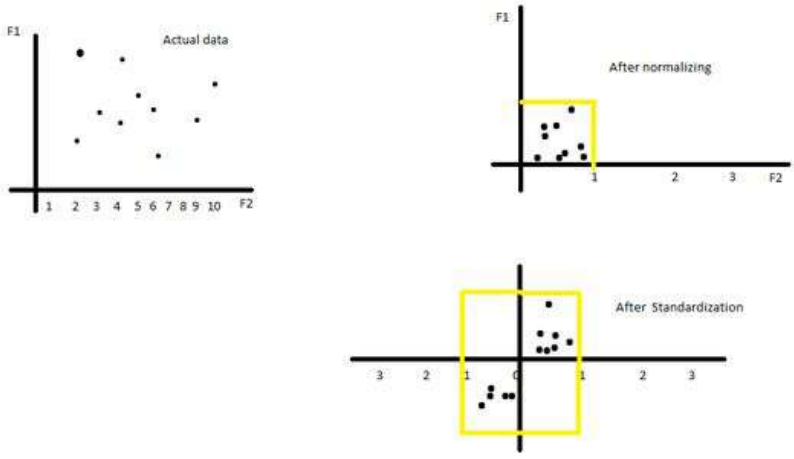



**Fig (4.10)** Standardization and Normalization

Another point of comparison between these two scaling techniques is better explained through their formulae.

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$ … Eq. (4.2) [Normalization]

$x$ represents the old value.

$$x_{new} = \frac{x - \bar{x}}{s}$$ … Eq. (4.3) [Standardization]

$\bar{x}$ is the mean of all values in the feature. $s$ is the standard deviation of all values in the feature.

In eq.[4.2], the new value depends only on the current corresponding value and the minimum and maximum of all values in the feature. However, in eq.[4.3], the new value depends on all values in the feature(as mean and standard deviation is being used).

#### 4.5.2.b) Ordinal Features

Ordinal features as discussed in section 4.5.1 are features that contain values that have a relative-ranking implied onto them. These features can belong to both numerical as well as non-numerical categories. For example, a ranking scale of 1-5 is numerical. Similarly a non-numerical "level of education" feature will exhibit ordinal characteristics as High school < Bachelors < Masters < Doctorate. In non-numerical ordinal features, we shall follow categorical encoding [42], in specific "Label Encoding".

Label encoding is a procedure that converts labels into numerical values. It is a fairly simple operation and is best suited when the features in consideration are ordinal. The same is depicted in fig. (4.11).

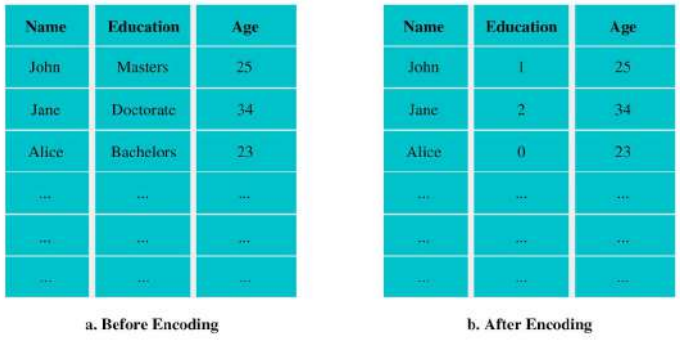



**Fig (4.11)** Label encoding

The label encoding is performed based on the domain knowledge that Doctorate > Masters > Bachelors.

#### 4.5.2.c) Nominal Features

In practice, nominal features are encoded in many ways [42], but for this system, we shall only use one-hot encoding. One-hot encoding is a highly robust approach when features are not ordinal. However, a potential drawback with one-hot encoding is that it introduces a large amount of new features and leads to high dimensionality [44].

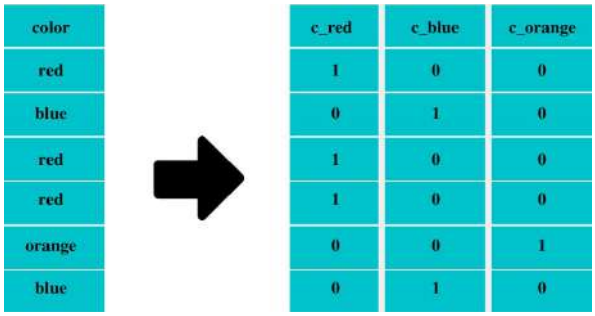



**Fig (4.12)** One-hot Encoding

#### 4.5.2.d) Interval Features

For interval features, we will be using mean encoding. In this relatively simpler step, we will pick both the minimum and maximum values in a given feature value, compute the mean and encode with this mean value.

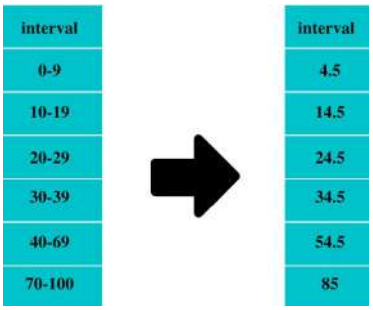



**Fig (4.13)** Interval Encoding

## 4.6 Data Visualization

Data visualization [47] identifies with the concept of communicating relations in data with the help of graphical representations and is often described as being both art and science [48]. In the context of data science, data visualization is used in the step of Exploratory Data Analysis(EDA). EDA is an important step that helps identify hidden relations within data. The discovery of such relationships prove of great importance to data scientists in order to be able to perform complex operations such as feature engineering and generation. Data analysts too require extensive data visualization in their projects in order to be identifying and communicating relevant patterns of interest to potential stakeholders. The scope of data visualization in this project is minimal at the time this report is prepared, however future versions of the proposed system will include more visualization techniques as presented in [49].

## 4.7 User-interaction Based Systems

In section 1.6, the limitations of the project have been identified. A main point that is of concern when building an automated system is “How much liberty does the system have to make choices on its own? Is complete automation a smart option?”. In the

context of our project, the focus is on data. Therefore, providing the system abilities to completely clean data in an automated fashion might lead to questionable end results. This concern takes the centre of attraction when we realise that sans domain knowledge, any kind of data analysis will be severely limited. It is also a matter of thought about adding domain knowledge to the system. However, such an idea will fail as it is technologically very difficult to incorporate all kinds of domain-related theory into a system prior to analysis. Therefore, the best alternative will be to create a new kind of system that would work with human domain experts to clean data and reduce the effort that needs to be undertaken to do so. In fig(4.14), the basic structure of a User Interaction Based System(UIBS) has been provided. User interaction has been a very important part of software development since ever. Studies have been made in the past to project the importance of user interactions in the success of software establishments [46].

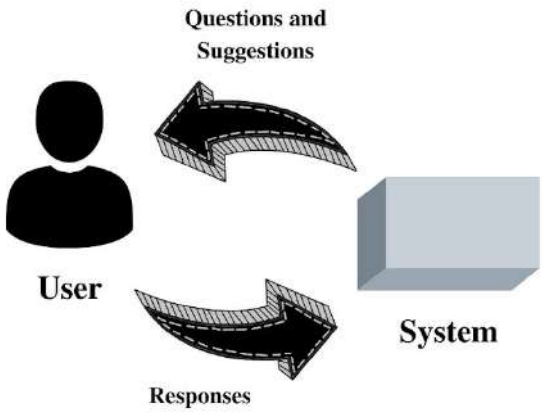



**Fig (4.14)** Basic structure of User-Interaction based systems

In this project, our aim is to use this idea where a user interacts with the system to use it in better fashion to help solve the limitations of the project that might arise due to complete automation. Hence, the name User Interaction Based System.

We define the UIBS as

“A software system built to perform a well-defined task with the help of constant feedback from a human user. This feedback includes responses to questions asked by the system to the user(questions that mostly take the shape of multiple-choice-questions or one-line answers). The system can also suggest preferred answer choices to the user, when it is possible to do so.”

In the next chapter, the main block diagram of the UIBS built for this project will be discussed.

## 4.8 Deep Learning Automation

The process of engineering an Artificially Intelligent solution for a given problem is a very time consuming one. The pre-processing and modelling steps included in the same are crucial parts of the workflow and take months to finish. We have studied this process extensively and have come to understand that the basic early steps of the process can be generalized. We aim to shorten the duration taken by this process by automating the repetitive part of these steps and consequently help professionals to focus on further improving the model by learning, creating and applying techniques that are not among the common practices and guidelines. Furthermore, we desire to encourage more people to dive into the innovative field of Artificial intelligence by providing guidance software to help them learn.

### 4.8.1 Supervised Learning

Supervised learning is the task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way [54]. We have included supervised learning in this work as the initial modelling steps can be generalized to an extent and applied. Moreover, the whole process can be specified in a clear step-wise manner. On the other hand. The methodology of developing a solution using unsupervised learning varies closely with each dataset and hence requires a lot of human intervention, rendering the automation of unsupervised learning unavailing.

### 4.8.2 Important Parameters

1. Bayes Optimal error: In statistical classification, the Bayes error rate is the lowest possible error rate for any classifier of a random outcome and is analogous to the irreducible error. [55]
2. Avoidable bias: It is the difference between the Bayes optimal error and training error. High avoidable bias indicates that the model is underfitting. This can be solved by making the neural network deeper or by increasing the number of epochs.

3. Variance: It is the difference between the training error and validation error. High variance indicates that the model is overfitting. This can be solved by using Regularization or getting more data

### 4.8.3 Choosing the number of Epochs

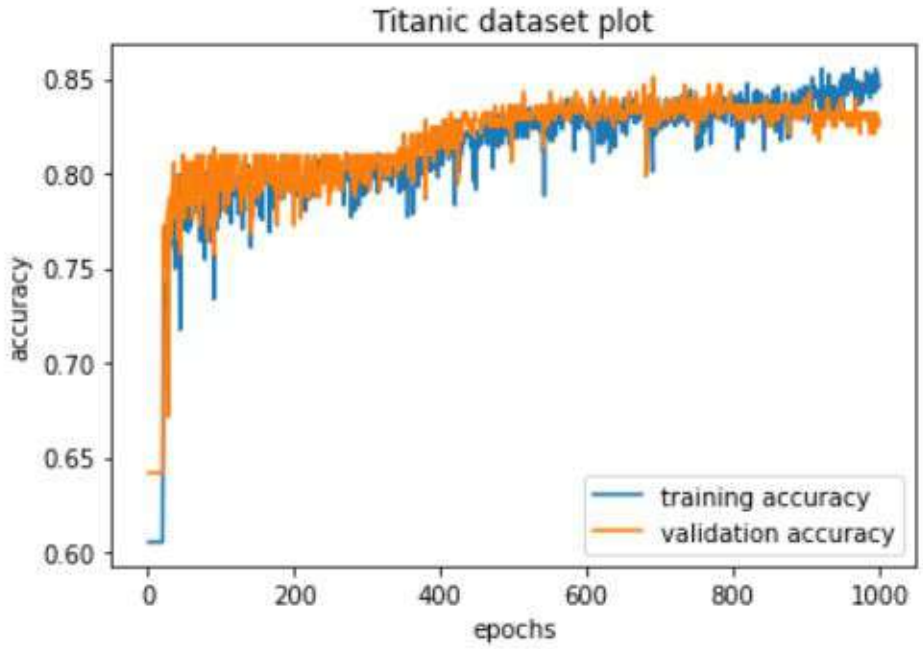



**Fig (4.15)** Epoch-wise accuracy line graph for the titanic dataset

The right balance between bias and variance is achieved at around 700 epochs. It is seen that the training accuracy starts rising after 700 epochs and the validation accuracy starts to fall. This indicates that the model is overfitting the training set. Ergo, the number is set to 700 and this gave good results for a number of datasets.

