IIT MADRAS

MS4110 : Data Analytics Project Report

Abhishek Rao(EE15B069), Divya K Raman(EE15B085), Kaushik Raghavan(EE15B105),
Sahana Ramnath(EE15B109), Umang Sinha(EE15B133), Vaibhav Nayel(EE15B117)
April 19, 2019

1 PROJECT STATEMENT

In this project, we aim to predict if an employee is still working for a company or not. Various indicators like happiness level of employees over time, how they react to comments made by other employees, the comments that they make, average satisfaction level of employees belonging to the company, proportion of good and bad comments liked by the employees, etc are used to arrive at this conclusion.
This project was implemented in Python, using the library scikit-learn.

2 THE DATASET

For training, we are given 4 files.

The file comments_employee_mapping.csv maps each comment to its owner and also specifies the date when the comment was made.

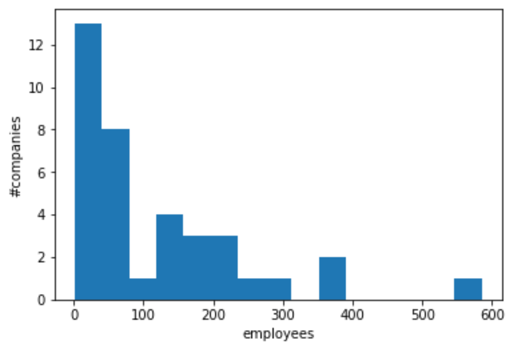
The comments_likeability.csv file has details about the reaction of employees to comments.

The file happiness_level.csv contains details about the happiness vote given by each employee on various days.

The employee_attrition.csv file maps each employee to the number of votes, last participation date and labels if the employee still exists in the company or not.

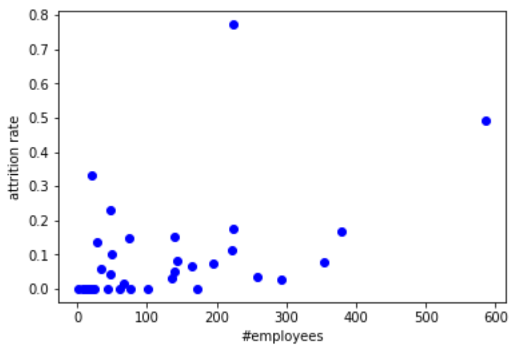
3 ANALYSIS OF THE DATASET

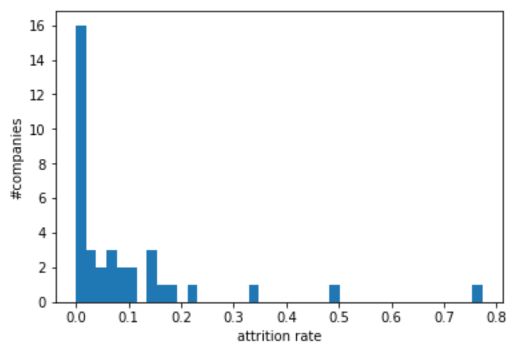
The dataset has 37 unique companies. Files employee_attrition.csv and happiness_level.csv contain all the companies but there are 3 companies whose employees don't like any comments and 1 company whose employees don't make any comments. This necessitates for our model to be able to handle inputs which have no likes or comments. Total employees in the dataset is 4418.Therefore, the average number of employees per company is 119.405405405.
The below histogram depicts the number of employees that each company has :



Split of employees overall, still working and those who left:

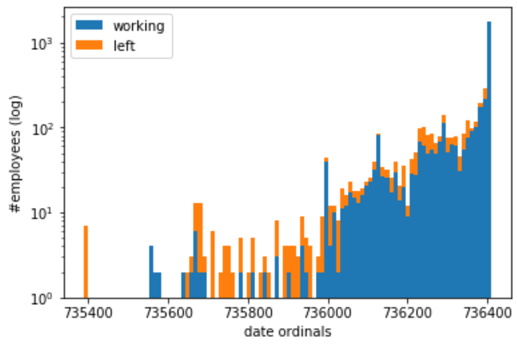
- still working: 3673
- left: 745

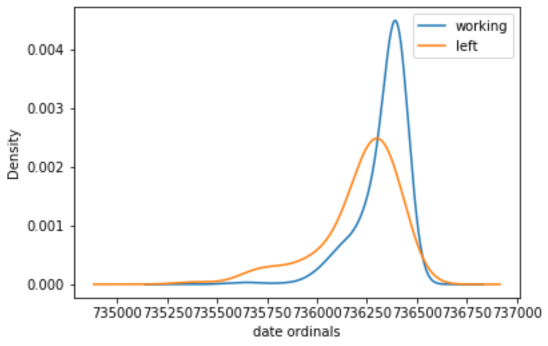




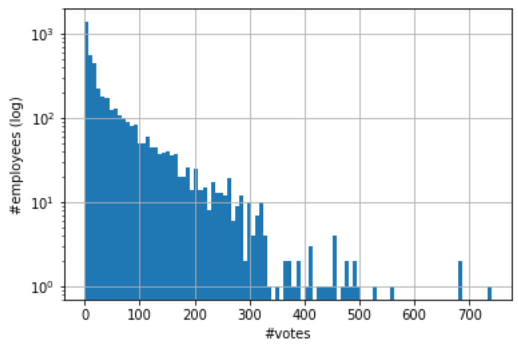
Correlation Coefficient: 0.47252133218050585
A correlation coefficient around 0.5 suggests a correlation of company size with attrition rate.

Relation between the last date of participation and likelihood of leaving :

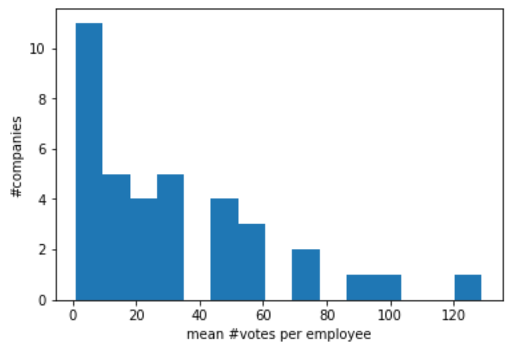


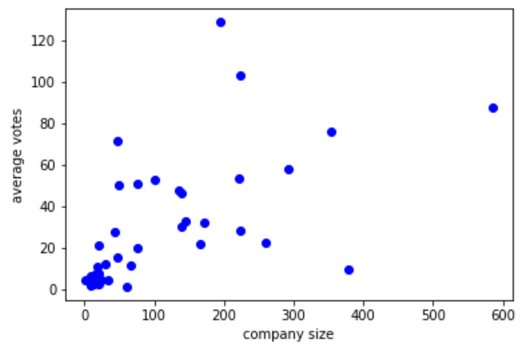


Clearly, the further back the last date of participation is, the more likely the employee is to have left.
Number of votes made by each employee :



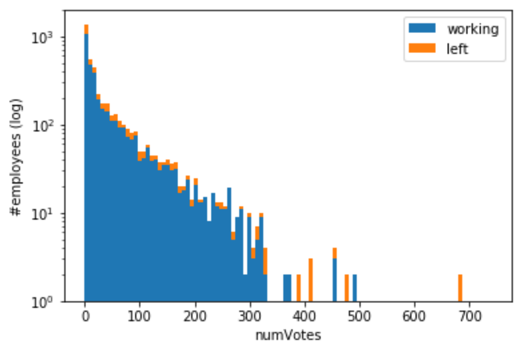
Most employees have very low vote counts.
Are the average number of votes dependent on the company? Correlation Coefficient : 0.5765644245292082

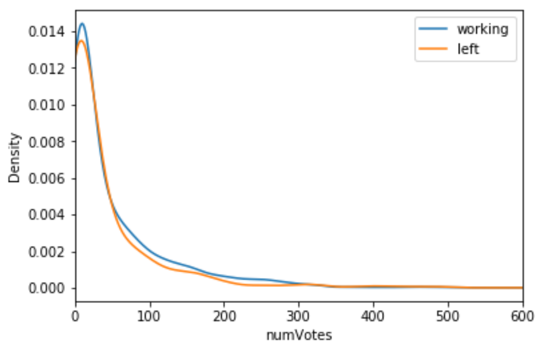




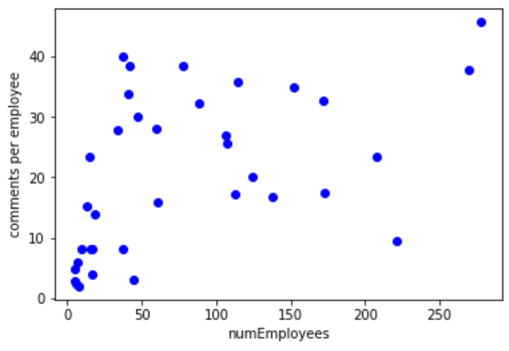
Some companies witness lower participation than other companies. This could be because employees in bigger companies are more likely to socialise or because some companies are very new and the numbers haven't had a chance to become higher.

