iVLAIR - Interface Design and Prototype for Interactive Visualization-Mediated Supervised Classification

by

Sanchitha Kuppusami
B.Tech., Anna University, India, 2013

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

iVLAIR - Interface Design and Prototype for Interactive Visualization-Mediated Supervised Classification

by

Sanchitha Kuppusami
B.Tech., Anna University, India, 2013

Supervisory Committee

Dr. Miguel Nacenta, Supervisor
(Department of Computer Science)

Dr. Brandon Haworth, Departmental Member
(Department of Computer Science)

ABSTRACT

Machine learning is widely used in various applications because of its advantages, but not everyone can apply it effectively as it requires the use of textual programming. To overcome this, there are several approaches that help users perform machine learning classification without writing code. However, for most machine learning models, it is difficult to understand how they arrive at a particular result. This challenge has triggered a lot of research on interpretable ML methods. However, these methods also require the user to learn how to code and implement them. In this work we introduce iVLAIR, a web application tool that eases machine learning classification for a wider audience and makes data more understandable to the users by transforming the data into visualizations, thereby improving the model interpretability. We also conduct an evaluation with machine learning experts and non-experts to compare iVLAIR with the python approach for performing machine learning classification.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

my spouse, Richard and our kid, Niralya for always supporting and accompanying me through this journey.

my parents, Kuppusami and Dhanalakshmi for always encouraging me.

my supervisory committee, Miguel and Xiyao for the mentorship, support, encouragement, and patience.

DEDICATION

This work is dedicated to the research community that advances the field of machine learning for beginners.

Chapter 1

Introduction

Machine learning is defined as the field of study, that gives computers the ability to learn from data without being specifically programmed to do so [66]. Because of its advantages, it is widely used in applications such as healthcare [19], finance [32], retail [38] and much more which in turn attracts users from a variety of fields. However, to most effectively utilize, generate, and execute machine learning models and algorithms, fundamental programming skills are necessary. Therefore, users without programming backgrounds may find it difficult to use machine learning algorithms and may have to depend on data scientists to use them effectively.

To overcome this challenge, there has been much research in no-code machine learning approach to build ML models without writing code. It allows technical and non-technical individuals to leverage ML techniques and create predictive models using easy-to-use graphical interfaces. With no code ML, users can choose algorithms, train models, and make predictions without the help of technical experts. This makes ML accessible to a wider audience by leveraging the power of ML without the need for extensive coding knowledge.

However, machine learning algorithms are non-intelligible and cannot explain why they arrived at a result. This can lead to a lack of trust in the algorithm and make it difficult to debug when it is not working correctly. Therefore, it is necessary to understand why the model has made a particular prediction. Making machine learning more accessible to non-programmers and more intelligible are both important because it can help increase the adoption of machine learning and make it more accessible to people who may not have a technical background. Additionally, it can help increase the transparency of the algorithm and make it easier to debug when it is not working correctly.

Models such as decision trees [26, 16] are called white-box models because they do not require an additional model to provide explanations [48]. The results of these models are easy to understand by experts in the application domain. On the other hand, models such as SVM [22] and deep learning algorithms [44] act as black-box models because they are very hard to explain and to be understood by experts in practical domains. There are several model-specific and model-agnostic interpretability methods to explain the black-box models [1]. These approaches also involve the use of some popular visualization techniques to explain why the model has reached a particular prediction by explaining the important features and how those features are related to each other. This is because visualizations are a popular way of representing data visually to help identify patterns in data that are not immediately apparent from raw data. However, these approaches require programming knowledge to implement them.

To address these issues, we propose the design and implementation of iVLAIR, a web application tool that helps non-programmers perform machine learning classification with an easy-to-use user interface. Our tool is designed on top of the VLAIR [36, 37] approach of using visualizations and deep learning techniques to solve classification problems. Here, we transform the raw data into visualizations using an in-built visualization tool and then use an integrated computer vision-based deep learning algorithm. The iVLAIR interface is designed to perform the entire machine learning workflow from data preparation to evaluation. It includes features such as data preparation, feature engineering, data transformation, training, testing, and evaluation. The inbuilt visualization tool is inspired by constructive visualization approach [34] and the design is adopted from iVOLVER [51]. Our visualization tool is designed to have two different canvases to separate the main visualization from its background operations.

In this thesis, we present the following contributions:

1 The design of a prototype tool, iVLAIR, that facilitates machine learning classification for a broader audience based on the VLAIR (Visualization as intermediate representation) approach.

2 New interaction techniques that help users create visual representations of data and make the data more accessible to humans, thereby improving the interpretability of machine learning models.

3 Empirical evaluation results that offer insights into how experts and non-experts

think about theiVLAIR tool compared to the Python approach for performing machine learning classification.

The rest of the thesis is organized as follows: Chapter 2 provides background and related research that motivated and guided the design of iVLAIR and establishes a foundation for the design and development of iVLAIR. Chapter 3 describes the overarching design goals and principles and how they were applied to the design process. Chapters 4 and 5 describe how iVLAIR has been designed with features that align with the design principles we propose. Chapter 6 describes a scenario that shows how the overall system is used. Chapters 7 and 8 present the evaluation methodology and its results. Chapter 8 includes a discussion on the approaches that form the base of our system and design, as well as limitations of our design choices, open questions, and the uniqueness of our tool compared with other approaches. Finally, it discusses technical constraints and future work. Chapter 9 concludes the dissertation by summarizing the findings and contributions.

Chapter 2

Background and related work

iVLAIR is a web application tool which is designed to help non-programmers to perform machine learning classification without the need of textual programming. This is an alternative method to using traditional machine algorithms to solve classification problems. It employs existing techniques in computer vision [74] and information visualization [50, 15]. In the subsections below, we summarize the four areas of work that are particularly relevant: traditional approaches to ML classification, Interpretability in machine learning, VLAIR and Data visualization.

2.1 Traditional approaches to machine learning classification

Data classification [31] is the process of grouping data into different categories based on their features. For instance, in the iris dataset [1], there are three different species (versicolor, virginica or setosa) of iris flowers and each iris flower has different measurements such as sepal length, sepal width, petal length, and petal width. Data classification is used to classify the flowers into different species based on their measurements.

Supervised classification algorithms are used to solve classification problems by categorizing one or multiple classes based on one or more features. The first step is to use the training data to train the model. The training dataset has features and a target variable, which is the label that we are going to predict using the machine learning algorithm. During the training process, the machine learning algorithm

[1] https://www.kaggle.com/datasets/uciml/iris

learns to identify the relationship between the features and the labels and use that knowledge to make predictions or decisions about new data, even if it has not seen it before. For example, we assign a label to each iris flower as either versicolor, virginica or setosa based on what species that flower belongs to. We can train the algorithm using this labeled data. The algorithm learns the features that distinguish the classes from each other and uses those features to categorize new data into one of the three classes for any unseen data.

Various algorithms have been applied to classification tasks. Some of these algorithms are:

1. Decision trees algorithm [26] is a method that uses a hierarchical tree structure to make decisions based on features and outcomes. The tree has a root node (on top), branches and leaf nodes. The root node and the interior nodes represent features, the branches represent the decisions made on those features, and the leaf nodes represent the final outcomes. The algorithm starts from the root node and splits the dataset into subsets according to the best feature for that node. This process is repeated recursively for each subset until no more splitting is possible or all the data in a subset belong to the same class. Then, the subset is assigned to a leaf node with the corresponding class label. To measure the quality of the splitting, the algorithm uses criteria such as Gini impurity, entropy and information gain. Pruning is a technique to reduce model complexity and to prevent overfitting. Pruning can be done in two ways: Pre-pruning is used to prevent further splitting, while post-pruning removes unnecessary nodes from the learned tree.

2. Random forest classifier algorithm [12] is a collection of decision trees where each tree is built on a random sampling of the original dataset. The new data will be classified by each of the decision trees and a majority vote will be taken and the new dataset will be classified based on that.

3. Support vector machines (SVM) [22] create a line or a hyperplane which separates the data into classes. The best hyperplane that maximizes the margin between the two classes will be chosen. The margin is defined as the distance between the hyperplane and the closest data points (support vectors) from each class. The best line or hyperplane is the decision boundary and new data that falls on one side of the class is classified as that group of the class whereas the data is classified as other group if it falls on the other side.

4. K-Nearest Neighbour [17, 8] is mostly used to classify a data point based on how its neighbours are classified. It is based on the assumption that similar data points can be found near one another. This algorithm is used to compare new data with labeled training data and find K most similar data (the nearest neighbours) and look at their labels. For example, if K is 1, we use only the closest neighbour to determine the category. If K is 10, we use the 10 closest neighbours. In order to determine which data points are closest to a given query point, the distance between the query point and the other data points will need to be calculated. A popular method to calculate the distance is using Euclidean distance method. The new data point is then assigned to the category that has the majority of votes among the K nearest neighbours.

2.2 Computer Vision

Computer vision [74] is a field of study that aims to automatically extract, analyze and understand meaningful information from visual data, such as images, videos and other visual inputs. It tries to imitate human vision, but it requires a large amount of data and iterative analysis to finds the differences and patterns in the images. For example, to perform face recognition, a computer needs to receive a large amount of images of faces and facial features to learn how to distinguish and identify different faces.

Machine learning algorithms use mathematical models to analyze images without coding every single rule, unlike the manual approaches to computer vision that require a lot of coding. Traditional machine learning techniques uses feature extraction techniques such as SIFT (Scale-Invariant Feature Transform), SURF (Speeded-Up Robust Features), BRIEF (Binary Robust Independent Elementary Features), etc. to identify features such as corners, colorschemes, texture of image, etc. and use those features as input to the algorithms such as SVM, Random Forest algorithms, etc. for prediction [57, 28, 45].

Another method is using deep learning algorithms based on neural networks. These algorithms work by automatically identifying features from a large number of labeled images and then use that knowledge to classify new images. Deep learning algorithms are widely used because of its ability to produce more accurate results [81, 43]. A common architectures within computer vision pipelines is multilayered Convolutional Neural Network (CNNs). CNNs are useful for image classification be-

cause they can recognize the same features in different locations of an image using a shared-weight architecture that is based on their shift in-variance characteristics [73].

VGG16 [72, 75, 76] is a convolutional neural network model that is useful for image classification because it has a deep architecture with 16 layers that have weights. In our prototype tool, iVLAIR, VGG16 is used as an example model which was trained a few years ago on the ImageNet datasets [20]. This allows it to achieve high accuracy even for smaller datasets. We used the VGG16 algorithm because it was available at the initial time. However, there are more advanced algorithms available now that could be used instead of VGG16.

2.3 Interpretability methods to explain ML Models

There is a growing concern in the fields of Artificial Intelligence, Machine Learning and Deep Learning that much of how these technologies work is opaque to human understanding. For instance, complex neural network models are used to analyze medical images such as X-rays, CT scans, and MRIs to help doctors identify diseases such as cancer with high accuracy [3]. In such cases, it is important that these algorithms are explainable so that doctors can understand the reasoning behind the diagnosis and make better decisions.

The interchangeable use of the terms interpretability and explainability in the literature is one of the issues that hinders the establishment of mutual understanding. Both interpretability and explainability refers to the understanding of a machine learning model. Interpretability allows us to create a hypothesis that describes the relationship between the input data and the output, while explainability is the understanding of how the system reached the conclusion of the algorithm. These concepts are closely related but distinct. Below are the definitions:

- *"**Interpretability** is mostly associated with the intuition behind the outputs of a model. The idea is that the more interpretable a machine learning system is, the easier it is to identify cause-and-effect relationships within the system's inputs and outputs"* [46].

- *"**Explainability** , is associated with the internal logic and mechanics that are inside a machine learning system. The more explainable a model, the deeper*

the understanding that humans achieve in terms of the internal procedures that take place while the model is training or making decisions" [46].

Most machine learning models act as black-box models because they are so complex that humans do not understand how they make predictions [48]. They are designed to take in data and produce an output without revealing how they arrived at that output. Some people argue that it is necessary for algorithms to explain their results to humans, especially in applications such as healthcare, defense, law and order, etc., where explaining how an answer was obtained is as important as obtaining the answer [30]. According to [83], this helps end-users understand the model and its outcomes and also helps them predict what it might do in unforeseen situations. This provides transparency and builds trust, encouraging users to use the system. Machine learning experts take advantage of this approach to understand how the trained model works, which helps them debug what the model has learned and identify any bias or issues with the model. It also gives experts the ability to audit a prediction or decision trail in detail and establish legal liability, particularly if something goes wrong [9].

There are several techniques used in explainable AI (XAI) to explain how and why an algorithm arrived at a particular solution. These techniques can be classified as either model specific [1], meaning they are only applicable to a single type or class of algorithm or model agnostic [1], meaning they can be applied to any type of machine learning algorithm.

2.3.1 Model-specific methods

There are several model-agnostic explanation methods such as: Class Activation Mapping (CAM) [89], Gradient-weighted Class Activation Mapping (Grad-CAM) [70, 69] and Guided Grad-CAM [69] are propagation-based methods that are mainly used for CNN algorithms. These methods use visualizations to highlight the regions of the input that are important for predictions. Backpropagation-based methods such as DeepLIFT (Deep Learning Important FeaTures) [71] and Layer-wise Relevance Propagation (LRP) [6] identify the parts of data that strongly influence the output by comparing the input data with the reference data. Perturbation-based explainability methods such as Occlusion [35] and RISE [61] work by modifying the input and observing the changes in output. These are used in DNN algorithms to help understand which part of the input are important [35].

2.3.2 Model-agnostic methods

There are several model-agnostic explanation methods [5, 2, 1, 7] to open the black-box machine learning models. LIME (Local Interpretable Model-Agnostic Explanations) [64] is one popular technique which works by building a simple model (surrogate model) to explain the main model. The surrogate model is trained on a small subset of the original data that is close to the instance being explained. The model is then used to generate explanations for the prediction of the original model. Another important approach is SHAP (SHapley Additive exPlanations) [49] which works by assigning each feature an importance value to show how and to what extent each feature contributed to the final prediction result. This helps to understand the most important features and to explain how the model made its decision. There are also other methods like DLIME[2], Eli5[3] and Anchor[4].

In addition, there are also visualization methods/tools to understand the relationship between the model outcome and the features that are used to make predictions [86]. Partial dependence plots (PDPs) [25] help visualize the average effect the features have on the prediction with an assumption that the feature(s) for which the PDP is calculated is independent of other features in the data. For instance, if the dataset has two features that are independent, like height and weight. Then, taking averages for the partial dependence plot (for example weight to take PDP for specific height) will include samples that are very uncommon or even impossible. To avoid this problem, we can use other plots such as ICE or ALE [52]. Whereas Individual conditional expectation curve (ICE) [29] help visualize the functional relationship between the predicted response and the feature separately for each observation. Accumulated Local Effect (ALE) [4] plots on the other hand address the drawback of PDPs, their assumption of independence among features, by computing the average differences in predictions instead of averaging the predictions which in turn blocks the effect of correlated features. However, implementing these techniques requires users to have sound programming knowledge.

[2]https://github.com/rehmanzafar/dlime$_e$*xperiments*
[3]https://github.com/TeamHG-Memex/eli5
[4]https://github.com/marcotcr/anchor

2.4 Visualization as Intermediate representation (VLAIR)

Visualization as Intermediate Representation (VLAIR) [36], [37] is an alternate approach to applying machine learning (ML) and deep learning (DL) algorithms to classification problems. It leverages existing computer vision and information visualization techniques. In this approach, we first transform the raw input data into visual representations that are then used to train a vision-based deep learning algorithm.

VLAIR was initially applied to the Human Activity Recognition (HAR) tasks. HAR is the process of using sensor data to automatically recognize and categorize human actions by interpreting human gestures or motion [59, 79]. To achieve high classification performance, machine learning classifiers are widely used to detect and classify human activities based on sensor input [10, 82]. However, there are lack of ways to integrate the position and the timing of activation of binary sensors for classification algorithm which makes it difficult to classify human activities with necessary level of detail. To address this problem, VLAIR approach was used by transforming the binary sensor data into visualization by encoding both spatial and temporal information in a more human understandable way. The created visualizations are generated as images and are used as input for the CNN algorithm. The goal was to create a small-sized architecture that could train fast and run on a device with limited resources. The model only had to deal with simple images of primitive shapes like lines, rectangles, and circles. Therefore, they designed a CNN model that consisted of three 2D convolutional layers, each followed by a max pooling layer, a dense layer with 512 neurons and a dropout layer, and a softmax classification layer. Also, a total of 5 different visualization mappings were created and were tested and compared with different classifiers that were widely used to perform human activity recognition and was found that the most effective VLAIR approach performed as good as or better than the baseline algorithms. It was found that the VLAIR recognized the least frequent activities which had the least training examples in dataset and it also distinguished users on the same activities and sometimes also better distinguished the activities that take place in the same location.

VLAIR's intermediate representation of data through visualization makes the data more accessible to both machines and humans. Visualization helps humans better understand the data rather than from raw input data. When these data are used to train machine learning models, it offers insight into the workings of the ML model

which in turn provides opportunities for model debugging. For example, if the model is unable to differentiate between wandering in the living room and watching TV, representing this data through visualization helps people understand why that activity is not being recognized by the model.

Our system is designed on top of VLAIR to extend the benefits of VLAIR to a large population and apply the technique to various datasets.

2.5 Data Visualization

Information Visualization [53] is a field of study that helps people comprehend data better by representing it visually. It is based on the idea that humans are better at processing visual information than other types of information. Data visualization transforms abstract data into visuals such as length, position, shape, color and more to present compelling stories of data to humans who are more visually oriented. Data visualization is cited in this paper because it is an important technique in VLAIR to help users understand the data better and make informed decisions based on the visual representation of the data. However, it is not a direct solution to the problem of supervised classification.

Data visualization has three parts: Data - which is the data that needs to be visualized, marks - is a basic graphical element (like squares, circles, points, lines etc.) in an image, and Mapping - which maps data to corresponding marks.

There are several ways to create visualizations. One of the ways is to use text-based programming. The visualizations created using this approach will be more expressive. APIs such as OpenGL, Java libraries such as Flare[5] and Prefuse [33], JavaScript library such as D3.js [11] and web-based tools such as Vega[6] and Reactive Vega[68] fall under this category. However, it does require some programming knowledge. Not everyone is comfortable with coding because it is a skill that takes significant time to learn. On the other hand, tools such as Tableau[7], Excel[8], Microsoft Power BI[9], Lyra[67], Google Sheets[10], iVisDesigner[63], Data Illustrator[47], Voyager[85], VizDeck[40], iVOLVER[51] etc are GUI-based tools which provide more

[5]http://flare.prefuse.org
[6]https://vega.github.io/vega/
[7]https://www.tableau.com.
[8]https://products.office.com/en-us/excel.
[9]https://powerbi. microsoft.com
[10]https:// www.google.com/sheets/about/

user-friendly access without having to write code. In addition, tools such as Voyager, VizDeck, iVOLVER etc also supports small multiples [78].

2.6 iVOLVER

iVoLVER [51] is a web-based tool based on the constructive visualization approach [34] that allows users to create interactive visualizations without requiring textual coding skills. It enables users to capture data from multiple source types such as bitmap charts, web pages, photographs, SVGs, and CSV files and use them to generate visualizations. The tool uses the concept of Mark which are geometric shapes and can be customized with different attributes such as color and shape. iVOLVER offers easy-to-use interactions and gestures like drag-and-drop within a zoomable canvas where visual representations of the data and elements of the visual language are linked.

The iVOLVER tool has two mechanisms to transform data before they are turned into visuals, without the need for textual programming: functions and mappers. These define a relationship between a set of inputs and a set of outputs where each input is mapped exactly to one output value. Mappers help define the relationships between values of varying types (such as numbers, colors, and dates). For example, mappers can be used to apply a color gradient to differentiate countries with dense and least population. The idea of univariate functions of real numbers is the basis for functions. Functions are similar to mappers but it only works with numbers. iVOLVER also provides options to perform basic arithmetic operations on values. These operators include addition, subtraction, multiplication and division.

iVOLVER supports small multiples. With iVOLVER, users will be able to create visualizations for individual rows of data instead of generating a large single visualization for all the rows of data. With small multiples, users can compare the separate images, look for patterns, trends, and outliers [78]. The design of our visualization tool is based on iVoLVER.

Chapter 3

Design Goals, Principles and Methodology

We designed iVLAIR with the overall goal of helping non-programmers to perform machine learning classification. More specifically, we aim at the following subgoals:

G1 To make ML classification tasks more interpretable (see definition of interpretability in section 2.3).

G2 To help users to perform data classification interactively without (learning) coding.

G3 To facilitate sense-making of ML results and the iterative improvement of models.

Visualizations are a popular way of representing data visually to help users understand the patterns in data that are not immediately apparent from raw data. Because of this advantage, there are several popular visualization techniques that are used to show how the model has made a prediction by explaining the important features and how those features are related to each other (see section 2.3).

Inspired by the research field of visualization, the input data is transformed into visual representations that are then used as input to the Convolutional Neural Network (CNN) algorithm to perform data classification. This approach is called the Visualization as Intermediate Representation (VLAIR). This was initially applied to the Human Activity Recognition (HAR) tasks but are capable of applying to other classification tasks. VLAIR doesn't achieve the goals stated above because this is

a technique that still needs to be programmed. Therefore, it should be useful to create an interface that makes VLAIR accessible to non-programmers, trying to still make it possible for those to understand what is going on in the classification. This constitutes the base of our system.

Most machine learning algorithms act as black-boxes, meaning that how the algorithm makes decisions is largely unintelligible to humans. Understanding the model and its outcomes helps human predict what it might do in unforeseen situations. This provides transparency and builds trust, encouraging users to use the system [83, 80]. This has made interpretability in ML an important area of research (**G1**).

Iterative improvement of the model is important in machine learning because it helps to gradually improve the accuracy of the model and reduce the margin of error. Sensemaking [14] is the process of constructing meaning from raw data. Sensemaking of machine learning results will help in analyzing and understanding how and why the model has arrived at a particular result. This will help identify patterns and trends that can be used to improve the model (**G3**).