### 4.8.4 Regularization

It is the process of adding information in order to solve an ill-posed problem or to prevent overfitting. Two types of regularization were considered in this work.

1. Weight Decay: This method adds a term that penalizes the weights if they are too high. Smaller weights promote function linearity and consequently reduce overfitting. The regularization parameter is set to 0.01 as it gave the best results
2. Dropout regularization: Weights of a neuron are adjusted for some specific feature. Due to this, the neurons end up relying on this specialization of other neurons. This concurs the power of individual neurons. Hence, neurons are selected at random and "dropped out" in each iteration, based on a specified probability. This enhances the individual strength of neurons as other neurons are pushed to take over when one neuron is dropped, resulting in better generalization. Dropout probability used is 0.2 as it is the default and is known to give good results.

# 5. PROPOSED SYSTEM

This chapter explains in detail the multiple steps involved in the design of the automated data science system which is the main subject of this project. As a means of building the system, we have followed a model that is loosely based on the Agile Software Development Life Cycle [51].

The core of the agile method lays importance on incremental development, constant planning, responsiveness to change, collaboration and iterative improvement. In building the automated data science system, which we shall henceforth refer to by the name “DataSwissKnife”, the team has followed a distinct process built on

- Incremental development
- Evolving plans and ideas
- Iterative feature addition

## 5.1 Project Goals

The main goals to be met by the tool are

1. Easy-to-use interface
2. Help clean data without the need to write code
3. Store results for future use in the user’s local system
4. Generate quick visualizations and reports
5. Incorporate automated modelling techniques
6. Comprehensive documentation to allow modifications by the open source community

## 5.2 Key Considerations

The 5 main considerations of this project are

1. **Functionality**: Must do what is specified by the problem statement
2. **Speed:** Must help users clean data and auto-generate results significantly faster than performance achieved by manual writing of code
3. **User Interface:** Must be easy to use and should provide clear instructions
4. **Well Written Code:** Must be written keeping in mind good coding practices
5. **Documentation:** Must be clear and concise so as to allow for open source contributions

## 5.3 User Stories

As described in section 1.4, there are 3 main users for whom this software has been intended. Fig(5.1) describes one line user stories for each of them to summarise their needs.

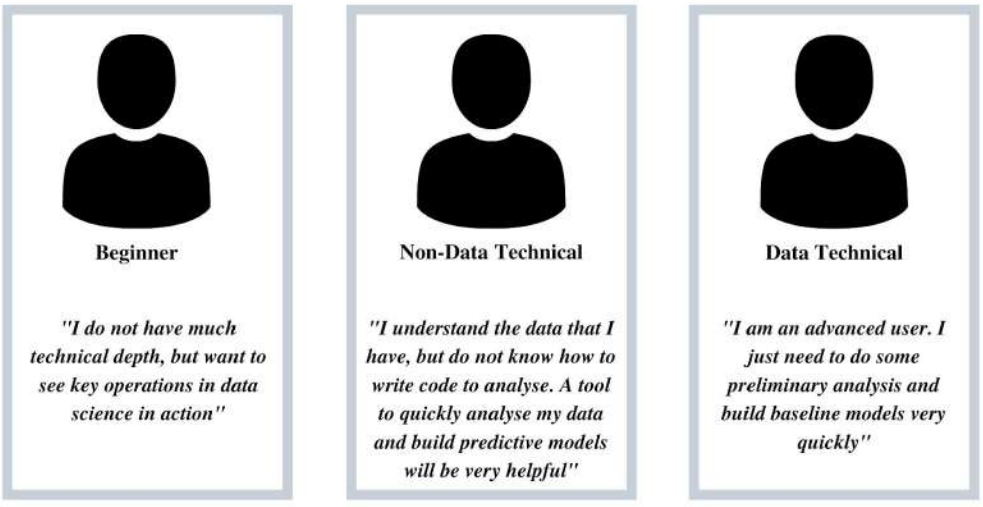



**Fig (5.1)** User Stories

## 5.4 System Block Diagram

DataSwissKnife's high level architecture is based on UIBS (section 4.6). The corresponding architecture is provided below(fig.(5.2)) and the explanation of the architecture is given in section 5.4.1.

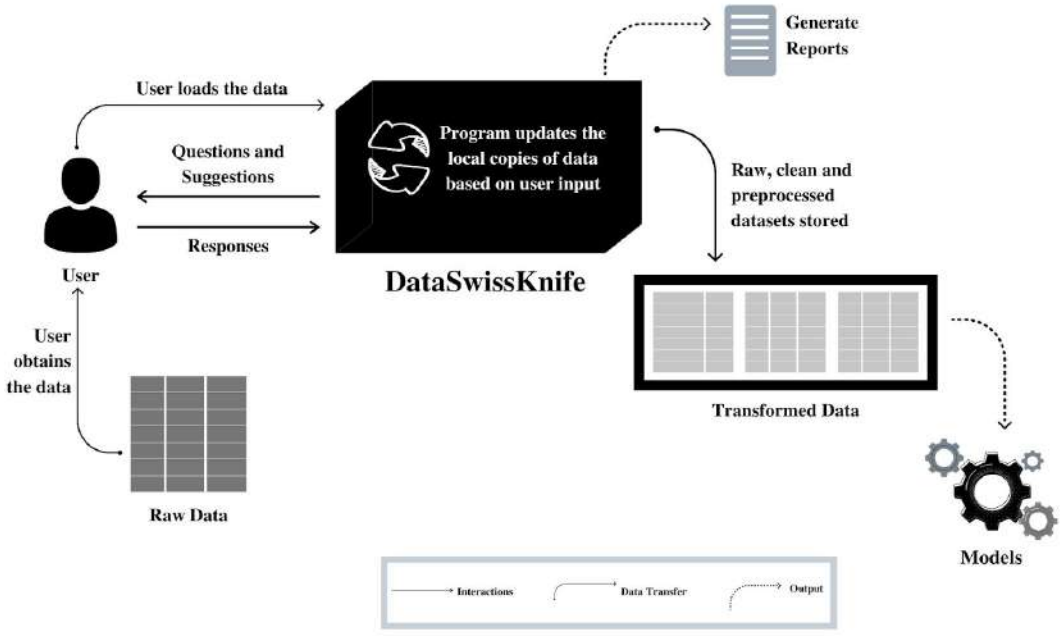



**Fig (5.2)** System High-Level Architecture

### 5.4.1 Explanation of Data Flow through the Architecture

First, the user **obtains** raw data. This raw data is expected to be in a tabular format and if it is to be used for supervised classification, it should have both descriptors as well as the target feature. The user then **loads** this data onto the DataSwissKnife program. From here, the system-user interaction begins. The first step is for the program to ask the user to **choose a directory** in his or her local system where all the

contents and results of the execution of the program will be stored. Once this has been decided, the program **creates a structure** in the user designated directory. After this, the program **starts asking questions** to the user that will be based on what the user wants to do with the data. For example, the program can ask the user the action he or she would like to take up to deal with missing values in a given feature.

The program also **provides suggestions** where possible. It **performs corresponding actions on the data** based on the user's responses. After all data operations(cleaning and preprocessing), the **final outputs are stored** in the project's directory(chosen earlier by the user) as raw data, cleaned data and preprocessed data.

The preprocessed data is **passed onto the model** training section. The program also **generates visualizations and reports** from the data provided and also the intermediate data outputs.

### 5.4.2 Fundamental Building Blocks of the System

The DataSwissKnife system is built on 8 fundamental building blocks. It can rather be said that the entire system will function at its highest ability when it has all of these blocks in working condition and integrated with the main system. Fig(5.3) depicts these 8 blocks.

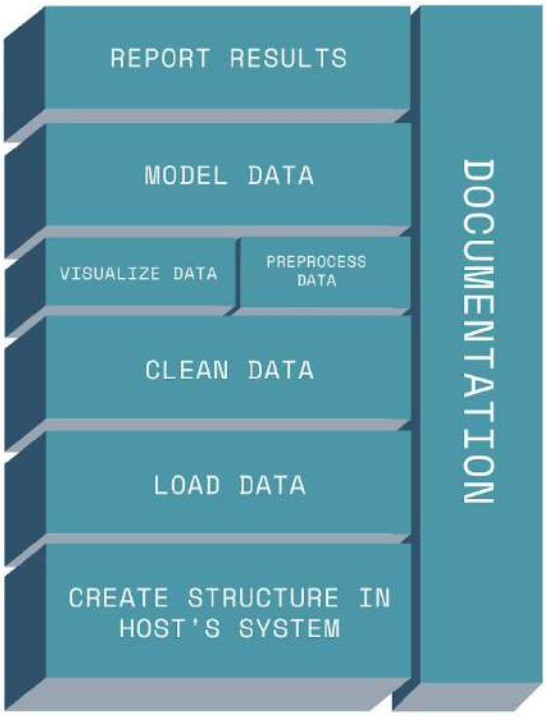



**Fig (5.3)** 8 Building Blocks of the System

**Create Structure in a Host's System**

- Foundational block
- Creates a project directory structure in the user's local system(More about the structure will be explained in section 5.6)

**Load Data**

- Loads raw data(resident on the user's local system) onto the program
- Also creates a copy of the dataset into the project directory structure

**Clean Data**

- Cleans raw data and stores cleaned copies in the project directory structure

**Visualize Data**

- Auto-generates visualizations from the data
- Stores these visualizations in the project directory structure

**Preprocess Data**

- Preprocesses data i.e readies cleaned data for modelling operations

**Model Data**

- Automate supervised classification methods
- Produce baseline results and preliminary models
- Store these models

**Report Results**

- Provide preliminary reports about the dataset

**Documentation**

- More like a pillar than just a block
- Important throughout the system
- Code documentation, how-to-use guides etc.

## 5.5 Dissecting DataSwissKnife

The aim of this section is to dissect the DataSwissKnife system and provide low-level details of how each component works.

### 5.5.1 Create Structure in the Host's System

The first action performed is to create a hierarchical directory structure within the user's computer system. The user can choose the location where he or she wants to place this project structure (as in fig. (5.4)). The ROOT name or the name of the project has to be set by the user.

