Predicting Data Science Salaries Using Glassdoor Data

Felicity Hade
Brown University
GitHub: https://github.com/felicityhade/data_salaries.git

1. Introduction

Data science is a diverse, rapidly evolving field. Considering the variance in data scientists' responsibilities and salaries, it can be difficult to determine industry standards. This is problematic for data scientist job applicants during an offer negotiation; when asked their desired pay, they need to have a reasonable idea of what the company might be willing to offer. The goal of this project was to address this issue through the development of a salary-predicting machine learning model.

The model was trained using the Glassdoor Data Science Job Listings dataset from Kaggle. The dataset contains 1500 rows of U.S. job listings scraped from Glassdoor.com on September, 18th, 2023 [1]. The dataset contains 12 features: job title, salary estimate, job description, company rating, company name, location, company size, year founded, type of company ownership, and industry. The target variable is the average estimated salary for a position. The continuous nature of this variable makes this a regression problem.

To the author's best knowledge, no other model trained on the Glassdoor dataset has had its results published. However, a model has been attempted to solve a similar problem using global data from 2020 to 2023. Note that this dataset contains different features from an alternate source, and the resulting model is not designed to predict U.S. salaries specifically. The model referenced achieved a RMSE of 51,155 USD [2].

2. EDA

The original dataset contained many duplicates, which were dropped before performing the rest of the EDA. Figure 1 illustrates the distribution of the target variable, revealing that the distribution of salaries is skewed right. It also shows outlying values close to zero. Upon closer inspection, these values were found to be associated with student roles and dropped from the dataset.

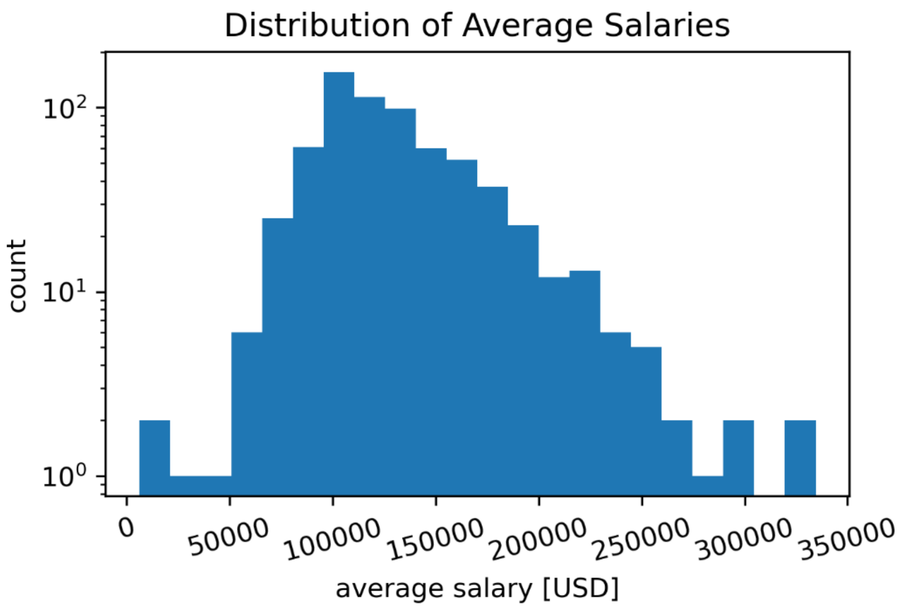

Figure 1: Distribution of the target variable

Every column originally contained missing values, and 33% of unique job listings had missing values. Rows missing salary data were dropped before calculating the fraction of missing values for the remaining features, reducing the final number of rows to 657. Figure 2 shows that missing values remained prevalent across all features, necessitating additional preprocessing.

