classification-using-tfidf-and-svm

April 17, 2023

0.0.1 Loading the Dataset

Importing pandas so that the csv file can be loaded into a dataframe named df

[29]:
```
import pandas as pd
```

[30]:
```
df = pd.read_csv('tweets_emotion.csv')
df.head()
```

[30]:
```
                        Date           User  \
0  2023-04-17 15:38:19+00:00   TheDizzle669
1  2023-04-17 15:38:15+00:00  MountainDogMa
2  2023-04-17 15:38:14+00:00      PopescuCo
3  2023-04-17 15:38:12+00:00      darwinesh
4  2023-04-17 15:38:10+00:00    Orhan583441

                                               Tweet emotion
0  @GuitarFamilyMan @brooklyn_jenny @Victorshi202…   anger
1  @DonaldJTrumpJr Stop lying. There are only adv…   anger
2  Noooo!!! You don't say!!! What a surprise for …   anger
3  @WarMonitors This war reminds me of the war be…     joy
4   Artillery of the 6th "Cossack Regiment" of th…   anger
```

Removing all rows where the ‘Tweet’ column has missing values using the dropna() function.

[31]:
```
df.dropna(subset=['Tweet'], inplace=True)
df.shape
```

[31]:
```
(15000, 4)
```

0.0.2 Maping the emotions to numeric labels

Creating a mapping dictionary called ‘mapping’ that maps each emotion to a numeric value.

[32]:
```
mapping = {
    'anger' : 0,
    'fear' : 1,
    'sadness' : 2,
    'surprise' : 3,
    'joy' : 4,
    'love' : 5,
}
```

Replacing the emotions in the ‘emotion’ column of the dataframe df with their corresponding numeric values using the replace() function and store the result in a new column called ‘emotion_numeric’.

[33]:
```
df['emotion_numeric'] = df['emotion'].replace(mapping)
```

[34]:
```
df.head()
```

[34]:
```
                        Date            User  \
0  2023-04-17 15:38:19+00:00   TheDizzle669
1  2023-04-17 15:38:15+00:00  MountainDogMa
2  2023-04-17 15:38:14+00:00      PopescuCo
3  2023-04-17 15:38:12+00:00      darwinesh
4  2023-04-17 15:38:10+00:00    Orhan583441

                                               Tweet emotion  emotion_numeric
0  @GuitarFamilyMan @brooklyn_jenny @Victorshi202…   anger                0
1  @DonaldJTrumpJr Stop lying. There are only adv…   anger                0
2  Noooo!!! You don't say!!! What a surprise for …   anger                0
3  @WarMonitors This war reminds me of the war be…     joy                4
4   Artillery of the 6th "Cossack Regiment" of th…   anger                0
```

