**Document/Text Classification using various models**

**Synopsis:**

**Problem Statement**

Large amounts of text data are just raw data with no means of providing information unless processed and classified. In this assignment, a set of documents (20 newsgroup) will be classified into predefined categories using various models. We may also use unsupervised machine learning and cluster similar documents together. I will try to compare as many models as I can and try to do some unsupervised learning clustering too.

**Objectives**

1. To pre-process the raw data by removing objects like stop words, punctuations, and convert them into a standardized format.
2. To extract features from the models using various methods such as TF-IDF, Word embeddings, etc.
3. To use various models on the extracted features to classify the documents and compare all the models with each other using various metrics such as Accuracy, Precision, Recall, F1-score, etc.

**General Steps:**

1. Data Collection: Collect a set of documents from different domains and annotate them with one or more labels or categories.
2. Data Pre-processing: Clean and pre-process the documents by removing stop words, punctuations, and other irrelevant characters, and convert them into a standardized format.
3. Feature Extraction: Extract features from the pre-processed documents using different methods, such as TF-IDF, Word Embeddings, or N-grams.
4. Model Selection: Select various machine learning models, such as Naive Bayes, SVM, Decision Trees, Random Forest, and Neural Networks, and evaluate their performance using cross-validation or train-test split.
5. Hyperparameter Tuning: Tune the hyperparameters of the selected models using grid search or randomized search to optimize their performance.
6. Model Evaluation: Evaluate the performance of each model using metrics such as Accuracy, Precision, Recall, F1-score, and Confusion Matrix, and compare them to select the best performing model.
7. Model Deployment: Deploy the selected model to classify new documents into predefined categories and evaluate its performance on unseen data.

**Expected Outcome**

The expected outcome is to identify the best supervised/ unsupervised model for document classification using various evaluation metrics.

**Method and Methodology:**

**Data:**

The dataset used in this assignment is the “20 Newsgroup” dataset which is a part of the scikit-learn library. It is a dataset that contains 20,000 newsgroup documents that cover a variety of topics such as politics, sports, religion among many others. A newsgroup document refers to a newsgroup post which are similar to emails in their format and content. However, instead of being sent directly to other people, they are posted to public newsgroups.

**Block Diagram:**

Below is the general block diagram of a text or document classifier:

Raw NLP Pre- Feature Extraction Classification

Documents Processing Algorithm

Model Evaluation

**Problem Domain:**

Text or Document classification falls under Natural Language Processing (NLP) and Machine Leaning (ML). In this, we categorize the text or data into different predefined categories based on their content. Some applications of text classification are:

* Information retrieval
* Spam filtering
* Sentiment analysis
* Topic modelling.

**Algorithm**

Even though various models are used in this assignment, all of them follow a general algorithm. Here, the general algorithm will be displayed. More details on specific algorithm will be under the implementation section next to each model.

1. *Data Pre-processing:*

* Involves converting text to lowercase, removing punctuations, numbers, and special characters.
* Tokenization of text, removing stop words, stemming or lemmatizing

1. *Feature Extraction:*

* Create a bag of words using a method like CountVectorizer()
* Perform TFI-DF transformation on it.

1. *Split Data*: Split the dataset into training and testing
2. *Train the model:* Choose the classification algorithm to be used.
3. *Evaluate the model:* Use the classification report metric
4. *Fine tune the model:*

* Perform Hyperparameter tuning
* Perform Regularization
* Use above 2 methods only if necessary

1. *Apply the model:* Use the trained model to classify new documents

**Technologies Used**

There are many technologies I have used in this assignment to classify the “20 Newsgroups” dataset. Some of them are:

1. Natural Language Processing(NLP): Many NLP techniques have been used to prepare the data to be fed to the classifier.
2. Bag-of-Words(BoW) model: It is a simple way to represent text as a numerical vector. Each element represents the frequency of occurrence of a word in a document. It is a feature extraction technique used in document classification.
3. Term Frequency-Inverse Document Frequency (TF-IDF): It assigns weights to words based on the its frequency and importance in the document. It is also part of feature extraction.
4. Naïve Bayes: Probabilistic machine learning algorithm that works by computing the conditional probability of each class given the features and then selecting the class which has the highest probability.
5. Support Vector Machines(SVM):

**Implementation**

For each implementation, the hyperparameter tuning has been done via trial-and-error, and a GridSearchCV() method. This method trains the model over a range of hyperparameters. We can select the best parameters for us by seeing which combination of parameters provides us with the best results.