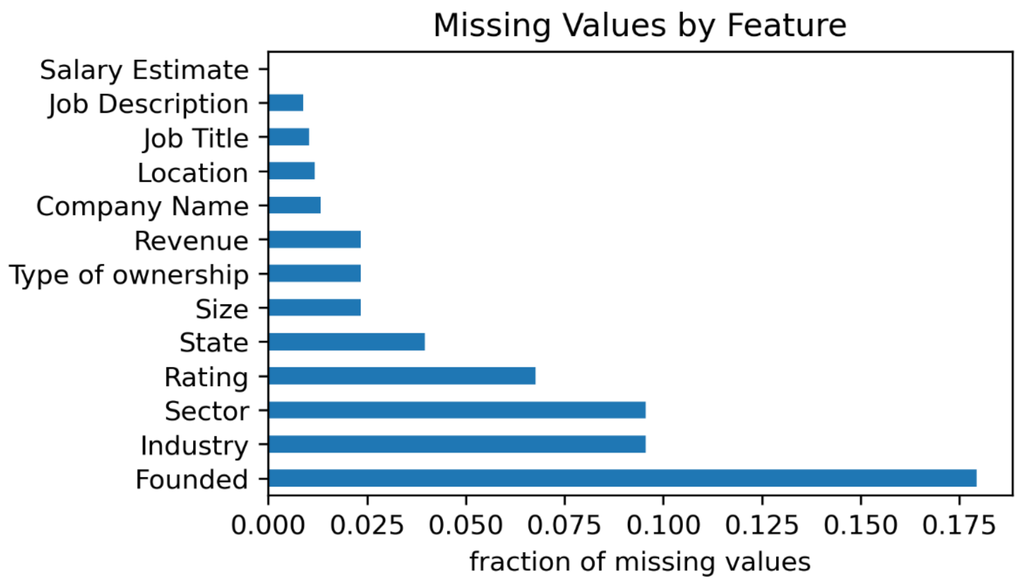

Figure 2: Proportion of missing values by feature

After feature engineering, several plots were created to visualize the correlation between individual features and the target variable. Figure 3 shows that salaries tend to be higher for senior and very senior positions. Therefore, experience level may be an important factor in making salary predictions.

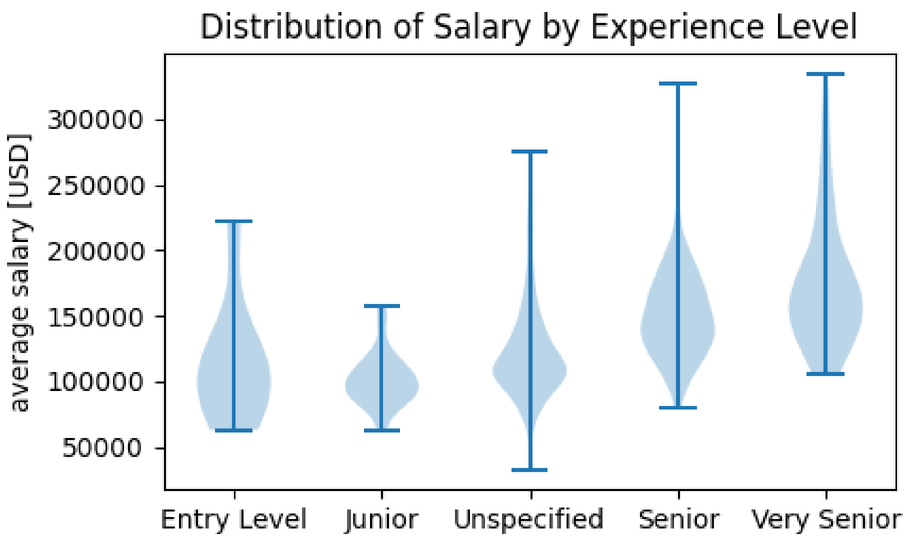

Figure 3: Salary distribution by experience level

EDA also revealed that there are as many as 6 job listings from the same company within the dataset.

3. Methods

3.1. Feature Engineering

The target variable, average salary, was constructed by averaging the upper and lower bounds of each salary estimate range. Hourly wages were converted to annual salaries assuming a 40 hour work week for 52 weeks. “Job Type” and “Experience Level” were generated using keywords found in job titles. Possible job types include Data Engineer, Data Scientist, Data Analyst, and Other. Experience categories are Entry Level, Junior, Unspecified, Senior, and Very Senior. Variables “Python”, “R”, “SQL”, and “AI” were created using keywords in the job description to indicate whether each skill is mentioned. A “State” column was created from the original location column to link jobs from different cities within the same state.

3.2. Splitting

The dataset is not independent and identically distributed because it contains multiple entries from the same company. To ensure the model could predict salaries for unseen companies, group-based splitting was implemented using sklearn’s GroupShuffleSplit. Company names were used as groups. A 60/20/20 split was used to divide the data into train/validation/test sets; because the dataset is small, a substantial proportion of the data was withheld for validation and testing. The dataset was split five times using five different random seeds to measure variability in model performance due to splitting and non-deterministic ML methods.

