GENIUS VOTES PREDICTOR

EECS349 Machine Learning Final Report

Gabe Rojas-Westall: Conducted all data collection and dataset creation from Genius Client
Chaitra Subramaniam: Tested and analyzed results of all regression and classification models, created the website
Michael Shane-Smith: Tested and analyzed results of nearest neighbor model, performed feature engineering on dataset, and modified target output variable for classification task

Task

The goal of our project is to find the best regression task to predict the total number of votes (upvotes minus downvotes) that an annotation receives on Genius.com based on attributes such as number of followers, length of the referent and number of contributors. Each annotation is linked to a referent or a specific chunk of text in the song so it provides really detailed analysis. Additionally, we wanted to find the best classification task that would help us classify the annotation as good or bad (depending on total number of votes).

As users, it would be helpful to know the number of upvotes we would get on an annotation post to and see whether we are actually improving it or not. If this machine learning project is applied, we can build a simple application where users type in their annotation and they get a response saying “Yes, good post - it will increase votes by x” or “No - give it another shot”. It will help improve users annotation skills or improve overall usability of this website.

Dataset

We wrote python scripts to make requests to the Genius’ API, clean the data, account for missing attributes, and wrote it to a csv file using pandas. In order to limit the data we requested, we limited our search to the US top 10 rap artists (from Billboard/Apple charts). Our scripts made a request to get the top 30 songs from each artist that we chose, where each song has a different number of referents, giving us a total N = 2752 referents. We have 19 different attributes per referent, some of which include number of comments, number of contributors, and length of referent. Our target variable is stored as total votes (upvotes - downvotes). For our classification tasks, we converted the total votes target to Good or Bad depending on whether the votes for a specific annotation were higher than the median number of total votes or not. Once we got a dataset, we split it 70/30 (a standard convention) and are storing this testing data for later use. Among the remaining development data, we used 10 fold cross validation on our training data to build our model.

Initial Data Analysis

In Image 1 (Appendix), our total votes distribution is right skewed so we could try to make it more uniform by taking the log of the total votes. From our correlation heat map (Image 2), the red colors means that the features are less correlated as the coefficient is closer to 0. If we look at simply the last row or last column, we can see that all the variables have very little relationship (low correlation) with total votes on first glance since they’re all purple so one predictor variable may not be driving the relationship necessarily. Also, most of the chart is dark indicating that there is little correlation within the predictor variables themselves. This is a good thing as it shows little multicollinearity indicating that this is not too much of a serious problem in our analysis that might might skew the validity of our results.

Evaluation Metrics: What does “best” mean?

For the regression task, the evaluation metric we chose to focus on was the Pearson correlation coefficient. Primarily, the coefficient can tell you the strength of the relationship between our features and the total votes. The closer that they are to +1 or -1, the stronger the model is is and

if it is closer to 0, it means there is an extremely weak relationship. The sign of the coefficient can also give us intuition on whether the relationship is positive or negative as well.

For the classification task, we are using accuracy as our primary evaluation metric. we used accuracy to tell us how many our model classified right (as Good or Bad) out of the total. Accuracy ranges from 0 - 100% and a higher accuracy could imply a better model. At the same time, we looked at root mean square error for both regression and classification to test how much our fitted/predicted values differ from the observed values.

Part 1: Finding the Best Regression Model

Method

The first model that we tried was a multiple linear regression with Weka. Multiple linear regression attempts to model the the relationship between two or more variables by fitting a linear equation to the data in order to predict an independent variable.

For our linear regression models, we ran two attribute selection methods using Weka. We tested both attribute selection methods, M5 and Greedy. The M5 method uses a separate and conquer strategy where it builds numerous trees with each leaf linking to a multivariate model. It then selects the attributes that result in the model with the highest Akaike coefficient which measures “relative goodness of fit”. Then, we ran the same regression again but this time with the greedy attribute selections method which tries all possible subsets of features and chooses the one which minimizes mean squared errors. We kept in mind though that the greedy selection method is not ideal as it is extremely time consuming and not an efficient selection method. We also tried nearest neighbor and we get the output and the ground truth number of votes and we calculate the same correlation coefficient so they become comparable. Finally, we also built a model where the target output was logged to see if this lead to improvements.

We also performed statistical analysis after running the linear regression models and chose the features that proved to be statistically significant (those whose coefficient p-value was greater than 0.05, a standard statistical metric used for comparison). We then selected only the top 10 features such as length of referent or number of verified contributors and reran our regression model and the results are listed below:

RESULTS (Regression)

Model	Correlation Coeff. (R^2)	Root MSE
Linear Reg M5	0.82	106.3743
Linear Reg Greedy	0.83	106.3382
Linear Reg Selected Features	0.79	114.5845
Log Linear M5	0.85	0.4079
3-Nearest Neighbor	0.7941	76.0591

The best method for linear regression seems to be Linear Reg M5 with the logged output since correlation coefficient is 0.03 higher than regular Linear reg with M5. M5 also performs almost as well as the Greedy but is more time and computationally efficient. It also performs better than the best nearest neighbor model we found (3-NN). The Root MSE's seem high mostly because the slightly abnormal data distributions of total votes were far apart so the errors were high. Additionally, it is important to note that the selected features on their own gives a relatively high correlation coefficient of 0.79 which might make more sense to use if your goal is to limit the number of features in your dataset.

