Homework 4: Using pre-trained BERT vectors for text classification

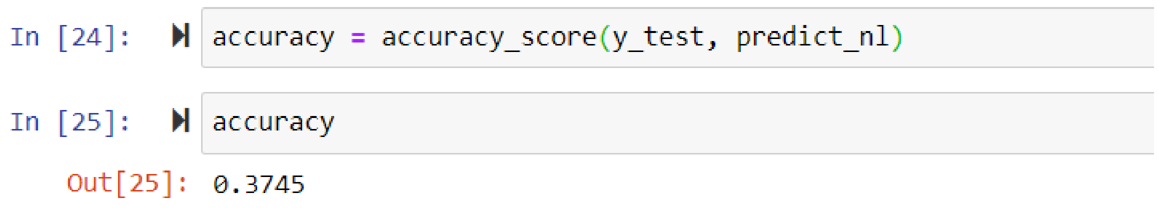
Part I: Steps to run the code

1. Make a project directory as given below
   --Project
      --BERT_BASE_DIR
      (*BERT repository is stored here)*
      --BERT_DATA_DIR
      (*Given three input csv files are stored here)*
      --bert_input_data
      (*Copy the three input csv files here to process by the BERT)*
      --bert_output_data
      (*The jsonlines files are generated by shell script and stored here)*
      --models
      (*Pretrained model is stored here)*
2. Open the python file (BERT_TextClassification.ipynb) and edit the paths to the files and folders as required and as mentioned in the python file.
3. Run the code in each cell either by clicking *Run* or by pressing *shift+enter.*

Part II: Evaluation Findings

1. **Model overall performance**
   We calculate the overall performance of the model by using *accuracy_score( )* function from *sklearn* package. The accuracy of the model is calculated as a fraction of correct predictions by the model to the total number of predictions.

$$accuracy = \frac{correct\ predictions}{total\ number\ of\ predictions}$$



```
In [24]:   accuracy = accuracy_score(y_test, predict_nl)

In [25]:   accuracy

  Out[25]: 0.3745
```

We get an accuracy of 37.45% on our model. As the accuracy score independently does not tell about the performance of a model, for that we need to calculate precision, recall and f1 score. These measures provide an accurate performance of the model even if the data is skewed.

2. **Metric by class**
   To calculate these measures I used *classification_report( )* function from *sklearn* package. This report provides the metrics by class information.

   Precision: The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

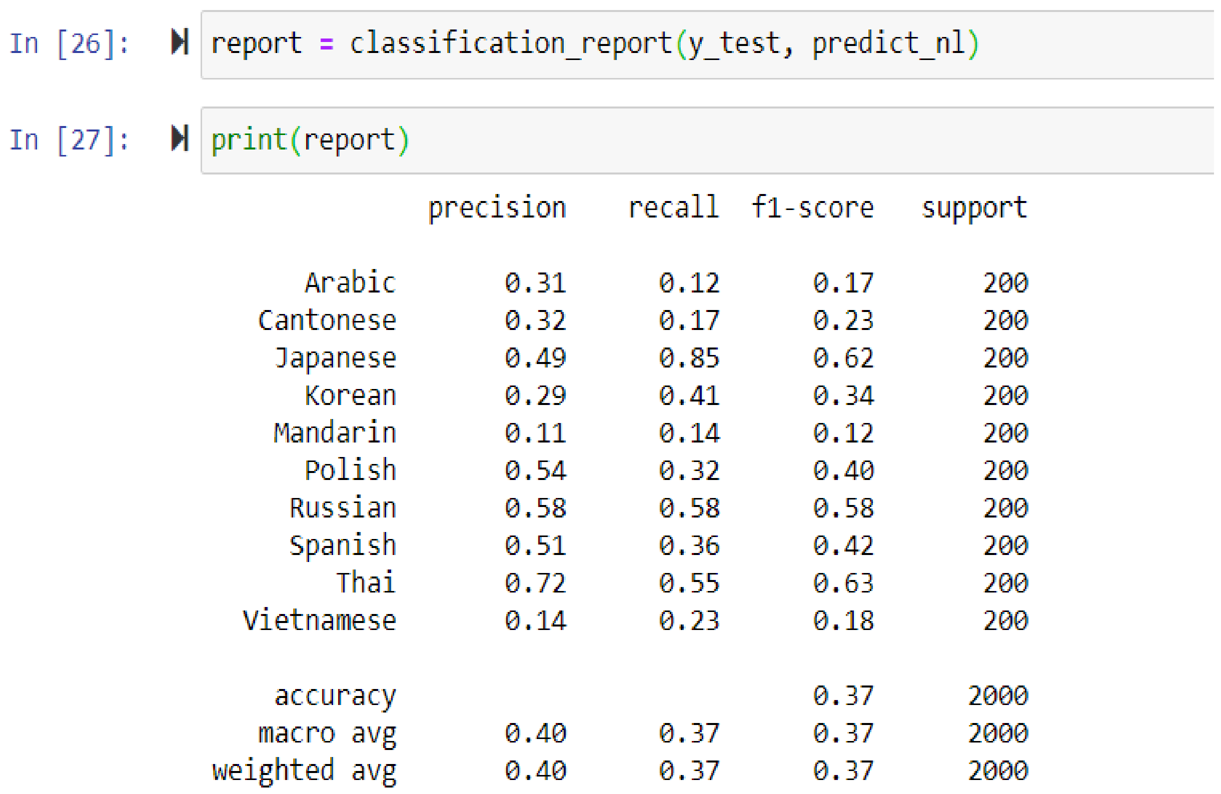
$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

