# HOW MUCH WILL YOU EARN?

Predicting DS, DE, and DA salaries from Glassdoor Job Posting


THE GOODFELLAS FROM THE FIRST ROW

Francesco Caccia - fvc2108
Filippo Mattio - fm2790
Lorenzo Rega - lr3103
Alex Wang - yw3917

# Introduction

The goal of this project is to create and compare different machine learning models capable of estimate data scientist's salary.

The choice to use job data for a project on salaries in data science is likely because job data provides valuable information about the current state of the job market, including the supply and demand for specific skills, the industries, and regions with the highest demand for those skills, and the typical salaries and benefits offered for various positions. By analyzing job data, we can gain insights into the factors that influence salaries in data science, such as location, company's culture, rating, size, and industry of belongness. Additionally, job data can provide information about emerging trends in the field, such as new technologies or skills that are becoming more in demand, which can be useful for career planning and professional development. We felt that since we are all entering the job market this kind of information could help us understand better the industry we are entering, and ultimately be more prepared for the job.

# Obtaining Data

We utilized the Selenium package to extract data from Glassdoor, expanding our scope beyond solely data scientist roles to include data engineering and more. This expansion was motivated by several obstacles we encountered during the scraping process.

Three specific challenges we faced include: first, encountering random pop-up windows that appeared when Selenium was scraping jobs. It seemed as though Glassdoor was aware that people were scraping data from their site, so certain job types would trigger these windows to appear. As a result, our initial Selenium code became stuck when these incidents occurred. We subsequently added a function to detect pop-up windows and close them if they were detected.

Second, we discovered that there were not enough unique data scientist jobs on Glassdoor to scrape. After scraping 900 data scientist jobs, we found only 140 unique jobs, with duplicates comprising 85% of the entire job board. We resolved this issue by scraping data in three ways: by scraping 30 times, including three additional types of jobs (data analysts, data engineers, and statisticians), and combining these with the other 900 jobs we found online in the same format (Ken, 2020).

Finally, we encountered the issue of job descriptions being hidden under the "show more" button. Simply asking Selenium to click on the job and scrape the description was insufficient, as more description was hidden under the button. Therefore, we designed Selenium to click the "show more" button before scraping the job description.

# Data Cleaning and Preprocessing

To clean the dataset, we built a cleaning function. This function is articulated in several steps. We first removed duplicates and unnecessary columns such as "Type of Ownership" and "Founded". We then focused on the "Salary Estimate" column. This is our label for our models.

We wanted to have labeled examples only in our dataset, so we removed all the rows that were missing information on the salary estimate. Some salaries were yearly salaries, some of them were hourly salaries. We turned all of them into yearly to have a uniform dataset. We also removed salary outliers by just keeping those between 0 and 1M$. The column “Job Location” had way too many unique values. This would’ve not helped to build a model: too many dummy variables. Since the Location specified both City and State, we created a column “Job State”, specifying the state in which the job was. The column “Job Title” as we would expect had too many unique values as well. Companies give all kinds of different names for the same position. In order to overcome this problem, we simplified the job title by using a function “title_simplifier”. This function takes a string and checks if the input title contains certain keywords (such as "data engineer", "analyst", "machine learning", etc.) and returns a corresponding simplified title (such as "data engineer", "analyst", "mle", etc.). In order to understand which were the most common job titles we used a wordcloud visualization (Fig 1). From the job title we also built a new feature: seniority level. We did this by creating the function “seniority”. This function simplifies job titles by categorizing them into seniority levels (senior, junior, executive, manager, associate) based on certain keywords present in the input title. If no matching keywords are found, it returns None. We also decided to add another feature to our dataset. We ran a Vader sentiment analysis on Job Descriptions and took the compound score for each description.

## Exploratory Data Analysis

The exploratory data analysis had the objective to enhance the main characteristics, trends (if existing) and summarize our dataset. For this last purpose, we used the pandas function `describe()`, and we plotted it (see Appendix). It highlighted a mean salary of 116,000 $ with a standard deviation of 89,000 $: very high compared to the mean, signal of many outliers. It easier to drive the EDA with numerical variables, because starting from correlations you can understand if there’s any type of trend. For categorical variables it is more difficult, but using visualization tools, such as `maplotlib`, some characteristics may be captured.

