testing

December 28, 2023

```
0609
0523
0587
0376
0378
```

```


selection import train_test_split
yplot as plt
ns
selection import train_test_split
cessing import StandardScaler, LabelEncoder
_model import LinearRegression
s import r2_score
mport DecisionTreeClassifier
s import accuracy_score
 tree

port Counter
kends.backend_pdf import PdfPages
port stopwords

import word_tokenize




rt WordNetLemmatizer
e_extraction.text import CountVectorizer, TfidfVectorizer
selection import train_test_split, GridSearchCV
port LinearSVC
s import accuracy_score, classification_report
_network import MLPClassifier
rs import Adam


import word_tokenize
```

```
from google.colab import drive
```

```
import tensorflow.keras as keras
from tensorflow.keras.models
import Sequential from
tensorflow.keras.layers import
Dense

import warnings
warnings.filterwarnings("ignore")
```

```
[222]: count_vectorizer = joblib.load('count_vectorizer.pkl')
```

```
[223]: linear_svc_model = joblib.load('linear_svc_model.pkl')
```

```
[224]: test_data = ["i am traveling, and i am so happy","happy
birhtday!", "I am␣
 ↪extremely happy", "I am really sad now","I really enjoyed the
 movie", "The␣
 ↪food was terrible", "The weather is perfect today", "This
 place was␣
 ↪amazing", "I wouldn't recommend this", "I regret buying this
 product", "The␣
 ↪internet speed is slow", "The hotel room was spacious and
 clean", "The beach␣
 ↪was crowded", "The car broke down on the highway", "I haven't
 answered well,␣
 ↪it was really difficult.", "The scenery here is beautiful",
 "The service at␣
 ↪this restaurant was excellent", "The hike was refreshing"]
print(test_data)
```

```
['i am traveling, and i am so happy', 'happy birhtday!', 'I am
extremely happy',
'I am really sad now', 'I really enjoyed the movie', 'The food
was terrible', 'The weather is perfect today', 'This place was
amazing', "I wouldn't recommend this", 'I regret buying this
product', 'The internet speed is slow', 'The hotel room was
spacious and clean', 'The beach was crowded', 'The car broke
down on the highway', "I haven't answered well, it was really
difficult.", 'The scenery here is beautiful', 'The service at
this restaurant was excellent', 'The hike was refreshing']
```

[225]:
```python
def removePunctuation(sentence):
    sentenceWithoutPunc = ""
    sentenceWithoutPunc = "".join(i for i in sentence if i not in string.
 ↪punctuation)
    return sentenceWithoutPunc

# removePunctuation from new test data
test_data = [removePunctuation(sentence) for sentence in test_data]
print(test_data)
```

```
['i am traveling and i am so happy', 'happy birhtday', 'I am
extremely happy',
'I am really sad now', 'I really enjoyed the movie', 'The food
was terrible',
'The weather is perfect today', 'This place was amazing', 'I
wouldnt recommend
this', 'I regret buying this product', 'The internet speed is slow',
'The hotel room was spacious and clean', 'The beach was crowded',
'The car broke down on the highway', 'I havent answered well it was
really difficult', 'The scenery here is beautiful', 'The service at
this restaurant was excellent', 'The hike was refreshing']
```

[226]:
```python
# lowercase words
test_data = [s.lower() for s in test_data]
print(test_data)
```

```
['i am traveling and i am so happy', 'happy birhtday', 'i am
extremely happy',
'i am really sad now', 'i really enjoyed the movie', 'the food was
terrible', 'the weather is perfect today', 'this place was amazing',
'i wouldnt recommend this', 'i regret buying this product', 'the
internet speed is slow', 'the hotel room was spacious and clean',
'the beach was crowded', 'the car broke down on the highway', 'i
havent answered well it was really difficult', 'the scenery here is
beautiful', 'the service at this restaurant was excellent', 'the hike
was refreshing']
```