3.3. Preprocessing

For categorical and ordinal features, missing values were either assigned to existing “Unknown” categories or grouped into their own new category. Categorical features were transformed using sklearn’s OneHotEncoder. Ordinal features like experience level were transformed using an OrdinalEncoder to preserve their inherent ranking. All features were then scaled to have a mean of zero and standard deviation of 1 using a StandardScaler; this is a requirement for some of the feature importance metrics generated after training. Each scaler and encoder was fitted to the train data separately for each random seed. The pattern submodel approach was then used to handle missing numerical values. This approach was chosen over multivariate imputation because datapoints with the same missing value pattern may exhibit similar salary trends.

3.4. Models

The machine learning algorithms used and the parameters tuned for each are summarized in Table 1.

Algorithm	**Parameter Grid**
Elastic Net	alpha: [1e-3, 1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3] l1_ratio: [0.0, 0.25, 0.5, 0.75, 1.0]
KNN	n_neighbors: integers 1-20 weights: [‘uniform’, ‘distance’]
Random Forest	max_depth: [1, 3, 10, 30, None] max_features: [0.25, 0.5, 0.75, 1.0]
XGBoost	max_depth: [1, 2, 3, 10, 30, None] learning_rate: [0.01, 0.05, 0.1, 0.5, 1.0]

Table 1: Parameters tuned for each ML algorithm

XGBoost’s ‘colsample_bytree’ and ‘subsample’ were changed from sklearn’s default values to 0.9 and 0.66 respectively to prevent overfitting. XGBoost models were trained using 50 early stopping rounds to automatically determine the optimal number of trees. XGBoost’s ‘reg_alpha’ and ‘reg_lambda’ parameters were omitted from tuning due to runtime. Two versions of the XGBoost algorithm were implemented: one which was used in combination with the pattern submodel approach, and one which was trained with missing numerical values.

Parameters were tuned using a grid search strategy, implemented manually to incorporate early stopping. The optimal parameters were selected to minimize RMSE on the cross-validation set. Each random seed required three to four different models to be optimized for each machine learning algorithm, corresponding to the number of missing value patterns in the test set. This excludes the XGBoost model trained using missing values, for which only one model was optimized per random seed.

4. Results

4.1. Performance

The mean and standard deviation of each model’s test RMSE was computed across the five random states. Baseline performance was calculated by simulating a model which consistently predicts the mean test salary. Figure 4 compares the models’ results.

