Advanced Machine Learning

SENTIMENTAL ANALYSIS

SNAPPFOOD DATA

MEMBERS: Sara sadat Nasr, Mostafa Abdolmaleki, Amirhosein Shabani

ABSTRACT

Sentiment analysis poses a significant challenge for large enterprises, especially when dealing with user comments across various products. The ability to analyze these comments holds immense potential for business owners, enabling them to enhance employee satisfaction, adjust compensation structures, and guide team members effectively. While some advanced applications like Snapp may struggle to handle this task seamlessly, employing automated machine learning methods offers a superior solution, ensuring consistently high accuracy in sentiment analysis.

Challenges

Throughout this project, we encountered various challenges:

Language Specificity: A primary obstacle arose from the fact that the majority of NLP tools are designed specifically for the English language and its alphabet. This made handling Persian text a complex task, necessitating the use of specialized packages and libraries. As a viable solution, we employed Hazm and other related libraries, and in certain instances, manually managed special characters.

Data Size and Complexity: Another challenge emerged from the simplicity and limited size of our dataset. Its modest scale posed constraints on utilizing large and sophisticated language models. Additionally, sourcing similar datasets for augmentation proved to be a complicated endeavor. Consequently, we opted for simpler models that exhibited commendable performance under these constraints.

Fine-Tuning and Model Selection: Fine-tuning and identifying the optimal model presented another layer of complexity. The landscape of classic and simple neural network models offered numerous options. To navigate this, we systematically selected a subset and

conducted rigorous testing and trials to pinpoint the most effective approach.

DATASET

We utilized a dataset shared in a Telegram group, refraining from augmenting it with additional data. Through a meticulous process, we partitioned the dataset into training and validation sets. This dataset comprises Persian comments from SnappFood users, with a binary sentiment target—either [SAD] or [HAPPY]. We visualized the distribution of these sentiment classes, revealing insights into the dataset's composition. In total, the dataset encompasses 565,699 rows of information, organized into columns: [comment, label, label_id].

