Paychex Job Title Prediction

Rohaan Ahmad, Joey Tschopp, Keitaro Ogawa, Gus Vietze
University of Rochester
500 Joseph C. Wilson Blvd. Rochester, NY 14627
rahmad3@u.rochester.edu, jtschopp@u.rochester.edu, kogawa@u.rochester.edu, gvietze@u.rochester.edu

1. Introduction

Paychex provides human resources, payroll, and employee benefits outsourcing services tailored for small- to medium-sized businesses. Currently, Paychex has limited data on aspects beyond the scope of payroll processing, so being able to predict such information based on their existing data would be valuable. A key piece of missing information is employee job titles, which can range widely from software engineer to cashier. Our task was to develop a predictive model for these job titles using available user data to enhance Paychex's understanding of their workforce.

Paychex utilizes a standardized job title classification system provided by the North American Industry Classification System (NAICS), which groups similar job titles into a unified OEWS Code. Our project had two main objectives: first, to create a model that accurately predicts job titles (OEWS codes), and second, to improve our understanding of the dataset and identify effective approaches. One potential application of our model is to recommend a shortlist of job titles for clients when they input data, thereby expanding the data available to Paychex. Additionally, our model could assist Paychex with compensation benchmarking, labor cost forecasting, tax credit recommendations, ensuring equal pay across job demographics, and gaining comprehensive economic understanding.

2. Dataset Description

Our data came in 4 separate tables: Clients, Workers, Payments, and Products. All tables were given to us in CSV format. The Clients data set contains information about clients/employers including the industry that they are in, indicators about their size and financial performance, their legal status (LLC, S-Crop, etc.), what US states the clients are based in, the number of employees, if the operate yearly or seasonally, and other information about each employer. The Workers table contained information about the workers such as gender, ethnicity, whether they are full-time or part-time, their age, which client they work for, if they are salaried or not, and their job title. The Payments data set included information at the payment level. Each payment had an associated worker and client along with the type of payment it was (wages, overtime, commissions, tips, health and benefits, retirement employer contribution, or other) and the amount that was paid. The Products data set contained information about the products that a given client purchased from paycheck. For our analysis, we did not find the Products data set to be extremely helpful as it did not seem to be predictive of job titles in our preliminary analysis, so it was largely ignored throughout our project. Below is a graphical representation of our data set to give you a better understanding of how each field relates to another.

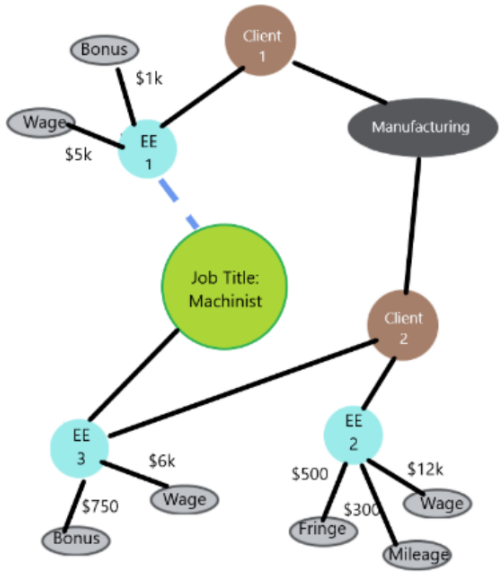

Figure 1. Graphical Representation of Some of the Data Fields

A majority of the data present was masked to prevent the leakage of private employee information. Our target class had 850 unique job title labels, which were masked in the form of OEWS Codes. The dataset contained information for 67,278 Clients (companies) and 4,226,517 workers. All 4 tables we were given for datasets were able to be merged on the Client ID variable.