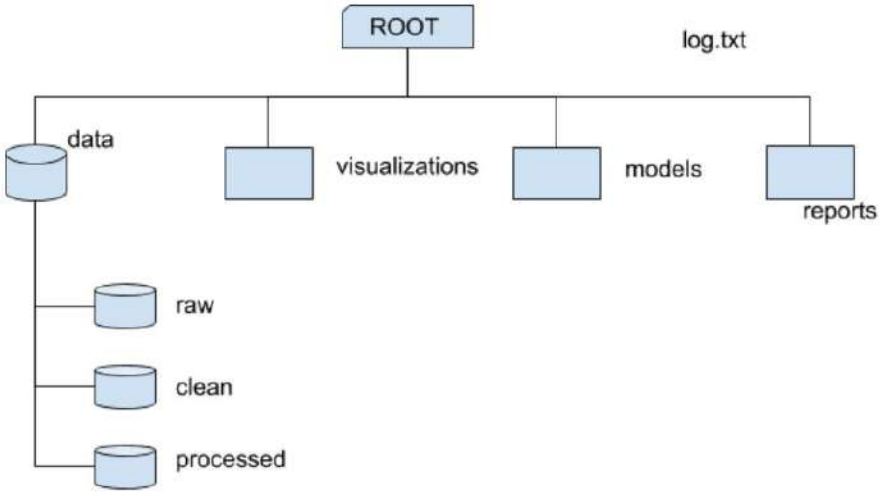



**Fig (5.4)** Project Structure in the Host's System

The subdirectories are data, visualizations, models and reports. The data subdirectory is further divided into raw, clean and processed. Raw has the original copy of the dataset, clean has the cleaned copy and processed has the data that is ready to be passed into an ML model.

Visualizations contain the auto-generated visualizations from data and models has the machine learning models as .pkl files. Finally, reports will contain basic, preliminary reports of the data and analysis performed.

The importance of this step is that it makes the results reproducible. This means that the user can access cleaned and preprocessed intermediaries of the dataset, visualizations and basic models at a later stage without having to run this whole program again.

Figures 5.5 ) and 5.6) provide screenshots of this procedure.

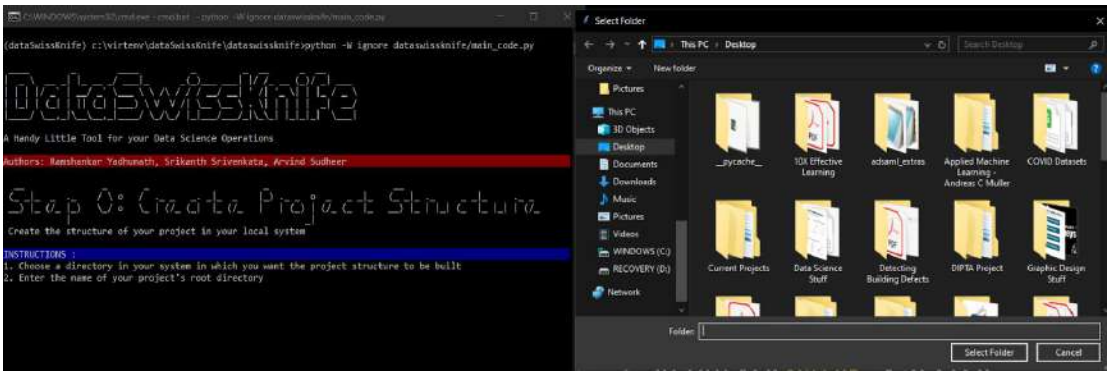



**Fig (5.5)** Choosing the location to place the project directory

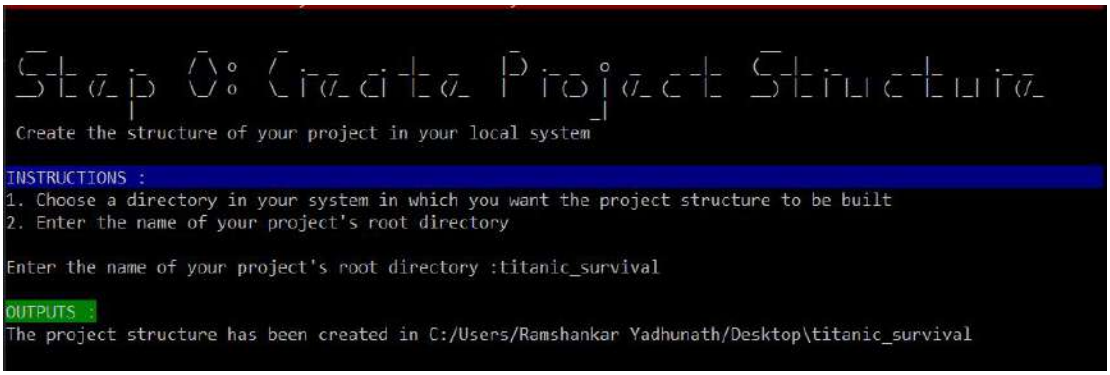



**Fig (5.6)** Naming the project directory; The project directory created in the system and path is provided

### 5.5.2 Load Data

After the project structure has been made, the next step is to load the data. As of its first version, DataSwissKnife only supports .csv files. It is expected that the user already has the dataset in his local system. When this dataset is loaded onto the project through DataSwissKnife(DSK), a copy of this is stored in root/data/raw.

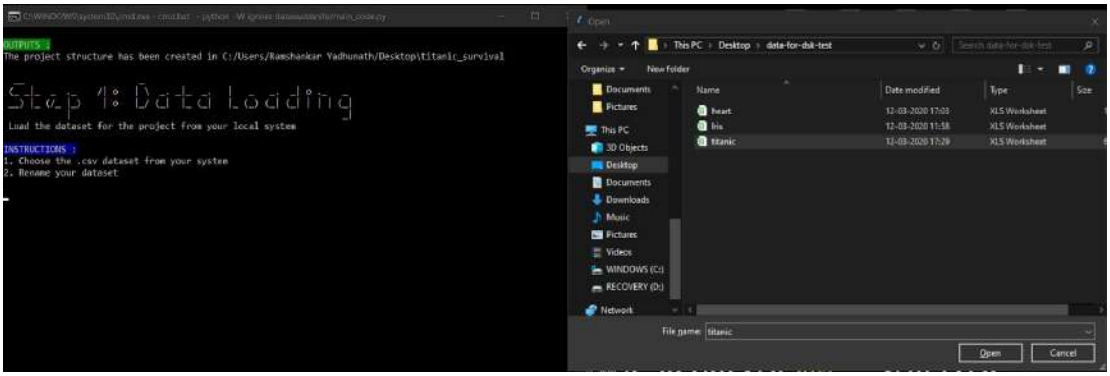



**Fig (5.7)** Choosing the dataset from the local system

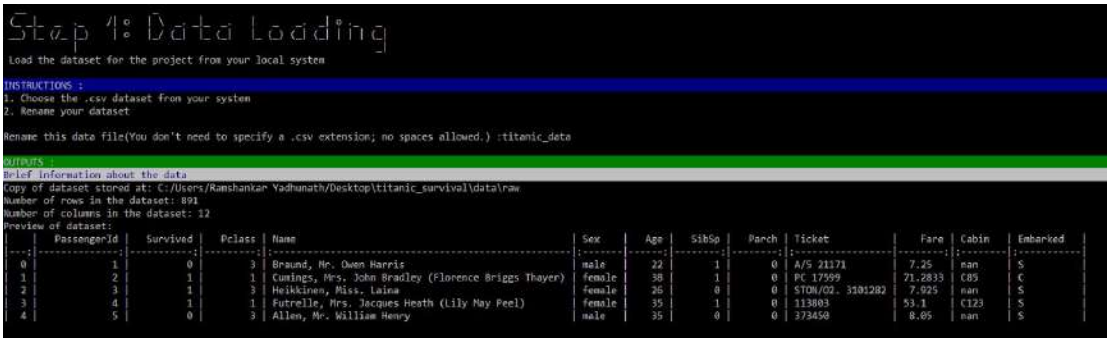



**Fig (5.8)** Output after loading and renaming the dataset; a preview is generated

### 5.5.3 Clean Data

This block is concerned with cleaning up the data. A series of questions are asked by DSK and based on user response, the data is cleaned. DSK also performs some preliminary cleaning at the start as soon as it receives the data(without user intervention) and these automated cleaning steps are:

- Rename column names such that there are no spaces in the names and all characters are lowercase
- Remove columns that have only a single value for every instance
- Remove empty columns and rows
- Round of decimal values to 3 decimal places
- Remove duplicate rows

- Remove trailing and leading spaces for values

The other cleaning steps are dealing with missing values and consistency checkers. DSK asks questions to the user, which have one word answers. Based on user responses, cleaning is performed.

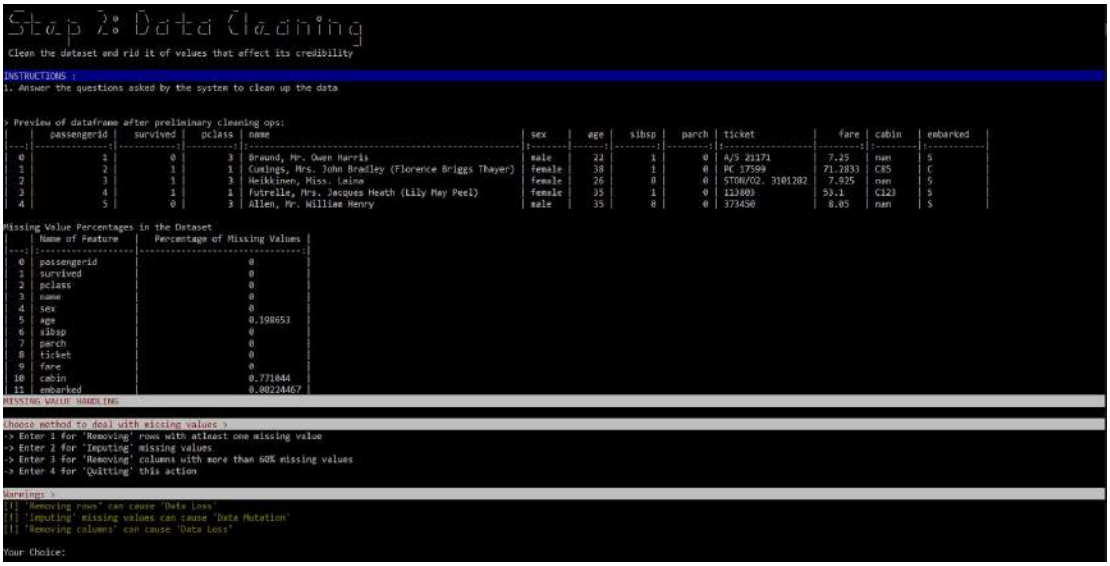



**Fig (5.9)** The initiation of Data Cleaning

Missing Value Percentages in the Dataset

| | Name of Feature | Percentage of Missing Values |
|---|---|---|
| 0 | passengerid | 0 |
| 1 | survived | 0 |
| 2 | pclass | 0 |
| 3 | name | 0 |
| 4 | sex | 0 |
| 5 | age | 0.198653 |
| 6 | sibsp | 0 |
| 7 | parch | 0 |
| 8 | ticket | 0 |
| 9 | fare | 0 |
| 10 | cabin | 0.771044 |
| 11 | embarked | 0.00224467 |

**Fig (5.10)** Missing value percentages in the data

Missing Value Percentages in the Dataset

| | Name of Feature | Percentage of Missing Values |
|---|---|---|
| 0 | passengerid | 0 |
| 1 | survived | 0 |
| 2 | pclass | 0 |
| 3 | name | 0 |
| 4 | sex | 0 |
| 5 | age | 0.198653 |
| 6 | sibsp | 0 |
| 7 | parch | 0 |
| 8 | ticket | 0 |
| 9 | fare | 0 |
| 10 | embarked | 0.00224467 |

