**Predicting Median Annual Salary**

**Final Report for DSC148-WI25**

Timothy Kam

Christine Law

Daniel Zhu

**Introduction**

As students are actively searching for their first job, one of the most sought-after questions is: How much will the salary be? Individuals seek higher education in chances to secure a job with a higher salary, and with the world becoming more technologically driven, certain jobs tend to be on the higher salary spectrum. In this report, we delve deep into LinkedIn job postings to analyze job listing characteristics (location, industry, experience, work type, and company) that can most accurately predict the median annual salary.

**1 Dataset**

**1.1 Identify a Dataset**

The LinkedIn Job Postings (stored as postings.csv) is from the Kaggle website. There are 31 columns ranging from ‘job\_id’ to ‘work\_type’ and 124k unique entries.

This dataset contains 124,000+ job postings from 2023-2024, making the data entries relevant and accurate. Attached to the Kaggle Dataset, there are various csv files that detail specific characteristics of jobs and companies such as benefits, company description, number of employees and more. For the purpose of this project, we decided to focus on postings.csv, company\_industries.csv, and job\_skills.csv.

**1.2 Merging Datasets**

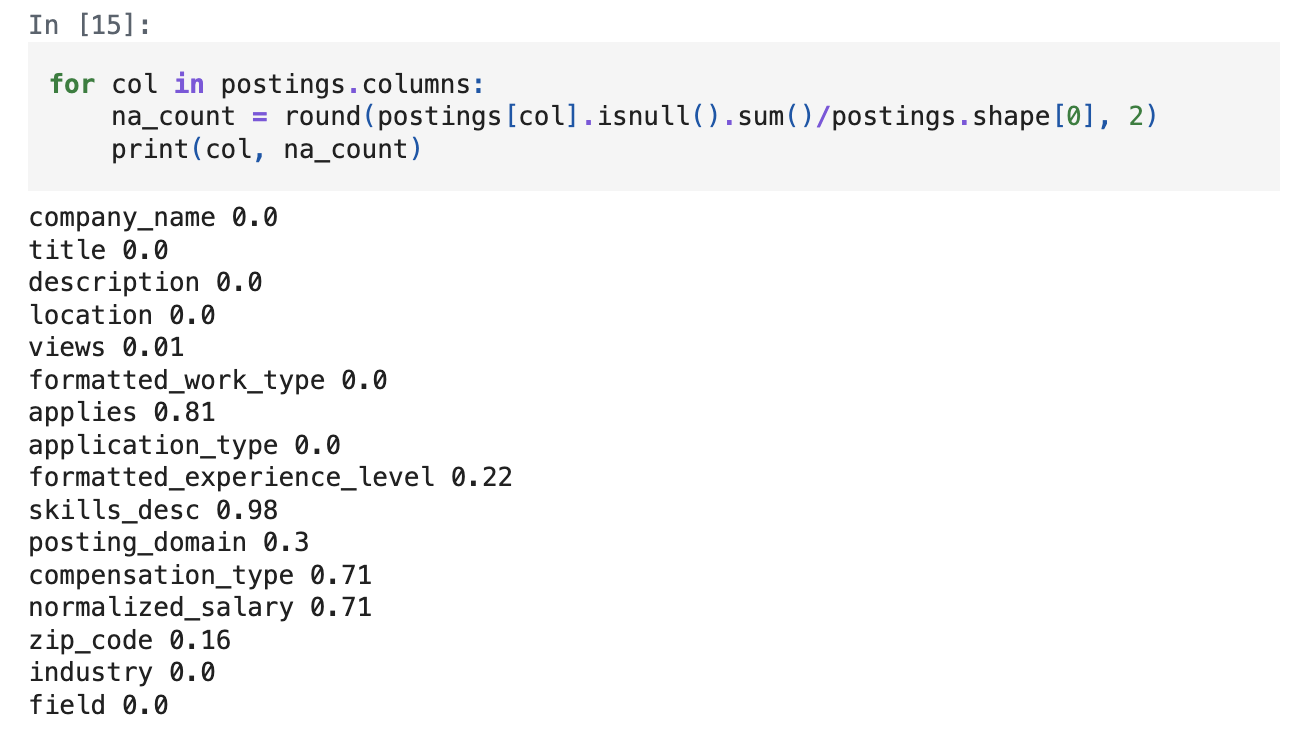
**Merge:** We began by merging three datasets to enhance our primary dataset, postings.csv, which contains job listing information like job titles, company names, pay periods, and other relevant job characteristics. In total, the csv contained 123,849 entries and 31 columns. We then looked at company\_industries.csv which had 24,375 entries and 2 columns and saw that it consists of “company\_id” and the respective “industry” the company identifies as. Since we believe industry is an important indicator of an individual’s median salary, we merged postings.csv with company\_industries.csv using ‘company\_id.’ As a result of merging these two datasets, we got a dataframe with 122,121 entries and 32 columns. This integration allowed us to categorize jobs by industry, adding an essential feature for our predictive model. We then looked into the job\_skills.csv which contains “job\_id” and “skill\_abr,” which indicates what education/skills are needed for its corresponding field. We also believed this is a significant indicator for predicting median salary so to make this data more interpretable, we mapped ‘skill\_abr’ to broader skill categories using skills.csv and merged it into our dataset using ‘job\_id.’ After merging all relevant data sources, our dataset contained 35,116 job postings and 33 columns, ensuring that we retained all necessary job characteristics, industry classifications, and skill data for accurate salary prediction.

