AMMI Bootcamp In-class Project

1. **Introduction**:

Building intelligent computers or machines was a long standing problem for decades/centuries in the scientific community. This has evolved into different stages, from writing complex rule-based programs to more intelligent machine learning systems. Initially, traditional programming was mostly about building these rule-based systems where humans give a set of instructions to a computer which then gives an output (see Fig 1).

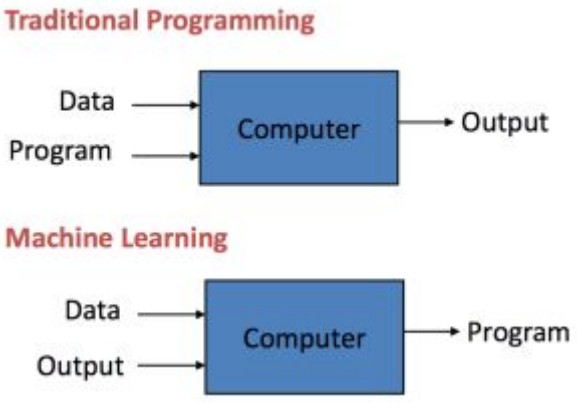


Fig 1: Overview of conventional computer programming and machine learning.

The goal of this project is to build a simple rule-based program. Throughout the project, you will have the opportunity to familiarize yourself with how to use classes, functions, and various data structures e.g. dictionaries, lists, and few others.

2. **Problem Statement**:

Your task is to develop a simple intuitive rule-based classifier. You are given a dataset consisting of attributes with values describing the characteristics of each sample. The data has two classes (in our case, 1 and 2), and the classifier has to predict whether a sample belongs to the class 1 or 2. Once you've built your classifier, you should then evaluate it on a separate unseen dataset.

3. **Dataset description**:

You have two separate comma-separated values (csv) files. One of these files contains the train data and the other contains test data. In simple terms, the train data is the data we will use to build our classifier and the test data is what we will use to evaluate our classifier. Both files has 13 attributes/columns as shown below:

Column names: `['ID', 'f1', 'f2', 'f3', 'f4', 'f5', 'f6', 'f7', 'f8', 'f9', 'f10', 'f11', 'class']`.

The 'ID' column is a unique identifier for each sample and the 'class' column has two values, 1 and 2. The remaining columns consist of numerical values characterizing each observation. In this project, you don't have to worry about what these column names mean, all you need to know is that they are measurement values for the different classes, and that the two classes tend to have the different feature values.

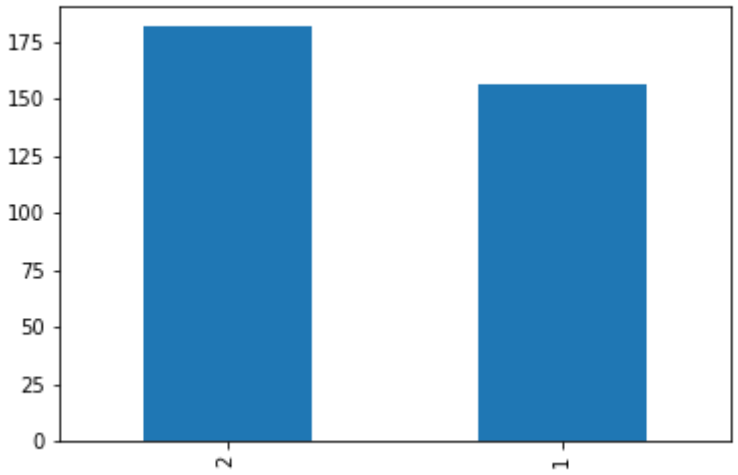
4. **Your Tasks**:

Following are the main tasks you are required to do, but you might need to write some other helper functions to achieve each task. **You should avoid using any libraries in this project, except tasks 1-3 where you might need to use pandas, and matplotlib.** Also, for tasks 1-3 you will need to use a **jupyter notebook** since they are visualization related tasks. For the other tasks you are required to use VScode or any other IDE that you are comfortable with.