In this assignment with a combined use of GridSearchCV() and various trial-and-error manually done by me, I have selected the best(to the best of my ability) hyper-parameters, regularization values for each model.

I have not included the GridSearchCV() method in the actual program codes because it is computationally very expensive, hence I have run it only once for each model and will be showing a sample of how GridSearchCV() works.

Here is an example of how Grid Search code looks:



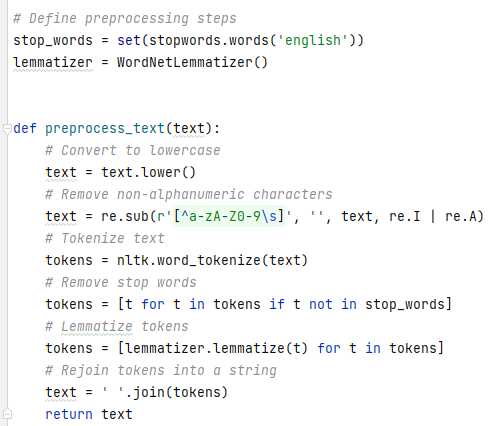
The main thing to note is the ‘params’ dictionary. This contains a list of all parameter values we would like to test out to get the best output.

Another method that I have used in all the models is the preprocess\_text method. This is a method that I defined that performs text preprocessing on the model. The preprocessing includes:

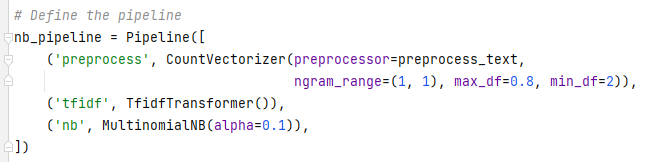
1. Lowering the case
2. Removing all non-alphanumeric characters and replacing them with blank space.
3. Performing word tokenization

Here is an example of what the code looks like. I have called this method inside the pipeline building. It has a parameter called preprocessor which can be set to a user defined pre processing method. Here is the code:

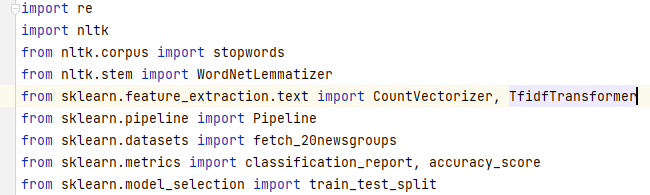
Method code:



Example of using it in the pipeline:

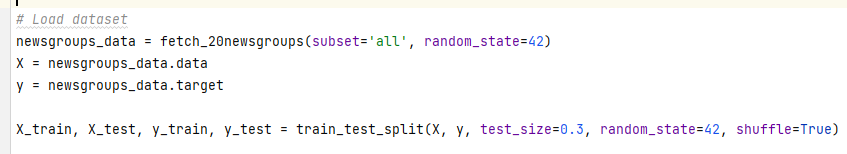


Here are the imports that are common to all the codes:



These imports are used in all the models that I have made. Will show the specific imports to each model as I display them.

