Homework 7 - Grant Jackson

November 6, 2024

[1]:
```
import os
import numpy as np
import pandas as pd
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, make_scorer,␣
 ↪precision_score, recall_score
```

[2]:
```
# Load the 20newsgroups dataset
categories = [
    'alt.atheism', 'comp.graphics', 'sci.space', 'talk.religion.misc',
    'rec.sport.baseball', 'rec.sport.hockey', 'sci.crypt', 'sci.electronics',
    'comp.os.ms-windows.misc', 'comp.sys.mac.hardware'
]
remove = ('headers', 'footers', 'quotes')

data_train = fetch_20newsgroups(subset='train', categories=categories,␣
 ↪remove=remove, shuffle=True, random_state=42)
data_test = fetch_20newsgroups(subset='test', categories=categories,␣
 ↪remove=remove, shuffle=True, random_state=42)

# extract Y and X from the datasets
y_train = data_train.target
y_test = data_test.target

X_train = data_train.data
X_test = data_test.data
```

[3]:
```
y_train
```

[3]:
```
array([3, 4, 2, …, 7, 7, 4], dtype=int64)
```

[4]:
```
# Check how each category is indexed
data_train.target_names
```

[4]:
```
['alt.atheism',
 'comp.graphics',
 'comp.os.ms-windows.misc',
 'comp.sys.mac.hardware',
 'rec.sport.baseball',
 'rec.sport.hockey',
 'sci.crypt',
 'sci.electronics',
 'sci.space',
 'talk.religion.misc']
```

[5]:
```
print(X_train[0]) # text
print(y_train) # integers (0-3)
```

```
Applied Engineering makes a NuBus card called the QuadraLink which is
a board that contains 4 serial ports, which I believe can be used
simultaneously.  I'm not a user of one of these, but I have installed
a couple for people at work (I'm a technician).  Hope this helps.

[3 4 2 … 7 7 4]
```

[6]:
```
# Convert texts to numerical vectors using TfidfVectorizer
vectorizer = TfidfVectorizer(stop_words='english')
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

# Check the size of X_train
print(X_train.shape)
print(X_test.shape)
```

```
(5586, 71611)
(3717, 71611)
```

[7]:
```
# Define parameter grid of SVC
tuned_parameters_svc = [
    {'kernel': ['rbf'], 'C': [1, 10, 100, 1000]},
    {'kernel': ['linear'], 'C': [1, 10, 100, 1000]},
    {'kernel': ['poly'], 'degree': [5, 5], 'C': [1, 10, 100]},
    {'kernel': ['sigmoid'], 'coef0': [0, 1, 2, 'C': [1, 10, 100]}
]
```

[8]:
```
# Define scoring metrics
scores = ['precision_macro', 'recall_macro']
```