Coding is a skill that requires a significant investment of time. Some people may not have the necessary background orthe required time to learn it. However, they may still need to carry out these tasks and, perhaps more importantly, need some understanding of how the machine performs classification. To help non-programmers, we built a user interface with a simple drag-and-drop platform that allows users to create visualizations (using familiar elements based on the constructive visualization approach [34]), build machine learning models and to generate prediction results without manual coding (**G2**).

In order to effectively design the iVLAIR interface, we need some design principles, which support the design of the interface based on the goals that we aim for.

DP1 *Support operation without coding*: To make machine learning accessible to a wide variety of users, we aim to avoid the need to write textual programming whenever possible with the help of a simple interface (**G2**).

DP2 *Leverage visuals as much as possible* Visuals can help communicate information more effectively. They can also aid in better understanding of the information (**all Gs**). To leverage the VLAIR approach the design should use visuals and visualizations as much as possible.

DP3 *Enable continuous improvement*: Continuous improvement is important in machine learning because it allows for the creation of more accurate models that

can be used to make better predictions and decisions. The system design should allow users to understand the results and improve the models (**G1** + **G3**).

DP4 *Allow participants to query iteratively*: Sometimes users may find it difficult to understand the results or find what they are looking for. Iterative queries allow users to refine their search and get more specific results to analyze. Users should then be able to test the results repeatedly until they get a deeper understanding (**G1** + **G3**).

DP5 *Avoid unnecessary complexity*: Our design should focus on building a UI which is simple and clean. This can be done by including only the elements that are absolutely necessary and to remove all the unnecessary elements (**G2**).

DP6 *Avoid menus and hidden operations*: Non-transparent computation and hidden menus may limit the use of tools. Our tool should help users access all the information they need without having to navigate through complex menus or hidden operations (**G2**).

DP7 *Avoid reliance on visual memory*: Recalling visual information increases cognitive load [62]. To avoid requiring users to remember information from one part of the interface to another, our system design should helps users navigate through the interface without having to rely on visual memory and should try to present blocks of information that are likely to be used simultaneously in the same screen (**G2**).

3.1 Design Methodology

We performed the following design and implementation activities concurrently and in an iterative manner. First, we sketched the overall design of the tool, taking into account the workflow of the machine learning algorithms and the design principles outlined above. Then we decided on the user interface and what elements would be used to create it, and how these elements would contribute to the overall interaction of the tool. After that, we began implementation. Then, we demonstrated our work and discussed it with the research team each week to validate the design. Initially, we planned to integrate iVOLVER [51] for visualization construction and OpenRefine[1]

[1]http://www.openrefine.org

for data cleaning and transformation processes. However, the design of these tools do not fully suit our needs and needed too much customization. For example, iVOLVER is designed with a touch and pen-friendly interface, so the objects in the canvas are sized slightly larger than a fingertip to enable touch manipulation; however, we did not want a touch or a pen-friendly interface and we wanted more space in the canvas to create visualizations, with smaller objects. iVOLVER has a single interface to handle both the main visualization creation and the background operations; however, our design was to separate both these operations for clarity. Finally, the iVOLVER tool has a component to extract data from multiple data sources but our requirement was to accept the input data in .xslv or .csv formats and to transform them into visualization. Similarly, we encountered challenges when integrating OpenRefine with our web application tool because our front-end technology differed from that used in OpenRefine. We also wanted to build a simple data cleaning tool that could support basic functions such as addressing missing values and highlighting data type mismatch errors. We ultimately created our own visualization tool and data cleaning tool from scratch, although inspired in those existing tools.

Chapter 4

iVLAIR

This chapter describes the current version of the iVLAIR interface, which is the main contribution of this thesis. The chapter first describes the overall designed flow and then describes each of the screens in detail. The screens correspond roughly to the tasks and activities of the flow. References to the design principles (e.g., DP1) indicate the design motivation.

The iVLAIR tool has a total of six screens, each of which focuses on a particular subtask of the process of building an ML-based classification model. Figure 4.1 shows an overview of the overall flow of the tool.

1. Users first log in to the application and upload the input file.

2. They can then clean the data set using the transform data screen before moving to the visualization screen or they can directly navigate to the visualization screen if the data cleaning is not required.

3. In the visualization screen, users create a visual mapping, and then the system uses this mapping to convert the raw data into visualizations.

4. In the ML screen, an image classification algorithm is used for data classification using the generated images, and the results are displayed.

5. If users are satisfied with the results, they can make use of the validate button on the ML screen to perform validation.

6. If the results are not satisfactory, users can use the compare results screen to view and compare images that gave correct and incorrect results. This can help

them get a deeper understanding of why and where the algorithm has made correct and incorrect predictions. Based on this understanding, users can go back to step 3 to modify or create a new visualization and then execute the next steps. Users can repeat this process until they get the desired result.

DP5 and **DP6** design principles are used to construct all screens.

4.1 Screen 1: Login and File Upload

To provide personalized and persistent access for the application to individuals and to prevent unauthorized access to data by others, every user has to first log in to the application.

The iVLAIR login screen has two text boxes 'username' and 'password' (marked A and B) in the middle and an action button 'Login' (marked C) at the bottom. To log in to the application, iVLAIR users have to enter their username and password and click on the 'Login' button.

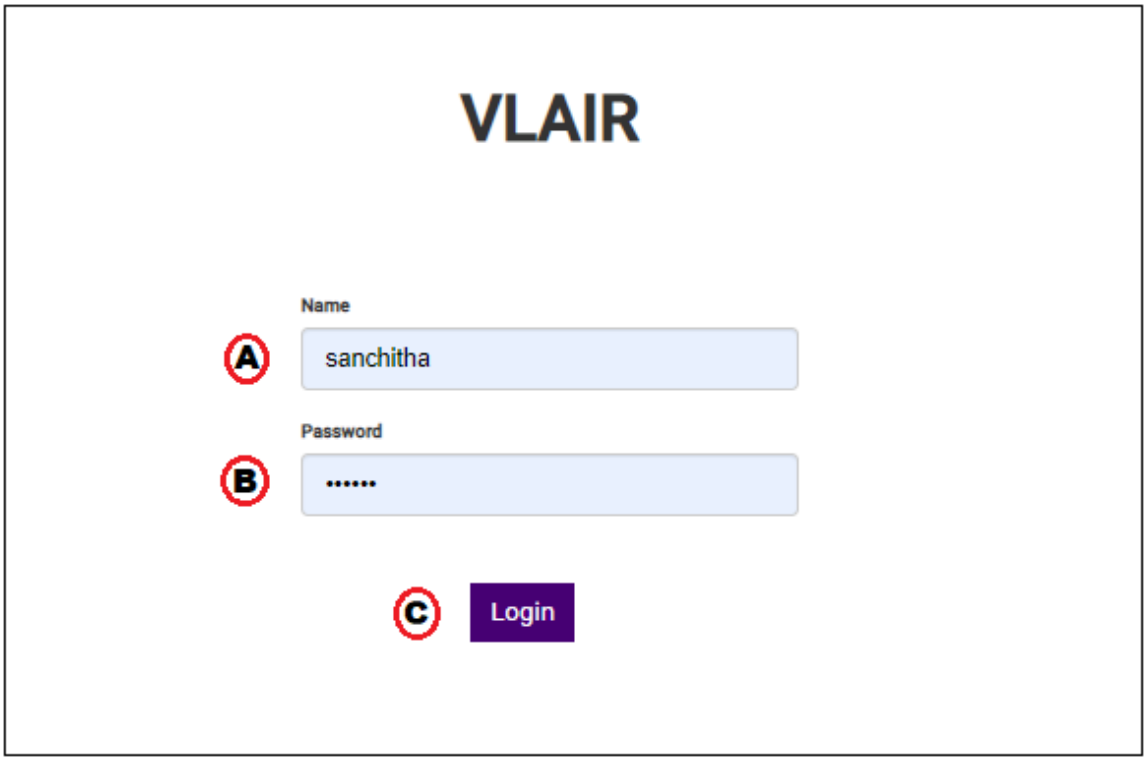


Figure 4.2: Login screen

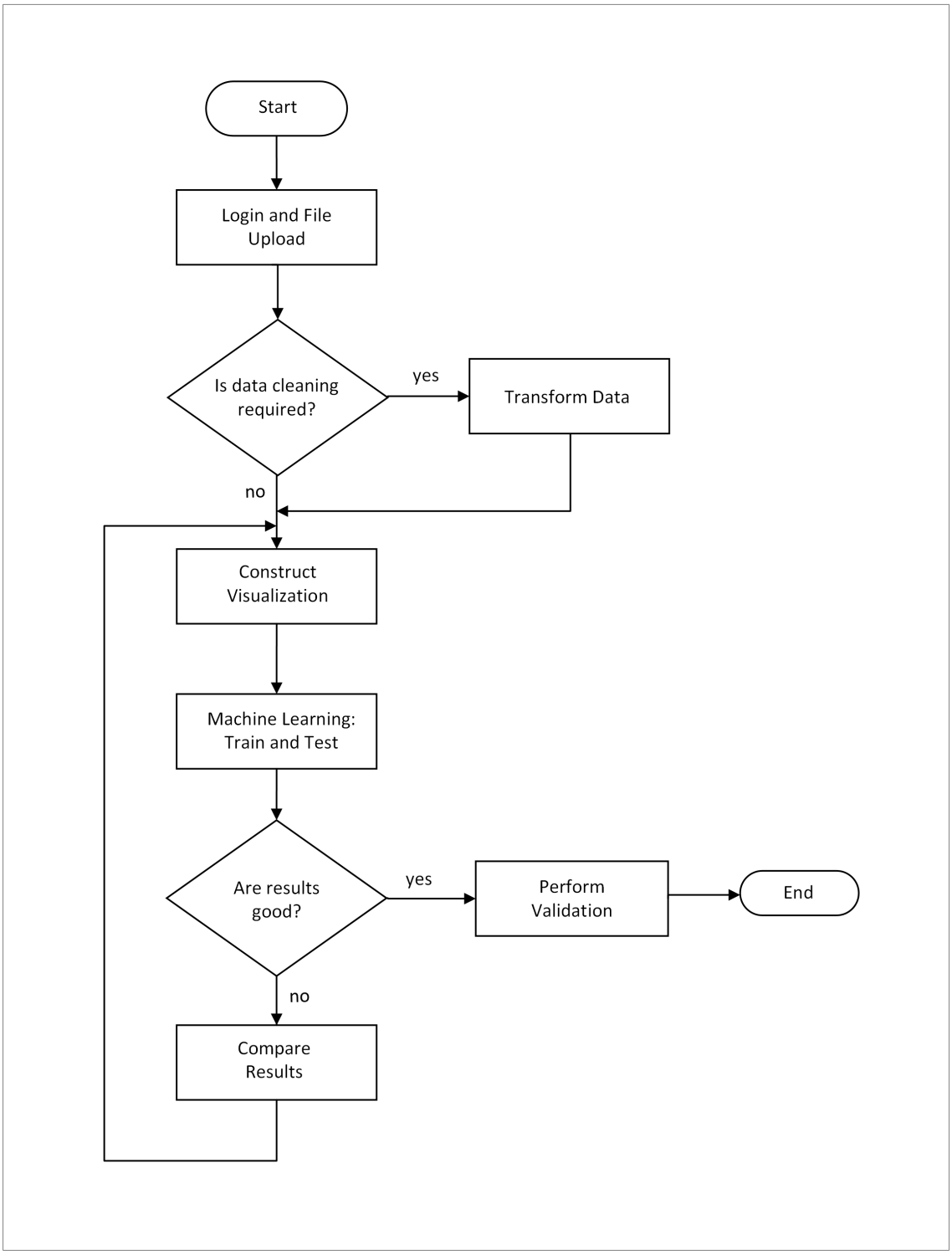


Figure 4.1: System flow

Users can make use of this file upload screen (see Figure 4.3) to upload data in a format that the system can process. This screen has commands selection bar at the top, a data table in the middle, and action commands at the bottom to move forward in the process.

In this chapter, we will use the term training, testing and validation datasets. Traditionally validation data is used to test dataset is used to evaluate the final performance of the model. However, the terminology test and the validate datasets are sometimes interchanged. Sometimes, the terms testing and validation are used interchangeably, as in the book Applied Predictive Modeling [42]: *"Ideally, the model should be evaluated on samples that were not used to build or fine-tune the model, so that they provide an unbiased sense of model effectiveness. When a large amount of data is at hand, a set of samples can be set aside to evaluate the final model. The "training" data set is the general term for the samples used to create the model, while the "test" or "validation" data set is used to qualify performance"*. We have also followed this convention and used testing and validation as synonyms. Below are the roles of these datasets in our tool. [84, 65, 77].

- Training Dataset: This dataset is used to train the model.
- Testing Dataset: This dataset is used to evaluate the model fit on the train dataset without any bias while adjusting the parameters of a classifier.
- Validation Dataset: This dataset is used to evaluate the model's performance on new data that it has not seen before and provides an unbiased estimate of how well the model will perform when deployed for actual predictions in real-world situations. The reason is if you optimize your algorithm for a given training-testing set, you will never know if the algorithm is good generally, or just good for that particular combination of training-testing data. If the results for the validation set are dramatically lower is because one has overfitted to the training data, and the validation step provides that extra protection.

  Data allocated for validation is recommend to be used only once at the end for the final evaluation of the model eventhough it is still possible to use it multiple times.

To upload a file, users can click on the 'Choose file' button (marked A) and upload the input file in .xlsx and .csv formats. Users can then allocate a certain percentage of data for validation (marked B), and the rest of the data will be displayed in a

table (marked E). Users can choose to see either the first 10, last 10, random 10 or all available rows of data in the table (marked E) by choosing the options from the 'Available rows' dropdown (marked C). To help the tool understand the column it has to predict using the machine learning model, users can make use of the 'Select label' dropdown (marked D) to select the label that has to be predicted from the list of all the column names. Finally, users can move on to the next screen by either clicking on the 'Transform data' button (marked F) if they are likely to deal with missing or null values or they can click on the 'Visualize' button (marked G) to create visual mapping and to use that mapping to re-represent the input data into visualizations.

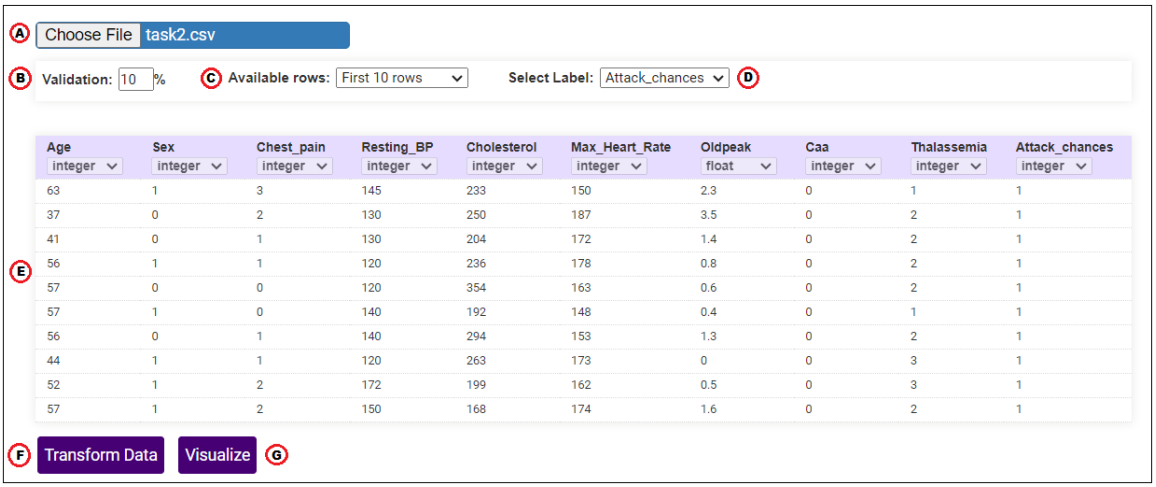


Figure 4.3: File upload screen

4.2 Screen 3: Transform Data

The main purpose of this screen is to identify and address any missing or null values in the data. It also helps users identify any datatype mismatches that may exist. Additionally, it allows users to create a new column. This screen has selection bars at the top for commands, a data table in the middle, and action commands at the bottom for performing some actions or moving back to the previous screen.

Upon entering this screen, the uploaded input data is evaluated and displayed in table (marked D). The table also highlights the columns with null values and datatype mismatches. Users can choose to see the first 10, random 10, last 10 or all available rows in the table marked D from the 'Available rows' dropdown (marked A). The screen has two sections marked B and C which allow users to address null values and

create new columns respectively (discussed in the subsections below). Finally, there are two buttons at the bottom of the screen. The 'Cancel' button (marked G) allows users to close this screen and navigate back to the previous screen (see File Upload screen discussed in section 4.1). Finally, users can click on the 'Re-evaluate' button (marked H) to re-evaluate the entire dataset and highlight any missing errors. This feature helps ensure that the dataset is as precise and comprehensive as possible, which is crucial for making informed decisions based on the data.

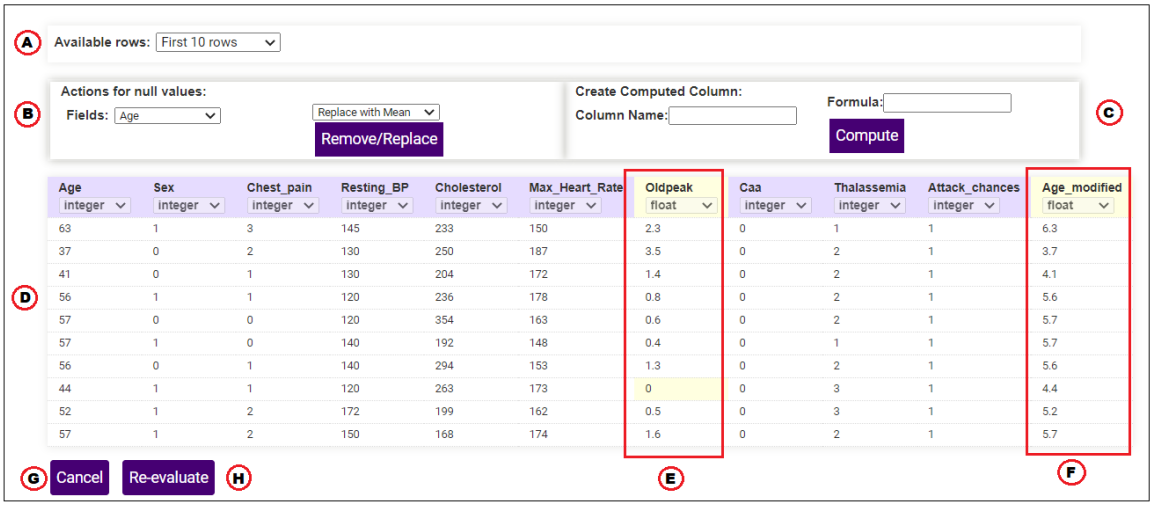
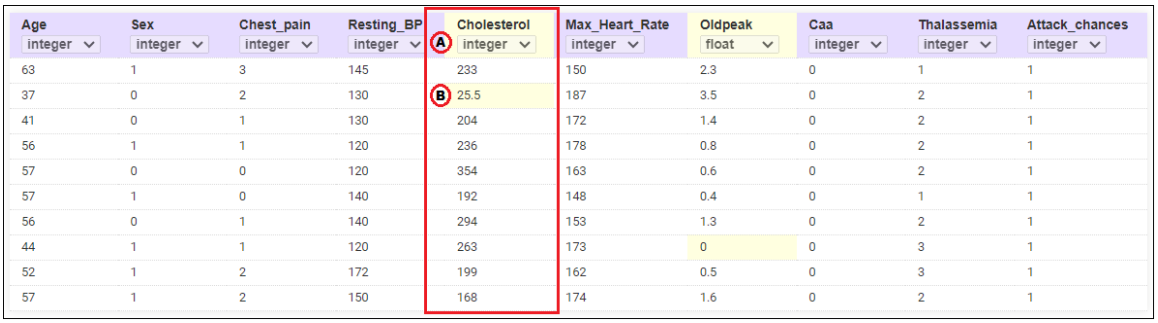


Figure 4.4: Transform data screen

4.2.1 Identify Data Type

The data type of each column in the input dataset is determined by assigning the data type that has the most values and displaying it in the table header (marked A in Figure 4.5). Then, data whose data type is different from the identified one are highlighted (marked B in the same Figure), and users can manually correct those errors.



Age	Sex	Chest_pain	Resting_BP	Cholesterol	Max_Heart_Rate	Oldpeak	Caa	Thalassemia	Attack_chances
integer	integer	integer	integer	integer	integer	float	integer	integer	integer
63	1	3	145	233	150	2.3	0	1	1
37	0	2	130	25.5	187	3.5	0	2	1
41	0	1	130	204	172	1.4	0	2	1
56	1	1	120	236	178	0.8	0	2	1
57	0	0	120	354	163	0.6	0	2	1
57	1	0	140	192	148	0.4	0	1	1
56	0	1	140	294	153	1.3	0	2	1
44	1	1	120	263	173	0	0	3	1
52	1	2	172	199	162	0.5	0	3	1
57	1	2	150	168	174	1.6	0	2	1

Figure 4.5: Tool highlighting the column with datatype mismatch

4.2.2 Actions for null values

When null values are detected in the data, they will be highlighted automatically (marked E) as shown in Figure 4.4. From the section 'Actions for null values' (see Figure 4.6), users can select the highlighted column name from the 'Fields' dropdown (marked A). The null values can then be removed either by deleting the entire row or replacing them with a mean or median value. Replacing missing values with mean, median and mode values are some of the most common methods used in data preprocessing for small dimensional datasets because of their simplicity [23, 39]. These actions can be chosen from the dropdown (marked B) and click on the 'Remove/Replace' button (marked C) to apply the selected action.

