Rosolino Mangano
Professor Jairam
CIS 3100
22 May 2024

Project 2 Report

For this project, I was required to select a dataset from Kaggle and apply machine-learning classification algorithms to it. Although the assignment specified choosing two out of three algorithms—Naive Bayes, K-Nearest Neighbors (KNN), and Support Vector Machine (SVM)—in either Python or C++, I opted to implement all three algorithms in both languages in order to get as much extra credit as possible. I chose the infamous Banana Quality dataset for this task.

The Banana Quality dataset comprises features that determine the quality of bananas, categorized as ‘good’ or ‘bad’. Each entry in this dataset includes various attributes such as color, size, weight, and other physical measurements that are crucial for quality determination. These features are also integral for quality assessment and provide a robust foundation for applying machine learning techniques. Despite my professor saying in class that the Banana Quality dataset was rather challenging, I decided to choose it anyway because this dataset is particularly suitable for classification tasks due to its clear binary outcome, therefore making it a perfect choice for this project.

I started initiating the algorithms in Python. I decided to do this because I have been coding in Python much longer than I have been coding in C++. I have been coding in Python extensively for 8 years now meanwhile I learned C++ 4 years ago and have barely used it. And to no surprise, coding the program in Python was by far much easier than it was to do in C++. Having the Naive Bayes program provided was helpful, but implementing the K-Nearest Neighbor and the Support Vector Machine algorithms was also not much of a challenge compared to the C++ version. If I had to choose something to complain about while doing this part, it would be the fact that we couldn’t use any complex libraries at all such as NumPy or pandas, but finding workarounds wasn’t that hard. The Python experience was so much better than C++ that I even managed to implement a simple user interface that allows the user to select which algorithm they want to perform and then once the execution is complete the interface asks the user if they want to either input numbers to see the prediction (good or bad) if they want to change the algorithm they are using or if they want to stop using the program entirely.

Each algorithm was chosen for its unique approach to classification. Naive Bayes, known for its simplicity and efficiency in handling categorical data, served as the baseline. Its probabilistic model was adept at processing the dataset’s straightforward categorical outputs, which included color, size, and weight among others. The implementation required careful manual handling of probability calculations to maintain accuracy without the aid of advanced libraries.

K-Nearest Neighbors was utilized to explore the effectiveness of instance-based learning, where classifications are made by a simple majority vote of the nearest neighbors of each point. This demanded an efficient calculation of distances and an optimized way of selecting the k-nearest points.

Support Vector Machines was implemented to leverage its capability in high-dimensional spaces. This algorithm’s goal to find the optimal hyperplane that best separates the data into two

predefined categories demanded a robust method for handling linear separations. Out of all three algorithms, this one was probably the most challenging for me.

The biggest challenge I faced while doing this project was converting my Python program into C++. I know converting Python to C++ is just switching the syntax of the code, but when I tried to do that, I either got a ton of errors or things wouldn't come out the way I wanted it to. I guess this is because C++ demands more verbose code and lacks the high-level functionalities present in Python since C++ is closer to machine code than Python. When I did eventually get all of the syntax converted, my simple user interface would only show the Naive Bayes option. Once I added the K-Nearest Neighbors and Support Vector Machine algorithms back to the menu, it would still only display the Naive Bayes result. So eventually, I decided to scrap the whole simple user interface for the C++ program so the accurate results would show. Also, I guess I have an older version of a compiler (C++11) because when I tried to run the auto keyword it wouldn't allow me to do so. So I also had to find alternatives to make sure my program ran on the outdated compiler on top of all the syntax issues I was having.

The Python implementation was notably more efficient and less time-consuming which allowed for faster development and testing times compared to C++ which offered valuable insights into lower-level programming aspects such as memory management and manual data structure handling. This contrast between the two environments showed the trade-offs between ease of use and control over system resources. Despite its challenges, the C++ version provided valuable insights into memory management, and the manual implementation of algorithms deepened my understanding of their underlying mechanics. The interactive user interface in Python enhanced usability and facilitated real-time testing and comparisons between different algorithms.

So in conclusion, this project not only strengthened my skills in applying machine learning algorithms but also provided a comparative insight into programming in Python and C++, each offering unique advantages and challenges. This has prepared me for future projects that may require a deep understanding and optimization of algorithms. The hands-on experience with both Python and C++ offered a balanced perspective on choosing the right tool for the task based on the specific requirements and constraints of the project.