**Fig (5.11)** Missing value percentages in the data after removing features with more than 60% as missing values

```
MISSING VALUE HANDLING

Choose method to deal with missing values :
-> Enter 1 for 'Removing' rows with atleast one missing value
-> Enter 2 for 'Imputing' missing values
-> Enter 3 for 'Removing' columns with more than 60% missing values
-> Enter 4 for 'Quitting' this action

Warnings :
[!] 'Removing rows' can cause 'Data Loss'
[!] 'Imputing' missing values can cause 'Data Mutation'
[!] 'Removing columns' can cause 'Data Loss'

Your Choice: 2
Imputing Missing Values as per your request...
Imputation For  age  :

Choose method for imputing values :
-> Enter 1 for 'Statistical Imputation'
-> Enter 2 for 'Custom Imputation'

Warnings :
[!] 'Statistical Imputation' can be 'Overgeneralized'
[!] 'Custom Imputation' can be 'Time Consuming'

Your Choice: 1
Performing Numeric Imputation...
Choose method for imputing missing values in numerical features :
-> Enter 1 for 'Imputing with Mean'
-> Enter 2 for 'Imputing with Median'
-> Enter 3 for 'Imputing with Mode'

Your Choice: 1
Imputation For  embarked  :

Choose method for imputing values :
-> Enter 1 for 'Statistical Imputation'
-> Enter 2 for 'Custom Imputation'

Warnings :
[!] 'Statistical Imputation' can be 'Overgeneralized'
[!] 'Custom Imputation' can be 'Time Consuming'

Your Choice: 1
Performing Non-numeric Imputation...
```

Missing Value Percentages in the Dataset

| | Name of Feature | Percentage of Missing Values |
|---|---|---|
| 0 | passengerid | 0 |
| 1 | survived | 0 |
| 2 | pclass | 0 |
| 3 | name | 0 |
| 4 | sex | 0 |
| 5 | age | 0 |
| 6 | sibsp | 0 |
| 7 | parch | 0 |
| 8 | ticket | 0 |
| 9 | fare | 0 |
| 10 | embarked | 0 |

**Fig (5.12)** Performing Imputation to fill in the remaining missing values; all missing values have been dealt with

Another step of data cleaning is to check for datatype inconsistencies. A data type inconsistency occurs when a feature is in a datatype that does not facilitate analysis that the user would like to do. An example of this is when a feature like money is

encoded as a non-numeric type(i.e object in pandas) because of entries like “50$”. In such a case, DSK’s datatype inconsistency dealer module will first identify these inconsistencies based on the user’s responses to how he or she would prefer a feature to be (numeric or non-numeric) and checks these against the actual types with which these features have been encoded. After identifying inconsistent features, DSK flags off non-numeric characters. In our example of money, “$” is a non-numeric character. DSK flags this off and provides the user with an opportunity to remove these values from the data. After removing “$” symbols, we can see that our money feature can now be converted into numeric type easily. If there are no inconsistencies, then that too is informed to the user (fig 5.7 e).

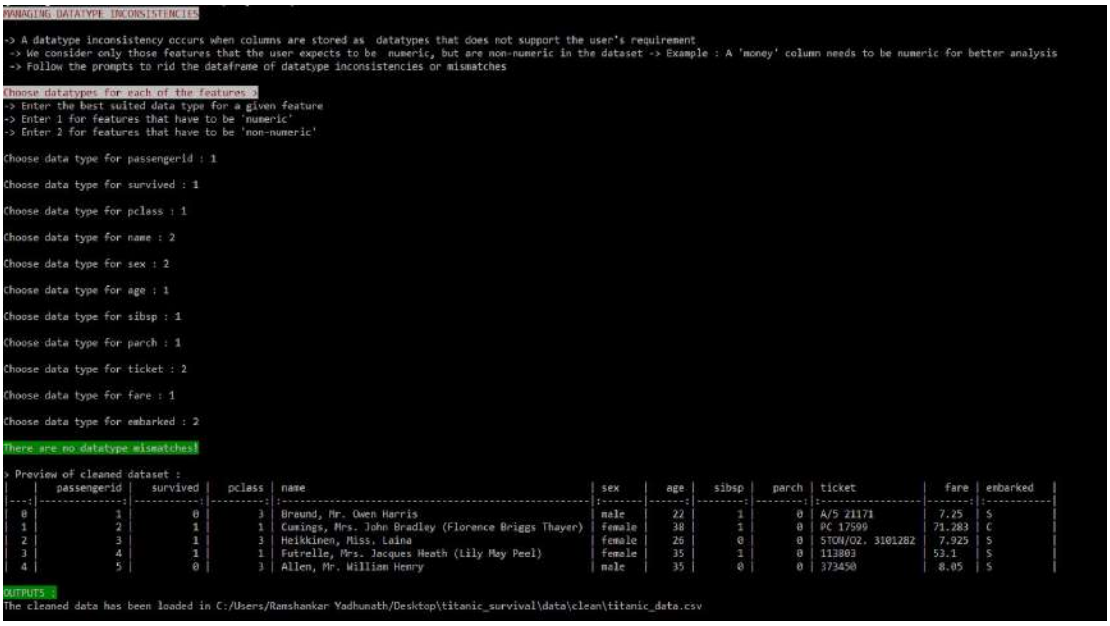



**Fig (5.13)** Datatype consistency management; clean data stored

After this step, the clean data is stored in root/data/clean

### 5.5.4 Preprocess Data

Data preprocessing concerns itself with converting clean data into a format more suited for machine learning modelling. So far, the data has only been a single .csv file with both descriptors and target features together. In preprocessing, a series of steps are performed that convert this single .csv file into train.csv, test.csv(has test data descriptor values) and test_solution.csv(has test data target values). 80% of the original data is maintained as the train set and 20% is maintained as the test set. Initially, these 3 files are stored in root/data/clean. But after all preprocessing steps, the preprocessed versions of these 3 files are stored in root/data/processed.

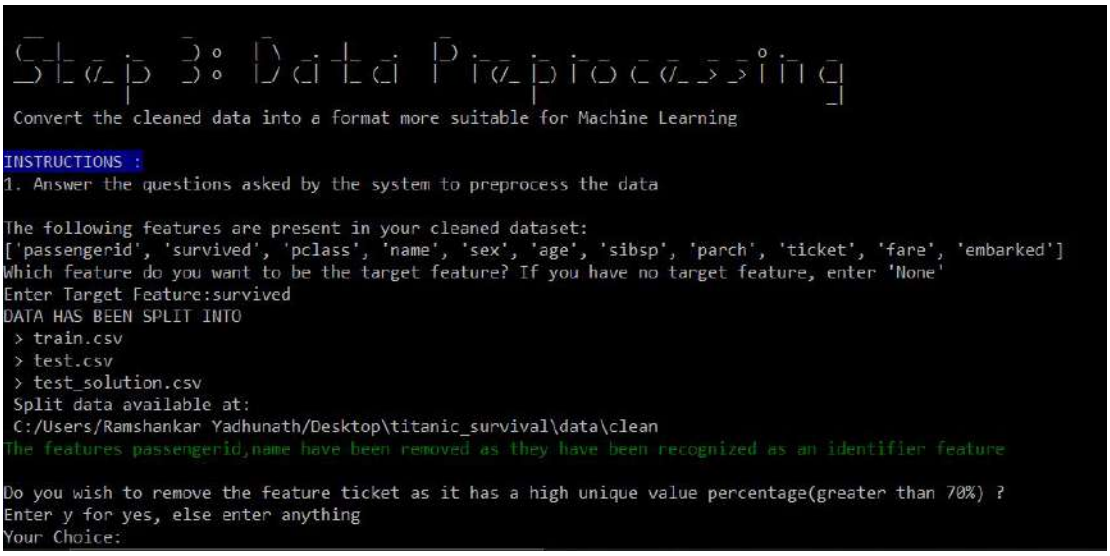



**Fig (5.14)** Prompts to enter the target

Steps performed are

- Automatically detect identifier features(unique for every instance in the dataset) and remove them
- Detect features with more than 70% values as unique and ask the user if he or she wants to keep these values or remove them

- Interact with the user to identify the specific data type for each given feature. Section 4.5.1 explains the four different feature types DSK considers.
    - Asks if feature is **ordinal**
        - If response is no
            - If feature is numerical, the feature is of **ratio** type
            - If feature is non-numerical, asks if feature is **interval** type
                - If response is no, feature is **nominal**
                - Else, feature is **interval**
        - Else, feature is **ordinal**

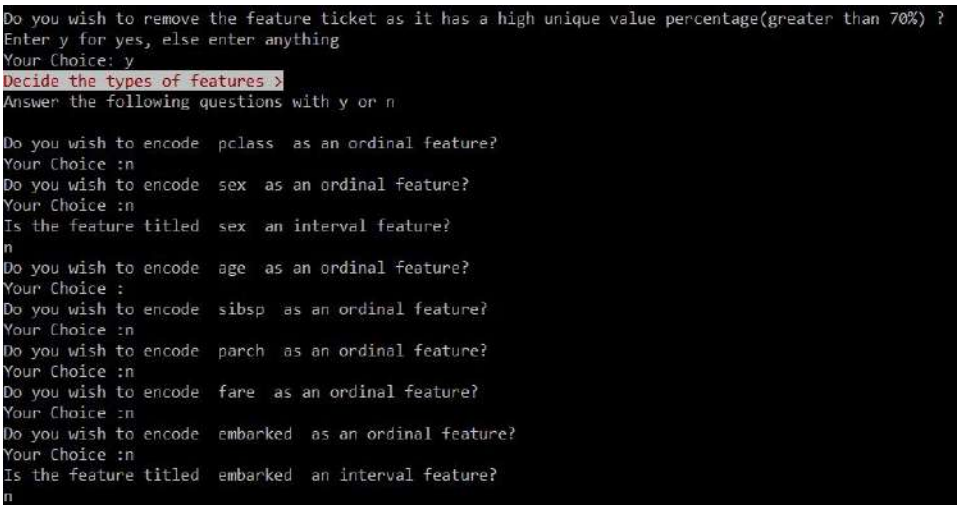



**Fig (5.15)** Interact with the user to choose feature types

- Interact with user to identify the features from which the user will like to remove outliers
- Normalize features as per user's requirement
- Currently, only one-hot encoding is performed for nominal features. Label encoding will be performed only for ordinal features(if any)

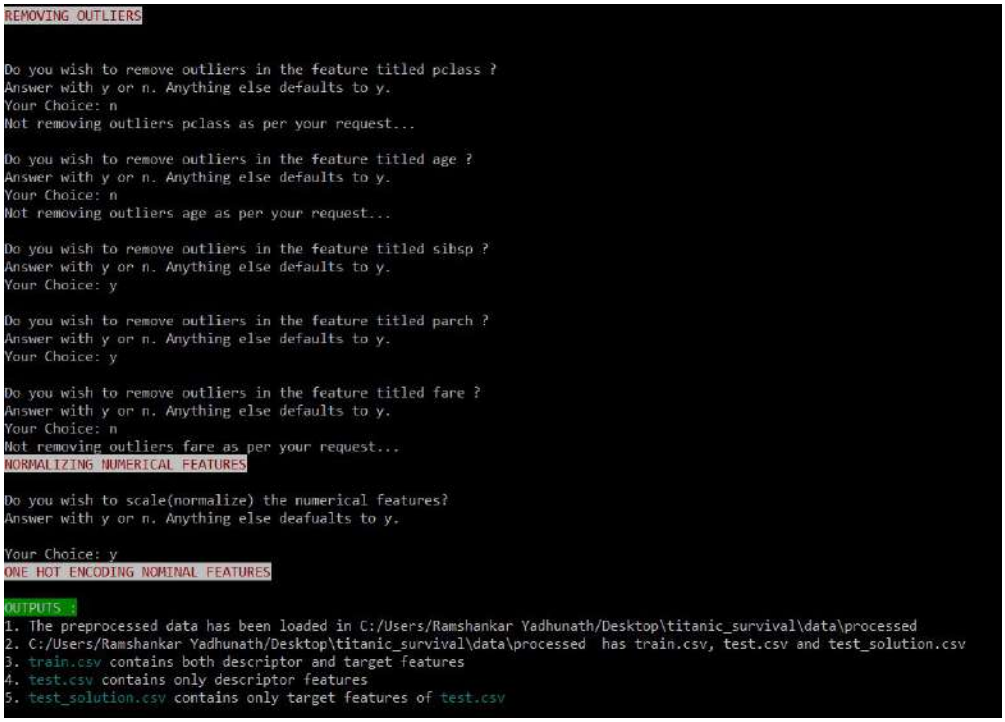



**Fig (5.16)** Other preprocessing steps with the help of user input

### 5.5.5 Visualize Data

DSK auto-generates visualizations based on the data it has. Currently, only density plots, box plots and count/frequency plots are generated.

