1. **Description of the project**
    - Scrapes over 3000 job listings from simplyhired.com and usajobs.com
    - Reformats data to be useful to a machine learning model
    - Feeds data into two different machine learning models (MultinomialNB & Logistic Regression), to compare results
2. **How to use**

Github Repo: [AndrewCoale/CMPSC_445_Midterm_Project]

The data from the websites is collected into its original csv forms in scrape_simply_html.py and scrape_usa_html.py. clean_simply_data.py can then be run to clean the simplyhired data, and put it into the final data file cleaned_jobs.csv. Finally, clean_usa_data.py cleans the few jobs taken from usajobs, and appends them to that csv file.

The models can be run via LogisticRegression.py and NaiveBayes.py, respectively. The graphs used for section 7 can be generated via LR_Visualization.py.

3. **Data Collection**
    - Used tools: Selenium, BeautifulSoup4
    - Data sources:
        - [20 Best software engineer jobs in New York, NY (Hiring Now!) | SimplyHired]
        - [USAJOBS - Search]
    - Collected attributes: Job Title, Company, Location, Salary Range
    - Number of data samples: about 4000 before cleaning/filtering
4. **Data Preprocessing**
    - Data cleaning:
        - Drops all rows (around 800) that lack a listed salary
        - Standardizes job titles into a small, predefined list based on keywords, later dropping the few (about 100) that lack said keywords
        - Adds an experience column based on keywords found in the job title
    - Data integration & Data ingestion:
        - Takes in data from two separate job websites, cleans it into the same format, and puts it into the same csv file
    - Data Description and Meta data specification in the data repository with a sample data
        - Job Title: the title of the position
        - Company: the name of the company offering the position
        - Location: the city and state the position
5. **Feature Engineering**
    - How data is processed and prepared for the machine learning model after loading from data repositories

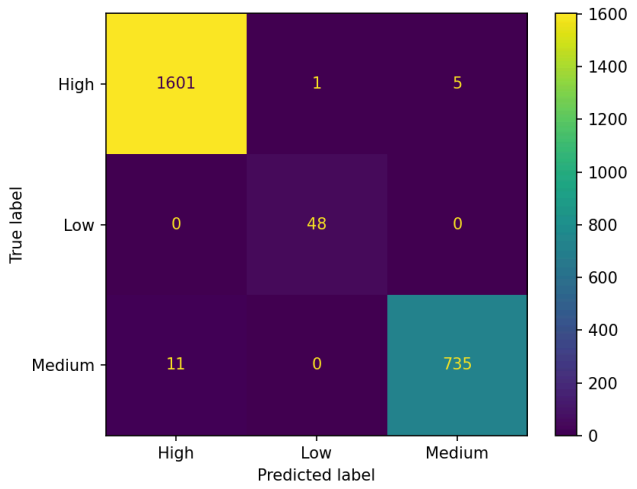
- The salary column is split into 3 different categories (high, medium and low) based on number thresholds, and then converted into numerical representations
- All nonstandard job titles are removed from the data pool
- One-hot encoding is applied to categorical features

6. **Model Development and Evaluation**

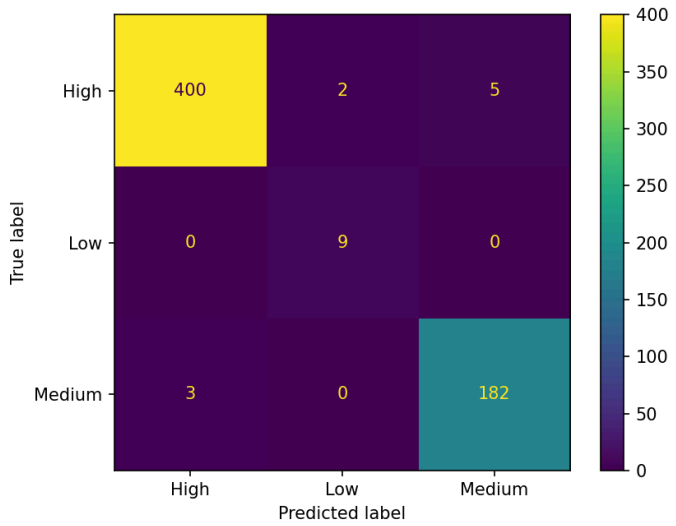
○ Train and Test data partition: 20% of the data is allocated for testing, the rest is fed into training the models

