q2-a2-l215625

March 11, 2024

[3]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import Perceptron
```

[13]:
```
data = pd.read_csv("Q2 Sentiment Analysis Dataset.csv", encoding='latin1')
# dropping columns
data=data.drop(['id','date','Unnamed: 4','Unnamed: 5'],axis=1)

def preprocess_text(text):

    text = text.lower()
    # Remove punctuation
    text = re.sub(r'[^\w\s]', '', text)
    return text

data['text'] = data['text'].apply(preprocess_text)
```

[15]:
```
data
```

[15]:

	sentiment	text
0	0	wtf my battery was 31 one second ago and now i…
1	0	apple contact sync between yosemite and ios8 i…
2	0	warning if you buy an iphone 5s unlocked from …

```
3             0  apple for the love of gawd center the 1on the …
4             0  i get the storage almost full notification lit…
…           …                                                 …
3881          3  aaplaccel partners leads 50m series c funding …
3882          3  counting down the minutes interest in full tim…
3883          3           justinpulitzer any comment on aapl today
3884          3  have been brave and taken out an aapl cfd as t…
3885          3  tim cook met with jesse jackson for positive a…

[3886 rows x 2 columns]
```

[14]:
```
# labels encoding
label_encoder = LabelEncoder()
data['sentiment'] = label_encoder.fit_transform(data['sentiment'])
```

[6]:
```
# Split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(data['text'],␣
 ↪data['sentiment'], test_size=0.2, random_state=42)
```

[7]:
```
# Feature extraction methods
vectorizers = {
    "Bag of Words (Raw Counts)": CountVectorizer(),
    "Bag of Words (TfIDF)": TfidfVectorizer(),
    "Ngrams (Unigrams, Bigrams, Trigrams)": TfidfVectorizer(ngram_range=(1, 3))
}
```

[10]:
```
vectorizer_bow_raw = CountVectorizer()
X_bow_raw = vectorizer_bow_raw.fit_transform(data['text'])
```

[11]:
```
vectorizer_bow_tfidf = TfidfVectorizer()
X_bow_tfidf = vectorizer_bow_tfidf.fit_transform(data['text'])
```

N-grams

[12]:
```
vectorizer_uni = CountVectorizer(ngram_range=(1, 1))  # Unigrams
vectorizer_bi = CountVectorizer(ngram_range=(1, 2))  # Bigrams
vectorizer_tri = CountVectorizer(ngram_range=(1, 3))  # Trigrams

X_uni = vectorizer_uni.fit_transform(data['text'])
X_bi = vectorizer_bi.fit_transform(data['text'])
X_tri = vectorizer_tri.fit_transform(data['text'])

print("UNIGRAM")
print(" ")
print(X_uni)
print("_____␣
 ↪")
```

```
print("BIGRAM")
print(" ")
print(X_bi)
print("_____________________________________________________________________________␣
 ↪")
print("TRIGRAM")
print(" ")
print(X_tri)
print("_____________________________________________________________________________␣
 ↪")
```

```
UNIGRAM

  (0, 8579)     2
  (0, 5920)     1
  (0, 909)      1
  (0, 8364)     1
  (0, 163)      1
  (0, 6137)     1
  (0, 7103)     1
  (0, 476)      1
  (0, 584)      1
  (0, 6071)     1
  (0, 5069)     2
  (0, 154)      1
  (0, 7896)     1
  (0, 651)      1
  (1, 584)      1
  (1, 5069)     1
  (1, 651)      1
  (1, 1606)     1
  (1, 7697)     1
  (1, 990)      1
  (1, 8645)     1
  (1, 5016)     1
  (1, 7161)     1
  (1, 7086)     1
  (1, 8201)     1
  :     :
  (3884, 302)   1
  (3884, 763)   1
  (3884, 6201)  1
  (3884, 942)   1
  (3884, 3044)  1
  (3884, 2135)  1
  (3884, 7127)  1
  (3884, 7721)  1
```

