Shankar Veludandi
Social Computing
03/21/2024
Homework 2 Report

1. **Feature Construction**: The gender classifier was built using textual and visual features from Twitter profiles. Features included the TF-IDF scores of unigrams from tweets and profile descriptions, capturing the relevance of specific words to gender identity. Visual cues were incorporated through the RGB values of sidebar and link colors, reflecting personalization choices potentially tied to gender. Gender confidence scores were also used to weight the classifier's decisions, prioritizing data points with higher certainty. These features were selected for their balance between direct language use and subtler, style-related gender indicators.
2. **Description of the classifier**: The classifier I chose is the Multinomial Naive Bayes (MultinomialNB). This classifier is particularly well-suited for text classification problems where features are typically represented as word vector counts (like TF-IDF vectors). It assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature (naive assumption), which works well for text data. Multinomial Naive Bayes is a good fit for this problem because: It effectively handles high-dimensional data, typical in natural language processing tasks, it's computationally efficient, making it suitable for a dataset with a large number of features, and it performs well with discrete features, such as word counts, which are common in text classification tasks. Compared to other classifiers, such as k-Nearest Neighbors (k-NN) or Support Vector Machines (SVM), Multinomial Naive Bayes tends to be faster and easier to implement, especially when dealing with large text data. While SVM can offer higher accuracy, it is also more computationally intensive and may require careful tuning of its parameters. On the other hand, k-NN may not perform as well with high-dimensional sparse data typical of text classification.
3. **Evaluation technique**: The evaluation of the classifier's ability to distinguish gender was conducted using k-fold cross-validation, which is a robust statistical method for assessing the performance of a model. This technique partitions the dataset into k equally sized subsets or 'folds.' In each of k iterations, a different fold is held out for testing while the model is trained on the remaining k-1 folds. This process is repeated until each fold has been used for testing exactly once. This technique is particularly effective because it ensures that every data point is used for both training and validation, and it mitigates the risk of the model's performance being dependent on a particular random split of the data. In this specific evaluation, a 5-fold cross-validation was implemented, meaning the dataset was split into five parts. The classifier, a Multinomial Naive Bayes model, was assessed on each fold with the following metrics:
    a. Accuracy: The proportion of correct predictions over all predictions. It provides a general indication of the model's ability to label the gender correctly.
    b. Precision (Weighted): It reflects the proportion of positive identifications that were actually correct, with weighting to account for class imbalance.
    c. Recall (Weighted): It measures the model's ability to find all relevant instances (all actual genders) in the dataset, with weighting to handle class imbalance.

4. **Implementation**
   a. **Data Preprocessing**: The data preprocessing involved several steps to ensure the text data was clean and structured for feature extraction. Using the Natural Language Toolkit (`nltk`), the text from tweets and descriptions was tokenized with `nltk.tokenize.word_tokenize`. Stopwords were removed using `nltk.corpus.stopwords` to filter out common words that are typically irrelevant to the analysis. Missing values in the text and description fields were filled with empty strings to avoid errors during tokenization.
   b. **Feature Extraction**:
      i. Text Features: The `TfidfVectorizer` from `sklearn.feature_extraction.text` was utilized to convert both the tweet texts (`processed_text_joined`) and user descriptions (`processed_description`) into TF-IDF features. This was done to reflect the importance of words in relation to the entire dataset. The `max_features` parameter was set to 1000 to focus on the most relevant features by limiting the feature set to the top 1000 terms, effectively filtering out less common unigrams.
      ii. Color Features: A custom function, `hex_to_rgb`, was written to transform hexadecimal color values from user profile sidebars and links into RGB color values. This function handled any invalid hexadecimal values by defaulting to (0, 0, 0) for sidebar colors and "FFFFFF" for link colors if the value did not match the standard length for a hexadecimal color code. The resulting RGB values were normalized by dividing each by 255 to ensure that the RGB features would be on a 0-1 scale. This normalization is standard practice for color values in machine learning to ensure they contribute appropriately to models that may be sensitive to the scale of the input data.
      iii. Gender Confidence: The `gender:confidence` feature was directly extracted from the dataset without the need for external libraries. This feature represents the confidence of the gender label and was used as provided, assuming it was already normalized between 0 and 1. No further normalization or standardization was applied to this feature.
   c. **Classifier Implementation**: The `MultinomialNB` classifier from `sklearn.naive_bayes` was used. Multinomial Naive Bayes is suitable for classification with discrete features (like word counts or frequencies). The classifier was used with its default parameters, as it often performs well without extensive parameter tuning on text classification tasks. Inputs to the classifier were the feature matrix `X` and the target vector `y`, while the output was the predicted gender labels.
   d. **Dataset Partitioning**: The dataset was partitioned for k-fold cross-validation using the `KFold` class from `sklearn.model_selection`, which is a standard procedure in machine learning for estimating the model's performance on unseen data. The chosen `k` value was 5, meaning that the dataset was split into 5 equal parts, or folds. During each iteration, one fold was reserved for testing the model, and the remaining four folds were used for training. This process was repeated 5 times, with each fold getting a chance to be the test set once. This approach

ensures that every data point contributes to both training and testing, providing a comprehensive evaluation of the classifier's performance.

   e. **Performance Metrics Calculation**: To calculate the performance metrics—accuracy, weighted precision, and weighted recall—the `cross_validate` function from `sklearn.model_selection` was used, which allows multiple scoring metrics to be evaluated. This function performs the k-fold cross-validation and computes the scores for each fold for the specified metrics. The `precision_score` and `recall_score` functions were used with the `average='weighted'` parameter to account for any class imbalances by considering the proportion of true instances for each label when calculating the average. This choice is especially important in datasets where some classes are underrepresented. The input to the `cross_validate` function was the feature matrix `X`, the target vector `y`, the classifier, and the scoring dictionary defining which metrics to compute. The output was a dictionary containing the scores for each metric across all folds, which were then averaged to get a single performance estimate for each metric.

5. **Analysis of results**: The performance of the Multinomial Naive Bayes classifier was assessed using k-fold cross-validation with k set to 5. The metrics reported were accuracy, precision (weighted), and recall (weighted), which take into account the potential class imbalance within the dataset. These metrics indicate the classifier's ability to correctly identify the gender based on the provided features, with the average accuracy across all folds at 64%. The average precision is slightly higher at 65%, suggesting that when the classifier predicts a certain class, it is correct about that prediction 65% of the time, on average. The average recall of 64% indicates that the classifier is able to find 64% of all positive samples for each class. Here are the reported metrics for each fold, as well as the average performance across all folds:

	Accuracy	Precision	Recall
Fold 1	0.64	0.64	0.64
Fold 2	0.66	0.65	0.66
Fold 3	0.67	0.67	0.67
Fold 4	0.65	0.66	0.65
Fold 5	0.62	0.64	0.62

Average Accuracy	0.65
Average Precision	0.65
Average Recall	0.65