INM701: Introduction to Artificial Intelligence

Predicting Data Science and STEM Salaries

Davis, Joel
Velagala, Priyanka

Table of Contents

1.0 Overview

For our analysis of artificial intelligence techniques, we have selected a dataset composed of user-entered salary data from www.levels.fyi, a site for those employed in STEM industries to share compensation data. For each salary reported, users also report information such as age, gender, level of education, position, company and among other relevant factors.

2.0 Objective

Our objective is to build a regression model that can predict the total compensation of a user based on available data such as education level, job title, years of experience and more. Our motivation is that this model will allow those seeking employment in STEM industries to target locations, job titles or specific 'tags' in job postings to maximise their total compensation, and show the impacts of their current experience on future earnings. Based on the accuracy from the model, we hope to make available to job seekers a baseline/approximate figure they can use to make the negotiation process more transparent.

3.0 Description of dataset

The dataset used for analysis was scraped from levels and made available in a csv format on Kaggle [1]. In total the dataset consisted of 29 columns and 62,642 rows and consists of data reported over a 4 year period from 2017-2021. All dollar amounts are reported in USD.

	Column Name	**Data Type**	**Free-form field**	**% of data populated (2 d.p)**	**Description**
1	timestamp	Datetime	n/a	100%	Timestamp when user reported compensation
2	company	String	✓	99.99%	Company for which the compensation was reported
3	level	String	✓	99.81%	Level associated with user's position in company-specific hierarchy
4	title	String	✓	100%	Job title associated with user's position
5	totalyearlycompensation	Numeric	n/a	100%	Total yearly compensation for role being reported

6	location	String	✓	100%	The workplace location for the role being reported
7	yearsofexperience	Numeric	n/a	100%	The total number of years of work experience (not necessarily in the field)
8	yearsatcompany	Numeric	n/a	100%	The total number of years user worked at the company at the time of reporting
9	tag	String	x	98.64%	General category associated with user's job description
10	basesalary	Numeric	n/a	100%	The amount earned annually before benefits, bonuses etc.
11	stockgrantvalue	Numeric	n/a	100%	The total stock grant value awarded (term undefined)
12	bonus	Numeric	n/a	100%	The total annual bonus awarded (e.g. merit, company performance etc.)
13	gender	String	x	68.81%	Gender identity of the reporting user
14	otherdetails	String	✓	64.07%	Freeform text field capturing various aspects of terms of employment offer
15	cityid	Numeric	n/a	100%	Unique identifier based on the location field
16	dmaid	Numeric	n/a	100%	Unknown
17	rowNumber	Numeric	n/a	100%	Unique row identifier generated by the source system
18	Masters_Degree Bachelors_Degree Doctorate_Degree Highschool Some_College	Binary	n/a	100%	1 if user's highest level of education was the specified degree, 0 otherwise
23	Race_Asian Race_White Race_Black Race_Hispanic Race_Two_or_More	Binary	n/a	100%	1 if user's ethnicity is of the specified descent, 0 otherwise

28	Race	String	x	35.80%	Ethnicity of user
29	Education	String	x	48.48%	Highest level of education obtained by user

Table 3.1 - Explanation of features in dataset

3.1 Data Exploration

For Python code supporting observations noted in this section, please refer to the 'Data_exploration.ipynb' notebook.

Our initial data exploration began with an investigation into each of the existing features to identify any data quality issues. Overall, most columns were well populated with exceptions to race, education and gender which had roughly 35%, 50% and 70%, respectively. This poses a unique challenge in studying the effects of these factors on total compensation. Some of the imputation techniques used for handling missing values are discussed in section 3.2.

For the race and education columns, note that each had an additional 5 columns capturing the same information in a one-hot encoded format, hence only the one-hot-encoded columns were retained.

