Assessment of major contributors to salary and career outlook

Part 3

Bohui Nong, Tyler Landivar, Tianye Chen - CSE 587

Problem Statement: Younger students struggle to decide what they want to major in and become. This is a big problem because college normally determines which field students will be working in, meaning them deciding on a whim is not a good thing. They might decide later on that they want to change majors/profession, costing them a lot of money and time. One way to help them out is to provide real information on the outlook of their careers as this information along with their interests play a huge part in their decision process.

Contribution: In this study, we attempt to see what factors contribute to salary and job opportunities. We hope to build effective models and informative visualizations that showcase exactly how these deciding factors affect careers. This will help the younger students make a strong, informative decision on their career goal and get them on the right track

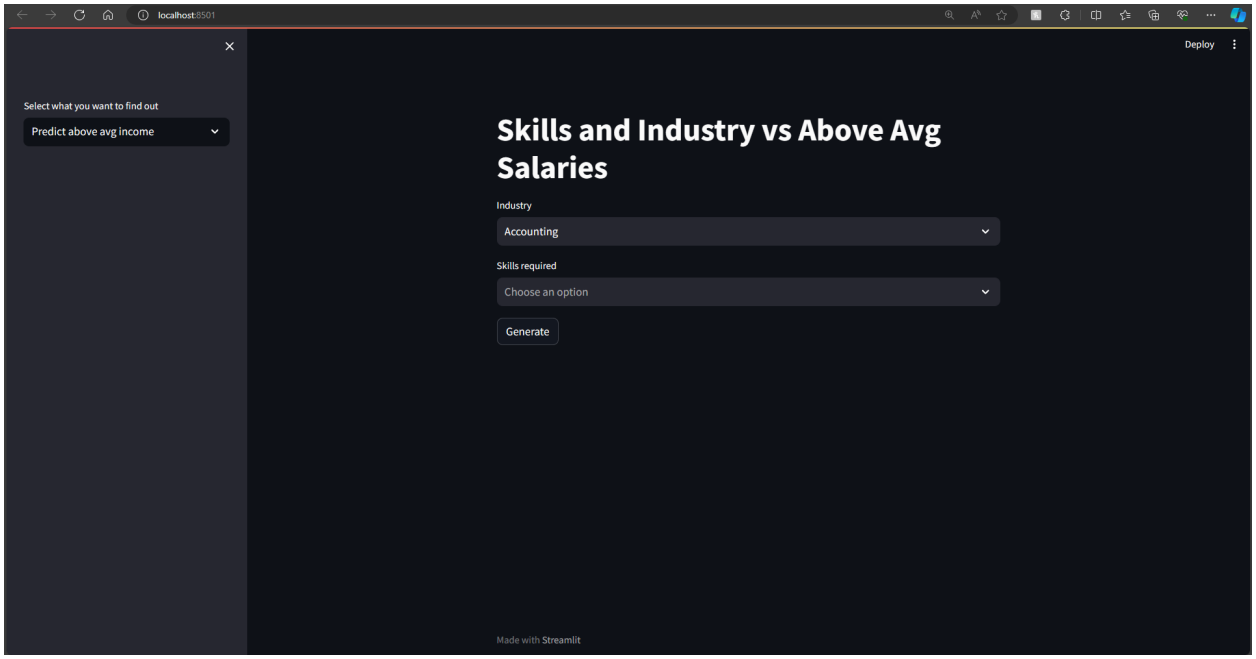
Dataset URL

https://www.kaggle.com/datasets/rajatraj0502/linkedin-job-2023?select=job_postings.csv

Instructions on how to run the data app

This app will run on the localhost in your default browser with the help of the python library streamlit. In order to run it properly, you'll need to create a virtual environment (python) with the following libraries installed: streamlit, pandas, numpy, scikit-learn, fuzzywuzzy, tensorflow (for keras) and pickle. Then go into the phase 3 directory with main.py through terminal/command

line with the environment activated and run the following command: streamlit run main.py. You should be met with a screen like this



Here we see the landing page which is actually one of the pages where you can use one of our ML models. On the left, you can see the navigation. If you click on the drop down bar in the navigation box, you can select which page you would like to move to so you can use our other ML models.