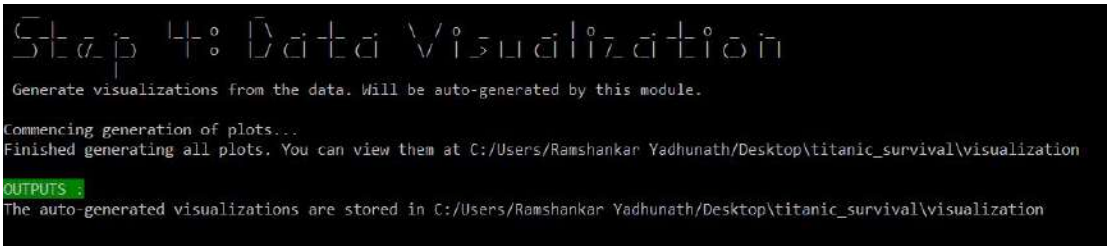



**Fig (5.17)** Data Visualization

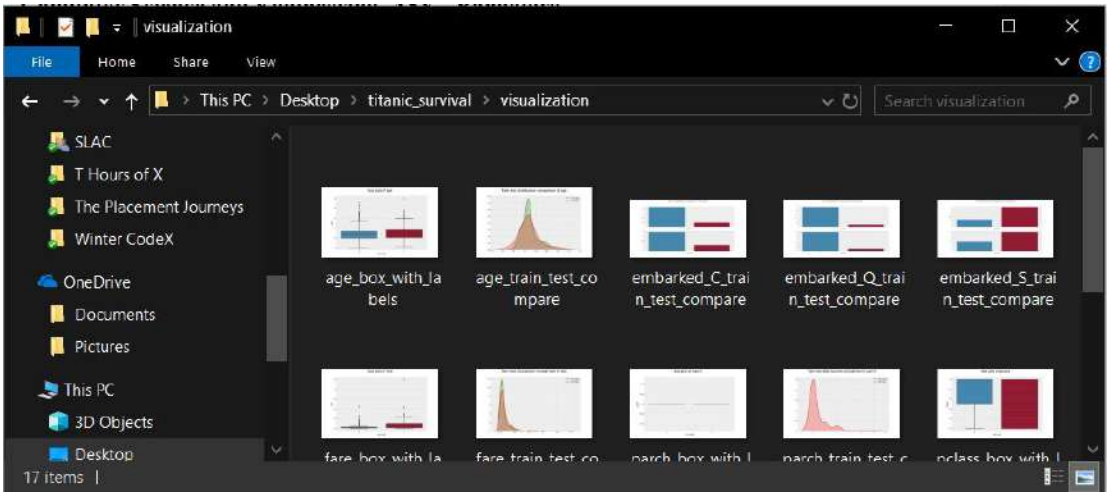



**Fig (5.18)** Auto-generated plots

### 5.5.6 Model Data

DSK performs supervised classification. It takes the preprocessed datasets and builds baseline models and evaluates the models using a 10-fold stratified cross validation strategy [52]. The baseline models are built on the following algorithms

- Logistic regression
- Decision Trees
- Random Forests
- K Nearest Neighbors
- Support Vector Machines

DSK first fits and evaluates on the train.csv file. After it produces the results for the same, the user is provided an option to choose which model he or she would like to store and use on the test.csv file. Based on the user's choice, a particular model estimator is saved in root/models.

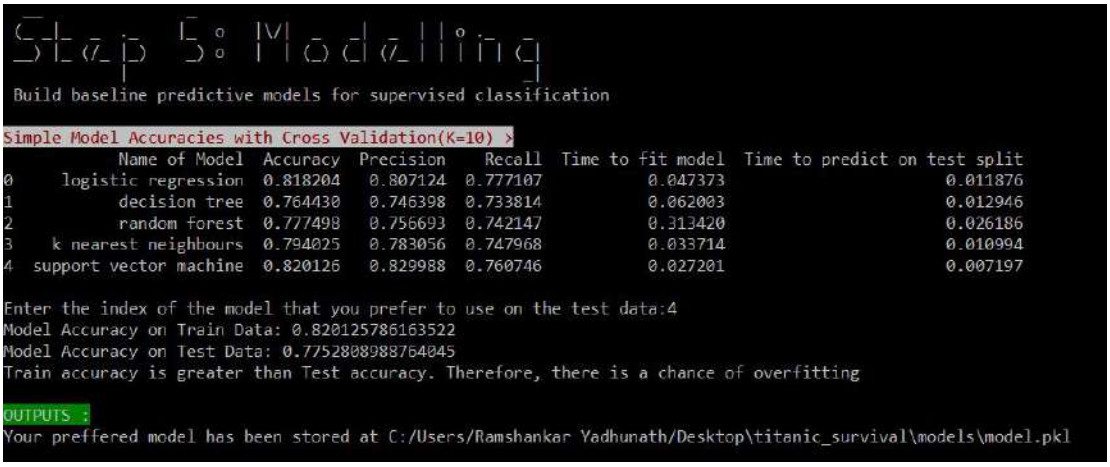



**Fig (5.19)** Modelling Results

The **Report results** block will be implemented as a part of this project's future work.

## 5.6 The devised algorithm for deep learning automation

The algorithm has been devised upon thorough inspection of the modelling step and automates the basic early steps of the same. The algorithm takes as input, the training set and the class labels that are given out by 'load_data.py', and the Bayes optimal error value for the given problem from the user.

### 5.6.1 Stepwise elucidation of the algorithm

1. The algorithm first trains the data using a single neuron. The trained function is the same as that of logistic regression. We have established empirically that shallow and deep neural networks perform better than logistic regression in all cases. Hence the algorithm will simply display the logistic regression evaluation metrics rather than compare and consider it for error analysis
2. The algorithm will then train a shallow neural network with the training data. This network will have just one hidden layer asserting to the condition that the

number of nodes is the same as the input dimension, rather than being initialized to an arbitrary constant value. This reason behind this architecture is the assumption that each attribute has a single line to contribute to the function given out by the model and we believe that this architecture will perform better than a constant baseline architecture. [56]

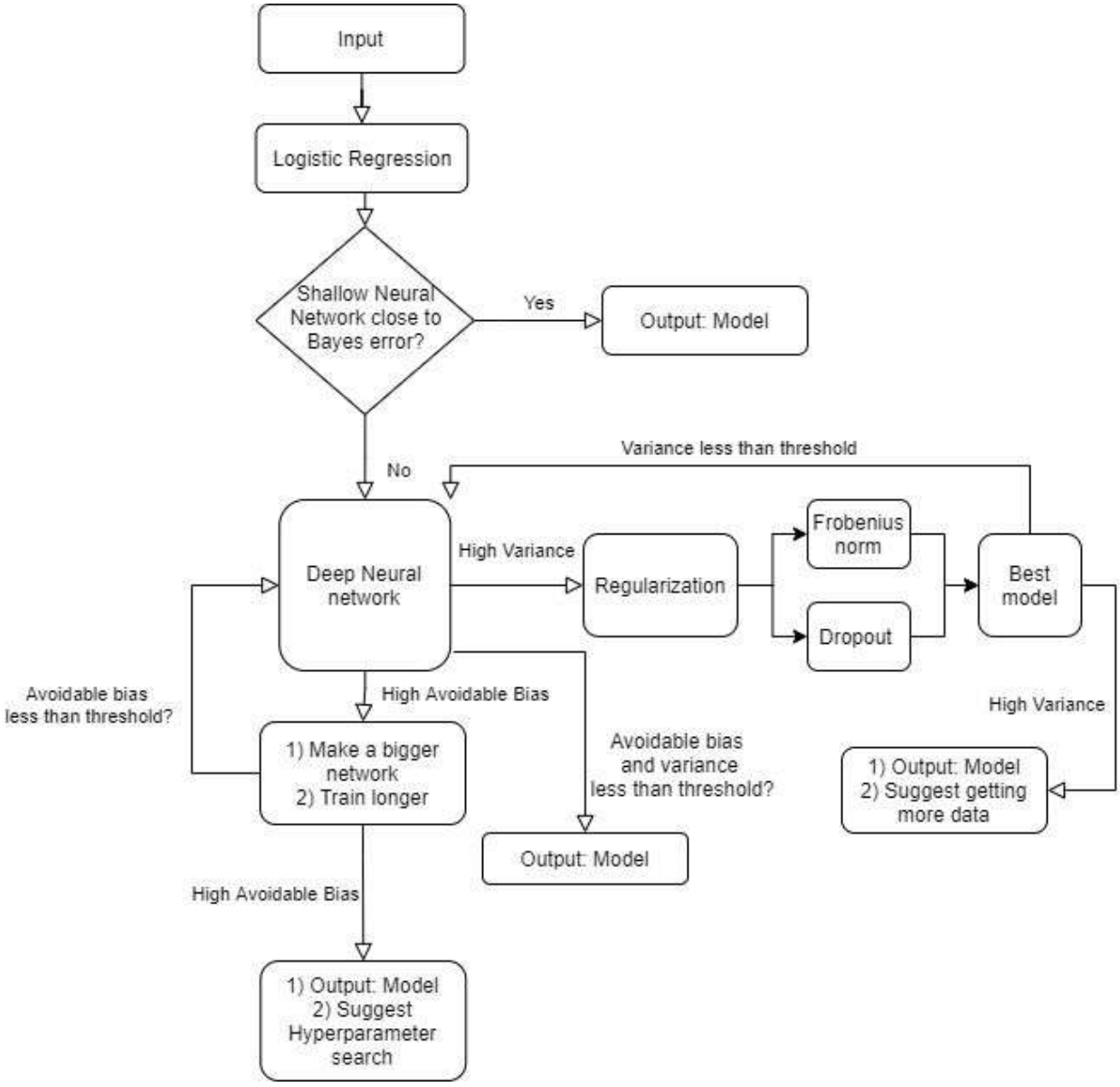



**Fig (5.20)** The devised deep learning automation algorithm

3. The number of epochs has been specified as seven hundred and the neural network is trained.
4. The validation accuracy and the training accuracy are then used to compute the variance and the avoidable bias
5. Now, error analysis is performed. (threshold value used is 3)
    a. Case 1:
        i. If the avoidable bias is found to be higher than the variance + threshold, then we train a deep neural network. Ideally, further increasing the epochs will help in getting better results. But pragmatically, this might result in overfitting the training set. Hence we keep it at 700.
        ii. Now, if the avoidable bias is less than the variance + threshold, the algorithm checks for a high variance and goes on to case 2 steps. If the variance is not high, then the model is given as output.
        iii. If the avoidable bias remains high, the algorithm returns the model and suggests a hyperparameter search

    b. Case 2:
        i. On the contrary, if the variance is found to be higher than the bias + threshold, then we use regularization to fix the overfitting issue. The algorithm will train two different deep neural networks, one with weight decay and the other with dropout and will return the model with the lowest variance.
        ii. If the variance is still found to be high, the algorithm suggests getting more data
        iii. Else, the algorithm moves on to the steps of Case 1