```
  (3884, 1339)  1
  (3884, 29)    1
  (3884, 1122)  1
  (3884, 6221)  1
  (3885, 584)   1
  (3885, 2647)  1
  (3885, 8509)  1
  (3885, 302)   1
  (3885, 7928)  1
  (3885, 1622)  1
  (3885, 6473)  1
  (3885, 2017)  1
  (3885, 5145)  1
  (3885, 5728)  1
  (3885, 5118)  1
  (3885, 6568)  1
  (3885, 4147)  1
--------------------------------------------------------------------------------
BIGRAM

  (0, 33901)    2
  (0, 21183)    1
  (0, 5041)     1
  (0, 32561)    1
  (0, 486)      1
  (0, 22793)    1
  (0, 26429)    1
  (0, 1908)     1
  (0, 2514)     1
  (0, 22131)    1
  (0, 17506)    2
  (0, 465)      1
  (0, 30225)    1
  (0, 3075)     1
  (0, 33906)    1
  (0, 21214)    1
  (0, 5063)     1
  (0, 32562)    1
  (0, 488)      1
  (0, 22834)    1
  (0, 26430)    1
  (0, 1909)     1
  (0, 2661)     1
  (0, 22162)    1
  (0, 17515)    1
  :     :
  (3885, 2514)  1
  (3885, 11256) 1
```

```
  (3885, 33429) 1
  (3885, 835)   1
  (3885, 30453) 1
  (3885, 7809)  1
  (3885, 30454) 1
  (3885, 24210) 1
  (3885, 9141)  1
  (3885, 18260) 1
  (3885, 20489) 1
  (3885, 18195) 1
  (3885, 24523) 1
  (3885, 15106) 1
  (3885, 7821)  1
  (3885, 20490) 1
  (3885, 33482) 1
  (3885, 18261) 1
  (3885, 18196) 1
  (3885, 11427) 1
  (3885, 24211) 1
  (3885, 2681)  1
  (3885, 24524) 1
  (3885, 9143)  1
  (3885, 15107) 1
--------------------------------------------------------------------------------
TRIGRAM

  (0, 64091)    2
  (0, 39015)    1
  (0, 10329)    1
  (0, 61326)    1
  (0, 802)      1
  (0, 42628)    1
  (0, 49186)    1
  (0, 3307)     1
  (0, 4471)     1
  (0, 41087)    1
  (0, 31963)    2
  (0, 769)      1
  (0, 56573)    1
  (0, 5646)     1
  (0, 64108)    1
  (0, 39088)    1
  (0, 10376)    1
  (0, 61327)    1
  (0, 805)      1
  (0, 42721)    1
  (0, 49187)    1
  (0, 3308)     1
```

```
  (0, 4823)     1
  (0, 41144)    1
  (0, 31980)    1
  :     :
  (3885, 33583) 1
  (3885, 45841) 1
  (3885, 27801) 1
  (3885, 15330) 1
  (3885, 37789) 1
  (3885, 63242) 1
  (3885, 33701) 1
  (3885, 33584) 1
  (3885, 21741) 1
  (3885, 45292) 1
  (3885, 4871)  1
  (3885, 45842) 1
  (3885, 17589) 1
  (3885, 27802) 1
  (3885, 57072) 1
  (3885, 15331) 1
  (3885, 37790) 1
  (3885, 63243) 1
  (3885, 33702) 1
  (3885, 33585) 1
  (3885, 21742) 1
  (3885, 45293) 1
  (3885, 4872)  1
  (3885, 45843) 1
  (3885, 17590) 1
```

Checking test train split

[16]:
```python
print(f"Number of training samples: {X_train.shape[0]}")
print(f"Number of test samples: {X_test.shape[0]}")
```

```
Number of training samples: 3108
Number of test samples: 778
```

Classifiers

[8]:
```python
classifiers = {
    "Naïve Bayes": MultinomialNB(),
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "Random Forest": RandomForestClassifier(),
    "SVM": SVC(),
    "Perceptron": Perceptron()
}
```

Accuracy

[17]:
```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7352185089974294
```

Fitting model and displaying results

[9]:
```
# Results storage
results = []