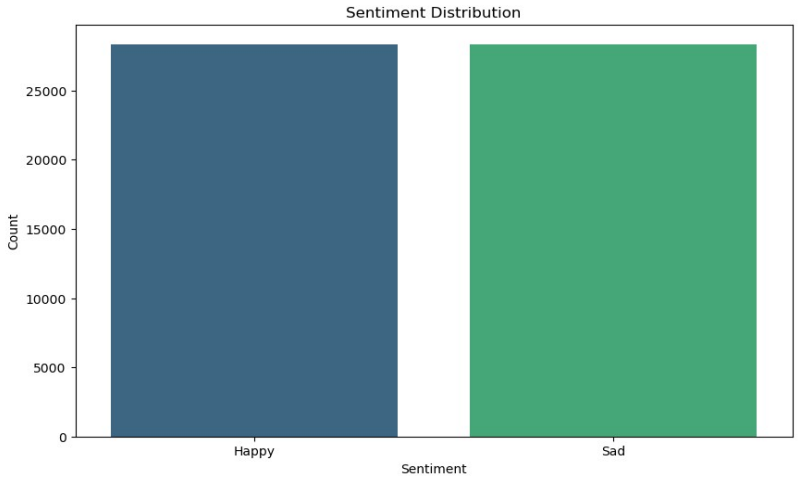

Fig.1) Sentiment Distribution target

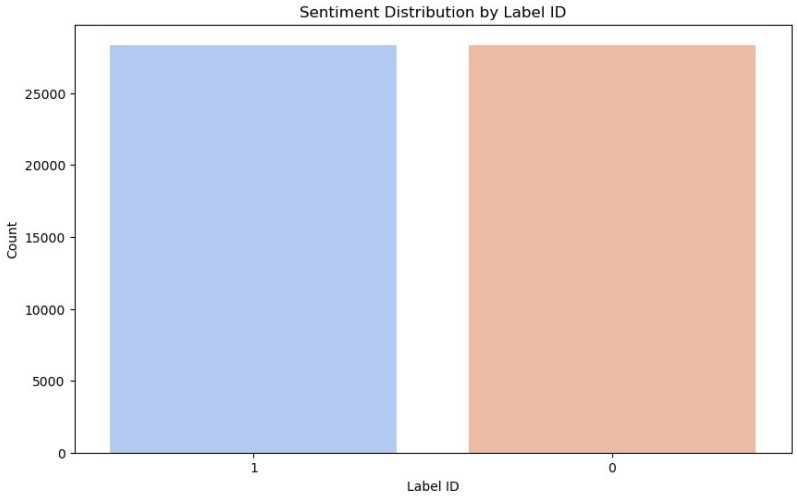

Fig.2) Sentiment Distribution by Label ID

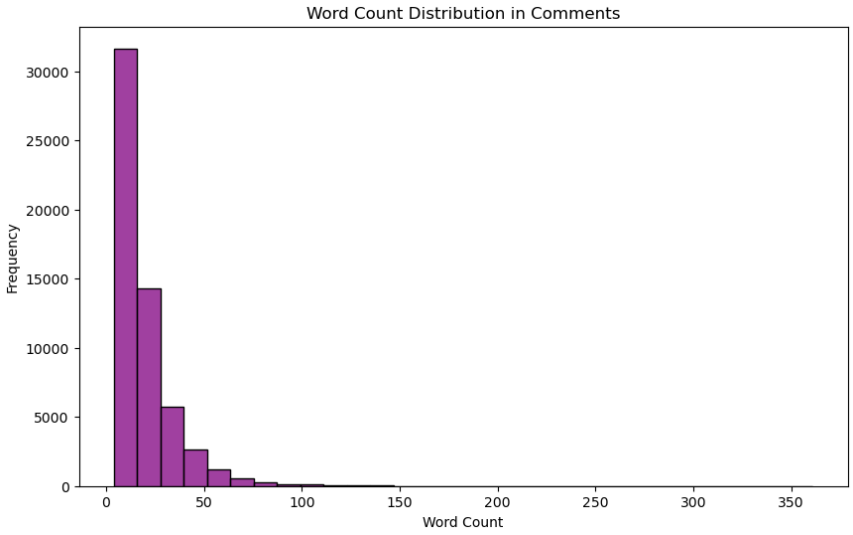

Fig.3) Word Count Distribution in Comments

PreProccesing

I In this section, we utilize a **Preprocessor** class containing key definitions. A pivotal private method, **_multiple_replace()**, employs regular expressions and a mapping dictionary to enhance the flexibility of text preprocessing.

The class integrates two crucial methods, **convert_fa_numbers()** and **convert_ar_characters()**, which contribute to the localization of numbers and characters. The former converts Persian (Farsi) numbers to Arabic numerals, while the latter transforms specific Arabic characters into their Persian (Farsi) counterparts.

The central method, **preprocess()**, oversees a series of operations to refine and standardize input text. This encompasses tasks such as replacing URLs and emojis with placeholders ("<URL>" and "<emoji>"), normalizing numeric representations, handling smiley patterns, lowercasing, removing leading/trailing whitespaces, substituting punctuation marks with spaces, consolidating consecutive spaces, and addressing repeating characters within words. Then, rows with missing values are dropped, and an unnamed column is removed.

The method concludes by evaluating if the processed text contains Persian (Farsi) characters; if affirmative, we return the cleaned text, otherwise, we return 'None'. Essentially, our **Preprocessor** class encapsulates a versatile set of text preprocessing functionalities adept at effectively managing diverse linguistic and formatting challenges.

Models

Our dataset undergoes training using diverse methodologies, including Long Short-Term Memory (LSTM), Multilayer Perceptron (MLP), Random Forest, and XGBOOSTING. While I won't delve into the intricacies of each method here, a succinct overview reveals that the MLP architecture incorporates multiple densely connected layers featuring Rectified Linear Unit (ReLU) activation functions. Dropout layers are incorporated for regularization, and the final layer employs a softmax activation function tailored for multi-class classification tasks. The model is compiled with categorical crossentropy loss and the Adam optimizer.

Subsequently, we employ both Count Vectorization and TF-IDF Vectorization techniques to transform the preprocessed text into matrices, capturing token counts and TF-IDF weights, respectively. The dataset is then partitioned into training and testing sets for both count vectorized and TF-IDF vectorized data, each accompanied by corresponding labels. The model is trained on the training data, and predictions are generated on the test set.

Evaluation

Following the training of our data on Multilayer Perceptron (MLP), Random Forest, and XGBoosting, we conducted a comprehensive evaluation of our methods and compiled a detailed report on the outcomes. The final results yielded closely comparable performances across all three models, showcasing consistently high levels of effectiveness.

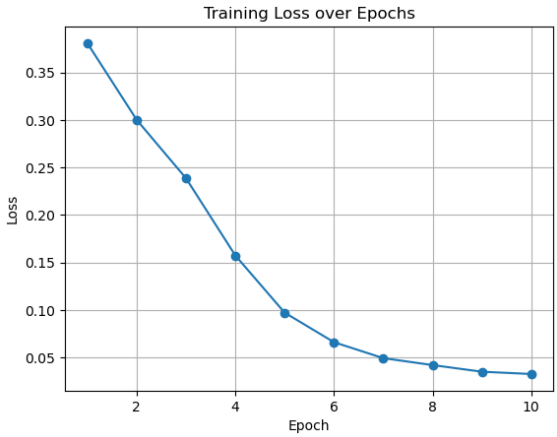

Fig.4)Loss over Epochs MLP

Models	Test Accuracy	F1-score
MLP	82.86%	82.86%
Random Forest	84.72%	85.00%
XGBoost	84.87%	85.00%

TBL.1) evaluation Results