○ **Salary Prediction : Model-1**

1. Machine learning model: Multinomial Naive Bayes
2. Input to model: 5 job attribute columns from csv file
3. Size of train data: 2400 jobs
4. Attributes to the machine learning model
   - Job title
   - Company
   - Location
   - Salary
   - Experience
5. Performance with training data: 99.30% accuracy

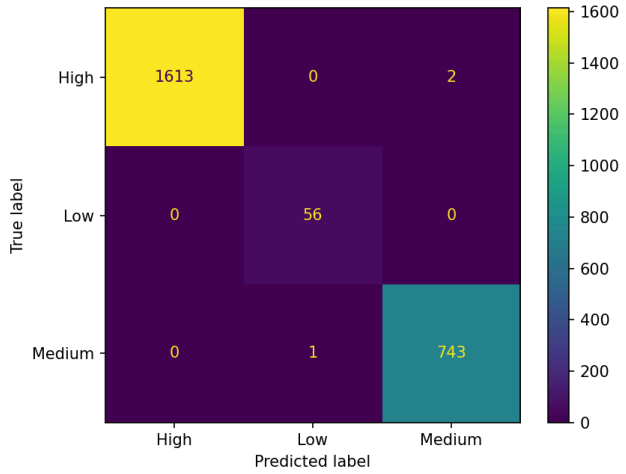


6. Performance with test data: 98.18% accuracy

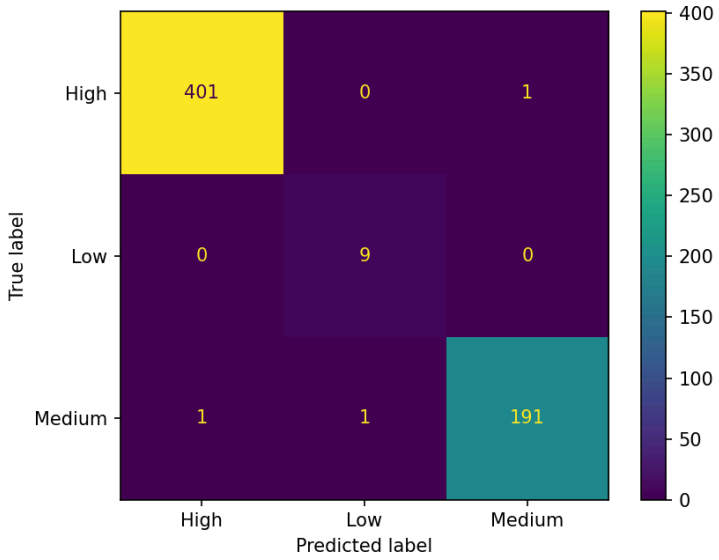


- **Salary Prediction : Model-2**

1. Machine learning model: linear regression
2. Input to model: 5 job attribute columns from csv file
3. Size of train data: 2400 jobs
4. Attributes to the machine learning model:
   - Job title
   - Company
   - Location
   - Salary
   - Experience
5. Performance with training data: 99.88% accuracy