**1.3 Data Cleaning**

To ensure that our dataset is optimized, we removed unnecessary columns, handled missing values, and standardized categorical data. By refining our dataset, we aimed to retain only the most meaningful features that contributed to median salary prediction while eliminating unnecessary noise.

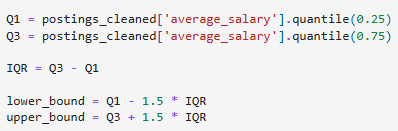
**Dropped columns**: We identified several columns that were not useful for our predictive model such as the unique identifiers, URLs, and redundant fields. Unique identifiers such as ‘job\_id’ and ‘company\_id’ were removed since they provided no predictive value. Salary-related columns like ‘max\_salary,’ ‘med\_salary,’ and ‘min\_salary’ were also dropped to ensure that our model predicts ‘normalized\_salary’ independently. Data-related fields such as ‘original\_listed\_time,’ ‘expiry,’ ‘closed\_time,’ and ‘listed\_time’ were excluded since they had no direct correlation with salary levels. We also removed application-related fields, such as ‘job\_posting\_url,’ ‘application\_url,’ and ‘sponsored’ because they only pertain to job posting’s listing behavior rather than salary determinants. Lastly, non-informative columns like ‘pay\_period,’ ‘work\_type,’ ‘currency,’ ‘fips,’ and ‘remote\_allowed’ were also discarded due to redundancy, excessive missing values, or lack of interpretability.

**Handling missing value**s: With unnecessary columns removed, we then handled missing values to ensure data integrity. The most critical column, ‘normalized\_salary’ which we later renamed to ‘median\_salary’ serves as our target variable. From there, since our dataset consists of 124k entries, we agreed it would be best to simply drop all NaN entries rather than implementing mean imputation to ensure the integrity of our model. Our reasoning being that the salaries are independent from one another due to differences in job title, industry, and skill.



**Figure 1: Proportion of NaN Values in each Column**

**Renaming Categorical Variables:** To improve interpretability in some of the features within our data, we reformatted ‘company\_name’ to ‘company,’ ‘formatted\_work\_type’ to ‘work\_type,’ ‘formatted\_experience\_level’ to ‘experience,’ ‘location’ to ‘state’, and ‘normalized\_salary’ to ‘median\_salary’.



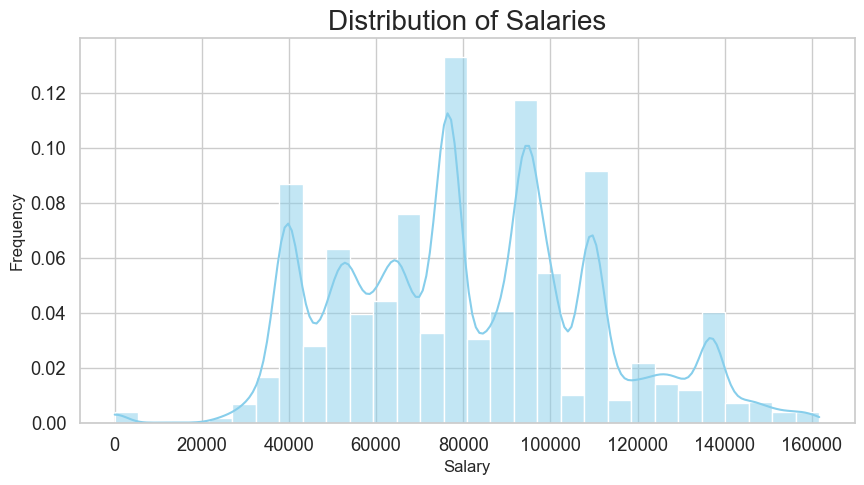
**Figure 2: Interquartile (IQR) Method for Identifying Salary Outliers**

**Removing outliers:** We then removed any outstanding outliers in the dataset. As shown in Figure 2, we set the threshold only to include median salaries that reside within the lower and upper quartile range. This means any salaries falling outside this range were removed to prevent extreme values from skewing the predictive model. As a result we got rid of 933 entries, reducing our postings\_cleaned dataset to 34,183 entries.

