Sarcasm_the_secret_language

May 11, 2023

With the NLP toolkit we will be attempting to detect sarcasm in text. I will be using pratice text data with /s at the end of the text indicating the text was sarcastic. Later I will created a prediction model from this to determine if future text are sarcastic

[4]:
```
#Import the needed librairies
%matplotlib inline

import bz2
import json
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

from collections import Counter
from lime.lime_text import LimeTextExplainer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, f1_score
from sklearn.pipeline import make_pipeline
from sklearn.dummy import DummyClassifier
from sklearn.ensemble import RandomForestClassifier
```

[5]:
```
# random seed is to ensure the reproducibility of my model
RANDOM_SEED = 655
```

[6]:
```
#loading pratice data
train_df = pd.read_csv('assets/sarcasm.train.tsv.gz', sep='\t',␣
 ↪compression='gzip').dropna()
test_imb_df = pd.read_csv('assets/sarcasm.test-imb.tsv.gz', sep='\t',␣
 ↪compression='gzip').dropna()
test_bal_df = pd.read_csv('assets/sarcasm.test-bal.tsv.gz', sep='\t',␣
 ↪compression='gzip').dropna()
```

[7]:
```
#TfidfVec is for featurizing the corpus, to speed this process up I will use a␣
 ↪min document freq of 100
def data_convert(df):
    X = df.text
    y = list(df.label)
    return X,y

#init function
X_train, y_train = data_convert(train_df)
X_test_bal, y_test_bal = data_convert(test_bal_df)
X_test_imb, y_test_imb = data_convert(test_imb_df)

#create vec
vectorizer = TfidfVectorizer(min_df = 100, stop_words = 'english',ngram_range␣
 ↪=(1,2) )

#X's
X_train = vectorizer.fit_transform(X_train)
X_test_bal = vectorizer.transform(X_test_bal)
X_test_imb = vectorizer.transform(X_test_imb)
```

[8]:
```
# Lets check the label count to ensure everything under the hood is working␣
 ↪according to plan
label_count = Counter(y_train)
label_count
```

[8]:
```
Counter({1: 128540, 0: 128541})
```

Now it's time to train my model I will use a LogisticRegression classifier and also a RandomForest-Classifier for comprasion. I will also use a dummy classifier why you ask? I want to be certain that my model is prediciting higher than random chance. Consider my dummy classifier a basline for performance since I am expecting to predicit higher than random chance.

[13]:
```
lr_clf = LogisticRegression(solver = 'lbfgs', multi_class = 'auto')
rf_clf = RandomForestClassifier(n_estimators = 50, max_depth = 15, random_state␣
 ↪= RANDOM_SEED)
random_clf = DummyClassifier(strategy='uniform',random_state = RANDOM_SEED).
 ↪fit(X_train, y_train)
# can't forget to fit our model--
lr_clf.fit(X_train, y_train)
rf_clf.fit(X_train, y_train)
```

[13]:
```
RandomForestClassifier(max_depth=15, n_estimators=50, random_state=655)
```

[16]:
```
y_pred_bal = lr_clf.predict(X_test_bal)
y_pred_imb = lr_clf.predict(X_test_imb)
rf_y_pred_bal = rf_clf.predict(X_test_bal)
rf_y_pred_imb = rf_clf.predict(X_test_imb)
random_y_pred_bal = random_clf.predict(X_test_bal)
random_y_pred_imb = random_clf.predict(X_test_imb)
```

```
[18]: #Now lets evaluate and see if sarcasm detection is possible
      # I will be using f1 scores since this problem is binary (sarcastic or not). I␣
       ↪will use binary for average
      def score(y,pred):
          f1 = f1_score(y, pred, average='binary')
          return f1

      lr_imb_f1 = score(y_test_imb,y_pred_imb)
      lr_bal_f1 = score(y_test_bal,y_pred_bal)

      rf_imb_f1 = score(y_test_imb,rf_y_pred_imb)
      rf_bal_f1 = score(y_test_bal,rf_y_pred_bal)

      rand_imb_f1 = score(y_test_imb,random_y_pred_imb)
      rand_bal_f1 = score(y_test_bal,random_y_pred_bal)
```

```
[25]: print('LogisticRegression_pred_scores',lr_bal_f1)
      print('RandomForrest_pred_scores',rf_bal_f1,)
      print('RandomForrest_pred_scores',rand_bal_f1)
```

```
LogisticRegression_pred_scores 0.6045505212772933
RandomForrest_pred_scores 0.4345472123751208
RandomForrest_pred_scores 0.502941629447097
```

