Predicting Glassdoor Job Ratings

Abstract

In the current instability of the job market, the challenge of finding a job that's the right fit and keeping that job becomes more difficult than ever. Job reviewing sites like Glassdoor are not uncommon as the exchange of information has become more accessible and more pivotal in people's decision making. Using the Glassdoor Job Ratings data, the purpose of this project was to determine the features and model(s) that would most accurately predict the overall rating of the job. We compared several different models with both numerical and text data and we found that a hyper-parametrized logistic regression model had the best accuracy utilizing various numeric features, sentiment analysis, and TF-IDF.

Dataset and EDA

For this assignment we decided to explore Glassdoor job reviews from a Kaggle dataset. Before any data processing, this dataset had a total of 838,566 entries and 18 columns ranging from basic information such as the company being reviewed, the user's job title, and the user's overall rating to more detailed information such as whether the user recommends the company, their location, and sub-ratings on various categories.

To prepare for our model, we dropped three columns – diversity inclusion, CEO approval, and outlook – as we deemed them more irrelevant to our task of predicting overall rating. We then created some graphs to visualize our data, particularly those related to overall rating.

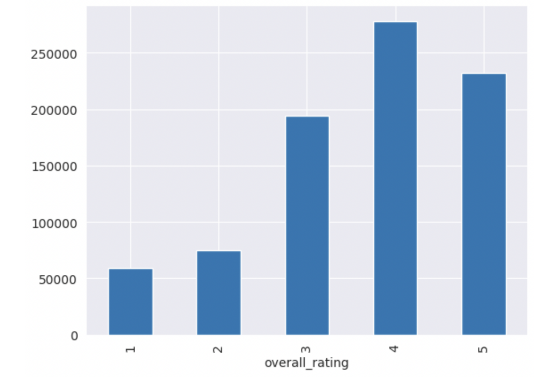
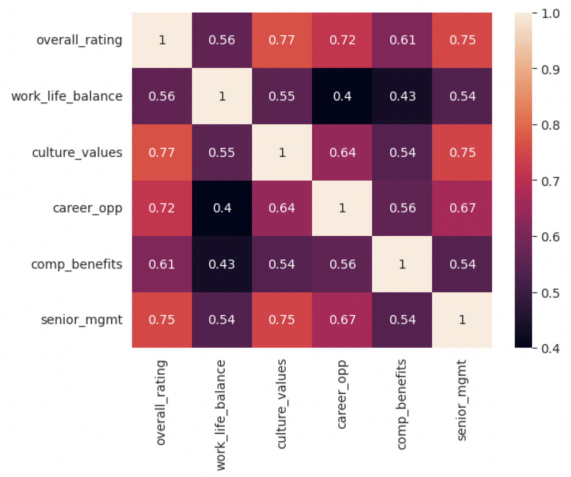


Figure 1: Distribution of ratings in Overall_Rating Column

In Figure 1, we see that there is an imbalance of ratings with there being less lower ratings and more higher ratings, a rating of four being most common. With this information, we can possibly assign weights to each rating to create a model that predicts ratings that are more common compared to others.



	overall_rating	work_life_balance	culture_values	career_opp	comp_benefits	senior_mgmt
overall_rating	1	0.56	0.77	0.72	0.61	0.75
work_life_balance	0.56	1	0.55	0.4	0.43	0.54
culture_values	0.77	0.55	1	0.64	0.54	0.75
career_opp	0.72	0.4	0.64	1	0.56	0.67
comp_benefits	0.61	0.43	0.54	0.56	1	0.54
senior_mgmt	0.75	0.54	0.75	0.67	0.54	1

Figure 2: Heat Map of Numerical Columns

The first row of the heat map in Figure 2 allows us to see which column has the highest correlation to overall rating. Based on this information, we decided that we were most interested in exploring ratings for work life balance, career opportunities, and culture values for our predictive task in order to get the highest accuracy rate for our models. While senior_mgmt has a high correlation to overall rating, we decided to go with work life balance instead as we believed this factor was more indicative of a better or worse rating in the real world.