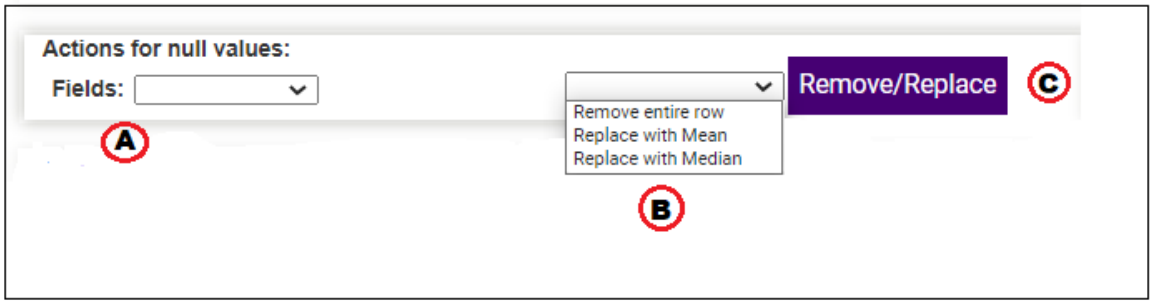


Figure 4.6: Different actions for null values

4.2.3 Create new column

In some cases, features of the data need to be derived from existing data because they do not exist but are easy to compute. For example, in a dataset with prices and area of apartments, it might be useful to have a column (feature) with the price per

square meter.

Such columns can be created by combining two existing features or a feature with a constant using basic arithmetic operations (addition, subtraction, multiplication, division) and their combinations, enter the name of the new column and the formula in the text boxes marked A and B respectively and click on the 'Compute' button marked C in Figure 4.7.

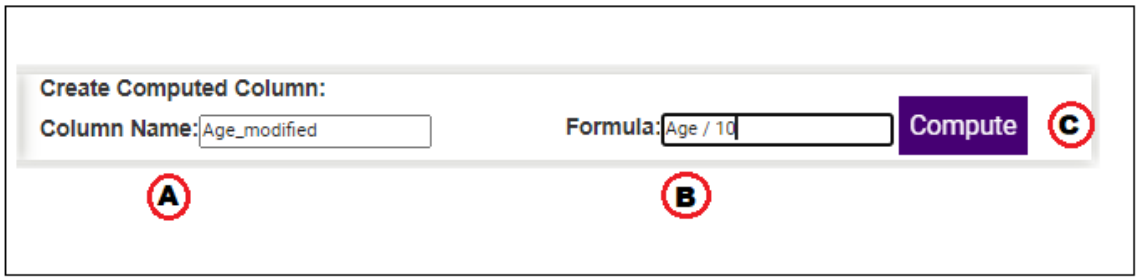


Figure 4.7: Create new column

4.3 Screen 4: Construct Visualization

The VLAIR approach is based on the creation of visualizations as intermediate representations. The interface on this screen enables users to define mappings that would create those visualizations. You can find the concepts of mappings in section 2.5.

This section first describes the main components of the tool and then the elements that make up its visual language.

4.3.1 Overall tool Interaction

In Figure 4.8, iVLAIR's visualization tool has two canvases: main canvas and bottom canvas (C and E). In both these canvases objects can be dropped and can be relocated by dragging (DP1). The visualizations are created on the main canvas and the data manipulations are performed on the bottom canvas. The main canvas has a tool bar on the top (marked B). The uploaded input data is displayed in the form of a table on the left panel (marked A) to help users easily refer to the input dataset instead of having to move back to the previous page (DP7). Each row of data in the table has a radio button to its left and users can modify the selection. The middle panel (marked D) shows the list of input column names with values. The values to these

columns are based on the selection on the left panel. Finally, the right panel (marked G) displays the visualizations created for all rows of data.

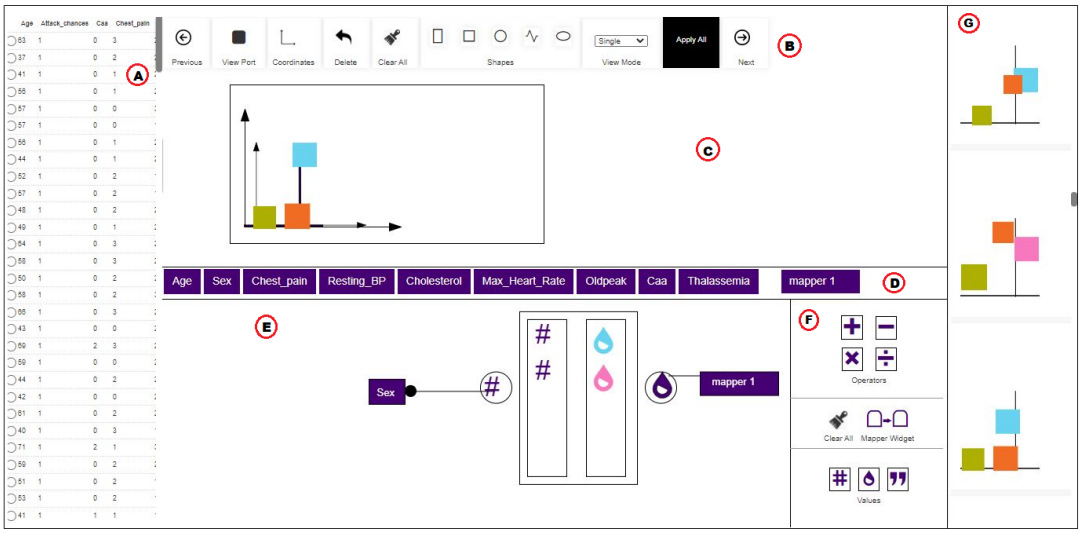


Figure 4.8: visualization tool

4.3.2 Values and Data Types

The iVLAIR visualization tool supports values with two different data types: numbers and colors (marked C in Figure 4.9). Users can drag and drop these values into the bottom canvas (marked F) (DP1). Values are represented as rectangles with a symbol inside, and the symbols identify the type of the value. Both numeric and color values accept manual input (marked A and B).

Users can click on the numeric value (with a single click) to open a text box (marked C) where they can enter a number input. Numeric values have circular output ports on their right, and users can assign the value to another object by creating a connection from the output port.

Users can click on the colors value (with a single click) to open a color palette, where they can choose the desired color as shown in the same figure (marked D).

This tool also accepts input column values (which users can drag from the middle panel, marked H) and drop them into the canvas (marked E). Similar to numeric values, this also has an output circular port on its right, and users can assign this value to another object by creating a connection from the output port.

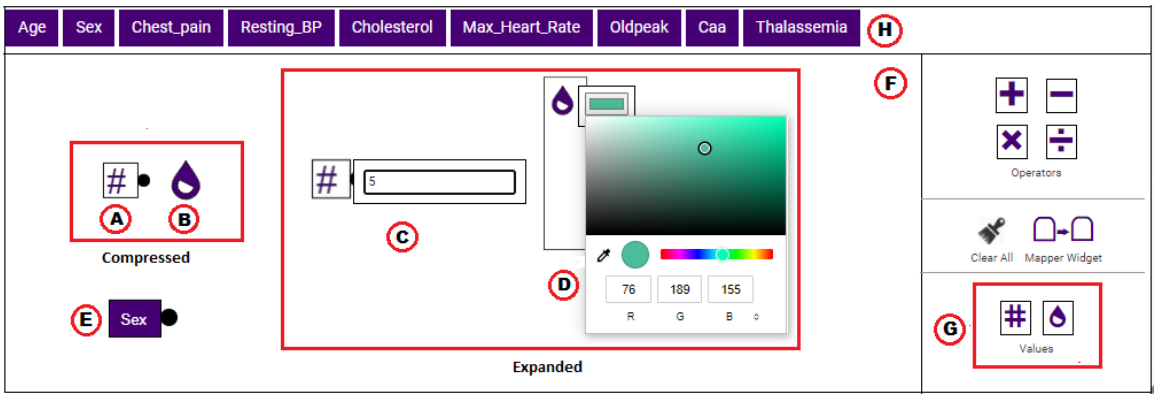


Figure 4.9: Values and Data Types

4.3.3 Marks

Marks are the graphical elements in the visualization. The use of visual marks to create visualization is inspired by iVOLVER [51].

iVLAIR has five types of visual marks: rectangles, squares, circles, lines and ellipses (marked D in Figure 4.10). Users can create the marks by dragging and dropping from the top panel (marked C) onto the main canvas. Users can click on the marks (with a single click) to view their visual properties on the right panel inside the main canvas (marked E). The visual properties include position, coordinates, color, rotation and some or a combination of geometrical dimensions such as length, width, height, side, radius and area. Users can adjust the visual properties of the marks by assigning a manual input (marked F) or by assigning the input column values (highlighted in purple color as shown in Figure marked G) through a simple drag and drop from the middle panel (DP1) (marked H).

If the user wants to delete a particular mark, they can first select the mark that has to be removed and they can click on the 'Delete' button (marked A). On the other hand, they can use the 'Clear All' button (marked B) to clear all the objects added to the main panel.

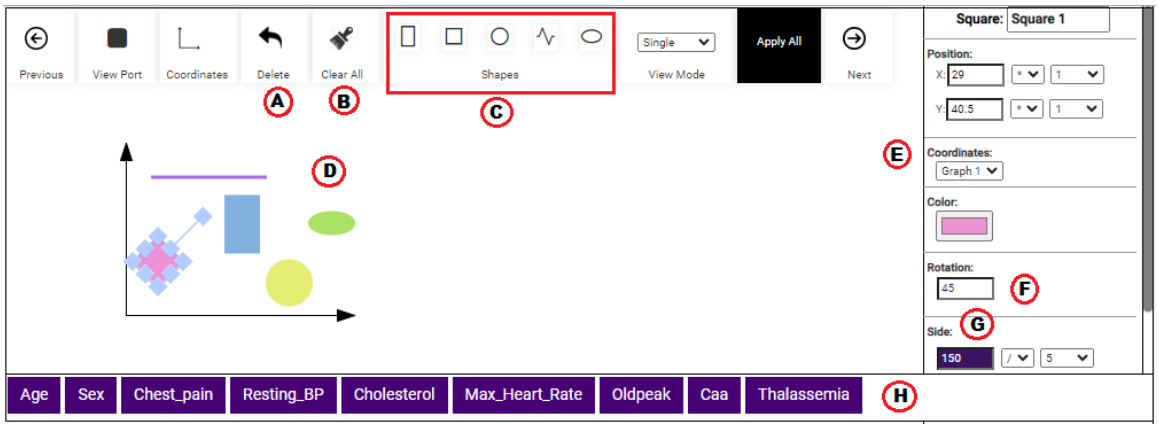


Figure 4.10: Marks

4.3.4 Operators

Imagine that if a dataset has two columns (features) that show the number of hours worked and the hourly wage of employees, users can derive data for a new column with the total wage earned by each employee by performing arithmetic operations on the existing column values. These operations take place per-row of data and is inspired by iVOLVER [51].

The visualization tool has four basic arithmetic operators: addition, subtraction, multiplication and division (marked G in Figure 4.11). All these operators are represented as rectangles with the operator's symbol inside them. These operators accept two incoming connections (represented by circular ports on the left with arrows inside) marked C and D and one outgoing connection (represented by a circular port on the right with arrow inside) marked E. Operators accept numeric inputs that are passed either as a constant or an input column value or a combination of both. Inputs (marked A and B) are passed to the operators by enabling a connection from output port of the input values to the input ports of the operators. Input values carried by the incoming connections are combined through arithmetic operations and the corresponding numeric output is generated (marked F). The output can be dragged and dropped to the middle panel (marked H) and can be passed as input to the visual properties of different marks (Section 4.3.3) or can be passed as an input to another operator or to a mapper widget (Section 4.3.5).

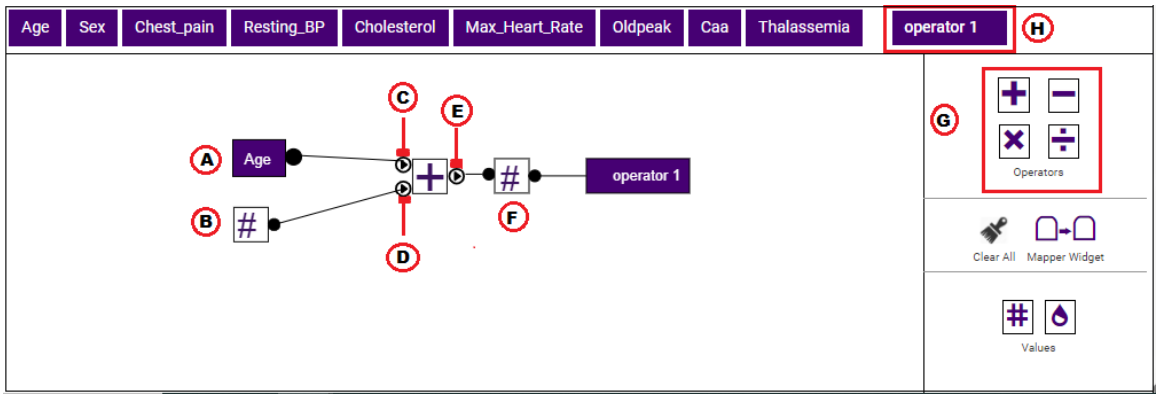


Figure 4.11: Operators

4.3.5 Mapper

The mapper widget helps to transform numeric data into colors in a more visual way without the need for programming (DP1). The use of this mapper widget is inspired by iVOLVER [51].

The mapper widget is created by dragging the mapper icon (marked I) into the bottom panel (marked K) as shown in Figure 4.12. The mapper widget has two vertical rectangles: the left rectangle (marked E) holds a collection of numeric inputs (marked A), and the right rectangle (marked F) holds a collection of color outputs (marked B). Each numeric input is added to the left rectangle and is aligned horizontally with a color output on the right rectangle by simple drag and drop. This alignment determines the relationship between the input and output elements. Also, there are two circular ports: input and output ports. The input port (marked C) is on the left of the left rectangle and it accepts numeric values from input columns (marked G). The input values are assigned to the input port of the mapper widget by enabling a connection from output port of the input values. Similarly, the output port (marked D) is to the right of the right rectangle and it outputs a color value that corresponds to the horizontal alignment of the specific numeric input. For example, in the mapper widget shown in Figure 4.12, the value 0 is horizontally aligned with the color blue and 1 is horizontally aligned with the color pink. Now, when 1 is passed as input to the mapper widget, it outputs pink color. For values in-between 0 and 1, the mapper creates linear interpolations. The output from the mapper widget can be dragged from its output port and dropped into the middle panel (marked I). This makes the mapper output available to be used as input for the visual properties of

marks Section 4.3.3.

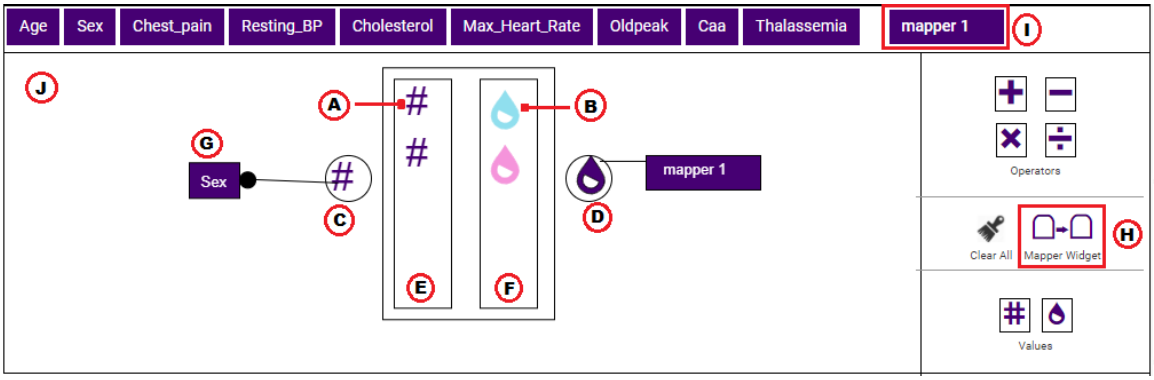


Figure 4.12: Mapper

4.3.6 Coordinate axes

In Figure 4.14, the main canvas has a default coordinate axis marked F that controls the x and y position of all visual marks added to the main canvas. The coordinates icon (marked C) allows users to add another coordinate axis (marked I). Users can change the coordinates property of the selected visual mark to point to the second coordinate axis (as shown in the figure marked J). The x and y position of the visual mark is now controlled by the second coordinate axis.

Multiple coordinate axes are used when user wants the position of a mark to be associated with the position of another mark. For example, you can create a square by dragging and dropping the square icon onto the main panel. Then assign some value to the x and y position of that square and make the position of the square controlled by the default coordinate axis. Next, create another coordinate axis and assign the same x and y position value to this coordinate axis. Finally, create another square and make the x and y position of this square controlled by the second coordinate axis. This makes the position of the second square associated with the position of the first square.

4.3.7 View port

Viewport helps users to choose the whole or part of the visual mapping they want to use to transform the input data into visualization.

Users can draw a rectangular region on the main canvas around the created visual mapping (marked G) by clicking on the view port icon (marked B in Figure 4.14).

The selected region represents the part of the created visual mapping that can be used to transform all the rows of data into visualization and then to convert them into images. The mapping outside the selected region will be cropped.

This feature is useful in iVLAIR when the user wants only the portion of the created visualization to be converted into images.

4.3.8 Modes

The view mode dropdown (marked D in Figure 4.14) allows users to view the created visualization in two different modes. Single mode allows users to view only the visualization created for a single row of data (as shown in Figure 4.14). Whereas, collective mode allows users to view the visualizations created for all the rows of data (marked B in Figure 4.13). This feature is useful when the user wants to see the visualizations created for all the rows of data in a single place and to detect any data outliers.

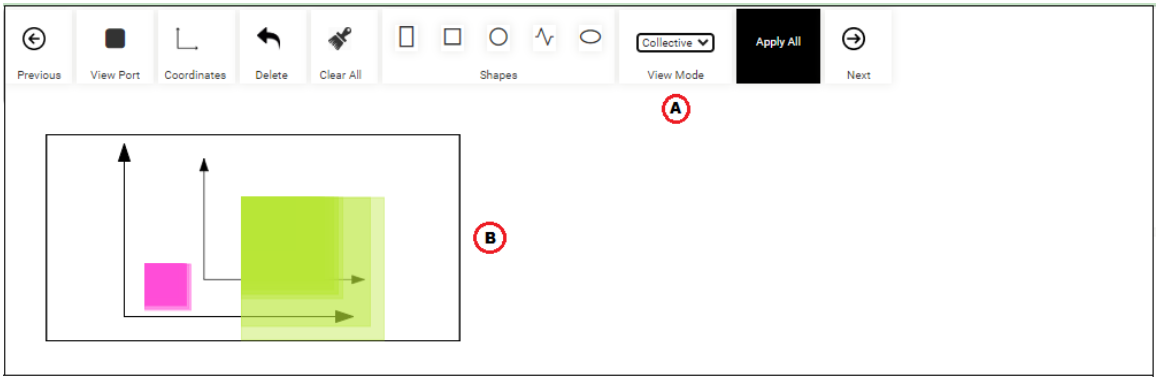


Figure 4.13: Collective mode

4.3.9 Apply to all rows

As shown in the Figure 4.14, clicking on the 'Apply all' button (marked E) uses the visual mapping inside the rectangular region (marked G) to convert all the rows of data into visualizations. The created visualizations are then converted into images and are displayed on the right panel (marked K). After the visualizations are created, users can click on the 'Next' button (marked F) to navigate to the next screen (Section 4.4) and clicking on the 'Previous' button (marked A) will take user to the previous screen (Section 4.1).

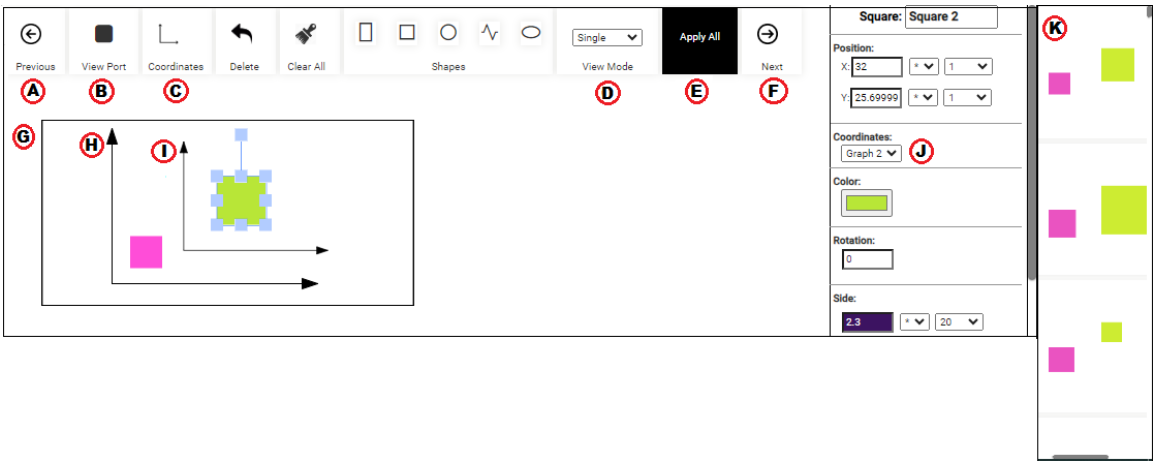


Figure 4.14: Top panel

4.4 Screen 5: Machine Learning: Train and Test

The main purpose of this screen is to use the images generated in the previous screen to train and test the CNN algorithm. Additionally, if the users are satisfied with the results, they can use the created visual mapping to transform the validation dataset into visualization, which are then used to perform validation on the final model.

This screen has text boxes at the top, data tables in the middle, and action commands at the bottom to move forward in the process.

