EXERCISE 1 - LANGUAGE IDENTIFICATION WITH SKLEARN

Data

We generated the overview data report for both training data and test data using pandas-profiling. The main **distribution properties** are discribed below.

train_dev_set.tsv: Consists of 52675 pieces of data, each of which has two variables - tweet and label.

	tweet	label
0	يا من أناديها ويخنقني البكاء ويكاد صمت الدمع أن يتكلما يا قلبي الدامي وآه وأين ومن فاضت على عواطفاً وترحما	ar
1	فيه فرق بين اهل غزة اللى مطحونين من ناحيتين وبين حماس ؟ هنفهم ولا نبدا من ا ب ت	ar
2	عن اللحظة الحلوة اللي بتغمض فيها عينيك بتفكر ف حاجات حلوة بتتمناها وتفتح عينيك .. بضحكة جميلة . مع كلمة #يارب	ar
3	يا ابو سلو عرفتني	ar
4	ريال أكفل معتمر في رمضان ، ولك بإذن الله مثل أجر عمرته وتفطيره وصلواته بالحرم 50بhttp://t.co/jQoUiiVPjX	ar
5	التحميل لسامسونج ROM توجيه كيفية تثبيت البرامج الثابتة http://t.co/8QRpxfgYYW	ar
6	{وأنه هو أغنى وأقنى} [48:النجم] http://t.co/islmq7Ry0T	ar
7	اللهم قدر لنا الفرح بكل اشكاله ، انت الكريم الذي لا حدود لعطائه .. :")	ar
8	#غزه_تحت_القصف داعش أخواني حيل عندكم بالمدنيين نحر وجز رؤوس !	ar
9	{يعلمون ظاهرا من الحياة الدنيا وهم عن الآخرة هم غافلون} [7:الروم] http://t.co/dqpR1L0hpY	ar

There are no missing cells in this dataset, and the total size of the file is 11.5MB.

Dataset statistics

Number of variables	2
Number of observations	52675
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	53
Duplicate rows (%)	0.1%
Total size in memory	11.5 MiB
Average record size in memory	229.0 B

The tweets are string-type comments divided into 69 categories by the languages they are using, the top 3 languages are English, Japanese and Spanish. More about the category distribution can be found in the figure below.

Value	Count	Frequency (%)
en	18508	35.1%
ja	10421	19.8%
es	5930	11.3%
und	4537	8.6%
id	3006	5.7%
pt	2878	5.5%
ar	2199	4.2%
ru	978	1.9%
fr	946	1.8%
tr	669	1.3%
Other values (59)	2603	4.9%

test_set.tsv: Consists of 13279 pieces of data with the same variables of the training dataset. The file size is 2.9 MB and there are no missing values. Only 60 kinds of languages are showed in this dataset. The category distribution is showed in the figure below. We found it similar to the training dataset.

Value	Count	Frequency (%)
en	4758	35.8%
ja	2478	18.7%
es	1476	11.1%
und	1229	9.3%
id	817	6.2%
pt	699	5.3%
ar	529	4.0%
ru	243	1.8%
fr	224	1.7%
tr	174	1.3%
Other values (50)	652	4.9%

PART1 - LINEAR CLASSIFICATION

Step 1 - Generate a Pipeline

We first generated the preprocessing **pipeline**. It contains 3 parts: Dataloader, TweetCleaner and FeatureExtractor.

Pipeline

```
[ ]  from sklearn.pipeline import Pipeline

     preprocess = Pipeline(steps = [
         ('data_loader', DataLoader()),
         ('tweet_cleaner',TweetCleaner()),
         ('feature_extractor', FeatureExtractor()),
     ],verbose=True)
```