60% accuracy not bad not bad at all, but lets see it at work shall we? Before we start let me give a warning I am pulling comments at random. The problem with this is the comment are random so please excuse the inapporiate language. Now that we got that out the way let begin!

```
[33]: class_names = ['sarcastic','not-sarcastic']

      lr_pipe = make_pipeline(vectorizer,lr_clf)
      rf_pipe = make_pipeline(vectorizer,rf_clf)

      explainer = LimeTextExplainer(class_names = class_names)

      test_row_df = test_imb_df.sample(1, random_state=RANDOM_SEED)
      _, inst_text, label = next(test_row_df.itertuples())

      print('Comment has text "%s"' %(inst_text))
      # print('True label: %d' % label)
      print('LogisticRegression Probability(Sarcastic) =', lr_pipe.
       ↪predict_proba([inst_text])[0, 1])
      print('RandomForest Probability(Sarcastic) =', rf_pipe.
       ↪predict_proba([inst_text])[0, 1])
```

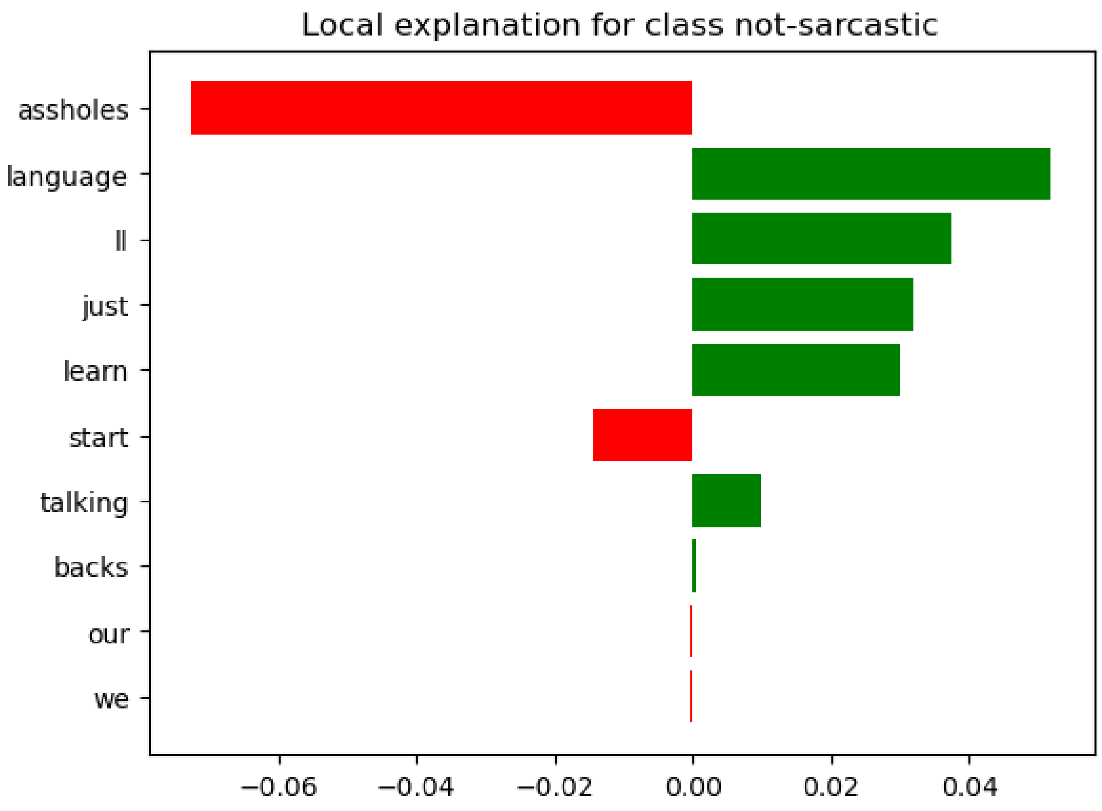
```
Comment has text "Now just get a supercomputer to learn the language and they'll
start talking with dolphins behind our backs about what assholes we are."
LogisticRegression Probability(Sarcastic) = 0.5316511260593914
```

```
RandomForest Probability(Sarcastic) = 0.4930586390484008
```