4.4.1 UI Interaction

In this screen (see Figure 4.16), the images generated in the previous screen (see Section 2.5) are randomly divided into training and testing datasets based on the input in text boxes (marked A) and are displayed on the table (DP1, DP2 and DP7), marked E. The images allocated for training are represented with solid borders while the images allocated for testing are represented with dotted lines which can be understood with the help of the labels (marked B). The zoom option (marked C) allows users to view multiple images at once. The label (marked D) displays the values of the column to be predicted. Clicking on the value will load the corresponding images to the table on the right (marked E). The Train, Test, and Validate buttons (marked F and G) allows users to train and test the ML model and then to perform validation. The results of prediction are shown in the Figure marked I and the confusion matrix

on the right (marked J). Finally, users can navigate to the next screen by clicking on the button at the bottom of the screen (marked K).

4.4.2 Train & Test

During the training process in machine learning, the algorithm learns from the labeled data. During the testing process, the algorithm uses that knowledge to classify new data. Simply click on the 'Train' button followed by the 'Test' button (marked F in Figure 4.16) to use the images allocated for training and testing to train and test the integrated VGG16 algorithm (DP1, DP2). After the training is completed, users can view the log by clicking on the 'view log' hyperlink (marked H). The generated log is shown in Figure 4.15. After the testing is completed, users can view the prediction results (marked I) along with the confusion matrix (marked J). Users should go back to the previous screen to modify or create a new visualization. Then, they should re-train and re-test the model with the new images to improve the prediction result.(DP3).

Results X

Epoch	Loss	Categorical Accuracy
1	0.6418622732162476	0.6125654578208923
2	0.5957230925559998	0.7015706896781921
3	0.5879809260368347	0.6858638525009155
4	0.5827963352203369	0.7068063020706177
5	0.5869995951652527	0.7015706896781921
6	0.5754866003990173	0.7120419144630432
7	0.565565824508667	0.717277467250824
8	0.5774223804473877	0.7120419144630432
9	0.5717018842697144	0.7068063020706177
10	0.5761213898658752	0.7068063020706177
11	0.5592342019081116	0.717277467250824
12	0.5595495700836182	0.7068063020706177
13	0.5602486729621887	0.7120419144630432
14	0.5636743307113647	0.7434554696083069
15	0.5509885549545288	0.7225130796432495
16	0.560524046421051	0.6963350772857666
17	0.5462248921394348	0.7382199168205261
18	0.5482637286186218	0.7120419144630432

Figure 4.15: Training logs

4.4.3 Validate

After the users are satisfied with the test results, they can use the validation dataset (see section 4.1) allocated in File Upload screen (marked B in Figure 4.3) to evaluate model's performance for unseen data. To perform validation, users can click on the 'Validate' button (marked H) to validate the model with the allocated validation dataset . Once the validation is completed, the final prediction results (marked I) and the confusion matrix (marked J) are updated with the validation results. Please note that the data allocated for validation is recommend to be used only once at the end for the final evaluation of the model eventhough it is still possible to use it multiple times.

4.4.4 Metrics

Our main evaluation metrics are:

Accuracy: The proportion of correctly classified samples.

$$\text{Accuracy} = (\text{TP} + \text{TN})/(\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

Precision: *"Precision evaluates the fraction of correct classified instances among the ones classified as positive"* [24]. We use the macro precision which is the average of precision scores for each class.

$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP})$$

Recall: Recall shows the proportion of true positives that the model correctly predicts out of all the possible positive cases. We use the macro recall which is the arithmetic mean of recalls.

$$\text{Recall} = \text{TP}/(\text{TP} + \text{FN})$$

F1-scores: *"F1-score is the harmonic mean of precision and recall"* [90]. We used the macro f1-score which is the average of F1-scores from all activity classes values [56].

$$\text{F1 score} = (2 * \text{Precision} * \text{Recall})/(\text{Precision} + \text{Recall})$$

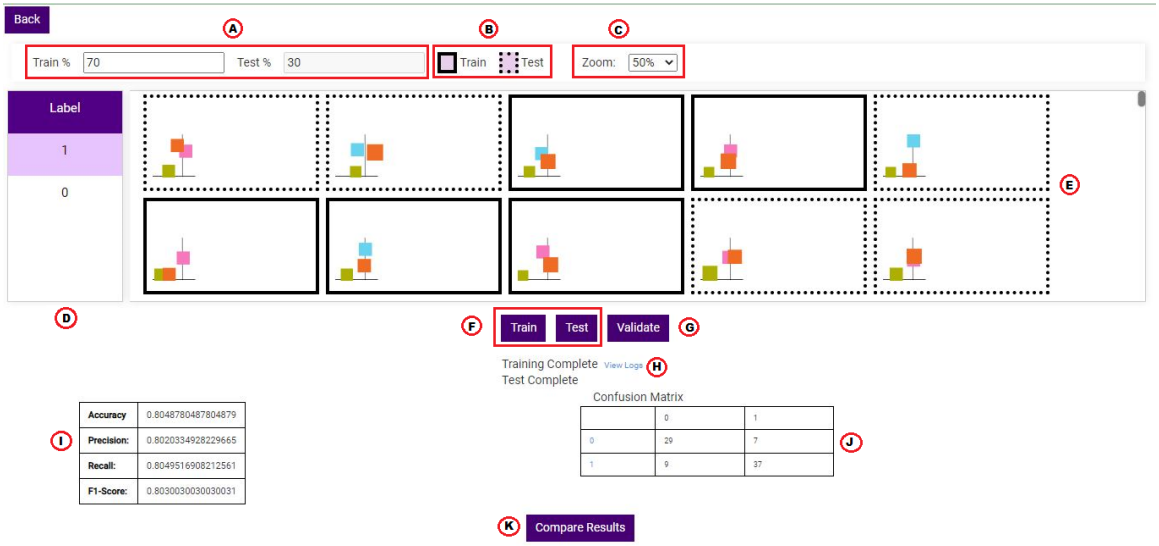


Figure 4.16: Machine Learning Screen

4.5 Screen 6: Compare Results

Users can use this screen to load and compare images until they get a better understanding of where the algorithm succeeded and where it failed (DP4). Based on this understanding, users can also go back to previous screens to either modify visualizations or retrain models to improve prediction results (DP3).

This screen has four panels (C, D, E and F) as shown in Figure 4.17. There are two confusion matrices on the right: the train and the test (DP2) (marked G and H respectively). The test confusion matrix is taken from the previous screen (see Figure 4.16.J). Clicking on a panel will highlight that panel (marked E) and enabling the checkbox on the confusion matrix will load all the images corresponding to the selection (DP2). Enabling the checkbox on the train confusion matrix will load all the images that were used to train the machine learning model whereas enabling a checkbox on the test confusion matrix will load all the images that gave true positive, false positive, false negative, and true negative results. Image with solid border represents the train dataset and dotted line represents the test dataset. Also, the images that gave correct results are surrounded by blue lines whereas the images that gave in-correct results are surrounded by red lines. This can be understood with the help of the labels (marked A).

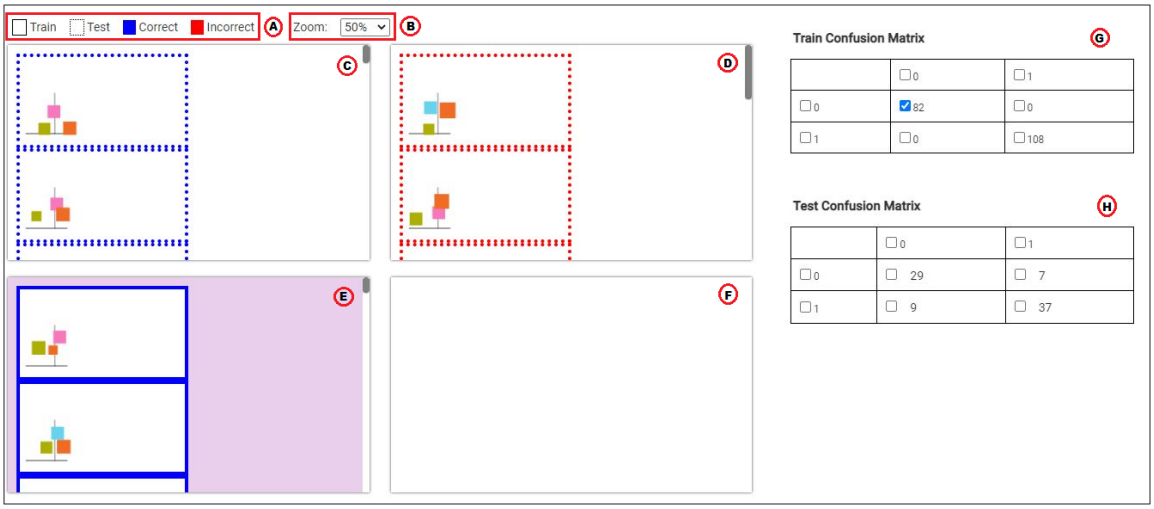


Figure 4.17: Compare Results screen

Chapter 5

Example Scenario

In this chapter we present a hypothetical scenario to illustrate how the overall system is used to complete the generation of a classification model starting with the data.

Alice is a project manager at a large retail company. Her company has recently started using machine learning to classify the employees in terms of their needs, so that the company could provide better support adapted to their own characteristics (but still in groups, so that it does not become extremely expensive to provide support one-by-one). Alice is responsible for classifying the data of 450 employees in her department. However, she is not familiar with machine learning and does not have the time to learn it. Therefore, she decides to use a no-code machine learning tool to perform data classification. Additionally, she needs to know how the model arrived at the prediction result as she has to present it to the project directors. After researching many tools, Alice finds that iVLAIR would help her with her needs and learns how to use it.

5.1 Screen 1: Login and File upload screen

Alice uses the login screen (see Figure 4.2) from the previous chapter to enter her username and password and clicks on the login button to access the application. She then uses the file upload screen (see Figure 4.3) and clicks on the file upload button and uploads the employee dataset. After that, she enters '10' in the validation percentage text box to allocate 10% of data for validation. She then selects 'performance' which is the column name that has to be predicted from the select labels dropdown.

5.2 Screen 2: Transform data screen

Alice is sure that she needs to clean her data first so she clicks on the 'Transform Data' button to move to the transform data page (see Figure 4.4). On that page, she first identifies that her input data has some null values and decides to replace them with the mean value. The employee dataset does not have a column that shows the years of experience of each employee; it only has a column that holds the joining date of the employees. Alice wants to calculate the years of experience of the employees, so she uses the 'create new column' section to create a new column named 'Years of Experience' by subtracting the values of the column 'Date of Joining' from the current date.

5.3 Screen 3: Visualization screen

Alice has her data ready. So she then uses the visualization screen (see Figure 4.8) to create a visual mapping to transform her data into a visualization. She thinks that the Number of tasks performed, Years of Experience, and Sex are important features and decides to use those features to create a visualization and perform machine learning classification only with those features [1]. To create the visualization, she first drags and drops the square icon onto the main panel. She then drags the column Years of Experience from the middle panel and assigns that value to the size parameter of the square. She then uses the bottom panel to create a mapper widget to transform values of the column Sex (0-male and 1-female) to blue and red colors respectively. She then drags the mapper output from the middle panel and assigns them to the color parameter of the square. Alice then drags and drops a circle into the main panel and drags and drops the column value 'Number of tasks performed' from the middle panel and assigns them to the radius of the circle.

[1] By doing this, Alice could introduce some bias in the system

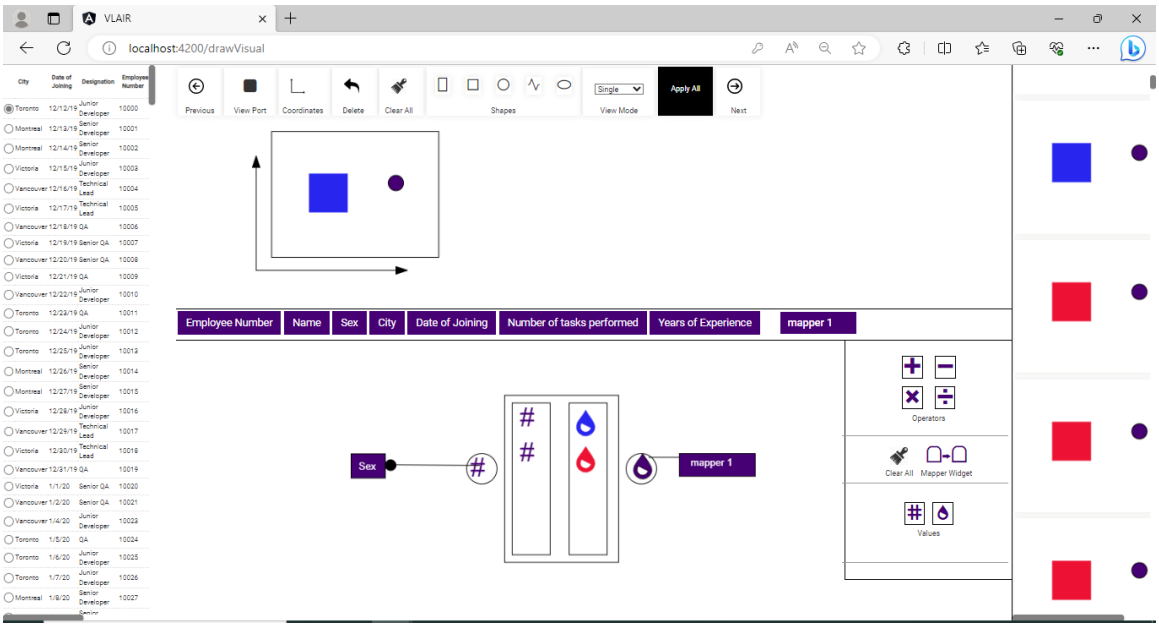


Figure 5.1: Genrated visualization

Next, she wants to verify if there are any data outliers and so she uses the view mode option and modifies the selection from 'single' to 'collective' to quickly view the visualizations created for all the rows of data and to see if there any data outliers [2]. Then she uses the view port to draw a rectangle outside the created visual mapping that she wants to use to transform her data into visualization and then she clicks on the apply all button to apply the created visualization into all the rows of data and then convert them into images. Alice makes use of the right panel to view the visualizations created for all the rows of data and is satisfied with the created visualization.

5.4 Screen 4: Machine learning screen

Next, she navigates to the machine learning screen (see Figure 4.16) and splits the generated images for training and testing datasets in a 70-30 ratio. She used this ratio because she found from [54] that 70-30 split is ideal for small datasets. She then uses those images to first train the model and then tests it. After testing is completed, Alice views the results. She is not happy with the results because there are more false positives. She wants to understand why and how the algorithm made that prediction

[2]This system can detect outliers, but it does not offer a way to correct them.

so that she can improve the results.

5.5 Screen 5: Compare results screen

In the compare results screen (see Figure 4.17), Alice added the images that gave different prediction results to the panels and carefully compared them. She identified that the algorithm fails in places where the range of values for a certain feature is so small that it does not result in sufficiently different looking images. She suspected that this might make it difficult for the algorithm to see differences. So she decided to go back to the visualization screen and modify the created visualization to add more space between square and circle so they do not overlap. After modifying the visual mapping and retraining and retesting the model, Alice achieved 80% accuracy. She was happy with the results and decided to use the model to classify new data. She ran validation with the allocated validation dataset to see how the model works for unseen data. The validation and the testing results were almost the same, and she understood that the model was not overfitting and is performing well. Overfitting occurs when the model performs perfectly on training set (observed data) and performs poorly on testing set (unseen data) [87]. Therefore, Alice decided to use iVLAIR in future because she can perform classification tasks more accurately without having to learn machine learning, and she can also explain why and how she gets the result.

Chapter 6

Implementation

This chapter describes the architecture and implementation of the system.

iVLAIR's client-side web framework is built using Angular as shown in Figure 6.1, which is built on top of JavaScript. It also relies on several external libraries to assemble its graphical user interface and perform geometric constructions. The main and bottom canvas of the visualization tool are built on top of HTML5 canvas. Fabric.js [88] is used to create and populate graphical objects on the HTML5 canvas element, such as drawing geometrical shapes, and JQuery is used to drag and drop the elements inside the canvas. The input column values are added to the middle panel as DOM elements. Additionally, the objects on the right panel inside the main canvas are implemented as DOM elements. Font Awesome [27] and Bootstrap libraries are used to style the graphical interface.

The machine learning API is built using Python. The Scikit-learn library [41, 60] is used to split the images for training and testing. The Pandas library is used to perform numeric operations and is built on top of Numpy [55]. The VGG 16 [72] image classification algorithm is implemented using Python programming and is used to solve data classification tasks. This algorithm is hosted using the Flask API and is consumed from the client side.

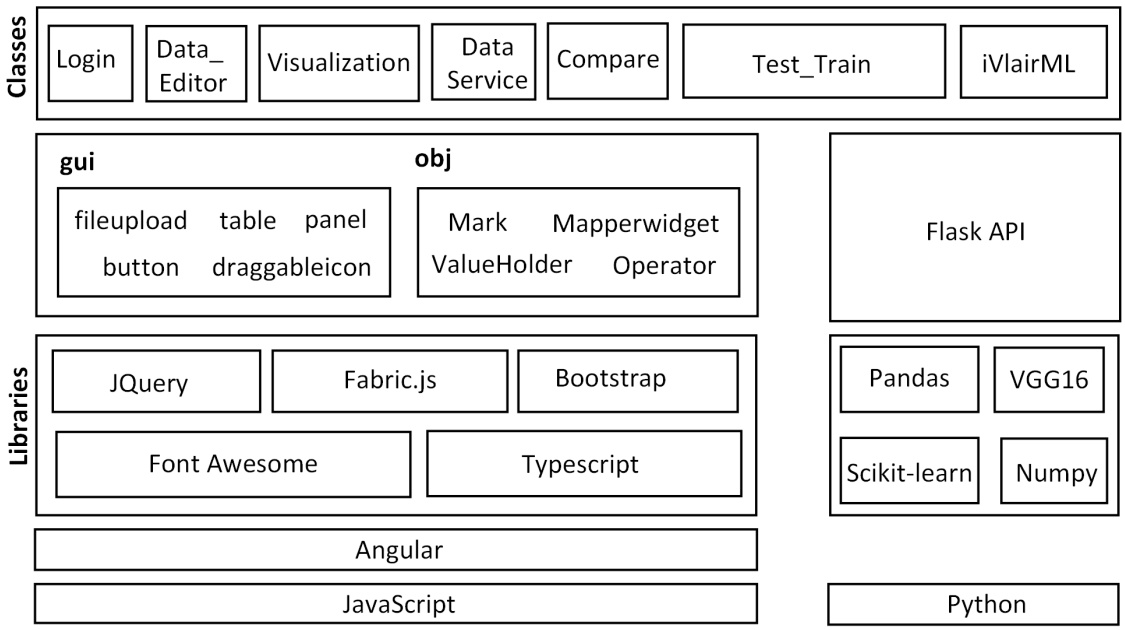


Figure 6.1: iVLAIR Architecture

Chapter 7

Evaluation

We conducted a study to understand and evaluate the usability and effectiveness of our tool from both expert and novice perspectives. We also aimed to determine whether the tool achieves the goals we designed it for. The evaluation is primarily qualitative, where participants are individually exposed to a task that simulates the creation of an ML model. The study was conducted after receiving approval from the ethics board at the University of Victoria.

7.1 Participants

We recruited 16 participants (6 female and 10 male) aged between 22 and 39 years old through email announcements and referrals from previous participants (the snowball method) [58]. The participants were from the University of Victoria and had varying backgrounds and expertise. Please refer to Table 7.1 for more details on the participants.

We chose all participants to have basic knowledge of Python programming. We expected participants to understand and write a basic Python code because they should be able to understand and modify the given Python code for one of the tasks that involved performing machine learning classification using an existing ML algorithm. During that task, we allowed them to experiment with the code and make changes to the given code or write their own code to try to improve the prediction results. However, we divided the participants into two groups based on whether they had prior experience working with ML algorithms. Participants were considered to have prior experience with ML if they had created at least one ML model or had

taken at least one ML course. We deliberately wanted two participant groups to know how each group felt about our tool and how useful they found it. Specifically, we wanted to see if the tool would be more useful for novices of ML technologies. After the participants agreed to participate in the study, we asked them through email if they had prior experience working with ML or not. Based on that information, we split the participants into two groups. Participants belonging to the first group (referred to as Non-ML experts) were only familiar with the term machine learning, but they didn't have any experience working on ML software, while participants of the other group (referred to as ML experts) have practical ML experience. The group of ML experts itself had a variety of levels of expertise. Among the ML experts, two were beginner programmers who were capable of working with ML software that were mostly from existing examples; two were intermediate programmers who could use complex ML software but didn't have an understanding of most parts of what the code does; four were sophisticated programmers who could understand most of the ML code and could write and run complex ML software independently.

The participants received a $15 incentive to participate in the experiment. The typical duration of the experiment for most participants was between 1 hour 10 minutes to 1 hour 15 minutes. However, the Research Ethics Board approved the study for approximately 1 hour.

Participants	**Level of Education**	**Educational background**	**ML Expertise**	**Levels of ML expertise**
P1	Masters	Computer Science	non-ML experts	
P2	Masters	Computer Science	ML experts	Intermediate Programmer
P3	PhD	Electrical Engineering	non-ML experts	
P4	PhD	Computer Science	ML experts	Beginner Programmer
P5	Masters	Engineering	ML experts	Intermediate Programmer
P6	Masters	Math and Computer Science	non-ML experts	
P7	Masters	Computer Science	ML experts	Sophisticated Programmer
P8	Masters	Computer Science	ML experts	Beginner Programmer
P9	Masters	Computer Science	non-ML experts	
P10	Masters	Computer Science	ML experts	Sophisticated Programmer
P11	PhD	Computer Science	ML experts	Sophisticated Programmer
P12	PhD	Computer Science	ML experts	Sophisticated Programmer
P13	Undergrad	Computer science	non-ML experts	
P14	PhD	Ocean Remote sensing	non-ML experts	
P15	Undergrad	Art History	non-ML experts	
P16	Undergrad	Linguistics	non-ML experts	

Table 7.1: Participants Details

7.2 Apparatus and Experiment Setup

The experimenter conducted the experiments on his/her personal laptop in a separate room ECS 431 at the University of Victoria. The laptop has an Intel i5-6200U CPU @ 2.30 GHz with 2 cores and 8 GB memory with a NVIDIA GeForce 940M. The software required for the experiment includes Angular, node.js, python, numpy, flask, opencv-python, tensorflowpip, matplotlib, Flask-Cors, pandas and torch torchvision torchaudio. During the experiment the experimenter sat next to the participants to guide them through the process.