In our case, we had 8 categorical variables (Location, Size, Industry, Sector, Revenue, Job State, Job Simplified, Seniority). To properly analyze them we kept in mind 2 metrics:

- The mean salary among each group
- The count of the entries in each group (lower threshold: 10)

We managed to do in this way because otherwise outliers played a big role. For example, happened that a sector with just one entry had a “mean” salary of over 700,000 $. Considering just the first metric, we would have been tempted to believe that sector was really significant. But just one row is not enough to drive conclusion. This has been done with all variables (see Appendix for graphs).

Finally, also a chi-squared test has been run within the independent variables, to understand the level of independence among them, and with the dependent variable, to understand the level of

significance of the independent variables. These results have been combined with the previous analysis described above to properly decide which variable could have been removed or not In the ML models.

# ML Models

## *1. Linear Regression*

We ran several machine learning models; the first one is Linear Regression which was our baseline.

We used as variables for our model: `['Salary Estimate', 'Rating', 'Location', 'Size', 'Industry', 'Sector', 'Job State', 'Job Simplified', 'Vader']`. We made sure to drop all rows having any nan value and used `get_dummies` function to turn all categorical variables into dummy variables. We split our dataset into train ad test set using `sklearn` and ran Linear Regression using `statsmodels`.

## *2. Lasso Regression*

We then tried to improve our Linear regression model using Lasso Regression. Lasso regression is a type of linear regression that can be used for feature selection. It shrinks the regression coefficients towards zero, which can make some coefficients equal to zero and effectively remove those corresponding features from the model. This helps to prevent overfitting and can improve the model's predictive accuracy. We also used cross validation to find the best value of the regularization parameter (alpha). Lasso regression improved our MAE.

## *3. Decision tree*

A decision tree is a graphical representation of a decision-making process that uses a tree-like model of decisions and their possible consequences. It is a supervised learning algorithm used in machine learning and data mining.

It revealed to be good for MAE, which decreased, but very bad for R-squared metric (negative).

## *4. Random Forest*

Random Forest is a type of ensemble learning method that combines multiple decision trees to create a forest. The idea behind Random Forest is to build multiple decision trees and combine their predictions to obtain a more accurate and stable prediction. After gridsearch, Random forest got the lowest error among all models. Below is a graph of feature importance out of random forest. We can observe that rating, job type, sentiment of the job description, job industry as information technology, and whether the job is in California are some of the most important features in predicting the salary.

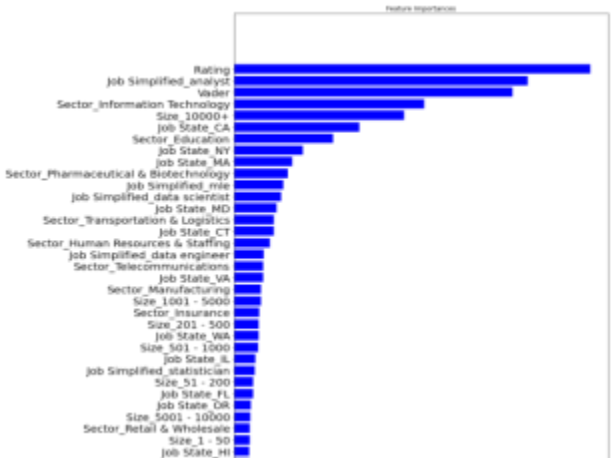



*Figure 1 Feature Importance of Random Forest*

### *5. Support Vector Regression*

The main objective of SVR is to find a line that fits the data points as closely as possible while still allowing some margin of error. We first scale the features since support vector regression is a distance-based algorithm. We picked support vector regression since this model is suitable for sparse data.

It highlighted low MAE, but also low R-squared.