### 5.6.2 Algorithm implementation details

#### 5.6.2 a) Optimization

Adam optimizer is known to outperform other optimizers such as RMS prop and SGD in almost all scenarios. Thus, it has been used with all models in this work

Parameters:

1. Learning rate: 0.001
2. Beta1: 0.9
3. Beta2: 0.999
4. Epsilon:10e-8

#### 5.6.2 b) Activation functions used

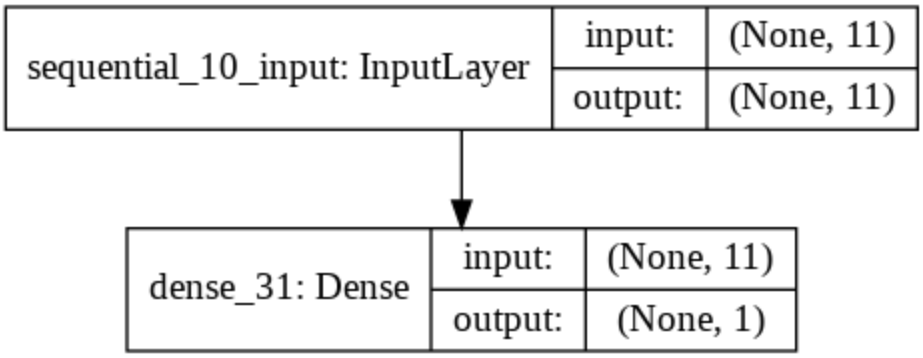
1. Sigmoid activation function has been used in Logistic Regression
2. In Deep and shallow Neural Networks, ReLU activation function has been used in hidden layers and Sigmoid in the output layer
3. Neural Networks for multi-class problems: ReLU activation function has been used in hidden layers and Softmax in the output layer. It is to be noted that the ReLU activation function has been used in all hidden layers as it is established to give better results than the other activation functions

#### 5.6.2 c) Regularization

1. In the neural network with Weight decay, L2 regularization has been applied with a regularization parameter of 0.2.
2. In the neural network with Dropout regularization, a Dropout probability of 0.2 is used

### 5.6.3 The neural network architectures used in the algorithm

### 5.6.3 a) Logistic Regression




```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_21 (Dense)             (None, 1)                 12
=================================================================
Total params: 12
Trainable params: 12
Non-trainable params: 0
```

**Fig (5.21)** Logistic Regression model architecture

### 5.6.3 b) Shallow neural network

It is to be noted that the number of neurons is the same as the number of input features

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_22 (Dense)             (None, 10)                120
_________________________________________________________________
dense_23 (Dense)             (None, 1)                 11
=================================================================
Total params: 131
Trainable params: 131
Non-trainable params: 0
```

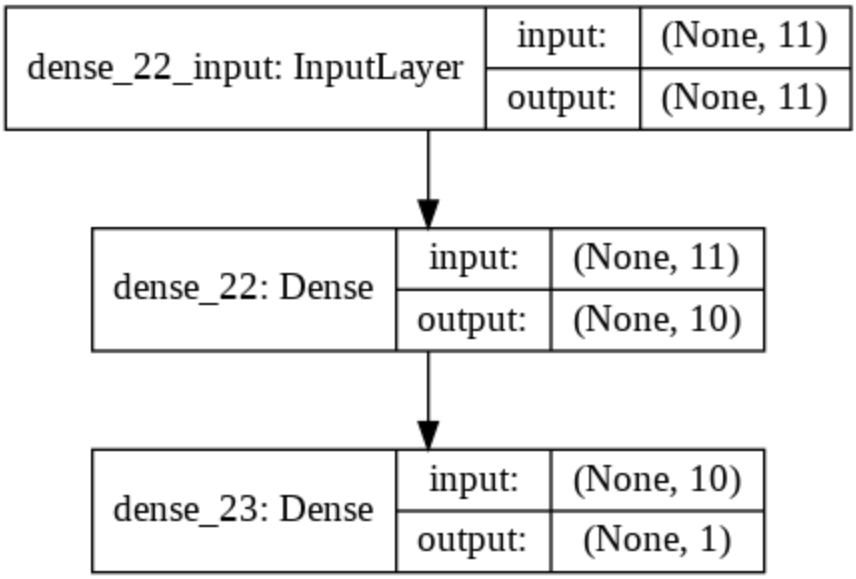



**Fig (5.22)** Shallow neural network architecture

### 5.6.3 c) Deep neural network

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_14 (Dense)             (None, 64)                768
_________________________________________________________________
dense_15 (Dense)             (None, 32)                2080
_________________________________________________________________
dense_16 (Dense)             (None, 16)                528
_________________________________________________________________
dense_17 (Dense)             (None, 8)                 136
_________________________________________________________________
dense_18 (Dense)             (None, 4)                 36
_________________________________________________________________
dense_19 (Dense)             (None, 2)                 10
_________________________________________________________________
dense_20 (Dense)             (None, 1)                 3
=================================================================
Total params: 3,561
Trainable params: 3,561
Non-trainable params: 0
```

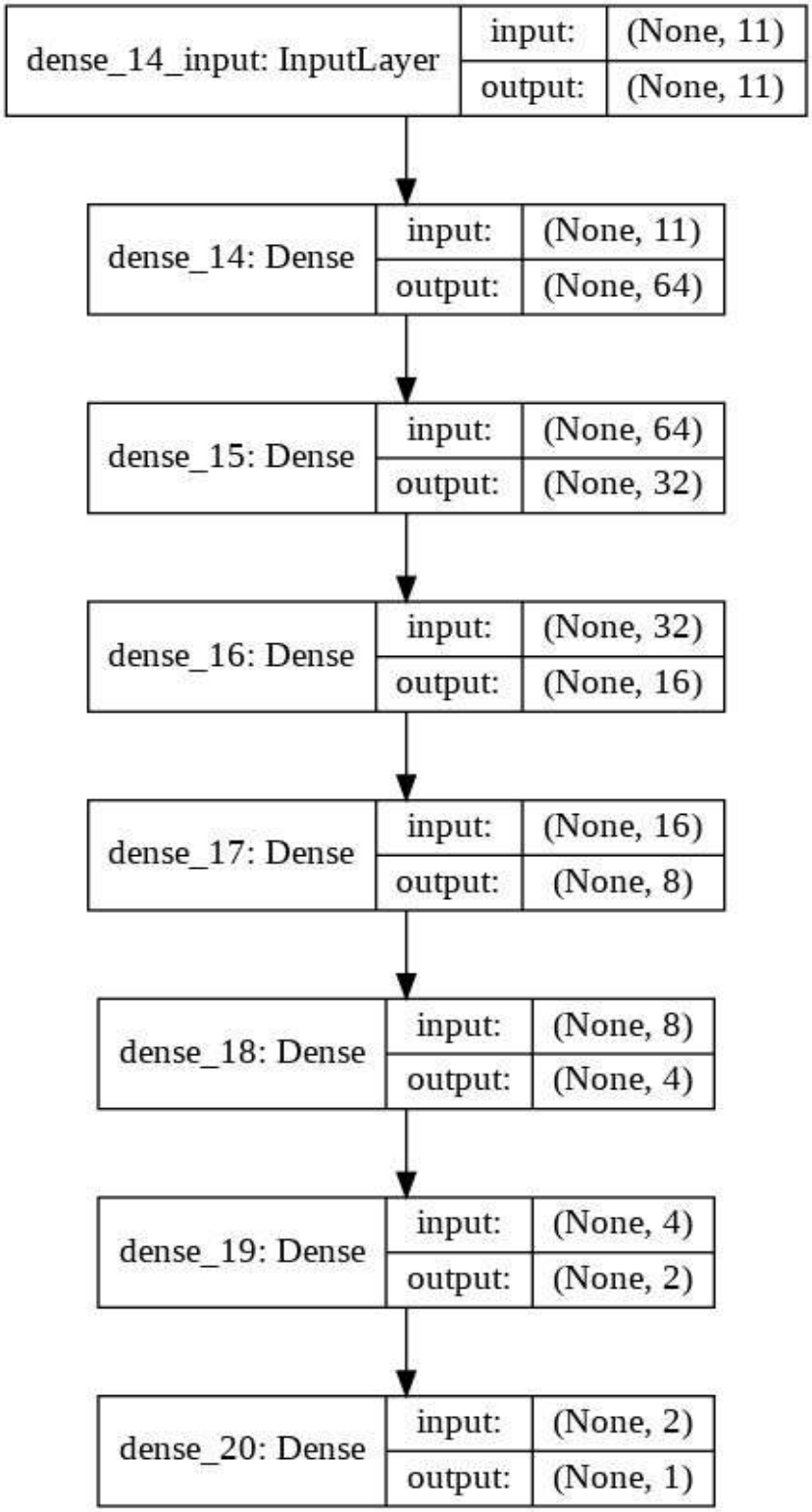



**Fig (5.23)** Deep neural network architecture

# 6. IMPLEMENTATION

The focus of this chapter is to provide a high-level overview of the flow of control in DSK. Fig(5.3) depicts the 8 main building blocks of DSK and in this chapter, we shall be elaborating on how each of these blocks are implemented with programming. The program for DSK has been entirely written in Python 3.7.4 and includes the use of libraries such as numpy, matplotlib, seaborn, pandas, scikit_learn, tabulate, colorama, termcolor, keras and tensorflow(the last two specifically dealing with the deep learning architecture included).

