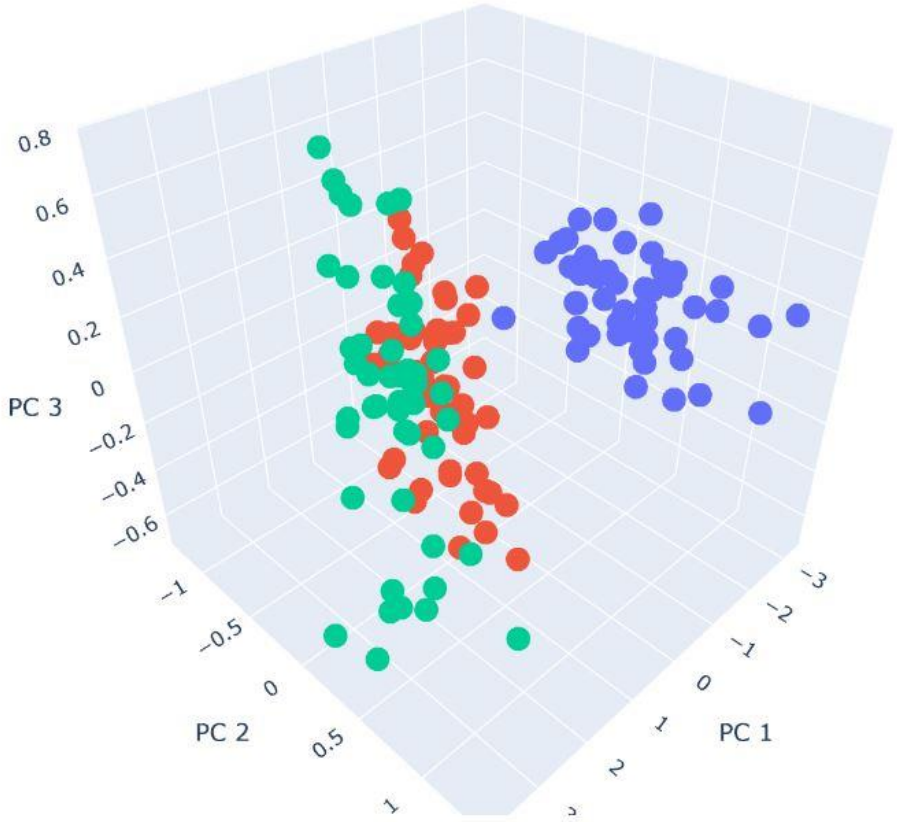
Machine Learning Algorithms Simulator

Kareem Jabareen – 211406343



University of Haifa

2021/2022

Table of interest

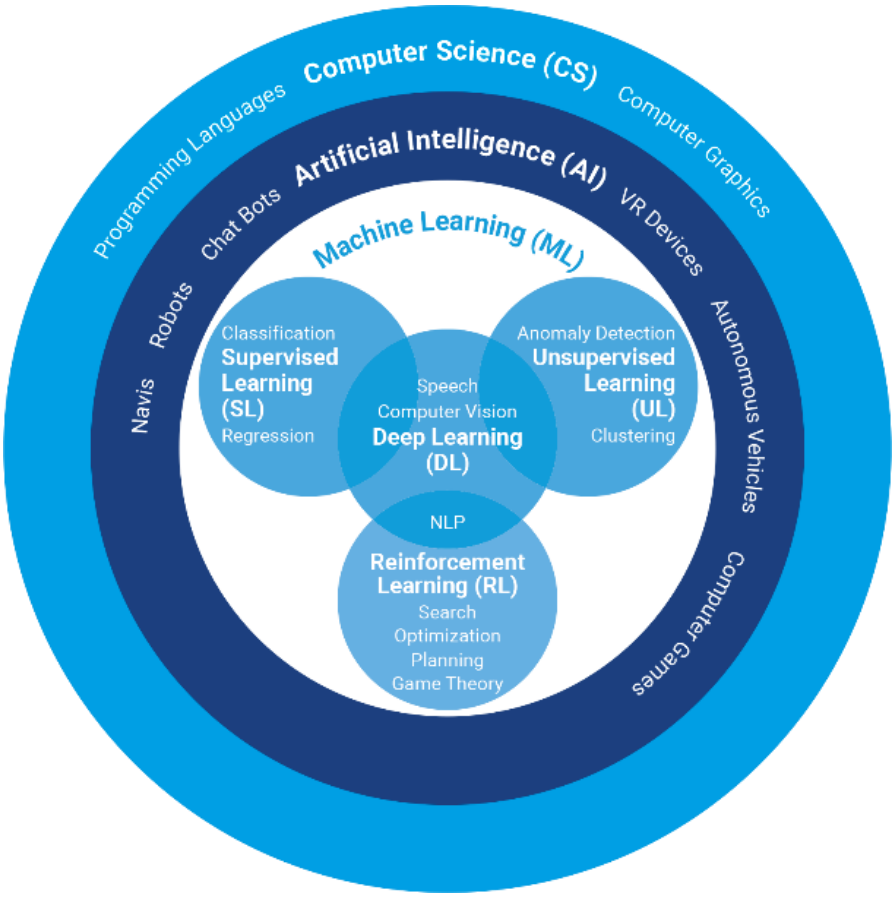
Problem Stating

Machine Learning is relatively a new field and a tough one at the same time. Many students find it challenging to learn ML algorithms, that is due to several factors like:

1. Each one of the ML algorithms has his own components that you need to learn before you can apply them.
2. Most of these algorithms require good knowledge of many aspects of mathematics especially Algebra.