### *6. Bayesian Regression*

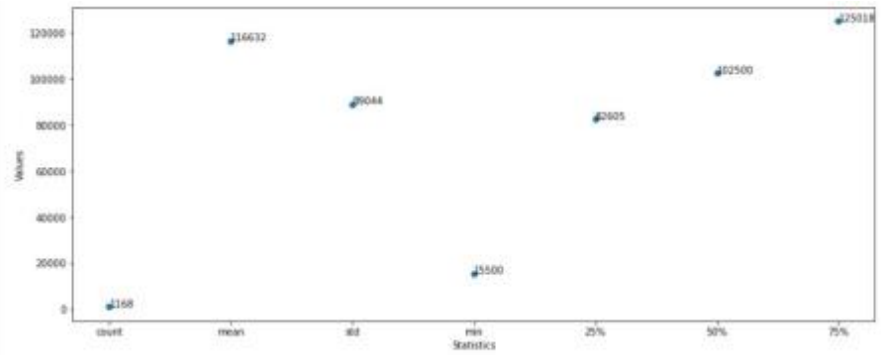
Bayesian regression can be particularly useful when working with sparse data because it can help to regularize the model and avoid overfitting. In the Appendix there is the predicted value versus real value graph of our Bayesian model, which shows a positive correlation.

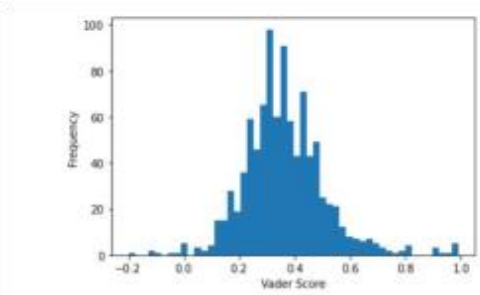
It revealed to be a good model for this dataset.

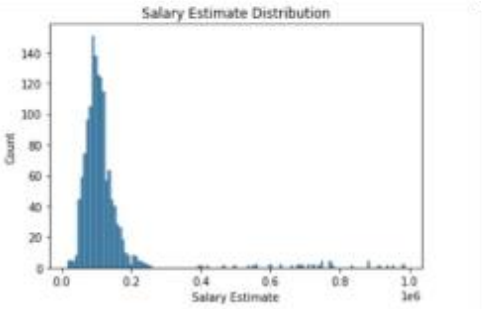
| | RMSE | MAE | R^2 |
|---|---|---|---|
| Linear Regression | 92.98 | 42.66 | 0.224 |
| Lasso Regression | 87.4 | 30.71 | 0.314 |
| Random Forest | 27.14 | 20.59 | 0.376 |
| SV Regression | 35.02 | 26.91 | 0.069 |
| Decision Tree | 39.66 | 28.38 | -0.19 |
| Bayesian Regression | 30.30 | 23.37 | 0.30 |

