Salary Prediction for Data Science Candidates

By: Randy Grant

Abstract

This attempts to predict a salary of a data science position based on the web scraped data of indeed.com. The hypothesis is that if the scraped data has enough correlatable features pointing to the target (salary), then there is enough data to predict a fair salary. In contrast, the null hypothesis is that there won't be a way to predict a salary with the scraped data. For this project, the search term "data scientist" was input into the job site search bar, the results of each page were scraped, and then the job's company existing employee "work happiness scores" were scraped as well. The data was then added to machine learning models in order to predict a salary based on the position and company scores.

Design

The website indeed.com provides open source data that shows the basic details of a job, what company posted it, the salary range if available, and a brief description of the position. When digging into each company, most provide data that appear to be good for features. The names of those features are included in the "Data" section of this document. The types of graphs used for analysis were scatter plots, histograms, pairplots, and heatmaps.

Data

The dataset created that was scraped from the website which ended up as having all needed fields had 1,012 rows with 21 columns before dummy variables were added. The 21 columns are as follows:

- *'job_title', 'salary', 'company', 'company_rating', 'job_locations_available', 'region', 'happiness', 'flexibility', 'learning', 'achievement', 'appreciation', 'inclusion', 'support', 'purpose', 'energy', 'compensation', 'satisfaction', 'management', 'trust', 'belonging', 'stress-free'*

More information about these columns are available here. All columns were numeric except for ‘job_title’ and ‘company’ which were later converted with dummy variables or dropped. There was some added data from external sources to feature engineer the data better such as converting the state to the region. The field of the original location was then dropped.

Algorithms

Feature Engineering

1. Converting some objects to integer or float
2. Creating a region column instead of city and state
3. Using max value to extract a single number from the salary field that contained a range
4. Converting 100+ complicated job titles to 4 simple ones

5. Created dummy variables for job_title and region
6. Variance inflation factor (VIF) for multicollinearity detection

Models

Linear regression, linear regression with poly transform, LassoCV with scaler transform, and Ridge were used. The dataset was split into a 80/20 train-test-split.

Model Scores

Linear Regression R^2 training score: 0.01618360980383471 R^2 test score: -0.007692116921428127 RMSE on training data: 48944.14238476381 RMSE on test data: 47415.545716120294	Polynomial R^2 training score: 0.12707569128357754 R^2 test score: -0.07760624483654843 RMSE on training data: 46103.29761274949 RMSE on test data: 49032.82013287742
LassoCV R^2 training score: 0.0 R^2 test score: -0.0018434616459070963 RMSE on training data: 149778.84425216317 RMSE on test data: 149778.84425216317	Ridge R^2 training score: 0.015355791130852614 R^2 test score: -0.007542775610081209 RMSE on training data: 144237.86640779983 RMSE on test data: 155688.19540889576

Tools

BeautifulSoup and Selenium for web scraping, Pandas for data manipulation, Scikit-learn for models and scores, Matplotlib and Seaborn for plots and graphs.

Communication

These slides and visualizations were presented, and this was uploaded to github.