Therefore, students start quitting ML courses which might slow down the ML research. When talking about ML research, it is important to mention that ML affects most of the modern applications we used almost daily. In addition, machine learning has a great influence on many other fields related to Computer Science like Computer Vision, Natural Language Processing and Computer Graphics to name a few.

Therefore, it is important to encourage others to study ML, so they develop solutions for different aspects in life.



Project Definition

To tackle the issue I presented, I have developed a website which simulates 8 different algorithms, 7 of the algorithms are ML algorithms, while the last one is DFS.

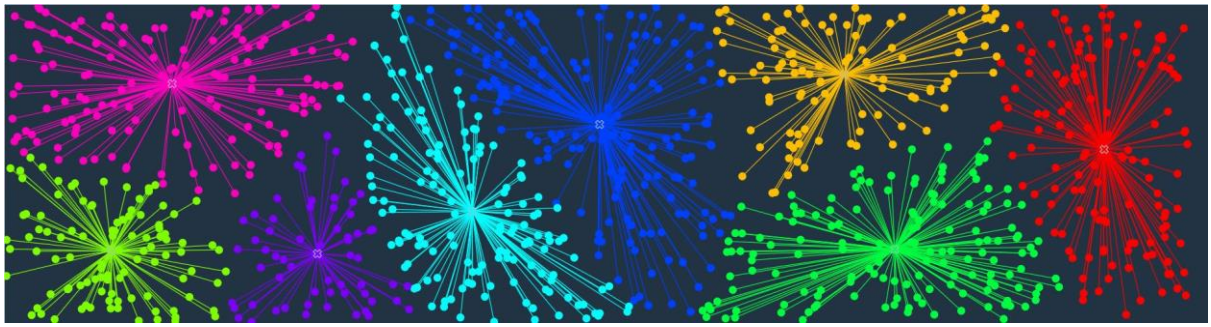
When opening the homepage of the website, I show the user 8 cards, one for each algorithm. When clicking a card, a new page will be open and the user will be able to enter the algorithm output and to set the algorithm parameters, and he will see how the algorithm works and what its results are.

For my work I have coded all the algorithms from scratch to improve that learning ML algorithms is possible and can be fun, here are the algorithms the website has:

1. DFS
2. K-Means Clustering
3. K-Nearest Neighbors (KNN)
4. Linear Discriminant Analysis (LDA)
5. Maximum Likelihood Estimation (MLE)
6. Naïve Bayes Classifier
7. Principal Component Analysis (PCA)
8. Support-Vector Machine (SVM)

In addition, I have developed two different model types: Saved dataset model and Custom dataset model. These models are available for K-Means, KNN, LDA, MLE, Naïve Bayes Classifier and PCA since they seemed to be unclear when running them on a single dataset.

Example of K-Means running on the default dataset with 1000 input points and 8 clusters:



In addition to the simulation, each page has a brief explanation of the algorithm, what it is used for and a helpful link for more information.

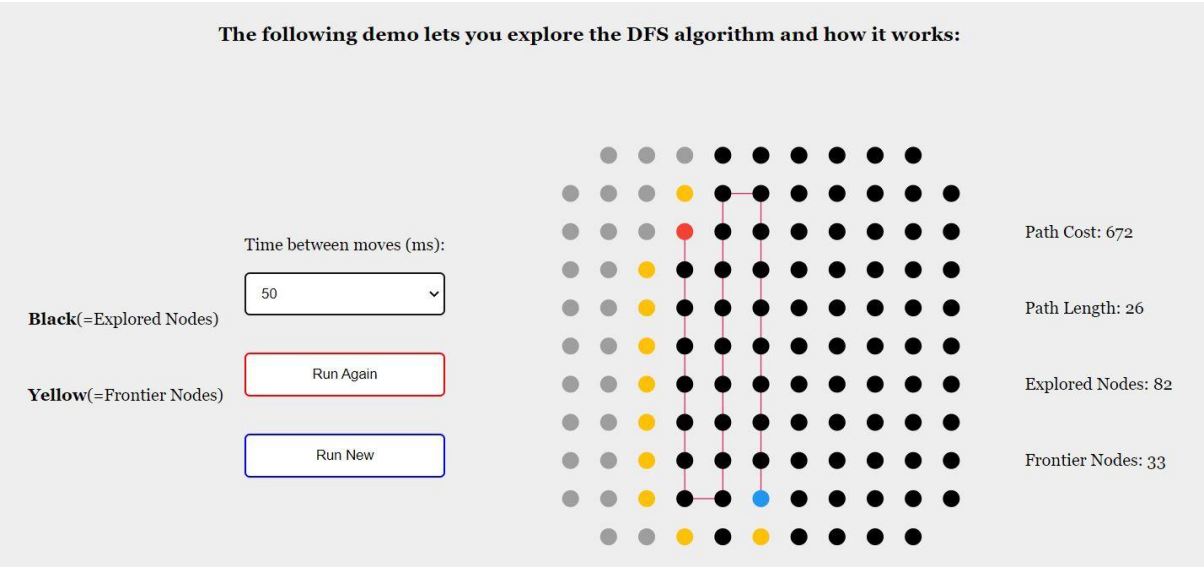
Implementation

As I have mentioned earlier, I coded all the algorithms from scratch. For coding the algorithms, I have used Python programming language mainly for its easy and fast indexing. The Python part was done in Django framework for developing web apps. However, I needed to build the web templates using HTML, CSS and used JavaScript for implementing the dynamic views on the website.

