Salary Prediction Model

Alex Ojemann	Tyler Kimbell	Adrian Reghitto
Department of Computer Science	Department of Computer Science	Department of Computer Science
University of Colorado Boulder	University of Colorado Boulder	University of Colorado Boulder
Boulder CO, USA	Boulder CO, USA	Boulder CO, USA
Alexander.Ojemann@colorado.edu	Tyler.Kimbell@colorado.edu	Adrian.Reghitto@colorado.edu

Abstract:

When thinking about applying for a job or starting a career, it is important to understand how different positions, skill sets, and geographic location will affect your salary. With many different job postings it would be beneficial to give an estimate of an average salary given some rudimentary data. Stakeholders include students looking at majors they want to go into, people planning out careers, and people looking for a job. Our goals in this paper are to generate the best possible model for predicting salaries for data science related positions and to dissect the features that are most predictive of salary.

Introduction:

Understanding salary determinants in the job market is crucial for individuals planning their careers, especially in data science-related fields. This research paper leverages a dataset comprising 742 entries of data science and related jobs from Glassdoor to predict salaries. Building on the work of a previous study, *Salary Prediction Based on the Dual-Adaboosting System*, we explore various techniques for predicting salary.

One challenge involves the curse of dimensionality due to categorical features with numerous distinct values. We address this by employing LASSO regression, Principal Component Analysis (PCA), and other feature engineering techniques. Hyperparameter optimization is another challenge, mitigated through randomized search.

We tested AdaBoost, random forest, and xgBoost models on multiple sets of features we generated from the original data and used root mean squared error (RMSE) and mean absolute error (MAE). We compared our results with the benchmark provided by the previous study.

Our findings reveal that manually selected features, particularly when incorporated into an AdaBoost model, outperform other techniques and models. They also reveal critical insights about what factors are most associated with salary.

Related Work:

Our data set contains 742 entries of data science and related jobs from glassdoor. One publication, *Salary Prediction Based on the Dual-Adaboosting System*, explores the same data set explores a number of techniques including Ridge and Lasso regression, random forests, support vector regression, gradient boosting, and adaboosting, along with a dual adaboosting system of their design[1]. The authors use min-max scaling to preprocess their numeric features and transform all categorical features into binary dummy features for each category using one hot encoding. They

[1] "Salary Prediction Based on the Dual-Adaboosting System | Semantic Scholar."

use adaboosting feature importance in order to limit the excessive dummy features, filtering out any with a feature importance of less than 0.001. They ultimately evaluate the root mean squared error on their training set, the root mean squared error on their test set, and the mean absolute error on their test set for each of their models. The adaboost and dual adaboost models perform the best but exhibit a high degree of overfitting given the large split between their training root mean squared error and testing root mean squared error. Their accuracies can be used as a reference point for the accuracies of our model because they use the same data set.

One of the challenges of our data set is that the categorical features have many distinct values. For example, one of the features in our data set is the job title, which has 264 unique values. If we were to one hot encode this feature, we would get 263 binary dummy features. This subjects us to the curse of dimensionality, which is the effect that large numbers of features leave you more vulnerable to finding trends in your data that are not generalizable[2]. Two frequently used tactics for addressing this are Ridge[3] and Lasso[4] regression, which penalize feature weights in a regression model to limit the number of features for which the optimal model will have a nonzero coefficient. While these methods maintain the interpretability of the model, they may limit accuracy. Principal Component Analysis, or PCA, compares favorably to other methods for encoding features in terms of maintaining accuracy in some applications with high dimensional categorical data[5]. However, this method sacrifices interpretability as PCA creates features using orthogonal linear components of all the features that explain the most variation possible.

One of the other challenges that occurs when creating a model to predict salaries is hyperparameter optimization. Many machine learning models require hyperparameters that are computationally expensive to estimate with grid search and can lead to a suboptimal model if manual trial and error does not arrive at an optimal

---

[2] Debie and Shafi, “Implications of the Curse of Dimensionality for Supervised Learning Classifier Systems.”
[3] Marquardt and Snee, “Ridge Regression in Practice.”
[4] Tibshirani, “Regression Shrinkage and Selection via the Lasso.”
[5] Farkhari et al., “New PCA-Based Category Encoder for Cybersecurity and Processing Data in IoT Devices.”