The following lists attributes and the rationale for discarding certain attributed from analysis after the data exploration phase:

1. **Timestamp** - system datetime field when user reported compensation for a role. This does not necessarily reflect the compensation of the employee for the same year therefore its not useful in studying the temporal effects (e.g. trends) in compensation in the industry or calculating inflation-adjusted salaries
2. **Level** - unique id identifying the user's position in the company hierarchy. While there can be overlap in the a parent company and their subsidiaries, the system is not standardized and may not be useful for analysis when studying effects on compensation across companies
3. **Dmaid** - value appears to be an id field, was unable to find any field description from levels.fyi or any third party source
4. **Location** - the geographical location of the employee's workplace. While this can greatly affect the compensation, as this is a free form field, there were significant data quality issues. The significant challenge being that the level of granularity varied by row, e.g. some identified country while others identified only the city. Since names of cities are unique to the country, it would be difficult to impute a since salaries are reported from all countries
5. **cityid** - this is an index for each location but is not very useful since data quality of location field is poor
6. **Other details** - further exploration of this field revealed it was concatenating values from attributes in the dataset with some rows consisting additional terms of employment such

as signing bonuses, overtime etc. As the format was not standardized, to extract and analyze the information being stored in this field would have required significant data scrubbing.

7. **Company -** initial analysis of this field revealed 1631 unique values. On further inspection, we found that there were multiple variations for some companies. On applying data cleaning techniques such as removing leading/trailing spaces and using Levenstein to calculate similarity between names to remove duplicates, we were able to shorten the list by ~33% to 1100 unique values. After evaluating the distribution of salaries reported per employer, we decided to not include this as a feature in our model as ultimately we didn't feel that there were sufficient data points reported per employer to justify incorporating this as a feature.
8. **Years at company -** In the majority of examples, years at company was left null (and therefore assumed to be zero) or identical to the number of years of experience. Therefore, we believe there was high multicollinearity present between these variables, potentially increasing variance and decreasing our R^2 value, so we decided to omit this from our dataset.
9. **Base Salary, Stock, Bonus -** these fields were believed to be the components of the 'total yearly compensation' field and so we realised where these fields are populated correctly, they would be 100% accurate at predicting total yearly compensation, and would therefore create overestimates for the ability of our model to perform on unseen data. As a result, we decided to eliminate these fields.

3.2 Data Cleaning

For Python code supporting observations noted in this section, please refer to the 'Data cleaning.ipynb' notebook.

Resolving any issues in data quality through imputation or removal would increase the accuracy of our model as our training and testing sets would have more complete data on which we base our regression algorithms. Therefore, we have analysed each feature to find missing data and where appropriate, either omit or impute values:

1) **'Gender'** - There were approximately 16,000 missing values for gender which could be allocated to one of 'Male', 'Female' or 'Other'. We first generated a dataframe with salary groupings of 50k and computed the ratio of Male/Female/Other within each group. We then chose to use a variation of random sample imputation, where records with a missing value in each salary group were randomly assigned a value with the same proportions as that of the earlier discussed dataframe. This allows us to preserve the distribution of values in an unbiased (under the assumptions of Missing Completely at Random data), computationally-efficient manner. Additionally, there was 1 row with the value 'Software Engineer'. This was assumed to be in error and as such a new value was imputed using the same method.
   One of the challenges in employing this method is that we could potentially be under-representing the "Other" category. By observing the ratio of gender identities for

each salary group, “Other” had at most 1.45% (in the 400-550k salary range) and at least 0.51% (in the <50k range). This implies that for the lower end, at least 200 values need to be imputed for “Other” to have any representation, otherwise the other categories would dominate.

2) **‘Race’** - there was 1 row which the user had both “Asian” and “Hispanic” selected. For consistency, this was addressed by encoding “Two or More Races” to 1 and updating the former columns 0.

3) **‘Tag’** - there were a total of 3,058 unique values found in this column. On review of the levels.fyi platform, we were able to reverse-engineer the UI field (called ‘Tag/Focus’) from which this data was extracted and noticed that currently on the platform users are only able to select one from a list of 17 predefined values [2]. Since data cleaning for >3k values would be a fairly time consuming task, we decided to limit the scope of our data cleaning activity to just the set of tags that appeared for at least a 100 records. This significantly shortened the list to 36 unique values. For those same values, based on our research and judgement, we mapped the values to one of the 17 predefined values for the platform. As a consequence of this, we had to discard ~8k rows from our dataset.

4.0 Models

4.1 Education Classification

For Python code supporting observations noted in this section, please refer to the ‘EduClassification.ipynb’ notebook

In order for our final regression model to provide good accuracy, it was necessary to have a complete dataset. The last remaining feature, education, had a significant number of blank values (51.52% of total). Dropping records for which this field was blank wasn’t a viable option as this would result in for one, a loss of over 25k data points and two, a loss in what we believe to be a highly influential factor in salary prediction . Given the relevance of this feature, we decided to employ a more sophisticated classification model to impute values for missing records as we believe this would allow us to generate a value closer to the true value.