[10]:
```
# GridSearch Optimize SVC
for score in scores:
    print(f"# Tuning hyper-parameters for {score}\n")
```

```python
clf_svc = GridSearchCV(SVC(), tuned_parameters_svc, scoring=score, cv=5)
clf_svc.fit(X_train, y_train)

print("Best parameters set found on train set:")
print(clf_svc.best_params_)
print("\nGrid scores on train set:")
means = clf_svc.cv_results_['mean_test_score']
stds = clf_svc.cv_results_['std_test_score']
for mean, std, params in zip(means, stds, clf_svc.cv_results_['params']):
    print(f"{mean:.3f} (+/-{std * 2:.03f}) for {params}")

print("\nDetailed classification report:")
print("The scores are computed on test set.")
y_pred = clf_svc.predict(X_test)
print(classification_report(y_test, y_pred, zero_division=1))
print()
```

```
# Tuning hyper-parameters for precision_macro


C:\ProgramData\anaconda3\Lib\site-
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))
C:\ProgramData\anaconda3\Lib\site-
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))
C:\ProgramData\anaconda3\Lib\site-
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))
C:\ProgramData\anaconda3\Lib\site-
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))
C:\ProgramData\anaconda3\Lib\site-
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))
C:\ProgramData\anaconda3\Lib\site-
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))
C:\ProgramData\anaconda3\Lib\site-
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))

Best parameters set found on train set:
{'C': 1, 'degree': 5, 'kernel': 'poly'}

Grid scores on train set:
0.809 (+/-0.031) for {'C': 1, 'kernel': 'rbf'}
0.815 (+/-0.029) for {'C': 10, 'kernel': 'rbf'}
0.815 (+/-0.029) for {'C': 100, 'kernel': 'rbf'}
0.815 (+/-0.029) for {'C': 1000, 'kernel': 'rbf'}
0.811 (+/-0.025) for {'C': 1, 'kernel': 'linear'}
0.801 (+/-0.034) for {'C': 10, 'kernel': 'linear'}
0.801 (+/-0.027) for {'C': 100, 'kernel': 'linear'}
0.802 (+/-0.022) for {'C': 1000, 'kernel': 'linear'}
0.800 (+/-0.022) for {'C': 1, 'degree': 3, 'kernel': 'poly'}
0.883 (+/-0.033) for {'C': 1, 'degree': 5, 'kernel': 'poly'}
0.846 (+/-0.031) for {'C': 10, 'degree': 3, 'kernel': 'poly'}
0.839 (+/-0.079) for {'C': 10, 'degree': 5, 'kernel': 'poly'}
0.878 (+/-0.045) for {'C': 100, 'degree': 3, 'kernel': 'poly'}
0.773 (+/-0.063) for {'C': 100, 'degree': 5, 'kernel': 'poly'}
0.809 (+/-0.029) for {'C': 1, 'coef0': 0, 'kernel': 'sigmoid'}
0.813 (+/-0.032) for {'C': 1, 'coef0': 1, 'kernel': 'sigmoid'}
0.783 (+/-0.025) for {'C': 10, 'coef0': 0, 'kernel': 'sigmoid'}
0.774 (+/-0.038) for {'C': 10, 'coef0': 1, 'kernel': 'sigmoid'}
0.778 (+/-0.020) for {'C': 100, 'coef0': 0, 'kernel': 'sigmoid'}
0.695 (+/-0.031) for {'C': 100, 'coef0': 1, 'kernel': 'sigmoid'}

Detailed classification report:
The scores are computed on test set.
              precision    recall  f1-score   support

           0       0.93      0.04      0.08       319
           1       0.90      0.12      0.21       389
           2       0.87      0.10      0.18       394
           3       0.88      0.10      0.18       385
           4       0.12      1.00      0.21       397
           5       0.96      0.20      0.33       399
           6       0.85      0.13      0.22       396
           7       0.81      0.06      0.10       393
           8       0.88      0.15      0.25       394
           9       0.00      0.00      0.00       251
```

```
    accuracy                           0.20      3717
   macro avg       0.72      0.19      0.18      3717
weighted avg       0.74      0.20      0.19      3717


# Tuning hyper-parameters for recall_macro

Best parameters set found on train set:
{'C': 1, 'kernel': 'linear'}

Grid scores on train set:
0.769 (+/-0.026) for {'C': 1, 'kernel': 'rbf'}
0.786 (+/-0.030) for {'C': 10, 'kernel': 'rbf'}
0.786 (+/-0.030) for {'C': 100, 'kernel': 'rbf'}
0.786 (+/-0.030) for {'C': 1000, 'kernel': 'rbf'}
0.788 (+/-0.029) for {'C': 1, 'kernel': 'linear'}
0.763 (+/-0.040) for {'C': 10, 'kernel': 'linear'}
0.695 (+/-0.029) for {'C': 100, 'kernel': 'linear'}
0.688 (+/-0.032) for {'C': 1000, 'kernel': 'linear'}
0.601 (+/-0.051) for {'C': 1, 'degree': 3, 'kernel': 'poly'}
0.255 (+/-0.022) for {'C': 1, 'degree': 5, 'kernel': 'poly'}
0.378 (+/-0.065) for {'C': 10, 'degree': 3, 'kernel': 'poly'}
0.172 (+/-0.034) for {'C': 10, 'degree': 5, 'kernel': 'poly'}
0.153 (+/-0.038) for {'C': 100, 'degree': 3, 'kernel': 'poly'}
0.118 (+/-0.011) for {'C': 100, 'degree': 5, 'kernel': 'poly'}
0.782 (+/-0.033) for {'C': 1, 'coef0': 0, 'kernel': 'sigmoid'}
0.723 (+/-0.034) for {'C': 1, 'coef0': 1, 'kernel': 'sigmoid'}
0.758 (+/-0.029) for {'C': 10, 'coef0': 0, 'kernel': 'sigmoid'}
0.763 (+/-0.036) for {'C': 10, 'coef0': 1, 'kernel': 'sigmoid'}
0.701 (+/-0.027) for {'C': 100, 'coef0': 0, 'kernel': 'sigmoid'}
0.658 (+/-0.022) for {'C': 100, 'coef0': 1, 'kernel': 'sigmoid'}

Detailed classification report:
The scores are computed on test set.
              precision    recall  f1-score   support

           0       0.59      0.59      0.59       319
           1       0.71      0.77      0.74       389
           2       0.80      0.68      0.74       394
           3       0.80      0.72      0.76       385
           4       0.83      0.83      0.83       397
           5       0.91      0.84      0.87       399
           6       0.88      0.72      0.79       396
           7       0.63      0.71      0.67       393
           8       0.57      0.81      0.67       394
           9       0.62      0.53      0.57       251
```

```
    accuracy                           0.73      3717
   macro avg       0.74      0.72      0.72      3717
weighted avg       0.74      0.73      0.73      3717
```