6. Performance with test data: 99.5% accuracy



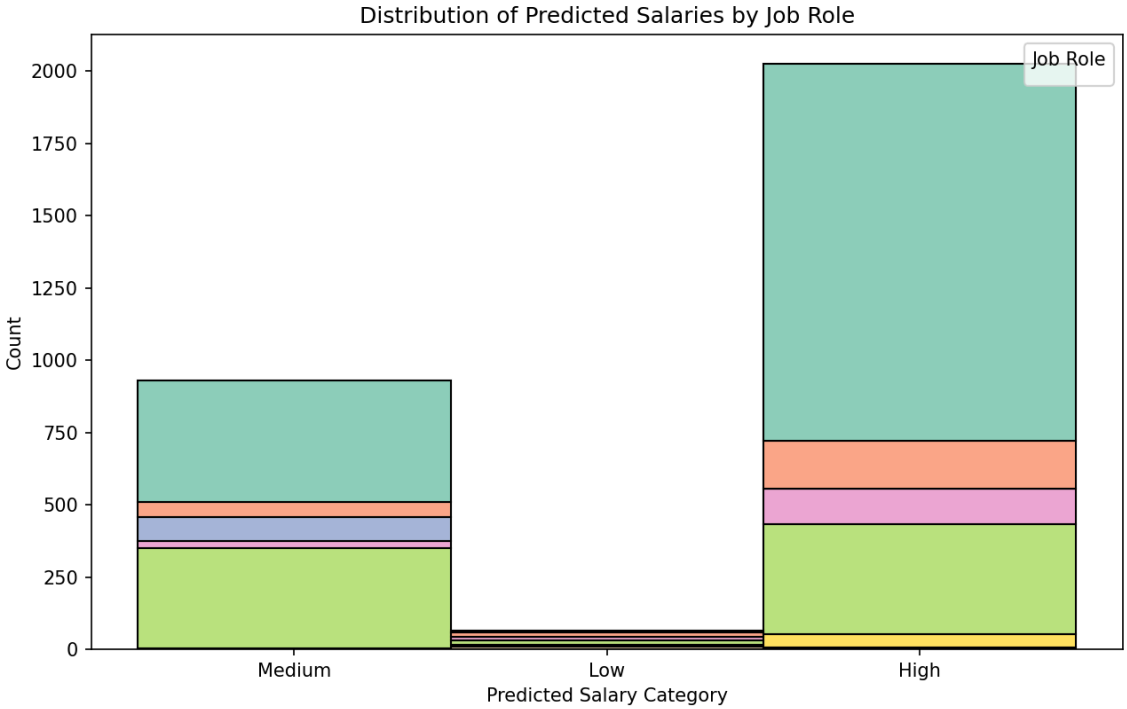
- **Skill Importance**

1. Description of the exploited feature importance techniques to identify the most important skills for different job roles in computer science, data science, and AI
2. Identified important skills for 5 job roles of your choice

