Oracle PFE Internship 2024

ML Challenge : DeepTweets

Done by: Hind MOUTALATTIF

31 juil. 2023

Data Preparation :

In this task we have two datasets, the first one contains tweet text with its label which divides tweets into two classes Sports and Politics, the other dataset contains tweet text without a label and we need to build a machine learning model to get labels of this test data.

In order to build this machine learning model we will start by importing our train and test data:

Here we could see the components of our data:

```
train = pd.read_csv('train.csv') # importing train and test data
test = pd.read_csv('test.csv')
```

```
train.head(10)
```

	TweetId	Label	TweetText
0	304271250237304833	Politics	'#SecKerry: The value of the @StateDept and @U...
1	304834304222064640	Politics	'@rraina1481 I fear so'
2	303568995880144898	Sports	'Watch video highlights of the #wwc13 final be...
3	304366580664528896	Sports	'RT @chelscanlan: At Nitro Circus at #AlbertPa...
4	296770931098009601	Sports	'@cricketfox Always a good thing. Thanks for t...
5	306713195832307712	Politics	'Dr. Rajan: Fiscal consolidation will create m...
6	306100962337112064	Politics	FACT: More than 800,000 defense employees will...
7	305951758759366657	Sports	'1st Test. Over 39: 0 runs, 1 wkt (M Wade 0, M...
8	304482567158104065	Sports	Some of Africa's top teams will try and take a...
9	303806584964935680	Sports	'Can you beat the tweet of @RoryGribbell and z...

In this phase we are using an info function to see the types of our columns and make sure that there are no non_available cells in the train data.

```
train.info() ## getting info about our data to see if there is any r
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6525 entries, 0 to 6524
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   TweetId    6525 non-null   int64 
 1   Label      6525 non-null   object
 2   TweetText  6525 non-null   object
dtypes: int64(1), object(2)
memory usage: 153.1+ KB
```

Using the value_counts() function, we can see that our data is balanced and we have approximately same amount of data for each class:

```
train['Label'].value_counts() ## here we see that our data is balenc
```

```
Sports      3325
Politics    3200
Name: Label, dtype: int64
```

Machine learning models don't understand text data so we need to convert our label classes into numerical data using the map() function:

```
train['Label']=train['Label'].map({'Sports':0, 'Politics':1})
```

This is the new label that we got after the conversion:

```
train['Label']
```

```
0       1
1       1
2       0
3       0
4       0
       ..
6520    1
6521    0
6522    0
6523    0
6524    1
Name: Label, Length: 6525, dtype: int64
```

Feature extraction

Feature extraction is a crucial step in building a machine learning model. Here we will convert the text into a numerical format using the term frequency-inverse document frequency (Tf-Idf) to represent each tweet:

```
# we split data
X = train['TweetText']
y = train['Label']
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(X)
```

After vectorizing our data we need to split it into two parts one for the training of ml model and the other part is for testing this model:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Model Training:

We start our model training using support vector machine, which is more efficient for binary classification problems:

Here we train the model using training data:

```
svm_model = SVC()
svm_model.fit(X_train, y_train)
```

```
SVC()
```

Hyper parameter tuning:

For hyper parameter tuning we will use Grid Search that will give the best and optimal set of parameters for our model:

```
from sklearn.model_selection import GridSearchCV
param_grid = {'C': [1, 10, 100], 'kernel': ['linear', 'rbf']}
grid_search = GridSearchCV(svm_model, param_grid, cv=5)
grid_search.fit(X_train, y_train)
best_params = grid_search.best_params_
```

```
best_params
```

```
{'C': 1, 'kernel': 'linear'}
```

Performance:

In this step we try to evaluate the model's performance on the testing set by getting its predictions, and calculating the accuracy:

```
from sklearn.metrics import accuracy_score

y_pred = svm_model.predict(X_test)
accuracy = accuracy_score(y_test , y_pred)
```

```
accuracy
```

```
0.9484167517875383
```

Improvements:

To improve the performance of our work, we will test a set of machine learning models to select the best model at the end :

Here we import the needed models:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn import svm
```

In the next step we will create a function that will train each model with the data and will calculate the predictions of test data to give us a final report containing the accuracies of each model:

```
def model_training(models):
    for element in models:
        model = models[element]
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        print("the accuracy of "+element+" is: {:.2f}%".format(accuracy_score(y_test , y_pred)*100))
```

```
model_training(our_models)
```

```
the accuracy of Logistic Regression is: 93.97%
the accuracy of Decision Tree is: 84.32%
the accuracy of Nearest Neighbors is: 92.34%
the accuracy of Random Forest is: 91.01%
the accuracy of XGBClassifier is: 89.99%
the accuracy of SVM is: 94.84%
```

Here we could see that the best model that gives the bigger accuracy is the Support vector machine.

Other Possible Approaches for Improvement:

Advanced Text Vectorization Techniques: Instead of using TF-IDF, we could use word embeddings like Word2Vec or pre-trained language models like BERT to represent the text data.

Preparation of the submission dataset :

Here we will use the test set that contains tweet text without a label:

```
test
```

	TweetId	TweetText
0	306486520121012224	'28. The home side threaten again through Maso...
1	286353402605228032	'@mrbrown @aulia Thx for asking. See http://t....
2	289531046037438464	'@Sochi2014 construction along the shores of t...
3	306451661403062273	'#SecKerry\u2019s remarks after meeting with F...
4	297941800658812928	'The #IPLauction has begun. Ricky Ponting is t...
...	...	...
2605	282023761044189184	'Qualifier 1 and Eliminator games will be play...
2606	303879735006601216	@reesedward Hi Edward, it's not a #peacekeepin...
2607	297956846046703616	'Perera was @SunRisersIPL first #IPL purchase ...
2608	304265049537658880	'#SecKerry: Thanks to Senator @TimKaine, @RepR...
2609	306430391928115200	Here's a picture from our official Pinterest a...

2610 rows × 2 columns

Next, we need to vectorize our data to get predictions:

```
Z=test['TweetText']
test_vectorized = vectorizer.transform(Z)
```

```
test_vectorized
```

```
<2610x16193 sparse matrix of type '<class 'numpy.float64'>'
        with 30777 stored elements in Compressed Sparse Row format>
```

Here we get the predicted data of test using the best model SVM:

```
test_prediction=svm_model.predict(test_vectorized)
```

```
test_prediction
```

```
array([0, 1, 1, ..., 0, 1, 1], dtype=int64)
```

Next step we need to create the dataset of submission that contains tweet id and labels, so first we should convert our numerical data into primary classes : sports and politics and make a csv file based on this dataframe:

```
submission=pd.DataFrame({'TweetId' : test['TweetId'] , 'Label' : test_prediction })
```

```
submission
```

	TweetId	Label
0	306486520121012224	0
1	286353402605228032	1
2	289531046037438464	1
3	306451661403062273	1
4	297941800658812928	0
...	...	...
2605	282023761044189184	1
2606	303879735006601216	0
2607	297956846046703616	0
2608	304265049537658880	1
2609	306430391928115200	1

2610 rows × 2 columns

```
submission['Label'] = submission['Label'].map({0 : 'Sports' , 1 : 'Politics'})
```

```
submission
```

	TweetId	Label
0	306486520121012224	Sports
1	286353402605228032	Politics
2	289531046037438464	Politics
3	306451661403062273	Politics
4	297941800658812928	Sports
...	...	...
2605	282023761044189184	Politics
2606	303879735006601216	Sports
2607	297956846046703616	Sports
2608	304265049537658880	Politics
2609	306430391928115200	Politics

2610 rows × 2 columns

```
submission.to_csv('submission.csv')
```

And this is the dataset that we got:

```
,TweetId,Label
0,306486520121012224,Sports
1,286353402605228032,Politics
2,289531046037438464,Politics
3,306451661403062273,Politics
4,297941800658812928,Sports
5,305722428531802112,Politics
6,304713516256997377,Sports
7,234999630725783553,Politics
8,303712268372283392,Sports
9,304215754130194432,Sports
10,305498714527633408,Politics
11,302482560242565120,Politics
12,305496375985070080,Politics
13,305562747888865280,Politics
14,302124227975335937,Politics
15,279308630564679680,Politics
16,294826406272188416,Politics
17,304688274469969920,Sports
18,234244701153288192,Sports
19,274267981998075904,Politics
20,305715656295329793,Sports
21,306154104613396480,Sports
22,304684623068282880,Sports
```