**Problem Statement**

Using dataset of 10,000 reviews of mobile phone and their accessories, create a machine learning model which can predict whether the user recommend the product or not

**Preprocessing**

**Lower:** All the words are converted to lower cases.

**Punctuations:** We removed all punctuations

**Digits:** All digits are removed

**Stopwords:** We used NLTK stopwords and removed all of the stop words

**Lemmatization:** Lemmatization returns the base words. For example. It will convert charging to charge.

**Feature Extraction**

We used count vectorizer from sklearn to capture n-grams. We extracted n-grams based on words. Unigram, bi-gram and tri-gram are extracted out of the corpus. We used only those unigrams as tokens, which must occur in at least 10 documents. Similarly, Unigrams and Bigrams are used as tokens, which must occur in at least 20 documents. The same technique is applied where Unigrams, Bigrams and Trigrams are used as tokens, which must occur in at least 30 documents.

**Classification**

For classification purpose, the following models are used.

1. Support vector classifier

2. Naïve Baiyes Classifier

3. Random forest classifier

**Binary Classification**

In binary classification, our motives are to predict whether the user recommended the product or not. If the users gave a rating of 8 or higher, it means that the user has recommended the product. If the rating score is less than eight users did not recommend the product. In training dataset, there are total 10,000 reviews, 75% are positive reviews that the user recommends the product, 25% are negative reviews, it means that the user did not recommend the product.

|  |  |  |
| --- | --- | --- |
| Class | Total Reviews | Percentage |
| 1 (Recommended) | 75126 | 75.13% |
| 0 (Not recommended) | 24874 | 24.87% |

Next, we visualize the number of words in the recommended reviews and not recommended reviews. We plotted the kernel density estimate plot for this purpose. The plot shows that either people write a comment of 50 words mostly, or 300 words. It means people either write a very short or very long comment. The plot also shows that when a user like any product, they mostly write a brief comment

A close up of a map

Description automatically generated

We have collected the top 10 most frequent words in the whole training corpus. The formula of normalization is

As our minimum values are very less, *i.e.* 1. We take it at zero.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Common\_words | count | norm\_count |
| 0 | Phone | 74806 | 1 |
| 1 | good | 29491 | 0.394233 |
| 2 | great | 18213 | 0.24347 |
| 3 | camera | 15682 | 0.209636 |
| 4 | battery | 15552 | 0.207898 |
| 5 | one | 12397 | 0.165722 |
| 6 | price | 10729 | 0.143424 |
| 7 | best | 10310 | 0.137823 |
| 8 | screen | 10178 | 0.136059 |
| 9 | like | 8898 | 0.118948 |

The most common words are phone, camera, screen, battery which shows that most of reviews are about mobile phones.

We have tried 3 feature extraction techniques, and for each feature extraction technique, three different classifiers are used.

**Evaluation Criteria**

We have used accuracy, macro precision, macro recall and macro f1 score as our evaluation criteria. As the dataset is imbalanced, so our main criteria are macro f1 score.

Precision is measured as instances of true positive, divided by the total number of positive predictions

The recall is the number of correct positive predictions out of all actual positive.

F1 score converts precision and recall into one function by taking weightage average of recall and precision. F1-score is considered as the best evaluation criteria when data is not balanced.

Accuracy is the number of correct predictions out of all predictions

**Cross-Validation**

We have applied K fold cross-validation in order to achieve the confidence of our model. In K fold cross-validation, data is split into K parts. One part is used for testing and K-1 is used for training. Details steps of K fold cross-validation is as follow

1. Shuffle the data randomly.
2. Split the data into k groups
3. For each unique K group:
   * 1. Take out the group for test set
     2. Consider the remaining groups as a training set
     3. Fit a model on the training set
     4. Evaluate the model on the test set
     5. Repeat the steps by taking another group for the test.
4. Summarize the evaluation criteria of the model

**Results**

**Uni-gram**

**Naïve Bayes**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Folds | Accuracy | Precision | Recall | F1-score |
| 0 | 83.8 | 78.8 | 76 | 77.2 |
| 1 | 84.2 | 79.3 | 77.2 | 78.2 |
| 2 | 84.2 | 78.8 | 79.6 | 79.2 |
| 3 | 85.4 | 81.1 | 78.3 | 79.5 |
| 4 | 85 | 80.7 | 77.3 | 78.7 |
| mean | 84.5 | 79.7 | 77.7 | 78.6 |
| std | 0.6 | 1 | 1.2 | 0.8 |

Using uni-gram features, F1-score of 80% is achieved when classified with Naïve Bayes. We achieved a precision of 79.7%, recall of 77.7% and accuracy of 84.5%.

**Random Forest**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Folds | Accuracy | Precision | Recall | F1-score |
| 0 | 84.8 | 81.3 | 75.4 | 77.6 |
| 1 | 84.8 | 81 | 75.9 | 77.9 |
| 2 | 85.1 | 81 | 77.3 | 78.9 |
| 3 | 86.5 | 83.3 | 79 | 80.8 |
| 4 | 85.6 | 82.3 | 77.1 | 79.1 |
| mean | 85.4 | 81.8 | 77 | 78.9 |
| std | 0.7 | 0.9 | 1.3 | 1.1 |

When random forest classifier is used, the accuracy of 85.4%, 81.8%, 77.0%, 78.9% precision, recall and F1-score is obtained. In unigram feature extraction, the random forest gives the best result with f1-score of 78.9%.