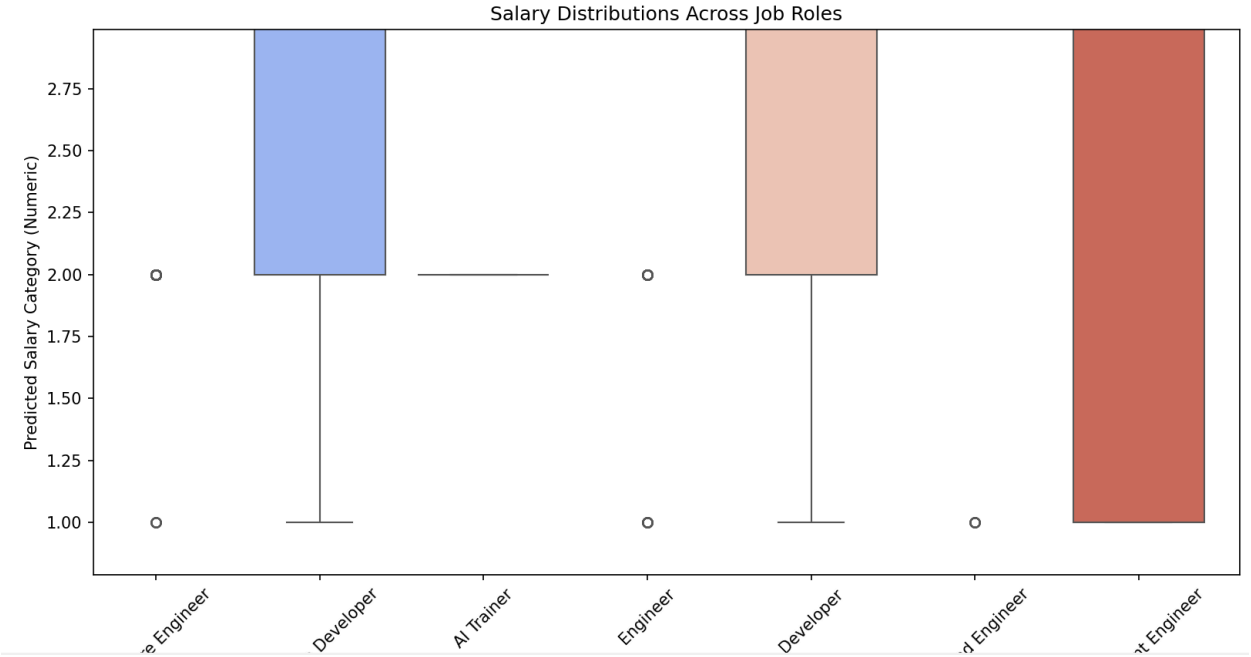
As I did not successfully retrieve required job skills, they could not be included in the model. General skills of mathematical ability, pattern recognition, attention to detail, patience, and problem solving could be applied to any of the job roles listed.

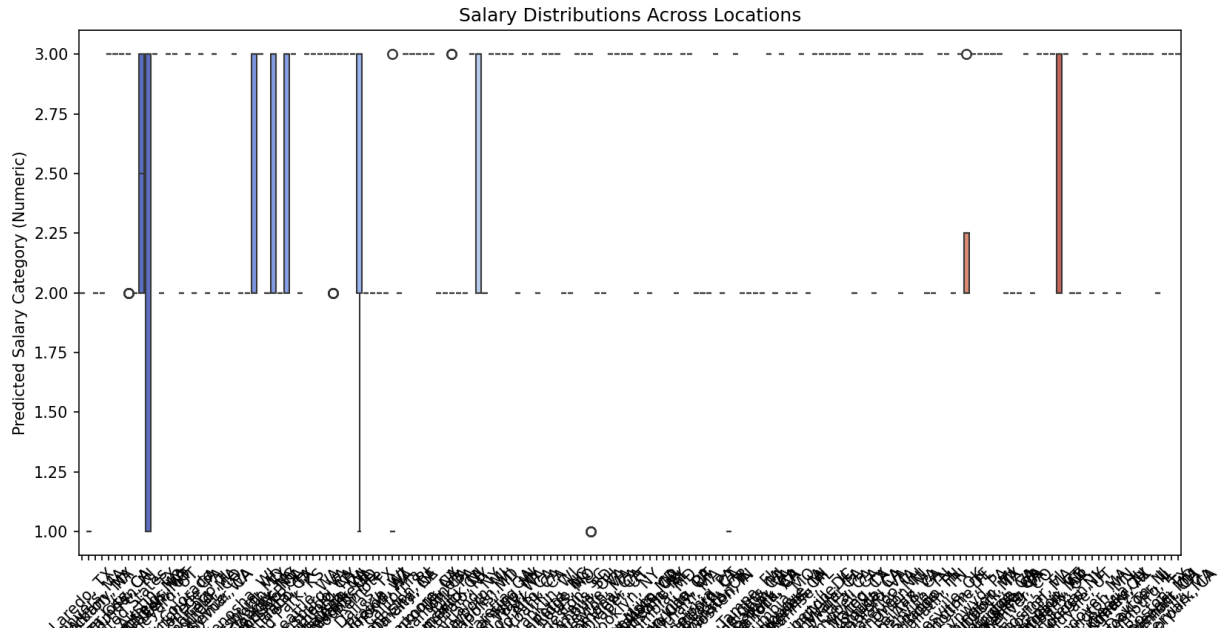
7. **Visualization**

- Histograms to show the distribution of predicted salaries for different job roles.