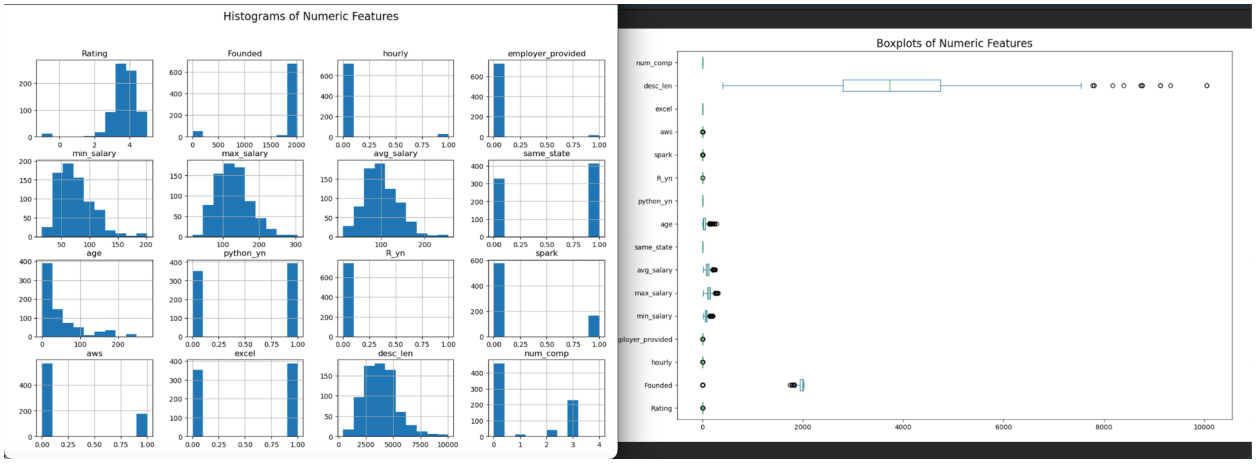
value. Randomized parameter optimization avoids the computational expense of grid search but is less prone to missing optimal combinations than manual exploration[6].

Methodology:

Preprocessing

The first step in preprocessing was to fill NA values. These were filled with the mean of the given feature for numeric features or the mode of the given feature for categorical features.

Next, the numeric features were analyzed for outliers using a histogram and boxplot. These plots are shown below.



Some redundant columns were discovered, such as “Founded” and “age”, where age is just the number of years since the founding year as of 2020, so the “Founded” column was removed. In addition, there are multiple columns for salary, our response variable, one of which is text and three of which are numeric representing the minimum, maximum, and mean. The text based column corresponds to the numeric columns in all cases except for those in which the text based one specifies that the salary is hourly rather than yearly, in which the numeric columns appear to be scaled to yearly assuming 2000 hours worked per year. We decided to use the mean salary as our

[6] Bergstra and Bengio, “Random Search for Hyper-Parameter Optimization.”

response so the text based salary, the minimum numeric salary and the maximum numeric salary were removed.

The “age” and “rating” features contained values of -1, which did not make sense in context and caused a right skew when scaled in the next step. These features were filled with the mean of the given feature value similar to the na values so this was no longer the case. Some other features had additional outliers, but none that did not make sense in context so they were not removed.

Finally, min-max scaling was performed on all numeric features. Min-max scaling was chosen instead of standardization because many of the numeric features are not normally distributed as shown in the histograms above.

Feature Engineering

Our data set had many categorical features that each had hundreds of unique values. If we were to generate binary dummy features for all of their values, we could suffer from the curse of dimensionality. This method would be especially problematic in this case because the resulting data set would have more features than rows, thus it would be rank deficient. For the categorical features we took three different approaches to encoding them and compared the results.