**1.4 Findings in EDA**

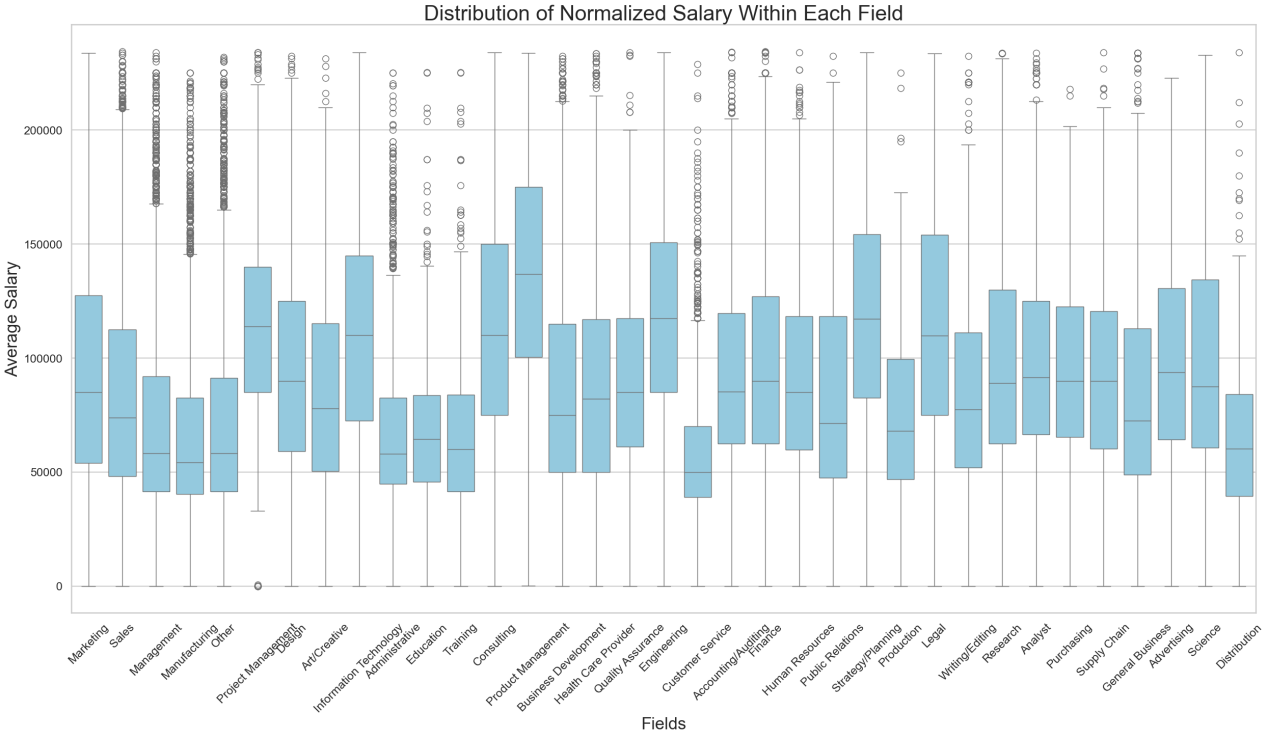
Once we had a cleaned dataset, we conducted an exploratory data analysis to gain sights into salary distributions and examine which job-related factors influence salary variations. We focused on key variables such as salary distribution, salary difference by job field, location-based salary variations, and the impact of work type on earnings.

**Salary Median:** Once we had our cleaned dataset, we first looked at the distribution of median salary across the whole dataset. Figure 3 shows a normal salary distribution which indicates that the mean, median, and mode are similar to one another. The majority of salaries fall between $40,000 to $100,000, with fewer roles exceeding $150,000.