I used a Django template for starting the work, implemented the 8 algorithms, then started building the templates using HTML. After that, I created the stylesheets for the web pages using CSS, and at the end I created the JavaScript files which took the biggest part of the project. The JS code enabled the interactivity of the web app, created the connection between the Backend and the Frontend using POST and GET requests. On the other hand, the Python scripts received the JS requests, processed it using the algorithms I had coded myself, and sent the processing results back to the Frontend as JSON objects.

Results:

DFS:



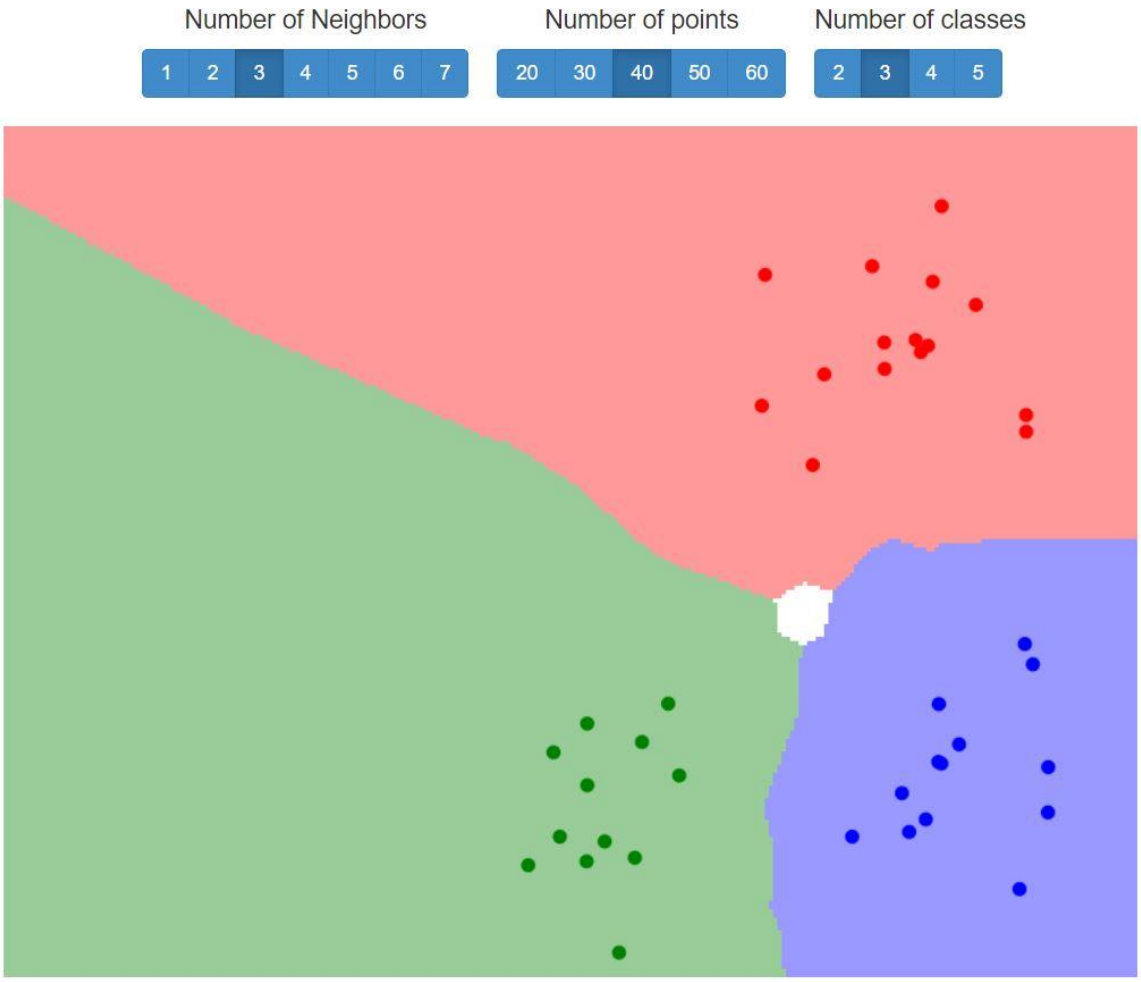
K-Means Clustering:

Step 1:

Step 2:

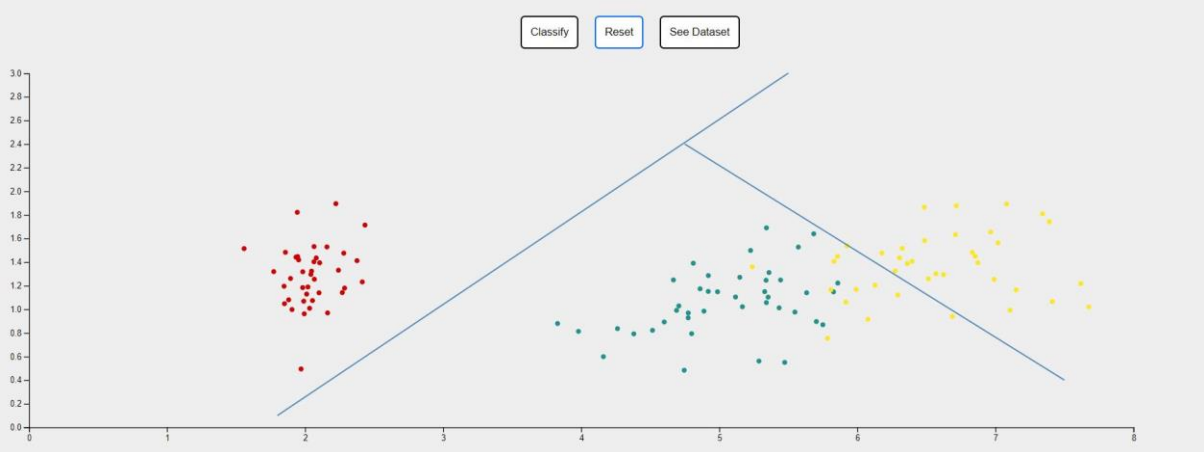
We can see in the last screenshot how the 2 furthest points from the red cluster moved to the other class which made the cyan cluster move a bit towards the red points.

K-Nearest Neighbors (KNN):

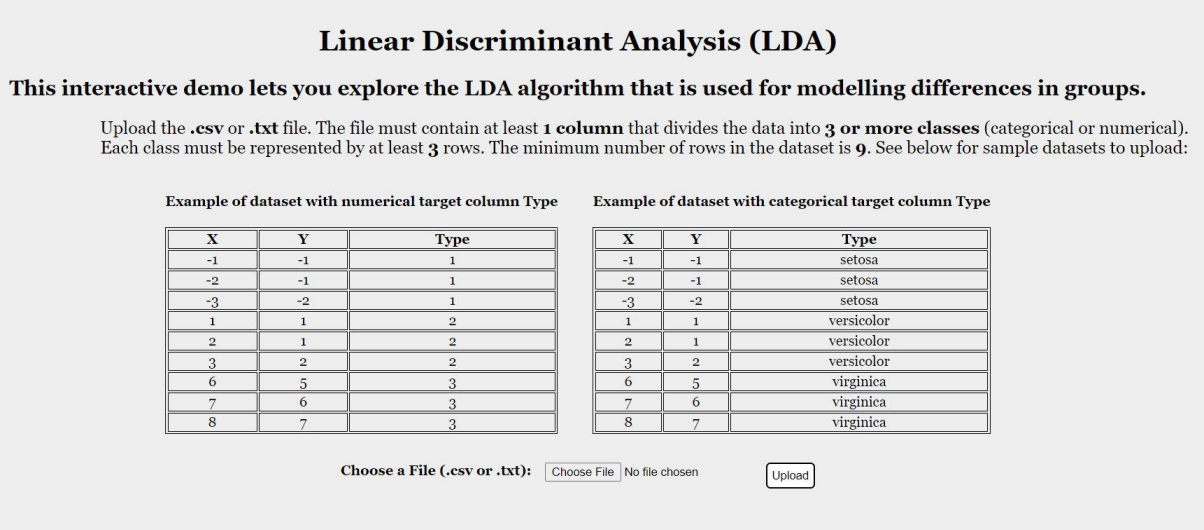


The white area are a “draw situation” in which the algorithm cannot decide to which class he should assign the points. That happens when there are at least two classes with the same number of votes for a certain point and no other class has more votes than them.

Linear Discriminant Analysis (LDA):



The above results are for the default dataset, however; when choosing to upload a dataset from the local machine, the user gets the following interface to clarify how the file should look with examples:



Linear Discriminant Analysis (LDA)

This interactive demo lets you explore the LDA algorithm that is used for modelling differences in groups.

Upload the **.csv** or **.txt** file. The file must contain at least **1 column** that divides the data into **3 or more classes** (categorical or numerical). Each class must be represented by at least **3** rows. The minimum number of rows in the dataset is **9**. See below for sample datasets to upload:

