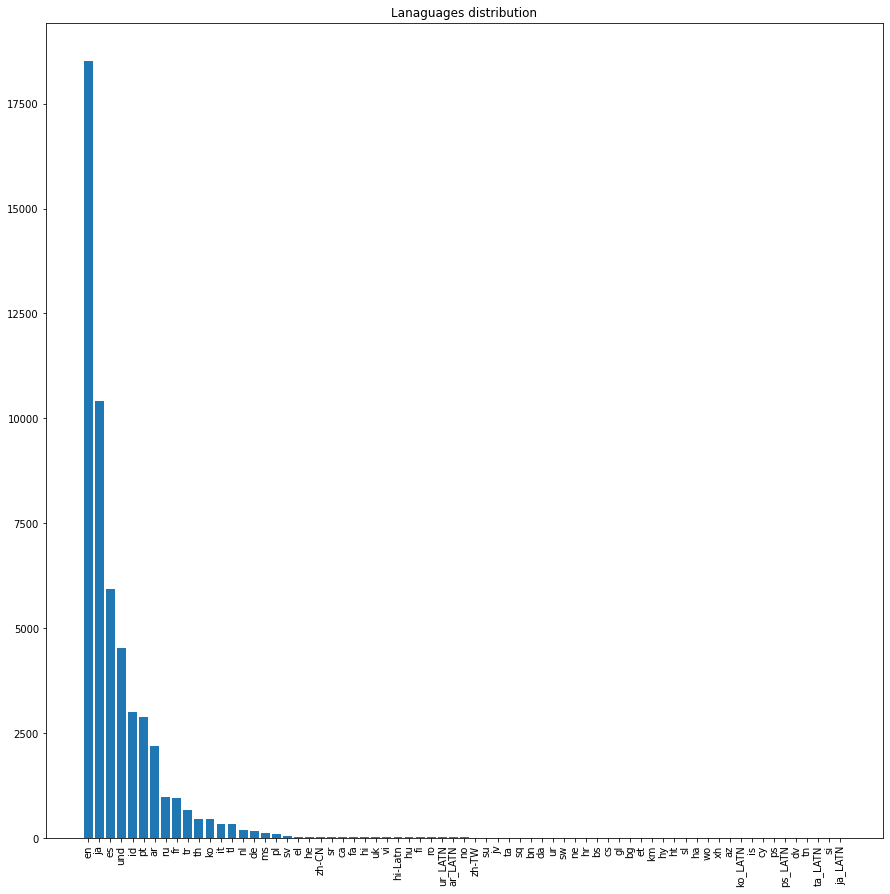
Machine Learning for Natural Language Processing 1

Exercise 1: Language Identification with sklearn

**Data Distribution**

The training dataset used consisted of Class imbalanced data consisting of 52675 tweets with 69 different language identification labels. The data was was heavily skewed towards the following languages (top five listed below):

| **Sl No.** | **Language** | **As Percentage of all tweets in dataset (%)** |
| --- | --- | --- |
| 1 | English | 35.14% |
| 2 | Japanese | 19.79% |
| 3 | Spanish | 11.26% |
| 4 | ‘und’ - undetermined | 8.61% |
| 5 | Indonesian | 5.71% |



It was observed that some tweets consisted of multiple different languages which were mapped to a single label which could lead to errors when generating predictions using the test dataset.

**Dataset Description**

The dataset was already pre-partitioned into training and test sets . The training dataset had 52675 records and the test dataset consisted of 13279 records. There was one column for the content of each tweet and a second containing a language label.

### Part 1 - Language identification with linear classification

#### Design features

A number of relevant data features, chiefly linguistic, were appended to dataset to improve the prediction accuracy and incorporated into the classification model. These are listed below:

* Sequences of adjacent words such as unigrams(one-word) and bigrams(two-words)
* Character trigram (sequence of three consecutive letters)
* Number of spaces between words per tweet
* Average number of words in a tweet per sentence.

#### Pipeline Setup for Logistic Regression and Grid Search

##### Preprocessing and features combination

The pipeline starts with converting words to lowercase. This is crucial as we do not want the number of features (generated by the dictionary) to expand unnecessarily. Emojis were removed from the text of the tweet as these convey emotions which are more are less universal, and thus are not particularly helpful in determining distinction between languages - thus having very limited scope for the purpose of language identification. URLs included in the text of a tweet were similarly excluded as these are written in Latin Alphabet without accents and are the same across languages the tweet might be written in. Next, average number of words per sentence, number of spaces per tweet were added to the data pipeline. Similarly, Word and Character vectors were also added to the pipeline.

To account for the possibility that some tweets in the test dataset are in a language not indexed in the training dataset, we dropped each instance of a tweet in a language not previously accounted for in the training dataset.

##### Parameters settings

For the Logistic Regression model, we used the solver: stochastic average gradient ascent (saga) as it was recommended by sklearn for multi-classification problems.

