**Lab Report 1**

Exercise 1 - Language Identification with sklearn

**Submitted by Saiteja Reddy Pottanigari – 19763093**

**Note: I have used a bit different terminology: Development Set is Training Set, Validation Set is Development set and Testing set is Test set.**

**Learning outcome:**

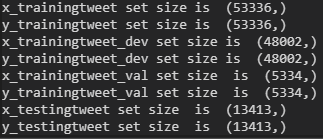
With regards to exercise 1’s part 1 as in the exercise\_part1.py file, I have used a single featured multiple hyperparameters Model. As this is a classification task, Here I have used the Stochastic Gradient Descent Classifier and Multinomial Naïve Bayes Classifier. In Part 2 , I have tried to predict the Tweet label using on multilayer perceptron classification model MLP Classifier.

Below are the detailed steps of my approach:

Data Pre-Processing:

1. Loaded the Data from .tsv files and .json file into Pandas DataFrame.
2. Since normal train Dataframe didn’t have tweet value, I had to merge them with Tweets.json.
3. Witnessing that there are very few samples for label, which may not contribute anything towards the training and will be actually affect negatively so I have deleted the tweets from training dataset which have less than 10 sample to train.
4. I made sure that there is only the label in Train sample are in Test sample.
5. After this I have split the Training sample to Development set (Dev) and Validation (Val) Set in 90:10 ratio proportionately using the stratified shuffle split.

Hence, the number of unique labels for classification are 36



**Part 1.**

Data Processing:

1. I have used Label Encoder for encoding the labels(y) with integer values.
2. CountVectorizer: So I have tokenized using one hot encoder technique count vector. Converted the text documents to a sparse matrix of token counts.
3. TF-IDF transformer: used this for converting the counted matrix into Term frequency -Inverse Document-Frequency. As raw frequency may not impact. But only used for SGDClassifier. Since MultinomialNB takes multinomial distribution which normally require integer feature counts.
4. Determining the features I was deciding the what is the way I have to vectorize the labels in unigram or bigram or trigram.
5. I have decided the arguments of individual transformer based on the GridSearchCV performed on both SGDClassifier and MutlinomialNB

After that by using GridSearchCV, I finalized

**CountVectorizer:** NGRAM\_RANGE between 1 to 3 was potentially better and with analyzer with CHAR\_WB.

**TfidfTransformer**: SMOOTH\_IDF: False

**MutlinomialNB:** FIT\_PRIOR: False

**SGDClassifier**: loss='MODIFIED\_HUBER', penalty='l2', FIT\_INTERCEPT=False, MAX\_ITER=1500.

(Modified Huber is a different Loss function specific for Classification problems than Huber Los function)

**Results:**

|  |  |  |
| --- | --- | --- |
| **Metrics** | **Multinomial Naïve Bayes** | **SGD Classifier** |
| **Development Set Accuracy** | **84.27%** | **95.86%** |
| **Validation Set Accuracy** | **81.96%** | **91.11%** |
| **Test Set Accuracy** | **82.34%** | **91.41%** |
| **F1\_SCORE(weighted)** | **79.01%** | **90.90%** |
| **F1\_SCORE(Micro)** | **82.34%** | **91.41%** |
|  |  |  |
|  |  |  |

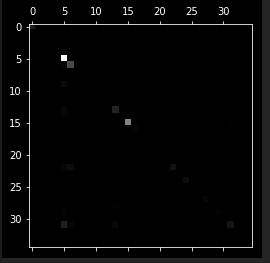
1. **Multinomial Naïve Bayes:**

Best Model:

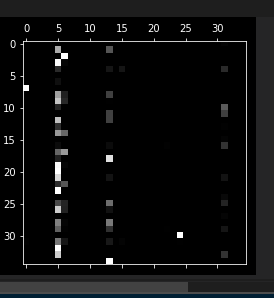
text\_nbclf = Pipeline([('vect', CountVectorizer(ngram\_range=(1,3),analyzer='char\_wb')),('tfidf', TfidfTransformer(smooth\_idf=False)),('nb\_clf', MultinomialNB(fit\_prior=False))])

text\_nbclf.fit(x\_trainingtweet\_dev,y\_dev\_trainingtweet)