Information about code files and pages

The way streamlit is designed, we can designate a file to define a corresponding webpage and then bring it all together through functions in one central file (which is our main.py file here). It's pretty similar to defining components in React or Angular. Since main.py is pretty straightforward with it being our landing page and responsible for navigation, we'll go through the other files which define the different pages that utilize our models.

Models and archive directory

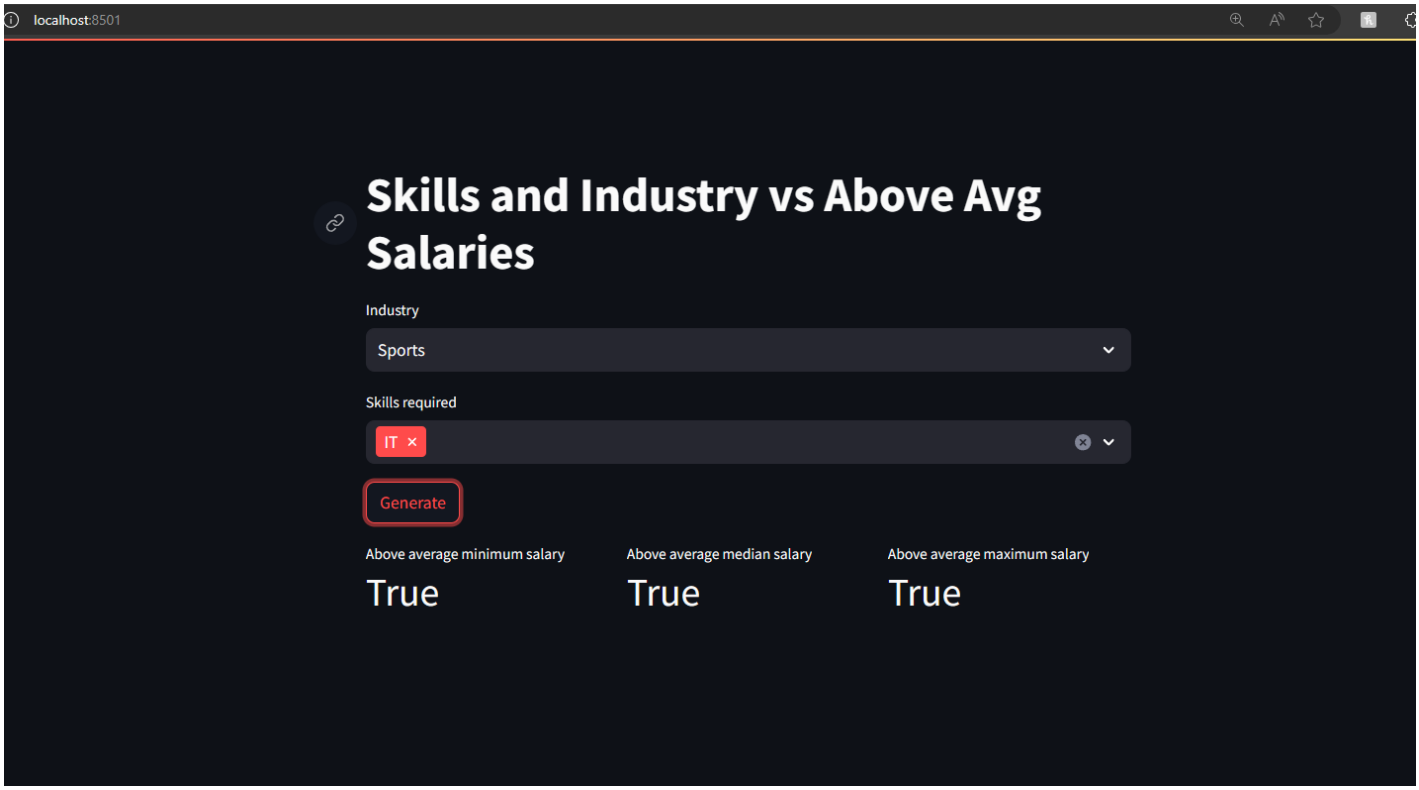
Models is where we keep our saved models and other important information like predefined clusters that our page files reference. This is just used to store information that our pages will need. Archive serves a similar purpose as that's where our dataset lies.