**​​**

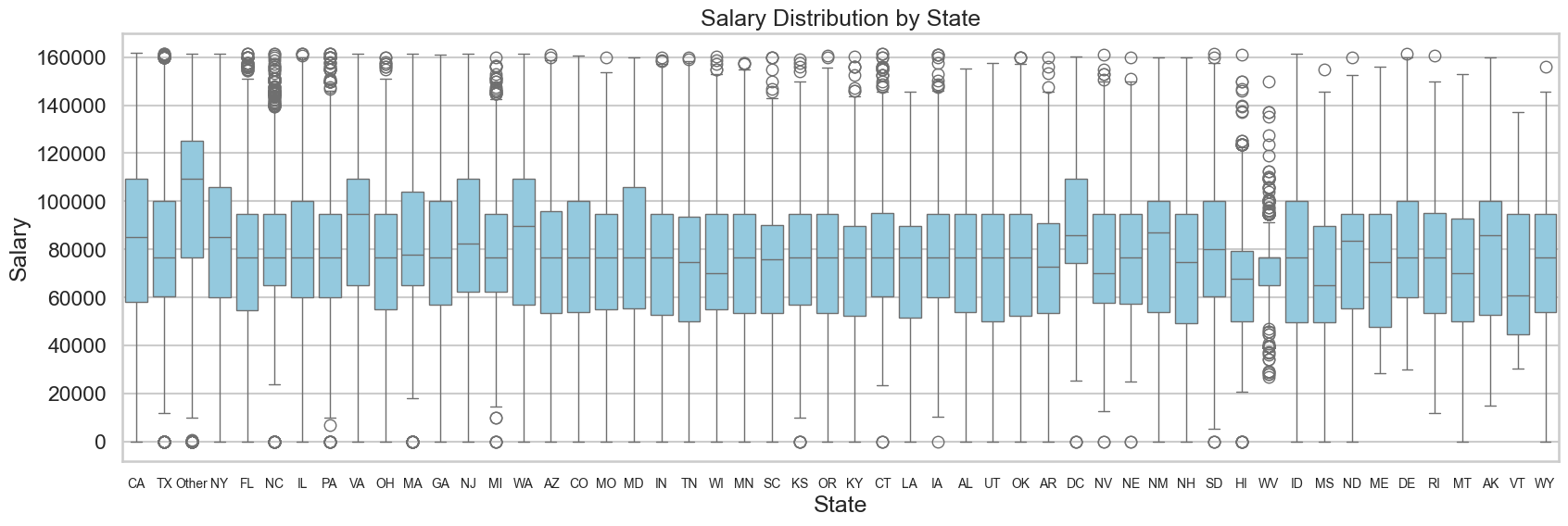
**Figure 3: Distribution of Salary**

**Salary vs Field:** We then delved deeper into the distribution of salaries within each field. As seen in Figure 4, higher-paying roles are concentrated in Product Management, Engineering, Strategy/Planning, and Legal which all tend to require specialized skills and advanced degrees. In contrast fields like Customer Service and Manufacturing tend to offer lower wages. These findings confirm that industry selection plays a crucial role in salary expectations, with technical and strategic offering positions offering significantly higher compensation.

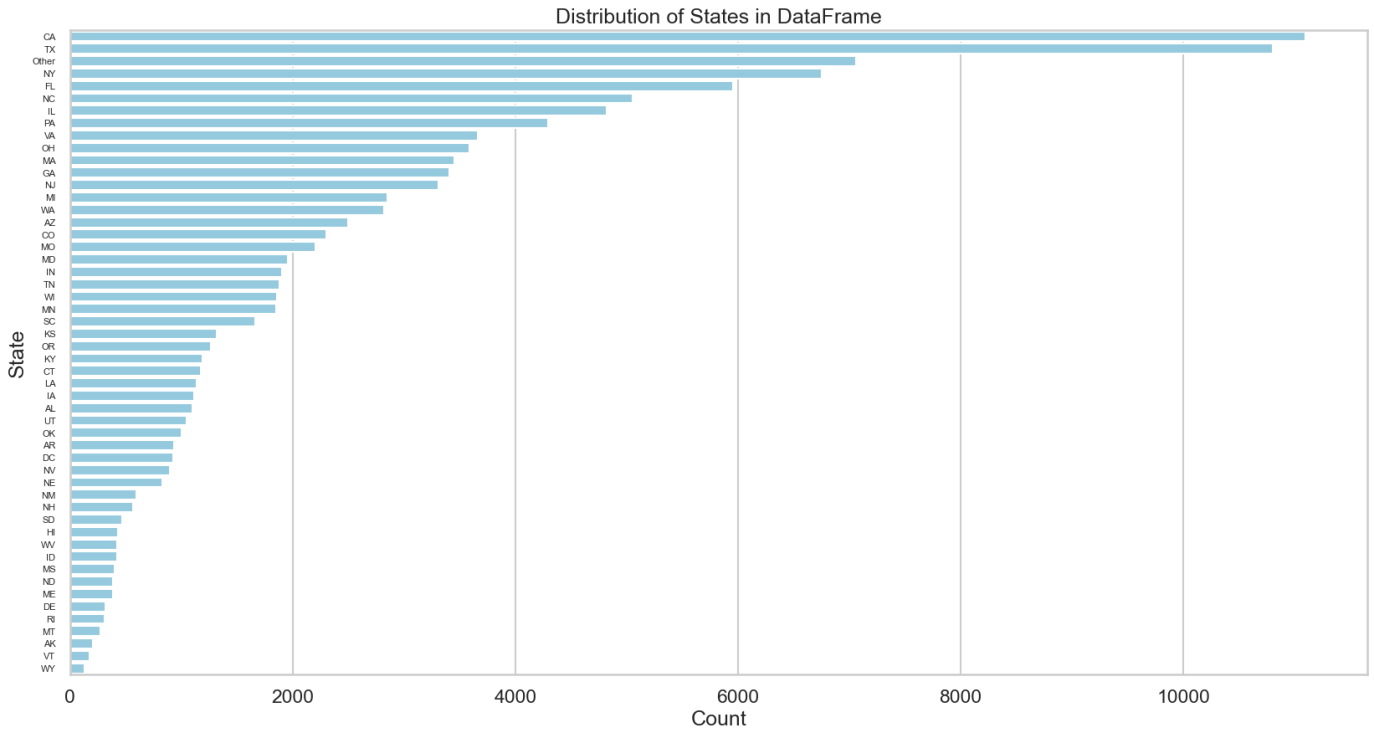
****

**Figure 4: Distribution of Salary within each Field**

**Salary vs State:** In order to obtain the relationship between median salary and state, we extracted the last two characters from the ‘location’ column to create a new column representing the state abbreviation. If no state was listed, we categorized the entry as ‘Other.’ Furthermore, some locations consisted of just the state so we utilized the “us” package which consists of all of the state abbreviations and capitals. We can see in Figure 5 that higher paying jobs reside in Washington, Delaware, and New Jersey and the lower paying jobs are from Mississippi, Idaho, and Vermont. This trend aligns with economic factors such as cost of living, industry presence, and local job market demand.