Part 2: Finding the Best Classification Model

Methods

We then chose to compare decision trees, logistic regression, and random forests. We first calculated ZeroR to get a baseline accuracy of about 50%. We then used the Decision Tree J48 algorithm on Weka, turning pruning on to reduce overfitting. After that, we tried random forests which can reduce overfitting and also reduce variance as it will not rely on the bad trees that might perform badly on the training data. Finally, we chose to look at logistic regression because they are simpler than decision trees as they work well with a single decision boundary and are less likely to run into overfitting. We chose some important features visually from our decision tree which got the highest information gain and interestingly they were almost the same features we found statistically significant in the linear regression stats but including number of followers was also significant in the decision tree so we ran a model with 10 features found earlier plus this feature. All these models give us fast evaluation times which is crucial if we want to develop an user based application where they can immediately get a prediction of whether their annotation is good or bad. We also tested K - nearest neighbor but no model performed as well as the results below.

RESULTS: Classification

Model	**Accuracy**	**Root MSE**
Zero R	49.12%	0.5008
Decision Tree (J48 with Pruning)	87.02%	0.3193
Random Forest	84.30%	0.3200
Logistic Regression	84.29%	0.3254
Decision Tree (Select Features)	82.67%	0.3605
Random Forest (Select Features)	80.63%	0.3665
Logistic Reg (Select Features)	84.92%	0.3412

Our goal was to achieve highest accuracy and lowest root mse which is why decision trees seemed to work the best. They improve ZeroR baseline accuracy by 30-40% and don't run into too much of an overfitting problem and even perform better than logistic regression and random forests. It is also good to note that our simplified model achieved pretty high accuracy of 83% for decision trees, validating our feature selection process. This shows that if we wanted a simplistic model as well, the best model for classification would be decision trees with only the selected features used.

Conclusion

In conclusion, for our regression tasks, we would use Linear Regression with the M5 selection method with the logged target output as this gives us the highest correlation coefficient and a low mean squared error. For classification, we would want to use the Decision Tree J48 model with pruning as this gives us the highest accuracy and lowest squared error. But in both cases, a more simplified model with almost half the features performs almost as well indicating that if our purpose was to find the most simple model, we could quite safely choose that model.

Limitations and Future Goals

Even though we achieved high performing models, there are many more methods we can try. A huge focus of Genius is on lyrics but apart from looking at length of the comment/lyric, we did not analyse sentiment or type of words used. We think it would've been really interesting fix that likely using the bag of words method and assigning different values to different words for both the lyrics and annotations. We could then look into what words or kind of words in an annotation are associated with more votes given the lyrics of a song.

We were also proud that our feature selection lead to significant improvements but we can still continue to narrow down our search using other methods such as other forms of regressions such as polynomial, kernel, or local linear regression to find non linear fits of our features on total votes. This can help us achieve a clear idea of what the real 5-6 driving variables were instead of narrowing our 19 features only down to 11. We could also include interaction terms of all the features as additional features to our regression models to see if this would improve correlation as well.

Finally, we have developed models but we have not worked on adapting it to an application for users yet. We hope to develop this as an application with an attractive front end where users can plug in an annotation and immediately find how many votes they will get which can even be an extension on the Genius page as well for convenience purposes.

Appendix

Image 1: Comparing Total Votes to Log Total Votes Distribution

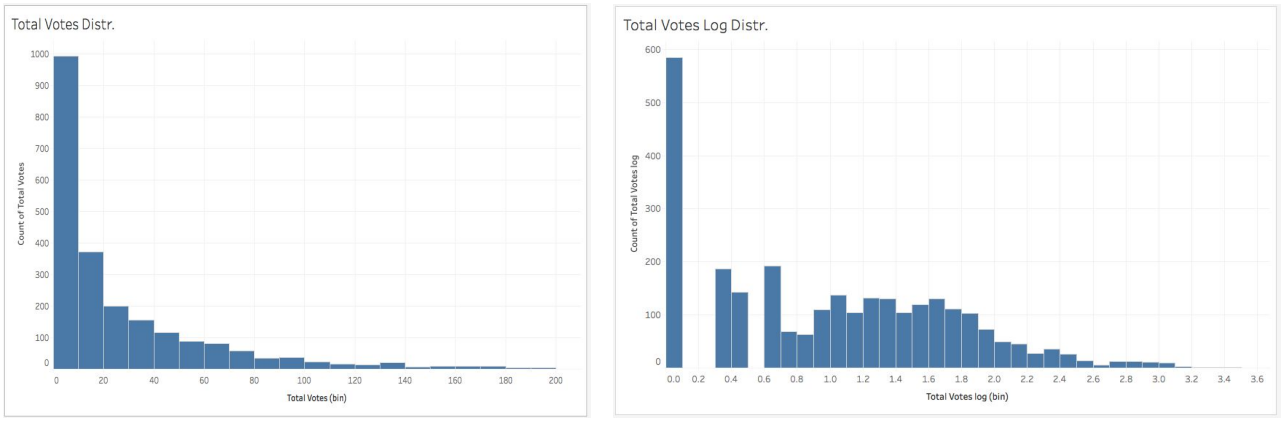

Image 2: Correlation Heatmap