The ***confusion matrix*** obtained for the Multinomial NB in the grayscale colormap is shown below:



Error analysis in the grayscale color map as shown below:



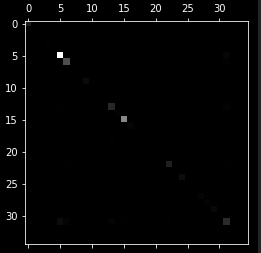
1. **SGD Classifier:**

**Best Model :**

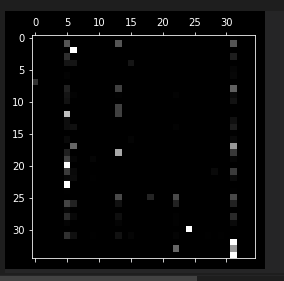
text\_sgdclf = Pipeline([('vect', CountVectorizer(ngram\_range=(1,3),analyzer='char\_wb')),('tfidf', TfidfTransformer(smooth\_idf=False)),('sg\_train', SGDClassifier(loss='modified\_huber',penalty='l2',fit\_intercept=False))])

text\_sgdclf.fit(x\_trainingtweet\_dev,y\_dev\_trainingtweet)

The ***confusion matrix*** obtained for the SGD Classifier in the grayscale colormap is shown below:



Error analysis in the grayscale color map as shown below:



**Comparison – Multinomial NB Vs SGD Classifier**

The best accuracy amongst the two classifiers is obtained for SGD Classifier with accuracy being close to 92% while for Multinomial NB Classifier it is close to 83%.

*Why SGD Classifier is better?*

* There is no gradient descent to perform in the Multinomial Naïve Bayes (MNB). This implementation just uses relative frequencies counts (with smoothing) to find the parameters. Hence, the simple MNB performs lower than the SGD Classifier, which by default, fits a linear support vector machine (SVM).
* We can perform balancing using class\_weight function which is better to classification problems where the

Advantage of Grid Search Cross Validation:

Grid search cross validation generalizes the results across training and test set combination and try to minimize the effect of overfitting. Grid search is used to find the optimal *hyperparameters* of a model which results in the most ‘accurate’ predictions.

It can be seen easily that the number of True Positives in the confusion matrix of SGD (541) is more than that of the MNB (530) for the label 1. That means that the SGD Classifier performs better than the MNB and it is further justified by the test accuracy.

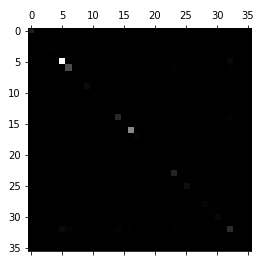
**Part 2.**

**Multilayer Perceptron (MLP)**

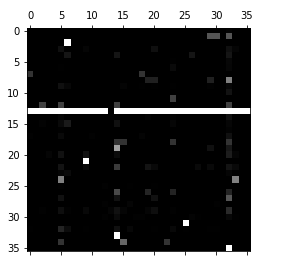
**Accuracy achieved:**

|  |  |
| --- | --- |
| **Metrics** | **Multi-Layer Perceptron** |
| **Development Set Accuracy** | 99.93% |
| **Validation Set Accuracy** | 90.4762 |
| **Test Set Accuracy** | 90.1737 |
| **F1\_SCORE(weighted)** | 90.113 |
| **F1\_SCORE(Micro)** | 90.1737 |
|  |  |
|  |  |

The grayscale color map of Confusion matrix is shown below:



Error analysis matrix:



***Note:*** *The Grid Search implementation for MLP has been commented out for code-time running purpose. The reviewer is welcome to uncomment it and run it.*

End of Report