Here is the training and testing data set splitting which is also common to all the codes:



**Models**

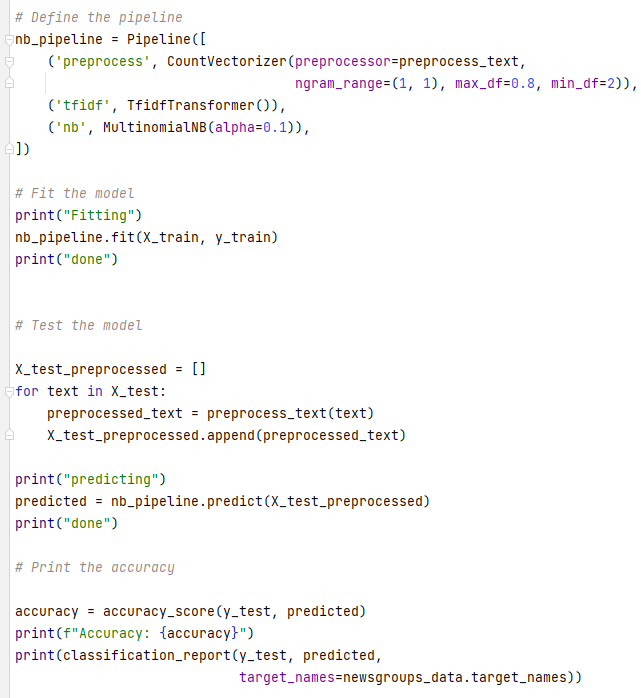
**Naïve-Bayes model**

Uses multinomial Naïve Bayes

Package import:



Code:



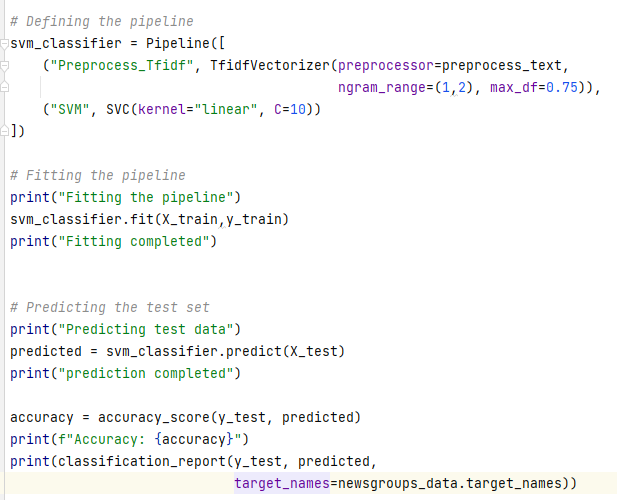
**Support Vector Machine (SVM)**

Uses SVC (Support Vector classification)

Package import:



Code:

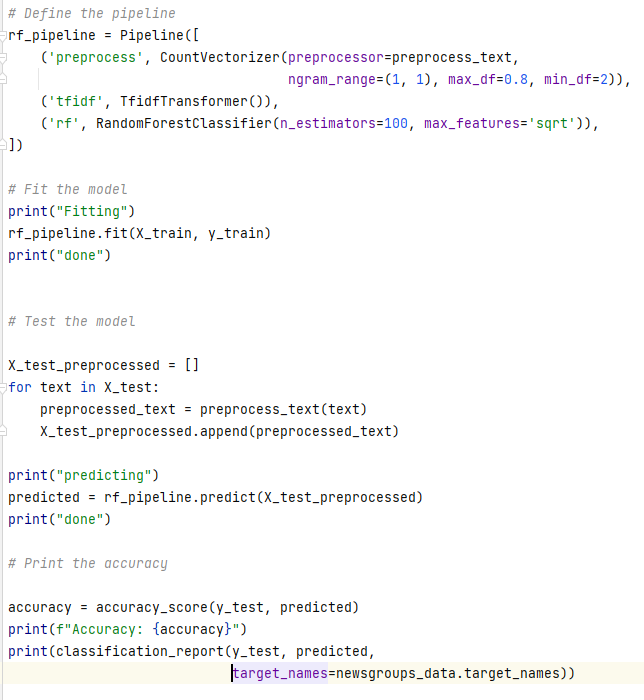


**Random Forest Classifier**

Package import:



Code:



**Logistic Regression**

Package import:

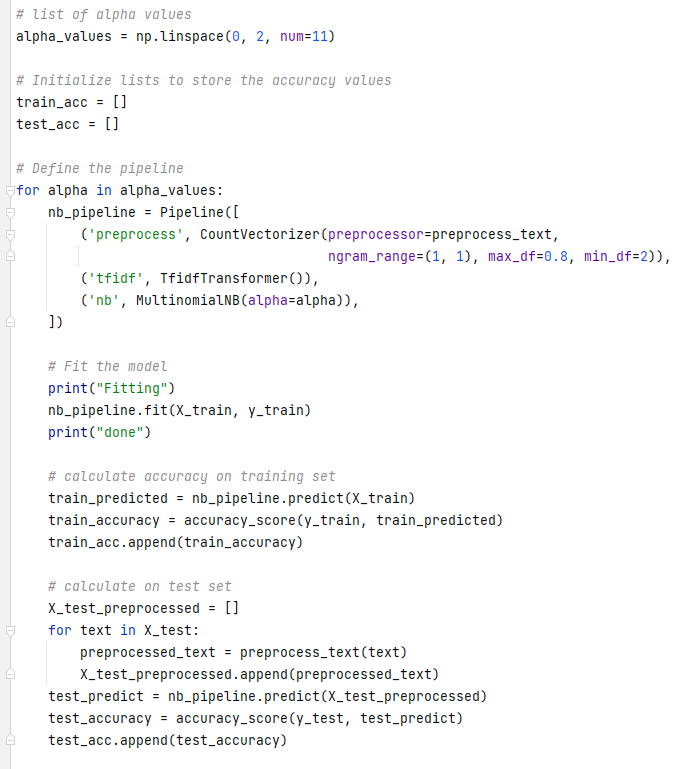


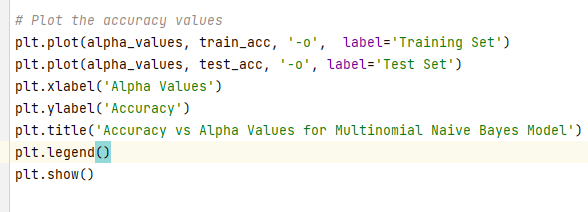
Code:



**Graph with Naïve Bayes**

This graph code shows the plot of accuracy vs alpha value:



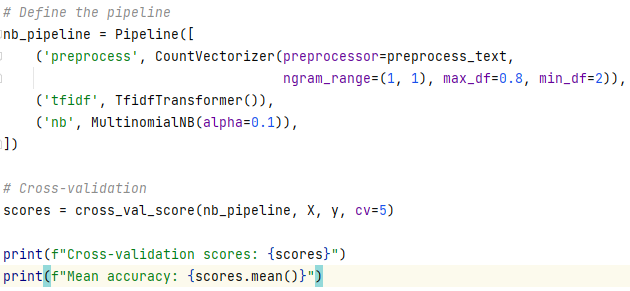


**Naïve Bayes cross validation:**

Package import:

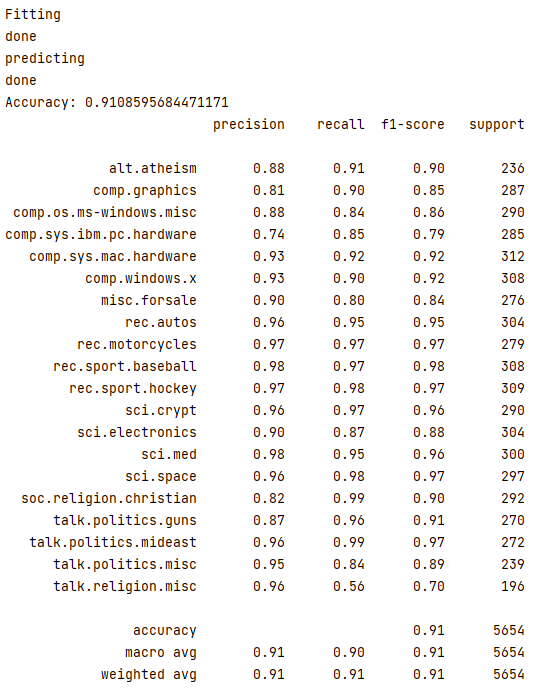
****

Code:

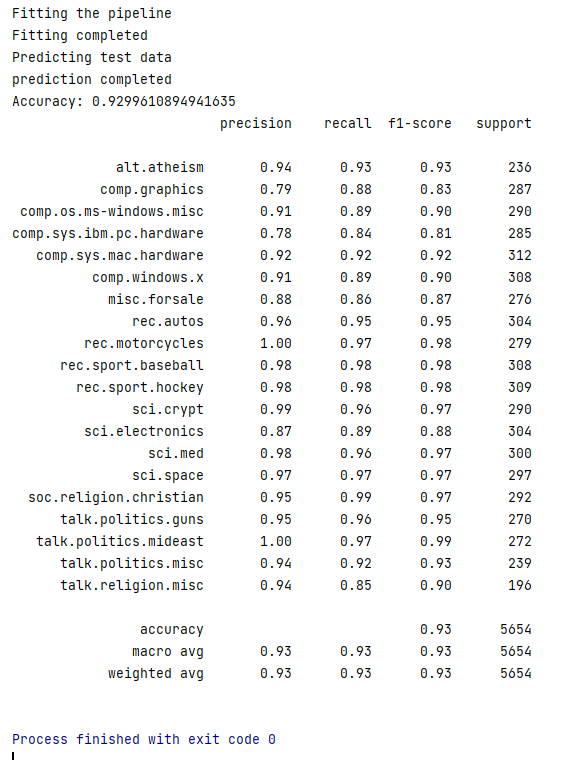


**Results**

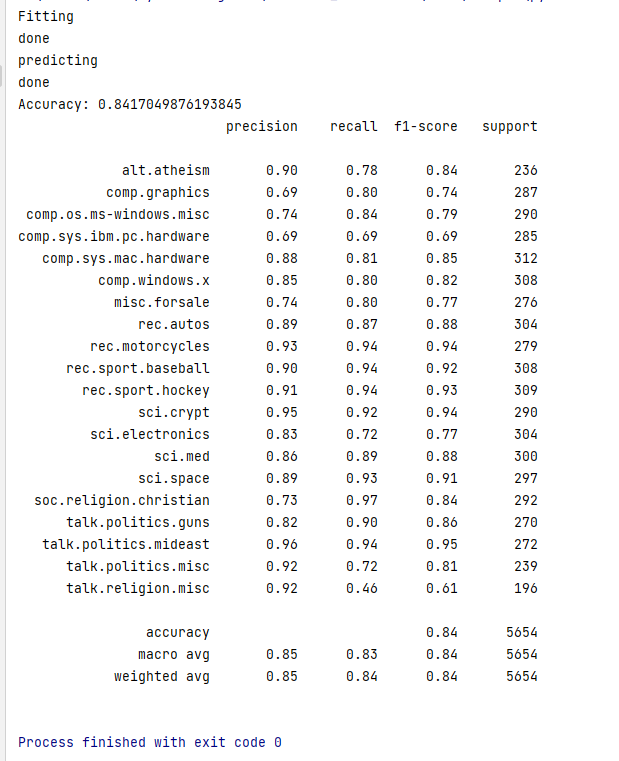
**Naïve Bayes Result:**

****

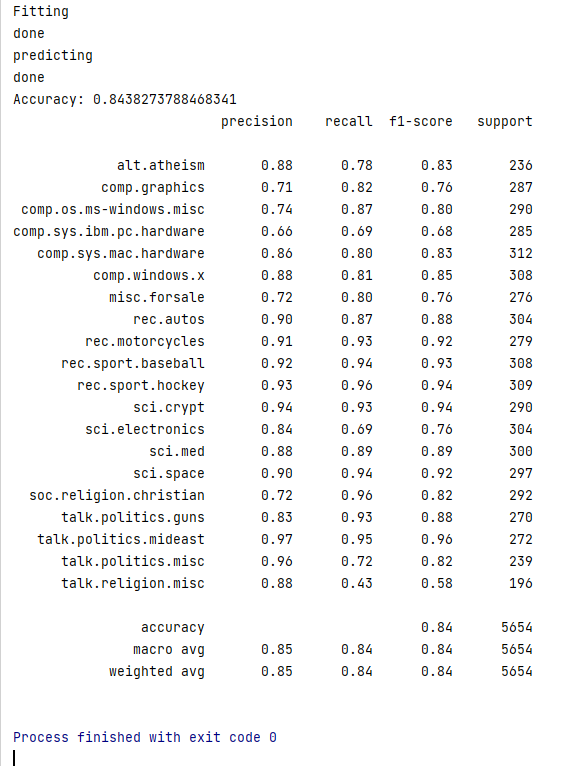
**SVM Result:**

****

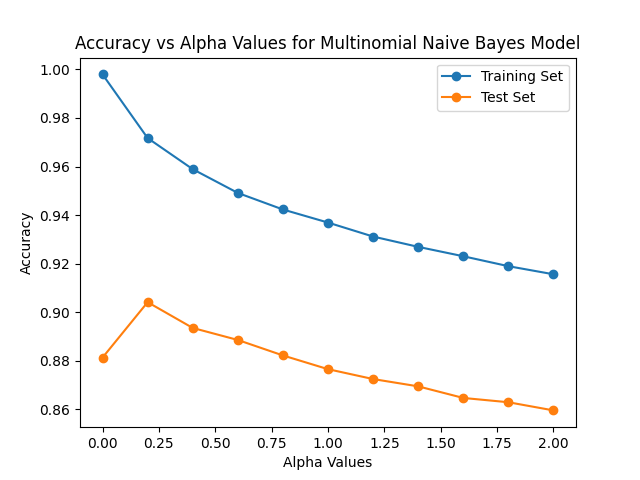
**Random Forest Classifier Results:**

****

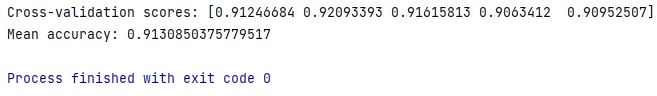
**Logistic Regression Results:**

****

**Naïve Bayes Alpha vs Accuracy Graph:**

****

**Naïve Bayes cross validation score:**

****

Cross Validation score same as testing scores obtained previously. Hence model is not overfit.

**Conclusions made from the results:**

* The order of computational complexity, i.e., training time from lowest to highest is as follows:
  + Naïve Bayes
  + Logistic Regression
  + Random Forest Classifier
  + SVM
* The order of accuracy that I have achieved from highest to lowest is as follows:
  + SVM
  + Naïve Bayes
  + Logistic Regression
  + Random Forest Classifier (almost same as LR)
* Therefore, the best model to use for text classification is SVM. However, if you prefer shorter computational times while training the model, use Naïve Bayes.
* Here is a table showing the accuracies and Computational time ranks of each model that has been used for our Text classification problem.

|  |  |  |
| --- | --- | --- |
|  | **Accuracy** | **Computational Efficiency** |
| *Naïve Bayes* | 0.9108 | 1 |
| *SVM* | 0.9299 | 4 |
| *Random Forest* | 0.8412 | 3 |
| *Logistic Regression* | 0.8438 | 2 |

**How good was the hyperparameter tuning? Have any of the models overfitted?**

Do note that the hyperparameter tuning of the models have been done to the best of my ability. The order may vary in practice. Also note that I have gone for the most balanced accuracy, i.e., an accuracy with the least amount of overfitting. Have been able to achieve higher training accuracy but it was overfitted and hence not selected. None of the above models have been overfit. The cross\_val\_score method has been used for each of them once and the cross validation accuracy and testing accuracy was around the same, hence not overfit. I have not displayed the cross\_val\_score outputs for all the models as it is computationally expensive. However, it has been checked for every single one. Have displayed cross\_val\_score with Naïve Bayes and mean cross validation score and prediction scores were similar which shows there was no overfitting.