**Observations Criteria:**

Note:

1. **clean\_data** and **clean\_data\_version\_2** are the two versions of data pre-processing that were used for testing the accuracy of the LR model. To understand to what extent the data pre-processing can improve the performance of the LR model.
2. Two vectorization modules have been used to test the LR model performance one at a time, the first is the count vectorizer and the other one is the TF-IDF vectorizer
3. **Without N-grams feature extraction:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Regularization Parameter C**  **Default=1.0** | **Optimization Algorithm Solver**  **Default=’lbfgs’** | **Class\_weight**  **Default=None** | **Data version** | **Vectorization** | **Accuracy** |
| Default | Default | Default | Clean\_data | CountVectorizer with Max-features=4000 | 62% |
| Default | liblinear | Default | Clean\_data | CountVectorizer with Max-features=4000 | 62% |
| Default | Default | ‘balanced’ | Clean\_data | CountVectorizer with Max-features=4000 | 56% |
| 0.5 | Default | Default | Clean\_data | CountVectorizer with Max-features=4000 | 64% |
| 0.5 | Default | Default | Clean\_data\_version\_2 | CountVectorizer with Max-features=1600 | **70%** |

1. **With N-grams feature extraction:** Regularization parameter C and optimization Algorithm is set to default, added the class\_weight=’balanced’ to see the difference

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Regularization Parameter C**  **Default=1.0** | **Optimization Algorithm Solver**  **Default=’lbfgs’** | **Class\_weight**  **Default=None** | **Data version** | **Vectorization**  With ngram\_range(1,2) | **Accuracy** |
| Default | Default | balanced | Clean\_data | CountVectorizer with max\_features=50 | 56% |
| Default | Default | balanced | Clean\_data\_version\_2 | CountVectorizer  With max\_features=50 | 68% |
| Default | Default | ‘balanced’ | Clean\_data | TfidfVectorizer with max\_features=50 | 53% |
| Default | Default | balanced | Clean\_data\_version\_2 | TfidfVectorizer With max\_features=50 | **69%** |
| Default | Default | Default | Clean\_data\_version\_2 | CountVectorizer with max\_features=100 | 63% |
| Default | Default | balanced | Clean\_data\_version\_2 | TfidfVectorizer With max\_features=100 | **63%** |

**# Regularization parameter C and optimization Algorithm is set now, reverted the class\_weight to default and, max features set to 100**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Regularization Parameter C**  **Default=1.0** | **Optimization Algorithm Solver**  **Default=’lbfgs’** | **Class\_weight**  **Default=None** | **Data version** | **Vectorization**  With max\_features=100,  ngram\_range(1,2) | **Accuracy** |
| 0.1 | Default | Default | Clean\_data | CountVectorizer | 71% |
| 0.1 | Default | Default | Clean\_data\_version\_2 | CountVectorizer | **72%** |
| 0.1 | Default | ‘Default | Clean\_data | TfidfVectorizer | 70% |
| 0.1 | Default | Default | Clean\_data\_version\_2 | TfidfVectorizer | **72%** |
| 0.1 | liblinear | Default | Clean\_data\_version\_2 | CountVectorizer | **72%** |
| 0.1 | liblinear | Default | Clean\_data\_version\_2 | TfidfVectorizer | **72%** |

**SUMMARY OF ANALYSIS OF RESULTS:**

1. Overall accuracy of the LR model is better when n-gram feature extraction is used while doing vectorization irrespective of TF-IDF or count vectorizer and also manipulating regularization, and optimization algorithm parameter, which is **72%.**
2. Without n-gram extraction and just manipulating the regularization and algorithm optimization parameters for the logistic regression model, the accuracy is **70%.**
3. The reason why with n-gram the model performed better as compared to without doing the n-gram feature extraction is the context perseverance. Because with vectorization the text will be converted into a bag of words, which is not ordered as they are collected, so a lot of contexts are lost from the text.
4. Hence N-gram resolves the issue by maintaining some order to preserve context
5. But this accuracy slightly improved when the version of clean data has been changed to a more improved version of data preprocessing i.e., clean\_data\_version\_2

Because more clean data might induce more sense to the information and it becomes for the model to learn the language if the data is not filled with too much noise

1. One more interesting thing to notice is the logistic regression model hyperparameter `**C**`, the regularization parameter. When providing a lower value to C i.e., 0.1, the model performed better as compared to providing higher values. The high value of C guides the model to give more weight to the training data and the lower value indicates the model gives complexity more weight at the cost of fitting data. It indicates that the training data has more importance and reflect the real-world data but here it is the opposite.
2. The vectorization captured unigram and bigrams feature extraction because trigrams were dropping the accuracy by 2 percent. Might be because, when it comes to n-grams, more is not necessarily better. In some cases, having too much information or features will result in a less optimal model.
3. Fine-tuned the number of maximum features as well. With and without both n-gram feature extraction the LR model accuracy improved only by decreasing the number of features to almost half. Because too many features in the regression will cause the model to overfit which leads to model complexity, making it poor predictions on new data.
4. When class\_weight is **balanced** with an improved version of processed data i.e., clean\_data\_version\_2 is used for the regression, the accuracy increased to 69% for TD-IDF vectorization. It is because the data is pre-processed enough to give the correct information. Our data is imbalanced it has the majority of the class as non-offensive so used balanced class weight.
5. Both the vectorization module performed same i.e., 63% with n-gram feature extraction, regularization and optimization algorithm parameters provided, but with fewer features, TF-IDF performed well as compared to countvectorizer