preprocess.py

This file preprocesses the dataset and returns important information like the dataset itself and encoder objects. We'll need the encoder objects for when we convert the encodings back to their original values. The original values need to be used when users select the values for the input features (e.g. the value 47 for industry won't make any sense to a user, but the value "healthcare" will). This file and the class/function it defines is only run once when we build the data app.

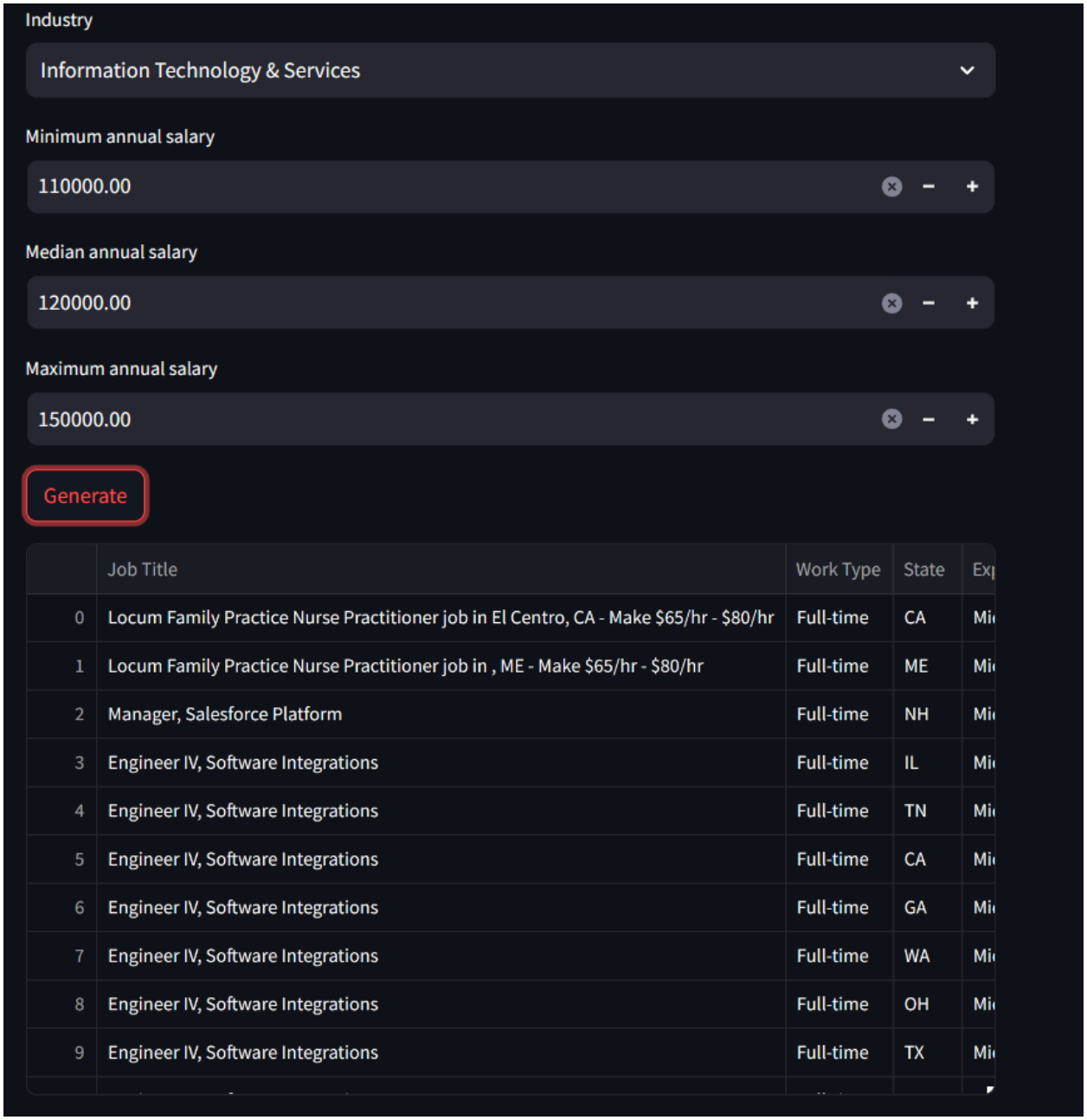
nb.py

This file defines the page where we use our Naive Bayes model to predict if a job will yield an above average minimum, median, and maximum salary when given a set of skills and industry for that job. It utilizes the saved model from models folder. No tuning was necessary as our model performs well enough. On this page, a user can select their own values for the input features using the dropdown selections we have defined. When they have chosen their features, they can then click on the generate button to call our naive bayes model on those features to see if those features can lead to above average minimum, median, and maximum salaries. Below is what a user might see



kmeans.py

This file defines the page where we use our KMeans model to find similar jobs based on the input features employee count, min salary, max salary, median salary, industry, location, formatted_experience_level, formatted_work_type, and title. It utilizes the saved model from the models folder. No tuning or other preprocessing was necessary as our model performs well enough and we use our saved model and clusters. On this page, a user can select their own values for the input features using the dropdown selections and text inputs (constrained to numbers) we have defined. When they have chosen their features, they can then click on the generate button to call our KMeans on those features to see jobs similar to the one they inputted. The KMeans algorithm of course labels the input sample with the correct index for the clusters array which we use to retrieve all the stored similar jobs. The similar jobs are then placed into a dataframe to be displayed on the page. Below is what a user might see