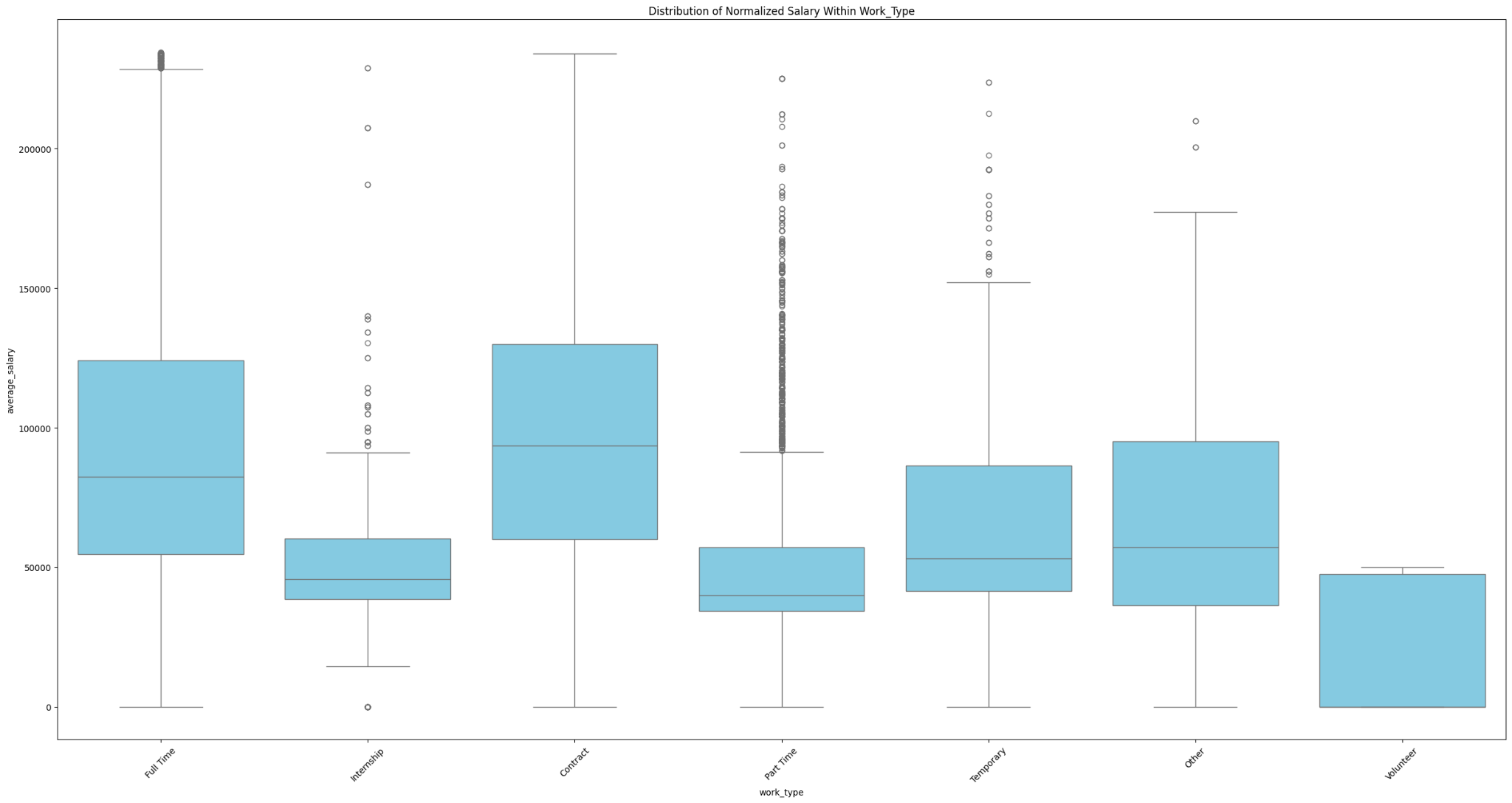
****

**Figure 5: Distribution of Salary within State**

****

**Figure 5.1: Distribution of States**

**Salary vs work\_type:** Work type can often be a deciding factor when choosing a job. Individuals may prefer the flexibility of full-time or part-time, especially if they are planning on simultaneously obtaining their masters. In Figure 6, full-time and contract positions tend to offer higher mean salaries, whereas part-time roles and internships fall within a lower salary range. This trend shows how permanent positions tend to provide greater financial security while temporary roles offer lower but more adaptable earnings, reflecting the broader labor market.

