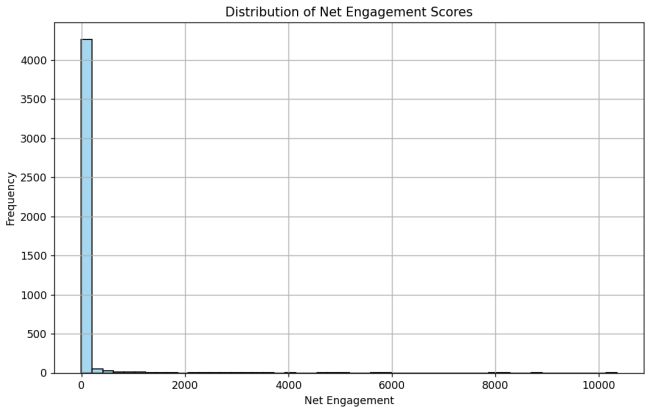
Shankar Veludandi
Social Computing
04/18/2024
Homework 3 Report

1. **Feature Construction**: In constructing features for gender classification from social media profiles, I extracted various linguistic and statistical features from the text data. The selected features included unigrams and bigrams to capture the most frequent terms and term pairs. I utilized TF-IDF to weigh the importance of terms within tweets and across the corpus, considering the frequency of a term in a tweet and its inverse frequency across all tweets, which helps emphasize terms more descriptive of the content. I also used the Sentiment Intensity Analyzer from the NLTK library to obtain sentiment scores of tweets, hypothesizing that sentiment could correlate with gender expression. Additionally, I implemented Part-of-Speech tagging to understand the grammatical structure of tweets, which could indicate gender-specific communication styles. Moreover, the length of words and the complexity of sentences were included, assuming different genders might use language complexity differently. Spacy's linguistic features were employed to assess sentence structure complexity, such as the number of subordinate clauses and passive voice.
2. **Description of the classifier**: I employed a Support Vector Machine (SVM) for our classification task due to its effectiveness in high-dimensional spaces and its ability to use a non-linear kernel to capture complex relationships between features. SVM is particularly well-suited for text classification problems with large and sparse feature space. Compared to other classifiers like k-Nearest Neighbors or Naïve Bayes, SVM performs better when there is a clear margin of separation in the feature space, which I aimed to achieve through feature engineering.
3. **Evaluation technique**: I employed a binary classification approach using a Support Vector Machine (SVM) model to evaluate the classifier's performance in distinguishing between high and low-engagement posts. The posts were classified into two categories based on their net engagement scores: those above the median were labeled as high engagement (1), and those below the median were labeled as low engagement (0). This binary labeling was derived from the continuous variable of net engagement scores, providing a clear target for our classifier. The `train_test_split` function from `sklearn.model_selection` is used for data partitioning, a utility function to split data arrays into two subsets: for training data and testing data. 20% of the data was reserved for testing and 80% for training. The `classification_report` from scikit-learn provides these metrics for both classes and support, which is the number of actual occurrences for each class in the specified dataset. The `confusion_matrix` also provides insights into the true negatives, false positives, false negatives, and true positives. For performance metrics, I focused on the following:
    a. **Accuracy**: This metric represents the overall correctness of the model and is calculated as the number of correct predictions divided by the total number of predictions. It gives us a quick indication of how often the model is correct. The model's overall accuracy is 0.82, which means it correctly predicts the engagement level of a post 82% of the time.

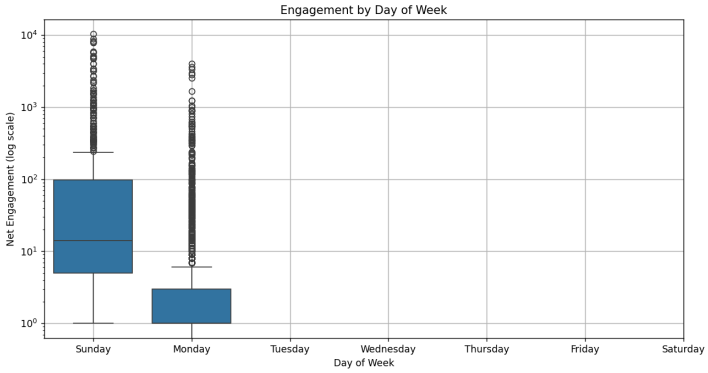
b. **Precision**: Precision for each class indicates the ratio of true positives to the combined total of true positives and false positives. It measures the model's accuracy in classifying a post as high or low engagement. For class 0 (low engagement), the precision is 0.83, indicating that when the model predicts a post will have low engagement, it is correct about 83% of the time. For class 1 (high engagement), the precision is 0.80.
c. **Recall**: Recall for each class is the ratio of true positives to the number of positive cases. It tells us how well the model can find all the relevant high or low-engagement instances. The model has a high recall of 0.92 for class 0, meaning it correctly identifies 92% of actual low-engagement posts. The recall for class 1 is lower at 0.64.
d. **F1-Score**: The F1-score is the harmonic mean of precision and recall and is particularly useful when the class distribution is imbalanced. It accounts for both false positives and false negatives and measures the classifier's accuracy concerning a specific class. The F1-score balances precision and recall, with the model scoring 0.87 for class 0 and 0.71 for class 1.
e. **Confusion Matrix**: The matrix shows the number of true positives, false positives, true negatives, and false negatives. With 527 true negatives and 197 true positives, the model seems better at identifying low-engagement posts than high-engagement ones.