7.3 Study Procedure

The experiment followed a within-subject design where all participants completed the same set of tasks. During the experiment, an equal number of machine learning experts and non-experts were given either the iVLAIR tool or the Python approach for Task 1 and the other tool for Task 2. The order in which the participants were allocated the tools was counterbalanced. Throughout the experiment, each participant was guided by the experimenter to complete the study individually while following the 'think aloud' protocol.

The experiment is divided into the following subsections:

1. **Consent form and Introduction:** When participants agreed to participate in the study, we sent them an online consent form via email one day before the experiment. At the beginning of the experiment, we presented them with the general study goal and asked them to sign the printed consent form if they agreed to participate.

2. **Demographic form:** Then, we asked the participants to fill out a printed demographic form through which we collected their general demographic information and background information such as age, sex, educational qualification, and experience with Python programming and machine learning.

3. **Task 1:** During task 1, we gave half of the participants the iVLAIR tool while the rest were given the python code (random forest classifier algorithm). We chose to use the random forest classifier algorithm because most people would

have already learned the algorithm in their machine learning courses, and it would be easy for someone who is new to machine learning to understand the algorithm. These tools are randomly assigned to the participants. We also gave them the water potability dataset and asked them to classify if the given water body is safe or unsafe for drinking using the given tool. Task 1 consists of:

A. **Phase 1:** During this phase, we split the overall process into four steps and asked participants a couple of questions to understand their previous knowledge on the process. We then provided instructions to execute each step.

B. **End of Phase 1 Questions:** At the end of phase 1, participants answered a few questions about their overall interpretation of the tool, their thoughts on the prediction results, and what they would do to improve the results.

C. **Phase 2:** During the next phase, we gave participants 10 minutes to freely experiment with the given tool. They tried to replicate what they did previously or tried their own way to classify the given dataset using the given tool and improve the prediction results.

D. **End of Phase 2 Questions:** At the end of phase 2, participants answered a few questions about their new understanding of the tool after exploring it, their understanding of the change in results, their new ideas for improving the results, and what they would do to improve the results if they had more time.

E. **Phase 2 Questionnaires:** Additionally, participants completed an SUS questionnaire for the given tool at the end of phase 2.

4. **Task 2:** Task 2 had the same structure as Task 1. However, for this task, we gave the participants the iVLAIR tool who were given the python code for task 1 and vice versa. We also provided the heart attack prediction dataset and asked the participants to classify whether patients had a possibility of heart attack or not using the given tool. This task also had the following subphases: A. Phase 1, B. End of Phase 1 Questions, C. Phase 2, D. End of Phase 2 Questions, E. Phase 2 Questionnaires

5. **Final questions and debrief:** At the end of both tasks, we asked the participants few post-study questions about the main differences, pros, cons and their

preferences for each of the two tools they used during the tasks. Finally, we verbally debriefed and thanked the participants for their participation at the end of the experiment.

7.4 Measurements and analysis

We collected data of qualitative and quantitative types:

1. We recorded audio of the entire process and captured videos of the on-screen activities.

2. We performed combined inductive and deductive coding on the qualitative data. First, we created a set of initial codes with a focus on interview question themes such as the pros and cons of the two tools and user preference before analyzing the collected data. Then, we transcribed the recorded audio and analyzed the collected data for each participant using an 'open coding' approach and added more codes. Finally, our research team refined and finalized the codes.

3. We collected System Usability Scale (SUS) data using a questionnaire (see Table 7.2). SUS is a composite measure of the overall usability of the system being studied, and the SUS score was calculated by summing the score contributions from each item [13]. We performed quantitative analysis with the bootstrapping method. We then interpreted the results using effect sizes and confidence intervals (CIs) rather than $p-$values, following recommendations for statistical practices in HCI and visualization (e. g., [18, 21]).

Q1	I think that I would like to use this system frequently.
Q2	I found the system complex.
Q3	I thought the system was easy to use.
Q4	I think that I would need the support of a technical person to be able to use this system.
Q5	I found the various functions in this system were well integrated.
Q6	I thought there was too many inconsistency in this system
Q7	I would imagine that most people would learn to use this system very quickly
Q8	I found the system very awkward to use
Q9	I felt very confident using the system
Q10	I needed to learn a lot of things before I could get going with this system

Table 7.2: SUS questionnaires used in this study.

Chapter 8

Results

In this chapter, we present both the quantitative analysis of participants' self-rated System Usability Scale (SUS) scores and the qualitative analysis results of their answers and comments.

8.1 Quantitative SUS Analysis Results

We collected System Usability Scale (SUS) questionnaire data for iVLAIR and Python approach separately from each of the 16 participants. We then calculated the overall SUS score for all the questions and then separately for each of the questions in the subsections below.

8.1.1 Overall Scores

First, we compute the average aggregated SUS score between iVLAIR and Python approach for all participants, as well as for ML experts and non-ML experts separately. Results are presented in Figure 8.1.

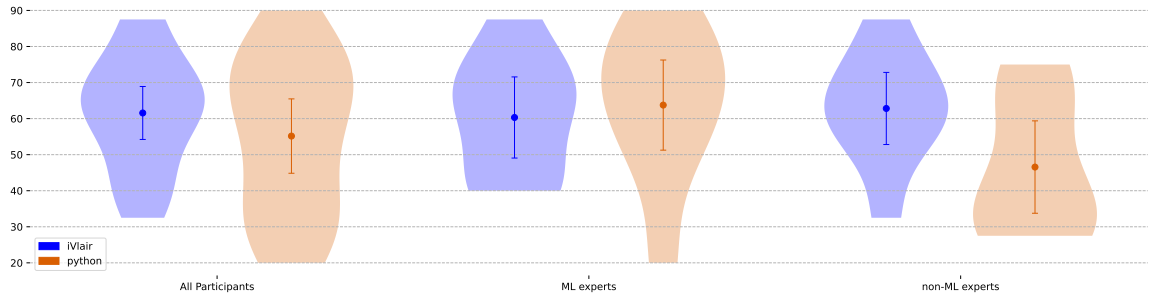


Figure 8.1: Average aggregated SUS scores. Error bars represent 95% bootstrapped CI.

From the above graph, we can see that the mean SUS scores of iVLAIR and Python approach for all participants are 61.56 [54.22, 69.38] and 55.16 [44.69, 65.63] respectively. This indicates that iVLAIR has a higher mean score than the Python approach. The mean SUS scores for iVLAIR and Python approach for non-ML experts are 62.81 [52.81, 73.75] and 46.56 [33.44, 58.44] respectively, which also shows that iVLAIR is more preferred. However, a different pattern is observed for ML experts, with a mean of 60.31 [49.06, 70.94] and 63.75 [50.94, 78.75] respectively. This suggests that the Python approach is more favored by ML experts than iVLAIR.

8.1.2 Overall Difference in Scores

Next, we compute the average aggregated SUS score difference between iVLAIR and Python approach for all participants, as well as for ML experts and non-ML experts separately. We also report the average aggregated difference in SUS score between ML experts and non-ML experts for iVLAIR and Python approach separately. Results are presented in Figure 8.2.

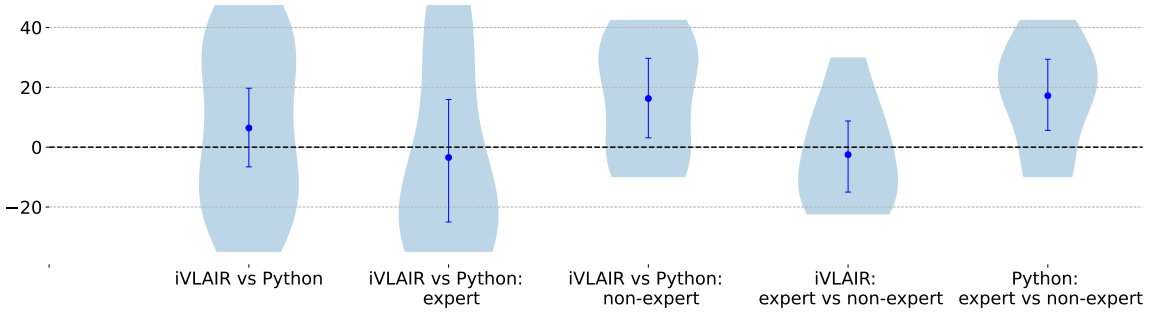


Figure 8.2: Average aggregated difference in SUS scores. Error bars represent 95% bootstrapped CI.

From the above graph, we can see that for non-ML experts, the mean difference in SUS score between iVLAIR and Python approach is 16.25 (clearly greater than 0) which shows that iVLAIR scores are higher than Python. The confidence interval (CI) is ([3.43 29.68]) which does not overlap with 0, indicating statistically significant evidence for the existence of such a difference. For all participants and for ML experts separately, the mean SUS score differences between iVLAIR and Python are 6.41 ([-7.03 19.53]) and -3.44([-23.31 15.63]) respectively. Their mean values are close to 0 and their CI values largely overlap with 0. This shows that there is no evidence to support the difference.

When we compare the scores between the two user groups for each technique, the mean SUS score difference for iVLAIR approach between ML experts and non-ML experts is -2.5 (CI [-14.6875 8.4375]). The mean value is close to 0 and the CI largely overlaps with 0. So, we do not find evidence to support a difference. However, for Python approach, the mean difference is 17.1875 (CI [5.625 29.375]). The mean is greater than 0 and the CI does not overlap with 0. This shows that the experts scores are higher than non-ML experts and the evidence to support such a difference is strong.

8.1.3 Differences Between iVLAIR and Python Approach for Individual Questions

Next, we conducted the same analysis as in the previous section. However, we made comparisons separately for each of the 10 questions. The results of the comparison are shown in Table 8.1. Scores with significant differences are represented with a double

red asterisk (**), while scores with close to significant differences are represented with single red asterisks (*).

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
iVLAIR vs Python: all participants	Mean: 0.13 CI: [-0.56, 0.75]	Mean: 0.19 CI: [-0.63, 0.94]	Mean: -0.25 CI: [-1.25, 0.75]	Mean: -0.44 CI: [-1.44, 0.56]	Mean: 0.56 CI: [0.19, 0.94]**	Mean: -0.19 CI: [-0.75, 0.38]	Mean: 1.13 CI: [0.5, 1.69]	Mean: -0.25 CI: [-0.88, 0.38]	Mean: 0.13 CI: [-0.63, 0.81]	Mean: -0.19 CI: [-1.19, 0.81]
iVLAIR vs Python: Experts	Mean: -0.38 CI: [-1.25, 0.5]	Mean: 0.38 CI: [-1.13, 1.88]	Mean: -1.13 CI: [-2.63, 0.25]*	Mean: 0.13 CI: [-1.5, 1.75]	Mean: 0.63 CI: [0.13, 1.13]**	Mean: 0.25 CI: [-0.5, 1.0]	Mean: 1.25 CI: [0.5, 2.13]**	Mean: -0.38 CI: [-1.5, 0.75]	Mean: -0.5 CI: [-1.5, 0.38]	Mean: 0.88 CI: [-0.38, 2.25]
iVLAIR vs Python: non-Experts	Mean: 0.63 CI: [-0.25, 1.38]	Mean: 0.0 CI: [-0.5, 0.5]	Mean: 0.63 CI: [-0.38, 1.75]	Mean: -1.0 CI: [-2.13, 0.25]*	Mean: 0.5 CI: [0.0, 1.0]*	Mean: -0.63 CI: [-1.38, 0.13]	Mean: 1.0 CI: [0.0, 1.75]*	Mean: -0.13 CI: [-0.63, 0.38]	Mean: 0.75 CI: [-0.13, 1.63]*	Mean: -1.25 CI: [-2.25, -0.38]**
iVLAIR: Experts vs non-Experts	Mean: 0.5 CI: [-0.38, 1.25]	Mean: -0.25 CI: [-1.13, 0.63]	Mean: -0.5 CI: [-1.63, 0.5]	Mean: 0.25 CI: [-0.75, 1.25]	Mean: 0.25 CI: [-0.13, 0.5]	Mean: 0.25 CI: [-0.38, 1.0]	Mean: -0.38 CI: [-1.63, 0.63]	Mean: -0.13 CI: [-0.63, 0.5]	Mean: 0.0 CI: [-0.75, 0.75]	Mean: 0.75 CI: [0.13, 1.5]**
Python Experts vs non-Experts	Mean: 1.5 CI: [0.5, 2.5]**	Mean: -0.63 CI: [-1.38, 0.13]*	Mean: 1.25 CI: [0.5, 2.0]**	Mean: -0.88 CI: [-2.25, 0.5]	Mean: 0.13 CI: [-0.38, 0.63]	Mean: -0.63 CI: [-1.25, 0.0]**	Mean: -0.63 CI: [-1.38, 0.25]	Mean: 0.13 CI: [-1.0, 1.0]	Mean: 1.25 CI: [0.13, 2.38]**	Mean: -1.38 CI: [-2.5, -0.38]**

Table 8.1: Mean and CI for the average difference in SUS scores for each of the questions.

1. Q1: "I think that I would like to use this system frequently" (see A in Figure 8.3)-[]

   The mean SUS score difference between experts and non-Experts for Python approach for Q1 is 1.5 (clearly greater than 0). The CI is [0.5 2.38] which does not overlap with 0. This shows that the experts scores are higher and the evidence is significant.

2. Q2: "I found the system complex" (see B in Figure 8.3)

   For Python approach, the mean difference between experts and non-experts score is -0.63 (clearly less than 0). This shows that non-experts scored higher than experts in terms of system complexity. The confidence interval is [-1.38, 0.13], which partly overlaps with 0. This shows that evidence is close to significant.

3. Q3: "I thought the system was easy to use" (see C in Figure 8.3)

   The mean difference in SUS score between experts and non-experts for Python approach is 1.25. The mean is clearly greater than 0 and this shows that the experts scored higher than non-experts. The confidence interval is ([0.5, 2.0]) which does not overlap with 0. This provides a significant evidence to support the difference.

   When comparing the mean difference of -1.13 between iVLAIR and Python for experts we found that the mean value is clearly less than 0. This shows that the Python scores are greater than iVLAIR. The confidence interval is [-2.63, 0.25] which partly overlaps with 0. This shows that the evidence to support the difference is close to significant.

4. Q4: "I think that I would need the support of a technical person to be able to use this system" (see D in Figure 8.3)

   The mean difference between iVLAIR and Python approach for non-experts is -1.00 (clearly less than 0) and CI is [-2.13, 0.25]. When comparing the mean value, we found that the Python scores are higher and as the CI partly overlaps with 0, the supporting evidence is close to significant.

5. Q5: "I found the various functions in this system were well integrated" (see E in Figure 8.3)

The mean values between iVLAIR and Python for all participants, experts and non-experts are 0.56 [0.25, 0.94], 0.63 [0.13, 1.13] and 0.50 [0.00, 1.00] respectively. In all cases mean values are clearly greater than 0 which shows that the iVLAIR scores are higher than Python. The CI do not overlap with 0 for all participants and experts which shows that the evidence to support these differences is significant. Whereas the CI touches 0 for non-experts and so the supporting evidence is close to significant for non-experts.

6. Q6: "I thought there was too many inconsistency in this system" (see F in Figure 8.3)

   The difference in mean and CI for experts and non-experts for Python is -0.63 and [-1.25, 0.0] respectively. The mean value is less than 0 which shows that the non-experts score is higher than experts. The confidence interval overlap with 0 which shows that the evidence to support the difference is close to significant.

7. Q7: "I would imagine that most people would learn to use this system very quickly" (see G in Figure 8.3)

   The average score differences between iVLAIR and Python for all participants and also for experts and non-experts separately are 1.13 [0.5, 1.69], 1.25 [0.5, 2.13] and 1.0 [0.0, 1.63] respectively.We found that the mean values are greater than 0 in all cases and so the iVLAIR scores are higher than those of Python. Also the CI's for all participants and experts do not overlap with 0 and the evidence to support these differences is significant. However, CI for non-experts touches 0 and the evidence to support the differences is close to significant.

8. Q8: "I found the system very awkward to use" (see H in Figure 8.3)

   We did not find any significant difference in the scores.

9. Q9: "I felt very confident using the system" (see I in Figure 8.3)

   The mean differences between experts and non-experts for Python approach is 1.25 and it is clearly greater than 0. This shows that the experts score is higher than non-experts. The CI is [0.13, 2.38] which doesn't overlap with 0. This shows that the supporting evidence is significant.

   The mean difference between iVLAIR and Python for non-experts is 0.75 and the CI is [-0.13, 1.63]. The mean is greater than 0 which shows that the iVLAIR

scores are higher than Python and the CI partly overlaps with 0 which shows that the evidence is close to significant.

10. Q10: "I needed to learn a lot of things before I could get going with this system" (see J in Figure 8.3)

    The mean difference and CI between iVLAIR and Python for non-experts is -1.25 and [-2.25 -0.25]. The mean value is clearly less than 0 which shows that the Python scores are higher. The CI do not overlap with 0 and so the evidence to support the difference is significant.

    We then compared the mean score differences between experts and non-experts separately for iVLAIR (0.75([0.13 1.38]) and Python (-1.38 ([-2.5 -0.38])). From that we found that for iVLAIR, the mean value is greater than 0 which shows that the experts score is higher than non-experts. However for Python, the mean value is less than 0 which shows that the non-experts score is higher. The CI in both these cases do not overlap with 0 and so the evidence to support these differences are significant.

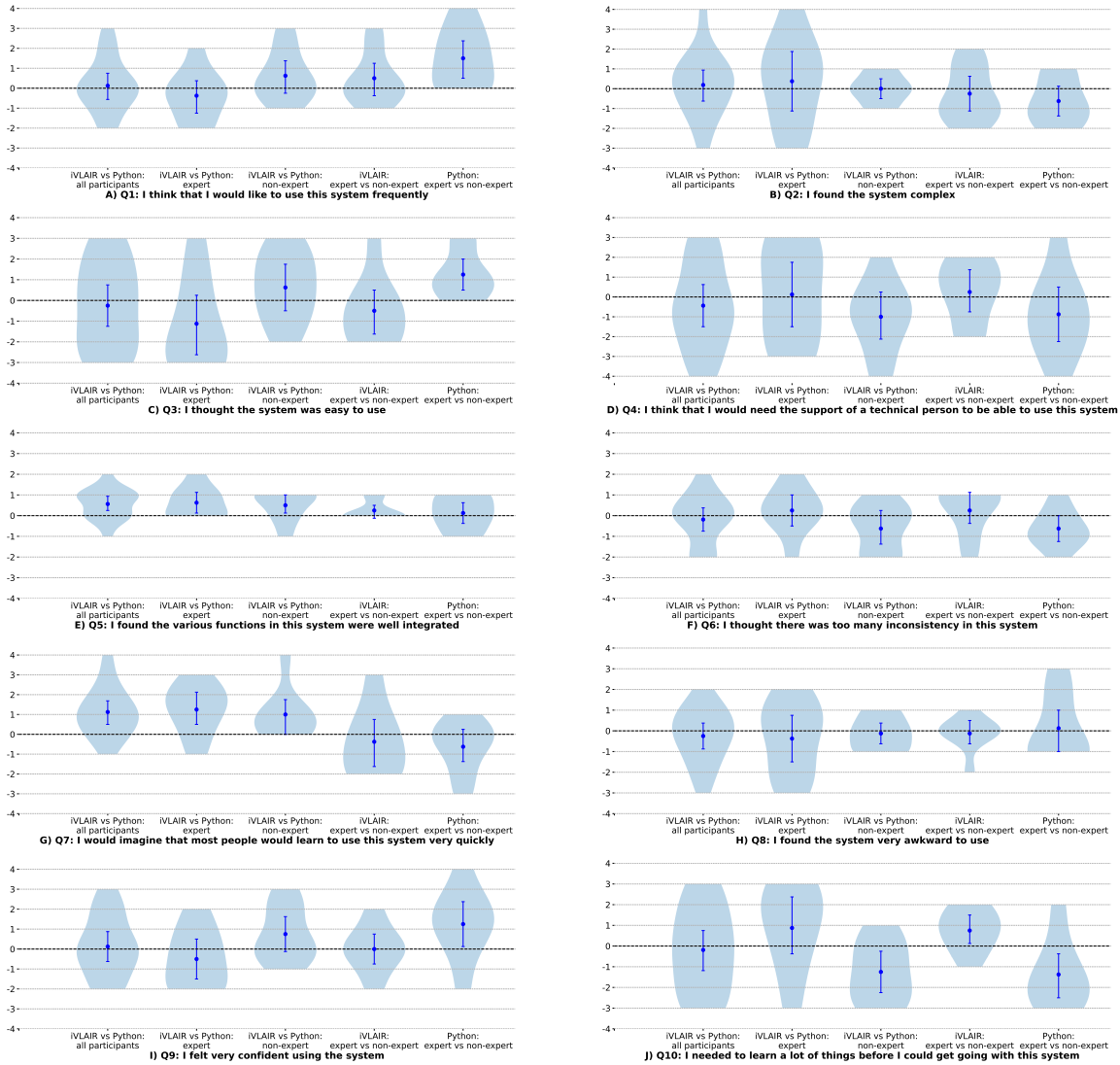


Figure 8.3: Average difference in SUS scores for questions Q1 to Q10. Error bars represent 95% bootstrapped CI.

8.2 Qualitative Analysis Results

To generate qualitative results, we analyzed the data using a thematic analysis approach. We conducted experiment with 16 participants and the collected data in audio format. We first created a set of initial codes focusing on pros and cons of iVLAIR tool and the python approach to ML classification and the user preferences before analyzing the collected data. We then transcribed the recorded audio and analyzed the collected data for each participant and then then coded them using NVivo

software using an 'open coding' approach. We used an inductive approach to identify themes and sub-themes that emerged from the data. We then reviewed and refined these themes until we finalized on the themes. We created a total of 45 codes, 10 sub-themes and 6 themes.