[ ]:
```
# I had trouble debugging the code above,
# I tried several different methods in defining the scoring metrics and editing␣
 ↪the GridSearchCV itself.
```

[14]:
```
# GridSearch Optimize NB
print("\nOptimizing Naive Bayes...")
nb_parameters = {
    'alpha': [0.1, 0.5, 1.0, 2.0, 5.0]
}

for score in scores:
    print(f"\nTuning NB hyper-parameters for {score}")
    nb_clf = GridSearchCV(
        MultinomialNB(),
        nb_parameters,
        cv=5,
        scoring=score  # Remove the _macro suffix since 'scores' already␣
 ↪includes it
    )
    nb_clf.fit(X_train, y_train)

    print("\nBest parameters found:")
    print(nb_clf.best_params_)

    print("\nGrid scores on training set:")
    means = nb_clf.cv_results_['mean_test_score']
    stds = nb_clf.cv_results_['std_test_score']
    for mean, std, params in zip(means, stds, nb_clf.cv_results_['params']):
        print(f"{mean:0.3f} (+/-{std*2:0.03f}) for {params}")

    print("\nDetailed classification report on test set:")
    y_pred = nb_clf.predict(X_test)
    print(classification_report(y_test, y_pred))
```

```
Optimizing Naive Bayes...

Tuning NB hyper-parameters for precision_macro

Best parameters found:
{'alpha': 0.1}
```

```
Grid scores on training set:
0.832 (+/-0.033) for {'alpha': 0.1}
0.813 (+/-0.032) for {'alpha': 0.5}
0.803 (+/-0.036) for {'alpha': 1.0}
0.790 (+/-0.057) for {'alpha': 2.0}
0.775 (+/-0.050) for {'alpha': 5.0}

Detailed classification report on test set:
              precision    recall  f1-score   support

           0       0.66      0.65      0.66       319
           1       0.75      0.78      0.76       389
           2       0.81      0.70      0.75       394
           3       0.80      0.77      0.79       385
           4       0.92      0.85      0.88       397
           5       0.71      0.94      0.81       399
           6       0.66      0.84      0.74       396
           7       0.83      0.68      0.75       393
           8       0.79      0.83      0.81       394
           9       0.75      0.43      0.54       251

    accuracy                           0.76      3717
   macro avg       0.77      0.75      0.75      3717
weighted avg       0.77      0.76      0.76      3717


Tuning NB hyper-parameters for recall_macro

Best parameters found:
{'alpha': 0.1}

Grid scores on training set:
0.810 (+/-0.028) for {'alpha': 0.1}
0.774 (+/-0.018) for {'alpha': 0.5}
0.755 (+/-0.021) for {'alpha': 1.0}
0.733 (+/-0.030) for {'alpha': 2.0}
0.699 (+/-0.025) for {'alpha': 5.0}

Detailed classification report on test set:
              precision    recall  f1-score   support

           0       0.66      0.65      0.66       319
           1       0.75      0.78      0.76       389
           2       0.81      0.70      0.75       394
           3       0.80      0.77      0.79       385
           4       0.92      0.85      0.88       397
           5       0.71      0.94      0.81       399
           6       0.66      0.84      0.74       396
```

7	0.83	0.68	0.75	393
8	0.79	0.83	0.81	394
9	0.75	0.43	0.54	251
accuracy			0.76	3717
macro avg	0.77	0.75	0.75	3717
weighted avg	0.77	0.76	0.76	3717