After some basic exploration of our data, we processed some more data. To begin, we replaced the original encoding of positive (v), negative (x), neutral (o), and mild (r) values in the "recommend" column to their actual values ("positive", "negative", "neutral", and "mild") to avoid confusion.

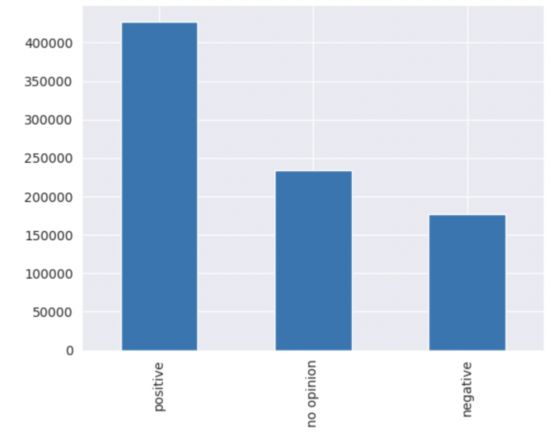


Figure 3: Distribution of "recommend" column

Looking at the distribution of positive, negative, and neutral recommendations in Figure 3, it is interesting to see a lot more positive recommendations than negative or neutral ones. This could potentially be due to the nature of being provided a job which leads to an overall positive opinion of the company. Additionally, it is interesting that there are the least amount of negative recommendations as one would assume that people write reviews either extremely for or against the company that they are reviewing. However, it's also important to note that a lot of people opt out of answering any optional fields which could account for more "no opinion" responses.

Finally, we combined the "pros" and "cons" columns to create a "combined_review" column in which we removed all punctuation, capitalization, and English stop words using the Natural Language Toolkit library.

Predictive Task

The main goal of this project is to predict the overall_rating score using sub-ratings – work life balance, career opportunities, and culture values – as features alongside the combined positive and negative reviews. We chose these ratings: work-life balance and career opportunities, in particular, because we believed these values to be most relevant in determining an overall rating. For the last rating, we chose culture values from a variety of other values due to the fact that it resulted in the highest accuracy when using a logistic regression classifier in combination with our two other selected values.

We decided to combine the positive and negative reviews columns into one column as "combined_review". Subsequently, we removed stopwords and other unnecessary characters such as extra spaces. After cleaning the combined_review column and renaming it to "cleaned", we found the sentiment for the pros and cons columns, respectively. The reasoning behind creating these additional features is the assumption that reviews with more extreme usage of adjectives in either the pros or cons section would ultimately help determine what that user’s overall impression is of that particular company. With this in mind, we additionally found the TF-IDF of the combined reviews as we believe this would further help us predict the overall rating of a review.

Only one metric was used throughout our report. We decided to use accuracy as our primary metric because although the dataset is imbalanced, we wanted to generalize our findings such that the accuracy score is representative of what is most likely to occur in the real world (e.g. an individual either really enjoys working at a company or doesn’t).

Accuracy is calculated as such:

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Model

Initially, we decided to create a baseline model with no manipulations of the hyperparameters. We used the TfidfVectorizer and determined that MultinomialNB would be the best choice for the classifier. MultinomialNB is a good choice because it is good for classification for data that has features such as word counts for text classification [2].

After using a pipeline to consolidate these two methods, we took the column with the cleaned review data as our X values and the overall_rating as the y. The final step to this section of the procedure was to use the train_test_split method to divide our data into training and testing sets. The final step was to use the pipeline and fit the training set and predict the reviews using the testing set, which got us an accuracy of 0.33. Obviously this accuracy is low due to the limited number of hyperparameter manipulation. To deal with this issue, we decided to use other models/methods and see what we could do to increase the accuracy.

One such method was using a similar procedure, except rather than dealing with multiclass classification, we converted the overall_ratings to binary. Ratings four and five were assigned to group “1” and ratings one, two, and three were assigned to group “0”. “1” symbolized the more positive ratings and “0” symbolized the negative ratings. The process after that was quite similar to the one detailed above. The cleaned column was used, along with a pipeline. The only real difference is that the classifier we used this time was Logistic Regression. Logistic regression is a good choice for instances where we must predict the likelihood of a certain occurrence using past data. In this particular project, we are doing exactly that.

