INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY BANGALORE

Report on Kaggle Assignment

Team Name :- ML Learning

Team Members:- 1) Manasi Purkar (MT2023158)
2) Ankush Patil (MT2023101)

CONTENTS

1. Understanding the Problem:

- Task Description:

  Given the book related information like reviews, no of likes, date of review etc the goal is to predict ratings (0 to 5) for reviews based on their textual content of reviews given by customers who bought the books.
  This is a multi-class classification problem since rating lies in the range 0 to 5.
  It is a Natural Language Processing (NLP) dataset and we are expected to analyse the text data and perform appropriate preprocessing steps such as text cleaning, tokenization, or stemming/lemmatization.
  We also have to create features using basic techniques like bag of words or TF-IDF whichever gives best results.

- Evaluation Metric:

  Model performance assessed using the weighted F1 score, considering both precision and recall.

2. Data Exploration:

- Load the Dataset:

  Import the dataset, containing both training and testing data.

- Libraries Used
  - NumPy and Pandas:
  - Regular Expressions: import re
  - Spelling Correction:
    - from symspellpy import SymSpell, Verbosity
  - Text Processing and Tokenization:
    - import nltk
    - from nltk.corpus import stopwords
    - from nltk.tokenize import word_tokenize
  - Remove Punctuations:
    - from string import punctuation
  - Contractions Expansion: import contractions
  - Data Cleaning, Lemmatization, POS tagging:
    - from nltk.stem import WordNetLemmatizer
    - from nltk import pos_tag
    - from nltk.corpus import wordnet

- Remove Stopwords:
    - nltk.download('stopwords')
    - nltk.download('punkt')
    - nltk.download('averaged_perceptron_tagger')
- Explore the Data:

  Conduct an initial exploration to understand the dataset's structure, check for missing values, anomalies, and analyse the distribution of ratings.

3. Data Preprocessing:

- Text Cleaning:
    1. Check emojis are present or not (not present in our dataset)
    2. Remove URLs using regular expression
    3. Remove new line symbol (\n) using regular expression
    4. Remove dates using regular expression
    5. Make lowercase
    6. Expanding the Contractions(replacement of contracted words with their expanded forms)
    7. Remove the Punctuations
    8. Remove Stopwords(removal of English stop words from the 'review_text' column)
    9. Spelling Correction
    10. Remove Repeated Words
    11. POS tagging(assigning a specific part-of-speech category (such as noun, verb, adjective, adverb, etc.) to each word

Removal of repeated words and the incorporation of Part-of-speech tagging, significantly enhanced the f1 score from 0.52 to 0.54.

- Tokenization:

  Using the NLTK library, Break down the textual content into individual words or tokens which is a fundamental step in Natural Language Processing (NLP).

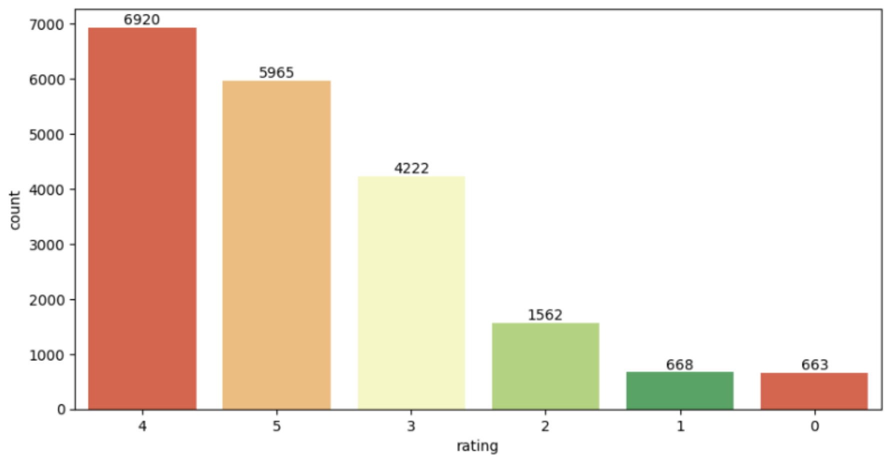
- Lemmatization:

  Using the NLTK library, Apply lemmatization to reduce words to their root form(lemma), simplifying the vocabulary for analysis and also helps in reducing the dimensionality of the data.

4. Exploratory Data Analysis (EDA):

- Class Distribution:

  Examine the distribution of ratings to understand the balance or imbalance among different classes (0 to 5).



5. Feature Engineering:

- Text Vectorization(TF-IDF):

  Use TF-IDF (Term Frequency-Inverse Document Frequency) to convert text data into numerical features, weighing words based on their importance.

  - Utilised TF-IDF vectorization with a combination of unigrams and bigrams.
  - Transformed both the training and test data into numerical features.The TfidfVectorizer from scikit-learn is used to convert a collection of raw text documents into a matrix of TF-IDF (Term Frequency-Inverse Document Frequency) features. The ngram_range parameter determines the range of n-grams to be considered during the vectorization process.
  - TfidfVectorizer(ngram_range=(1,2)): This sets the ngram_range parameter to consider both unigrams (single words) and bigrams (two consecutive words) during the vectorization process. In other words, each feature in the resulting TF-IDF matrix represents either a single word or a pair of consecutive words from the input text.

6. Model Building & Model Evaluation:

- **Choose Models:**

  Before choosing the model to train, we used TF-IDF which is Text Vectorization method to convert a collection of raw text documents into a matrix of TF-IDF features i.e. model doesn't understand the string instead model want/knows numerical data here this method of Vectorization helps us.

First we tried all these models on a 1lk dataset and the one which gives better results on the complete dataset. Models which we used to carried out the task are:

1) Logistic Regression:

   - Implemented Logistic Regression with a maximum of 1000 iterations.