In the following subsections, we will discuss the advantages and disadvantages of using iVLAIR compared to the python approach for machine learning classification for each of the themes we created. We will also provide evidence from the participants to support our claims and discuss user preferences.

8.2.1 Understanding Data

Every participant mentioned at some point that **iVLAIR** was helpful for them to understand the data and to find data patterns when looking at the small multiples (*"because like even when I was seeing for 20 seconds, I started seeking like a pattern [...]"* [P1], *"it's easier to see when I have visuals to compare more quickly than just numbers* [P12]). P8, an ML expert, specifically mentioned that it was useful for detecting outliers (*"I think it's good for detecting outliers maybe. If you go through all the data set and you see one point that is like really far and does not relate to any other points [...]"*). Six participants: 3 experts (P2, P5, P8) and 3 non experts (P1, P6, P13) said that it was easy to understand why a particular prediction is made and to debug the model (*"I think that one could be used when you are stuck and you don't know what is wrong with your model"* [P8], *"I think maybe debugging is easier there, like understanding patterns is easier, so patterns in data is easier in the visualization"* [P1]). However, P14 said that they needed more time to identify the data patterns *"I need some more time"*).

Alternatively, when using the **python** approach, 12/16 participants were not able to either understand or find patterns in data. For example, P4 said that *"It's hard for me to understand from the data"* and P1 said that *"I would need time to look at them because it's hard to figure out if there is a similarity because I need to go through the numbers at-least one by one"*. Additionally, P5 could not understand why the model made that prediction. *"I can't understand why the false positive values have occurred here."*. In contrast, four other participants (P4, P7, P8, P10) who were all ML experts said that they were able to find at least some patterns in data (*"Not much. But yes, It does"* [P10]).

8.2.2 Prior Knowledge

7 out of 16 participants (P2, P5, P6, P7, P11, P12, P15) felt that **iVLAIR** helps people learn machine learning by allowing them to perform machine learning classification without having to write any code *"I think it would benefit new users or new people that are learning machine learning"* [P12]. It has some built-in functions and is more suitable for non-programmers *"help you visualize the data like easily and you do not really have to know how to code. [...] It has a lot of built-in features that you can use easily without programming.[...]"* [P11]. Of these 7 participants, 3 (P5, P6, P7) felt that even though the tool helps non-coders, they should have some prior knowledge on input data *"could be useful in a very niche setting where you have a very specific kind of data set that you already sort of know something about."* [p6] or they need to spend some time learning the application as a whole *"I think it might take a while to understand and use the system well I think that would take some time"* [P7]. Out of these 3 participants, P6 was a non-ML expert. In addition, 4 out of 16 participants, including an equal number of experts (P4, P8) and non-experts (P1, P3), felt that they needed to have some knowledge of the data (*"you need to learn it like from scratch what every variable means"* [P4]) and visualization (*"... requires some knowledge like what does the size mean? what does the color mean? what does the position mean? For them to interpret that information and map them all together, they should know what a visual variable is and what a visual mark is"* [P8]) and also some support to learn to use the tool (*"visually could be helpful, but it needs some technical support before that, so you can learn that tool"* [P8].

On the other hand, 10 out of 16 participants felt that using the **python** approach for machine learning classification requires sound programming knowledge. Among them, seven participants (P2, P4, P5, P7, P8, P10, P12) were experts and three were non-ML experts (P6, P14, P15). However, we did not find any difference in results between these participant groups. *"obviously this needs some learning. [...] if someone who is not really aware of python, they would have difficulty"* [P2]. *"I definitely need a lot of support from someone else. Technical person or somebody"* [P14]. *"If you don't know coding then you can't use it"* [P15].

8.2.3 Easy to Use

When describing how easy it is to use iVLAIR approach for machine learning classification, 4 (P1, P3, P5 and P10) out of 16 participants said that **iVLAIR** was intuitive

because of the appropriate choices of visual elements and the user-friendly interaction design. For example, P? mentioned that (*"I think labeling using the blue and red makes it easier for me to remember which was true and which was incorrect"*). Also, the user controls were considered to be intuitive, simple, and easy to understand: (*"I don't have to worry about anything. I simply have to just drag and drop the UI elements and I'm done"* [P3]) and only a button click away (*"This tool is easier for me because everything is given here and it's just a button click away from me"*[P5]).

Three participants (P1, P4, P12) found the **python** approach to be a step-by-step process with everything on one screen, so they didn't have to navigate to another screen. Participant [P8] said, *"If I wanted to do it like in a simple script, I just use this one"*. Another participant [P13] said, *"it was easier because it's basically in numbers and it's easier to calculate and make predictions"*.

8.2.4 Reusability

Two participants (P12 and P15) said that once they write the code using **python** approach to machine learning, they don't have to rewrite it again for another input. Instead, they can use the same code for different inputs. One of the two participants said, *"if you don't know the code, then you don't know the code. But if you do know, then it's very repeatable"* [P15]. However, for **iVLAIR**, participant P15 said that they had to create a new visualization for every input (*"But with this, you might have to create a new visualization for every input"*).

8.2.5 Flexibility

Out of 16 participants with an equal number of ML experts (P2, P10, P11) and non-ML experts (P3, P6, P7) said that **python** is very flexible and adaptable. They can use different models or have a lot of options to improve the prediction results. For example, P11 said that *"the pros of python is that you can, you know, do whatever you want with the data, build any different type of model"* and P3 said that *"with code there are many options to try"*. In addition, 3 out of those 6 participants (P2, P3, and P10) felt that **iVLAIR** was a bit constrained when it comes to multiple options to improve the results (*"you can do a lot more by actually coding than just visualizations"* [p7], *"And if the tool is not giving good results, I don't think there is much that can be done with the tool. But with code there are many options to try "* [P3]).

8.2.6 Familiarity

One participant [P12] suggested that using the **python** approach to solve machine learning problems would be easier since they are already familiar with it, instead of learning and using a new tool. Three other participants (2 ML experts [P5, P8] and 1 non-expert [P9]) mentioned that the use of the python approach is advantageous because there are many resources available online that they can use to learn it. However, participant [P8] mentioned that there are not enough resources available for **iVLAIR** as it is a new tool: *"I'm comfortable with it and there's a lot of resource on the Internet that I can use. But with that one, there's no resource. Just a new tool."*.

8.2.7 Preferences

At the end of the study session, we asked participants when they would prefer to use iVLAIR over python and vice versa.

ML experts

Half of the machine learning experts (4 out of 8) preferred iVLAIR while the other half (4 out of 8) favored python. However, one of the four participants who favored python mentioned that iVLAIR approach is interesting as it makes it easier to understand data: *"I think if I am experienced programmer coder in machine learning the previous jupyter notebook should be the first choice because it is verbose, but somehow we are used to those lines. We can reuse and save code. And but at the same time, it's hard to see the data set using only tables. So that is the reason this second tool is interesting and should be a good option when we are trying to understand the data. Because only looking the numbers is not enough, so having the ability to see each instance creating these visuals is really interesting."* [P12]. Another participant preferred python approach because of its flexibility, but considered iVLAIR approach a better approach for non-programmers in learning Machine learning: *"because I do know python [...] I would prefer that because there's different things you can do as far as like normalizing the data. But to anyone who doesn't know how to code, I would surely recommend this [...]. I think it helps people understand machine learning better."* [P11].

Most of the participants found that the purpose determined the choice of the tool. iVLAIR was considered useful to visualize and understand the data: *"Here, I'm just*

imagining things how they are going to be and then I'm just trying to imagine how the model is going to work and not how it's going to give me the reasons. But there I can see the data. I think it makes a huge difference. I can see the data, I can see how and why it's right and why it is wrong." [P2], to identify the data outliers: *"I think it's good for detecting outliers. I think I would use it with that that case. Don't want to look at the numbers. Numbers are confusing to me, but if you look at the picture bigger picture, you can definitely like hit the spot. This is what what's wrong with my model or something like that"* [P8], to debug the model and to perform data transformation: *"This tool is very strong with data transformations. In the previous tool, if I want to transform my data set it, it will take me forever [...]. You know, I have to have a lot of knowledge to be able to do that, but in this tool, the data transformation part was quite intuitive"* [P10]. python approach was considered when participant choose to use a simpler approach to solve ML classification problem: *"If I wanted to do it like in a simple script, I just use this one"* [P8].

One participant suggested a hybrid approach that uses both tools. They found iVoLVER to be good for understanding the data and preferred using the python approach to implement the ML model: *"If I just want to take a quick overview of the data and just if I want to understand the data set, I think first system would be really nice for actually understanding the data set and I think the second system would be good for implementing The ML model after you understand the data set."* [P7].

Non-ML experts

Out of 8 non-ML experts, 6 preferred the python approach. One of the participants considered using python because they are familiar with coding but preferred the iVLAIR approach for model debugging: *"I think again, if I were doing something related to if I'm debugging then maybe that tool over this. But if I'm doing like operations and stuff then maybe this because I'm used to it"* [P1]. Another participant considered iVLAIR to be easy and more useful to solve smaller problems whereas python was considered to solve complex problems: *"I will still use traditional ml to solve very hard problems. If it is a small problem, I would go for the visualization approach because it provides easy interface where I don't have to worry about writing code."* [P3]. Finally, three of these six participants mentioned that they would consider using the python approach because they find the process simpler, faster and more suitable for unfamiliar data. However, they said that they would choose iVLAIR

if they had prior knowledge of data and visualization: *"I think what I was trying to say is I feel like this sort of low code approach could be useful in a very niche setting where you have a very specific kind of data set that you already sort of know something about. Whereas the more flexible programming interface is more useful in a broader setting where we don't already know what we're trying to do"* [P6]; *"I prefer the first one [iVLAIR tool] when I have some raw data and I'm not very familiar. With first process, so I just need to pass the files like the row data data files into the machine and it will provide the prediction results for us. In that case, it will be convenient to have that one. I prefer to use the second one when I'm kind of like have experience with visualization and then I can understand from the visualization the edge case that the machine Cannot perform."* [P13]; *"I might like the visualization one because that like I know exactly which variables I'm trying to put in to measure. whereas here [...] python seems faster"* [P16].

Two other participants preferred the use of iVoLVER because they find it easier to understand, manipulate and analyze data. Also they find the user controls more easier to use: *"I think. I'm actually not sure I I don't see a situation where I would prefer the jupyter one over this one, but that's mainly because I like playing with it and it's actually easier to manipulate The data as well"* [P9]; *"maybe I would go for the visualization tool because may be it's easy for me to like Just look through the data and understand what it is actually going on. And I can make modifications to the visualization. I can select different columns whatever the columns, I can just drag it and it's kind of more useful for analysis"* [P14].

Chapter 9

Discussions

VLAIR [36, 37] is a novel method that transforms input data into visualizations (pixel images) similar to those used by humans for data analysis and communication. These images can be used as input for convolutional neural networks, which are adept at extracting visual features. VLAIR was initially applied to the Human Activity Recognition (HAR) tasks but are capable of applying to other classification tasks. Our iVLAIR tool is designed on top of VLAIR to generalize the advantages of VLAIR to a larger population and for other datasets. To achieve this, we developed a single application that allows users to upload an input dataset (Our tool can handle input data in .csv or .xslx formats, as long as it does not exceed 3000 rows and 50 columns.), represent the input data as visualization using an inbuilt visualization tool and then use those visualizations to perform data classification using an image classification algorithm. The interface of iVLAIR's visualization tool adopts elements from iVOLVER.

We designed iVLAIR to allow participants to query iteratively (DP4), enable continuous improvement (DP3) and leverage visuals as much as possible (DP2) so that machine learning classification tasks would be more interpretable (G1) and to facilitate sense-making of machine learning results and iterative improvement of models (G3). Our study results indicates that, for many participants, it was easier to understand the data and its patterns. The study also provides some evidence that the tool supports people's understanding of how and why the model has reached a particular prediction which in turn helps to debug the model. However, the study also showed some of the limitations of the approach. For example participant [P8] said *" Well like if I had just this this picture of blue square, I would understand it {...}. But when there are more colors and different positions that are away from the coordinates, then*

it's hard for me to understand it because it's away from the coordinates and it has a different color, which I don't know what it means.{...}. So it's better if I see a legend {...}"

Goal 2 (To help users to perform data classification interactively without (learning) coding) was addressed by applying three design principles: Support operation without coding (DP1), Avoid unnecessary complexity (DP5), Avoid menus and hidden operations (DP6) and Avoid reliance on visual memory (DP7). The study found that the tool was less dependent on having to learn to write code. However, our study results also showed that iVLAIR is not a tool that can be used without prior learning. It has a learning curve of its own, which may create some overhead for learners. The challenges with iVLAIR related to learnability are that it requires users to have an understanding of data, visual marks, visual variables, and visual mapping and the kind of influences it might have with the machine learning result. For example, participant [P8] said *"One requires some knowledge like what is, what does the size mean? what does the color mean? what does the position mean? For them to interpret the information and map them all together, they should know what a visual variable is and what a visual mark is".*

We think that the current design has some unexpected benefits that we did not plan for or aim at. iVLAIR enables users to complete the whole machine learning workflow from data preparation to result evaluation. This helps the non-ML users to comprehend how machine learning works. Therefore, iVLAIR might be helpful for teaching machine learning. Also, one of the participants in our study mentioned that iVLAIR improves their understanding of machine learning: *"I think it's just like it helps people understanding machine learning better"* [P11].

From the quantitative analysis results, while comparing the overall SUS score differences between iVLAIR and python for non-ML experts, we found that iVLAIR scores are higher than python with a significant difference. Also, when comparing the diferences between experts and non-experts for Python approach, we found that experts scores for python are higher than non-experts score with a significant difference. From this we interpret that non-experts would prefer to use iVLAIR when they are not very comfortable with the python approach. However, our tool might be of limited use to experts.

Furthermore, our study results raised several important questions. When a larger dataset is used as input, what happens? Will the application's performance be affected? Will data transformations take longer? And perhaps more importantly, will

the visualization approach scale when the data is very large, both in terms of features (columns) and cases (rows)?. To handle large datasets and to provide meaningful insights and patterns from the data, implementing expandable canvas to create visualization might help.

Further research is needed to determine whether adopting UI elements from iVOLVER over tools such as Tableau is a good or bad choice. Although tools such as Tableau are available, they are not suitable for small multiples. With the design of iVOLVER, we were able to create visualizations for individual rows of data.

When we compared iVLAIR to other approaches, we found it to be unique. Model-specific and model agnostic methods such as : Class Activation Mapping (CAM) [72], Gradient-weighted Class Activation Mapping (Grad-CAM) [57, 56], Guided Grad-CAM [56], Layer-wise Relevance Propagation (LRP), DeepLIFT, LIME, SHAP (mentioned in Section 2.3) all uses visualization methods to explain the model predictions. In addition, Partial Dependence Plots (PDP), Individual Conditional Expectation (ICE), Accumulated Local Effects (ALE) and Feature Interaction Plots are visualization methods that help identify the effect a feature has on the predicted outcome of a machine learning model. These techniques either work at the instance level or describe the average behavior of a machine learning model. They provide post-hoc explanations meaning that they explain the predictions of a model after it has been trained. However, iVLAIR allows users to understand the data and its patterns for each row of data even before training the model. This helps users identify the feature that might contribute to the model prediction. If no visible pattern is identified, users can modify the visualization until they are satisfied. This helps them improve the results.

9.1 Technical Constraints and Future Work

1. If the user wants to choose a different algorithm to improve the prediction results, iVLAIR does not currently have an option that allows users to choose an algorithm of their choice. However, in the future, we might include a drop-down with a list of high-performance image classification algorithms from which the users can choose their desired algorithm.

2. Data pre-processing is an important step to achieve good classification performance. One of the essential pre-processing steps is data normalization.

Data normalization involves transforming features into a common range so that greater numeric feature values cannot dominate the smaller numeric feature values. Currently, iVLAIR does not support data normalization. However, we might be working on expanding our tool to support normalization in the future.

3. If a user is using the iVLAIR visualization tool to create a visual mapping and the page refreshes for some reason, their entire work will be erased and they will have to start over. We might be planning to redesign our system so that users can save their work and resume where they left off in the future.

4. In iVLAIR, when a large dataset is used as input, the process of representing the data as visualizations and converting them to images takes longer to execute on a web browser. To reduce the processing time, we plan to implement multiple threads for parallel processing. To achieve this, we may implement a back-end server in the future.

   Also, loading a large number of images on a web browser can be difficult. To address this issue, we might implement techniques such as dynamic loading or pagination on page scroll in the future.

5. In the visualization screen, we may add features in the future that will allow users to calculate the average, mean, and median of a training set. Users can map these generated values to the created visual marks to quickly identify if the data falls above or below these values. This helps users better understand data patterns.

Chapter 10

Conclusion

Machine learning algorithms are widely used in various applications, but they require expertise and knowledge to use effectively. This makes it challenging for people with different backgrounds to use them. However, there are several no-code or low-code tools available to address this issue. Additionally, most machine learning algorithms act as black boxes, which makes it difficult for humans to understand why the algorithm has arrived at a particular result. There are many ways to explain the internal workings of the model, but implementing these approaches also requires some knowledge and technical expertise.

To solve this problem, we designed and implemented a web application tool iVLAIR, which is designed on top of the VLAIR approach to take advantage of deep learning and visualization techniques to solve classification problems. The tool allows users to perform machine learning classification without the need of learning to write textual programming. The tool can also be used to make the machine learning process more interpretable by making it easier to understand the data and to improve the model over time.

We conducted a study with 16 participants (8 experts and 8 non-experts) and presented the results. The study highlighted the tool's usability, advantages, and disadvantages from both expert and novice points of view. Furthermore, the study results revealed user preferences between iVLAIR and Python approaches. As machine learning algorithms are used in a variety of applications, the need to make them more intelligible is equally important as making them accessible to a large population.

Bibliography

[1] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018.

[2] Namita Agarwal and Saikat Das. Interpretable machine learning tools: A survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1528–1534. IEEE, 2020.

[3] Mohamed Alloghani, Dhiya Al-Jumeily, Ahmed J Aljaaf, Mohammed Khalaf, Jamila Mustafina, and Sin Y Tan. The application of artificial intelligence technology in healthcare: a systematic review. In *International conference on applied computing to support industry: Innovation and technology*, pages 248–261. Springer, 2019.

[4] Daniel W Apley and Jingyu Zhu. Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 82(4):1059–1086, 2020.

[5] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020.

[6] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[7] Vaishak Belle and Ioannis Papantonis. Principles and practice of explainable machine learning. *Frontiers in big Data*, page 39, 2021.

[8] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *Database Theory—ICDT'99: 7th International Conference Jerusalem, Israel, January 10–12, 1999 Proceedings 7*, pages 217–235. Springer, 1999.

[9] Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José MF Moura, and Peter Eckersley. Explainable machine learning in deployment. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 648–657, 2020.

[10] Valentina Bianchi, Marco Bassoli, Gianfranco Lombardo, Paolo Fornacciari, Monica Mordonini, and Ilaria De Munari. Iot wearable sensor and deep learning: An integrated approach for personalized human activity recognition in a smart home environment. *IEEE Internet of Things Journal*, 6(5):8553–8562, 2019.

[11] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.

[12] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[13] John Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[14] Ángel Alexander Cabrera, Marco Tulio Ribeiro, Bongshin Lee, Robert Deline, Adam Perer, and Steven M Drucker. What did my ai learn? how data scientists make sense of model behavior. *ACM Transactions on Computer-Human Interaction*, 30(1):1–27, 2023.

[15] Mackinlay Card. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999.

[16] Bahzad Charbuty and Adnan Abdulazeez. Classification based on decision tree algorithm for machine learning. *Journal of Applied Science and Technology Trends*, 2(01):20–28, 2021.

[17] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[18] Geoff Cumming. The new statistics: Why and how. *Psychological science*, 25(1):7–29, 2014.

[19] Jeffrey De Fauw, Joseph R Ledsam, Bernardino Romera-Paredes, Stanislav Nikolov, Nenad Tomasev, Sam Blackwell, Harry Askham, Xavier Glorot, Brendan O'Donoghue, Daniel Visentin, et al. Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nature medicine*, 24(9):1342–1350, 2018.

[20] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[21] Pierre Dragicevic. Fair statistical communication in hci. *Modern statistical methods for HCI*, pages 291–330, 2016.

[22] K SRIVASTAVA Durgesh and B Lekha. Data classification using support vector machine. *Journal of theoretical and applied information technology*, 12(1):1–7, 2010.

[23] Tlamelo Emmanuel, Thabiso Maupong, Dimane Mpoeleng, Thabo Semong, Banyatsang Mphago, and Oteng Tabona. A survey on missing data in machine learning. *Journal of Big Data*, 8(1):1–37, 2021.

[24] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C Prati, Bartosz Krawczyk, and Francisco Herrera. *Learning from imbalanced data sets*, volume 10. Springer, 2018.

[25] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[26] Johannes Fürnkranz. Decision tree. In *Encyclopedia of Machine Learning and Data Mining*, pages 330–335. Springer US, Boston, MA.

[27] Dave Gandy. Font awesome, the iconic font and css toolkit. *Fortawesome. github. io*, 2015.

[28] Theodoros Georgiou, Yu Liu, Wei Chen, and Michael Lew. A survey of traditional and deep learning-based feature descriptors for high dimensional data in computer vision. *International Journal of Multimedia Information Retrieval*, 9(3):135–170, 2020.

[29] Anu-Ujin Gerelt-Od, Lee Kho, Anhthy Ngo, and Andrew Yeh. Generating feature impact from individual conditional expectation plots.

[30] Prashant Gohel, Priyanka Singh, and Manoranjan Mohanty. Explainable ai: current status and future directions. *arXiv preprint arXiv:2107.07045*, 2021.

[31] Peter Harrington. *Machine learning in action*. Simon and Schuster, 2012.

