# Predicting Salary Estimates in Data Science Job Market

(December 2024)

Carlos A. Lopez, RED-ID: 827117558

***Abstract*—This report details the measures and findings of this group project to accurately predict estimated salaries in the data science market by using different regression models trained on the *Data Science Jobs and Salaries 2024* dataset created by Fahad Rehman. This dataset's data is sourced from the Glassdoor job posting and rating website and includes information such as job title, estimated salary range, job description, and company rating. To prepare the data, we clean multiple data columns and extract relevant information. Once the dataset is prepared, we visualize the data by looking at correlations of numerical columns and by plotting the distribution of different categorical columns. We use this exploratory data analysis to choose specific columns from our cleaned dataset to use as features for training our models. Our models used are a Multiple Linear Regression and a more complicated Random Forest Regression model. We evaluate both models appropriately and adjust our data in order to improve these evaluations. We find that a Random Forest Regression has better evaluations than a Linear Regression for our data by comparing the mean squared error, the $R^2$ value, and the mean absolute error. We show that decreasing the number of features in our data led to improved accuracy by dropping columns in our dataset and training a second Random Forest Regression model.**

## I. INTRODUCTION

THE capability to analyze and visualize data related to the job market is important to forecast future trends, identify emerging sectors, predict job demand for specific skills, and anticipate future job opportunities. These insights into the job market are sought after by potential workers, current employees, and employers alike. The ability to make such analyses and visualizations on the job market from predictions is powered by machine learning models and accurately sourced, relevant data.

This project is collaboration between computer science students at San Diego State University for a course in Machine Learning, CS-549. In choosing a topic to capability, we looked at our own interests in applying our knowledge and mastery of machine learning to our potential future career field in data science. As such, we searched for an accurately sourced, relevant dataset on Kaggle, a web platform for data scientists and machine learning practitioners which is also the host for many publicly available datasets.

The dataset that we have chosen is the *Data Science Jobs and Salaries 2024* dataset created by Fahad Rehman. This dataset was created by extracting job postings in data science from Glassdoor, a job and recruiting web platform that distinguishes itself by allowing former and current employees to review companies.

Using this dataset, the goal was to train a regression model to accurately predict estimated salaries for unseen input. To do this, however, we first must prepare the dataset by extracting relevant

column information and performing data cleaning. After this is done, we can then split the data into training and testing sets, in which the X-values are all the column feature inputs that we have prepared and chosen to be relevant to predict the Y-values for the salary.

We then fit a regression model on the training set. This allows us to have a model that we can now make predictions with. We do this by taking the testing set and invoking the model on these samples to obtain salary predictions. Finally, we can evaluate the results of our model by using different metrics such as error rates and statistical measures.

In this report, we also detail major challenges encountered in the project and how they were overcome, observations in the data, and our conclusions.

## II. TASK DESCRIPTION

The overall goal of this project is to master the development lifespan of a machine learning system and its application to a real-world problem. By choosing This work is split into different tasks. The major tasks in this project are data preparation, model fitting, and model evaluations.

In this project, data preparation is the transformation of dataset column values into applicable formats, such as converting categorical data into numerical and creating new column features based on the extraction of important information from current data columns. One of the first things done in the code data preparation process is the conversion of the Salary Estimate column, which is a range of a minimum and maximum amount in thousands of dollars, to a single estimate that is the average of the minimum and maximum.

Most of the code in this project belongs to the data preparation task. Another example of somethign done in the data preparation section is the removal of the new line and rating from the Company Name column value. We also extract the state from the Location column and make a new column based on this abbreviated state value. Also, we create a new column based on the Founded column that is the difference of the founded year and 2024 to describe the Age of the company. We also remove the complexities from the Job Title column and make a Job_simp column that is the lower case of certain jobs titles found in the dataset. We can also extract the seniority level from the Job Title column and make a new column based on the result to be a possible explanatory variable for the salary. As mentioned earlier, a form of converting categorical data to numerical is present in the code as we convert the count of competitors, if any, in the Competitors column to a new column. We continuously peek into the dataset with every major change to verify that the dataset is becoming more structured, accurate, and accessible in the model training and testing section.