4.1.1. Data Preprocessing

Using the clean dataset, for the non-numeric features such as ‘titles’, ‘gender’ and ‘tags’, we used one-hot encoding to preprocess the data for the classifier. This method allowed us to create a binary column for each value in each feature. In our case, this resulted in a total of 34 features.

For our numeric features, ‘totalyearlycompensation’ and ‘yearsofexperience’, we used normalization and standardization techniques in order for the models to converge faster. As

‘totalyearlycompensation’ is normally distributed between a wide range of values (from 0 to 10^6) this was a good candidate for data standardization. We used scikit’s StandardScaler which centers the mean of the data to 0 and scales them by their standard deviation. For ‘yearsofexperience’ however, as the data doesn’t follow a gaussian distribution, we used sci-kit’s MinMax scaler. This method sets the smallest value in the feature to 0, the largest 1 and scales the remaining values to fall in between. These techniques minimise input value sensitivity issues and improve our computational performance by simplifying calculations.

Lastly, we encoded our target variable, education, which had string values. Here, we have used ordinal encoding to maintain an order to the values - 1 indicates the lowest level of education (‘High School’) and 5 indicates the highest (‘PhD’). This is because education has correlations between degree classification, time and ability, which can allow for some more statistical computation than nominal values as found in ‘tags’, ‘titles’ and ‘gender’, where the labels have no bearing on one another (e.g. ‘Male’ and ‘Software Engineer’).

4.1.2 Classification Algorithm Selection

To evaluate between different classification algorithms, we used k-fold validation to determine which model has the best performance. As the goal of the classifier is to impute a value and given that we were using a supervised technique, we first split the dataset with non-null education values as the train set and those with null education values as the test set. For the k-fold cross validation we used 5 different classification methods (Logistic Regression, Random Forest Classifier, K-Nearest Neighbors, Gaussian Naive Bayes, Support Vector Machine (SVM)) with default parameters to evaluate if certain algorithms outperformed the others in our train set. Both SVM and logistic regression yielded comparable results with 58.3% and 57.8% accuracy, respectively. However, SVM had a significantly higher computation time of 219.82s, compared to the 9.68s of logistic regression. As the accuracy for both methods was very similar, we decided to move forward with the more computationally efficient method and use logistic regression.

```
Logistic Regression  mean accuracy:  57.812 % std:  0.005 % training time:  9.68 s
RandomForest Classifier  mean accuracy:  53.376 % std:  0.001 % training time:  11.81 s
K nearest neighbors  mean accuracy:  54.784 % std:  0.003 % training time:  10.0 s
Guassian Naive Bayes  mean accuracy:  5.741 % std:  0.002 % training time:  0.17 s
SVM - one vs rest  mean accuracy:  58.305 % std:  0.005 % training time:  219.82 s
```

Figure 4.1: Outputs from k-fold cross validation

4.1.3 Hyperparameter Tuning

Following this, for the logistic regression model, we began tuning our hyperparameters to improve the performance. The hyperparameters were set based on certain aspects of the nature of our problem. For example, in our case, for solvers, since we’re working with a multi-class problem, both ‘newton-cg’ and ‘lbfgs’ were used as they handle multinomial loss. We also tried ‘liblinear’ which unlike the other two uses a one versus the rest scheme. This was in addition to other parameters such as penalty and regularization [3]. Using a grid search cross-validation method , we found the best model had an accuracy of 58.0%, showing minimal

improvement from the pre-tuned model [4]. Using the same model, we imputed values for education for the test set and joined this to the original set, resulting in a complete dataset.

4.2 Salary Prediction

For Python code supporting observations noted in this section, please refer to the 'PredictTotalCompensation-entireDS.ipynb' notebook

4.2.1 Regression Algorithms

With the resulting dataset, we began exploring different regression algorithms following a method similar to that described for the classifier in section 4.1. The algorithms explored initially through k-fold cross validation were linear regression, stochastic gradient descent, random forest and gradient-boosted regressors (GBR). The outcome of this initial experimentation was the gradient-boosted regressor yielding the best results with an R^2 value of 0.281 and an FVR value of 71.91% (fraction of variance unexplained [5], calculated as $1 - R^2$). The model's total training time was 19.91s.