4. **Implementation**
    a. **Data Preprocessing**: I cleaned the text data by removing stopwords, using tokenization, and excluding non-alphanumeric characters, which was facilitated by the `nltk` and `re` libraries.
    b. **Feature Extraction**: Feature extraction was done using the `TfidfVectorizer` from the `scikit-learn` library for unigrams and bigrams. The `SentimentIntensityAnalyzer` from `nltk` was employed for sentiment analysis.
    c. **Classifier Implementation**: I chose the `SVC` class from `scikit-learn` for the SVM implementation. To fine-tune the model, I experimented with different kernels and regularization parameters using `GridSearchCV.`
    d. **Dataset Partitioning**: The `train_test_split` function from `sklearn.model_selection` is used, a utility function that splits data arrays into two subsets: for training data and for testing data. Twenty percent of the data was reserved for testing, and eighty percent was reserved for training.
    e. **Performance Metrics Calculation**: I calculated and presented our evaluation metrics using the `classification_report` and `confusion_matrix` functions from `scikit-learn`.
5. **Visualizations**:
    a. **Distribution of Net Engagement Scores**: The histogram shows the frequency distribution of net engagement scores across posts. Many posts have low engagement scores, with a few outliers reaching very high engagement.



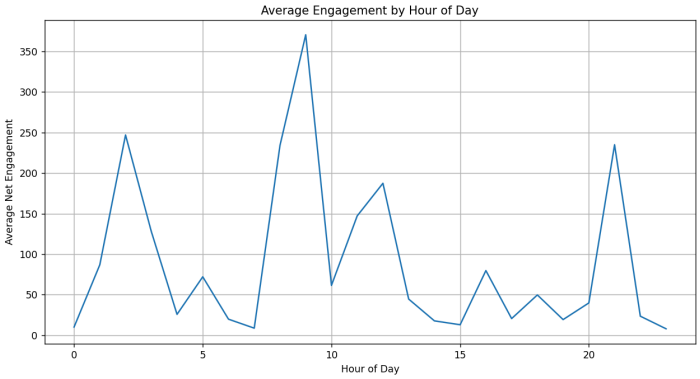
b.

c. **Engagement by Day of Week (Log Scale)**: The boxplot displays the distribution of engagement scores by day of the week on a logarithmic scale to handle the wide range of values. This reveals any potential patterns on which days posts tend to receive higher engagement.



d.

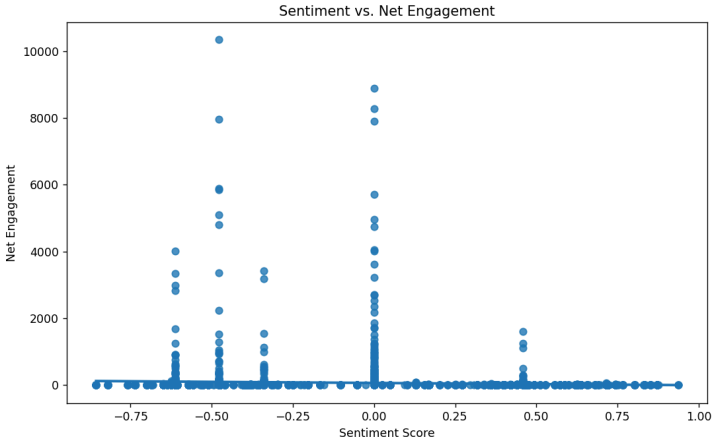
e. **Average Engagement by Hour of Day**: The line graph shows the average engagement scores by the hour of the day, offering insights into when users are most active and likely to engage with content on r/AskReddit.



f.

g. **Sentiment vs. Net Engagement**: The scatter plot with a regression line shows the relationship between the sentiment of the post and the net engagement it receives.

It can indicate if more positively or negatively worded questions lead to higher engagement.

h. 

6. **Analysis of results**: Utilizing a Support Vector Machine (SVM) classifier, the study aimed to categorize r/AskReddit posts into high and low engagement based on net upvotes. The classifier achieved an accuracy of 82%, indicating a robust predictive capability. It exhibited a precision of 80% for high-engagement posts. It was correct in 80% of its high engagement predictions and a recall of 64%, showing it identified 64% of all actual high engagement instances. The F1-score, a balance between precision and recall, was 0.71 for these posts. The confusion matrix revealed that the model correctly identified 527 low-engagement posts while misclassifying 48 as high engagement and correctly predicted 197 high-engagement posts with 111 misclassified as low engagement. Visual analyses enhanced these findings, depicting engagement trends across hours and days and examining correlations between post sentiment and engagement levels, enriching the understanding of post interactions on the platform and validating the research hypothesis on the influence of question phrasing on engagement.

	Accuracy	Precision	Recall	F1-Score
0 (low engagement)	0.82	0.83	0.92	0.87
1 (high engagement)	0.82	0.80	0.64	0.71

```
Best model parameters: {'C': 10, 'kernel': 'linear'}
              precision    recall  f1-score   support

           0       0.83      0.92      0.87       575
           1       0.80      0.64      0.71       308

    accuracy                           0.82       883
   macro avg       0.82      0.78      0.79       883
weighted avg       0.82      0.82      0.81       883

[[527  48]
 [111 197]]
```