**Machine Learning Case Study**

M. Ruthvik Reddy

CB.SC.P2AIE25013

**Vibe Coding**

**What?**

“Vibe coding” is a newer term (coined by Andrej Karpathy in early 2025) for a style of software development that relies heavily on natural-language prompts + AI (specifically large language models, LLMs) to generate code, often with minimal manual writing or review.

Vibe coding is a modern style of software development where programmers rely heavily on natural language prompts and AI tools, such as large language models, to generate code. Instead of manually writing every line, developers describe the desired functionality in plain language, let the AI produce the implementation, and refine it through iterative prompting and testing. This approach prioritizes rapid prototyping, experimentation, and accessibility, making it easier for both beginners and experienced developers to turn ideas into working software quickly. While vibe coding accelerates development and reduces boilerplate work, it also comes with challenges such as limited code understanding, potential security issues, and difficulties in long-term maintenance if the generated code is not carefully reviewed.

**Why?**

Vibe coding is used because it significantly reduces the effort spent on repetitive or boilerplate programming tasks, allowing developers to focus on problem-solving and creativity. By using natural language prompts with AI tools, routine operations such as data cleaning, visualization, or setting up basic code structures can be automated quickly. This speeds up development, makes coding more accessible to beginners, and enhances productivity for experienced programmers by eliminating the need to manually write standard code snippets. Ultimately, vibe coding is adopted to save time, improve efficiency, and streamline the overall development workflow.

**Where used?**

I applied vibe coding to handle redundant and time-consuming tasks such as plotting graphs with Matplotlib and performing data cleaning operations with Pandas. Instead of manually writing repetitive code, I leveraged natural language prompts with AI-assisted coding to quickly generate scripts, visualize data, and streamline preprocessing. This approach not only saved development time but also allowed me to focus more on model building and analysis while ensuring efficiency in routine coding steps.

**Psephology**

The scientific study of elections, voting trends, and electoral behavior is known as psephology. The word comes from the Greek word psēphos, which means "pebble," and refers to the ancient Greek custom of casting pebbles into urns to cast ballots. The term "psephology," which was first used in 1948 by British historian R.B. McCallum, refers to the study of election outcomes, voter turnout, and the efficiency of electoral systems.

A specialist who investigates elections, voting trends, and electoral behavior is known as a psephologist. To understand how people vote and predict election outcomes, they examine demographic data, opinion polls, historical election data, and other pertinent factors. Psephologists assess the efficacy of electoral systems, identify patterns in voter behavior, and offer insights to the public, researchers, and political campaigns using tools from political science, statistics, and occasionally machine learning.

**What role does ML play in psephology?**

Election outcome prediction: Machine learning models, including Random Forests, Gradient Boosting, and Neural Networks, are trained using demographic data, polling data, and past election outcomes. These models offer data-driven insights that supplement conventional polling techniques by predicting which candidate or party is most likely to win in a constituency or region.

Voter Behavior Analysis: Voters can be divided into groups according to their interests, demographics, and voting history using clustering algorithms like K-Means or hierarchical clustering. This allows psephologists to understand swing voter populations, pinpoint groups that are likely to support a specific party or candidate, and adjust campaign tactics accordingly.

Polling and Survey Optimization: By modifying answers to correspond with population distributions, machine learning can address biases in surveys and polls. Additionally, predictive models assist in completing missing data, enhancing the accuracy and dependability of election forecasts.

Finding Odd Voting Patterns or Inconsistencies: Unsupervised machine learning algorithms are able to identify anomalies and electoral fraud, highlighting areas that require additional research. This preserves the integrity of the election process, assures transparency, and helps spot any inconsistencies in the results that are reported.