   Recall: *The recall is intuitively the ability of the classifier to find all the positive samples.*

$$recall = \frac{true\ positives}{true\ positives + false\ negetives}$$

   F1 Score: The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0.

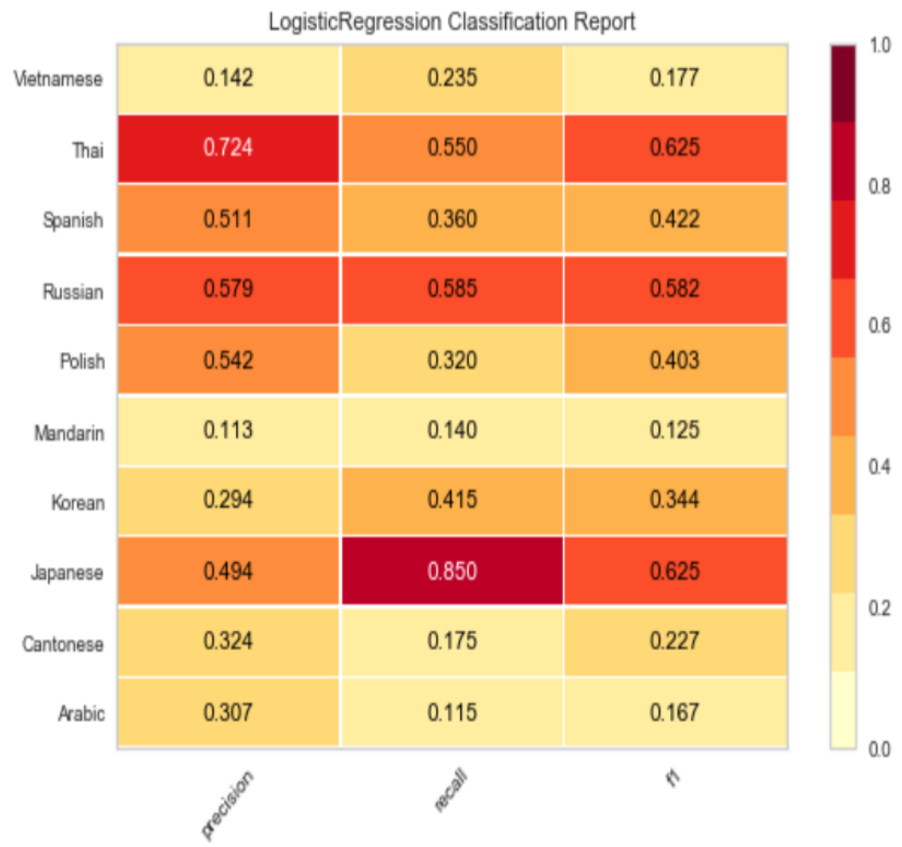
$$F1\ score = \frac{2 * precision * recall}{precision + recall}$$



```
In [26]:   report = classification_report(y_test, predict_nl)

In [27]:   print(report)

               precision    recall  f1-score   support

      Arabic       0.31      0.12      0.17       200
   Cantonese       0.32      0.17      0.23       200
    Japanese       0.49      0.85      0.62       200
      Korean       0.29      0.41      0.34       200
    Mandarin       0.11      0.14      0.12       200
      Polish       0.54      0.32      0.40       200
     Russian       0.58      0.58      0.58       200
     Spanish       0.51      0.36      0.42       200
        Thai       0.72      0.55      0.63       200
  Vietnamese       0.14      0.23      0.18       200

    accuracy                           0.37      2000
   macro avg       0.40      0.37      0.37      2000
weighted avg       0.40      0.37      0.37      2000
```

From the above report we can find that the model prediction for the Thai native language is greater followed by Japanese and Russian as their F1 score is greater among all other native languages.

Also, we can find that Thai has maximum precision which makes its F1 score greater and we can observe that the precision value for Japanese is smaller as compared to Russian and Polish but its recall value is higher than those which makes its greater F1 score.