Relation between number of votes and likelihood of leaving the company:

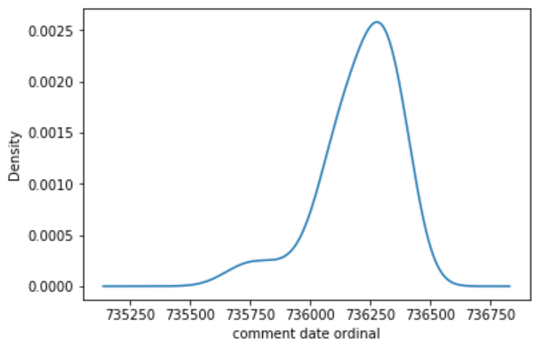




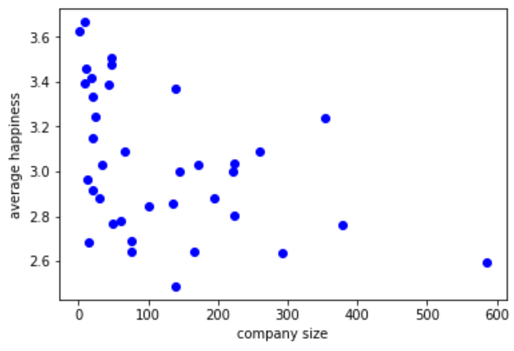
The number of votes by an employee doesn't seem to indicate anything about likelihood of leaving,i.e, unhappy employees are just as likely to vote as happy ones.
Comments per employee:



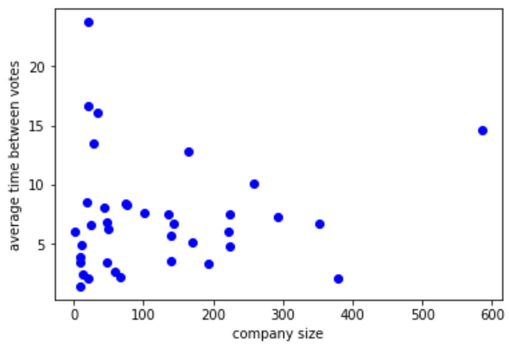
There is a clear correlation between company size and comments made per employee



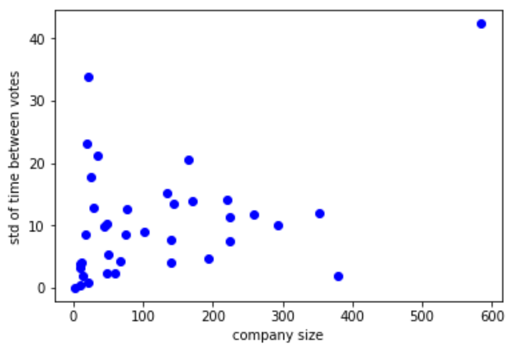
This seems to mirror the last participation date density. The number of comments seems to mirror the number of workers in the system at any time.
Variation of happiness with company : Correlation Coefficient - -0.4302256906883859



The plot shows a distinct negative correlation of company size with employee happiness.
Time between votes: Correlation Coefficient - 0.06228331496742335



Number of employees who voted only once: 458



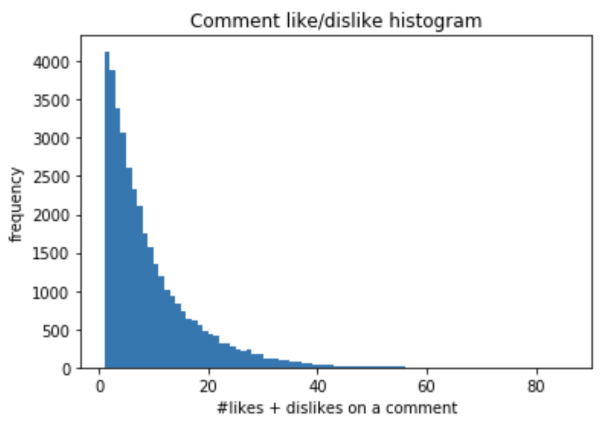
A low correlation suggests that company size does not seem to affect the average time or variance of times between votes.