For Grid Search, we consider saga and lbfgs. For penalty, we take L1, L2, and none, meaning no penalty term as parameters. We thought it might be interesting to see how the performance of the model varies for a given penalty. A 10-fold grid search was used so that the training data is split into a 90 to 10 ratio of training and validation set. The grid search will run the models 10 times for every combination of parameters.

#### Result from logistic regression

Model: A standard logistic regression model was used to classify the language in which the text contained in a tweet might be written. Some of the features used were numerical (discrete continuous), such as average number of words per sentence, whereas others were categorial. A prediction accuracy of 87.13% was achieved with the training dataset.

#### Result from GridSearchCV

A prediction accuracy of 87.31% was achieved using the test set.

##### Advantage of Gridsearch for Cross-Validation

Grid search is a method for performing hyper-parameter optimization for a given model (eg.logistic regression) and test dataset, which involves determining the optimal combination of hyper-parameters for the different models generated. Each of these parameter combinations that correspond to a single model is said to lie on a “grid” point. Our goal is to eventually train and evaluate each of these models using a technique such as statistical cross-validation, and then choosing one which performs the best as per the relevant criteria. Specifically, GridSearchCV tries all the combinations of the values passed in the dictionary and evaluates the model for each combination using the Cross-Validation method. When we use this function we get an accuracy/loss metric for every combination of hyperparameters and we can then choose the one with the best performance.

##### Optimal Parameter Setting

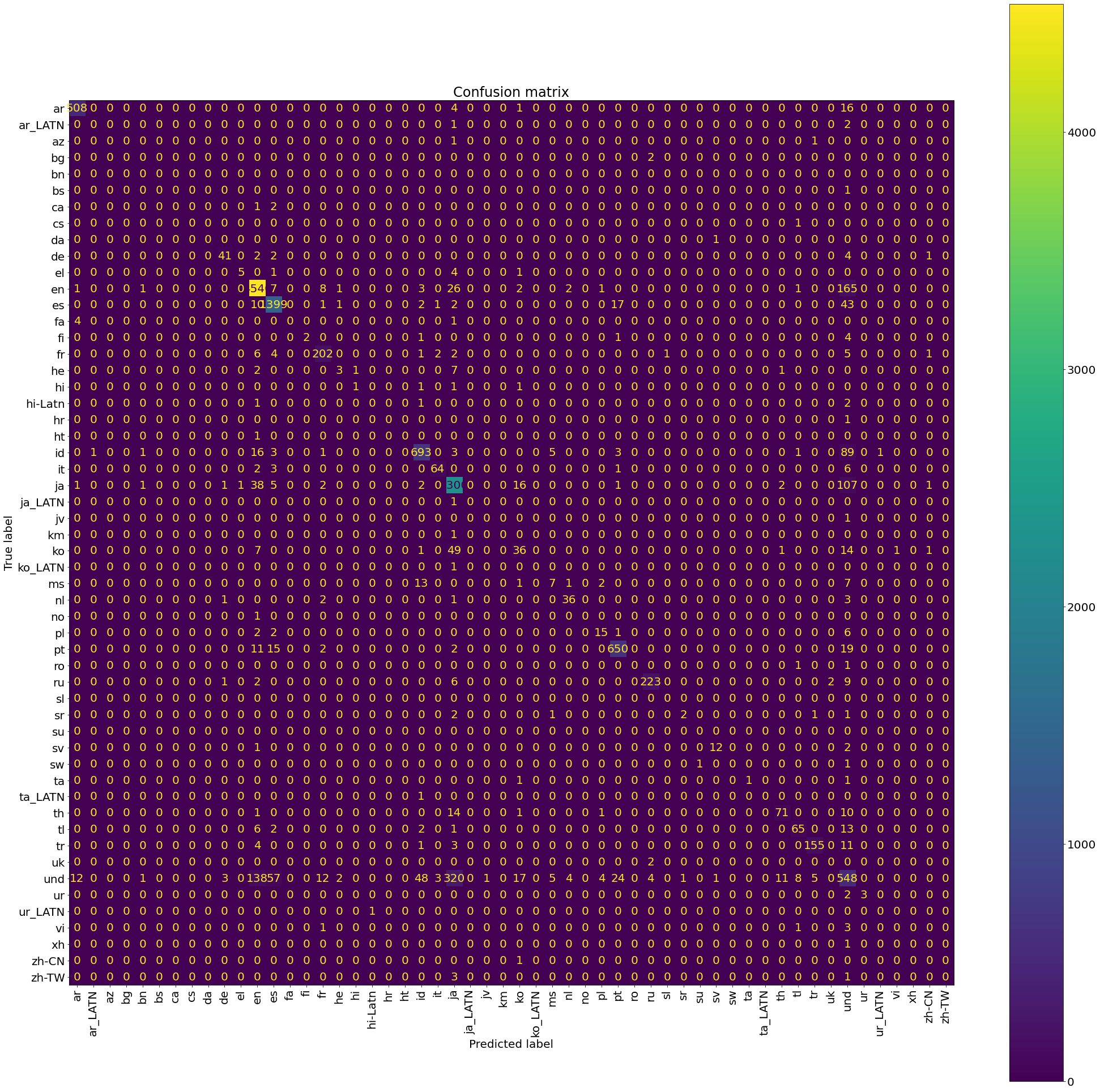
From the model we obtained, it is best to set lbfgs as the solver algorithm and the penalty to none. As we do not have a uniformly distributed number of tweets across languages, in certain cases where less data is available in the training set for tweets in certain languages- it is easy to misclassify labels. It becomes worse when working with the test dataset, which typically has less data in general compared to the training set, and leads to models suffering heavily from the regularizer penalty.

