Tad Coursework 2021 May June - Post Sentiment Classification Task

Name - Ruchika Singhi.
Date - 04/06/2021
Student Number – 2576183S
https://colab.research.google.com/drive/1SsD7Z2Te39DyGYyJd517eYL2JnLmmDG3?usp=sharing

Question 1

Explore Dataset

- This is a single label-multi-class classification Task
- As seen from the Fig. 1 bar chart most (~63% of the labels are neutral overall in the training data set). About 27% are positive. This means we have an issue of class imbalance.

- But we can see Fig. 2 hat all the train/validate and test data all have about 63% neutral labels, 27 % positive and 0-1% very-negative classes. This is the same percentage as the overall collection. Hence, the data isn't biased.

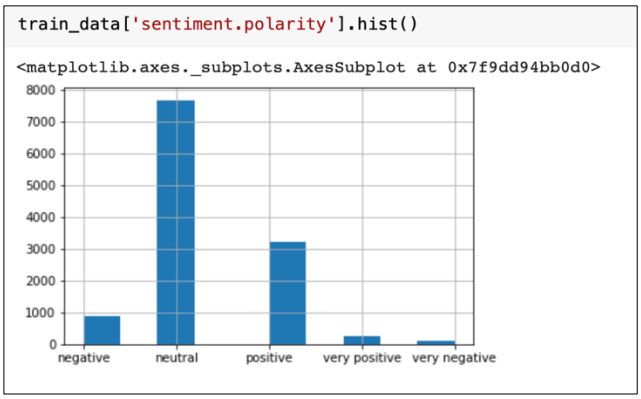


Fig 1 Distribution of labels in training data

```
----Training set----
Total posts - 12138
Fraction of neutral posts -  63%
Fraction of very negative posts -  1%
Fraction of positive posts -  27%
----Validation set----
Total posts - 3109
Fraction of neutral posts -  63%
Fraction of very negative posts -  0%
Fraction of positive posts -  27%
----Test set----
Total posts - 4016
Fraction of neutral posts -  63%
Fraction of very negative posts -  1%
Fraction of positive posts -  27%
```

Fig 2. Distribution of 3 labels in train/validate/test data sets

Discussion on Classifiers Performance

- Comparing the results with the dummy classifiers, other classifier's performance can be best described as "good". The 'most-frequent' classifier gets an accuracy of about 62.6% and F1 of about 0.77. This shouldn't be surprisingly, because that's the percentage of the data that is labelled 'neutral'.
- The stratified class has an F1 of 0.48 and an accuracy 0f 47.9%. So, other classifiers are better than random guessing.
- F1 score is a better measure of performance because class in imbalanced.
- Logistic Regression with one-hot vectorization overfits on training data.It does great on training and comparatively bad on testing. It is obvious as on sparse data LR overfits, Appropriate Regularization techniques can resolve this. The key parameter solver is set to 'saga' as it can handle multinomial loss.
- Logistic Regression with TF-IDF vectorization, generalises well on unseen tests data and has good F1 score of 0.779 compared to LR with one-hot vectorisation of 0.759 with same parameter settings.
- SVC Classifier with RBF Kernel has the highest F1(weighted) as seen it nearly outperforms Logistic Regression models. But this model also needs regularization and need

to tune kernel parameters for improved performance as it is sensitive to imbalanced dataset.

	Accuracy	Precision	Recall	F1(macro)	F1(weighted)
Dummy Classifier (most_frequent)	0.626	0.200	0.125	0.154	0.770
Dummy Classifier(stratified)	0.479	0.202	0.202	0.202	0.480
LogisticRegression (One-hot vectorization)	0.744	0.436	0.640	0.488	0.759
LogisticRegression (TF-IDF vectorization)	0.739	0.328	0.583	0.353	0.779
SVC Classifier	0.728	0.288	0.458	0.286	0.781
Decision Tree classifier	0.691	0.488	0.483	0.485	0.692

Fig 3. Overall results obtained by the classifiers on the Test sets

	Accuracy	Precision	Recall	F1(macro)	F1(weighted)
Dummy Classifier (most_frequent)	0.633	0.200	0.127	0.155	0.775
Dummy Classifier(stratified)	0.472	0.198	0.198	0.198	0.473
LogisticRegression (One-hot vectorization)	0.964	0.848	0.982	0.904	0.965
LogisticRegression (TF-IDF vectorization)	0.817	0.409	0.916	0.461	0.845
SVC Classifier	0.862	0.425	0.751	0.457	0.887
Decision Tree classifier	0.669	0.233	0.285	0.218	0.758