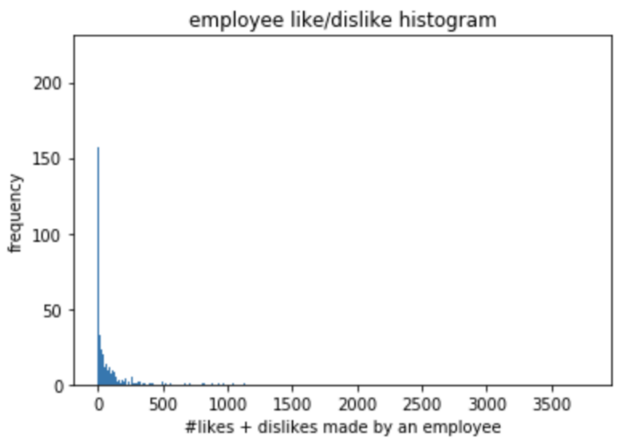
4 Preprocessing and Feature Engineering

4.1 Comment Matrices

In order to consider the comments made and liked by every employee, we created 2 matrices to track interactions between each employee and each comment.

- Like matrix: This sparse matrix has a 1 corresponding to likes and -1 corresponding to dislikes by each comment and a 0 everywhere. The matrix has around 37k columns and limits the types of models which can be used. Using very powerful models like neural networks and random forests results in a high degree of overfitting

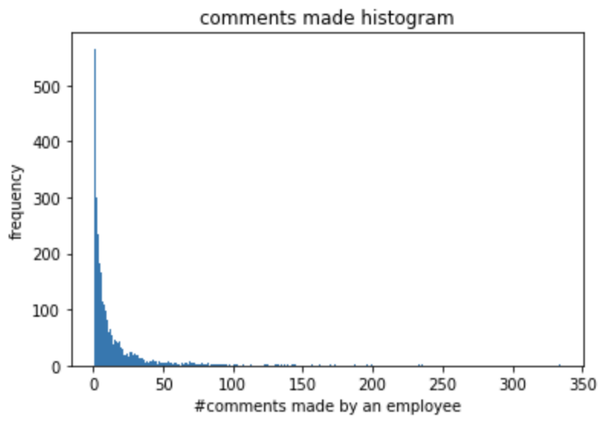
  We initially discarded any comments with zero likes and dislikes. The employee like histogram indicates that most employees don't interact with comments at all and the likes are coming from only a few employees. Notably, about 1200 people didn't like or dislike any comments, indicating that a little over a third of the rows of this matrix contain only zeros. This necessitates further feature engineering.
  We experimented with a few versions of this matrix.

  – Goodness score weighting: We multiplied each comment's occurrence with the calculated goodness score and ran a model on this new matrix hoping that this additional information would help. However this only amounts to a linear transformation of the data which is easily unlearned by a model as simple as logistic regression. This did not result in performance gains.

  – Comment popularity thresholding: We applied a filter on the matrix to reduce its dimensionality. We considered only the comments which had more than a certain number of likes+dislikes. We set thresholds at 2,5,10 and 50. none of these resulted in performance gains.

  – Dimensionality reduction: methods such as PCAand SVD were also tried. These are detailed in section 4.4.

- Comment matrix: This matrix is built similarly to the previous matrix except that it only contains a 1 where an employee made a comment. Obviously, the columns of this matrix sum to 1 since only 1 employee could have made a particular comment. 1500 People never made any comments.



We ignored this feature in the final model because since each column only has a single entry, it leads to overfitting possibilities even for a simple model.

4.2 Comment Goodness Score and Features

Based on the likes and dislikes of comments by the various employees, a comment *goodness* score was calculated and four new features were introduced for each employee based on how much he/she has liked/disliked good/bad comments.

The goodness score was calculated as follows : For every unique comment, if an employee has liked and stayed on at the company, or disliked and left the company, or if the employee who made the comment has stayed on,the comment's goodness feature is increased by 1(indicating a good comment about the company). If an employee has disliked and stayed on, or has liked and left the company, or if the employee who made the comment left the company,the comment's goodness score is decreased by 1(indicating a bad comment about the company).This score is then normalized by the total number of employees who have reacted to the comment. The final normalized score is thresholded to discriminate between good and bad comments.
Using the comment goodness feature described above, we create 4 more features for each employee. Two features describe the number of good and bad comments made by each employee and two features describe the number of good and bad comments liked(and disliked) by each employee.

4.3 Happiness Feature

The happiness feature is generated from the happinessLevel.csv file. We associate each employee with a happiness score based on their votes over time. Date time stamps are mapped to an ordinal scale considering the time stamp at which the first vote was made as the epoch. The ordinals are then scaled to take values between 0 and 1. For each employee, the happiness score is calculated as sum(vote*exp(time ordinal at which the vote was made) over all happiness votes given by the employee.