Once again, while tuning the GBR model, for cross validation, we used grid search with RepeatedStratifiedKFold from the scikit-learn library. This method allows us to both set the number of splits (5) and the number of times to repeat the cross-validation (3), so we assess based on the mean response value across all runs. This significantly reduces bias in our model as we are utilising the maximum amount of data possible for our training and testing sets and reduce variance, as we are using the maximum amount of data possible for our validation set.

As a result we saw improvement of our R^2 metric to 0.34 and found the best performing model had a learning rate = 0.01, a maximum tree depth = 8, a minimum samples per leaf of 100, a minimum samples per split = 10 and a subsample = 0.8. As shown in the notebook, this model resulted in a root-mean-squared error (RMSE) value of 272.32.

4.2.2 Neural Network

For comparison, we decided to try another method: a neural network. Using TensorFlow and our cleaned dataset, we set a baseline value using 3 hidden layers of arbitrary length (100, 200 and 50 nodes, respectively), a relu activation function, an Adam optimiser and mean-squared error as our performance metric. This yielded a RMSE value of 274.49 and an R^2 value of 0.29.

From here, we adjusted the features of this base model to try and improve performance: implementing dropout to improve generalisation error and reduce overfitting, adding more hidden layers and adjusting the number of nodes in each layer. However, despite our attempts at improvement, we were unsuccessful, the results from our tuned neural network having marginally improved RMSE (down to 273.11 in our best-case scenario), our R^2value fell more significantly to 0.27 in our best case and as low as 0.19 in our worst.

5.0 Model Evaluation

As shown, our best results show an RMSE value of 272.32, implying large differences between our estimations and the true values, and an R^2 value of 0.34, meaning that we can only explain approximately 34% of variance in our model. To do this, we can compare to other models completed in our domain. These comparisons show that using similar regression models and techniques, R^2 values of 84% [6] and 97.5% [7] can be found, indicating that our model did not, overall, perform particularly well. It is difficult to compare RMSE to other models as the values are specific to each problem domain, but as the majority of our distribution of values was between approximately 50,000 and 300,000, our value of 272.32 does appear to be quite large, further indicating low model performance.

We have used RMSE and R^2 as our model evaluation tools as they are common in use for regression problems, allowing us to easily indicate the differences between true and predicted values, along with the variance of those values. However, RMSE can be negatively affected by a right-skewed distribution of outputs, as it penalises large error values more harshly than small errors. In our model, the total yearly compensation is right-skewed even after our normalisation, so it is possible that we should have used an alternative metric, such as mean absolute error (MAE). This penalises error values in a linear manner, reducing the impact of outliers on our model and so would likely yield a smaller error value.

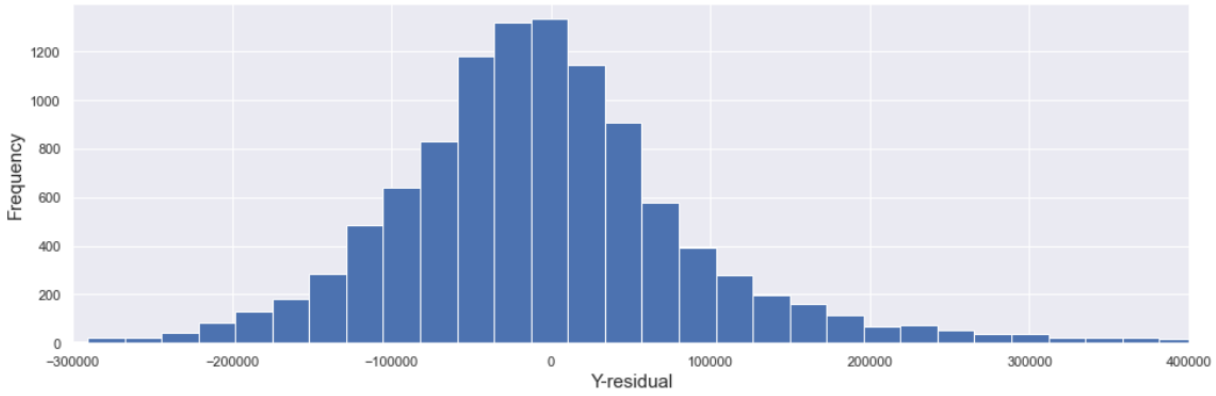

Figure 5.1 - Y-residual graph with calculated errors

6.0 Conclusion

Overall, our model did not perform well but we did implement several machine learning and artificial intelligence techniques to investigate the impacts of a variety of features on our output, total yearly compensation for employees in STEM and data science.