## 6.1 Flow of Control

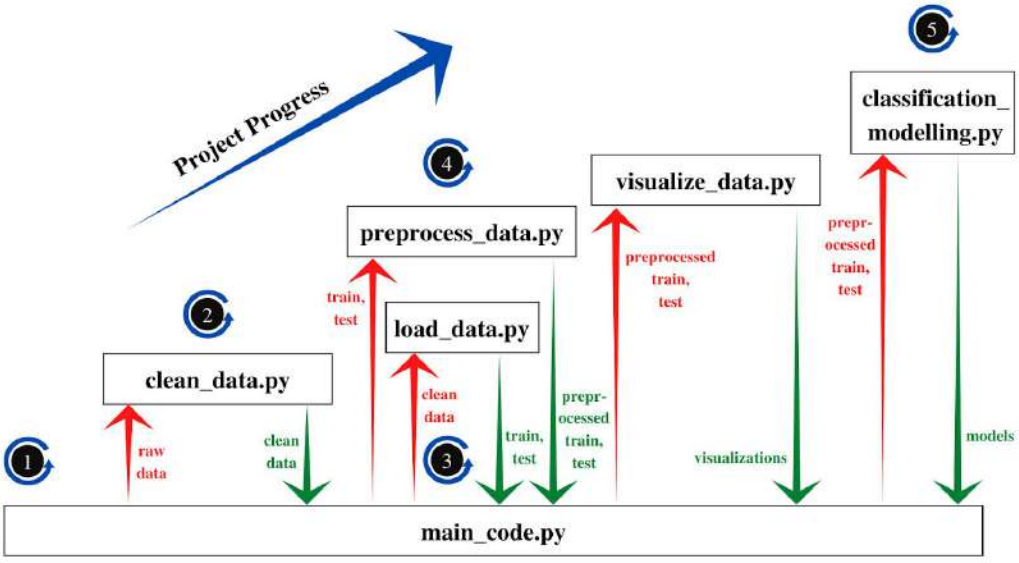



**Fig (6.1)** Flow of control through DSK

Fig(6.1) provides an overview of how the internal components of DSK transfer control among each other to provide output based on user input.

There are 6 main python files that are responsible for the operation of DSK(current version). These files are namely main_code.py, clean_data.py, load_data.py, preprocess_data.py, visualize_data.py and classification_modelling.py. These are represented as rectangular blocks in the diagram. The circular symbols with circular arrows at the edges represent user-interaction sets. A user-interaction set contains functionalities by virtue of which DSK asks questions to the user and performs actions based on the user's response. Altogether, DSK employs 5 sets of user-interactions:

- **Set 1:** User is asked to choose a directory from the local system and also asked to load data
- **Set 2:** Questions regarding data cleaning operations
- **Set 3:** User is asked to enter the "target" feature
- **Set 4:** Questions regarding data preprocessing operations
- **Set 5:** User is asked which model is to be saved

## 6.2 DataSwissKnife is built with Object Oriented Programming paradigms

Object Oriented Programming(OOP) refers to a computing paradigm that encourages the design of programs as "objects", that contain data and the associated "methods" or functions to manipulate these objects [53]. DSK is built on this idea and employs the uses of class-based OOP to encompass similar functions within a common class. While main_code.py is procedural in nature, the other 4 .py files are designed as aggregations of several classes, most of which inherit properties from other classes within the same code file.

A pressing need to use OOP in the design of DSK was because of DSK's nature to manipulate entire datasets. A procedural approach here would have induced the much-less desirable effect of clutter. Procedural execution is also undesirable for the given system as it could create complex scenarios such as having to explicitly maintain states of multiple variables.

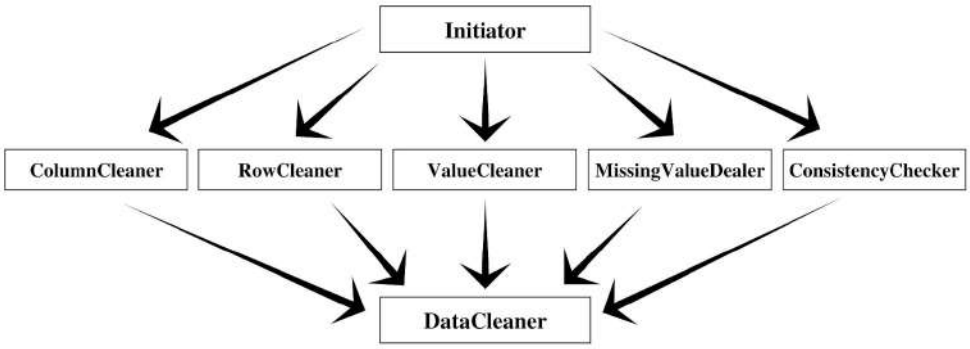



**Fig (6.2)** Class Hierarchy in clean_data.py

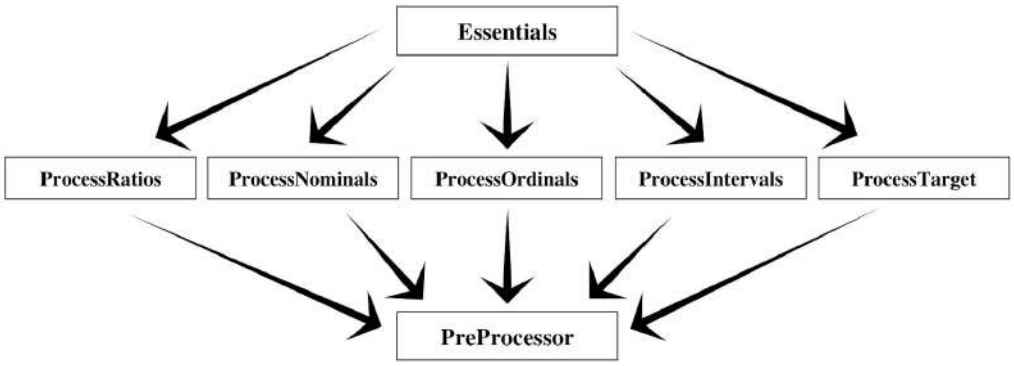



**Fig (6.3)** Class Hierarchy in preprocess_data.py

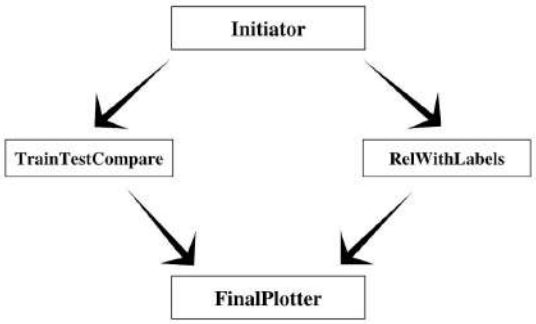



**Fig (6.4)** Class Hierarchy in visualize_data.py

## 6.3 Links to Repository

Github Repository: https://github.com/ry05/dataswissknife

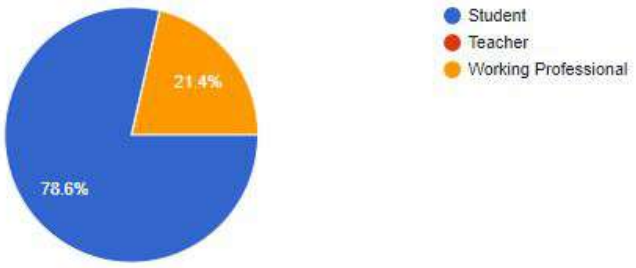
# 7.TESTING

Testing is indeed a vital part in the development journey because it tells us where we have done well and where we have not and also shows us where we can improve. A major part of testing has been done by ourselves, the DataSwissKnife team, but we found it necessary that we share it across to our friends and colleagues who are students and working professionals and compile their insights helping us improve the future iterations of DataSwissKnife. We call this a “Community Testing” approach.

For this , we created a google form with valid questions and shared it across the groups at ASE-B. We received 14 responses and in the remainder of this chapter, shall display the results of these responses.

## 7.1 Who the responders were

A majority of the responses were given by students, which were more detailed. This shows that students are more intrigued about this than working professionals.



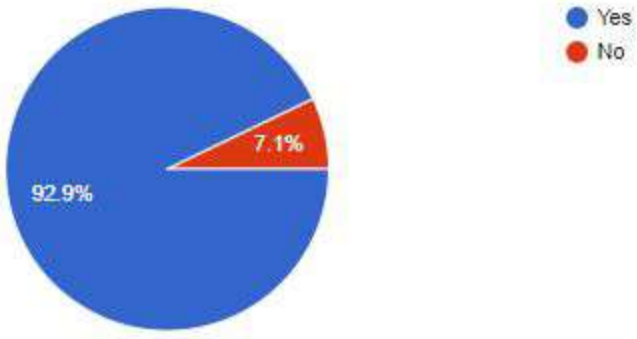

**Fig(7.1)** Number of students and working professionals providing feedback

A good number of our responses were provided by individuals with little to intermediate level of exposure to data science which is understandable because most of them were students, which points out the need for this tool.


**Fig(7.2)** Exposure or experience in data science a scale of 1-10

## 7.2 Did the respondents need a no-code tool for data science

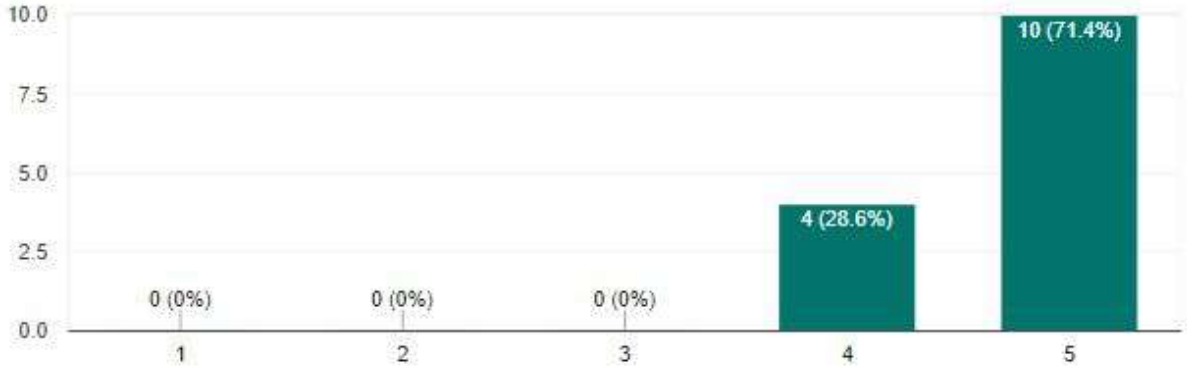
A very good question which can be asked is , do I need a tool to do my preliminary operations? This question has been answered and it can be understood that this tool can indeed make the lives of data science enthusiasts easier .




**Fig(7.3)** Did users ever want a tool to perform preliminary data operations?

## 7.3 How likeable was the idea behind the tool

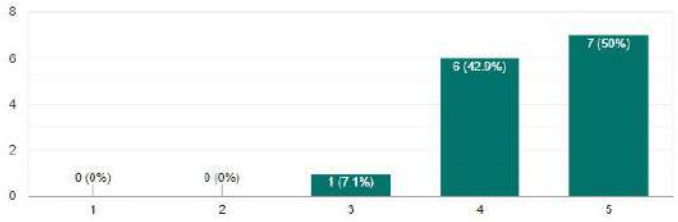
The tool can be successful only if our users and other data enthusiasts like the idea behind this tool and respect it. We are humans and we use things that we think are respectful and have some usefulness to us.On a scale of 1-5, this is how much our users liked the tool(Fig 7.4).