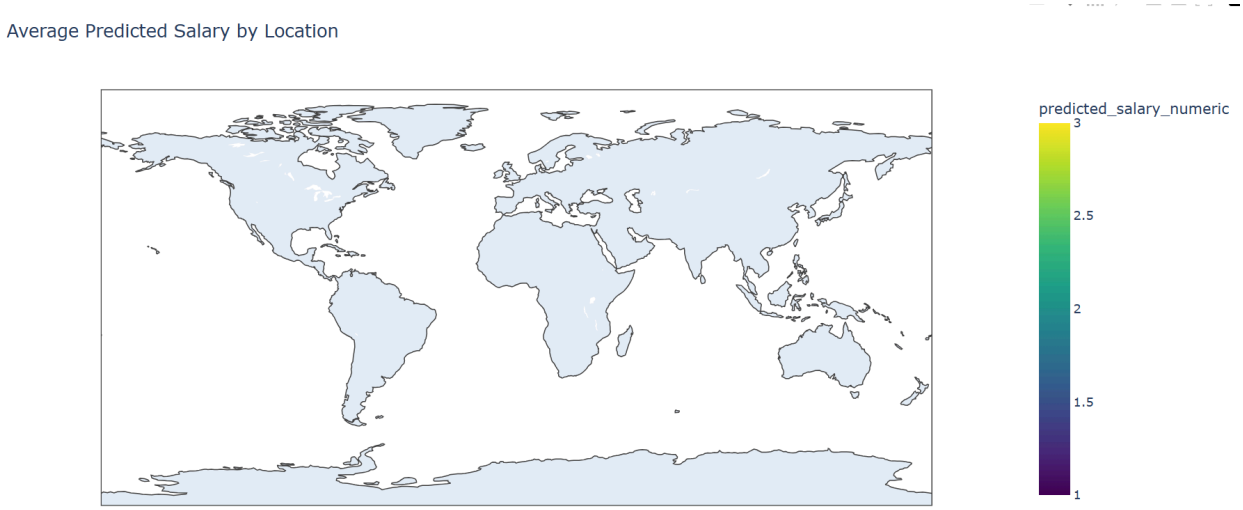
- Box plots to compare the salary distributions across different job roles and locations.





- A map to visualize the average predicted salaries across different locations. (You can use libraries like Plotly for interactive maps)

(I tried, have my attempt that showed no actual data)



- Bar plots to show the importance of different skills for each job role. Use feature importance scores from your model to rank the skills
- Heatmaps to visualize the importance of skills across different job roles.

(These last two won't work out, since I did not include job skills in the data)

8. **Discussion and Conclusions**

- Project findings, including insights gained from data analysis

Both of these models resulted in very high prediction accuracy, but a lot of that probably resulted in grouping continuous salary data into 3 large boxes, and turning it into a classification problem. If I had tried to predict the salary more exactly, I'm sure the results would be less than pretty.

- Challenges encountered during model development

Scraping the data alone took a great deal of effort, over a day of just looking through websites and troubleshooting beautifulsoup and selenium. I had to figure out how to make it click through to new pages to scrape more than 20ish results. Developing the actual models took even longer, and you can see how messy visualizing those ended up being. The data I was able to reasonably obtain en masse via webscraping was not particularly detailed, and lacking relevant skills made several points earlier in this report impossible to cover.

- Recommendations for improving the performance of the model.

Figuring out a way to retrieve required job skills would almost definitely help to make more precise predictions of salary. Job title is a rather arbitrary thing, that is not standardized across companies, and my method of standardization was sloppy at best.