Basketball vs Football

Web API And Classification Using NLP

Aditi Sharma

Objective: Classification of Submission Posts from two subreddits

Process:

- Data collection from Reddit using pushshift.io APIs
- Data cleaning and EDA
- Preprocessing and Modeling
- Evaluation
- Conclusion

Data Collection

- Data Collection has been done from the reddit website using the Pushshift.io API.
- This API has been designed and created by the /r/datasets mod team in order to provide enhanced functionality and search capabilities so that Reddit comments and submissions can be searched for data collection.
- Pushshift limits to 100 posts per request and no longer than 500 in the screencast.
- Using the pushshift.io Reddit API, I downloaded around 1000 submissions (430 submissions from /r/basketball and 702 submissions from /r/football subreddits)

Data Cleaning & Preprocessing

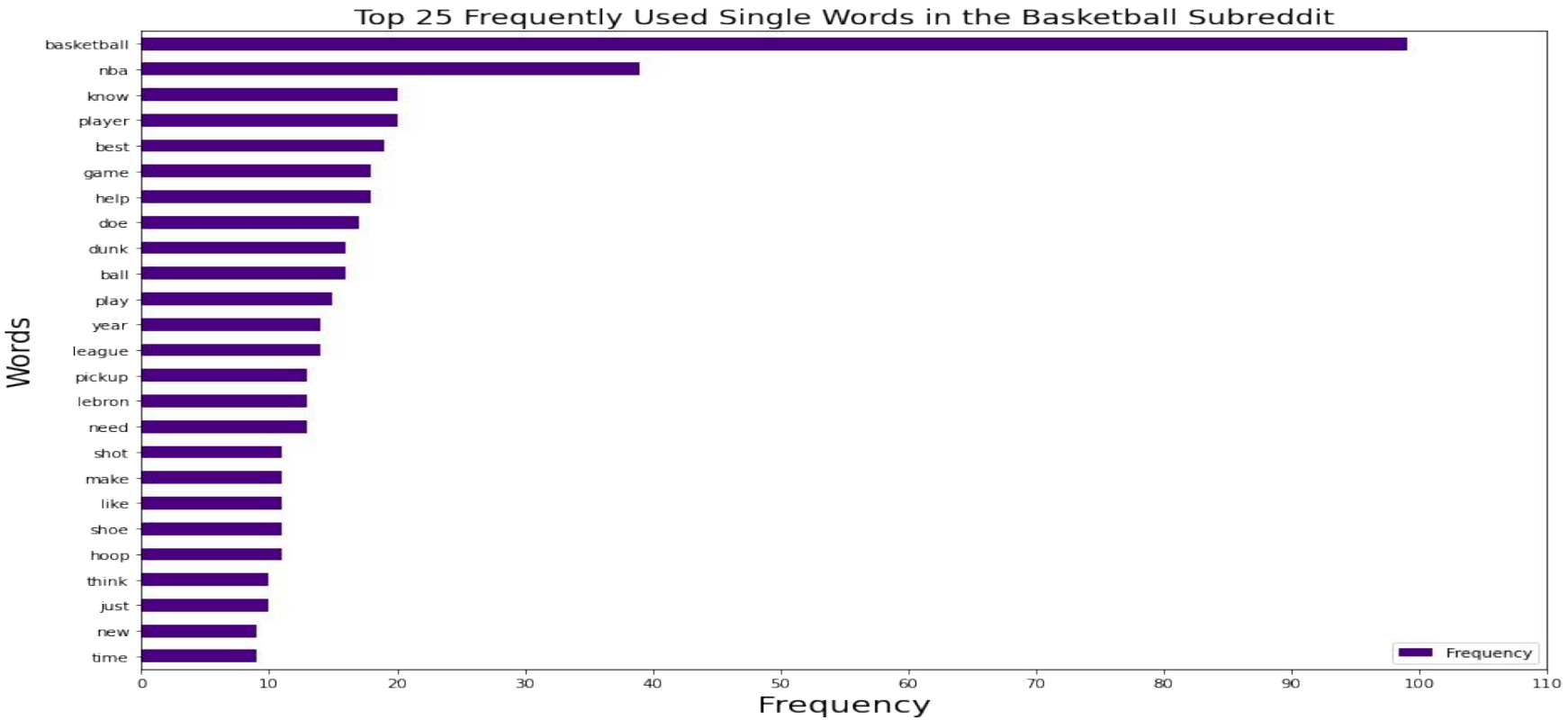
Cleaning:

- Got rid of duplicates ,
- Cleared unnecessary information like :- html, hyperlinks, punctuation, words with less than or equal to 2 letters, whitespace including line returns, emojis,
- After cleaning, the dataset reduced from 1133 entries to 1080 entries.