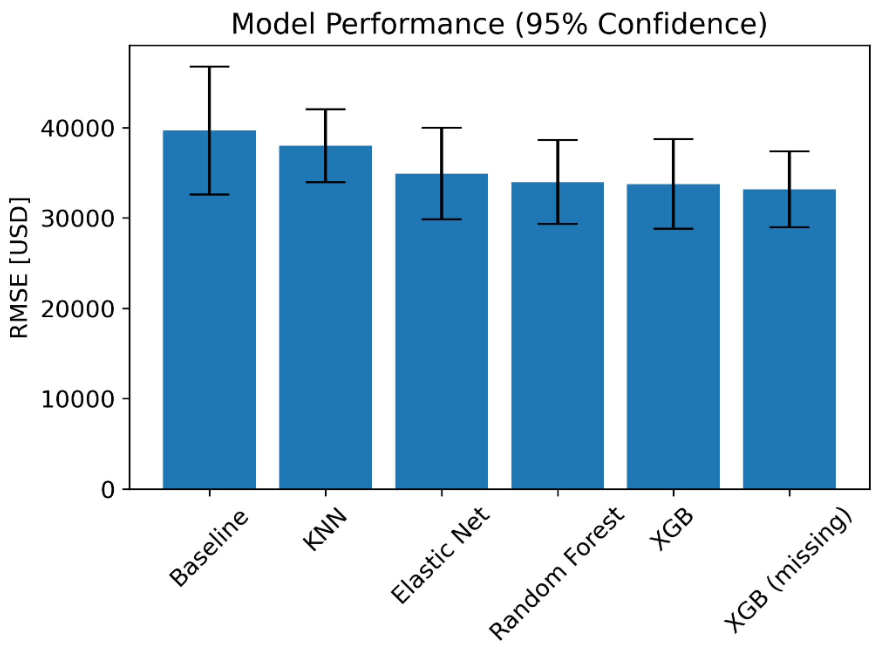

Figure 4: Models' test performance across 5 random states

Model	Average RMSE	Standard Deviation	Upper Bound (95% Confidence)	Standard Deviations from Baseline
Baseline	39,679	3,546	46,772	0
Elastic Net	34,922	2,524	42,014	-1.34
KNN	37,996	2,013	45,089	-0.47
Random Forest	33,958	2,323	41,051	-1.61
XGBoost (reduced features)	33,761	2,481	40,854	-1.67
XGBoost (with missing values)	33,181	2,095	40,273	-1.83

Table 2: Measurements of test performance by model

All 5 models achieved RMSE and standard deviation values below the baseline, indicating superior accuracy and precision. However, no model was able to achieve an average RMSE more than two standard deviations below the baseline.

The best overall model was the XGBoost model trained with missing numerical values (i.e., without using the pattern submodel approach). This model achieved an average test RMSE of $33,181, an improvement of $6,499 from the baseline. This model not only achieved the lowest average RMSE, but also yielded relatively consistent results. The upper bound on the 95% confidence interval for its test RMSE was the lowest of all five models. The model's test predictions are visualized in Figure 5. The gray line represents the baseline prediction.

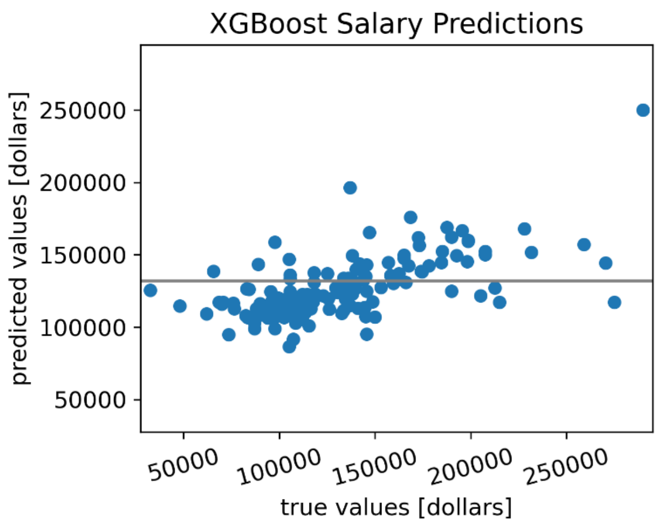

Figure 5: Test predictions made by the winning XGBoost model

There is a visible positive correlation between the model's predictions and the true values. Many predictions fall into a cluster centered around $120,000.

4.2. Feature Importance

Three metrics of global features importance were calculated for the winning model: gain, permutation importance, and SHAP importance. Figure 6 shows the top seven most important features using gain. For tree-based methods like XGBoost, gain measures the improvement in train performance which results from splitting on a particular feature. Compared to XGB's coverage and weight metrics, gain is seen as the most relevant for calculating relative feature importance [3].