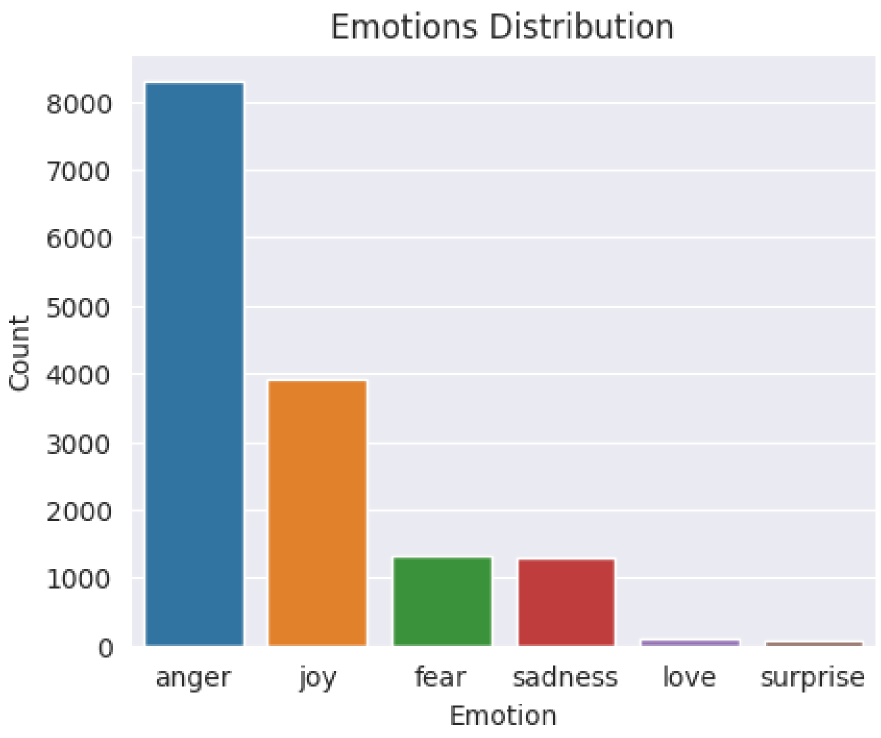
0.0.3 Visualising the Dataset

Generating a Bar graph to visualize the number of tweets of each emotion

[67]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

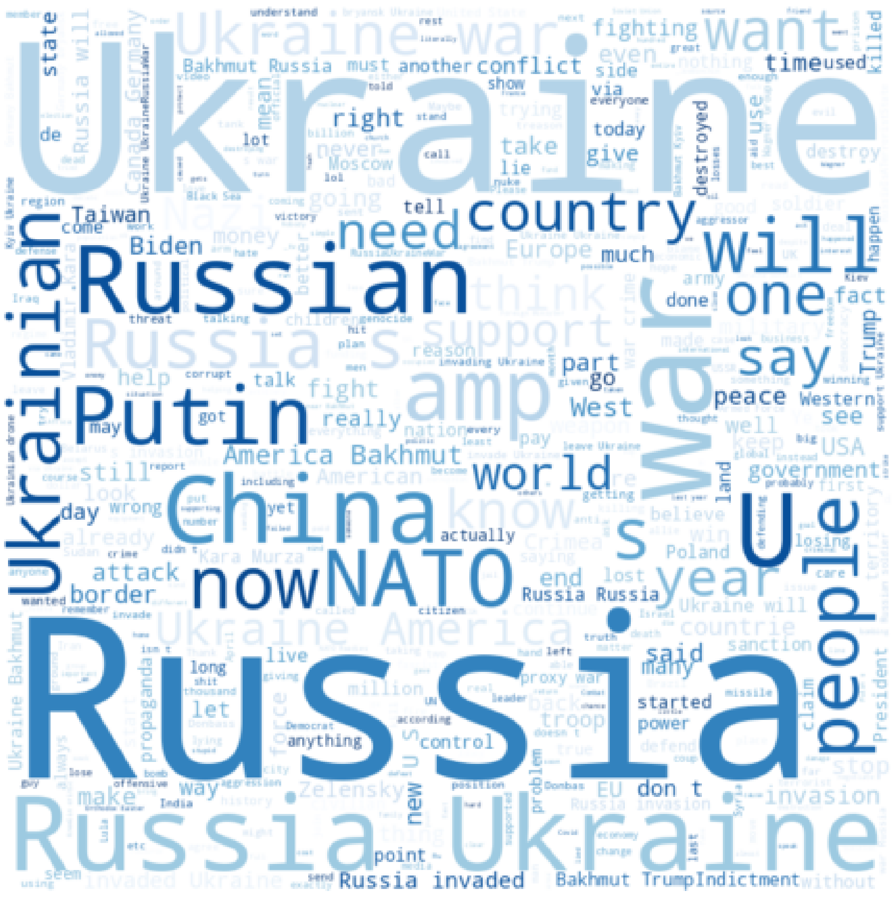
[70]:
```
emotions_count = df['emotion'].value_counts()
```

[73]:
```
sns.set_style('darkgrid')
plt.figure(figsize=(5, 4))
sns.barplot(x=emotions_count.index, y=emotions_count.values)
plt.title('Emotions Distribution')
plt.xlabel('Emotion')
plt.ylabel('Count')
plt.show()
```