**SVM**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | accuracy | precision | recall | f1\_score |
| 0 | 84.7 | 82.1 | 74.3 | 76.9 |
| 1 | 84.8 | 81.7 | 74.9 | 77.3 |
| 2 | 85.7 | 81.9 | 78.3 | 79.8 |
| 3 | 85.5 | 83.2 | 75.6 | 78.3 |
| 4 | 85.7 | 83.6 | 75.8 | 78.5 |
| mean | 85.3 | 82.5 | 75.8 | 78.2 |
| std | 0.4 | 0.7 | 1.4 | 1 |

When random forest classifier is used, F1-score is bit less then random forest as it is 78.2% as compared to 78.9%. Precision is high in SVM as compared to random forest, where as recall is low. There is not much difference in accuracy between both models.

**Bigram**

**Naïve Bayes**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | accuracy | precision | recall | f1\_score |
| 0 | 83.2 | 77.6 | 76.3 | 76.9 |
| 1 | 83.5 | 78 | 77.3 | 77.7 |
| 2 | 83.3 | 77.6 | 79.2 | 78.3 |
| 3 | 85.1 | 80.2 | 79.2 | 79.7 |
| 4 | 84.7 | 80 | 77.8 | 78.8 |
| mean | 84 | 78.7 | 78 | 78.3 |
| std | 0.8 | 1.2 | 1.1 | 0.9 |

Using uni-gram and bigram features Naïve Bayes F1 score reduces a bit from 78.6 to 78.3. A

**Random Forest**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | accuracy | precision | recall | f1\_score |
| 0 | 84.8 | 81.4 | 75.4 | 77.7 |
| 1 | 84.7 | 81 | 75.9 | 77.8 |
| 2 | 85.3 | 81.4 | 77.2 | 78.9 |
| 3 | 86.5 | 83.4 | 78.7 | 80.6 |
| 4 | 85.3 | 81.8 | 76.8 | 78.8 |
| mean | 85.3 | 81.8 | 76.8 | 78.8 |
| std | 0.6 | 0.8 | 1.2 | 1.1 |

There is not much difference in result obtained from uni-gram token features and bigram token features, when random forest classifier is used.

**SVM**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | accuracy | precision | recall | f1\_score |
| 0 | 84.5 | 81.8 | 73.9 | 76.6 |
| 1 | 84.6 | 81.4 | 74.8 | 77.2 |
| 2 | 85.4 | 81.5 | 77.9 | 79.4 |
| 3 | 85.9 | 83.7 | 76.3 | 78.9 |
| 4 | 85.8 | 83.9 | 76 | 78.7 |
| mean | 85.3 | 82.5 | 75.8 | 78.2 |
| std | 0.6 | 1.1 | 1.3 | 1.1 |

Like random forest show no much difference in bi-gram and unigram features, SVM did not show much different also.

**Trigram**

**NB**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | accuracy | precision | recall | f1\_score |
| 0 | 83.2 | 77.6 | 76.4 | 77 |
| 1 | 83.5 | 77.9 | 77.4 | 77.7 |
| 2 | 83.2 | 77.4 | 79.1 | 78.2 |
| 3 | 85 | 80 | 79.2 | 79.6 |
| 4 | 84.6 | 79.9 | 77.8 | 78.7 |
| mean | 83.9 | 78.6 | 78 | 78.2 |
| std | 0.8 | 1.1 | 1.1 | 0.9 |

Accuracy, precision, recall, f1-score of 83.9%, 78.6%, 78% and 78.2 % is achieved when tri-gram features are used with Naïve Bayes classifier. We noticed there is minor decrease in tir-gram model performance as compared to unigram naïve bayes model.

**RF**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | accuracy | precision | recall | f1\_score |
| 0 | 84.8 | 81.4 | 75.3 | 77.6 |
| 1 | 84.7 | 81 | 75.7 | 77.8 |
| 2 | 85.2 | 81.4 | 77.1 | 78.8 |
| 3 | 86.6 | 83.6 | 78.9 | 80.8 |
| 4 | 85.3 | 81.8 | 76.8 | 78.8 |
| mean | 85.3 | 81.8 | 76.8 | 78.7 |
| std | 0.7 | 0.9 | 1.2 | 1.1 |

Random forest shows a stable result all across three feature extraction techniques and there is not much difference in the model performance.

**SVM**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | accuracy | precision | recall | f1\_score |
| 0 | 84.5 | 81.7 | 73.9 | 76.5 |
| 1 | 84.6 | 81.5 | 74.8 | 77.2 |
| 2 | 85.3 | 81.3 | 77.7 | 79.2 |
| 3 | 85.8 | 83.7 | 76.2 | 78.8 |
| 4 | 85.8 | 83.9 | 76 | 78.7 |
| mean | 85.2 | 82.4 | 75.7 | 78.1 |
| std | 0.6 | 1.1 | 1.3 | 1.1 |

Like random forest SVM results are also stable no matter which n-gram tokens are used. An accuracy of 85.2% and F1 score of 78.1% is achieved using SVM with tri-gram features.

**Conclusion**

It is clear from the graph that decision tree with unigram token outperform the other models, though with slight difference. There is another trend to notice, increasing n in n-gram reduces the average F1 score slightly. Unigram token feature perform best in all three models as compared to bigram and trigram model. Lowest F1-score of 78.1%i s achieved in tri-gram when SVM model is used and the highest is 78.9% when unigrams are trained with decision trees.