**Model training, selection and hyperparameter tuning and evaluation:(20%)**

in this section we were trying to figure out the best model with the best parameters possible, To do that we have combined **Pipeline** with **the Grid Search** where the Pipeline would be using CountVectorizer and TFIDF transformer and one of 3 estimators

LogisticRegression

MultinomialNB

SGDClassifier

**Pipeline** is a process that enabled us to create all the steps needed for our text analytics model to work starting from text vectorization, TF-IDF and then applying our statistical model. We have also used **GridSearch** and that enabled us to test several parameters with several settings allowing us to test several parameters combinations and compare results in one run , we even tested functions we wrote in previous steps and saw how they impact the efficiency of the model. We did test multiple combination of models , estimators and parameters, We used dictionaries to load estimators and the parameters , using dictionaries enabled us to add remove estimators and parameters as quickly as needed.

The drawback of that process is it was very compute intensive process and complex to scale we did go through several iteration of processor and memory optimization to make the code work on larger servers ( the use of **n\_jobs** on several cores lead to known symptom of *memory explosion* , we had to optimize using **pre\_dispatch** to control the number of jobs that get dispatched during parallel execution ) .

What we have also learned by experiment that many of the parameters did not contribute positively or impact the results, in fact in many scenarios using the default parameters was the best option

Below is a sample of the model output using *LogisticRegression* with GridSearch **n-gram** of (1,1),(1,2),(1,3)

LogisticRegression scored 0.6396866350799831

Best parameter (CV score=0.639):

{'classifier\_\_C': 0.0001, 'cv\_\_max\_df': 1, 'cv\_\_ngram\_range': (1, 1), 'cv\_\_stop\_words': None, 'tfidf\_\_use\_idf': True}

precision recall f1-score support

1 0.00 0.00 0.00 9775

2 0.00 0.00 0.00 5518

3 0.00 0.00 0.00 8044

4 0.00 0.00 0.00 15067

5 0.64 1.00 0.78 68181

accuracy 0.64 106585

macro avg 0.13 0.20 0.16 106585

weighted avg 0.41 0.64 0.50 106585

Cross validation with unseen test data test data , not used in training

Model score on unseen data 0.6377274582377528

precision recall f1-score support

1 0.00 0.00 0.00 13075

2 0.00 0.00 0.00 7416

3 0.00 0.00 0.00 10647

4 0.00 0.00 0.00 20346

5 0.64 1.00 0.78 90630

accuracy 0.64 142114

macro avg 0.13 0.20 0.16 142114

weighted avg 0.41 0.64 0.50 142114

We have also tried to normalize the effect of the data imbalance by using weighted labels , while there was no major overall model accuracy impact, the test on the unseen test data (with N-gram) (1,2) improved by more than **9 pts** and we were able to get the below results

Cross validation with unseen test data

Model score on unseen data **0.762901614197053**

precision recall f1-score support

1 0.72 0.73 0.72 13075

2 0.58 0.17 0.27 7416

3 0.55 0.37 0.44 10647

4 0.58 0.25 0.35 20346

5 0.80 0.98 0.88 90630

accuracy 0.76 142114

macro avg 0.65 0.50 0.53 142114

weighted avg **0.73 0.76 0.72** 142114