[32] JB Heaton, Nicholas G Polson, and Jan Hendrik Witte. Deep learning in finance. *arXiv preprint arXiv:1602.06561*, 2016.

[33] Jeffrey Heer, Stuart K Card, and James A Landay. Prefuse: a toolkit for interactive information visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 421–430, 2005.

[34] Samuel Huron, Sheelagh Carpendale, Alice Thudt, Anthony Tang, and Michael Mauerer. Constructive visualization. In *Proceedings of the 2014 conference on Designing interactive systems*, pages 433–442, 2014.

[35] Maksims Ivanovs, Roberts Kadikis, and Kaspars Ozols. Perturbation-based methods for explaining deep neural networks: A survey. *Pattern Recognition Letters*, 150:228–234, 2021.

[36] Ai Jiang, Miguel A Nacenta, Kasim Terzic, and Juan Ye. Visualization as intermediate representations (vlair) for human activity recognition. In *Proceedings of the 14th EAI International Conference on Pervasive Computing Technologies for Healthcare*, pages 201–210, 2020.

[37] Ai Jiang, Miguel A Nacenta, and Juan Ye. Visualizations as intermediate representations (vlair): An approach for applying deep learning-based computer vision to non-image-based data. *Visual Informatics*, 6(3):35–50, 2022.

[38] Yuta Kaneko and Katsutoshi Yada. A deep learning approach for the prediction of retail store sales. In *2016 IEEE 16th International conference on data mining workshops (ICDMW)*, pages 531–537. IEEE, 2016.

[39] Hyun Kang. The prevention and handling of the missing data. *Korean journal of anesthesiology*, 64(5):402–406, 2013.

[40] Alicia Key, Bill Howe, Daniel Perry, and Cecilia Aragon. Vizdeck: self-organizing dashboards for visual analytics. In *Proceedings of the 2012 acm sigmod international conference on management of data*, pages 681–684, 2012.

[41] Oliver Kramer and Oliver Kramer. Scikit-learn. *Machine learning for evolution strategies*, pages 45–53, 2016.

[42] Max Kuhn, Kjell Johnson, et al. *Applied predictive modeling*, volume 26. Springer, 2013.

[43] Yunfei Lai. A comparison of traditional machine learning and deep learning in image recognition. *Journal of Physics: Conference Series*, 1314(1):012148, oct 2019.

[44] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[45] Andy Lee. Comparing deep neural networks and traditional vision algorithms in mobile robotics. *Swarthmore University*, 2015.

[46] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1):18, 2020.

[47] Zhicheng Liu, John Thompson, Alan Wilson, Mira Dontcheva, James Delorey, Sam Grigg, Bernard Kerr, and John Stasko. Data illustrator: Augmenting vector design tools with lazy data binding for expressive visualization authoring. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2018.

[48] Octavio Loyola-Gonzalez. Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view. *IEEE access*, 7:154096–154113, 2019.

[49] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[50] Lev Manovich. What is visualisation? *Visual Studies*, 26(1):36–49, 2011.

[51] Gonzalo Gabriel Méndez, Miguel A Nacenta, and Sebastien Vandenheste. ivolver: Interactive visual language for visualization extraction and reconstruction. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 4073–4085, 2016.

[52] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[53] Tamara Munzner. *Visualization analysis and design*. CRC press, 2014.

[54] Ismail Muraina. Ideal dataset splitting ratios in machine learning algorithms: general concerns for data scientists and data analysts. In *7th International Mardin Artuklu Scientific Research Conference*, 2022.

[55] Travis E Oliphant et al. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[56] Juri Opitz and Sebastian Burst. Macro f1 and macro f1. *arXiv preprint arXiv:1911.03347*, 2019.

[57] Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep learning vs. traditional computer vision. In *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 1 1*, pages 128–144. Springer, 2020.

[58] Charlie Parker, Sam Scott, and Alistair Geddes. Snowball sampling. *SAGE research methods foundations*, 2019.

[59] Ashish Patel and Jigarkumar Shah. Sensor-based activity recognition in the context of ambient assisted living systems: A review. *Journal of Ambient Intelligence and Smart Environments*, 11(4):301–322, 2019.

[60] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[61] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018.

[62] Jennifer Preece, Helen Sharp, and Yvonne Rogers. *Interaction design: beyond human-computer interaction*. John Wiley & Sons, 2015.

[63] Donghao Ren, Tobias Höllerer, and Xiaoru Yuan. ivisdesigner: Expressive interactive design of information visualizations. *IEEE transactions on visualization and computer graphics*, 20(12):2092–2101, 2014.

[64] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[65] Brian D Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007.

[66] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[67] Arvind Satyanarayan and Jeffrey Heer. Lyra: An interactive visualization design environment. In *Computer graphics forum*, volume 33, pages 351–360. Wiley Online Library, 2014.

[68] Arvind Satyanarayan, Ryan Russell, Jane Hoffswell, and Jeffrey Heer. Reactive vega: A streaming dataflow architecture for declarative interactive visualization. *IEEE transactions on visualization and computer graphics*, 22(1):659–668, 2015.

[69] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[70] Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? *arXiv preprint arXiv:1611.07450*, 2016.

[71] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR, 2017.

[72] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[73] Farhana Sultana, Abu Sufian, and Paramartha Dutta. Advancements in image classification using convolutional neural network. In *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pages 122–129, 2018.

[74] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022.

[75] Dhananjay Theckedath and RR Sedamkar. Detecting affect states using vgg16, resnet50 and se-resnet50 networks. *SN Computer Science*, 1:1–7, 2020.

[76] Vaibhav Tiwari, Chandrasen Pandey, Ankita Dwivedi, and Vrinda Yadav. Image classification using deep neural network. In *2020 2nd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, pages 730–733. IEEE, 2020.

[77] Andrius Vabalas, Emma Gowen, Ellen Poliakoff, and Alexander J Casson. Machine learning algorithm validation with a limited sample size. *PloS one*, 14(11):e0224365, 2019.

[78] Stef van den Elzen and Jarke J van Wijk. Small multiples, large singles: A new approach for visual data exploration. In *Computer Graphics Forum*, volume 32, pages 191–200. Wiley Online Library, 2013.

[79] Tim Van Kasteren, Athanasios Noulas, Gwenn Englebienne, and Ben Kröse. Accurate activity recognition in a home setting. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 1–9, 2008.

[80] Alfredo Vellido, José David Martín-Guerrero, and Paulo JG Lisboa. Making machine learning models interpretable. In *ESANN*, volume 12, pages 163–172. Citeseer, 2012.

[81] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, Eftychios Protopapadakis, et al. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[82] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. Deep learning for sensor-based activity recognition: A survey. *Pattern recognition letters*, 119:3–11, 2019.

[83] Adrian Weller. Challenges for transparency. 2017.

[84] Daniela Witten and Gareth James. *An introduction to statistical learning with applications in R.* springer publication, 2013.

[85] Kanit Wongsuphasawat, Zening Qu, Dominik Moritz, Riley Chang, Felix Ouk, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. Voyager 2: Augmenting visual analysis with partial view specifications. In *Proceedings of the 2017 chi conference on human factors in computing systems*, pages 2648–2659, 2017.

[86] Ray Wright. Interpreting black-box machine learning models using partial dependence and individual conditional expectation plots. *Exploring SAS® Enterprise Miner Special Collection*, 2018, 1950.

[87] Xue Ying. An overview of overfitting and its solutions. In *Journal of physics: Conference series*, volume 1168, page 022022. IOP Publishing, 2019.

[88] Juriy Zaytsev, Stefan Kienzle, and Andrea Bogazzi. Fabric. js–a powerful and simple javascript html5 canvas library, 2008.

[89] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[90] Zhi-Hua Zhou and Shaowu Liu. *Machine Learning.* Springer, Singapore, 2021.

Appendix A

Additional Information

We include two study scripts(one having iVLAIR tool for task 1 and Python code for task 2 and the other having Python code for task 1 and iVLAIR tool for task 2), codebook used for the qualitative study, the phase 2 task sheet for iVLAIR and Python approach and the post study questionnaire for iVLAIR and Python approach.

Study script: The complete script of the evaluation. Note that the script in red were action items for the experimenter.

Codebook: The codebook is broadly classified into Pros and Cons of iVLAIR and the user preferences.

Phase 2 task sheet: The tasks given to the participants during phase 2 of the tasks.

Post-study questionnaire: The SUS questionnaire filled out by the participants at the end of the study.

Study Script A
(Task1: iVLAIR; Task2: Python code)

Hi _____, thank you very much for taking the time to participate in our study. Before we start with our experiment, I need to let you know about your rights as a participant. If you feel uncomfortable you may quit at any time, and if you do so then I will delete your data right away and it will not be used. The data that we have collected will be destroyed in future and no data will be used without your explicit consent. Your participation in this experiment is confidential.

1. **Consent form**: Hope you find time to read the consent form I shared with you through email. Please feel free to ask me any questions you have with the consent form. Could you sign the consent form if everything looks good to you.

2. **Demographic form:** Here is a quick demographic survey and all questions are completely optional.

3. **Study:** Today, I will give you 2 tasks and each task will have two phases. In both the tasks, you will be asked to perform ML classification, but you will be given a different tool and a different dataset. During phase 1, I will guide you through the whole process. During the experiment, I will ask you a few questions. It's okay if you do not know the answer. Do the best according to your knowledge. During the whole experiment, I will use the think-aloud protocol to let you speak aloud what you think, and I will observe both your actions and the reasons you speak aloud. Now we are about to start the real experiment, please make yourself comfortable. Also, I would request you to put your mobile phone on silent mode so that you will not be distracted for the next 1 hour. I am now going to start the recording.

Task 1 (iVLAIR tool):

Let's start with task 1. For this task, you will be given the experimental tool and the water potability dataset. As part of our research, we designed this experimental tool to help users to perform Machine learning classification without expert knowledge or programming experience. This tool is designed using an alternative approach of using visualization for image classification. Ie, unlike the traditional approach to ML, where the raw input data is used directly to solve a classification problem, here, we first represent the input data as visualizations and then we convert the created visualizations into images. Then the generated images are used as input for the image classification algorithm to perform data classification.

The given dataset has water quality metrics for different water bodies, and you must use the experimental tool to classify if the water is safe or unsafe for drinking. In this dataset, Human consumption is the column that must be predicted. Value 1 in this column indicates that the water is safe for drinking whereas 0 indicates that the water is unsafe for drinking.

Phase 1:

Let's start with Phase 1 of this task. This tool has 4 screens. You will be asked a couple of questions at the beginning of each screen. This is screen 1.

Phase 1 Questions:

1. **Take a look at this screen and tell me from your understanding, what do you think would be the outcome of this screen execution?**

Click on this upload button and upload this input file. 'Human Consumption' is the column that must be predicted. Please select that column name under the label's dropdown. Next click on the visualize button.

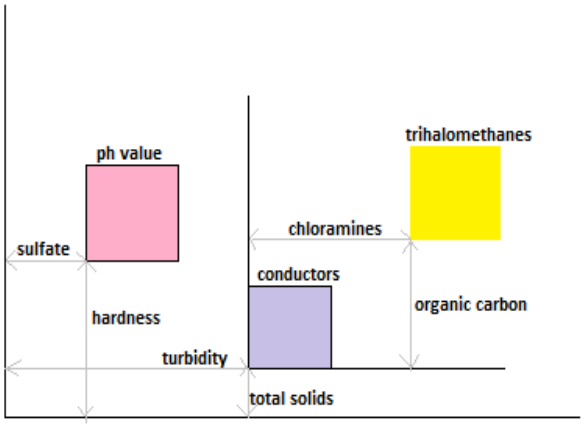
This is screen 2. I will give you 20 seconds to have a look at this screen.

Phase 1 Questions:

1. **Okay, please tell me from your understanding, what do you think would be the outcome of this screen execution?**
2. **Is there anything on this screen that you do not understand?**

This is a visualization tool. Here we are going to re-represent the input dataset into visualizations and then we will convert the created visualizations into images. Now let's create this sample visualization to complete phase 1 of this task. I will instruct you through this process. This is the top panel where we create the main visualization. This panel has a default coordinate axis and the visualizations created will be with respect to this axis.

Sample visualization created for water potability dataset:



First, let's draw a line by clicking on this line icon and dragging and dropping them into the top panel. Now a small right panel appears inside the top panel. This displays the different properties of the selected line. We are using this line to represent the good pH level of a drinking water. Good ph. level is 7. So, we shall enter 7 to the y-axis position of this selected line. We shall scale this value by multiplying 10 to make the line come to the middle. then set 0 to the x-axis position.

Now let's add another line by dragging and dropping the same icon. Now, we must draw a vertical line. so, we must click here and rotate it in anticlockwise. The maximum accepted hardness value of the drinking water is 170 ppm.

So, let's input 170 to the x-axis position of the selected line and scale it by dividing it by 5. enter 0 to the y-axis position. Now, we are done with the lines.

Next, let's create the first square by dragging and dropping the square icon. Let's assign blue color to the square by clicking on the color parameter and choose blue color from the color palette. Next, let's assign sulfate value to the size of the square. Then scale the value by dividing it by 5 to make the square look smaller. Then assign hardness to the x-axis position of the square and divide the value by 5. To do this, we must drag and drop this hardness from this middle panel to the x-axis position of the square. This middle panel will have all the column names except the human consumption which is the column that has to be predicted. The values to these columns will be based on the selection in the left panel. Now, the visualization created here is for the selected row of data on the left panel. When you select another row of data, the visualization created will be adjusted based on the selection here. Now, let's assign pH value to the y-axis position of the selected square and multiply the value by 10. Through this visualization, we can see how the value of sulfate varies for varying hardness and pH levels. Also from this visualization, we can find if the water is suitable or not suitable for drinking based on hardness values. Also, we can see if the water body falls under the good ph. level.

Now we are done with the first square. Let's create the second square and assign red color to this square. Next, assign conductors' value to the size of the square and divide the value by 10. Then assign turbidity to the x-axis position of the square and scale the value by multiplying it by 50 to move this square away from the other so that it won't be overlapping all the time. Because this may reduce the ML accuracy. Assign total solids to the y-axis position of the square. Scale this value by dividing it by 1000 as this is a big number. Through this visualization, we can see how the turbidity which is the cloudiness of the water and the total dissolved mineral in the water affects the conductivity of the water.

Finally, let's create the third square. But before that, we are going to make this third square aligned to the second square. i.e., the position of the third square not only depends on its x and y position but also based on the x and y position of the second square. To do this, we shall add another coordinate by simply clicking on the coordinate icon. Now, to this second coordinate, lets assign the turbidity and the total solids to the x and y position like the second square.

Now, let's drag and drop another square and assign a green color to this. Now, in the coordinates drop down change from default coordinates to subordinate 1 which is the new coordinates that we created. Now, assign trihalomethanes value to the side of this square and divide the value by 1. Then assign chloramines value to the x-position and multiply the value by 10. Then organic carbon to the y position and multiply the value by 5. Now this third square is associated with the second square. From this visualization we can see how the increase of organic carbon and chloramines in water have an impact on the amount of trihalomethanes in water.

Now it's the time to convert the created visualization into images so that they can be used as input to the image classification algorithm. Before we do that, we must first use the viewport button and draw a rectangle outside the created visualization. This rectangle acts as an image boundary. Next click on the apply all button. This creates the visualization inside this rectangle for all the other rows of data and then the created visualizations will be converted into images. the visualization outside this rectangle will be cropped. The generated images are displayed on the right panel.

I will give you 10 seconds to view the created visualization.

Phase 1 Questions:

3. **Okay, now tell me if the outcome of this execution matches your understanding?**

Now click on the next button. This is screen 3. I will give you 20 seconds to have a look at this screen.

Phase 1 Questions:

1. **Okay, please tell me from your understanding, what do you think would be the outcome of this execution?**
2. **Is there anything on this screen that you do not understand?**

You are right. We have integrated the VGG 16 image classification algorithm in this tool and we are going to use the created images to train and test the algorithm. Before that, first we must split the generated images for training and testing. We use the training images to train the algorithm and the testing images to test the algorithm. Values entered in the text boxes indicate the % of images that can be used for training and testing. Based on the input value, the generated images will be chosen at random. Here, the images surrounded by a solid line indicate that the images are allocated for training and the ones surrounded by dotted lines indicate that the images are allocated for testing. You can also modify the images allocated for testing into training images and vice versa by simply clicking on the images. Here, 0 and 1 is the value of column human consumption which is the column that we are going to predict. Clicking on 1 will display all the images that represent the data whose human consumption value is 1 and similarly clicking on 0 will display all the images that represent the data whose human consumption value is 0. The zoom here is to see the bird's eye view of the images.

click on the Train button. It will take a couple of minutes. Training is now completed. Now, click on the test button. I will give you 10 seconds to view the results.

Phase 1 Questions:

3. **Okay, now tell me if the outcome of this execution matches your understanding?**

Next, click on the compare results button. This is screen 4. I will give you 20 seconds to have a look at this screen.

Phase 1 Questions:

1. **Okay, please tell me from your understanding, what do you think would be the outcome of this execution?**
2. **Is there anything on this screen that you do not understand?**

This is the compare results screen. This screen has different panels. Clicking on the panel and enabling the checkbox in the confusion matrix loads the images corresponding to the selection in the selected panel. This screen will help us to view and understand the similarities and the differences in data that gave different results. Let's click panel 1 and enable this checkbox to view the images that gave the TP results. Similarly click on panel 2 and enable this checkbox to view the FP results. Here, the images on the left represent the data of the water samples that are correctly predicted to be drinkable. and the images on the right represent the data of the water samples that are incorrectly predicted to be non-drinkable but are drinkable. I will give you 20 seconds. Just see if you can find the similarities or the differences in these images that made the algorithm make a wrong prediction.

Phase 1 Questions:

3. **Okay, now tell me if the outcome of this execution matches your understanding.**

End of Phase 1 Questions:

Now, you are at the end of phase 1. Before we proceed to phase 2, you will be asked these questions.

1. **What is your interpretation of what the machine has done?**
2. **What do you think about the outcomes? Do you know if these results are good or bad?**
3. **If you had to improve the results, what would you try?**
4. **If you can only modify the dataset, such as modifying features for training and testing, what would you try?**

This is the end of phase 1.

Phase 2:

Let's start with phase 2. For this phase, I will give you 10 minutes to explore the tool and to perform data classification using this tool all by yourself. But you can ask me any questions you have. It's okay if you complete this task before the given time, we can move to the next step.

Phase 2 tasks:

Here is the task sheet. To complete this task, you must follow the given steps. Before you start, I will give you 20 seconds to investigate this task sheet and ask me any questions you have. Time up. Thank you for completing the given tasks. Before we complete this phase, I will ask a few questions.

End of Phase2 Questions:

1. **From exploring this by yourself, do you have any new understanding?**
2. **Do you understand why the results change?**
3. **Do you have new ideas of improving the results?**
4. **If you have more time, what would you want to try?**

Task 1 SUS Questionnaire:

Before we move on to the next task, you have to fill-in the questionnaire.

Congratulations, you have now completed task1.

Task 2:

Task 2 is the same as task 1, but for this task you I will give you an existing ML algorithm and a heart attack prediction dataset to classify if the patient has a possibility of heart attack or not based on different parameters. 0 indicates that the patients have less chances for heart attacks and 1 indicates that the patients have more chances for heart attack.

Phase 1:

Let's start with phase 1. This is the random forest algorithm, and the algorithm is split in several cells. Before you execute each cell, you will be first asked a couple of questions followed by instructions to execute each cell.

This is step1 of this algorithm.

Phase 1 Questions:

1. **Please have a look at this code and now tell me from your understanding, what do you think would be the outcome of this execution?**

This function is to read the input file. Please click on this run button to execute this cell.

This is step2. I will give you 20 seconds to look into this code in this cell.

Phase 1 Questions:

1. **Okay, now please tell me from your understanding, what do you think would be the outcome of this execution?**
2. **Is there anything in this part that you do not understand?**

This cell has different functions to delete or to modify the existing features/columns in the input dataset. First is the delete function to delete the existing column. These functions are used to create a new column by performing arithmetic operations on a column value with another value or by a constant. For phase 1, let's execute this function to modify the value of the column 'Oldpeak' by multiplying it by a constant 10. I will give you 10 seconds to view the result.

Phase 1 Questions:

3**. Now please tell me if the outcome of this execution matches your understanding**?

This is step3. I will give you 20 seconds to investigate this code.

Phase 1 Questions:

1. **Okay, now please tell me from your understanding, what do you think would be the outcome of this execution?**
2. **Is there anything in this part that you do not understand?**

Here, Y has values of the column - attack chances which is the column that has to be predicted and X has all the other column values. This function splits the data for training and testing which takes test data percentage as input and rest of the data will be allocated for training. Here, the test_size is 0.3. This means that 30% of data is allocated for testing and the rest of the 70% of data will be used for training. X-test and X-train have the X values that is split for test and train. Similarly, Y-test and Y-train have the Y values that is split for test and train. This function uses the X-train and Y-train data to train the model and the next function uses X-test data to obtain prediction results. Finally, this function will display the results along with the confusion matrix. Please execute cell 3. I will give you 10 seconds to view the result.

Phase 1 Questions:

3. **Now please tell me if the outcome of this execution matches your understanding?**

Next is step4 of this algorithm. This step extends from this cell to this cell. I will give you 20 seconds to look into the code.

Phase 1 Questions:

1. **Okay, now please tell me from your understanding, what do you think would be the outcome of this execution?**
2. **Is there anything in this part that you do not understand**?

These cells have different functions to compare the TP, TN, FP and FN results. For phase 1, please execute these two cells to compare the TP and FP results. Here, the first result represents the data of the patients who are correctly predicted to be healthy. and the second result represents the data of the patients who are incorrectly predicted to have heart disease but are actually healthy. I will give you 20 seconds. Just try if you can find the similarities or the differences in the data that made the algorithm to make an incorrect prediction.

Phase 1 Questions:

3. **Okay, now tell me if the outcome of this execution matches your understanding**?

Now, you are at the end of phase 1. Before we proceed to phase 2, I will ask you a few questions.