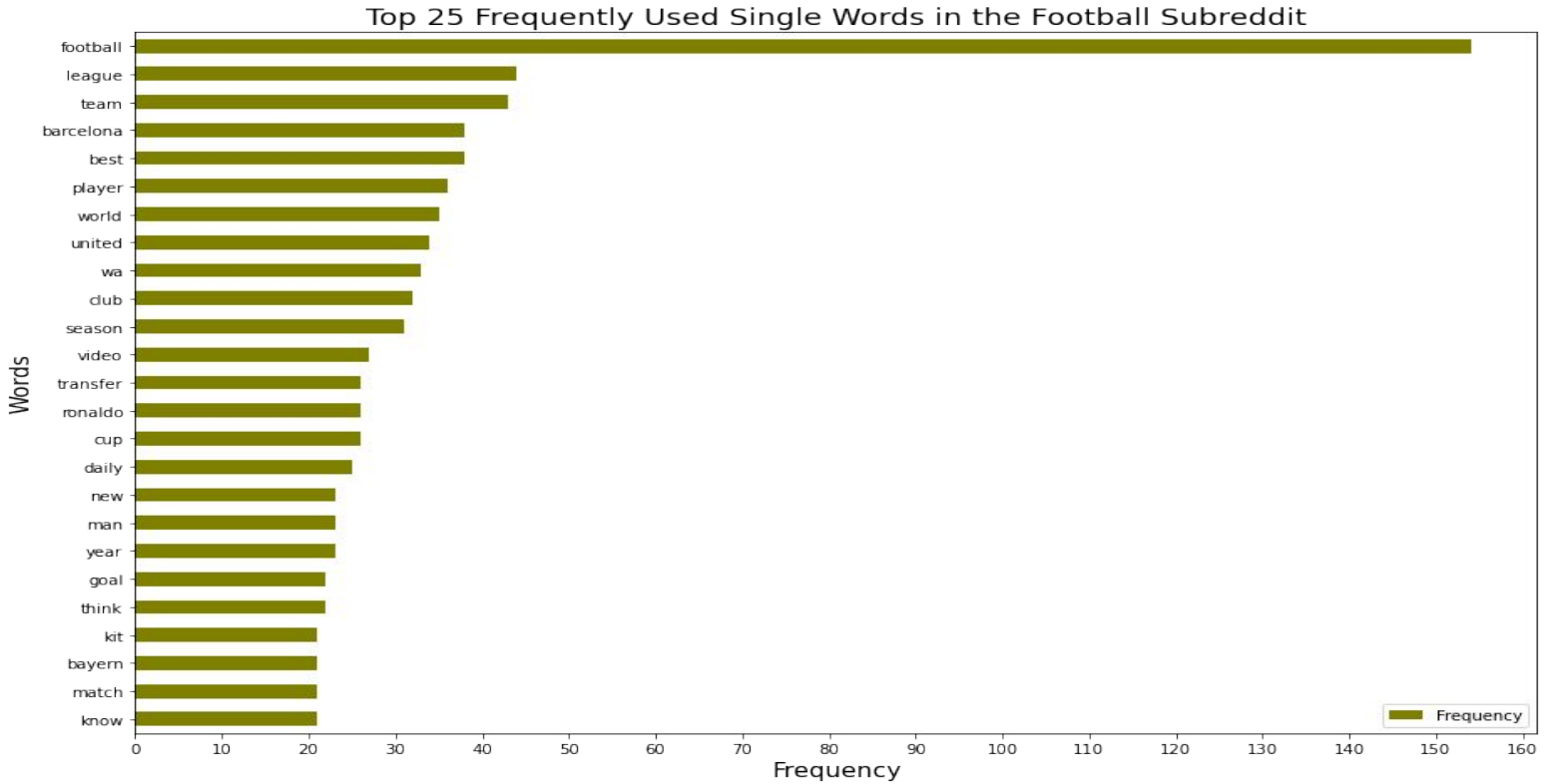
Preprocessing:

- Applied lemmatization,
- Used stopwords in ‘english’ to get rid of unnecessary words,
- Did train/test split (used default 0.25 test, stratify as yes).

EDA: Most frequent words in Basketball



EDA: Most frequent words in Football



Data Preprocessing

- CountVectorizer & Baseline logistic regression model train/test scores: 0.9864/ 0.8592
- Tf-idf & Baseline logistic regression model train/test scores: 0.9320 / 0.7889
- CountVectorizer & Random Forest model train/test scores: 0.9987 / 0.8111
- Tf-idf & Random Forest model train/test scores: 0.9987 / 0.7592
- CountVectorizer & SVC model train/test scores: 0.9938 / 0.8222
- Tf-idf & SVC model train/test scores: 0.9962 / 0.8259

Models & Tuning Using GridSearch:

Logistic regression:
Gridsearch best params: C = 1.0, penalty: l2
Train / test scores: 0.8368 / 0.8555

Random forest:
Gridsearch best params: max depth: None, rf__n_estimators: 30
Train / test scores: 0.7836 / 0.8111

Multinomial naive Bayes:
Gridsearch best params: alpha: 0.25
Train / test scores: 0.8442 / 0.8481

Conclusion

Best scoring model:

Multinomial Naive Bayes with Train / test scores: 0.8442 / 0.8481

Potential improvements:

Need to collect more data, indulge in more data cleaning and preprocessing of the given dataframe. Try to modify the stop words and filter out data like numbers and expressions. Practise more intensive gridsearch to optimize models