I believe something upon which we could have improved was our data cleaning and exploration. We decided to eliminate several features from our modelling due to time constraints or complexity - ‘Location’, for example - but on reflection, I believe the additional work could have

resulted in a significant improvement in the accuracy of our models. In the case in point (location), It stands to reason that there is a correlation between where you work and how much you earn, and using the site's own rudimentary statistical analysis features, the median salary of a software engineer in New York is $180,000 [8], whereas the median value of the same role is $43,000 in India [9]. This correlation would imply some improvements to our model could have been made by inclusion of this feature.

Furthermore, inclusion of the 'Company' feature could have yielded some useful results - our aim was to gain insight in compensation for roles across the STEM and data science industry, and though this column contains >1,500 unique values for 62,000 rows (and so we believed the data to only be of sufficient quality for a handful of companies), those companies with large numbers of reported data points could have been useful in making predictions. For example, the so-called "FAANG" group of companies composed of Facebook, Amazon, Apple, Netflix and Google make up almost 25% of our data points and it is reasonable to assume that these companies have higher average pay than other companies in the industry. Therefore, we could have created a new variable (a.k.a feature engineering) to indicate whether the row was FAANG or non-FAANG, further completing our data set.

A second point of reflection was that of our imputation of values in 'Education'. After completing our model, it occurred to us that using such a large number of imputed values (>50%) could negatively impact our performance. So, in comparison, we ran our models again on a smaller dataset composed of solely user-entered values for education, dropping all other rows. This can be found in the *'PredictTotalCompensation-UsingTrueEduValue-entireDS.ipynb'* notebook, in the GitHub repository. The results of this experiment were lower R^2 values and higher RMSE, possibly indicating either that our imputation was not largely impactful on our results, or that the dataset was reduced too far, and the approximately 20,000 remaining rows do not provide enough data to successfully implement a machine learning model.

Investigation too could have been made into the impacts of specific features on total compensation - race and gender to be precise. This would be an interesting piece of future work to undertake, as we believe that there could be some discoveries to be made from this data that could help improve equal pay conditions in the industry.

7.0 References

[1] Ogozaly, J., 2021. *Data Science and STEM Salaries*. [online] Kaggle.com. Available at: <https://www.kaggle.com/jackogozaly/data-science-and-stem-salaries> [Accessed 21 December 2021].
<https://www.levels.fyi/Salaries/Software-Engineer/India/> [Accessed 21 December 2021].
[2] Levels.fyi. 2021. *Levels.fyi - Add Your Compensation Anonymously*. [online] Available at: <https://www.levels.fyi/addcomp_form.html> [Accessed 21 December 2021].
[3] scikit-learn. 2021. *sklearn.linear_model.LogisticRegression*. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html> [Accessed 26 December 2021].
[4] Brownlee, J., 2021. *Tune Hyperparameters for Classification Machine Learning Algorithms*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/hyperparameters-for-classification-machine-learning-algorithms/> [Accessed 22 December 2021].
[5] Statistics How To. 2021. *Residual Variance (Unexplained / Error)*. [online] Available at: <https://www.statisticshowto.com/residual-variance/> [Accessed 27 December 2021].
[6] Pianalytix.com. 2021. *Salary Prediction Model Using ML - Pianalytix - Machine Learning*. [online] Available at: <https://pianalytix.com/salary-prediction-model-using-ml/> [Accessed 23 December 2021].
[7] InsideAIML. 2021. *Project 4: Prediction of salary Based on years of experience-InsideAIML*. [online] Available at: <https://insideaiml.com/blog/Project-4:-Prediction-of-salary-Based-on-years-of-experience-1121> [Accessed 23 December 2021].
[8] Levels.fyi. 2021. *Software Engineer Salaries in New York City | Levels. fyi*. [online] Available at: <https://www.levels.fyi/Salaries/Software-Engineer/New-York-City/> [Accessed 21 December 2021].
[9] Levels.fyi. 2021. *Software Engineer Salaries in India | Levels.fyi*. [online] Available at:

7.1 GitHub repository

https://github.com/JoelEDavis/INM701