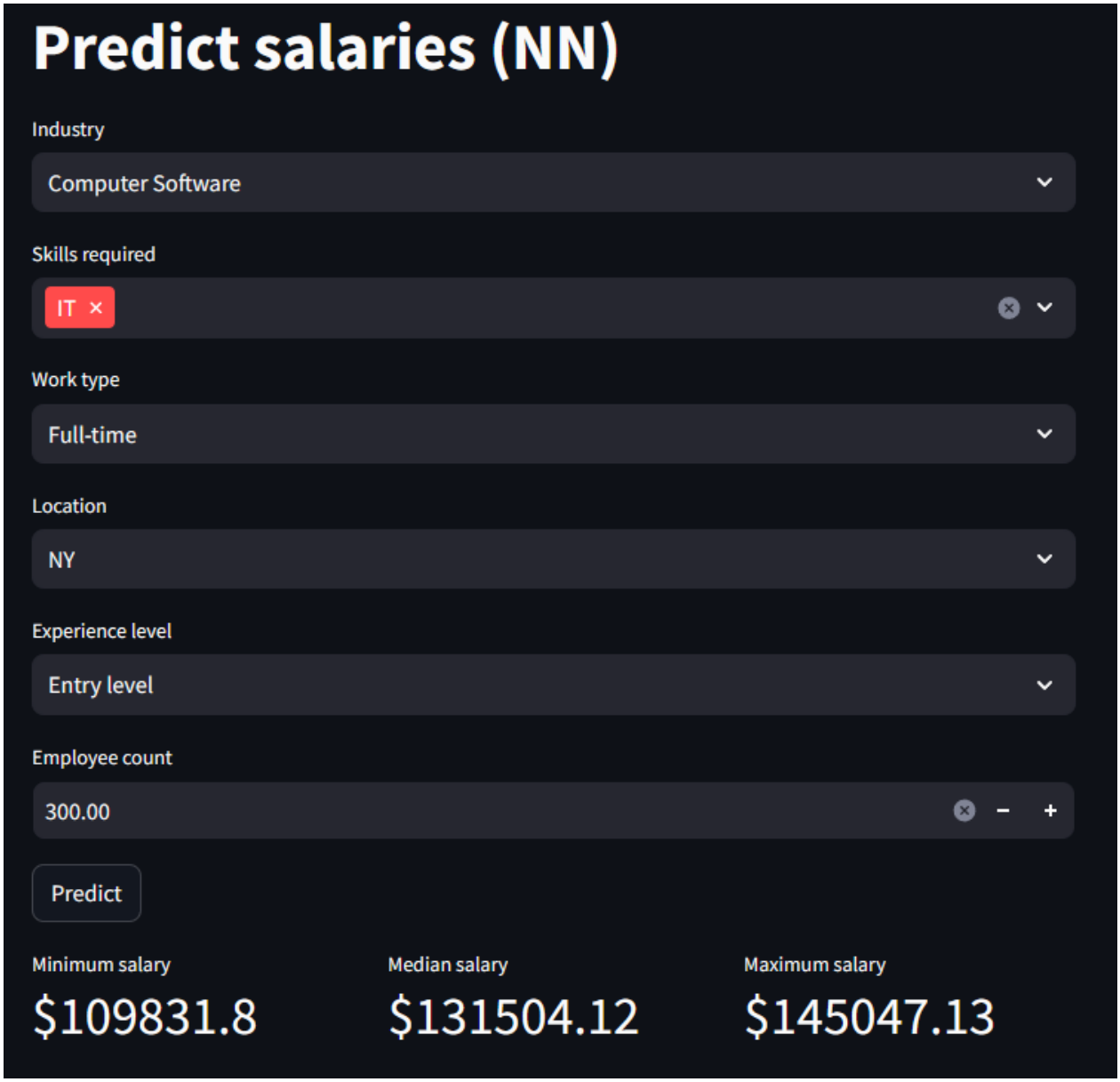


	Job Title	Work Type	State	Ex
0	Locum Family Practice Nurse Practitioner job in El Centro, CA - Make $65/hr - $80/hr	Full-time	CA	Mi
1	Locum Family Practice Nurse Practitioner job in , ME - Make $65/hr - $80/hr	Full-time	ME	Mi
2	Manager, Salesforce Platform	Full-time	NH	Mi
3	Engineer IV, Software Integrations	Full-time	IL	Mi
4	Engineer IV, Software Integrations	Full-time	TN	Mi
5	Engineer IV, Software Integrations	Full-time	CA	Mi
6	Engineer IV, Software Integrations	Full-time	GA	Mi
7	Engineer IV, Software Integrations	Full-time	WA	Mi
8	Engineer IV, Software Integrations	Full-time	OH	Mi
9	Engineer IV, Software Integrations	Full-time	TX	Mi

nn.py

This file defines the page where we use our Neural Network model to predict the salary ranges based on the user's input on industry, required skills, work type, location, experience level, and employee count. No tuning was necessary as our model works sufficiently well and we directly load our pre-trained Keras model from TensorFlow from phase 2 which is nn.h5 in the models

folder. The user may input their values through dropdowns, multi-select, and numeric input, The page then takes the user's inputs and passes them through our model, the model will return an object containing the predicted minimum salary, median salary, and maximum salary. We take these results and display them individually on the page to the user. Below is what a user might see.

