**Fall 2024 Georgia Tech**

**CS 7641 Machine Learning Projects**

During my fall 2024 semester as a graduate student at Georgia Tech, I took a project-based Machine Learning class CS 7641. The class consisted of 4 comprehensive projects, each with a separate focus on supervised and unsupervised learning methods, randomized search algorithms, and Bayesian learning methods. The projects required a latex report explaining the methods used, thorough analysis, and results/conclusion. Students were required to choose 2 datasets that would be used for the entirety of the semester. I chose a job automation dataset and a Trending YouTube Videos dataset. Below are short descriptions of the 4 projects that involved either using these datasets or coding methods. **A private GitHub repo containing my latex reports can be shared on request.**

**Project 1 Supervised Learning:** From generating an A+ essay to masquerading as writers, artists, and programmers, AI continues to become more and more advanced, threatening the automation of many people's jobs. Although there are certain skills that AI cannot replace, jobs that often involve repetition of tasks, analytics, troubleshooting, and prediction are at a higher risk of being automated. Determining the likelihood that any given job will be computerized forces us to consider several factors, such as salary and employment statistics combined with skill ratings and industry types. From a machine learning perspective, these characteristics when fed into models can help us better understand patterns in this data, allowing for a thorough analysis on why specific jobs are more likely to be overrun by AI. I hypothesize that jobs that require technical skills and operations monitoring are less likely to be automated, whereas, industries that involve repetitive tasks, regular management, and quality control are a higher risk of being replaced by AI. Data such as employment statistics, skills ratings, and salary levels, will help in identifying clear patterns across a diverse matrix of jobs and will be reliable in providing people with a comprehensive idea of whether or not their job will be automated in the near future.

YouTube, the world’s second most popular search engine, has played a huge role in entertainment over the years, with over 5 billion videos watched daily. Its trending page is updated every 15 minutes and serves as an indicator of current cultural phenomena, showcasing videos of many genres. While factors like views and likes contribute to a video’s popularity, there are many other features that may influence how its trending status will fluctuate. Understanding these underlying factors that contribute to a video’s success on the trending page hold significant implications for content creators and marketers. By identifying and analyzing these characteristics, we can gain insights into the dynamics of video popularity and inform strategies for content creation and promotion. Most of the time, these trending videos tend to reflect universal events, in demand music, or new film releases. However, videos are not always defined by tedious characteristics such as the number of views or likes they receive to qualify for a spot on the trending page. What are some of these other properties that popular videos seem to share? Additionally, what underlying factors result in a longer time frame between a video's publish data and trending date? From a machine learning point of view, using features like views, likes, dislikes, and other metadata about trending videos can help us understand what type of videos tend to take longer to go viral and which ones are quick to the punch. I hypothesize that videos that attract more engagement, especially in terms of views, likes and dislikes are more likely to go viral within a shorter time frame, while niche categories of videos are likely to take longer

**Project 2 Randomized Optimization:** When thinking about optimization, most will automatically infer that it involves finding the best solution to something. For complex problems that contain many "best solutions", finding an optima can be difficult if the algorithm gets stuck at local optima. Others struggle with computational complexity or time needed to reach the best solution. Therefore, there is a trade-off between exploration of different parameters versus exploitation, or honing in on the decision that maximizes a function. To understand this trade-off, two different fitness functions were considered: four-peaks and N-Queens. Like the name suggests, four-peaks is a landscape with a reputation of having four maxima (two global and two local), creating a dilemma of when the algorithm should stop. This provides a perfect illustration of the trade-off, given that the entirety of the search space should be explored while avoiding getting stuck at local optima. I hypothesize that for four-peaks, the genetic algorithm might outdo random hill climbing and simulated annealing, due to its strategy of exploring the entire space and maintaining a population of solutions. The objective of the N-Queens function, on the other hand, involves figuring out where to place a certain number of queens such that they cannot attack each other. The non-triviality in this problem is that as N increases, it can get challenging to find the best positions for the queens. Therefore, for this algorithm, I hypothesize that for a smaller number of queens, random hill climbing will perform well for local searches. As N goes up, simulated annealing and genetic algorithm might perform better since they cover a larger search space.

**Project 3 Unsupervised Learning and Dimensionality Reduction:** The first dataset, job automation data, is a compilation of various jobs and information about each job, such as employment number, average salary, hourly wage and annual wage at different percentiles, etc. The purpose of this dataset is to assess what factors affect the likelihood of any particular occupation getting automated in the near future. It contains 23 binary variables, representing the industries of the jobs, and 19 continuous variables for 617 observations. Each row in the dataset represents a unique job, identified by a federal standard called Standard Occupational Classification (SOC) code. One may assume that features such as, salary, industry, and skills can vary a lot among SOCs, resulting in a diverse dataset. However when looking at over 600 jobs, overlap can occur and natural groups tend to form. Therefore, clustering algorithms will work well on this dataset, due to the fact that jobs can have similar characteristics to one another. Since there may be specific differences in features such as ratings and salary among various occupations, clustering algorithms may suggest more clusters to capture the nuances and patterns in the data. When it comes to dimensionality reduction on the job data, PCA will capture the most variance with less features, focusing on important factors like salary, employment, skill ratings. Since PCA assumes linear relationships, it may do well in dealing with the possible redundancy of correlated features, such as annual and hourly wage.

The second dataset, YouTube data, contains statistics about trending YouTube videos published in 2017-2018, such as likes, video category, etc. These variables can be used to analyze what specific attributes of a video contribute to its success, which in this case is the time elapsed until the trending date. It contains 15 binary variables, representing the category of the video, and 16 continuous variables for 687 observations. For clustering, expectation maximization would be suitable for finding clusters of videos that exhibit similar engagement patterns such as views and likes. This method might be able to accommodate for overlapping characteristics that trending videos might share. For dimensionality reduction, PCA can handle continuous features by capturing variance in the data. For this dataset, statistics like views and likes might be correlated, and it is important to deal with such features before modeling.

**Project 4 Markov Decision Processes:** A Markov Decision Process (MDP) is a powerful framework that can be used for making decisions in an environment where outcomes are uncertain. One such application of this process is in blackjack, a card game where the objective is to reach a hand value close to 21 without exceeding it. An MDP is well-suited for this scenario because of the game's sequential nature and the opportunity to incorporate probabilities of drawing various cards. With the states representing the player's hand value, the dealer's card, and a usable ace, there are many combinations of cards that make up a small state space. Additionally, the game embodies a stochastic process, as the outcome of the drawing any one card is based on random probabilities in the deck. Therefore, this problem is interesting because it introduces the challenge of both getting a better hand and avoiding exceeding the limit of 21. Since the game's outcome is determined at the end, the rewards are not constant until a player wins or loses. Lastly, every turn in blackjack ends with a hit or stand action, resulting in multiple decisions to be made throughout the game. These properties when combined result in a dynamic environment for a MDP to solve effectively.

A problem with a larger state space on the other hand, cartpole, involves balancing a pole on a moving cart by moving left or right. Multiple continuous variables make up this state space, such as the pole's angle, velocity, etc. This is also a stochastic process, given the variations in the environment and the actions governing the movement of the pole. Unlike blackjack, cartpole has a constant reward each time the pole remains balanced. It is interesting to see how long the pole can stay upright without falling, which is what MDP helps to maximize.