# Train and evaluate classifiers
for vectorizer_name, vectorizer in vectorizers.items():
    for classifier_name, classifier in classifiers.items():
        # Feature extraction
        X_train_features = vectorizer.fit_transform(X_train)
        X_test_features = vectorizer.transform(X_test)

        # Train classifier
        classifier.fit(X_train_features, y_train)

        # Predictions
        y_pred = classifier.predict(X_test_features)


        accuracy = accuracy_score(y_test, y_pred)
        precision_macro = precision_score(y_test, y_pred, average='macro')
        recall_macro = recall_score(y_test, y_pred, average='macro')
        f1_macro = f1_score(y_test, y_pred, average='macro')

        precision_micro = precision_score(y_test, y_pred, average='micro')
        recall_micro = recall_score(y_test, y_pred, average='micro')
        f1_micro = f1_score(y_test, y_pred, average='micro')

        # results
        results.append({
            "Vectorizer": vectorizer_name,
            "Classifier": classifier_name,
            "Accuracy": accuracy,
            "Precision (Macro)": precision_macro,
            "Recall (Macro)": recall_macro,
            "F1-score (Macro)": f1_macro,
            "Precision (Micro)": precision_micro,
            "Recall (Micro)": recall_micro,
            "F1-score (Micro)": f1_micro
        })
```

```python
results_df = pd.DataFrame(results)
print(results_df)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
```

```
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

	Vectorizer	Classifier	Accuracy	\
0	Bag of Words (Raw Counts)	Naïve Bayes	0.715938	
1	Bag of Words (Raw Counts)	Logistic Regression	0.754499	
2	Bag of Words (Raw Counts)	Random Forest	0.737789	
3	Bag of Words (Raw Counts)	SVM	0.742931	
4	Bag of Words (Raw Counts)	Perceptron	0.733933	
5	Bag of Words (TfIDF)	Naïve Bayes	0.746787	
6	Bag of Words (TfIDF)	Logistic Regression	0.751928	
7	Bag of Words (TfIDF)	Random Forest	0.749357	
8	Bag of Words (TfIDF)	SVM	0.759640	
9	Bag of Words (TfIDF)	Perceptron	0.674807	
10	Ngrams (Unigrams, Bigrams, Trigrams)	Naïve Bayes	0.704370	
11	Ngrams (Unigrams, Bigrams, Trigrams)	Logistic Regression	0.746787	
12	Ngrams (Unigrams, Bigrams, Trigrams)	Random Forest	0.727506	
13	Ngrams (Unigrams, Bigrams, Trigrams)	SVM	0.740360	
14	Ngrams (Unigrams, Bigrams, Trigrams)	Perceptron	0.735219	

	Precision (Macro)	Recall (Macro)	F1-score (Macro)	Precision (Micro)	\
0	0.496166	0.428204	0.424428	0.715938	
1	0.517420	0.457265	0.469227	0.754499	
2	0.517782	0.431115	0.440345	0.737789	
3	0.600598	0.425956	0.435329	0.742931	
4	0.554083	0.473937	0.496917	0.733933	
5	0.626929	0.420338	0.419004	0.746787	
6	0.526051	0.441169	0.449938	0.751928	
7	0.558084	0.440001	0.449280	0.749357	
8	0.610379	0.441470	0.451787	0.759640	
9	0.452619	0.456378	0.451830	0.674807	
10	0.634571	0.380379	0.384278	0.704370	
11	0.572034	0.432488	0.435362	0.746787	
12	0.562043	0.424991	0.436769	0.727506	
13	0.566794	0.422010	0.425109	0.740360	
14	0.511263	0.466999	0.480488	0.735219	

	Recall (Micro)	F1-score (Micro)

0	0.715938	0.715938
1	0.754499	0.754499
2	0.737789	0.737789
3	0.742931	0.742931
4	0.733933	0.733933
5	0.746787	0.746787
6	0.751928	0.751928
7	0.749357	0.749357
8	0.759640	0.759640
9	0.674807	0.674807
10	0.704370	0.704370
11	0.746787	0.746787
12	0.727506	0.727506
13	0.740360	0.740360
14	0.735219	0.735219