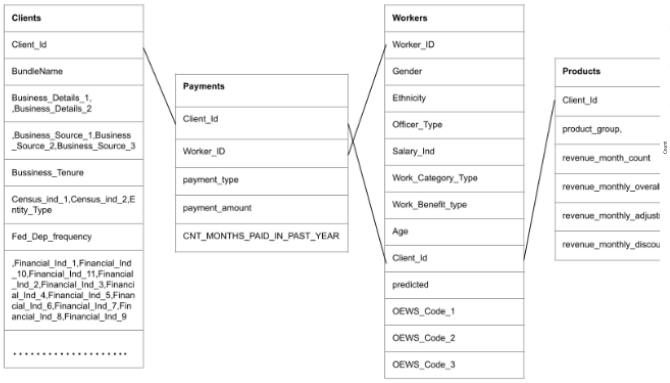

Figure 2. ER Diagram of the Fields

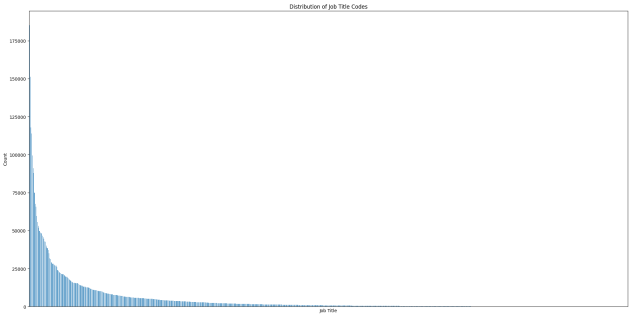

Figure 3. Job Title Code Distrbution

The final thing of note about our data set is that we were given a data set that maps job titles (OEWS Codes) to a larger more general job title code, the OEWS major. The OEWS major represents an in-built class hierarchy within

The dataset we were given possessed some unique characteristics that were important to be mindful of when it came to exploratory analysis and model development. One was that a majority of the variables were masked, which made it difficult to find or interpret any possible connections between different variables in the dataset. Another was that our variables were overwhelmingly categorical, which is an important factor to consider when beginning model development as preprocessing is generally different for categorical variables when compared to continuous variables.

3. Exploratory Analysis

Given the masked nature of our dataset, we were extremely limited in our ability to conduct meaningful Exploratory Data Analysis. However, we still attempted to glean some meaningful information from the data that would help us with further model development. The first thing we looked at was the distribution of the target variable, the OEWS code. In doing so we established that the target field was highly imbalanced. Figure 3 shows the distribution of the job title codes.

Our dataset contained many labels for the target class we were attempting to predict, which additionally came with the problem of many of the labels occurring infrequently in the data set. 5% of job title labels had a frequency less than 30, and 16% of the labels had a frequency less than 100. Additionally, only 16% of the job titles had a frequency over 50000. This means that only 16% of job titles individually represent $.0125\%$ of our total data set. This class imbalance led to many problems that had to be addressed later in model development.

Upon further exploration, we found similar trends with the OEWS Major and Industry columns. The distribution of the OEWS major code can be found in Figure 4.

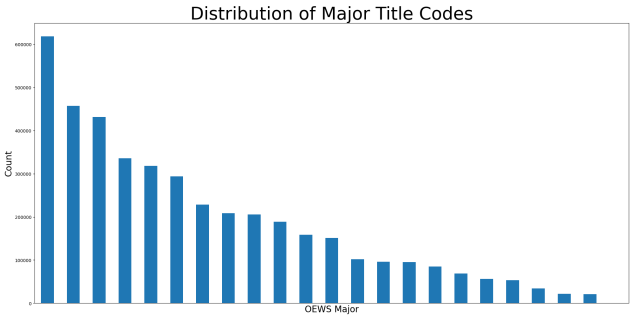

Figure 4. OEWS Major Distrbution

As you can see in Figure 4, the OEWS major is quite imbalanced but much more balanced than the OEWS code.

After this, we also investigated other variables in our data set. The variables: Ethnicity, Officer_Type, Nr_1099_1YrAgo, Nr_Employees_1YrAgo, Financial_Ind_8, and BundleName, all had sparsity problems, which was important to be mindful of when we began model development.