Generating a word cloud to get an idea of most used words

[82]:
```
from wordcloud import WordCloud

wordcloud = WordCloud(width=800, height=800, background_color='white',␣
 ↪max_words=500, colormap='Blues').generate(' '.join(df['Tweet']))
plt.figure(figsize=(8, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



0.0.4 Cleaning the tweet data

Defining a function called clean_tweet() that takes a text string as input and removes Twitter-specific characters such as @mentions, RT, hashtags, and URLs using regular expressions

[35]:
```python
import re

def clean_tweet(text):
    if not isinstance(text, str):
        return ''
    text = re.sub(r'@[A-Za-z0-9]+', '', text)
    text = re.sub(r'(RT[\s]+|:[\s]+)', '', text)
    text = re.sub(r'#', '', text)
    text = re.sub(r'https?:\/\/\S+', '', text)
    return text
```

Applying this function to the 'Tweet' column of the dataframe df using the apply() function

[36]:
```python
df['Tweet'] = df['Tweet'].apply(lambda twt: clean_tweet(twt))
```

[37]:
```
df.head()
```

[37]:
```
                        Date            User  \
0  2023-04-17 15:38:19+00:00   TheDizzle669
1  2023-04-17 15:38:15+00:00  MountainDogMa
2  2023-04-17 15:38:14+00:00      PopescuCo
3  2023-04-17 15:38:12+00:00      darwinesh
4  2023-04-17 15:38:10+00:00    Orhan583441

                                               Tweet emotion  emotion_numeric
0   _jenny  Its literally one of the smallest rif…   anger                0
1   Stop lying. There are only advisors on the gr…   anger                0
2  Noooo!!! You don't say!!! What a surprise for …   anger                0
3   This war reminds me of the war between China …     joy                4
4   Artillery of the 6th "Cossack Regiment" of th…   anger                0
```

0.0.5 Removing emojis from the tweets

Defining a function called deEmojify() that takes a text string as input and removes any emojis using regular expressions.

[38]:
```
def deEmojify(text):
    regrex_pattern = re.compile(pattern = "["
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           "]+", flags = re.UNICODE)
    return regrex_pattern.sub(r'',text)
```

Applying this function to the 'Tweet' column of the dataframe df using the apply() function.

[39]:
```
df['Tweet'] = df['Tweet'].apply(lambda twt: deEmojify(twt))
```

[40]:
```
df.head()
```

[40]:
```
                        Date            User  \
0  2023-04-17 15:38:19+00:00   TheDizzle669
1  2023-04-17 15:38:15+00:00  MountainDogMa
2  2023-04-17 15:38:14+00:00      PopescuCo
3  2023-04-17 15:38:12+00:00      darwinesh
4  2023-04-17 15:38:10+00:00    Orhan583441

                                               Tweet emotion  emotion_numeric
0   _jenny  Its literally one of the smallest rif…   anger                0
1   Stop lying. There are only advisors on the gr…   anger                0
2  Noooo!!! You don't say!!! What a surprise for …   anger                0
3   This war reminds me of the war between China …     joy                4
```

```
4  Artillery of the 6th "Cossack Regiment" of the…   anger                0
```

0.0.6 Importing and Downloading necessary NLTK resources

[41]:
```python
import nltk
from nltk.tokenize import word_tokenize
from nltk.stem.snowball import SnowballStemmer
from nltk.corpus import stopwords
from textblob import TextBlob

nltk.download('punkt')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

[41]:
```
True
```

Generating english stop words and storing them into the list names enlish_stopwords. Ignoring stopwords like not, cannot, etc as they would help in emotion classification

[42]:
```python
stemmer = SnowballStemmer(language = 'english')
english_stopwords = stopwords.words('english')
english_stopwords = english_stopwords[:116]
','.join(english_stopwords)
```

[42]:
```
"i,me,my,myself,we,our,ours,ourselves,you,you're,you've,you'll,you'd,your,yours,
yourself,yourselves,he,him,his,himself,she,she's,her,hers,herself,it,it's,its,it
self,they,them,their,theirs,themselves,what,which,who,whom,this,that,that'll,the
se,those,am,is,are,was,were,be,been,being,have,has,had,having,do,does,did,doing,
a,an,the,and,but,if,or,because,as,until,while,of,at,by,for,with,about,against,be
tween,into,through,during,before,after,above,below,to,from,up,down,in,out,on,off
,over,under,again,further,then,once,here,there,when,where,why,how,all,any,both,e
ach,few,more,most,other,some,such"
```

0.0.7 Function to convert chat acronyms to full form

Generating a chat word dictionary with full form of common chat acronyms. Then creating a function chat_conversation() which would replace the acronyms in a phrase if found.