**Fig(7.4)** How much our users liked the idea behind the tool on a scale of 1-5

## 7.4 How easy the tool was to use

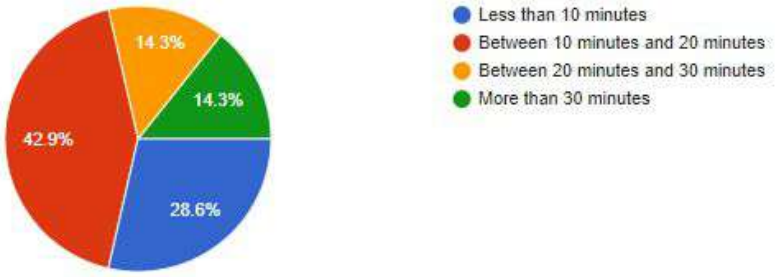
An important part of deployment is how easy it is to use the tool because at the end of the day, the purpose of this tool is to make the lives of data science enthusiasts more comfortable.Fig(7.5) shows how much the users felt comfortable with the tool.




**Fig(7.5)** How easy it was for the users to use the tool on a scale of 1-5

## 7.5 Time to use the tool

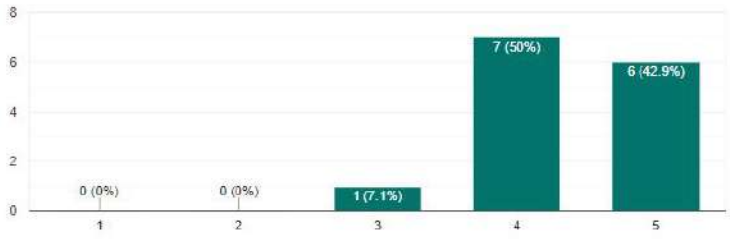
An important consideration to use this tool was the time it saved for the user. Fig(7.6) shows how much time the tool took to complete their tasks.




**Fig(7.6)** Time taken to run the tool on the user's system

## 7.6 How significant do the respondents feel the tool will be in the future

The success of something is the trust that the users have in it.Fig(7.7) shows how much the users think it will be important in the future to perform preliminary data science operations.




**Fig(7.7)** Significance in the future on a scale of 1-5

# 8. CONCLUSION, KEY TAKEAWAYS AND FUTURE SCOPE

## 8.1 Conclusion and Key Takeaways

Data science, now a ubiquitous field currently finds employment and usage in a lot of fields and industries that are not directly based on computer science or statistics. Data science as a discipline is mercurial, flexible, integrated and widespread in its use. The most significant use of data science has been to propel businesses. However, in order to even create the most basic analysis, writing code becomes an important step and is often time consuming. This project has been a journey into identifying a simpler alternative to generating preliminary data science results by doing away with the user's requirement to code. Through the project, it has been proven that this idea of automating data cleaning, though skeptical can be achieved. Also, considering the ease with which our test subjects were able to use the DataSwissKnife tool, it has become rather clear that the tool offers users a new way to get into data science or work with data science quickly. This is a tool which can be used by Industrialists looking to optimize their production line efficiency but don't have time to learn the nitty gritties of Data Science and AI or this can be used by engineering students who are knocking on the doors of a career in Data Science or this can be used by professionals looking to test their existing dashboard.

The future prospect for this tool looks bright, though there is some skepticism surrounding the idea of abolishing code to do data science. However, there is always a new concept that replaces an older one. In spite of this conflicting idea, DataSwissKnife is built on our fundamental belief that if information transitions into

the next battleground(in case it already has not), then the ability to code must not be the deterrent to an organization's progress, nor be the inhibitor to an individual's transition into the data field.

## 8.2 Future Scope

This project is one that has grown from an idea that initially seemed rather far-fetched, but now is a full fledged working prototype with important use cases in many fields now and in the years to come. We understand that what we have now is a very user friendly minimum viable product which has effective use cases to different strata of Data Scientists or Data Science enthusiasts or just ambitious individuals looking to implement Data Science and AI.

A few of the many future prospects or additions to this project can be:

8.1.1. A GUI version/ Web interface to make it more dynamic and scalable.

8.1.2. Dockerizing it so that it can be run within a docker container and specified environment ensuring security and smooth requirement handling.

8.1.3. Ability to store pipelines with models and a log file

8.1.4. More complex visualizations.

8.1.5. Better documentation using pre existing tools like readthedocs.io and the likes.

8.1.6. Promote the product and ensure it caters to a broader range of users

In the next version of DSK, or as we call it, version 1.0, the team will try to include a subset of these highly relevant features to the product. The feedback and suggestions provided by the community during the "Community Testing" phase will also be carefully scrutinized in order to improve the product's user compatibility.

# 9. REFERENCES


1. Provost, Foster, and Tom Fawcett. "Data science and its relationship to big data and data-driven decision making." Big data 1.1 (2013): 51-59.
2. Dhar, Vasant. "Data science and prediction." Communications of the ACM 56.12 (2013): 64-73.
3. Hayashi, Chikio. "What is data science? Fundamental concepts and a heuristic example." Data science, classification, and related methods. Springer, Tokyo, 1998. 40-51.
4. Wing JM. The Data Life Cycle. Harvard Data Science Review [Internet]. 2019 Jul 1;1(1). Available from: https://hdsr.mitpress.mit.edu/pub/577rq08d
5. Chamchoun,Yasmin. "Should Data Science be considered as its own discipline?". The Data Scientist. 17 July. 2019: https://thedatascientist.com/data-science-considered-own-discipline/
6. Davenport, Thomas, and Jeanne Harris. Competing on analytics: Updated, with a new introduction: The new science of winning. Harvard Business Press, 2017.
7. Irizarry RA. The Role of Academia in Data Science Education . Harvard Data Science Review [Internet]. 2020 Jan 31;2(1). Available from: https://hdsr.mitpress.mit.edu/pub/gg6swfqh
8. Cao, Longbing. "Data science: challenges and directions." Communications of the ACM 60.8 (2017): 59-68.
9. Cai, Li, and Yangyong Zhu. "The challenges of data quality and data quality assessment in the big data era." Data science journal 14 (2015).
10. Dasu, Tamraparni, and Theodore Johnson. Exploratory data mining and data cleaning. Vol. 479. John Wiley & Sons, 2003.
11. Strong, Diane M., Yang W. Lee, and Richard Y. Wang. "Data quality in context." Communications of the ACM 40.5 (1997): 103-110

12. Rahm, Erhard, and Hong Hai Do. "Data cleaning: Problems and current approaches." IEEE Data Eng. Bull. 23.4 (2000): 3-13.
13. Tierney, Nicholas J., and Dianne H. Cook. "Expanding tidy data principles to facilitate missing data exploration, visualization and assessment of imputations." arXiv preprint arXiv:1809.02264 (2018).
14. Wickham, Hadley. "Tidy data." Journal of Statistical Software 59.10 (2014): 1-23.
15. García, Salvador, Julián Luengo, and Francisco Herrera. Data preprocessing in data mining. Cham, Switzerland: Springer International Publishing, 2015.
16. Ramírez-Gallego, Sergio, et al. "A survey on data preprocessing for data stream mining: Current status and future directions." Neurocomputing 239 (2017): 39-57.
17. Famili, A., et al. "Data preprocessing and intelligent data analysis." Intelligent data analysis 1.1 (1997): 3-23.
18. García, Salvador, Julián Luengo, and Francisco Herrera. "Tutorial on practical tips of the most influential data preprocessing algorithms in data mining." Knowledge-Based Systems 98 (2016): 1-29.
19. Friendly, Michael. "A brief history of data visualization." Handbook of data visualization. Springer, Berlin, Heidelberg, 2008. 15-56.
20. Cleveland, William S. Visualizing data. Hobart Press, 1993.
21. Tukey, John W. Exploratory data analysis. Vol. 2. 1977.
22. Aparicio, Manuela, and Carlos J. Costa. "Data visualization." Communication design quarterly review 3.1 (2015): 7-11.
23. Segel, Edward, and Jeffrey Heer. "Narrative visualization: Telling stories with data." IEEE transactions on visualization and computer graphics 16.6 (2010): 1139-1148.
24. Hutter F, Caruana R, Bardenet R, Bilenko M, Guyon I, Kegl B, and Larochelle H. "AutoML 2014 @ ICML". AutoML 2014 Workshop @ ICML. Retrieved 2018-03-28.
25. Thornton, Chris, et al. "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms." Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. 2013.
26. Feurer, Matthias, et al. "Auto-sklearn: efficient and robust automated machine learning." Automated Machine Learning. Springer, Cham, 2019. 113-134.

27. Samulowitz, Horst, A. Sabharwal, and C. Reddy. "Cognitive automation of data science." ICML AutoML workshop. 2014.
28. OECD Glossary of Statistical Terms. OECD. 2008.
29. Wikipedia contributors. "Data set." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 17 Feb. 2020. Web. 7 May. 2020.
30. Bishop, Christopher (2006). Pattern recognition and machine learning. Berlin: Springer.
31. Available Online: http://robotics.stanford.edu/~ronnyk/glossary.html
32. Wikipedia contributors. "Domain knowledge." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 26 Sep. 2019. Web. 7 May. 2020.
33. Available Online: https://www.linkedin.com/pulse/role-domain-knowledge-data-science-patrick-bangert/
34. Available Online: https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4
35. Stevens, Stanley Smith. "On the theory of scales of measurement." (1946): 677-680.
36. Wikipedia contributors. "Level of measurement." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 7 May. 2020. Web. 8 May. 2020.
37. Kotsiantis, S. B., Dimitris Kanellopoulos, and P. E. Pintelas. "Data preprocessing for supervised learning." International Journal of Computer Science 1.2 (2006): 111-117.
38. Hawkins, Douglas M. Identification of outliers. Vol. 11. London: Chapman and Hall, 1980.
39. Available Online: https://statistics.laerd.com/statistical-guides/pearson-correlation-coefficient-statistical-guide-2.php
40. Available Online: https://www.socscistatistics.com/tests/pearson/
41. Available Online: https://www.statisticshowto.com/normalized/
42. Wikipedia contributors. "Normal distribution." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 29 Apr. 2020. Web. 9 May. 2020.
43. Available Online: https://pbpython.com/categorical-encoding.html

44. Available Online: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/
45. Available Online: https://www.statisticshowto.com/dimensionality/
46. Viswanath, Bimal, et al. "On the evolution of user interaction in facebook." Proceedings of the 2nd ACM workshop on Online social networks. 2009.
47. Wikipedia contributors. "Data visualization." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 1 May. 2020. Web. 20 May. 2020.
48. Aparicio, Manuela, and Carlos J. Costa. "Data visualization." Communication design quarterly review 3.1 (2015): 7-11.
49. Available Online: https://www.data-to-viz.com/
50. Available Online: https://www.kaggle.com/c/titanic/data
51. Wikipedia contributors. "Agile software development." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 7 May. 2020. Web. 11 May. 2020.
52. Available Online: https://scikit-learn.org/stable/modules/cross_validation.html
53. Wikipedia contributors. "Object-oriented programming." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 20 May. 2020. Web. 2 Jun. 2020.
54. Wikipedia contributors. "Supervised Learning." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 12 May. 2020. Available Online: https://en.wikipedia.org/wiki/Supervised_learning
55. Wikipedia contributors. "Bayes error rate." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 30 May. 2020. Available Online: https://en.wikipedia.org/wiki/Bayes_error_rate
56. Available Online: https://heartbeat.fritz.ai/selecting-the-best-architecture-for-artificial-neural-networks-7b051f775b4
57. J. M. Kanter and K. Veeramachaneni, "Deep feature synthesis: Towards automating data science endeavors," 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Paris, 2015, pp. 1-10, doi: 10.1109/DSAA.2015.7344858.