4. Model Development

Before we started with model development, there were a few preprocessing steps we needed to do. The first was to drop the prior mentioned sparse categories. After that, we joined the clients and workers data sets on client ID to give us all of the client data at the worker level. Next we min-max scaled our numerical and quantile data, one-hot encoded categorical variables with low cardinality, and label encoded categorical variables with high cardinality. We also needed to figure out how to take data from the payments data set that was at the payment level and convert it to data on the worker level to join with our combined clients and workers data frame. To do this we created one column for each payment type in our combined clients and workers data frame and added the amount paid to that worker for that specific type. Below is a visualization of how that was

done:

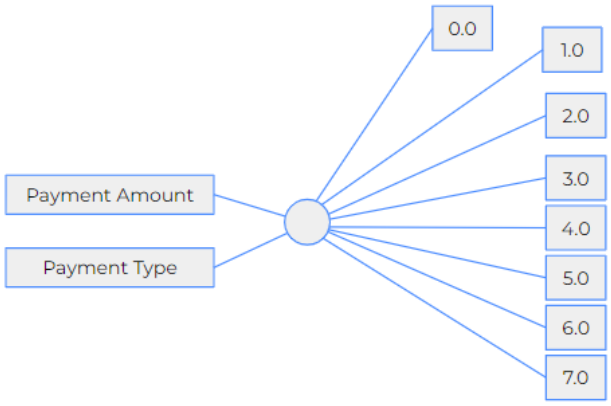

Figure 5. Payment Feature Modeling

Onto model development, it was important for us to establish a performance baseline to gain an understanding of how difficult the task of job title prediction would be. Using a dummy classifier that simply predicted the most frequently occurring class for every job title, we obtained an accuracy of 4.3%. We also created a model that assigned values based on randomly assigning classes proportionally to the distribution of job titles and obtained an accuracy of 1.9%. With this out of the way, we began to test a wide variety of models on subsets of our dataset.

For our initial model evaluation, we worked with a small subset (about 2.5%) of the data. Based on these initial results, and our literature review, we found that the most interesting models for us to try to implement were Random Forest Models, Artificial Neural Networks (ANN), and K-Nearest Neighbors (KNN). These models seemed to perform pretty well on the initial samples, and additionally, were found to be relatively robust to classes with many low-frequency labels.

With our models of interest selected, it was time to move on to scaling up our models for processing the whole dataset. With 4 million rows and 104 columns, we had a solid amount of data to work with. When working on the entire data set we were unable to scale up the Random Forest of KNN models due to their large memory overhead. We tried utilizing, Bluehive a campus-provided Linux cluster used for computationally expensive research. Even with the extra resources we still could not train them without huge accuracy losses.

We then moved on to working with two different models, a neural network that directly predicts job titles and a hierarchical model that mixes random forest and neural networks to predict job titles. The non-hierarchical model we call the flat-classification neural network.

For the flat-classification neural network, we utilized an architecture of three hidden layers, with 850, 512, and 128

Model	Best Accuracy
Baseline Accuracy	4.3%
Random Forest	40%
Neural Network	37%
Support Vector Machine (SVM)	36%
Naive Bayes	29%
Logistic Regression	19%
KNN	36%

Figure 6. Accuracies of Initial Models

neurons respectively. We utilized a dropout of 0.2 percent on all layers. It was trained using the Adam optimizer with a 0.001 learning rate. For the high-cardinality categorical variables, we implemented embedding layers to help in data preprocessing. These embedding layers had an output dimension of 10 and were learned during model training. To improve robustness to unseen data during training a random number of rows for each categorical value are mapped to zero, the index for unknown category values.