****

**Figure 6: Distribution of Salary within work\_type**

**2 Predictive Task**

**2.1 Predictive Task**

After we completed our exploratory data analysis and identified what features are more closely related to median salary, we moved on to our regression prediction problem: **Predicting median salary based on characteristics of the job listings (state, industry, experience, work type, company, views, application type, and zip code).**

**2.2 Evaluation**

For this project, we decided to use R-squared and mean absolute error (MAE) as our metrics to evaluate the performance of our prediction model. R-Squared, also known as the coefficient of determination, helps us understand model fit and feature importance. Our goal is to achieve a high R-squared value, as it indicates that the model explains a greater proportion of the data's variability. However the R-squared value does not show actual error, which is why it’s important to include mean absolute error. MAE is calculated based on the original unit of measurement which makes it easy for us to determine how accurately our model predicts the median salary of a job listing. Therefore, it is critical that we try to get the smallest value possible to prove model accuracy.

**2.3 Baseline**

For the baseline model, we opted for 2 models: XGBoost and Random Forest Regressor. These are the following features we used: company, state, views, work\_type, application\_type, experience, zip\_code, industry, and field.

**Figure 7: list of baseline features**

XGBoost: Extreme gradient boosting follows a structure where each tree learns from the errors of the previous tree. It uses sequential learning to improve its prediction and is very fast and efficient due to parallel computing and regularization techniques. It does extremely well with large datasets and feature importance ranking. Since our dataset is on the larger side, this model proved the most efficient.

Random Forest Regressor: The structure is an ensemble of decision trees, where multiple trees are built and the average prediction is taken. It utilizes bootstrap aggregating to reduce variance and prevent overfitting. Additionally, each tree gets a random subset of features and data, making it robust and stable. In other words, it handles missing data well, reduces overfitting, and works best with small datasets. Since our dataset is on the bigger side, it did take a while for it to run, but other than that the model performed well.



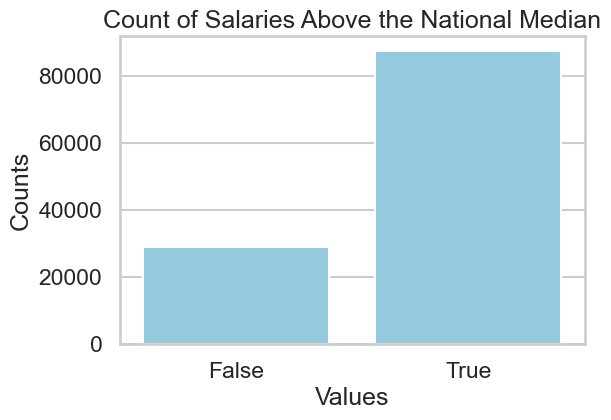
**Figure 8: Baseline model**

**3 Model**

We used a Random Forest Regressor to solve this problem and utilized the same list of features as our baseline model, except we added a column that signifies whether or not the median salary is above or under the national median. To determine if the model performed better, we look if there was an increase to the R-squared value and a decrease in the MAE value.

**3.1 Feature Engineering**

With our baseline model, there were multiple avenues for improvement, but we specifically focused on a way to generalize the “median\_salary” so that it could serve as a bridging feature that can accurately lead us to an accurate prediction of median salary. More specifically, if the given salary of a job tends to be associated with an industry that is above the national median, it already gives the model a lot of information to predict the actual median salary of the job listing.

****

**Figure 8: Distribution of Salaries Above and Below the National Median**

**Transforming Categorical Data**

The original dataset consists of lots of categorical data that is useful in predicting median salary, but first must be converted to numerical data.

Creating ‘State’ Column: The original dataset had a ‘location’ column which would either contain the city and state abbreviation, the country (‘United States’), or just the city name. In order to isolate the state from the location, we loaded a cities\_states.csv which contains all of the cities and states in the United States. We then defined a function to convert a location string to its state abbreviation. That way, no matter if the location is stored as just a city or just a state, we would be able to correctly assign the state the job listing belongs to.

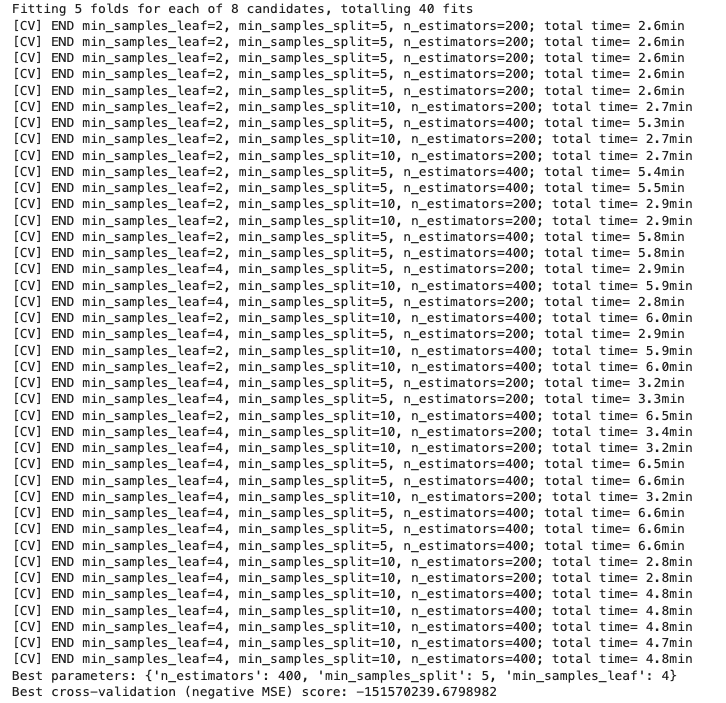
One Hot Encoding: We one hot encoded 'work\_type', 'experience', 'state', and ‘industry’ using the get\_dummies function. We decided to continue with one-hot encoding since none of the categorical variables were ordinal, so the function get\_dummies works perfectly for converting the mentioned categorical variables to numerical ones.