Fig 4. Overall results obtained by the classifiers on the Training sets

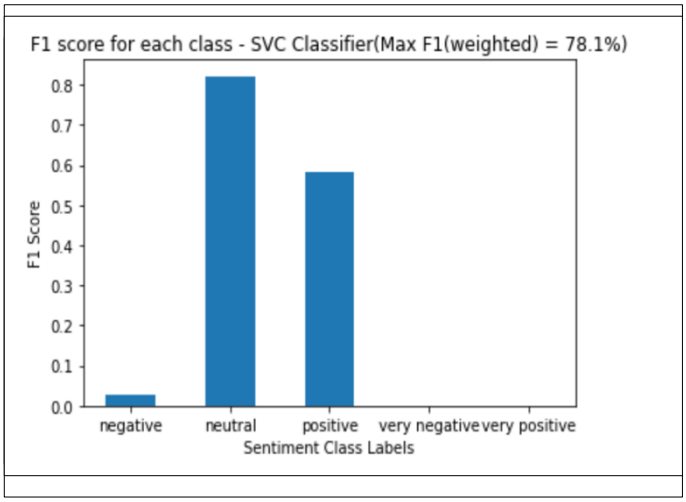


Fig 5. best performing classifier (by weighted F1 in test set

Choose your own classifier

The classifier chosen was Decision Tree Classifier with parameters random_state = 0 to nullify the randomness and max_depth= 2, to prevent the classifier from overfitting. The vectorization chosen is TfidfVectorizer (). This model, does not generalise well on Test data and F1 score on training data is greater than that on test data, this is a scenario of underfitting

It's interesting to create this model, but its less accurate than other classfiers and has low F1 compared to other 4 classifiers.

Question 2

- Initial: Before Parameter tuning , Validation Data metrics

```
Classifier 'LR' has Acc=0.731 P=0.320 R=0.564 F1=0.344
```

Parameters	**Values**	**Best Parameter**	**F1 score**	**Accuracy**	**Description**
			Validation data		
Vectorizer: **sublinear_tf**	True, False	True	0.348	0.731	Analysing one parameter at a time seeing if it increases the performance.
Vectorizer:	10,100,1000,	1000	0.383	0.735	A good parameter, increases f1 score

max_features	10000,20000				
Vectorizer: **max_features sublinear_tf**	(10,100,1000, 10000,20000), (True, False)	1000, True	0.396	0.743	Combining both params increase the f1 score, model is performing better
Classifier **Regularizatio n** C	1e-3,1e-2,1e-1,10,100,1000, 10000,1e5	10	0.461	0.75	Regularization alone has increased the model performance
Classifier **Regularizatio n** C Vectorizer: **max_features sublinear_tf**	(10,100,1000) (10,100,1000, 10000,20000) (True, False)	10 10000 True	0.48	0.749	Combining all three params to look for more better performance, and yes it increases
Selected the following parameters for improved pipeline **Regularization** C = 10, **max_features =1000, sublinear_tf = True**			0.515	75.6	So, selecting the above settings,but model performs better with 1000 max-features rather than 10000
Classifier class_weight =	('balanced', 'dict')	Extremely slow, did not converge	0.24	0.302	Extremely slow,Not considering it
Vectorizer: strip_accents	('ascii', 'unicode')	unicode	0.515	0.756	All these parameters had almost no impact on the classifier performance.
Vectorizer: analyzer	('word', 'char')	word	0.515	0.756	
Classifier random_state	(0,5,10,15,20,25,30)	0	0.515	0.756	
selected parameters for final pipeline **Regularization** C = 10, **max_features =1000, sublinear_tf = True,** random_state = 0,			0.515	0.756	Overall increased in the model performance acheived

Results on Test Data: Before parameter tuning

```
Classifier 'LogisticRegression (TF-IDF vectorization)' has Acc=0.739 P=0.328 R=0.583 F1=0.353
               precision    recall  f1-score   support

     negative      0.092     0.619     0.160        42
      neutral      0.945     0.736     0.827      3227
     positive      0.508     0.760     0.609       737
very negative      0.000     0.000     0.000         0
very positive      0.093     0.800     0.167        10

     accuracy                          0.739      4016
    macro avg      0.328     0.583     0.353      4016
 weighted avg      0.854     0.739     0.779      4016
```