The first approach was to use Principal Component Analysis, or PCA, to capture as much of the variability in the original feature set as possible using a certain number of principal components. This approach involved generating binary dummy features for all categorical features and generating principal components using these dummy features. We had originally intended to use the numeric features for PCA as well, but this resulted in an extremely large proportion of the variability explained by the first principal component. We ultimately decided to generate 20 principal components so that each of these principal components would explain at least one percent of the variability of the dummy features.

Another approach was to manually engineer these categorical features into a manageable number of numeric features, some of them binary, representing characteristics of the original feature. This approach involved a number of different

feature transformations tailored to what each feature is intended to capture. The “revenue” and “size” features were ordinal categorical features with only a few unique values, so we concluded that the best way to capture them numerically was to use one numeric feature rather than many dummy features because it maintains their order. The revenue feature was encoded into values between one and five while the maximum integer value was extracted from the size feature and a logarithmic transformation was applied to avoid a heavily skewed distribution. For the “Location” and "Headquarters” features, we used methods from Python’s geopy package to obtain the latitude and longitude for each of these features and merged the data frame with U.S. Census data to obtain the population of the city for both the job location and headquarters. A logarithmic transformation was applied to the new population features to avoid a heavily skewed distribution. Finally, there were three features, “job_simp”, “Type of ownership”, and “Sector”, for which it made sense to generate dummy features because they had the least number of unique values among features with similar information. For example, “job_simp”, the simplified job title, had far fewer unique values than “job_title” but contained the most significant part of the job title, such as “data scientist” or “data analyst”. The remaining categorical features that were not converted to numeric features in any way and did not have a similar feature converted already had some numeric representation of themselves in the original features, such as the length of the job description and the number of competitors of the company. Some of the new numeric features had NA values induced in the process of conversion. These values were imputed using a KNN imputer from Python’s scikitlearn package. This was done as opposed to using the mean or the mode as before to avoid heavy class imbalance as many of the new features had a small number of distinct values. All new numeric features were min-max scaled in accordance with the others.

The final approach was to use a LASSO regression model to select features. This involved generating binary dummy features for all categorical features, feeding these into a LASSO regression model, and keeping only the features that had a nonzero coefficient for future models. The regularization strength parameter was modified until the number of features with nonzero coefficients in the model was similar to the number of features in the two previously described feature sets. This approach

could still be subject to the curse of dimensionality in that one of the many dummy features could appear to be significant in the training data so the model includes it but it may not generalize. For example, in *Salary Prediction Based on the Dual-Adaboosting System*, the top feature in ADABoost feature importance was “GRM Actuarial”, a dummy feature only found in three entries.

Models

We put each of the three sets of features described in the previous section into a random forest, an xgBoost model, and an AdaBoost model, all applied using Python’s scikitlearn package, resulting in nine models total. Each of these models are considered ideal for a relatively small dataset and allow for some degree of interpretability using feature importance scores.

We used randomized search to optimize hyperparameters. It is superior to grid search for our purposes because it took a non-trivial amount of time to train each model even when using parallelization. We ran 100 iterations of randomized search for each of the nine models. All of the models had an n_estimators hyperparameter that was optimized in the distribution of integers between 0 and 500 and the AdaBoost models also had a learning rate hyperparameter that was optimized in the continuous distribution of all real numbers between 0 and 1.

To evaluate each of our models, we plan to use root mean squared error, or RMSE, and mean absolute error, or MAE, each of which are defined below.

Root Mean Squared Error = (sum(y - y_pred)/n)^½

Mean Absolute Error = (sum(abs(y - y_pred))/n)

These evaluation metrics were chosen because they allow for direct comparison to the models in the *Salary Prediction Based on the Dual-Adaboosting System* publication. We evaluated each of these metrics on each model using 10-fold cross validation to avoid the overfitting that appeared in some of the models from *Salary Prediction Based on the Dual-Adaboosting System*. We then reported the RMSE and

MAE from cross validation along with the RMSE and MAE on a held out test set containing 20% of the original data set for each of the models and compared their performance. In addition, we analyzed feature importance values for the models trained on the manually engineered and LASSO selected features because these feature sets allow for interpretability in contrast to PCA.