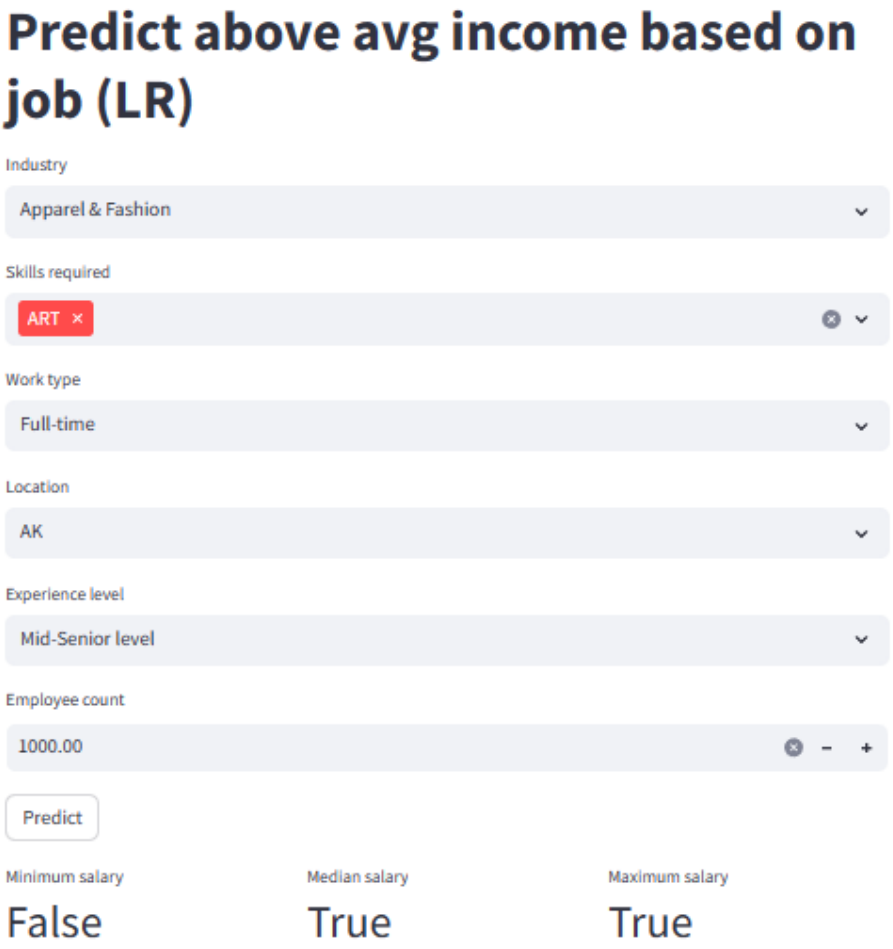


dt.py

This file defines the page where we use our Decision Tree model to give an estimated experience level based on the user's inputs on the industry, employee count, annual salary, and job title. No tuning was necessary as our model works sufficiently well and we directly load our decision tree model which is dt.pkl in the models folder. The user may input their values using dropdowns, numeric input, and alphanumeric input. Since the user's value for job title may vary, we match the user's input to an existing job title in our dataset, the closest match is automatically selected and the user may choose from the top 10 closest matches using the dropdown when the Job title field is filled. The matching is done via fuzzy string matching with the fuzzywuzzy library. The page then will take the user's inputs and categorize it as either Entry level, Associate, Mid-Senior level, or Management, the result is displayed on the page for the user. Below is what a user might see.