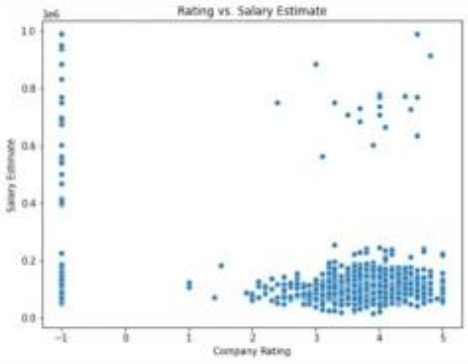
# APPENDIX

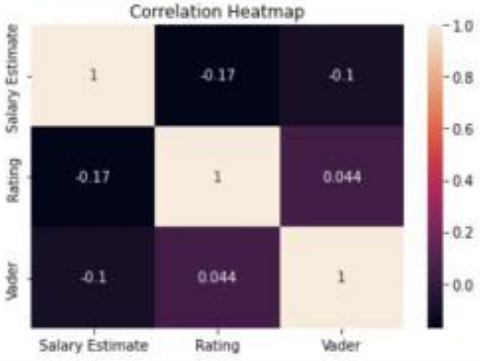
## EDA useful graphs

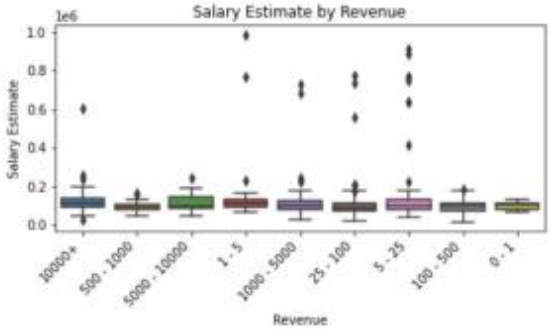


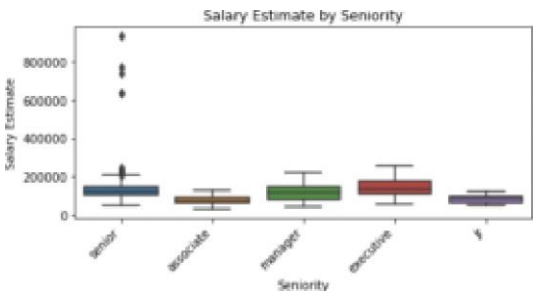



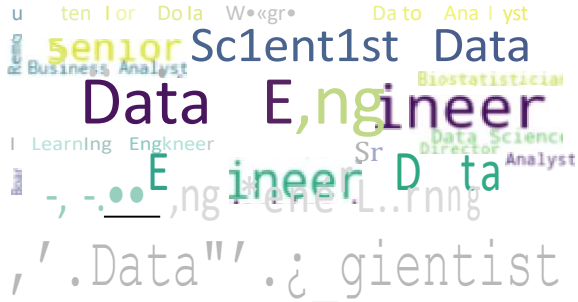



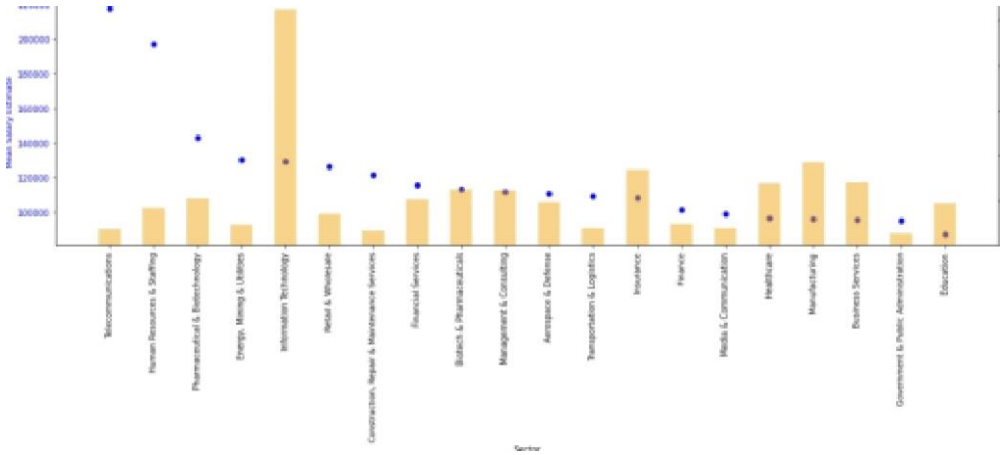



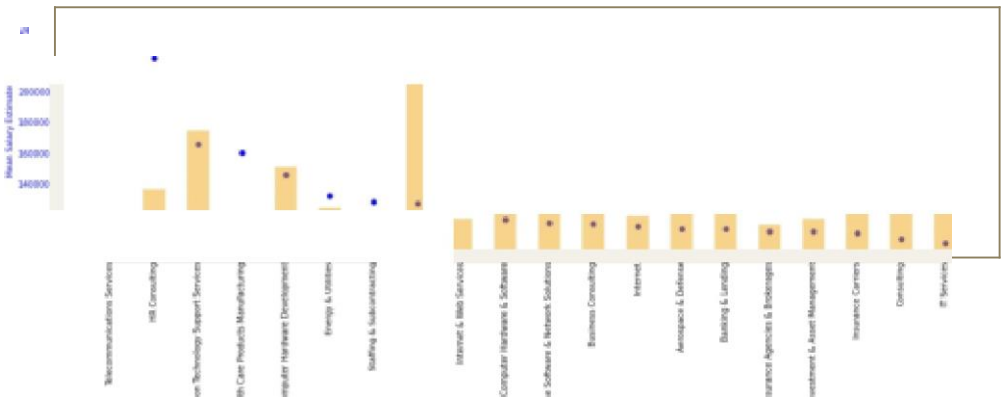