Example of dataset with numerical target column Type

X	Y	Type
-1	-1	1
-2	-1	1
-3	-2	1
1	1	2
2	1	2
3	2	2
6	5	3
7	6	3
8	7	3

Example of dataset with categorical target column Type

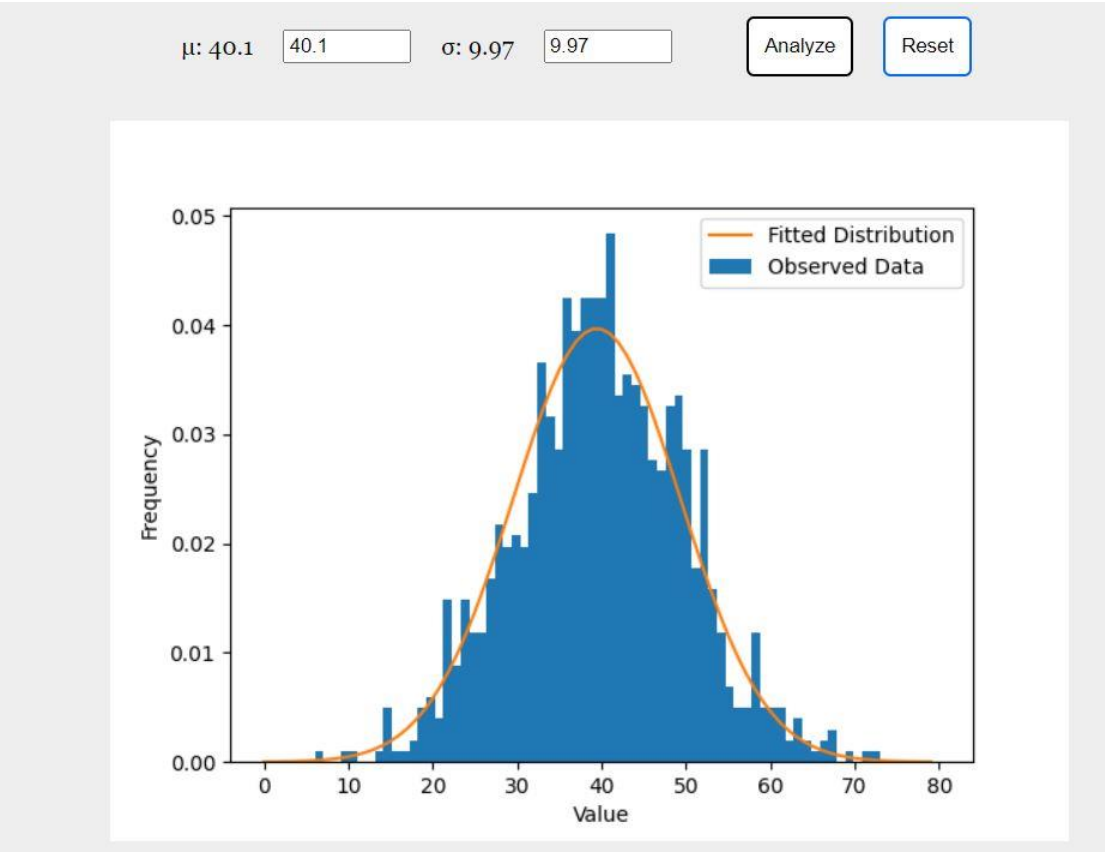
X	Y	Type
-1	-1	setosa
-2	-1	setosa
-3	-2	setosa
1	1	versicolor
2	1	versicolor
3	2	versicolor
6	5	virginica
7	6	virginica
8	7	virginica

Choose a File (.csv or .txt): Choose File No file chosen Upload

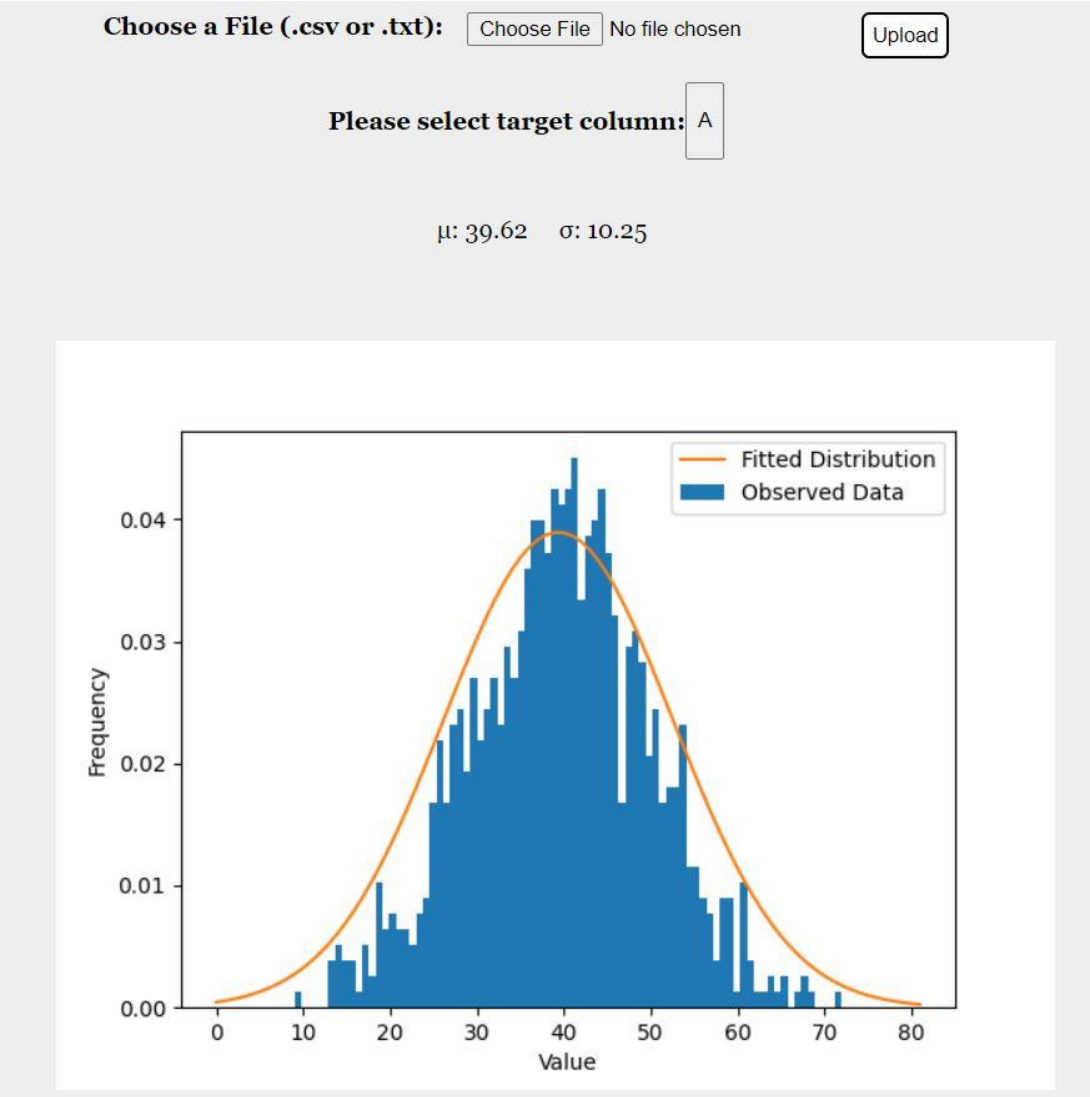
The user gets similar interfaces for all the algorithms with custom dataset option which helps the user with uploading his own data and running the chosen algorithm on it.