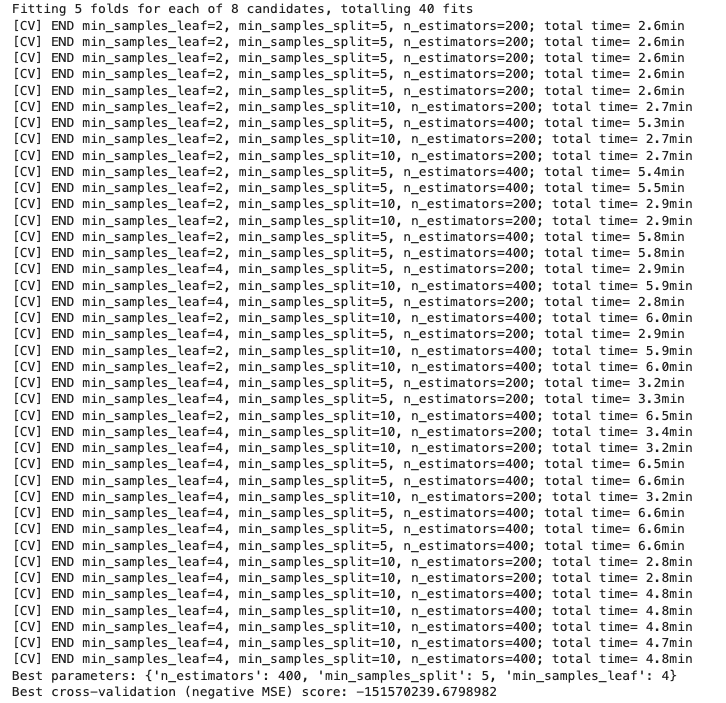
**3.2 Hyper Parameter Tuning**

After the addition of the ‘above\_median’ feature, we underwent hyperparameter tuning. For our baseline, we decided not to identify certain parameters except for setting random\_state = 42. When we moved on to our final model, we set the parameters as follows: n\_estimators: [200,400], min\_samples\_split: [5, 10], min\_samples\_leaf: [2,4].

**3.3 Model Optimization**

The model optimization was completed through our feature engineering and adding in the “above\_median” feature. We started by dropping our predicting variable, ‘median\_salary’ and did a 80-20 split for training and testing. Then, we set the Random Forest Regressor to the random\_state=42 and calculated the MAE and R-squared. We then utilized hyperparameter tuning to improve predictive performance. We set the parameters as follows: n\_estimators: [200,400], min\_samples\_split: [5, 10], min\_samples\_leaf: [2,4]. We also utilized the RandomizedSearchCV function to use cross validation results to determine the best parameters for the model, with n\_iter = 20 and cv=5.





**Figure 8: Cross-Validation and Final model**

**3.4 Final Optimization Result**

Upon analyzing our baseline model, we experimented around with various features we initially exempted to determine if they could increase our R-squared and decrease our MAE. We realized that the current features we used for the baseline had no direct information to determine whether or not a median salary would be one the higher or lower end, so we decided to generalize the ‘median\_salary’ column by creating the ‘above\_median’ feature which stores booleans regarding if the salary is above or below the national median. We stuck with the same model, RandomForestRegressor and saw an increase in R-squared and a decrease in the MAE (median annual salary) as a result of including our feature engineering.

**4 Literature**

This Kaggle dataset has been studied by others before in projects, however, we found that they are more classification-focused and less machine learning/predictive focused.

**"Decoding the Job Market: An In-depth Exploration by Pratul Goyal**

Goyal’s project begins loading multiple datasets and then merging datasets on ‘job\_id’ and ‘company\_id’. They then created various data visualizations on job and company analysis. We both created graphs to visualize top industries to work in, but for different purposes. Goyal created a graph to simply showcase the top 10 industries and job postings associated with them, while we created a box plot to reveal the salary distribution within the head industry/field. Additionally, Goyal went more in-depth in TFIDF and created a 2d projection of job descriptions and skills with clusters as well as explored more relationships concerning job postings (which is a feature we removed early on while cleaning). There was no distinct conclusion for this project, but was mostly a library of visualizations to see what relationship each feature has with one another.

**Text-based Resume Filtering Tool by Thangaraj, Zafar, and Riyahivafa**

This project focuses on building a “talent matching system that accurately matches candidates’ resumes with job descriptions to assess whether a candidate is a good fit for a given role. Their target audience is recruiters, for their project would benefit them by mitigating the process of finding best-fit candidates. They approach the project by identifying two methodologies: natural language processing (NLP) and GPT models. At the end of the project, they aim to provide insight into the effectiveness of GPT compared to more conventional machine learning techniques.