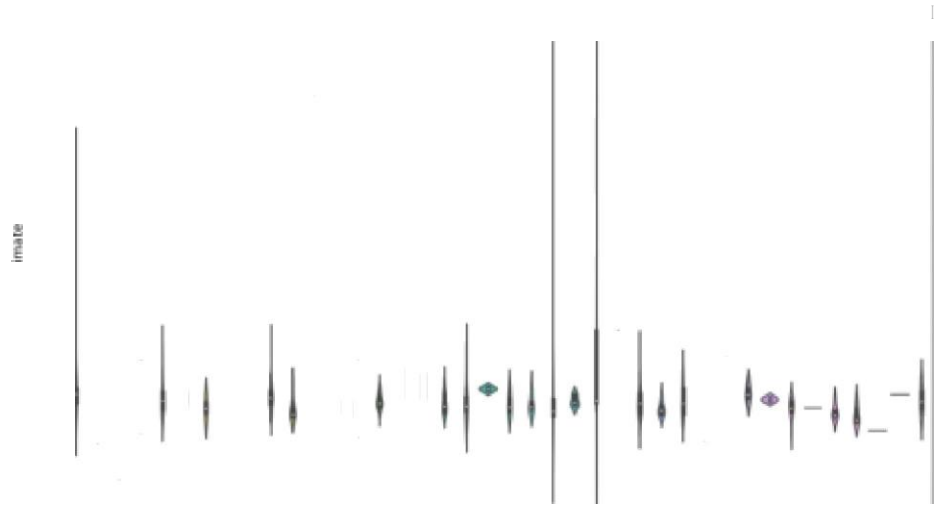


Salary Estimate by Seniority
Salary Estimate
800000
600000
400000
200000
0
senior
associate
manager
executive
Seniority

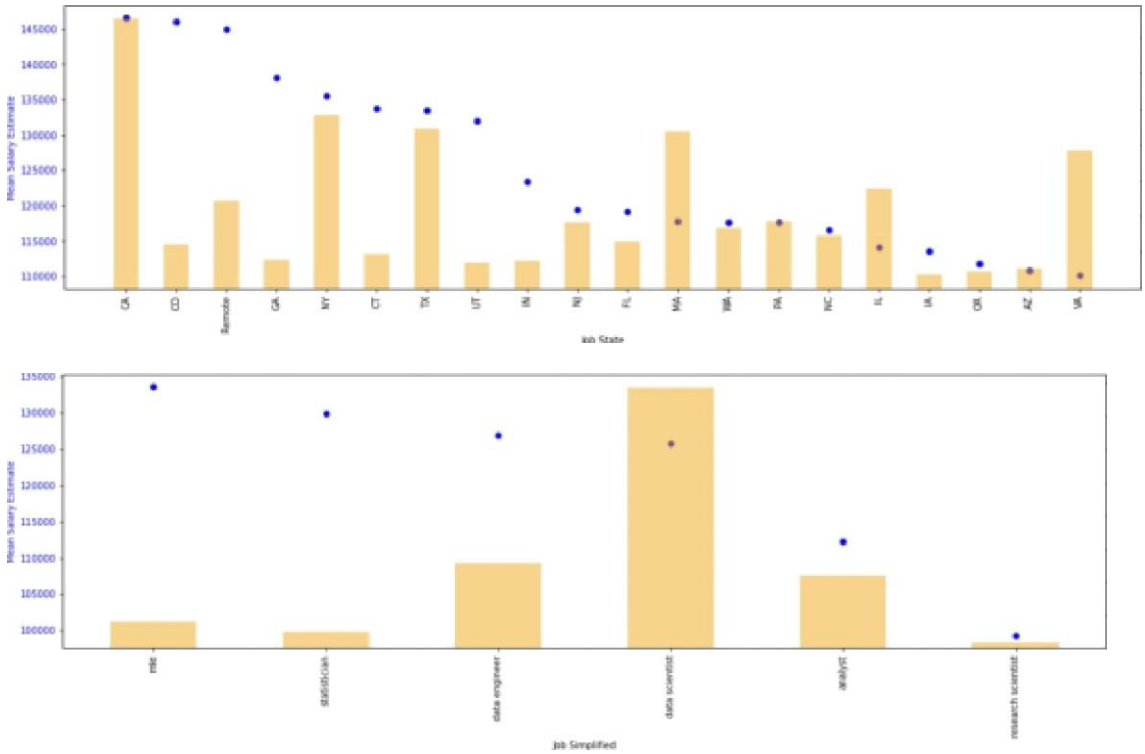

senior
Sc1ent1st
Data
Business Analyst
Data
E,ngineer
Learning
Engkneer
Sr
Analyst
E ineer D ta
,'.Data"'.¿_gientist