4.4 Dimensionality Reduction

Since the comment like matrix had over 37k columns, we tried to reduce the feature space in order to use more diverse models.

- PCA: Principal Component Analysis with 50,100,200,500 and 1000 features was attempted but all of these resulted in at least a small performance dip even though over 90% of the variance was explained by the first 2 components.
- SVD: Singular Value Decomposition reconstruction was used to de-sparsify the matrix so that models could learn to use each column in a more meaningful way. This is a common technique in NLP to de-sparsify term frequency matrices. However, in our case we didn't observe any performance gains.

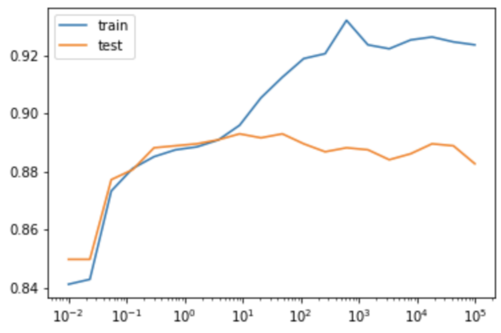
5 Experimentation and Results

In this section, we detail the different models we used to predict outcomes. All results were obtained using a 20% train-test split using the following features:

- Like matrix
- Company Alias with one-hot encoding
- Last date ordinal from the attrition file

Models were fitted on all previously mentioned feature-reduced data sets:

- **Logistic Regression**: L1 and L2 regularized logistic regression is the first model we tried.



The x-axis here denotes the regularization penalty inverse on L-2 regression. The best performance achieved was 89

- **Neural Network**: A shallow neural network was used with different activations, optimizers and learning rates but at best, it achieved the same performance as logistic regression.
- **Random Forests**: We next attempt to use random forest classifier to fit the model. It gives us an accuracy of around 83 percent. Parameters tuned were n_estimators and max_depth.
- **Decision Trees**: Resulted in a high degree of overfitting due to the large feature space. The train accuracy was 99.66 percent and test accuracy was 84 percent.
- **AdaBoost**: This is a boosting algorithm that uses simple base classifiers and combines their decision boundaries into a complex surface. We used decision stumps and highly regularized logistic regression as the base classifiers with 20,50,100 and 200 estimators. The best performance reached was 88% on both train and test
- **Bagging Classifier**: Bagging creates many datasets by sampling from the full dataset with replacement, trains simple models on each of these and combines their decision boundaries. This did not perform as well as AdaBoost and topped off at 85%
- **SVM**: Extensive grid search was conducted to find the best set of hyper-parameters. SVM classifier with rbf kernel having mis-classification penalty $C = 2500$ and $\gamma = 10^{-4}$ was able to produce a more generalized fitting on the dataset. The model was trained with dataset whose dimensions were reduced using PCA to only 50 components. The average classification accuracy on training and validation set were very close: 88.38% on training set and 88.34% on validation set.

KNN was not tried since these perform poorly in sparse and large feature spaces. Our final model uses logistic regression with C=1.0

6 Code

We have submitted all codes and the feature vectors in npy files or pickled format.

- EDA.ipynb : Does an analysis of the given dataset.
- HappinessFeature.ipynb : Computes the happiness feature as described above.
- Preprocessing.ipynb : Processes the data from the csv files into suitable formats.
- main.ipynb : Has the code for computing the comment goodness score and finding the number of good and bad comments made and commented by each employee.
- Models.ipynb : Has the code for various models that we trained to classify whether an employee is still present with the company or not.
- Full Model.ipynb : The test file that can be used for testing the model on new data.
- The npy and pkl files have the features created using the methods described in the previous section.

Add and modify this section as if more code is pushed to github.

7 Usage of Test Script

Full Model.ipynb is the test script to be used for testing the model on new data. The script is easy to use, the cells just have to be run in order after adding the names of the test feature files to the list 'testfiles .(Instructions and comments are given in the file as well).

The preprocessing is done for the test features in the same manner and order as for the train features.

The model is then trained again using the train set and then the accuracy on the test set is printed.

8 Conclusion

In this project, multiple models were tested on the employees dataset to predict if an employee will stay in the company or not. We use various feature engineering and pre processing techniques to process the given data. We also use dimensioanlity reduction techniques. We then test our features on various models and apply hyperparameter tuning to get the best results.