- Trained the model on TF-IDF transformed data.
- Achieved a Weighted Fi 1 Score of 0.46.
- Got Accuracy after training the model on 1 Lakh Rows out of 6.30 Lakh Rows

2) KMeans Clustering:

- Applied KMeans clustering with 6 clusters to the TF-IDF transformed data.
- Evaluated the model using the F1 Score, resulting in a relatively lower score of 0.20.
- KMeans may not be the most suitable for this problem, as it is an unsupervised clustering algorithm.
- Got Accuracy after training the model on 1 Lakh Rows out of 6.30 Lakh Rows

3) Naive Bayes Multinomial Classifier:

- Employed the Multinomial Naive Bayes classifier.
- Trained on TF-IDF transformed data, which is suitable for text classification tasks.
- Achieved a Micro F1 Score of 0.37.
- Got Accuracy after training the model on 1 Lakh Rows out of 6.30 Lakh Rows

4) Support Vector Machine (SVM):

- Used SVM with a linear kernel and specified parameters.
- Trained the model on TF-IDF transformed data, which captures the importance of terms.
- Demonstrated strong performance with a Weighted F1 Score of 0.48.
- SVM is powerful for high-dimensional data which we can see using the f1 score we got 0.48.
- Got Accuracy after training the model on 1 Lakh Rows.

5) Random Forest Classifier:

- Implemented Random Forest Classifier with 100 trees.
- Obtained a Weighted F1 Score of 0.38.
- Got Accuracy after training the model on 1 Lakh Rows out of 6.30 Lakh Rows

6) Neural Network:

- Implemented neural network with 2 hidden layers and activation function Relu and output layer with activation function softmax.
- Obtained a Weighted F1 Score of 0.41.
- Got Accuracy after training the model on 1 Lakh Rows out of 6.30 Lakh Rows

7) Voting Classifier:

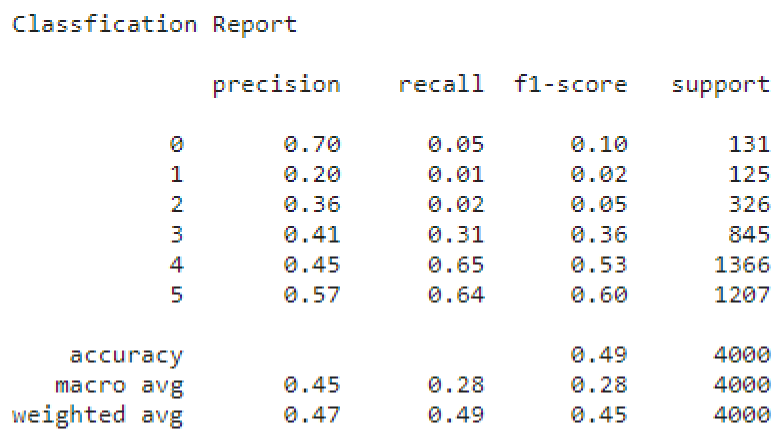
- VotingClassifier is an ensemble learning method in scikit-learn that combines the predictions of multiple base classifiers to improve overall performance. It supports both soft and hard voting strategies.
- Here we tried to use Logistic Regression and SVM as they gave better results.
- Obtained a Weighted F1 Score of 0.50.

- Hyperparameter Tuning:

  We did hyperparameter tuning in logistic regression using GridSearchCv for different values of C (inverse of regularisation strength). In GridSearchCv cross validation is used for better training results.
  The neural network tried to add and remove layers and neurons.

- Addressing Class Imbalance:

  First we checked the distribution of classes in the training dataset from that noticed classes are imbalanced. Because of that imbalance F1 scores of classes which have high counts came high and others low.



```
Classfication Report

              precision    recall  f1-score   support

           0       0.70      0.05      0.10       131
           1       0.20      0.01      0.02       125
           2       0.36      0.02      0.05       326
           3       0.41      0.31      0.36       845
           4       0.45      0.65      0.53      1366
           5       0.57      0.64      0.60      1207

    accuracy                           0.49      4000
   macro avg       0.45      0.28      0.28      4000
weighted avg       0.47      0.49      0.45      4000
```

  For handling this imbalance we tried some methods like undersampling, oversampling using SMOTE and by adding class_weight=balanced while training model. SMOTE and adding class_weight=balanced gave better results on small dataset but not significant increase on complete data.

7. Kaggle Submission:

- Prepare Submission File:

  Generated a submission file with review IDs and predicted ratings based on the selected model.

- Multiple Submissions:

  Take advantage of allowing multiple submissions to experiment with different approaches, keeping a record of changes made.

- Save Preprocessed Data:

  After running all preprocessed steps once we saved preprocessed data in CSV format to avoid re-execution of time-consuming preprocessing steps.

- Save Model:

  Saved trained model for future use using pickle file to avoid retraining.

8. Conclusion:

- Challenges Faced:

  Biggest challenge we faced during this project was handling large datasets. For running all preprocessing steps and training models it was taking a lot of time but we tried to handle this by storing results in files like csv or pickle. And instead of running all models on complete data first we tried on a small set of data.

  Focus on model performance and strive to climb up the leaderboard of your assigned group.

  Experiment with different models, parameters, and techniques to achieve the best results.

- Final Model Choice:

  After trying different models and checking its accuracy we selected a logistic regression model which works quite well while training and gives us a weighted f1 score to 0.542 on validation test dataset.

- Final Conclusion :

  Throughout the project, we kept learning and improving, understanding that each attempt helped us get better.

  In the end, we submitted our best predictions, showcasing not just our results but also the effort, learning, and improvement throughout the project.