Maximum Likelihood Estimation (MLE):

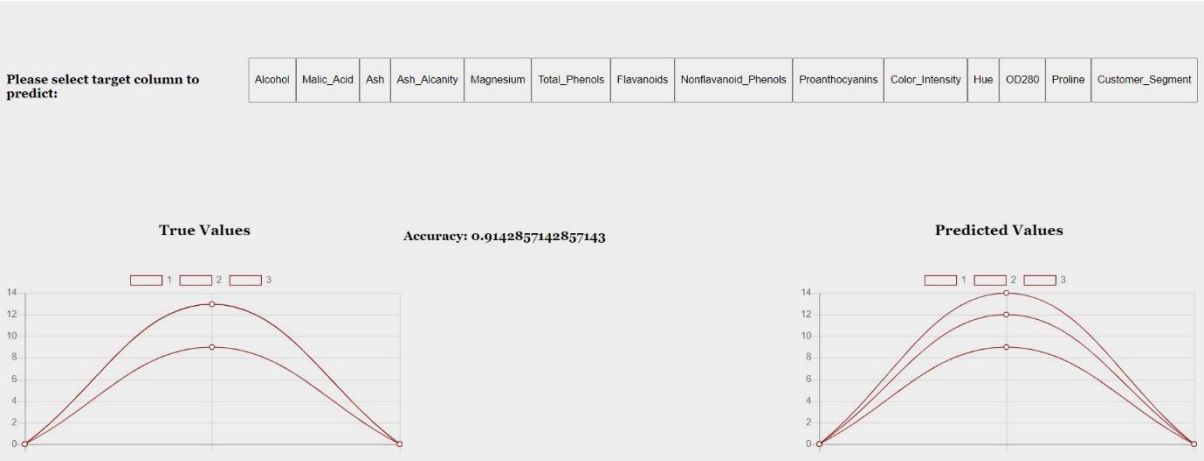
Default dataset option:



Custom dataset option:

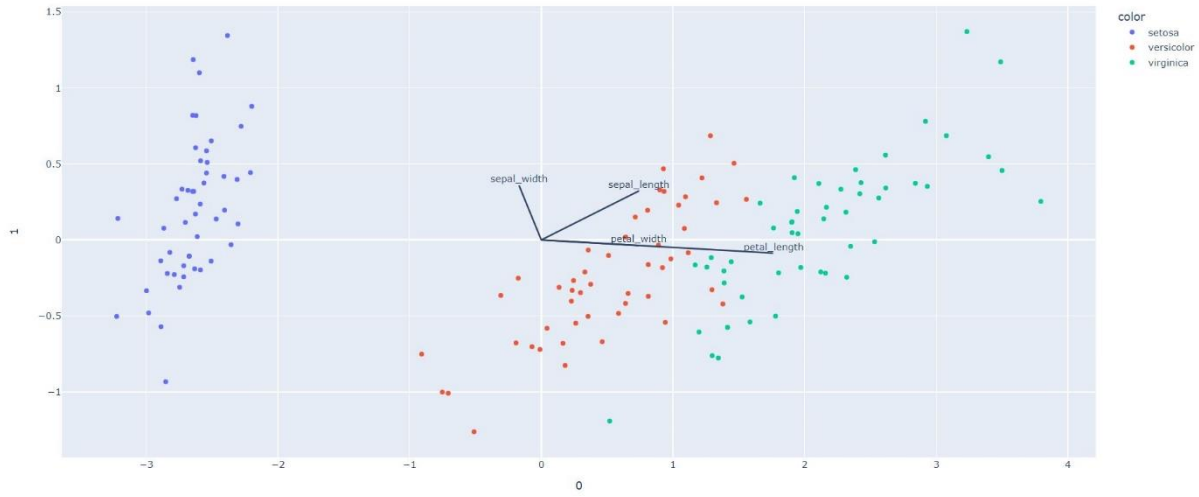


Naïve Bayes Classifier:

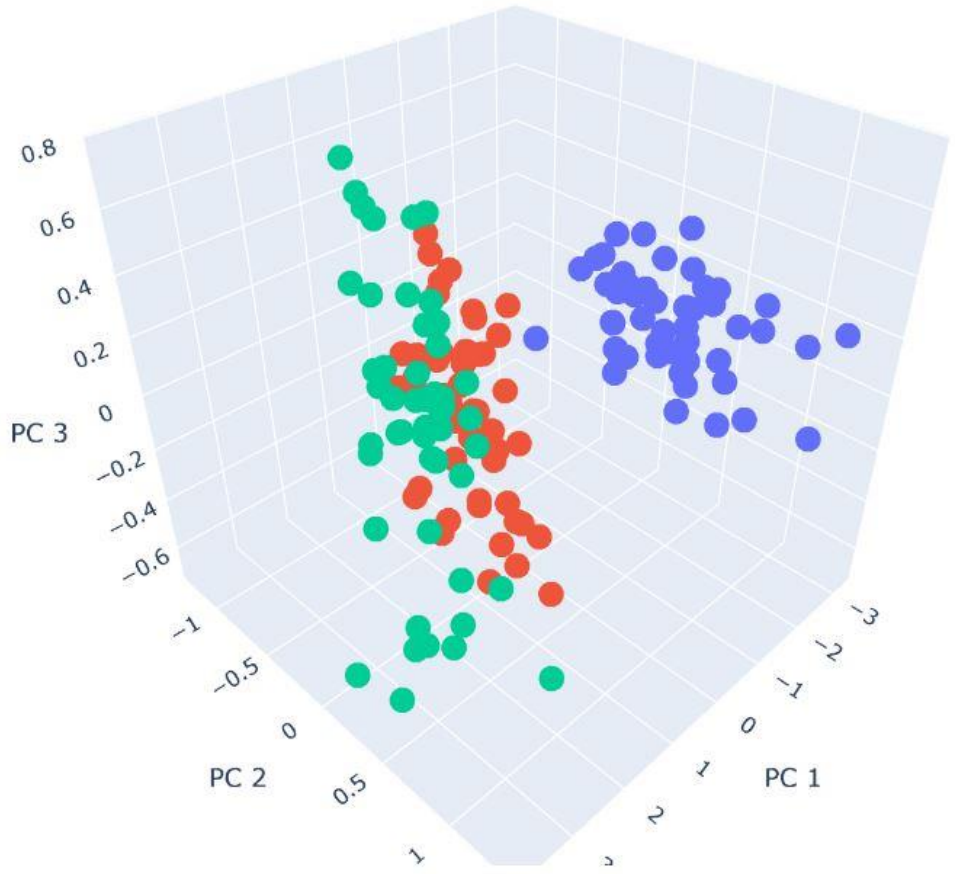


Principal Component Analysis (PCA):

2-Component solution:



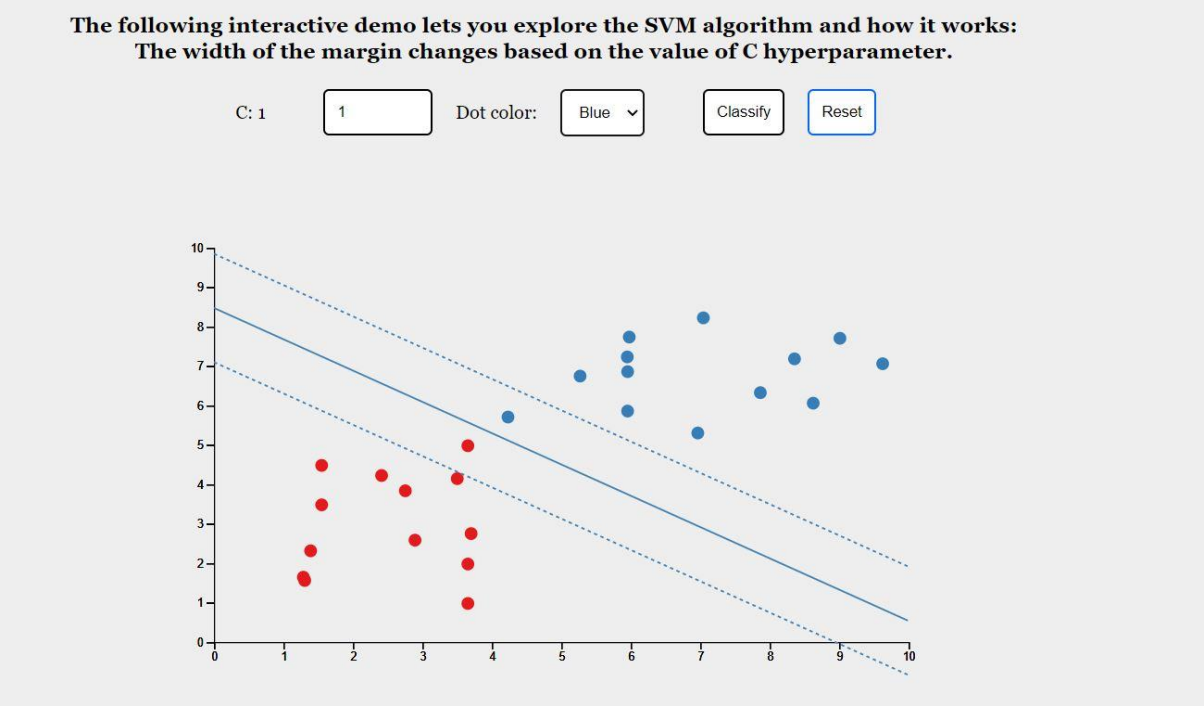
3-Component solution for the same dataset:



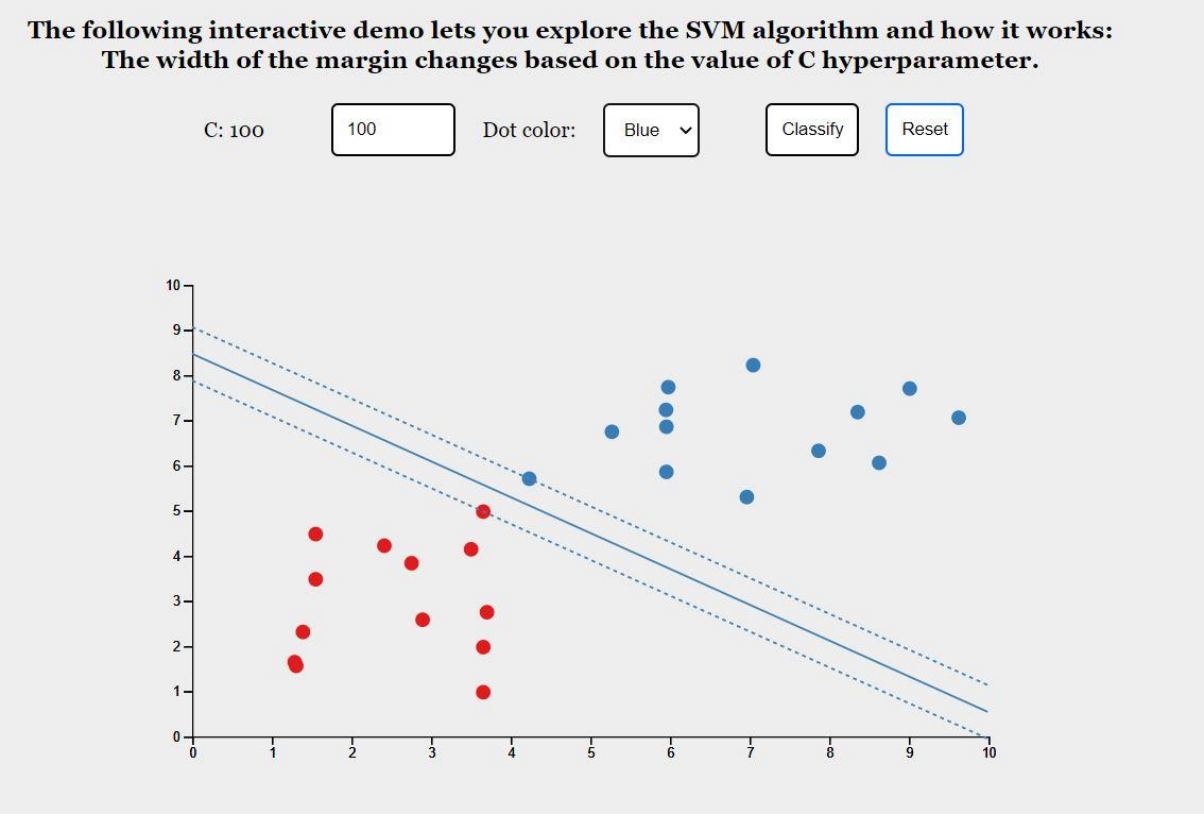
Support-Vector Machine (SVM):

Using the hyperparameter in the simulation, we can simulate both the hard-margin SVM and the soft-margin SVM as shown in the following screenshots:

Soft-Margin SVM:



Hard-Margin SVM:



My Learning

For the theoretical part, I had to go over the algorithms themselves and what they are used for. In addition, I went through new terms to me like Unsupervised learning and Supervised learning.

Furthermore, I had to learn many things related to the implementation itself to get the project done:

1. I learned JavaScript from zero which I used in making my website interactive and visually good looking.

2. I went over HTML and learned CSS for building and styling websites.

3. I learned to code in Django framework for developing web apps and implementing POST and GET requests which had me learning how HTTP protocol works.

4. This work introduced me to several useful Python packages like: Matplotlib, Pandas, Plotly to name a few.

5. Connecting between Backend and Fronted and how to handle exceptions and HTTP bad requests which introduced me to the Debug mode in developing web apps.

6. Developing the option for uploading files to the website and processing their content.

7. Reading the input files and making sure they are relevant.

Main Difficulty

The main difficulty of the project was to pass input data to the Python Backend code, so I process it and send it back to the Fronted (JS code).

I had to choose between implementing the algorithms in Python or in JavaScript. Coding them in JS would remove the need for Django framework in my work and all the work done transferring JSON objects between Python and JS. On the other hand, using Python for the algorithms' implementation is much easier due to the great functionality of Python indexing and ease of use.

I chose to work with Python which made the implementation of the algorithms easier both for developing and debugging. In addition, it made the code more readable. However, that led to my main difficulty during my work on this project, how to transfer data between JS and Python?

For that, I had to search and learn on the Internet until I found a solution. I implemented the GET and POST requests from the REST API, and with the help of the built-in Python package called 'json', I could eventually send and receive data as JSON objects.

How to Improve

There are many ideas for improving the project:

1. Add more algorithms (might need to add a search tool in the homepage).

2. Support downloading the algorithms results as images or csv files.

3. Developing a mobile app version.

4. Developing Sign-in and Sign-up to the website and developing a section for reports and questions.

5. Developing a quick demo when clicking an algorithm for the first time so the user will not have difficulties running the simulation.

6. Add a button in each algorithm page which lets the user download the current dataset.

7. Add daily quick multiple-choice questions to let the user test his knowledge of ML algorithms.

8. Create a ML terminology page and a link to it at the top of the homepage which might help the users understand the explanation of the algorithms I wrote.