[43]:
```python
chat_words = {
    "lol": "laugh out loud",
    "brb": "be right back",
    "omg": "oh my god",
    "jk": "just kidding",
    "fyi": "for your information",
    "btw": "by the way",
    "afaik": "as far as I know",
    "idk": "I don't know",
    "imo": "in my opinion",
    "tbh": "to be honest",
    "ty": "thank you",
    "yw": "you're welcome",
    "np": "no problem",
    "gg": "good game",
    "wp": "well played",
    "ggwp": "good game, well played",
    "irl": "in real life",
    "imo": "in my opinion",
    "smh": "shaking my head",
    "tfw": "that feeling when",
    "thx": "thanks",
    "wtf": "what the f***",
    "omw": "on my way",
    "jk": "just kidding",
    "rn": "right now",
    "afk": "away from keyboard",
    "b4": "before",
    "cu": "see you",
    "dbmib": "don't bother me I'm busy",
    "dl": "download",
    "dw": "don't worry",
    "ez": "easy",
    "ffs": "for f***'s sake",
    "fu": "f*** you",
    "hbu": "how about you",
    "hru": "how are you",
    "ic": "I see",
    "idc": "I don't care",
    "ikr": "I know, right",
    "ily": "I love you",
    "imho": "in my humble opinion",
    "lmao": "laughing my ass off",
    "lmk": "let me know",
    "nbd": "no big deal",
    "nvm": "nevermind",
    "ofc": "of course",
    "ppl": "people",
    "rofl": "rolling on the floor laughing",
    "srsly": "seriously",
    "stfu": "shut the f*** up",
    "tmi": "too much information",
    "ttyl": "talk to you later",
    "u": "you",
    "ur": "your",
    "wbu": "what about you",
    "wth": "what the hell",
    "yolo": "you only live once",
    "yw": "you're welcome",
    "amp": "and"
}

def chat_conversation(text):
  new_text = []
  for word in text.split():
    if word.lower() in chat_words:
      new_text.append(chat_words[word.lower()])
    else:
      new_text.append(word)
  return " ".join(new_text)
```

0.0.8 Tokenization function

Defining a function called tokenize() that takes a text string as input, converts it to lowercase, removes non-alphabetic characters, removes stop words, corrects incorrect spellings and applies stemming using the SnowballStemmer algorithm from the NLTK package.

[55]:
```
def tokenize(txt):
    txt = chat_conversation(txt.lower())
    textBlb = TextBlob(txt)
    text = textBlb.correct().string
    return [stemmer.stem(token) for token in word_tokenize(text) if token.
 ↪isalpha() and token not in english_stopwords]
```

[56]:
```
# Testing the tokenize function
tokenize('What a wonderful lifee !!')
```

[56]:
```
['wonder', 'life']
```

0.0.9 Initializing the TFIDF Vector

Creating the Tfidf Vector using TfidfVectorizer function from sklearn. The tokenize() function we created is passed in the tokenizer, ngram range is set to a maximum value of 2 and the maximum of 2000 features can be generated

[57]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

[58]:
```
vectorizer = TfidfVectorizer(
    tokenizer = tokenize,
    ngram_range = (1,2),
    max_features = 1500,
)
```

0.0.10 Spliting the dataset into Test and Train

Splitting the database into 2 parts: 80% data will be Training data (df1) and the rest would be Test Data (df2) with a randomness factor of 500.

[59]:
```
from sklearn.model_selection import train_test_split

df1, df2 = train_test_split(df, test_size=0.2, random_state=40)
```

Transforming the Tweet column of both the dataframes seperately

[61]:
```
train_inputs = vectorizer.fit_transform(df1.Tweet)
```

[62]:
```
val_inputs = vectorizer.transform(df2.Tweet)
```

Creating the resultant value lists of train and test data

[63]:
```
train_targets = df1.emotion_numeric
val_targets = df2.emotion_numeric
```