Unsurprisingly, our accuracy was significantly higher (0.75). This is because binary classification allows for more “error” in prediction, so to speak. Since the 5 classes were combined into 2, there is a 50% chance of choosing the right class, which immediately increases the odds. While 0.75 is a pretty good accuracy rate, we still wanted to look into more complex models. One approach was to change the binary classification to 3 classes, where “1” was for ratings four and five, “0” was for rating three, and “-1” was for ratings one and two.

The process is the same as the one detailed above, which gave an accuracy of 0.68. Additionally, we also decided to incorporate the numerical columns for our search for a better model. Out of the several numerical columns such as ("work_life_balance", "career_opp", "culture_values", "comp_benefits") and more, we collectively agreed to choose the ones that we deemed more relevant to the industry. We ultimately decided on ("work_life_balance", "career_opp", and "culture_values") as these seemed the most interesting and relevant to us out of all of the numerical columns. As for cleaning the data like fixing the missing values, we used IterativeImputer which actually decreased the accuracy of the model compared to when we just dropped missing values. This is likely due to the fact that by adding more data that wasn't initially there, more variability was added which would lower the accuracy of our model.

Besides Logistic Regression, we also tried a variety of different models. Initially, we explored KNN, Random Forest Classifier, and Naive Bayes. We used the same relevant numerical columns ("work_life_balance", "career_opp", and "culture_values") and dropped missing values. Our baseline accuracies for these starter models were 0.57, 0.62, and 0.33 respectively. Seeing as we were dropping about a quarter of the data, we decided to again impute using Sci-Kit Learn's IterativeImputer. We also decided to use Decision Tree Classifier rather than Naive Bayes given the low accuracy score. After imputing and adding the "recommend" column by encoding "negative" to -1, "no opinion" to 0, and "positive" to 1, our final accuracies for KNN, Random Forest Classifier, and Decision Tree Classifier were 0.54, 0.59, and 0.59 respectively. Given these accuracies and intuitively deciding on the best model, we decided to stick with Logistic Regression to utilize the numerical features.

After much consideration, we decided to use LogisticRegression over RandomForestClassifier due to the fact that the RandomForestClassifier is prone to overfitting if we do not hyper-parameterize. To optimize our parameters, we used scikit-learn's cross-validation library GridSearchCV, which finds the most optimal parameters by cross-checking different combinations of parameters on every validation subset in our training data. Using the same relevant columns as above ("work_life_balance", "career_opp", and "culture_values"), TF-IDF, features created using sentiment analysis, and cross-validation techniques to optimize our parameters, we had a final accuracy of 0.68.

Model	Features Used	Cleaning Method	Accuracy
Baseline Model (MultinomialNB)	Text	Removal of Stopwords/ unnecessary characters	0.33
Multi-Class Classification (3 classes)	Text	Removal of Stopwords/ unnecessary characters	0.68
Binary Classification	Text	Removal of Stopwords/ unnecessary characters	0.75
Logistic Regression	Work life balance, Career opportunity, Culture values	Dropped missing values	0.61
Logistic Regression	Work life balance, Career opportunity, Culture values	IterativeImputer	0.55
K-Nearest Neighbors	Work life balance, Career opportunity, Culture values, recommend	IterativeImputer	0.54
Random Forest Classifier	Work life balance, Career opportunity, Culture values, recommend	IterativeImputer	0.59
Naive Bayes (for multinomial models)	Work life balance, Career opportunity, Culture values	Dropped missing values	0.33
Decision Tree Classifier	Work life balance, Career opportunity, Culture values, recommend	IterativeImputer	0.59
Logistic Regression (Final Model)	**Work life balance, Career opportunity, Culture values, Sentiment (pros), Sentiment (cons), TF-IDF**	**IterativeImputer**	**0.68**

Table 1: Report on Model Performance

Relevant Literature
There are other methods that can be implemented to increase the accuracy of the predictive model. One such example is using tensorflow/keras to create a predictive model. In the article *Predicting the ratings of reviews of a hotel using Machine Learning* published by Analytics Vidhya, there is a detailed description of the steps taken to create their model using the text reviews and ratings from Tripadvisor [1].