The hierarchical model uses the in-built class hierarchy provided by the OEWS major code to help predict job titles. A visualization of how the class hierarchy works is below:

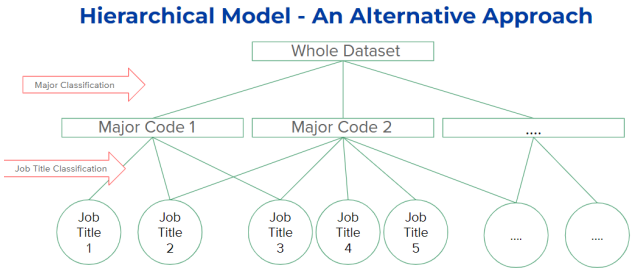

Figure 7. Overview of Hierarchical Approach

First, we built a similar neural network to the flat classification network to classify the OEWS major code. The embedding layers are handled the same way as they were in the flat-classification network. A visualization of the basic model architecture is seen below:

The major code neural network used a softmax activation function on the output layer. It was also trained using the Adam optimizer but this time with a learning rate of .00001 and with gradient clipping set at 10 to prevent exploding

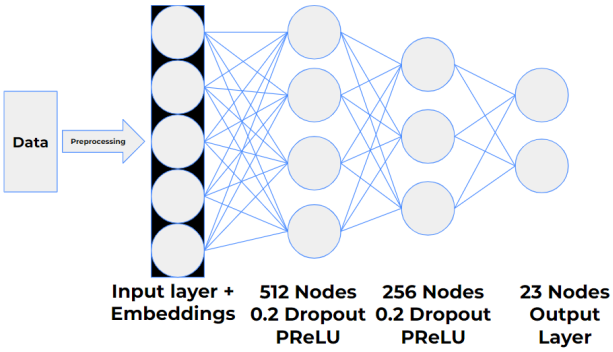

Figure 8. Neural Network Used for Major Classification

gradients. When designing both of these neural networks many experimenters were done to try to improve their classification accuracy. Experimentation included: trying both L_1 and L_2 regularisation methods, trying different levels of dropout, trying different numbers of hidden layers, trying different learning rates, trying class weights to improve the classification of underrepresented classes, and using various embedding output dimensions. We ultimately settled on these architectures as they performed the best, with as little overfitting as possible.

After the major code classification neural network classifies the major code, the next step is to put the data through a random forest classifier trained on only data from that given major code. For this purpose, we suggest a class-weighted random forest model. Class weights were computed on the training set to make sure the model generalizes well to underrepresented target classes.

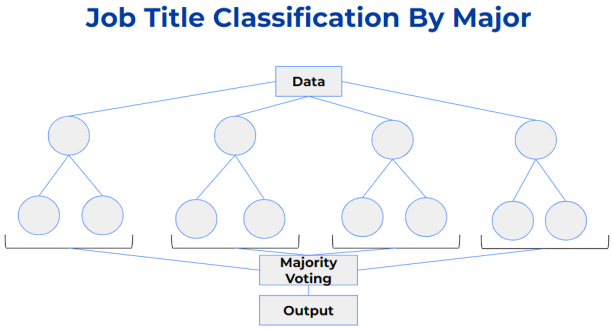

Figure 9. Random Forest Model Diagram (Used in Job Prediction within Major)

Above is a basic outline of how the random forest model works. As you can see, many decision trees are fit, and then vote to predict the output class. For our random forest model, we experimented with both top one and top 3 class prediction. This could be easily extended to top 5 class prediction to increase accuracy further.

