Лабораторна робота №4, Обробка та аналіз текстових даних на Python, Варіант 14

Виконав: студент групи ІП-11, Лошак Віктор Іванович
Перевірив: Юлія Тимофєєва Сергіївна

Тема роботи: Класифікація текстових даних
Мета роботи: Ознайомитись з класифікацією документів за допомогою моделей машинного навчання

16.03.2024

Завдання:
В якості текстової моделі використати TF-IDF. Виконати класифікацію за допомогою алгоритмів наївний байєсів класифікатор та логістичну регресію, порівняти їх точність. Спробувати покращити моделі за допомогою GridSearchCV.

Task:
Use TF-IDF as the text model. Perform classification using the naive Bayesian classifier and logistic regression algorithms, compare their accuracy. Try to improve the models with GridSearchCV.

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
import pandas as pd
```

In [ ]:
```
data = pd.read_csv('science.csv')
data.head()
```

Out[ ]:

	Id	Comment	Topic
0	0x840	A few things. You might have negative- frequen...	Biology
1	0xbf0	Is it so hard to believe that there exist part...	Physics
2	0x1dfc	There are bees	Biology
3	0xc7e	I'm a medication technician. And that's alot o...	Biology
4	0xbba	Cesium is such a pretty metal.	Chemistry

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(data['Comment'], data['Topic

tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

nb_classifier = MultinomialNB()
```

```
nb_classifier.fit(X_train_tfidf, y_train)
nb_pred = nb_classifier.predict(X_test_tfidf)
nb_accuracy = accuracy_score(y_test, nb_pred)

lr_classifier = LogisticRegression(random_state=42, max_iter=1000)
lr_classifier.fit(X_train_tfidf, y_train)
lr_pred = lr_classifier.predict(X_test_tfidf)
lr_accuracy = accuracy_score(y_test, lr_pred)

nb_accuracy, lr_accuracy
```

Out[ ]: (0.7201993100804907, 0.7041011881947106)

Використаємо grid search для підбору параметрів.

In [ ]:
```
nb_pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words='english')),
    ('nb', MultinomialNB())
])
nb_params = {
    'tfidf__max_df': (0.3, 0.4, 0.5),
    'tfidf__ngram_range': [(1, 1), (1, 2), (1,3)],  # unigrams or bigrams
    'nb__alpha': (0.1, 0.15, 0.2)
}

grid_nb = GridSearchCV(nb_pipeline, nb_params, cv=5, scoring='accuracy')
grid_nb.fit(X_train, y_train)

nb_best_score = grid_nb.best_score_
nb_best_params = grid_nb.best_params_
nb_best_score, nb_best_params
```

Out[ ]:
```
(0.7122901749031577,
 {'nb__alpha': 0.2, 'tfidf__max_df': 0.3, 'tfidf__ngram_range': (1, 1)})
```

In [ ]:
```
lr_pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words='english')),
    ('lr', LogisticRegression(random_state=42, max_iter=300))
])
lr_params = {
    'tfidf__max_df': (0.3, 0.4, 0.5),
    'tfidf__ngram_range': [(1, 2), (1,3)],
    'lr__C': (10, 20, 30)
}

grid_lr = GridSearchCV(lr_pipeline, lr_params, cv=5, scoring='accuracy')
grid_lr.fit(X_train, y_train)

lr_best_score = grid_lr.best_score_
lr_best_params = grid_lr.best_params_
lr_best_score, lr_best_params
```

Out[ ]:
```
(0.689124107977705,
 {'lr__C': 30, 'tfidf__max_df': 0.3, 'tfidf__ngram_range': (1, 2)})
```

Використаємо параметри знайдені за допомогою Grid search.

In [ ]:
```
best_nb_model = grid_nb.best_estimator_
nb_predictions = best_nb_model.predict(X_test)
```

```
best_lr_model = grid_lr.best_estimator_
lr_predictions = best_lr_model.predict(X_test)

nb_accuracy = accuracy_score(y_test, nb_predictions)
lr_accuracy = accuracy_score(y_test, lr_predictions)

print("Naive Bayes Accuracy:", nb_accuracy)
print("Logistic Regression Accuracy:", lr_accuracy)
```

```
Naive Bayes Accuracy: 0.745879647374473
Logistic Regression Accuracy: 0.7167497125335378
```

Отримані параметри надають більшу точність класифікації ніж значення за замовчуванням для обох класифікаторів.

Висновок:

В ході виконання даної лабораторної роботи я ознайомився з процесом класифікації текстових даних за допомогою алгоритмів машинного навчання. Була використана модель TF-IDF для перетворення текстових даних у векторний формат, що є необхідним для роботи з алгоритмами класифікації. Для аналізу ефективності класифікації були обрані два алгоритми: наївний байєсів класифікатор та логістична регресія.

Початкове порівняння показало, що обидва алгоритми мають схожу точність класифікації, але завдяки застосуванню GridSearchCV для оптимізації гіперпараметрів вдалося значно покращити результати. GridSearchCV дозволило автоматично підібрати найкращі параметри для кожного з алгоритмів, що сприяло підвищенню точності класифікації.

На підставі отриманих результатів можна зробити висновок, що правильний підбір гіперпараметрів і використання оптимізаційних інструментів, таких як GridSearchCV, відіграють ключову роль у підвищенні ефективності моделей машинного навчання. Отримані знання та навички можуть бути застосовані для вирішення реальних задач аналізу та класифікації текстових даних.