Initially, the stopwords and irrelevant characters are removed before building the model. Some extra steps they took was converting contractions to their full form. After cleaning the text data, the reviews were encoded so that the ratings were on a scale of zero to four, rather than one to five. Machine learning cannot take text data directly, so the text was converted to sequences, which was created using a function in Keras. Subsequently, a sequential model was created. After running the model, the accuracy is around 60%. When printing out the classification report, we see that the model they created was better at predicting the ratings for one, four, and five rather than two and three. Something that this article mentions is looking into ratings two and three and seeing if there is a way to possibly manipulate the model to take care of this.

Compared to our model, this model starts off with similar steps (cleaning the data) with nearly identical processes with the stop words. The tensorflow model takes the data cleaning step a little further, but it does not change the accuracy of the model significantly enough for us to take into consideration. The tensorflow model is definitely more complex and takes a significantly longer amount of time to run due to the different layers and size of the data, while our model runs relatively quickly. A similarity between our two models is that we both observed that our model is better at predicting certain ratings compared to others.

Results
Our final model consisted of a logistic regression classifier that utilized text features through TF-IDF, numerical features, and sentiment analysis that had a final accuracy of 68%. Compared to other alternative classification models, such as RandomForestClassifier, our model performed equally well, if not better, than the next model with the highest accuracy. This indicates that regardless of what model we used, the upper bound of our accuracy metric would be in the range of the high 60s up to the lower 70s. It would be possible to attain a higher accuracy using a RandomForestClassifier and hyper parameterizing the model, but this would be prone to overfitting and would likely not be as good as a logistic regression model.

Feature representations that worked well were the already existing numerical features. This included sub-ratings such as work-life balance, career opportunities, and more that were already in the dataset. This is likely due to the fact that these sub-ratings are highly correlated with the overall rating of the review. On top of using these features, we also looked into using sentiment analysis and different word models on the pros and cons section of each review. Our initial belief was that applying word models to these reviews would result in a higher accuracy rating because these tell us, in theory, the most about a person's opinion on that company. However, using word models, in our case TF-IDF, surprisingly didn't increase our accuracy by that much, only increasing our model accuracy by 1%. Using sentiment analysis on the pros and cons section increased our accuracy by 11%, a sizable difference from our model without these features.

Our interpretation of the model's parameters are as follows: We chose a C value of 100, meaning that we're essentially telling the model to weigh the training data heavily and a lower weight on the complexity penalty. As for our loss penalty, we chose 'l2', also known as ridge regression, because after extensive testing, we determined that it had the highest accuracy out of other metrics for penalizing loss.

This proposed model, utilizing logistic regression with various features including numerical data from the dataset, features generated from sentiment analysis, and a TF-IDF matrix based on the combined pros and cons section of every review, succeeded compared to other models because it was more sophisticated and factors in more features that are relevant to what we are trying to predict. Other models were not as good as the proposed model because they were either a) too generic or b) aren't easily manipulated such that they can be fine tuned.

One possible thing to look into in the future is to modify our models to take into account certain ratings that are "easier" to predict compared to others. For example, our final model has the hardest time predicting the rating "2". Two is an uncommon rating because it's more likely that people who felt extremely negative or extremely positive about their job would rate their job. People who had more neutral feelings about their job are less likely to leave a review, leading to overall less ratings with the rating two. Therefore, our future model could find ways that determine what characteristics in the review can be used to predict more neutral ratings like two. We could use oversampling or undersampling methods to help lessen the imbalance in the future, but this of course has its own drawbacks.

References

[1] B. Shiv Kumar. 2021. Predicting the ratings of reviews of a hotel using Machine Learning.
Retrieved from:
https://medium.com/analytics-vidhya/predicting-the-ratings-of-reviews-of-a-hotel-using-machine-learning-bd756e6a9b9b

[2] Yereya Berdugo. 2019. Review Rating Prediction: A Combined Approach.
Retrieved from:
https://towardsdatascience.com/review-rating-prediction-a-combined-approach-538c617c495c