Sentiment Analysis of Tweets

Q1: Simple Feature Extraction (Baseline Model)

I implemented the to_feature_vector(tokens) function to convert each tweet into a binary bag-of-words representation. The function returns a Python dictionary with tokens as keys and a weight of 1 for each token present in the tweet. Duplicate tokens are ignored, as repeated occurrences do not add extra weight in this representation. I also used a global dictionary (global_feature_dict) to keep track of all the unique tokens seen across the corpus.

Q2: Cross-Validation Strategy

I used 10-fold cross-validation on the 80% training split only (the 20% test split was left untouched until the end). Before splitting, I shuffled the dataset in the main code cell using a fixed random seed. This helps avoid any issues with the data being in a particular order or having uneven class distribution. In each fold, I trained the classifier on 9 parts of the data and tested it on the remaining 1 part. I used the given train_classifier() and predict_labels() functions to handle training and predictions. After each fold, I calculated precision, recall, F1-score (weighted), and accuracy using sklearn.metrics. I also printed how many positive and negative examples were in each split to make sure the classes were reasonably balanced. At the end of the process, I averaged the scores from all 10 folds and returned that as the cross-validation result.
Initially, I ran 10-fold cross-validation **without shuffling** the dataset and got the following average results across the folds:

Shuffling	Precision	Recall	F1-score	Accuracy
No shuffle	0.8287	0.8305	0.8289	0.8305
With shuffle	0.8340	0.8357	0.8343	0.8357

Conclusion: Decided to move forward with Shuffling based on the results.

Q3: Error Analysis

From results on the first test fold, the confusion matrix yielded:

- Precision: 0.832, Recall: 0.833, F1-score: 0.831, Accuracy: 0.833
- TP=1542, TN=695, FN=175, FP=272

Based on the observations from the FP and FN samples, key failure cases (error_analysis.text):
Mixed Sentiment: Tweets with both positive and negative tone (e.g., “beautiful night to throw myself off a bridge”).
Sarcasm: Literal words do not reveal negative intent (e.g., “Just sat through Twilight”).
Emotionless Negative Content: News about sad events without emotional words. (“Boko Haram suffered heavy losses on Sunday...)
Profanity Misuse: Intensifiers mistaken for negativity. (f***, a*******)
Missing Sentiment Words: Emotion implied, but no emotional vocabulary. (“Lost all faith in humanity.”)
Slang/Emoji Misinterpretation: Informal structures not well-matched to embedding space.

Q4: Improvements and Experiments (refer NLP_Assignment_1_Q4.ipynb)

All experiments were performed with class_weight='balanced' in SVM. A summary table follows.

Preprocessing Improvements - Preserved URLs, usernames, numbers, and retweets via <URL>, <USER>, <NUM>, <RT> tags, kept emojis and emotional punctuation like !, ?, ..., Normalised casing

and performed lemmatisation., Removed non-essential punctuation; implemented negation-aware stop word removal.

Feature Engineering and Vectorisation - Explored bigrams vs unigrams, switched from binary weights to TF-IDF vectorizer, tried combinations of word and character-level features, added optional sentence length as a stylistic feature, Integrated feature selection via vocab capping and chi², Included sentiment lexicon features from external resource.

Hyperparameter Tuning (SVM) - Cost parameter C was varied from 0.1 to 2.0. Best performance near C=1.

Experiment	F1-score
Basic Model (Q1-Q3)	0.8343
Step 1 (Token Removal and Preservation)	0.8625
Step 1 + Lowercase + Lemmatize	0.8725
Step 1 + Step 2 (Lowercase + Lemmatize + Stopword Removal)	0.8727
Step 2 Only	0.872
Bigrams Only (no Step 1 and 2)	0.8444
Bigrams + Step 1 + Step 2	0.8725
TF-IDF Only (no Step 1 and 2)	0.8495
TF-IDF + Step 1 + Step 2	0.8671
TF-IDF + Bigrams + Step 1 + Step 2	0.8765
TF-IDF + Bigrams + Step 1 + Step 2 + No Stopword Removal	0.8785
TF-IDF + Words(bigrams) + Chars + Step 1 + Step 2	0.8836
TF-IDF + Words(bigrams) + Chars + Step 1 + Step 2+ No Stopword Removal	0.8826
TF-IDF + Bigrams + Sentence Length	0.8734
TF-IDF + Bigrams + C=0.1	0.8607
TF-IDF + Bigrams + C=0.5	0.8741
TF-IDF + Bigrams + C=1	0.876
TF-IDF + Bigrams + C=2	0.8754
Limit Vocab to 10K	0.8656
Chi² Best k=4000	0.8729
Chi² Best k=8000	0.8739
Chi² Best k=12000	0.8741
TF-IDF + Bigrams + Opinion Lexicon	0.8829

Based on above results, Tf-idf Vectorizer's performance is better than the count vectorizer's performance so sticking with Tf-IDF. Obviously unigrams + bigrams are better than just unigrams. Sentence length stylistic feature didn't add any considerable performance improvement, not considering for best model

To obtain the best-performing model, I combined the most effective components from previous experiments, so using **TF-IDF with both word- and character-level features**, applying **Step 1 and Step 2 preprocessing**, integrating **opinion lexicon features**, and performing **chi² feature selection with k = 12,000**. The SVM classifier was tuned with **C = 0.9**, resulting in the highest **F1 score of 0.889495**. (see last model in notebook)