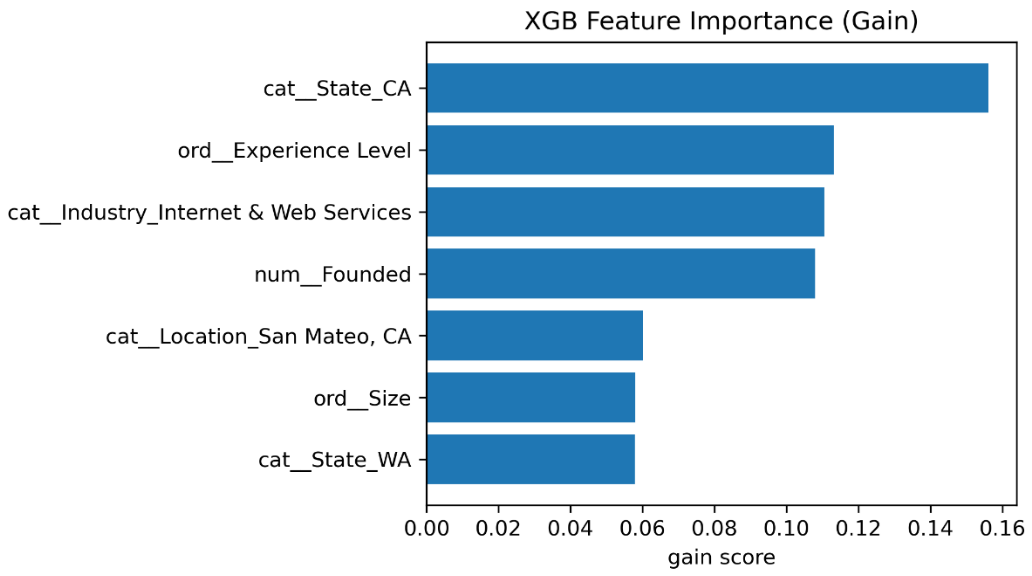

Figure 6: Top seven most important features using gain

Using gain, the most important contributor to the model's salary prediction is whether the job is located in is California. The experience level, whether the industry is Internet & Web Services, and the year the company was founded are also important.

Permutation importances were calculated by shuffling individual features and measuring the resulting impact on test performance across 10 iterations. Figure 7 shows the results.

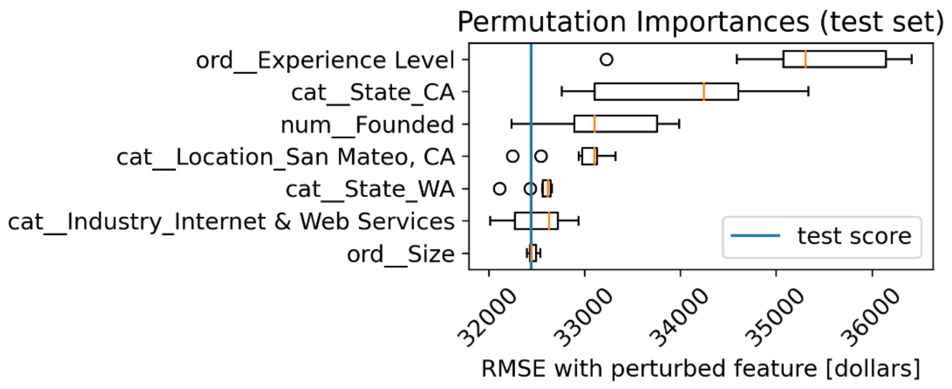

Figure 7: Top seven most important features using permutation importance

In contrast to the gain metric, permutation importance ranks experience level as the most important feature. There appear to be few features of substantial importance to the XGBoost model, with the seventh-most important feature having little impact on RMSE after shuffling.

Unlike permutation importance, SHAP scores account for feature interactions. Figure 8 shows global SHAP importance scores for the winning model.

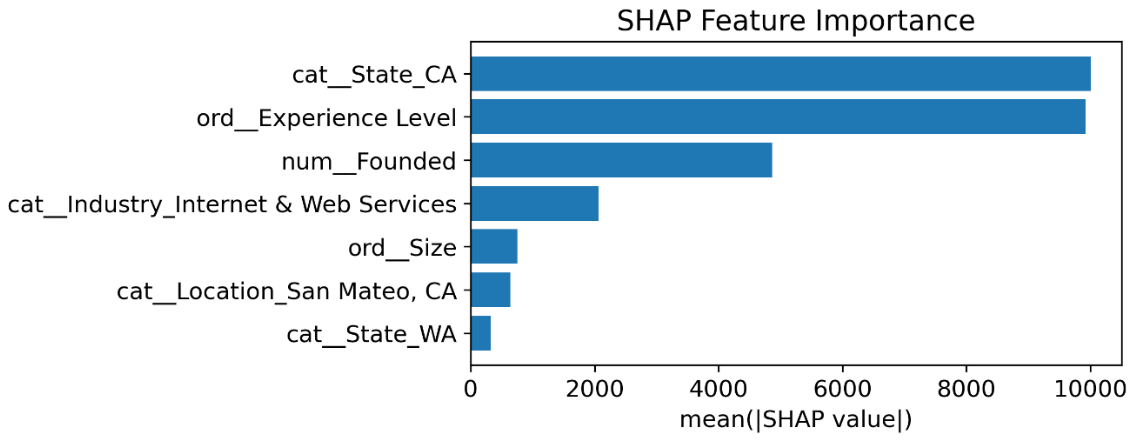

Figure 8: Top seven most important features using SHAP importance

Like gain, the SHAP method ranks whether a job is located in California as the most important feature in determining its predicted salary. Although the order differs, all three measures of global feature importance rank the experience level, whether the state is California, and the year the company was founded within the top five most important features. The least important features include those that are unique to a single datapoint in the test set. For example, the location Lititz, PA has zero importance when the sole job listing belonging to this location is sorted into the test set.