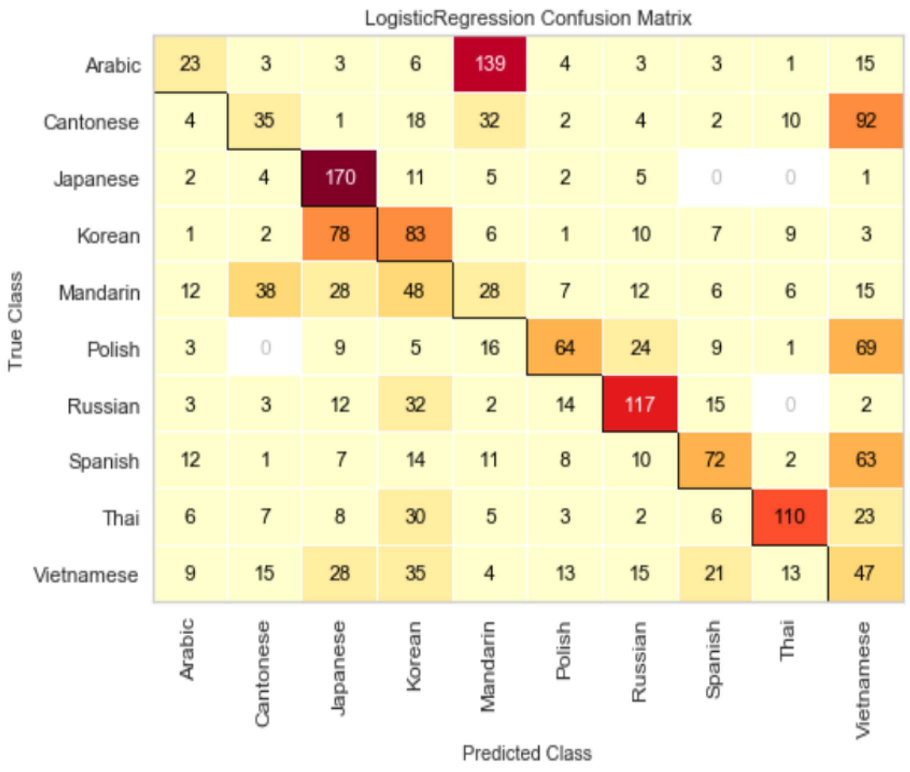
	precision	recall	f1
Vietnamese	0.142	0.235	0.177
Thai	0.724	0.550	0.625
Spanish	0.511	0.360	0.422
Russian	0.579	0.585	0.582
Polish	0.542	0.320	0.403
Mandarin	0.113	0.140	0.125
Korean	0.294	0.415	0.344
Japanese	0.494	0.850	0.625
Cantonese	0.324	0.175	0.227
Arabic	0.307	0.115	0.167

3. **Frequency of errors between classes**
   The frequency of errors between the classes is obtained by using *confusion_matrix( )* function from *sklearn* package. This function provides the distribution of predictions of particular class within other class. This information helps to identify the number of correct and incorrect predictions made by the model.

In [30]:

```
from sklearn.metrics import confusion_matrix
con_matrix = confusion_matrix(y_test, predict_nl)
print(con_matrix)
```

```
[[ 23   3   3   6 139   4   3   3   1  15]
 [  4  35   1  18  32   2   4   2  10  92]
 [  2   4 170  11   5   2   5   0   0   1]
 [  1   2  78  83   6   1  10   7   9   3]
 [ 12  38  28  48  28   7  12   6   6  15]
 [  3   0   9   5  16  64  24   9   1  69]
 [  3   3  12  32   2  14 117  15   0   2]
 [ 12   1   7  14  11   8  10  72   2  63]
 [  6   7   8  30   5   3   2   6 110  23]
 [  9  15  28  35   4  13  15  21  13  47]]
```



From above visualization we can find out how the predictions made by the model are scattered among all other classes (native languages). For example the prediction for Japanese native language is scattered as 2 predictions in Arabic, 4 in Cantonese, 170 in Japanese, 11 in Korean, 5 in Mandarin, 2 in Polish, 5 in Russian, 0 in Spanish & Thai and 1 in Vietnamese from total predictions made on Japanese i.e. 200.

4. Analysis and Conclusion

From above matrix we can calculate the error frequency for each class and find out that Japanese, Russian and Thai have highest number of correct predictions.

We can try different classifiers i.e. Naïve Bayes, Decision Tree, k.NN, Neural Network etc. for our classification problem and compare there accuracy with each other, it might be possible that Neural Networks produce greater accuracy than our Logistic Regression model.
As there are Ten classes in which we have to classify our input (written English sentences) we need more number of data sets, so it is possible that we can increase the accuracy of our model by increasing the number of instances of each class i.e. native languages.

In conclusion, the implementation of Logistic Regression model for classification of native languages with BERT vectors is done with accuracy of 37.45%. The F1 score and confusion matrix metrics is obtained and observed that Japanese, Russian and Thai languages are predicted well as compared to other languages.