Results and Analysis:

Below is how each of our models performed as described in the previous section along with their hyperparameter values:

PCA Selected Features:

Model	Number of Estimators	Learning Rate	Mean Cross-Validation RMSE	Mean Cross-Validation MAE	RMSE on test set	MAE on test set
Random Forest	92	N/A	20.095	12.595	20.077	12.710
xgBoost	72	N/A	19.217	8.784	19.320	9.259
Adaboost	44	0.804	18.219	7.851	19.825	9.107

Manually Selected Features

Model	Number of	Learning Rate	Mean Cross-Vali	Mean Cross-Vali	RMSE on test set	MAE on test set

	Estimators		dation RMSE	dation MAE		
Random Forest	202	N/A	19.035	11.955	17.397	11.076
xgBoost	88	N/A	18.280	9.264	17.277	9.197
Adaboost	131	0.972	16.954	7.479	16.184	6.909

LASSO Selected Features:

Model	Number of Estimators	Learning Rate	Mean Cross-Validation RMSE	Mean Cross-Validation MAE	RMSE on test set	MAE on test set
Random Forest	54	N/A	19.309	12.068	19.673	12.157
xgBoost	175	N/A	18.381	8.929	20.896	9.938
Adaboost	346	0.711	17.645	8.640	18.621	9.137

Overall, the AdaBoost models performed the best in nearly all of our evaluation metrics. None of the models exhibit nearly the same degree of overfitting that they did in *Salary Prediction Based on the Dual-Adaboosting System*, likely because we used cross validation to train them. In addition, each of the model types had its best performance on the manually engineered features in nearly all of our evaluation metrics, including all of the metrics on the test set. This suggests that the manual feature

transformations we performed were effective in capturing or even expanding the predictive value in the original categorical columns while preserving interpretability and generalizing well to new data. The best performing model in *Salary Prediction Based on the Dual-Adaboosting System* had an RMSE on the test set of 17.57 and an MAE on the test set of 8.36 while our best performing model had an RMSE on the test set of 16.184 and an MAE on the test set of 6.909.

The following table shows the feature importances for each model on the manually selected features and LASSO selected features.

Model	Random Forest on Manual Features	xgBoost on Manual Features	AdaBoost on Manual Features	Random Forest on LASSO Features	xgBoost on LASSO Features	AdaBoost on LASSO Features
Feature 1	job_title_analyst: 0.1297	hourly: 0.2804	Longitude: 0.0827	is_senior: 0.1270	Industry_Health Care Services & Hospitals: 0.1292	Rating: 0.0967
Feature 2	hourly: 0.1263	job_title_analyst: 0.1121	is_senior: 0.0779	job_simp_analyst: 0.1167	Headquarters_Mountain View, CA: 0.1227	python_yn: 0.0802
Feature 3	is_senior: 0.1077	job_title_director: 0.0940	job_title_analyst: 0.0702	Rating: 0.1024	Job Title_Data Science	job_state_CA: 0.0693

					Manager: 0.0775	
Feature 4	HQ_Longitude: 0.0675	Sector_Oil, Gas, Energy & Utilities: 0.0642	hourly: 0.0673	job_state_CA: 0.0775	job_simp_analyst: 0.0735	is_senior: 0.0680
Feature 5	Longitude: 0.0637	Ownership_Nonprofit Organization: 0.0568	desc_len: 0.0656	python_yn: 0.0597	job_simp_director: 0.0655	job_simp_director: 0.0605
Feature 6	job_title_director: 0.0496	employer_provided: 0.0563	job_title_director: 0.0600	job_simp_director: 0.0431	job_state_CA: 0.0400	job_simp_analyst: 0.0453
Feature 7	desc_len: 0.0492	is_senior: 0.0545	Rating: 0.0582	Sector_Health Care: 0.0310	Company Name_Takeda Pharmaceuticals 3.7: 0.0331	Job Title_Data Science Manager: 0.0399
Feature 8	Log_Population: 0.0462	python_yn: 0.0445	HQ_Longitude: 0.0580	Industry_Health Care Services &	is_senior: 0.0306	Location_San Francisco, CA: 0.0296