[227]:
```python
# Tokenization
def Tokenization(sentence):
    tokens = re.split(r'\W+', sentence)
    return tokens
test_data = [Tokenization(sentence) for sentence in test_data]
print(test_data)
```

```
[['i', 'am', 'traveling', 'and', 'i', 'am', 'so', 'happy'], ['happy',
'birhtday'], ['i', 'am', 'extremely', 'happy'], ['i', 'am', 'really',
'sad',
'now'], ['i', 'really', 'enjoyed', 'the', 'movie'], ['the', 'food',
'was',
'terrible'], ['the', 'weather', 'is', 'perfect', 'today'], ['this',
'place',
'was', 'amazing'], ['i', 'wouldnt', 'recommend', 'this'], ['i',
'regret',
'buying', 'this', 'product'], ['the', 'internet', 'speed', 'is',
'slow'],
['the', 'hotel', 'room', 'was', 'spacious', 'and', 'clean'], ['the',
'beach',
'was', 'crowded'], ['the', 'car', 'broke', 'down', 'on', 'the',
'highway'],
['i', 'havent', 'answered', 'well', 'it', 'was', 'really',
'difficult'], ['the',
'scenery', 'here', 'is', 'beautiful'], ['the', 'service', 'at',
'this',
'restaurant', 'was', 'excellent'], ['the', 'hike', 'was',
'refreshing']]
```

[228]:
```
# Remove stop words

# spacy.cli.download("en_core_web_sm")
nlp = spacy.load('en_core_web_sm')
default_stop_words = set(nlp.Defaults.stop_words)
      negationWords = set(["hadn't", "wouldn't", "doesn't",
      "mightn't", "won't",␣
       ↪"shouldn't", 'haven', 'aren' , 'doesn', 'couldn', 'didn',
       "didnt",'isn',␣
       ↪'wouldn', 'mustn', "isn't", "shan't", "didn't", 'shan',
       'hadn', 'wasn',␣
       ↪'weren', "hasn't", 'mightn', "couldn't", "needn't",
       "haven't", "weren't",␣
       ↪"aren't", 'needn', 'not', 'shouldn', 'hasn', "mustn't",
       "wasn't", "don't",␣ ↪'don'])

      custom_stop_words = default_stop_words - negationWords
      nlp.Defaults.stop_words = custom_stop_words
      #X_filter =
      pd.DataFrame(data) def
      stopWordsRemoval(sentenceToke
      nized):

          allInfo = nlp(' '.join(sentenceTokenized))
```

```
   filtered_tokens = [token.text for token in allInfo if

 token.text.lower()␣ ↪not in custom_stop_words] return

 filtered_tokens
```

[229]:
```
# stopword removal test_data =
[stopWordsRemoval(sentence) for sentence in
test_data] print(test_data)
```

```
[['traveling', 'happy'], ['happy', 'birhtday'], ['extremely',
'happy'], ['sad'],
['enjoyed', 'movie'], ['food', 'terrible'], ['weather',
'perfect', 'today'],
['place', 'amazing'], ['nt', 'recommend'], ['regret', 'buying',
'product'],
['internet', 'speed', 'slow'], ['hotel', 'room', 'spacious',
'clean'], ['beach',
'crowded'], ['car', 'broke', 'highway'], ['nt', 'answered',
'difficult'],
['scenery', 'beautiful'], ['service', 'restaurant',
'excellent'], ['hike', 'refreshing']]
```