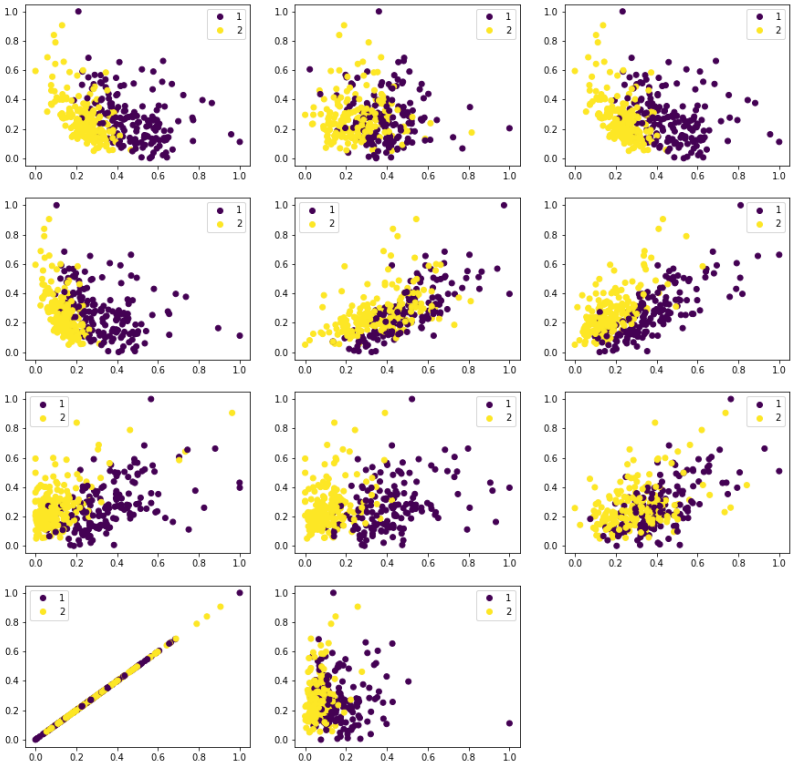
1. Read the *train_data.txt* file with pandas and show the first 5 rows, make sure you include column names when reading.
2. Plot the distribution of the 'class' column. You can use pandas or matplotlib.
3. Write a simple function that shows the relationship between one feature against the 10 other features. You have to plot a scatter plot of one column of your choice against all other columns (NB: ID and class column are not included). You are as well required to use the idea of subplots.
4. Write a minimum of one function or a maximum of two functions that reads your train and test data files. Please do not use pandas here but other means of reading text data and your functions should also return a list of dictionaries that encompasses the information in your data files. Please put these functions in the *utils.py* file.
5. Inside the *model.py* file, write a class that consists of all methods/functions for the classifier. Please, as a group, you are challenged to come up with an intuitive and simple rule-based classifier. The class should consist of as many functions as possible, and each time you feel that you need to reuse a code, you should place the code into a separate function and then call that function. Please don't use tens or hundreds of if-else statements to build the rule based classifier, but rather use the power of the group and come up with a simple, interesting, and mathematically-inspired solution.
6. Inside the class in *model.py* you should also implement an Accuracy function that evaluates your classifier on the unseen test data. Accuracy is the number of correct predictions over the total number of predictions.
7. Inside the main.py file, write a main function that runs tasks 4-5 as a script and provides necessary outputs (an example of possible output is provided below). (BONUS). Write a function inside the class created in Task 5, that gives the prediction of a sample when provided the sample ID as input.

Below are some of the possible results for some of the tasks.

Task 2:



Task 3:



Task 7:

========= Reading train dataset =========
======== Done reading =========.

========= Reading test data =========
========= Done reading =========.

==== Training classifier =====
======== Done training classifier ==========.

========= Classifying test samples =======
========== Done classifying =======
classifier's Accuracy __ %, classifier correctly predicted __ out of __
==========
========== Program finished ==========

NB:

- You should not use any machine learning based algorithm to develop this program
- You should not plagiarize or cheat. You can read the documentations to understand how various data structures are used but please do not directly look for a solution for a particular task from the internet. Collaborate together and come up with solutions. Remember you are here to learn but not to compete, as such think together and make sure everyone in your group understands what you have implemented.
- After you finish all the tasks, write a short paragraph about your program and how your classifier is working. Use simple words to explain.