We used TF_IDF vectorizer to extract features. It aims to quantify the importance of a given word relative to other words in the document and in the corpus. We set the **feature space** to 500 and some of the features are showed below.

```
'tomorrow', 'tonight', 'too', 'true', 'try', 'tu', 'tweet',
'twitter', 'two', 'udah', 'um', 'uma', 'un', 'una', 'und',
'unfollowers', 'untuk', 'up', 'ur', 'us', 'va', 'vai', 'vamos',
'vc', 've', 'ver', 'very', 'via', 'vida', 'video', 'você', 'vou',
'voy', 'wait', 'wanna', 'want', 'was', 'watch', 'watching', 'way',
'we', 'week', 'well', 'were', 'what', 'when', 'where', 'who',
'why', 'will', 'win', 'wish', 'with', 'work', 'world', 'would',
'xd', 'ya', 'yang', 'yeah', 'year', 'years', 'yes', 'yg', 'yo',
'you', 'your', 'из', 'на', 'не', 'что', 'لله' ,'إلا' ,'أن',
'لـي' ,'لا' ,'كل' ,'قرآني' ,'في' ,'عن' ,'على' ,'تطبيق' ,'اللهم',
'يـا' ,'ولا' ,'من' ,'ما', '定期'], dtype=object)
```

Step 2 - Train the LR Model

We trained the Logistic Regression model and used GridSearchCV to find the best parameters. The combinations are recorded with predicting accuracy.

Penalty↓ Solver→	lbfgs	Newton-cg	sag	Saga
L2	0.62	0.62	0.62	0.62
None	0,58	0.58	0.57	0.58

As we can see, **the best result** comes from {Penalty=12, Solver=lbfgs}.

By using grid search cross-validation, we were able to create better fitting models by training and testing on all parts of the training dataset.

Step 3 - Visualize the Results

We calculated the confusion matrix to do error analysis. Part of the matrix is showed below.

```
[[322   0   0 ...   0   0   0]
 [  0   0   0 ...   0   0   0]
 [  0   0   0 ...   0   0   0]
 ...
 [  0   0   0 ...   0   0   0]
 [  0   0   0 ...   0   0   0]
 [  0   0   0 ...   0   0   0]]
```

To be specific, the first line of the matrix is:

```
array([[ 322,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    2,
           0,  203,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    2,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0],
```

We also generated the classification report with the precision, reall, f1 score and support.

	precision	recall	f1-score	support
ar	0.86	0.61	0.71	529
ar_LATN	0.00	0.00	0.00	3
az	0.00	0.00	0.00	2
bg	0.00	0.00	0.00	2
bs	0.00	0.00	0.00	1
ca	0.00	0.00	0.00	3
cs	0.00	0.00	0.00	1
da	0.00	0.00	0.00	1
de	0.00	0.00	0.00	50
el	0.00	0.00	0.00	11
en	0.66	0.76	0.71	4758
es	0.43	0.27	0.33	1476

We generated a feature importance table for the top ten features for English, Japanese and Spanish.

y=en top features		y=es top features		y=ja top features	
Weight?	Feature	Weight?	Feature	Weight?	Feature
+4.955	<BIAS>	+5.343	el	+6.757	<BIAS>
+4.694	you	+4.896	que	+3.326	定期
+4.672	that	+4.700	gracias	*... 8 more positive ...*	
+4.292	it	+4.634	las	*... 483 more negative ...*	
+4.176	for	+4.283	los	−2.689	mtvhottest
+4.172	just	+4.276	quiero	−2.746	da
+4.142	and	+4.034	una	−2.759	ya
+3.909	my	+4.003	voy	−2.789	la
+3.859	this	+3.894	estoy	−2.864	na
+3.852	what	+3.893	por	−2.873	haha
... 311 more positive ...		*... 119 more positive ...*		−3.064	me
... 180 more negative ...		*... 372 more negative ...*		−3.298	de

We noticed there seems to be something wrong with the Japanese features. The TfidfVectorizer cannot do the word tokenization for Japanese properly. To verify our observation, we also tried nltk library for word tokenizing, but we still found it not working.

PART 2 - MLP

We played around 5 different sets of hyper parameters, and the results are showned below.

Parameters	**Accuracy**
hidden_layer_sizes = (150,), solver = lbfgs, early_stopping = True	0.66
hidden_layer_sizes = (150,), solver = Adam, early_stopping = True	0.68
hidden_layer_sizes = (150,), solver = Adam, early_stopping = False	0.61
hidden_layer_sizes = (100,), solver = lbfgs, early_stopping = True	0.59
hidden_layer_sizes = (100,), solver = lbfgs, early_stopping = False	0.57

The results we got are slightly better than the LR model. The reason may be its ability to deal with nonlinear relations.

(Notice: we used the default parameter settings in our sample notebook, so the result may differ from this form. Also, we found it inconvenient to divide this program into two files, so we left it in one file for both lr and mlp:-))