0.0.11 Applying Support Vector Machine Algorithm

Using SVM(SVC) from sklearn with rbf kernel, C=10 and gamma=0.8

[64]:
```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix

clf = SVC(kernel='rbf', C=10, gamma=0.8)

clf.fit(train_inputs, train_targets)

train_pred = clf.predict(train_inputs)
accuracy_train = accuracy_score(train_targets, train_pred)

val_pred = clf.predict(val_inputs)
accuracy_val = accuracy_score(val_targets, val_pred)

print('Accuracy on Train Set:', accuracy_train)
print('Accuracy on Test Set:', accuracy_val)
```

```
Accuracy on Train Set: 0.9928333333333333
Accuracy on Test Set: 0.681
```

Generating report on Precision, Recall and F1 score

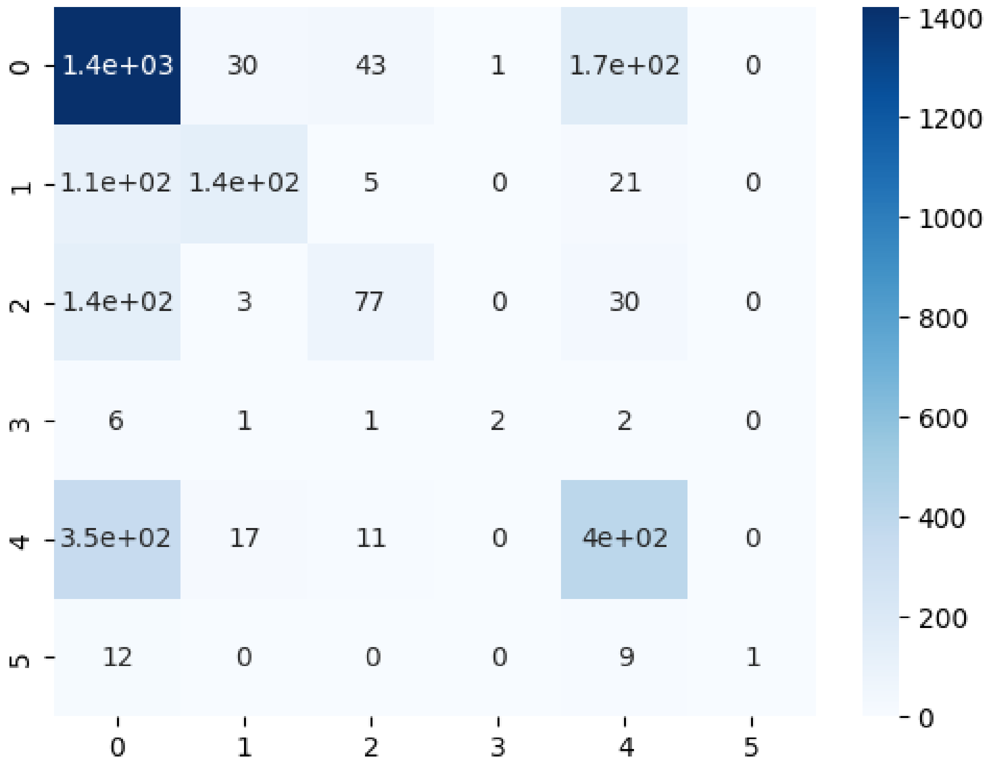
[65]:
```
report = classification_report(val_targets, val_pred, zero_division=1)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.70      0.85      0.77      1665
           1       0.73      0.51      0.60       268
           2       0.56      0.31      0.40       246
           3       0.67      0.17      0.27        12
           4       0.64      0.51      0.57       787
           5       1.00      0.05      0.09        22

    accuracy                           0.68      3000
   macro avg       0.72      0.40      0.45      3000
weighted avg       0.68      0.68      0.66      3000
```

Generating Confusion Matrix

[66]:
```
cm = confusion_matrix(val_targets, val_pred)
sns.heatmap(cm, annot=True, cmap='Blues')
```

[66]:
```
<Axes: >
```



1.4e+03
30
43
1
1.7e+02
0
1.1e+02
1.4e+02
5
0
21
0
1.4e+02
3
77
0
30
0
6
1
1
2
2
0
3.5e+02
17
11
0
4e+02
0
12
0
0
0
9
1
0
1
2
3
4
5
1400
1200
1000
800
600
400
200
0