# Danny Quang

Email: dquang@ucsd.edu
Mobile: 925-577-8678

Github: github.com/dquangucsd
Portfolio: dquangucsd.github.io/dq-portfolio

## Education

- **University of California, San Diego** — San Diego, California
  *Bachelor of Science - Computer Science; GPA: 3.644* — *Expected Graduation: 2024*

  ***Relevant Coursework:*** *Advanced Data Structures and Algorithms, Design and Analysis of Algorithms, Computer Systems and Systems Programming, Software Engineering, Object-Oriented Programming, Multivariable Calculus, Linear Algebra, Probability and Statistics, Multivariate Statistics/Statistical Learning, Discrete Maths, Representation Learning, Intro to Machine Learning, Recommender Systems and Web Mining, Database System Principles*

## Skills

- **Languages**: Python, Java, C, C++, Assembly, R, Swift, HTML, JavaScript, CSS, SQL
- **Frameworks**: ScikitLearn, Scipy, Matplotlib, Numpy, Pandas, Tensorflow (Keras), RegEx
- **Tools**: LaTeX, MATLAB, Vim, GitHub, Jupyter Notebook, VSCode

## Work Experience

- **Tutor for Upper Division Class: CSE 110**: March-June 2023
  - Duties: Attend weekly Supervison/ASE(s) and staff meetings, perform individual and group tutoring, lab support, content review and debugging, and Slack support
- **Data Analyst Intern at The Center for Community Energy**: June-October 2022
  - Position: Data Analyst Intern on the Carport Market Research Team
  - Plotted points from a file on a map with Python and embedded it on a site online with github.io
  - Web scraped multiple car company websites targeting key words with BeautifulSoup4(bs4), pandas, and numpy
  - Aided in market research involving V2G data. Website: https://centerforcommunityenergy.org/danny_quang/
- **Webmaster for the American Institute of Aeronautics and Astronautics**: May-Present
  - Position: Webmaster
  - Maintained and updated the website for elected positions and events for AIAA UCSD

## Projects

- **Top Movies app**: Swift
  - Pulled data from a movies database of the current top movies in the United States
  - Designed an app that pulls top data using AlamofireImage in Swift from a movies database (https://api.themoviedb.org) and listed them out with a cover image on the left, the title at the top, and a synopsis directly below.
- **File compression and decompression**: C++
  - Built Huffman Tree based on frequency of each character using Priority Queue with an overloaded operator
  - Used Post-order traversal to write Huffman Tree and encoded file contents to the compressed file
  - Reconstructed Huffman Tree using a stack and write byte to the decode the file
  - Checked memory leaks and designed test cases to verify output and test for edge cases
- **Developed Web Application for Multi-Function Photo-Editing Platform**: JavaScript, CSS, HTML
  - Styled a responsive home, gallery, and edit page with CSS with rotate, brightness, etc.
  - Used localStorage in JavaScript and indexDB so users can work locally and store large images
  - Developed unit tests with Jest and end-to-end tests with Puppeteer
- **Developed Prediction Algorithm and Visualization for Beer Ratings**: Python, Numpy, matplotlib, sklearn
  - Utilized matplotlib to plot data, pandas to clean and organize data. Used techniques like TF-IDF(TfidfVectorizer()) to represent reviews as a vector
  - Tried different combinations of prediction techniques (e.g. bag of words+linear regression (with train-test splitting the data to train and test model), TF-IDF+SVM+linear regression, etc.)
  - Utilized GridSearchCV to determine optimum parameters to predict rating, and Pipeline to streamline the process
- **Web Scraped tradingview and cars**: Python, BeautifulSoup4, Requests, Numpy, github.io
  - Automated changes in stock prices for top market cap companies
  - Used requests, BeautifulSoup4, to go to the websites, parse html, and numpy to organize and create the dataset
  - Used the same process to filter out for important data for cars within a budget and certain criteria
  - Plotted data and used github.io to host the plots
- **Analyzed COVID-19 data**: Java
  - Found which day had the highest average amount of cases for a particular race category
  - Found the number of cases for the race category and got the average/mean. Compare each date and return the date with the highest average cases for the race category.