##### 

##### Confusion matrix

Generally the model predicts well. The correct classification along the diagonal line are usually the highest in each row or column. There is however, a large spike in misclassification, of the tweets of the type(language) ‘und’ being predicted as ‘en’. This might be understandable, if we account for the possibility that many tweets with the label ‘und’ are actually in English ‘en’.

An interesting case of the confusion matrix occurs for the true label ‘ms’ which indicates Malay, but the predicted labels tend to steer to ‘id’ (Indonesian). Only seven tweets written in Malay are correctly predicted. The rest falls to Indonesian (twenty tweets) and undetermined. This might make sense because in the training data, we have more tweets in Indonesian than Malay, and the model adapts to predict Indonesian well. In addition, both languages share linguistic characteristics from the same family of languages - sharing many words and similarities in character bi/tri-grams.



##### 

##### Feature importance table (top 10) for English, Spanish and Japanese

From the result of the top 10 features, it can be seen that the tweet vector is more important than additional languages. This is expected as each language has unique words/terms and combination of characters. However, feature extension is also needed. For example, the number of spaces per tweet is a helpful factor to identify languages such as Arabic where the presence of spaces is not to be found.

English is characterized by the use of bigrams such as th, ed, ‘s, etc. Whereas Spanish has y, os, el,etc.

| **#** | **English (‘en’)** | **Spanish(‘es’)** | **Japanese(‘ja’)** |
| --- | --- | --- | --- |
| 1 | th | y | ました |
| 2 | ed | os | ちゃん |
| 3 | ‘s | el | かった |
| 4 | yo | jaj | ってる |
| 5 | amen | que | だから |
| 6 | test | as | ション |
| 7 | same | hola | いです |
| 8 | i’m | bue | みたい |
| 9 | that | aja | います |
| 10. | nig | las | まとめ |

### Part 2 - Multilayer Perceptron (MLP)

To build the model, we use the existing pipeline from grid search above. In the table of comparison below, the pipeline is indicated by the term ‘normal pipeline’.

We chose early termination for MLP for training dataset as true because of the following reasons:

1. Sklearn builds MLPClassifier to split train data into training and validation datasets, which means we don’t have to split data manually.

2. If the validation loss does not improve after a few epochs, the model stops training and updates weight. This is beneficial as it prevents the model from overfitting (from updating weights unnecessarily) and consumes less time.

Relu activation is better than sigmoid and tanh activation function as it can prevent gradient vanishing problems in deep neural networks. Therefore, all models were implemented using Relu.

We first built MLP1 with 2 layers, each of which had 100 neurons, but discovered the accuracy was very low. We speculated that having 1 less hidden layer would perhaps help and we incorporated this change in MLP2, however the result was the same with no improvement in accuracy.

Subsequently, we decided to change the solver algorithm from SGD to Adam, as the latter has a momentum term, which would result in the model being less likely stuck at a local minima. Our intuition was proven right as MLP3 had a much higher accuracy score (88.59% instead of 35.87%). MLP4 used 2 hidden layers but with fewer neurons, and with a solver algorithm set to Adam to see if perhaps there was an improvement in the accuracy of the model. The performance however did not change by much - in fact we noticed a very small decrease from 88.59% to 88.36%.

MLP5 was similar in construction to MLP3 but this time included both bigram and trigram characters from tweets instead of just trigram. The accuracy of this was observed to be the best among the models, but only by a small amount compared to MLP3 and MLP4

| Model | MLP1 (2 mins) | MLP2 | MLP3 | MLP4 | MLP5 |
| --- | --- | --- | --- | --- | --- |
| Number of hidden layers | 2 layers  100 neurons each | 1 layer  100 neurons | 1 layer  100 neurons | 2 layers  75 neurons each | 1 layer  100 neurons |
| Solver | SGD | SGD | Adam | Adam | Adam |
| Activation | Relu | Relu | Relu | Relu | Relu |
| Pipeline | Normal pipeline | Normal pipeline | Normal pipeline | Normal pipeline | Including both characters bigram and trigram |
| Running time | 2 mins | 2 mins | 4 mins | 3 mins | 4 mins |
| Accuracy | 35.87% | 35.87% | 88.59% | 88.36% | 88.62% |