After implementing SVM, TF-IDF, and random forests on their personal model, they then began running tests to see how it compares to GPT. To determine the probability that a resume showcases a good candidate, they compute the similarity between the job description and resume. After testing TF-IDF, Word2Vec, and LDA (topic modeling), they found that LDA performed the best. Upon comparing their personal model to GPT, they found that GPT-2 “lacks a built-in understanding of tasks like resume-job matching. It treats the provided prompt as a generic text generation task, leading to irrelevant or verbose responses.

The major novelty of our work lies in utilizing machine learning algorithms to predict which key features can most accurately predict the median salary in the current job market.

**5 Results**

After conducting extensive exploratory data analysis and data cleaning, we developed machine learning models to predict job salaries based on key features from LinkedIn job postings. This section presents a comparison of different models, evaluates their effectiveness, describes the impact of hyperparameter tuning, and highlights major takeaways from our analysis.

**5.1 Comparison**

To build an effective salary prediction model, we implemented and compared XGBoost and Random Forest Regressor as our baseline models. The initial XGBoost model achieved a Mean Absolute Error of 10,844 and an R^2 score of about 0.64. This indicates a moderate predictive power. However, the Random Forest model on the other hand performed slightly better with an MAE of 9,372 and an R^2 of 0.66. This suggested that our Random Forest model had a stronger ability to capture salary variation.

Above\_median: In an effort to improve

performance, we introduced a new categorical feature, above\_median, which denotes whether a job’s salary is above or below the national median. Incorporating this feature significantly enhanced model accuracy. The RFM ended up achieving a much lower MAE of 6,111 and an improved R^2 of 0.83. This demonstrated how categorical transformations can provide substantial predictive value.

**5.2 Effectiveness**

To assess model effectiveness, we focused on the Mean Absolute Error and R^2 score. With a lower MAE, we would be able to indicate that the model’s salary predictions are actually a lot closer to the real salaries. Likewise, a higher R^2 would suggest that the model better explains the variance in salaries.

Our initial models struggled with extremely high error values, reaching over a billion, indicating that salary prediction is inherently complex. However, after refining features and introducing categorical transformations, we observed a significant drop in MAE and an increase in R^2.

While our final provided strong results, some salary variation remains unexplained, likely due to external factors such as individual negotiation skills, company policies, and broader economic conditions that are not captured in job postings.

**5.3 Hyperparameter**

To optimize model performance, we applied hyperparameter tuning to our Random Forest Regressor using RandomizedSearchCV. The tuning process adjusted key parameters, including:

* Number of Estimators: [200,400]
* Minimum Samples Split: [5, 10]
* Minimum Samples Leaf: [2, 4]

The best-performing model was found with n\_estimators = 400, min\_samples\_split = 5, and min\_samples\_leaf = 4. This tuned model achieved an MAE of 6,028 and an R^2 score of 0.8476 which indicates further improvements in prediction accuracy compared to the untuned model.

**5.4 Major Takeaways**

From our modeling process, we identified several key takeaways:

1. **Feature Engineering Significantly Improves Accuracy** - The introduction of ‘above\_median’ as a feature played a crucial role in reducing MAE and increasing R^2 which demonstrates how categorical insights can enhance regression models.
2. **Random Forest Outperformed XGBoost** - While both models provided reasonable predictions, Random Forest consistently outperformed XGBoost, making it the preferred choice for salary prediction.
3. **Hyperparameter Tuning Matters** - Fine-tuning ‘n\_estimators,’ ‘min\_samples,’ and ‘min\_samples\_leaf’ led to meaningful performance gains, underscoring the importance of parameter optimization in machine learning models.
4. **Salary Prediction Remains a Complex Task** - Even with a strong model, salary predictions are not perfect, as real-world factors like negotiation skills, company benefits, and unstructured job descriptions introduce variability that our dataset cannot fully capture.

This project provided valuable insights into salary prediction trends and the factors influencing earnings in different industries, locations, and job types. Understanding these patterns is especially relevant for those entering the job market, as it offers a data-driven perspective on salary expectations and potential career paths. By applying machine learning techniques to real-world job postings, we were able to improve salary estimation accuracy while also gaining a deeper understanding of how structured data analysis can inform workforce decisions.

**REFERENCES**

**[1]**[**https://www.kaggle.com/datasets/arshkon/linkedin-job-postings**](https://www.kaggle.com/datasets/arshkon/linkedin-job-postings)

**[2]**[**https://www.kaggle.com/code/hasanz9/text-based-resume-filtering-tool**](https://www.kaggle.com/code/hasanz9/text-based-resume-filtering-tool)

**[3]**[**https://www.kaggle.com/code/pratul007/decoding-the-job-market-an-in-depth-exploration**](https://www.kaggle.com/code/pratul007/decoding-the-job-market-an-in-depth-exploration)