[230]:
```
# lemmatize
def lemmatize(tokens):
    doc = nlp(' '.join(tokens))
    lemmatized_tokens = [token.lemma_ for token in doc]
    return lemmatized_tokens
test_data = [lemmatize(sentence) for sentence in test_data]

print(test_data)
```

```
[['travel', 'happy'], ['happy', 'birhtday'], ['extremely',
'happy'], ['sad'],
['enjoy', 'movie'], ['food', 'terrible'], ['weather',
'perfect', 'today'],
['place', 'amazing'], ['not', 'recommend'], ['regret',
'buying', 'product'],
['internet', 'speed', 'slow'], ['hotel', 'room', 'spacious',
'clean'], ['beach',
'crowd'], ['car', 'break', 'highway'], ['not', 'answer',
'difficult'],
['scenery', 'beautiful'], ['service', 'restaurant',
'excellent'], ['hike', 'refreshing']]
```

```
[231]: print(test_data)

       test_data = [' '.join(sentence) for sentence in test_data]

       # Fit and transform the data
       Xx = count_vectorizer.transform(test_data)



       # Convert the result to a DataFrame (optional)
       test_data_df = pd.DataFrame(Xx.toarray(), columns=count_vectorizer.
        ↪get_feature_names_out())

       # Displaying the embeddings
       print(test_data_df)
```

```
[['travel', 'happy'], ['happy', 'birhtday'], ['extremely', 'happy'],
['sad'],
['enjoy', 'movie'], ['food', 'terrible'], ['weather', 'perfect',
'today'],
['place', 'amazing'], ['not', 'recommend'], ['regret', 'buying',
'product'],
['internet', 'speed', 'slow'], ['hotel', 'room', 'spacious',
'clean'], ['beach',
'crowd'], ['car', 'break', 'highway'], ['not', 'answer',
'difficult'],
['scenery', 'beautiful'], ['service', 'restaurant', 'excellent'],
['hike',
'refreshing']]
    010  10  100  1010  11  110  1199  12  13  15  …  yukon  yum  yummy  \
0     0    0    0     0    0    0     0    0   0   0  …      0    0
      0
1     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
3     0    0    0     0    0    0     0    0   0   0  …      0    0
      0
4     0    0    0     0    0    0     0    0   0   0  …      0    0
      0
5     0    0    0     0    0    0     0    0   0   0  …      0    0
      0
6     0    0    0     0    0    0     0    0   0   0  …      0    0
      0
```

```
7     0     0     0     0     0     0     0     0     0     0 …   0     0
      0
8     0     0     0     0     0     0     0     0     0     0 …   0     0
      0
9     0     0     0     0     0     0     0     0     0     0 …   0     0
      0
10    0     0     0     0     0     0     0     0     0     0 …   0     0
      0
11    0     0     0     0     0     0     0     0     0     0 …   0     0
      0
12    0     0     0     0     0     0     0     0     0     0 …   0     0
      0
13    0     0     0     0     0     0     0     0     0     0 …   0     0
      0
14    0     0     0     0     0     0     0     0     0     0 …   0     0
      0
15    0     0     0     0     0     0     0     0     0     0 …   0     0
      0
16    0     0     0     0     0     0     0     0     0     0 …   0     0
      0
17    0     0     0     0     0     0     0     0     0     0 …   0     0
      0

    yun z500a zero zillion zombie zombiestudent zombiez
      0     0     0     0     0     0     0     0
      1     0     0     0     0     0     0     0
      2     0     0     0     0     0     0     0
      3     0     0     0     0     0     0     0
      4     0     0     0     0     0     0     0
      5     0     0     0     0     0     0     0
      6     0     0     0     0     0     0     0
      7     0     0     0     0     0     0     0
      8     0     0     0     0     0     0     0
      9     0     0     0     0     0     0     0
      10    0     0     0     0     0     0     0
      11    0     0     0     0     0     0     0
      12    0     0     0     0     0     0     0
      13    0     0     0     0     0     0     0
      14    0     0     0     0     0     0     0
      15    0     0     0     0     0     0     0
      16    0     0     0     0     0     0     0
      17    0     0     0     0     0     0     0

      [18 rows x 4328 columns]
```

[232]:
```
y_predict = linear_svc_model.predict(test_data_df)
print(y_predict)
```

```
[1 1 1 0 1 0 1 1 0 0 0 1 0 0 0 1 1 0]
```

[232]:
```

```