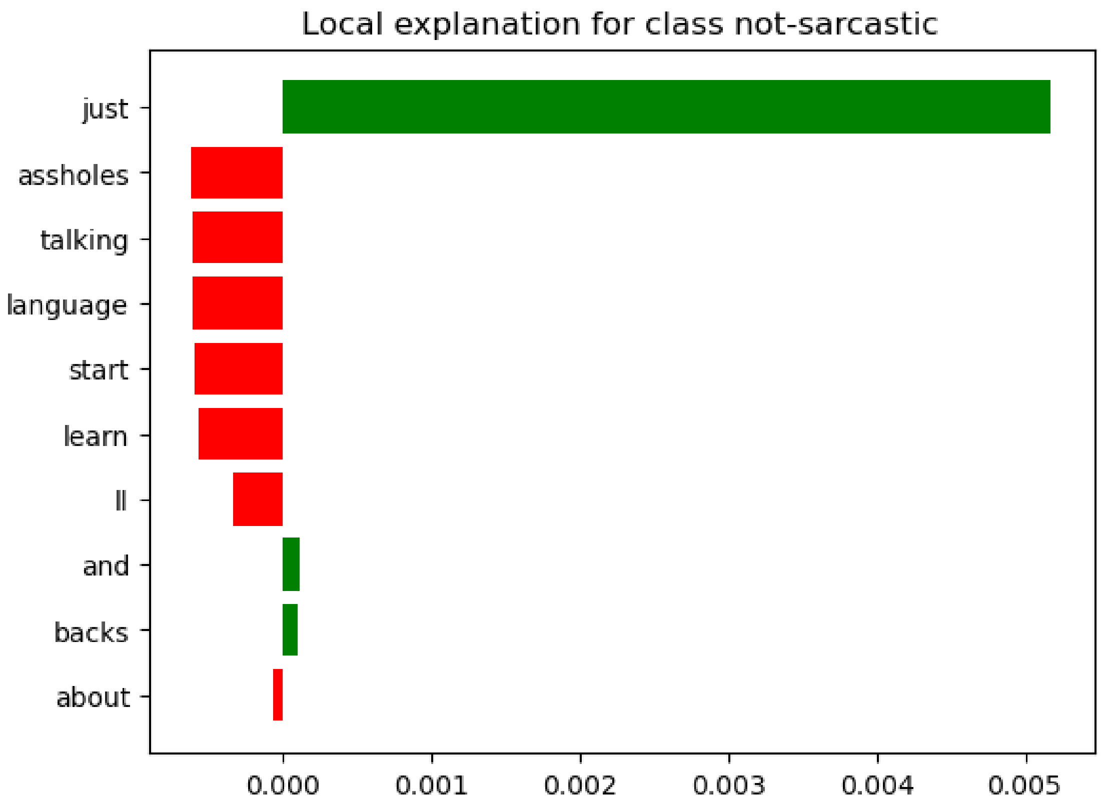
The funny part is I didn't detect any sarcasim at all, this all seem plausiable to me

[35]:
```
#I wonder what part of speech most likely lead to the high sarcasim score?

rf_explanation = explainer.explain_instance(inst_text, rf_pipe.predict_proba,␣
 ↪num_features=10)
lr_explanation = explainer.explain_instance(inst_text, lr_pipe.predict_proba,␣
 ↪num_features=10)
fig = lr_explanation.as_pyplot_figure()
```



[37]:
```
fig = rf_explanation.as_pyplot_figure()
```



I hope you enjoyed this exercise with me also… HIRE ME. I mean if you got this far you must see something you like right, and I'm not being sarcastic. Linkedin is in the bio… Have a great Day!!!

```
[ ]:
```