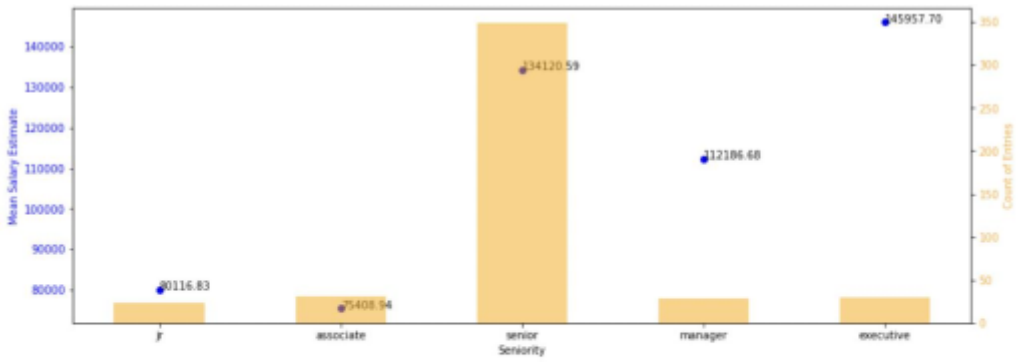

Mean Salary Estimate
280000
160000
140000
120000
100000

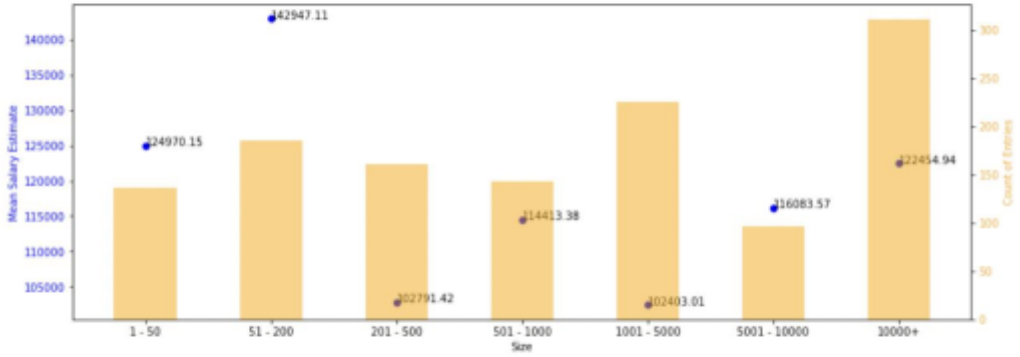

Mean Salary Estimate
280000
160000
140000

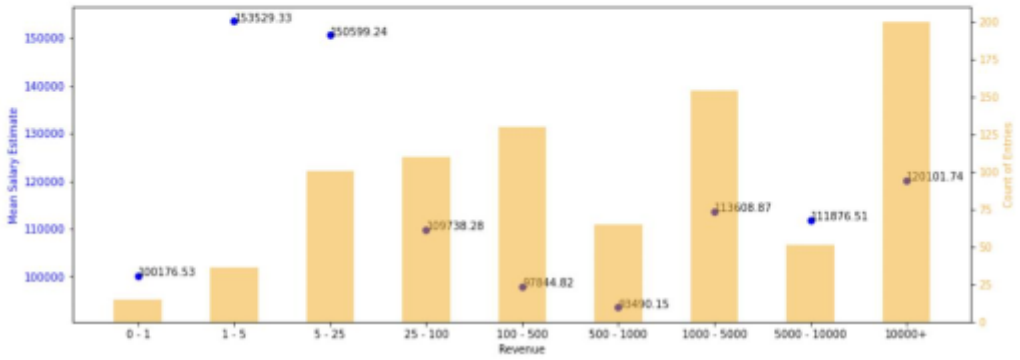
Saiaw estimate nr Se<cør






- **Bayesian Regression**