Local feature importance was analyzed in addition to global importance using SHAP scores. Figure 9 depicts the most significant contributors to the salary prediction for a particular datapoint.

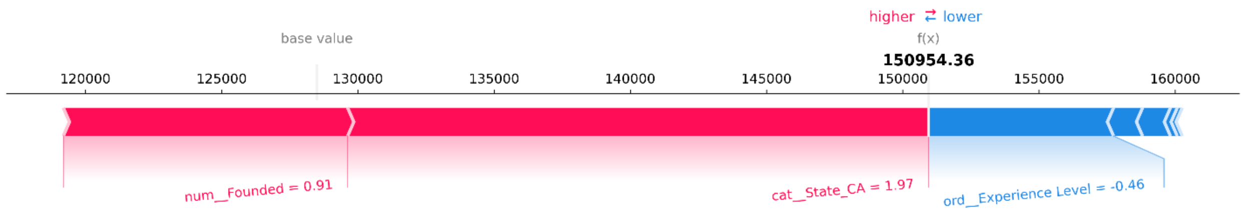

Figure 9: The winning XGBoost model's prediction for datapoint 1 of the test set

The year the company was founded, which is more recent than average, and the fact that the role is located in California contribute positively to the model's prediction relative to the base value. The below-average experience level contributes negatively. These three features have the greatest influence on the model's prediction for this particular job listing. As seen earlier, they also rank among the most important features globally. Figure 10 depicts the Elastic Net model's prediction for the same datapoint.

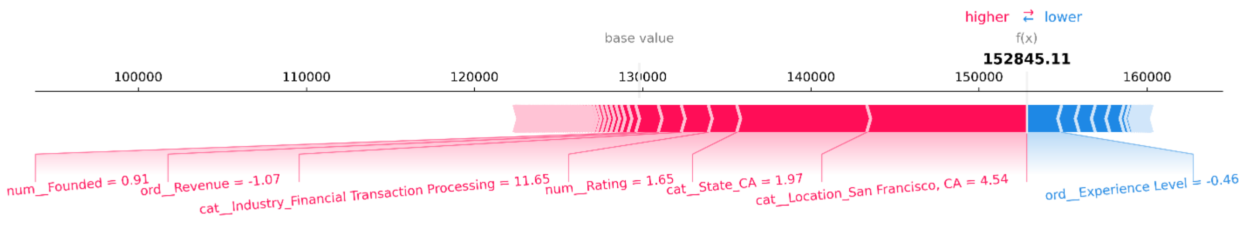

Figure 10: Elastic Net prediction for datapoint 1 of the test set

The Elastic Net's prediction for the same datapoint is influenced by a wider variety of features. It is possible that XGBoost's superior performance results from using fewer features, making it less susceptible to overfitting.

5. Outlook

Several strategies could be used to improve model performance. Job titles and descriptions likely hold a great deal of additional predictive power which is not leveraged by the current feature engineering process. Natural language processing techniques like topic modelling could be used to group similar descriptions instead of using manual keyword identification. Multivariate imputation may also boost performance compared to the pattern submodel approach for non-XGB models. Some missing value patterns contain few data points, limiting the training data available to the associated submodel.

Ultimately, the model's performance is hindered by the quality of data it is trained on. The number of usable datapoints in this dataset is small. Scraping more data or using a different, larger dataset would be necessary to train a highly accurate model.

6. References

[1] RRK-CODER. (2023, September). Glassdoor Data Science Job Listings, Version 1. Retrieved October, 2023 from https://www.kaggle.com/datasets/rrkcoder/glassdoor-data-science-job-listings/data.

[2] Fernando Torres, Lus. "Data Science Salaries 2023 EDA & Prediction." Kaggle, July 2023, https://www.kaggle.com/code/lusfernandotorres/data-science-salaries-2023-eda-prediction#predictions.

[3] Abu-Rmileh, Amjad. "The Multiple faces of 'Feature importance' in XGBoost." Towards Data Science, Medium, 8 Feb 2019, https://towardsdatascience.com/be-careful-when-interpreting-your-features-importance-in-xgboost-6e16132588e7.