				Hospitals: 0.0248		
Feature 9	Rating: 0.0445	Sector_Health Care: 0.0322	age: 0.0507	Job Title_Director II, Data Science - GRM Actuarial: 0.0217	Job Title_Director II, Data Science - GRM Actuarial: 0.0258	Revenue_$5 to $10 million (USD): 0.0212
Feature 10	age: 0.0383	Sector_Finance: 0.0239	Log_Population: 0.0477	Industry_Enterprise Software & Network Solutions: 0.0209	Size_Unknown: 0.0244	Competitors_-1: 0.0210

These feature importances offer us many valuable insights into what factors are most related to the average salary of a job. One thing that stands out is that “python-yn”, which represents whether an applicant has Python experience, appeared multiple times in the top 10 feature importances for these models but none of the other skills, which include R, AWS, Spark and Excel, appeared in the top 10. In addition, senior positions differ heavily in salary from non senior positions as evidenced by the frequent appearances of the is_senior binary feature towards the top of this list, which was expected. As for location, the manual features do not offer a ton of interpretability here, but we can see that longitude is more important than latitude for both the job location and the headquarters in general and the city population is somewhat of a factor as it appears towards the bottom of the list in two of the three models trained on the manually selected features. In the models trained on the LASSO features we can see

that California and more specifically San Francisco are the most prevalent locations on the list, perhaps unsurprisingly given the high number of tech companies in the Bay Area and in the state in general. We can see that “director” is by far the most significant simplified job title, as it appears in the top 6 most important features for all of the models. Based on the frequency of the dummy features for the industry and sector features in this table, it appears that the Healthcare industry differs the most from others. Hourly positions differ greatly from non hourly positions, perhaps because hourly positions tend to be lower level, as the binary feature for whether a job is paid by the hour is frequently towards the top of this list. Finally, we can see that the average review rating on Glassdoor is also highly correlated with the salary as it also appears multiple times towards the top of the list.

Conclusions:

In this work, we test a number of techniques for predicting salary based on a relatively small data set and compare their effectiveness. We find that an AdaBoost model trained on features that we engineered manually by tailoring the transformation of each feature to our beliefs about the significant aspects about that feature performs the best and surpasses the performance of models trained on the same data from another paper. The interpretability of the models we chose also offers critical insights into what predictors are most correlated with salary through feature importances that could help aspiring professionals looking to work with data.

Future Work:

The dataset was comparatively small, so in the future it makes sense to analyze a larger dataset using a more powerful machine. This would allow for the application of more powerful models such as neural networks. It would also be useful to obtain data from other regions of the world as our data set is limited to positions in the United

States. In addition, taking into account additional variables such as whether the job is remote or not and how substantial the benefits are could give our models more predictive power.

References :

Bergstra, James, and Yoshua Bengio. “Random Search for Hyper-Parameter Optimization,” n.d.

Debie, Essam, and Kamran Shafi. “Implications of the Curse of Dimensionality for Supervised Learning Classifier Systems: Theoretical and Empirical Analyses.” *Pattern Analysis and Applications* 22 (May 1, 2019). https://doi.org/10.1007/s10044-017-0649-0.

Farkhari, Hamed, Joseanne Viana, Luis Miguel Campos, Pedro Sebastiao, and Luis Bernardo. “New PCA-Based Category Encoder for Cybersecurity and Processing Data in IoT Devices.” arXiv, May 23, 2022. http://arxiv.org/abs/2111.14839.

Marquardt, Donald W., and Ronald D. Snee. “Ridge Regression in Practice.” *The American Statistician* 29, no. 1 (1975): 3–20. https://doi.org/10.2307/2683673.

“Salary Prediction Based on the Dual-Adaboosting System | Semantic Scholar.” Accessed September 26, 2023. https://www.semanticscholar.org/paper/Salary-Prediction-Based-on-the-Dual-Adaboosting-Chen-Zhan/641c323bce27ea7f1791f176991a3c10df5d0f70.

Tibshirani, Robert. “Regression Shrinkage and Selection via the Lasso.” *Journal of the Royal Statistical Society. Series B (Methodological)* 58, no. 1 (1996): 267–88.