lr.py

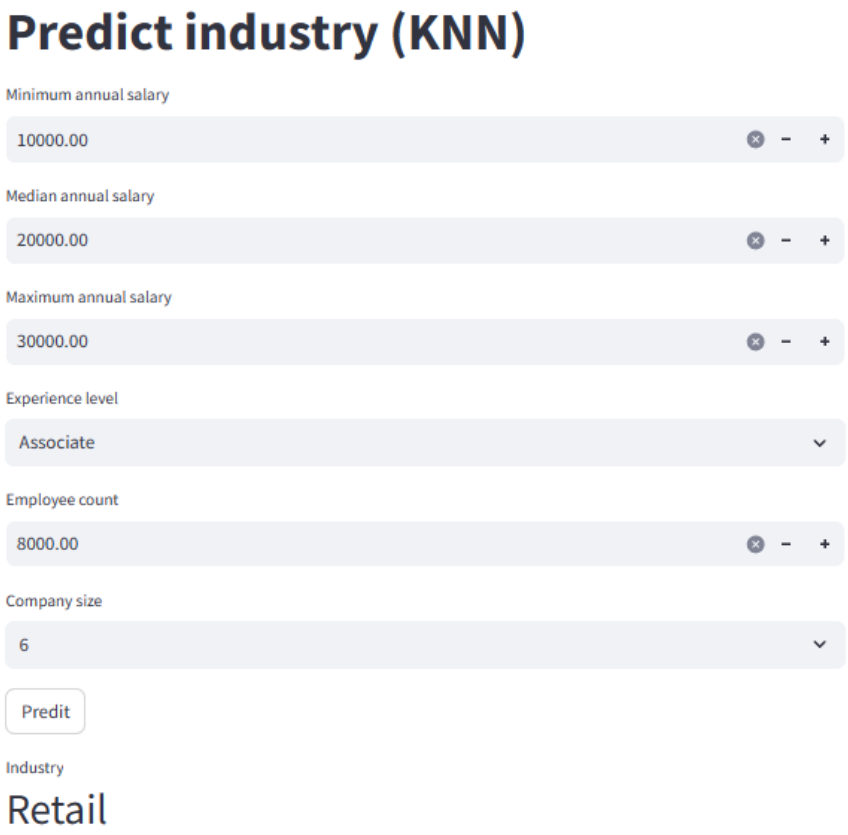
This file defines the page where we use our Logistic Regression model to give a binary prediction of whether the job will have above average minimum, median and maximum salary based on the industry, skills required, work type, location, experience level, and employee count. No tuning was necessary for this model since the model gives us pretty good accuracy. As shown above, there is already a model that is used to predict the binary outcome of above average minimum, median and maximum salary in the Predict above avg income tab. However, that model isn't based on the jobs but instead based on the industry and skills required. The Logistic Regression instead takes consideration of the jobs to give a deeper prediction instead of a broad prediction. As such, Predict above avg income based on job tab will be where the user enters the information in order for the model to predict the result. After the user finishes entering the information listed in the website and clicks the predict button, it will load the pre-trained Logistic Regression model which is lr1.pkl, lr2.pkl, lr3.pkl in the models folder. Each corresponding pickle file is used for predicting above average minimum, median and maximum salary. Then it will pass the user's inputs as parameters into those three models to predict whether the job will have above average minimum, median and maximum salary as shown below.



knn.py

This file defines the page where we use our K-Nearest Neighbors classifier to predict the industry based on the user's inputs on minimum salary, median salary, maximum salary, experience level, employee count, and company size. There are some tuning that are performed in order to increase the accuracy of the model such as looking for the optimal k value and having the weight set to distance instead of uniform. We directly load our pre-trained knn model which is knn.p in the models folder. It will be loaded when the user clicks the predict button in the knn

prediction page which is in the Predict industry tab. The knn prediction page will require the user to enter the minimum salary, median salary, maximum salary, experience level, employee count, and company size inputs in order for these parameters to be passed in the pre-trained knn model. After the user clicks on the predict button, the parameters will be passed into the pre-trained knn model and it will give the user the prediction of the industry as shown below.



Thoughts based on our results

There are no exact recommendations of course as it's up to the individual to select their career path. However, we do have some pretty good information that may help them. From our data, we were able to see that experience level (e.g. associate vs. management) has the largest impact on salary. So, if making a lot of money is a person's goal, then we would tell them that getting

higher degrees like masters can help them get the management positions to boost their odds of a high paying job. We also noticed an uneven distribution of jobs location wise. As somewhat expected, states with bigger cities like New York, Florida, and California had more positions listed than say Ohio. The pay was also predicted to be higher in our data product for these states as well. We would tell our audience that location is also something that should be highly considered as it could affect the number of opportunities available and the pay. There are more useful information like this that our audience can find out for themselves using our data product. We offer many different ways for users to change different information about the job to see how much it could affect it; whether the target is actual salary, if the salary would be above average, or other similar jobs. For example, someone could see the difference in pay between an associate and management in IT through our NN page. By offering these services and ways to test out different job configurations, our users can make a more educated choice when selecting their future career, which is our primary objective. We would expand our project by collecting more training data. Although our models perform pretty well, there is a notable uneven distribution of jobs (e.g. in terms of location). We feel that getting more training data would greatly improve our data product and provide more accurate information to our users who use it to help select their career. We would also try to include other properties like years of experience, and acceptance rates of jobs to give more information about things that could impact salary to our users.