Feature Engineering Homework

James Foster, Spring, 2017

Methodology

For each set of features explored, I limited `m` by randomly sampling 50% of the train set. I then split the remaining train data into training and validation sets by randomly assigning the it into 80% train and 20% validation. For promising feature sets, I got a more precise measure of accuracy by using the full original training set with 5-fold cross-validation. I kept records of the features and parameters used, along with their training and validation scores.

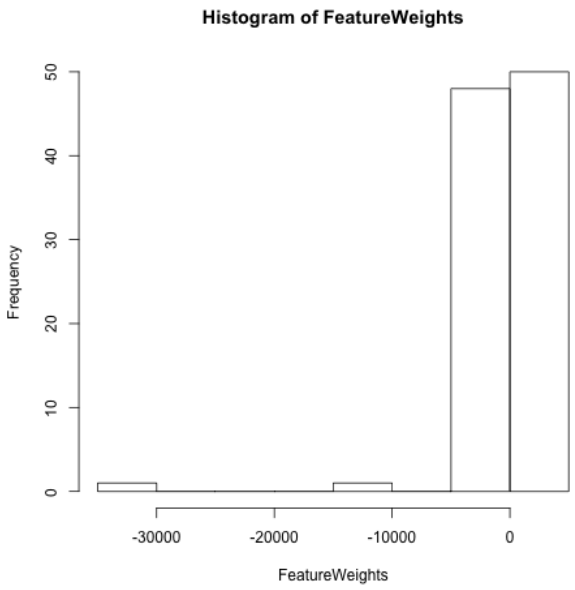
My methodology was start with the baseline model, assess the overall accuracy, recall, and precision, inspect the most influential features, and inspect the observations that were incorrectly labeled. Based on false positives and false negatives, I tried to design features to reduce those errors.

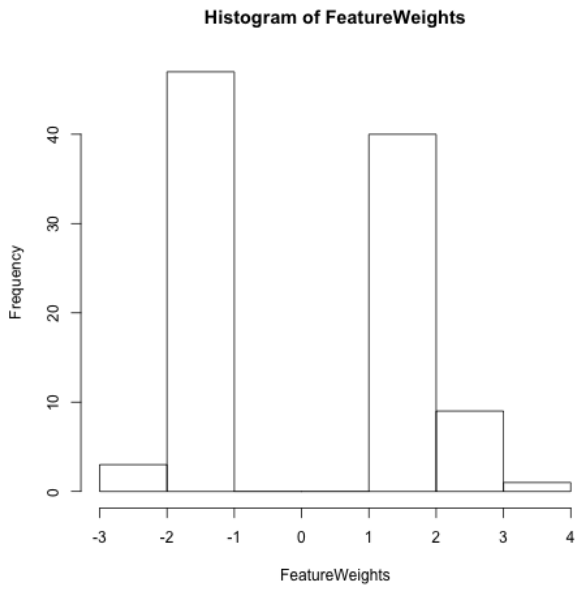
Features Explored

- CountVectorizer, stop_words, strip_accents
- TfidfVectorizer, stop_words, strip_accents
- Removing character names
- trope
- page
- imdb features: genre, year, votes, rating, number of seasons
- polynomials/interactions
- word2vec w/ MeanEmbeddingVector
- ManualWordsFeatures
- Lemmatizers, Tokenizers
- SpellingCorrection
- SelectFromModel using l1 (lasso) regularization
- Restricting to top K features
- feature value standardization
- Principle Component Analysis/SVD
- missing value imputation

Example transformation: Standardization

When I added the imdb features year, votes, rating, and number of seasons, accuracy decreased from 91 train / 71 validation to 47 train / 48 validation. Inspecting the best feature weights, I saw that the best features had extreme weights and were dominating the classifier even with the l2 (ridge) regularization. To combat this, I scaled all feature values using StandardScaler() and MinMaxScaler(). MinMaxScaler ended up producing better accuracy so I retained it. (See histogram Figures below.)





Final Model

The most accurate model I found used the following features/transformations:

- Sentence: TfidfVectorizer with strip_accents = “ascii” and lowercase=True
- Trope: One-Hot Encoding
- Genre: One-Hot Encoding
- Page: I used Page to look up the following info in IMDb: year, rating, votes, number of seasons.
- Manual Words: I created a feature that counted the number of appearances of any words in the following list, which I created by hand through introspection and looking at the most frequent words in the spoiler examples: [‘dead’, ‘die’, ‘died’, ‘dies’, ‘dying’, ‘death’, ‘kill’, ‘killed’, ‘suicide’, ‘shot’, ‘staked’, ‘stabbed’, ‘finale’, ‘survived’, ‘survive’, ‘stab’, ‘revealed’, ‘realize’, ‘realizes’, ‘reveal’, ‘realized’, ‘ends’, ‘crucify’, ‘averted’, ‘averts’, ‘avert’, ‘finds’, ‘finally’, ‘final’, ‘alive’, ‘murder’, ‘murdered’, ‘murderer’, ‘killer’, ‘learns’, ‘married’, ‘marries’, ‘wedding’, ‘realizes’, ‘actually’, ‘!’, ‘pregnant’, ‘end’, ‘ending’, ‘killing’, ‘kills’, ‘eventually’, ‘reason’, ‘discover’, ‘discovered’, ‘big’, ‘averted’, ‘bomb’, ‘shoot’, ‘truth’, ‘ultimately’, ‘causes’, ‘affair’, ‘captured’, ‘results’, ‘fired’]
- All features were scaled column-wize with MinMaxScaler from the Scikit-learn library.

Final Performance

- Public Leaderboard: 11th Place, 0.73919 Score
- Private Leaderboard: 4th Place, 0.77500 Score