5. Performance and Results

When modeling the whole data set using our flat-classification neural network, we got accuracies around 23%. This accuracy was not in line with our expectations, especially when we considered our accuracies for the subsets of the data we did for the initial model evaluation. The flat-classification neural network was also extremely prone to overfitting which can be shown by the figure below which details the training and validation accuracy across epochs:

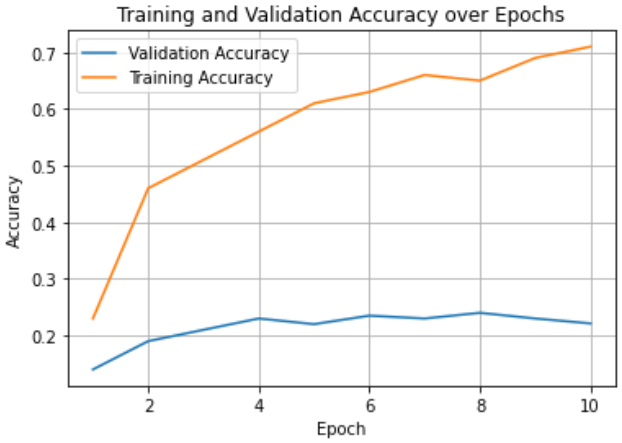

Figure 10. Train vs Evaluation Accuracy by Epoch

We were additionally unable to obtain accuracies for Random Forest and KNN using the whole dataset due to modeling troubles with the memory requirement.

From our hierarchical model, our results seemed much better. For the ANN major classification, we obtained an accuracy of 56% and an F1 score of 0.54, which was certainly an improvement from the accuracies we were obtaining from the original ANN. We chose to evaluate the models based on the F1 score in addition to accuracy due to its ability to indicate problems related to class imbalance in predictions. To further assess discrepancy in model performance for different major codes we have decided to graph the true vs predicted values for each major code across the test data set. That is displayed below:

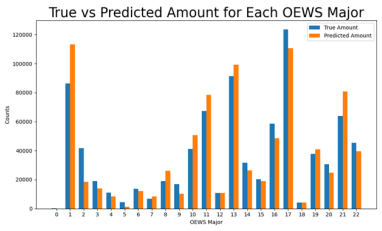

Figure 11. True vs Predicted Major Codes

For the Random Forest models, we found that with the

major code known, the prediction of job titles became much more accurate.

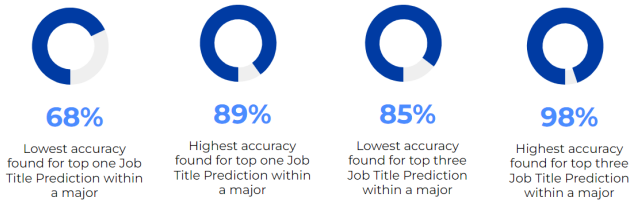

Figure 12. Accuracies for Job Title Predictions within Major

As shown above, our lowest found accuracy for single job title prediction was 68%, with a highest of 89% over the set of major codes we tested on. We were able to test 10 out of the 23 different major codes but due to time constraints, we were not able to run on all major codes. Given that Paychex expressed interest in being able to choose a job title from a list of suggested job titles, we also wanted to see model performance for the top three most likely job titles. For this, we obtained an accuracy range of 85% to 98%.

We also looked at our model performance in relation to the different ethnicities and genders in our data. To do this we examined how our major code prediction model performed for each gender and ethnicity value.

The resulting accuracy and f1 scores are displayed below:

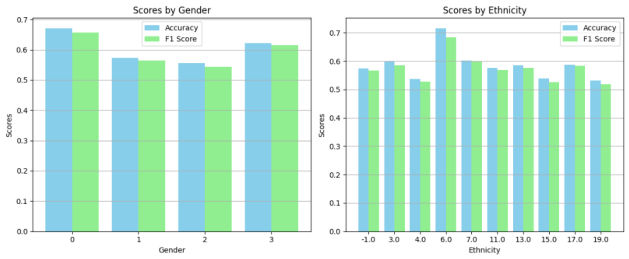

Figure 13. Accuracies and F1 Score for Different (Masked) Genders and Ethnicity

These initial results suggested that there could be a statistically significant difference in model performance, however, as is shown by the graph the effect size seems to be small as the accuracy values do not differ by large margins for ethnicity or gender. Given these results, we decided that it would be useful to run a chi-square test to see if the gender or ethnicity value was related to the accuracy of the model. To do so we added a 'correct' column to our test data which was 0 if the model incorrectly predicted the major code and 1 if the model predicted it correctly. Running the chi-square test on ethnicity concerning model correctness we found that there was a significant difference as we got a p-value of 6.85×10^{-182} with an alpha of 0.05. This suggested that the difference was highly significant. To further investigate, we plotted the residuals of this chi-square test below:

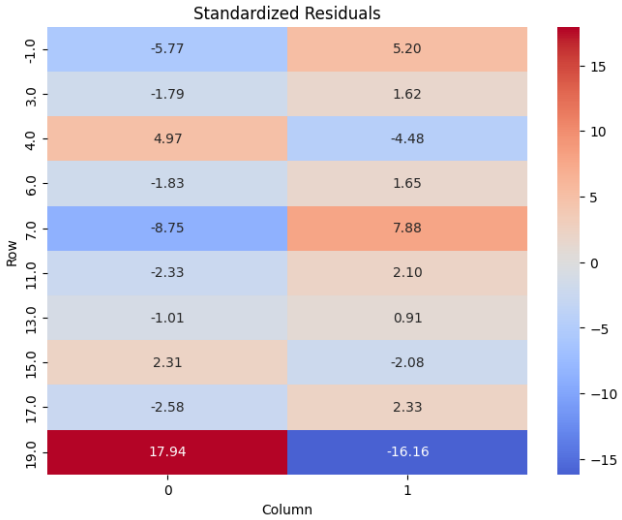

Figure 14. Standardized Ethnicity Residual Values

These results suggest that our model had a more difficult time predicting job titles for workers with ethnicity values of 4.0 and 19.0 and had an easier time predicting job titles for workers with ethnicity values of 1.0 or 7.0.

We then performed a similar chi-square test to test for a significant relationship between model correctness and gender. Again we found that there was a highly significant difference as we got a p-value of 3.500×10^{-236} with a significance of 0.05. We again investigated the residuals with our results shown in figure 15.

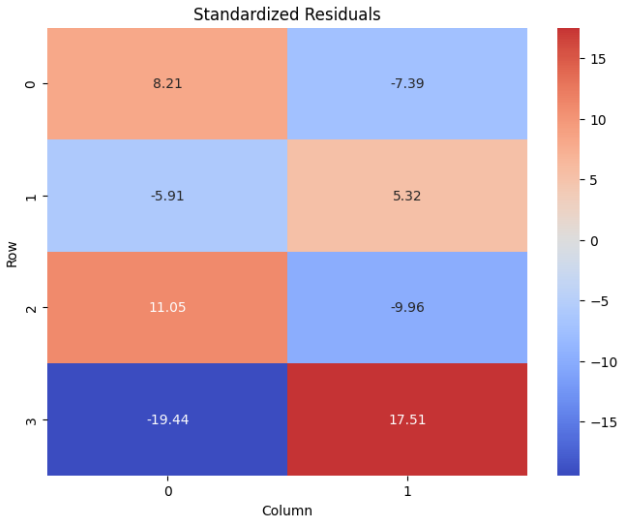

Figure 15. Standardized Gender Residual Values

These results suggest that our model had a more difficult

time predicting job titles for workers with gender 0, or 2 and an easier time for workers of gender 1 or 3. In this case, it is worth noting that 0 is the assigned value for nulls in the gender column.

Given our results in our fairness analysis, it is clear to see that our model does have bias issues, but due to the small effect sizes, it is unclear at this point if this is a major cause for concern.

Additionally, we also extracted feature importances from one of our random forest models which predicts job titles based on major code. This was requested by Paychex to gain more insight into what features predict job titles well. Our results are displayed in figure 16.

