Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет Інформатики та Обчислювальної Техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 4

з дисципліни «Обробка та аналіз текстових даних на Python»

На тему:

«Класифікація текстових даних»

Варіант №3

Виконала:
студентка групи ІС-12.
Мельникова К.О.
Перевірила:
Тимофєєва Ю. С.

Київ – 2024

Мета роботи: Ознайомитись з класифікацією документів за допомогою моделей машинного навчання.

Завдання до лабораторної роботи

Створити програму, яка зчитує заданий набір даних, виконує попередню обробку та класифікацію документів відповідно до варіанту. Якщо недостатньо ресурсів для роботи з повним набором даних, можна виділити частину, але таким чином, щоб були присутні усі класи.

В якості текстової моделі використати Word2Vec. Виконати класифікацію за допомогою алгоритмів випадкові ліси та опорні вектори, порівняти їх точність. Спробувати покращити моделі за допомогою GridSearchCV.

Код програми:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from gensim.models import Word2Vec
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
import numpy as np

# Зчитуємо дані з .csv файлу
data = pd.read_csv('bbc-news-data.csv', sep=r'\t', on_bad_lines='skip')

# Розділяємо дані на ознаки (X) та цільову змінну (y)
X = data['content']
y = data['category']

# Розділяємо дані на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Токенізація тексту перед побудовою Word2Vec моделі
tokenized_X_train = [text.split() for text in X_train]

# Побудова Word2Vec моделі
w2v_model = Word2Vec(X_train, vector_size=300, window=5, min_count=1, workers=4)
```

```
def document_vectorizer(corpus, model, num_features):
   vocabulary = set(model.wv.index_to_key)
   def average_word_vectors(words, model, vocabulary, num_features):
       feature_vector = np.zeros((num_features,), dtype="float64")
       nwords = 0.
       for word in words:
           if word in vocabulary:
               nwords = nwords + 1.
               feature_vector = np.add(feature_vector, model.wv[word])
       if nwords:
           feature_vector = np.divide(feature_vector, nwords)

       return feature_vector
   features = [average_word_vectors(tokenized_sentence, model, vocabulary,
num_features) for tokenized_sentence in corpus]
   return np.array(features)

# Побудова векторів для тренувального та тестового наборів даних
X_train_word_average = document_vectorizer(X_train, w2v_model, num_features=300)
X_test_word_average = document_vectorizer(X_test, w2v_model, num_features=300)

# Побудова моделі випадкових лісів
rfc = RandomForestClassifier()
rfc.fit(X_train_word_average, y_train)
rfc_predictions = rfc.predict(X_test_word_average)
rfc_accuracy = accuracy_score(y_test, rfc_predictions)
print("Random Forest Classifier Accuracy:", rfc_accuracy)

# Побудова моделі опорних векторів
svm = SVC()
svm.fit(X_train_word_average, y_train)
svm_predictions = svm.predict(X_test_word_average)
svm_accuracy = accuracy_score(y_test, svm_predictions)
print("Support Vector Machine Accuracy:", svm_accuracy)

# Налаштування моделей за допомогою GridSearchCV
param_grid = {'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001], 'kernel':
['rbf', 'linear']}
grid_search_svm = GridSearchCV(SVC(), param_grid, refit=True, verbose=2)
grid_search_svm.fit(X_train_word_average, y_train)
best_svm = grid_search_svm.best_estimator_
best_svm_predictions = best_svm.predict(X_test_word_average)
best_svm_accuracy = accuracy_score(y_test, best_svm_predictions)
print("Best SVM Accuracy after GridSearchCV:", best_svm_accuracy)
```

```
# Порівняння точності до та після налаштування
print("SVM Accuracy Improvement:", best_svm_accuracy - svm_accuracy)

# Налаштування моделі RandomForest за допомогою GridSearchCV
param_grid = {'n_estimators': [10, 50, 100], 'bootstrap': [True, False]}
grid_search_rfc = GridSearchCV(RandomForestClassifier(), param_grid, refit=True,
verbose=2)
grid_search_rfc.fit(X_train_word_average, y_train)
best_rfc = grid_search_rfc.best_estimator_
best_rfc_predictions = best_rfc.predict(X_test_word_average)
best_rfc_accuracy = accuracy_score(y_test, best_rfc_predictions)
print("Best Random Forest Classifier Accuracy after GridSearchCV:",
best_rfc_accuracy)

# Порівняння точності до та після налаштування
print("Random Forest Classifier Accuracy Improvement:", best_rfc_accuracy -
rfc_accuracy)
```

Результат виконання до покращення:

```
Random Forest Classifier Accuracy: 0.7410179640718563
Support Vector Machine Accuracy: 0.5658682634730539
```

Після покращення:

```
Best SVM Accuracy after GridSearchCV: 0.7859281437125748
SVM Accuracy Improvement: 0.22005988023952094
```

```
Best Random Forest Classifier Accuracy after GridSearchCV: 0.7485029940119761
Random Forest Classifier Accuracy Improvement: 0.010479041916167664
```