End of Phase 1 Questions:

1. **What is your interpretation of what the machine has done?**
2. **What do you think about the outcomes? Do you know if these results are good or bad?**
3. **If you had to improve the results, what would you try?**
4. **If you can only modify the dataset, such as modifying features for training and testing, what would you try?**

This is the end of phase 1.

Phase 2:

Let's start with phase 2. In this phase, you will be given 10 minutes to explore the tool and to perform data classification using this tool all by yourself. But you can ask me any questions you have. It's okay if you complete this task before the given time, we can move on to the next step. Here is the task sheet.

Phase 2 tasks:

To complete this task, you must follow the given steps. Before you start, I will give you 20 seconds to investigate this task sheet and ask me any questions you have. Time up. Thank you for completing the given tasks.

Phase 2 Questions:

Before we complete this phase, you will be asked a few questions.

1. **From exploring this by yourself, do you have any new understanding?**
2. **Do you understand why the results change?**
3. **Do you have new ideas of improving the results?**
4. **If you have more time, what would you want to try?**

We are at the end of task2.

Task 2 SUS Questionnaire:

Now, you must complete this questionnaire. Congratulations, you have now completed both the tasks.

Now, you will be asked a few post-study questions.

Post Study Questions:

1. **What do you think are the main differences between the two systems that you have seen?**
2. **What do you think are the pros and cons for each of the two tools?**
3. **Under which circumstances would you prefer to use the tool you used earlier over the tool you used now and vice versa?.**

Congratulations you have now completed the whole experiment. I am now going to stop the recording.

4. **Debriefing / Payment:**

Thank you very much for your willingness to participate in this study. Also, thank you for your efforts in completing all the given tasks and for providing your responses to all the questions asked during the study.

Your participation in this experiment helps us understand the pros and cons of our tool and the different circumstances in which you would prefer our experimental tool over the existing ML algorithms to solve data classification problems. Also, we are able to understand the usability of our tool from both the expert and the novice points of view. By participating in this study, I believe that you got an opportunity to explore an alternative approach to Machine Learning based on visualizations for image classification.

Thanks again! Do you have any questions?

Study Script B
(Task1: Python code; Task2: iVAIR)

Hi _____, thank you very much for taking the time to participate in our study. Before we start with our experiment, I need to let you know about your rights as a participant. If you feel uncomfortable you may quit at any time, and if you do so then I will delete your data right away and it will not be used. The data that we have collected will be destroyed in future and no data will be used without your explicit consent. Your participation in this experiment is confidential.

1. **Consent form**: Hope you find time to read the consent form I shared with you through email. Please feel free to ask me any questions you have with the consent form. Could you sign the consent form if everything looks good to you.

2. **Demographic form:** Here is a quick demographic survey and all questions are completely optional.

3. **Study:** Today, I will give you 2 tasks and each task will have two phases. In both the tasks, you will be asked to perform ML classification, but you will be given a different tool and a different dataset. During phase 1, I will guide you through the whole process. During the experiment, I will ask you a few questions. It's okay if you do not know the answer. Do the best according to your knowledge. During the whole experiment, I will use the think-aloud protocol to let you speak aloud what you think, and I will observe both your actions and the reasons you speak aloud. Now we are about to start the real experiment, please make yourself comfortable. Also, I would request you to put your mobile phone on silent mode so that you will not be distracted for the next 1 hour. I am now going to start the recording.

Task 1 (Python code):

Task 2 is the same as task 1, but for this task you will be given an existing ML and the water potability dataset. The given dataset has water quality metrics for different water bodies, and you must use the experimental tool to classify if the water is safe or unsafe for drinking. In this dataset, Human consumption is the column that must be predicted. Value 1 in this column indicates that the water is safe for drinking whereas 0 indicates that the water is unsafe for drinking.

Phase 1:

Let's start with phase 1. This is the random forest algorithm, and the algorithm is split in several cells. Before you execute each cell, you will be first asked a couple of questions followed by instructions to execute each cell.

This is step1 of this algorithm.

Phase 1 Questions:

1. **Please have a look at this code and now tell me from your understanding, what do you think would be the outcome of this execution?**

This function is to read the input file. Please click on this run button to execute this cell. This is step2. I will give you 20 seconds to look into this code in this cell.

Phase 1 Questions:

1. **Okay, now please tell me from your understanding, what do you think would be the outcome of this execution?**
2. **Is there anything in this part that you do not understand?**

This cell has different functions to delete or to modify the existing features/columns in the input dataset. First is the delete function to delete the existing column. These functions are used to create a new column by performing arithmetic operations on a column value with another value or by a constant. For phase 1, let's execute this function to modify the value of the column 'Oldpeak' by multiplying it by a constant 10. I will give you 10 seconds to view the result.

Phase 1 Questions:

3**. Now please tell me if the outcome of this execution matches your understanding**?

This is step3. I will give you 20 seconds to investigate this code.

Phase 1 Questions:

1. **Okay, now please tell me from your understanding, what do you think would be the outcome of this execution?**
2. **Is there anything in this part that you do not understand?**

Here, Y has values of the column - attack chances which is the column that has to be predicted and X has all the other column values. This function splits the data for training and testing which takes test data percentage as input and rest of the data will be allocated for training. Here, the test_size is 0.3. This means that 30% of data is allocated for testing and the rest of the 70% of data will be used for training. X-test and X-train have the X values that is split for test and train. Similarly, Y-test and Y-train have the Y values that is split for test and train. This function uses the X-train and Y-train data to train the model and the next function uses X-test data to obtain prediction results. Finally, this function will display the results along with the confusion matrix. Please execute cell 3. I will give you 10 seconds to view the result.

Phase 1 Questions:

3. **Now please tell me if the outcome of this execution matches your understanding?**

Next is step4 of this algorithm. This step extends from this cell to this cell. I will give you 20 seconds to look into the code.

Phase 1 Questions:

1. **Okay, now please tell me from your understanding, what do you think would be the outcome of this execution?**
2. **Is there anything in this part that you do not understand**?

These cells have different functions to compare the TP, TN, FP and FN results. For phase 1, please execute these two cells to compare the TP and FP results. Here, the first result represents the data of the patients who are correctly predicted to be healthy. and the second result represents the data of the patients who are incorrectly predicted to have heart disease but are actually healthy. I will give you 20 seconds. Just try if you can find the similarities or the differences in the data that made the algorithm to make an incorrect prediction.

Phase 1 Questions:

3. **Okay, now tell me if the outcome of this execution matches your understanding**?

Now, you are at the end of phase 1. Before we proceed to phase 2, I will ask you a few questions.

End of Phase 1 Questions:

1. **What is your interpretation of what the machine has done?**
2. **What do you think about the outcomes? Do you know if these results are good or bad?**
3. **If you had to improve the results, what would you try?**
4. **If you can only modify the dataset, such as modifying features for training and testing, what would you try?**

This is the end of phase 1.

Phase 2:

Let's start with phase 2. In this phase, you will be given 10 minutes to explore the tool and to perform data classification using this tool all by yourself. But you can ask me any questions you have. It's okay if you complete this task before the given time, we can move on to the next step. Here is the task sheet.

Phase 2 tasks:

To complete this task, you must follow the given steps. Before you start, I will give you 20 seconds to investigate this task sheet and ask me any questions you have. Time up. Thank you for completing the given tasks.

Phase 2 Questions:

Before we complete this phase, you will be asked a few questions.

1. **From exploring this by yourself, do you have any new understanding?**
2. **Do you understand why the results change?**
3. **Do you have new ideas of improving the results?**
4. **If you have more time, what would you want to try?**

We are at the end of task1.

Task 1 SUS Questionnaire:

Now, you must complete this questionnaire. Congratulations, you have now completed both the tasks.

Task 2 (iVLAIR tool):

Let's start with task 2. For this task, I will give you the experimental tool and a heart attack prediction dataset. As part of our research, we designed this experimental tool to help users to perform Machine learning classification without expert knowledge or programming experience. This tool is designed using an alternative approach of using visualization for image classification. Ie, unlike the traditional approach to ML, where the raw input data is used directly to solve a classification problem, here, we first represent the input data as visualizations and then we convert the created visualizations into images. Then the generated images are used as input for the image classification algorithm to perform data classification.

The given dataset to classify if the patient has a possibility of heart attack or not based on different parameters. 0 indicates that the patients have less chances for heart attacks and 1 indicates that the patients have more chances for heart attack.

Phase 1:

Let's start with Phase 1 of this task. This tool has 4 screens. You will be asked a couple of questions at the beginning of each screen. This is screen 1.

Phase 1 Questions:

1. **Take a look at this screen and tell me from your understanding, what do you think would be the outcome of this screen execution?**

Click on this upload button and upload this input file. 'Human Consumption' is the column that must be predicted. Please select that column name under the label's dropdown. Next click on the visualize button.

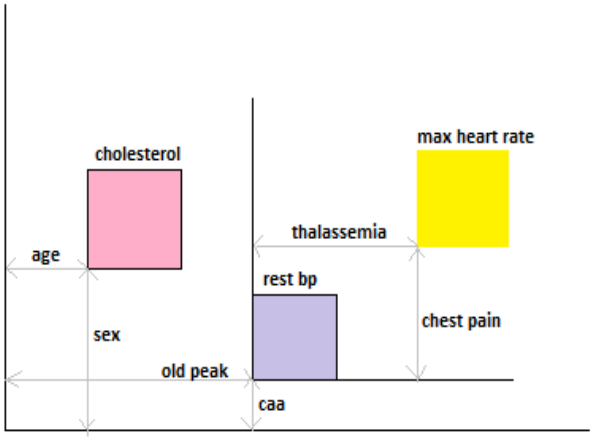
This is screen 2. I will give you 20 seconds to have a look at this screen.

Phase 1 Questions:

1. **Okay, please tell me from your understanding, what do you think would be the outcome of this screen execution?**
2. **Is there anything on this screen that you do not understand?**

This is a visualization tool. Here we are going to re-represent the input dataset into visualizations and then we will convert the created visualizations into images. Now let's create this sample visualization to complete phase 1 of this task. I will instruct you through this process. This is the top panel where we create the main visualization. This panel has a default coordinate axis and the visualizations created will be with respect to this axis.

Sample visualization created for heart attack dataset:



First, let's draw a line by clicking on this line icon and dragging and dropping them into the top panel. Now a small right panel appears inside the top panel. This displays the different properties of the selected line. We are using this line to represent the good pH level of a drinking water. Good ph. level is 7. So, we shall enter 7 to the y-axis position of this selected line. We shall scale this value by multiplying 10 to make the line come to the middle. then set 0 to the x-axis position.

Now let's add another line by dragging and dropping the same icon. Now, we must draw a vertical line. so, we must click here and rotate it in anticlockwise. The maximum accepted hardness value of the drinking water is 170 ppm. So, let's input 170 to the x-axis position of the selected line and scale it by dividing it by 5. enter 0 to the y-axis position. Now, we are done with the lines.

Next, let's create the first square by dragging and dropping the square icon. Let's assign blue color to the square by clicking on the color parameter and choose blue color from the color palette. Next, let's assign sulfate value to the size of the square. Then scale the value by dividing it by 5 to make the square look smaller. Then assign hardness to the x-axis position of the square and divide the value by 5. To do this, we must drag and drop this hardness from this middle panel to the x-axis position of the square. This middle panel will have all the column names except the human consumption which is the column that has to be predicted. The values to these columns will be based on the selection in the left panel. Now, the visualization created here is for the selected row of data on the left panel. When you select another row of data, the visualization created will be adjusted based on the selection here. Now, let's assign pH value to the y-axis position of the selected square and multiply the value by 10. Through this visualization, we can see how the value of sulfate varies for varying hardness and pH levels. Also from this visualization, we can find if the water is suitable or not suitable for drinking based on hardness values. Also, we can see if the water body falls under the good ph. level.

Now we are done with the first square. Let's create the second square and assign red color to this square. Next, assign conductors' value to the size of the square and divide the value by 10. Then assign turbidity to the x-axis

position of the square and scale the value by multiplying it by 50 to move this square away from the other so that it won't be overlapping all the time. Because this may reduce the ML accuracy. Assign total solids to the y-axis position of the square. Scale this value by dividing it by 1000 as this is a big number. Through this visualization, we can see how the turbidity which is the cloudiness of the water and the total dissolved mineral in the water affects the conductivity of the water.

Finally, let's create the third square. But before that, we are going to make this third square aligned to the second square. i.e., the position of the third square not only depends on its x and y position but also based on the x and y position of the second square. To do this, we shall add another coordinate by simply clicking on the coordinate icon. Now, to this second coordinate, lets assign the turbidity and the total solids to the x and y position like the second square.

Now, let's drag and drop another square and assign a green color to this. Now, in the coordinates drop down change from default coordinates to subordinate 1 which is the new coordinates that we created. Now, assign trihalomethanes value to the side of this square and divide the value by 1. Then assign chloramines value to the x-position and multiply the value by 10. Then organic carbon to the y position and multiply the value by 5. Now this third square is associated with the second square. From this visualization we can see how the increase of organic carbon and chloramines in water have an impact on the amount of trihalomethanes in water.

Now it's the time to convert the created visualization into images so that they can be used as input to the image classification algorithm. Before we do that, we must first use the viewport button and draw a rectangle outside the created visualization. This rectangle acts as an image boundary. Next click on the apply all button. This creates the visualization inside this rectangle for all the other rows of data and then the created visualizations will be converted into images. the visualization outside this rectangle will be cropped. The generated images are displayed on the right panel.

I will give you 10 seconds to view the created visualization.

Phase 1 Questions:

3. **Okay, now tell me if the outcome of this execution matches your understanding?**

Now click on the next button. This is screen 3. I will give you 20 seconds to have a look at this screen.

Phase 1 Questions:

1. **Okay, please tell me from your understanding, what do you think would be the outcome of this execution?**
2. **Is there anything on this screen that you do not understand?**

You are right. We have integrated the VGG 16 image classification algorithm in this tool and we are going to use the created images to train and test the algorithm. Before that, first we must split the generated images for training and testing. We use the training images to train the algorithm and the testing images to test the algorithm. Values entered in the text boxes indicate the % of images that can be used for training and testing. Based on the input value, the generated images will be chosen at random. Here, the images surrounded by a solid line indicate that the images are allocated for training and the ones surrounded by dotted lines indicate that the images are allocated for testing. You can also modify the images allocated for testing into training images and vice versa by simply clicking on the images. Here, 0 and 1 is the value of column human consumption which is the column that we are going to predict. Clicking on 1 will display all the images that represent the data whose human consumption value is 1 and

similarly clicking on 0 will display all the images that represent the data whose human consumption value is 0. The zoom here is to see the bird's eye view of the images.

click on the Train button. It will take a couple of minutes. Training is now completed. Now, click on the test button. I will give you 10 seconds to view the results.

Phase 1 Questions:

3. **Okay, now tell me if the outcome of this execution matches your understanding?**

Next, click on the compare results button. This is screen 4. I will give you 20 seconds to have a look at this screen.

Phase 1 Questions:

1. **Okay, please tell me from your understanding, what do you think would be the outcome of this execution?**
2. **Is there anything on this screen that you do not understand?**

This is the compare results screen. This screen has different panels. Clicking on the panel and enabling the checkbox in the confusion matrix loads the images corresponding to the selection in the selected panel. This screen will help us to view and understand the similarities and the differences in data that gave different results. Let's click panel 1 and enable this checkbox to view the images that gave the TP results. Similarly click on panel 2 and enable this checkbox to view the FP results. Here, the images on the left represent the data of the water samples that are correctly predicted to be drinkable. and the images on the right represent the data of the water samples that are incorrectly predicted to be non-drinkable but are drinkable. I will give you 20 seconds. Just see if you can find the similarities or the differences in these images that made the algorithm make a wrong prediction.

Phase 1 Questions:

3. **Okay, now tell me if the outcome of this execution matches your understanding.**

End of Phase 1 Questions:

Now, you are at the end of phase 1. Before we proceed to phase 2, you will be asked these questions.

1. **What is your interpretation of what the machine has done?**
2. **What do you think about the outcomes? Do you know if these results are good or bad?**
3. **If you had to improve the results, what would you try?**
4. **If you can only modify the dataset, such as modifying features for training and testing, what would you try?**

This is the end of phase 1.

Phase 2:

Let's start with phase 2. For this phase, I will give you 10 minutes to explore the tool and to perform data classification using this tool all by yourself. But you can ask me any questions you have. It's okay if you complete this task before the given time, we can move to the next step.

Phase 2 tasks:

Here is the task sheet. To complete this task, you must follow the given steps. Before you start, I will give you 20 seconds to investigate this task sheet and ask me any questions you have. Time up. Thank you for completing the given tasks. Before we complete this phase, I will ask a few questions.

End of Phase2 Questions:

1. **From exploring this by yourself, do you have any new understanding?**
2. **Do you understand why the results change?**
3. **Do you have new ideas of improving the results?**
4. **If you have more time, what would you want to try?**

Task 2 SUS Questionnaire:

Before we move on to the next task, you have to fill-in the questionnaire. Congratulations, you have now completed both the tasks.

Now, you will be asked a few post-study questions.

Post Study Questions:

1. **What do you think are the main differences between the two systems that you have seen?**
2. **What do you think are the pros and cons for each of the two tools?**
3. **Under which circumstances would you prefer to use the tool you used earlier over the tool you used now and vice versa?.**

Congratulations you have now completed the whole experiment. I am now going to stop the recording.

4. **Debriefing / Payment:**

Thank you very much for your willingness to participate in this study. Also, thank you for your efforts in completing all the given tasks and for providing your responses to all the questions asked during the study.

Your participation in this experiment helps us understand the pros and cons of our tool and the different circumstances in which you would prefer our experimental tool over the existing ML algorithms to solve data classification problems. Also, we are able to understand the usability of our tool from both the expert and the novice points of view. By participating in this study, I believe that you got an opportunity to explore an alternative approach to Machine Learning based on visualizations for image classification.

Thanks again! Do you have any questions?

iVLAIR Code Book

Name	Description	Files	References
Cons		0	0
iVlair		0	0
basic ML knowledge required		2	2
difficulty finding the data pattern		2	2
doesn't work for large data		4	6
less familiar		2	2
prior knowledge required		0	0
knowledge on data required		4	5
knowledge on tool required		3	3
people with visualization knowledge		2	3
restricted control		3	3
time consuming		1	1
trial and error		1	1
Python		0	0
difficult to transform data		1	1
difficult to understand data		0	0
difficult understanding data		2	2
hard to identify data patterns		10	10
prior or technical knowledge required		10	13
visualization is trickier		1	1
Pros		0	0
iVLAIR		0	0
different or interesting approach		3	3
easy to transform data		1	2
easy to use		5	7
no prior knowledge required		0	0
novice or first time ML users		7	10
To understand data		0	0
detect outliers		1	1
easy to understand data		7	10
Identify data patterns		12	18
visually see the changes you make		1	1
Python		0	0
can re-use the code		2	2
easier to use		2	2
easy to see the flow		3	3
easy to understand data		1	1
familiarity		4	4
faster		1	1
flexibility		6	6
understand data patterns		4	4
works for large data		2	2
Faster		1	1
large dataset		1	1
User preferences		0	0

iVLAIR		0	0
data transformation		2	2
debug		4	4
prior knowledge on data		1	1
prior visualization knowledge		2	2
understand ML		1	1
understand data		7	7
Python		0	0
familiarity		4	4
flexibility		2	2
hard problems		1	1
large datasets		2	2
other operations		1	1
reusability		1	1

Phase 2 – iVLAIR Tasks

To complete this phase, you will be only given 10 minutes:

1. You must use the same dataset you used for phase 1.
2. Create new visualization or modify the existing visualization created during phase1.
3. Next, perform ML classification with the new visualization. In this step, you are free to modify the images allocated for training and testing either by modifying the values entered in Train % and Test % text boxes or by clicking on the images.
4. Finally, view and compare the results.

Phase 2 – Traditional ML Tasks

To complete this phase, you will be only given 10 minutes.

1. You have to use the same dataset you used for phase 1.
2. You can modify the given dataset by executing any or all of the given functions. You can even modify the given functions or create your own functions.
3. Then, you can not modify the given algorithm. But in this step, you are free to modify the percentage of data allocated for training and testing.
4. Finally, view and compare the obtained results.

System Usability Scale - iVLAIR

		Strongly disagree				Strongly agree
1.	I think that I would like to use this system frequently.	☐	☐	☐	☐	☐
2.	I found the system unnecessarily complex	☐	☐	☐	☐	☐
3.	I thought the system was easy to use.	☐	☐	☐	☐	☐
4.	I think that I would need the support of a technical person to be able to use this system.	☐	☐	☐	☐	☐
5.	I found the various functions in this system were well integrated	☐	☐	☐	☐	☐
6.	I thought there was too many inconsistency in this system	☐	☐	☐	☐	☐
7.	I would imagine that most people would learn to use this system very quickly	☐	☐	☐	☐	☐
8.	I found the system very awkward to use	☐	☐	☐	☐	☐
9.	I felt very confident using the system	☐	☐	☐	☐	☐
10.	I needed to learn a lot of things before I could get going with this system	☐	☐	☐	☐	☐

System Usability Scale - Python Code

		Strongly disagree				Strongly agree
1.	I think that I would like to use this system frequently.	☐	☐	☐	☐	☐
2.	I found the system unnecessarily complex	☐	☐	☐	☐	☐
3.	I thought the system was easy to use.	☐	☐	☐	☐	☐
4.	I think that I would need the support of a technical person to be able to use this system.	☐	☐	☐	☐	☐
5.	I found the various functions in this system were well integrated	☐	☐	☐	☐	☐
6.	I thought there was too many inconsistency in this system	☐	☐	☐	☐	☐
7.	I would imagine that most people would learn to use this system very quickly	☐	☐	☐	☐	☐
8.	I found the system very awkward to use	☐	☐	☐	☐	☐
9.	I felt very confident using the system	☐	☐	☐	☐	☐
10.	I needed to learn a lot of things before I could get going with this system	☐	☐	☐	☐	☐