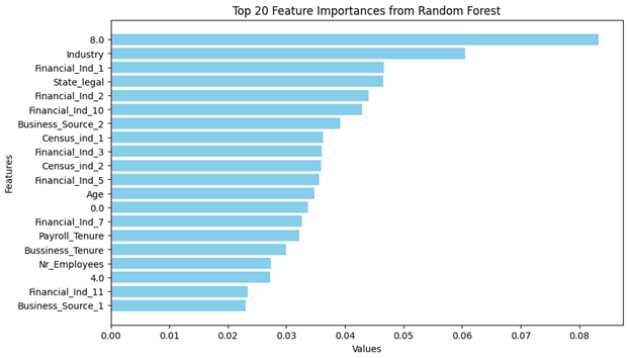

Figure 16. Random Forest Feature Importances

These results are obtained through training the random forest model on one specific major, however, the results are quite stable across different majors. The main takeaways are that payment type 8.0, which we believe to be wages, and industry are the most predictive. State legal and some of the financial indicators are also quite predictive. These results can be used during the creation of future models or to further fine-tune our suggested models.

6. Conclusion

In conclusion, our results indicate that the hierarchical classification approach is the most promising. Predicting the OEWS major code was significantly easier than predicting job titles directly.Moreover, predicting job titles based on the major code proved to be significantly easier and more promising compared to the traditional flat classification approach. However, we encountered bottlenecks in model performance, primarily due to challenges in accurately predicting the major codes. Enhancements could be made, particularly in optimizing embeddings before model training. Additionally, while Random Forest models are effective within each major, their training can be resource-intensive and time-consuming when incorporating new data. Exploring models that offer more dynamic retraining could address this issue.

Furthermore, incorporating temporal elements into job title classification could be beneficial. The job market is constantly evolving, with new roles emerging, existing roles becoming obsolete, and compensation fluctuating based on market demand. Addressing the temporal dynamics of job markets could enhance the model's applicability and accuracy, though this was not feasible with our current dataset due to the lack of temporal information.

Next Steps

1. **Optimize Embeddings:** Experiment with different methods for optimizing embeddings before training the model. Techniques such as pre-training embeddings or fine-tuning them could improve performance.
2. **Evaluate Dynamic Models:** Investigate alternative models or algorithms that offer more dynamic training and retraining capabilities, potentially reducing the time and cost associated with updating the model as new data becomes available.
3. **Incorporate Temporal Analysis:** Integrate temporal data into the analysis to capture changes in job markets over time. Future datasets should include temporal information to better understand and predict evolving job roles and compensation trends.
4. **Expand Data Scope:** Consider gathering additional data, including more granular job title information and external factors such as industry trends, to improve model accuracy and robustness.
5. **Address Biases:** Further investigate and address any identified biases in the model related to gender and ethnicity. Implement strategies to ensure fair and unbiased predictions across different demographic groups.

References

1. Romero, M. (2022). *A top-down supervised learning approach to hierarchical multi-label classification in networks*.
2. Diuk, C., Strehl, A. L., & Littman, M. (2006). *A hierarchical approach to efficient reinforcement learning in deterministic domains*.
3. Levatic, J., Kocev, D., & Dzeroski, S. (2013). *The Use of the Label Hierarchy in Hierarchical Multi-label Classification Improves Performance*.
4. Luo, J., Hu, J., Zhang, Y., Ye, S., & Xu, X. (2021). *Multi-label Classification Based on Label Hierarchical Compression*.
5. Khemani, B., Patil, S., Kotecha, K., et al. (2024). *A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions. Journal of Big Data, 11*(1), 18. https://doi.org/10.1186/s40537-023-00876-4
6. Khim, J., Xu, Z., & Singh, S. (2020). *Multiclass Classification via Class-Weighted Nearest Neighbors. arXiv preprint* arXiv:2004.04715.
7. Babbar, R., Partalas, I., Gaussier, É., & Amini, M. (2013). *On Flat versus Hierarchical Classification in Large-Scale Tax-*

onomies. In *Neural Information Processing Systems*.