This project uses two different types of regression models, as provided by sklearn. We use Linear Regression and Random Forest Regression. Both are supervised learning algorithms that solve the relationship between the input variables and the prediction output.

Sklearn's pre-tuned Linear Regression model has no hyperparameters because it opts to use the Ordinary Least Squares algorithm to always converge to the same weights no matter how many times it is run. However, it trades these hyperparameter tunings for always finding an optimal solution. On the other hand, an algorithm like Gradient Descent, which does have hyperparameters, only finds an approximate solution by optimization.

The other regression model used, Random Forest Regression, is an ensemble technique which can perform both regression and classification tasks. It works by using multiple decision trees and bagging algorithms to combine these trees in determining a final output. Some common hyperparameters that can be adjusted are the number of trees or estimators to use, which by default is 100, and setting the out-of-bag score as True. This OOB score is a validation technique used to estimate the model's generalization performance and validate the bagging algorithm. In our implementation, we use other metric scores, and we have a high dimensional dataset, so we leave the number of estimators as 100 by default.

The metric evaluations used in this project are mean squared error, the R-squared value, and the mean absolute error. The mean squared error measures the average squared difference between predicted values and true observed values. It is always a positive value that decreases as the error

decreases as well. Therefore, the closer the mean squared error value is to zero, the smaller the error is.

The R-squared value indicates how well a regression model fits the data and represents a percentage of the variance of the dependent variable explained by the independent variable, or the variance of the output label explained by the input features, in other words. The R-squared is calculated by computing the sum of squared residuals divided by the total sum of squares, subtracted by 1. A R-squared value always falls in between 0 and 1, where the closer the value is to 1, the better the model fits the variance.

The mean absolute error is a value that represents the arithmetic average of the absolute errors where the errors are the difference between the predicted value and the true value. It is useful for understanding the magnitude of errors without considering if they are overestimates or underestimates. A small mean absolute error indicates a close predicted value to the actual, observed value. The mean absolute error is calculated in the code without a library function unlike the R-squared and means squared error.

Because our y output value describes a salary described in thousands of dollars, our mean squared error and mean absolute error both appear to be very large. Thus, it is important to keep in mind the scale of the target variable when analyzing the results of these calculations.

## III. MAJOR CHALLENGES AND SOLUTIONS

One of the biggest challenges in this project was the data preparation task, which notably makes up for most of the code. This is due to the inconsistent formatting of some column values such as Job Description, Company Name, Salary Estimate, and Competitors. Given the chance to re-do the assignment again, a different dataset that was already pre-prepared for model training and testing would have been chosen. But, due to the time constraints for this assignment this dataset was sufficient, and data preparation was completed satisfactory regardless.

A specific challenge that came up during the data preparation section was knowing which pieces of information to extract. Initially, we wanted to be able to extract skills from the Job Description column such as programming languages like Python and software like Amazon Web Service. This proved to be ineffective as not every Job Description text value included such skills so there would be many missing values. We had hoped to be able to identify certain skill requirements and trends within the data science job market to be an explanatory variable in predicting salary. One initial idea of mine was to count the number of found skills and make a new column based on this count but the formatting of the Job Description column made this difficult and the idea was scrapped.

Another major challenge in this project was finding out how to have a regression model interpret categorical data like company rating and location to predict salary for. The solution for this came with encoding. Encoding is the process of converting textual data into numerical values and is a vital part of natural language processing to allow machines to analyze information effectively. We had used encoding before such as in homework assignment four to convert label values to vectors when preparing the data for the feedforward neural network. In this project, we use a built-in function provided by the pandas library called get_dummies which converts as many 0/1 variables as possible. Once we finished the data preparation, we chose the relevant extracted and cleaned columns to use in our final dataset for our model. We call the get_dummies function on this final dataset and obtain a new converted dataset where a single column has its categorical values converted into new columns with its values as binary values. For example, our final dataset includes a Job_simp column with values that include ‘data engineer’, ‘data scientist’, etc. In the dummy converted dataf rame, we find columns with the names of ‘Job_simp_data engineer’, ‘Job_simp_data scientist’, etc., where the values of this column are True or False depending on the the value from the final dataset before the dummies one.