After parameter tuning- Improved F1 – scores for individual class, the model is predicting better

```
Classifier 'Final LR' has Acc=0.756 P=0.460 R=0.611 F1=0.508
               precision    recall  f1-score   support

     negative       0.241     0.500     0.325       136
      neutral       0.887     0.782     0.831      2850
     positive       0.642     0.732     0.684       966
very negative       0.219     0.500     0.304        14
very positive       0.314     0.540     0.397        50

     accuracy                          0.756      4016
    macro avg       0.460     0.611     0.508      4016
 weighted avg       0.796     0.756     0.771      4016
```

Error Analysis – Possible ways error analysis could be have done –

1. Checking if model is under-fitting or over- fitting, by plotting training vs cross-validation score. It would have given us a step to perform either regularization or generate more data if it's over-fitting. Or add more features if its underfitting.
2. Checking for outliers, that would have caused imbalanced class predictions.
3. Over/under predictions can be solved by oversampling is data is less.

   There are more hypotheses that can be listed, basically error analysis is all about finding errors, creating hypothesis to fix the errors, test it, gain insights and repeat.

Question 3

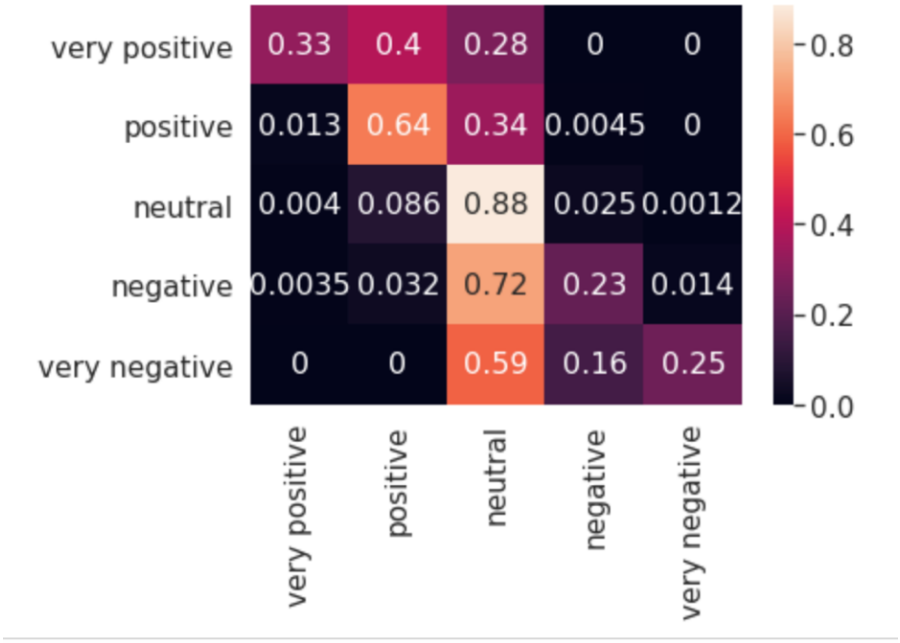
Proposed Features from Thread Properties	**Performance Analysis**		**Discussion**
	Validate (F1-Score Macro)	**Test (F1-Score Macro)**	
Subreddit	0.504	0.495	The thread property seem important because, it groups together similar reddit posts
Majority_Type	0.496	0.518	The majority_type , describes the nature of the post
Author	0.484	0.489	Selected this property because a person's style of posting can be unique
Title	0.399	0.439	A metadata about posts, that might have similar words as post

`Final features selected including body, subreddit, majority_type`

Validation data metrics on above proposed features:				
Accuracy	Precision	Recall	F1(macro)	F1(weighted)
0.755	0.449	0.595	0.496	0.771
Test data metrics:				
0.754	0.466	0.608	0.513	0.768

As seen the introduction of new feature hasn't improved the performance of the model. It's not beneficial. The search space is increased and the problem has become complex. Its clear that the features selected are not related much to the target labels, even though it seemed like they could have.Also, Parameter tuning could be done after feature selection, because data has become wider.

Well labelled confusion matrix: (Prediction on test_features vs test_labels)



This is a normalized confusion matrix. So many labels are mislabelled by the classifier, looking at the confusion matrix we can say that there are Less correct predictions. The regularisation can be improved, to predict more accurate classes.