## IV. EXPERIMENTS

The *Data Science Jobs and Salaries 2024* dataset has fourteen columns that provide key insight into the current 2024 data science job market. These columns are:

i. Job Title: the title of the job at the company

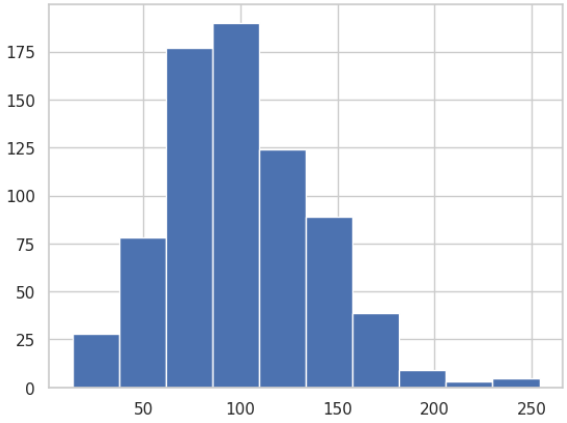
ii. Salary Estimate: company provided estimated salary range
iii. Job Description: company provided job description
iv. Rating: Rating based on former and current employee reviews at company
v. Company Name: The name of the company
vi. Location: Job posting location
vii. Headquarters: Company HQ location
viii. Size: Number of employees at the company
ix. Founded: Year the company was founded
x. Type of Ownership: Whether the company is private, public, government or non-profit organization
xi. Industry: The type of industry where the company provides services such as Energy
xii. Sector: The type of services the company provides in its respective industry such as Oil
xiii. Revenue: The total yearly revenue of the company
xiv. Competitors: List of industry competition the company has, if any, as provided by Glassdoor

With these key data fields available, we can define a machine learning model to predict output values of one of these columns by taking in other fields as input features. As noted above in section II and III, we first extract important information from the columns while cleaning up the formatting of others and make a final dataset to use as our input features.

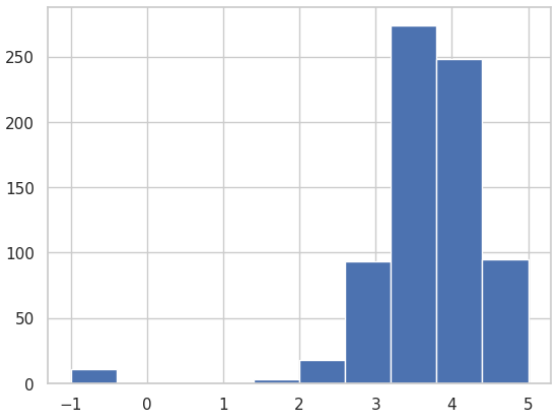
The columns of this final dataset, before we convert all categorical to numeric values, are:

i. Salary Estimate: Simplified average of the salary range
ii. Job_simp: Simplified job title
iii. Rating: Rating based on former and current employee reviews at company
iv. Revenue: total revenue of the company
v. Type of ownership: Whether the company is private, public, non-proft, etc.
vi. Industry: Type of industry the company provides services for
vii. Sector: The type of services the company provides in its respective industry
viii. Age: How old the company is
ix. Seniority: Extracted seniority level of job posting
x. Job_state: State location of the job in the U.S.
xi. Num_comp: The count of competitors the company has

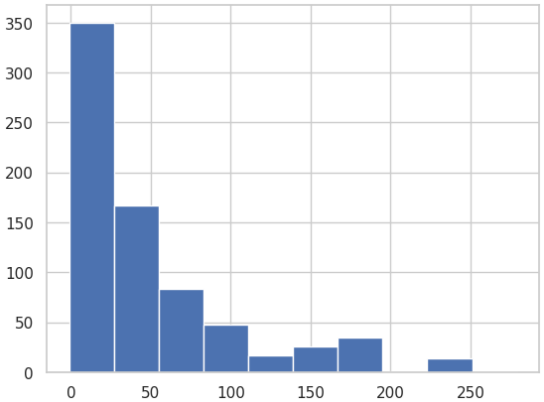
We explore the cleaned, final data and make some plots describing the distribution of different numerical and categorical column values. We also look at the correlation between some of the numerical data columns Salary Estimate, Rating, and Age. Below, we see the distribution of the Salary Estimate in the data science market:




We also look at the distribution of review ratings of each company in the dataset:



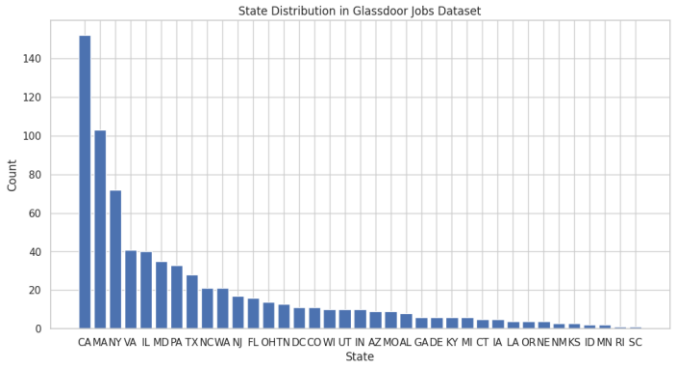

We can also see the Age distribution of each company:



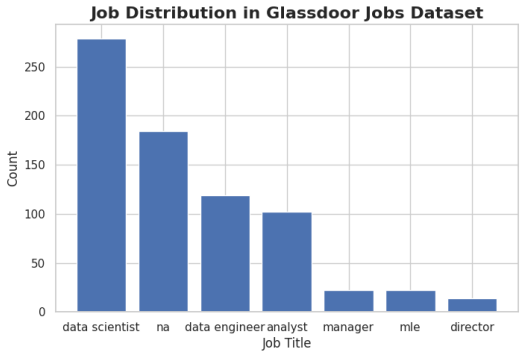

By plotting a correlation heatmap between these

numerical values, we see how important they are in explaining the salary, if at all. This is important for later when we end up dropping columns to decrease the feature noise.


We also look at distribution of company locations by state in the dataset.




Finally, we visualize the distribution of simplified job titles in the Glassdoor data science jobs dataset.




As mentioned in section III, the next step after finalizing a data cleaned dataset is converting categorical column variables into numerical ones by binarizing via dummy encoding. Printing the data types of this new data frame shows us that we have float, int, and boolean values for our column variables.

Now, we are finally ready to split the data into training and testing sets. We create an X variable that is all the columns in the converted data frame besides the Salary Estimate and a y variable that is solely the Salary Estimate column values. We use the sklearn library function train_test_split to split the X and y samples into training and testing sets.

The first model we fit the training set to is sklearn's Linear Regression model. Then, we make salary predictions on the testing set and evaluate the results by our defined metrics from section II to get the following output.

Mean Squared Error: 691.1725256749602
R-squared: 0.5771380896289589
Mean absolute error: 19.181702354375442

We obtain a mean squared error of 691.17, an R-squared value of 0.577, and a mean absolute error of 19.18. As we can see from the mean squared error result, our predictions are vastly different from the observed true value meaning our errors are relatively large. Based off our R-squared value, only half of the variance in the observed values could be explained by our predictions. Our mean absolute error tells us that the predictions were, on average, off by 19.18 thousand dollars for the salary.

We then fit a Random Forest Regression model, provided by the sklearn library, to fit on the training data. The goal here is to visualize if a more complicated model can make better predictions based on the nature of the complicated data input. In our first attempt, we invoke the Random Forest Regression model on the testing set and evaluate via the same metrics as the Linear Regression model to obtain the following output.

Mean Squared Error: 370.9976680480984
R-squared: 0.7730222530173356
Mean absolute error: 11.870624158000835

Here, we obtain a mean squared error of 370.998, an R-squared of 0.773, and a mean absolute error of 11.87. We interpret this mean squared error as the average error difference between the predicted and actual observed values with it being off by about 371 thousand dollars in salary. This is nearly half as small as the MSE from the Linear Regression model, so we can conclude that we made an improvement. The R-squared value tells us that 77% of the observed value variance could be explained by the predicted values. Our mean absolute error tells us that the magnitude of errors was off by 11.87 thousand dollars in salary. We will

evaluate these metric results more closely in section V.

These evaluations of our models tell us that our input variables are not doing such a great job at predicting the salary and that our average errors are off by quite a bit. Now, we attempt to decrease these errors and increase our R-squared results by dropping irrelevant features from our final cleaned data frame in an attempt to better predict salaries based on different data science job information. Due to the number of absent values, we drop the seniority column as it is not sufficiently reliable in explaining salary. We also drop the Rating as salary what employees feel towards a company is not relevant enough in the pay that a company offers because some negative or positive feels people hold go beyond just the pay that the company offers. In addition, we dropped the Age column since we found young companies in the dataset that paid just as highly as old companies, so it was also irrelevant in explaining the pay. Our streamlined data frame has the following column variables:

i. Salary Estimate
ii. Job_simp
iii. Revenue
iv. Type of ownership
v. Industry
vi. Sector
vii. job_state

We also drop rows in this data frame where there are simplified job titles that are missing and get the following distribution in the Job_simp column.

| Job_simp | count |
|---|---|
| data scientist | 279 |
| data engineer | 119 |
| analyst | 102 |
| manager | 22 |
| mle | 22 |
| director | 14 |

Now, we can dummy encode this data frame again to convert categorical variables and split it into training and testing sets just like we did in our initial model attempts. We fit a new Random Forest Regressor on the training set and make predictions on the input test set. Finally, we make the typical evaluation metrics as before to obtain the following output.

```
Mean Squared Error: 323.0630239017638
R-squared: 0.8031775108448103
Mean absolute error: 12.800702129686377
```

We obtain a slightly smaller mean squared error of 323.063 than we did in our first RFR model, a slightly larger R-squared value of 0.80, and a slightly larger mean absolute error of 12.8 than in our first attempt. This mean squared error can be interpreted as the average squared error difference between predicted and observed values. This R-squared value tells us that 80% of the observed value variance can be explained by the predicted values. The mean absolute error tells us that on average our predictions were off by 12.8 thousand dollars in salary.

## V. CONCLUSIONS AND FUTURE WORKS

Although our mean squared errors and mean absolute errors can be seen as relatively high at face value, in relation to the salary, our predictions are not too far off. As a reminder, in our data preparation, we transformed our output variable ‘Estimate Salary’ to be the average of the range of salary in the original dataset. Thus, these errors of 12.80 thousand dollars or 17.97 thousand dollars seem minimal in scale.

In comparing the results of the mean absolute error and mean squared error, we find that the square error is more susceptible to be influenced by outliers in the data than the absolute error likely because of the square term.

In comparing the results of our Linear Regression model to our Random Forest Regression model, we find that the RFR model was much better at handling complex, non-linear relationships between variables that this project focused on. Thus, our R-squared improved drastically between our Linear Regression model and our first RFR model.

In our second attempt at minimizing feature noise, by dropping irrelevant columns to improve the Random Forest Regression model, we can compute the models R-squared score for the training and testing set. These results tell us how well the model fits the data by assessing the proportion of variance in the output variable as explained by the input variable. We can evaluate the final RFR model’s R-squared score for both the training set and testing set to obtain the following output.

```
R-squared score for training data: 0.9035135360520188
```

R-squared score for testing data: 0.8031775108448103

We find that the R-squared score is much higher for the training set than the testing set. This is a sign of overfitting meaning that our model fits too well on the training set that it slightly fails to predict on the testing set.

## VI. REFERENCES

Google Colab link to project code:
https://colab.research.google.com/drive/1UMZqWLSEL2rS_cne5ywa9GbfgqjrxTiO?usp=sharing

Link to Dataset:
https://www.kaggle.com/datasets/fahadrehman07/data-science-jobs-and-salary-glassdoor

https://blog.dailydoseofds.com/p/why-sklearns-linear-regression-has

https://www.investopedia.com/terms/m/mlr.asp

https://www.geeksforgeeks.org/random-forest-regression-in-python/

https://arize.com/blog-course/r-squared-understanding-the-coefficient-of-determination/#:~:text=R%2Dsquared%2C%20also%20known%20as,explained%20by%20the%20independent%20variables.