Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

Лабораторная работа № 4

По дисциплине: «Прикладные интеллектуальные системы и экспертные системы**»**

Классификация текстовых данных

Студент Александрук А.М.

Группа М-ИАП-23-1

Руководитель Кургасов В.В.
к.т.н. доцент

Липецк 2023г.

Цель работы:

Получить практические навыки решения задачи классификации текстовых данных в среде Jupiter Notebook. Научиться проводить предварительную обработку текстовых данных, настраивать параметры методов классификации и обучать модели, оценивать точность полученных моделей.

Задание кафедры

1) Загрузить выборки по варианту из лабораторной работы №2

2) Используя GridSearchCV произвести предварительную обработку данных и настройку методов классификации в соответствие с заданием, вывести оптимальные значения параметров и результаты классификации модели (полнота, точность, f1-мера и аккуратности) с данными параметрами. Настройку проводить как на данных со стеммингом, так и на данных, на которых стемминг не применялся.

3) По каждому пункту работы занести в отчет программный код и результат вывода.

4) Оформить сравнительную таблицу с результатами классификации различными методами с разными настройками. Сделать выводы о наиболее подходящем методе классификации ваших данных с указанием параметров метода и описанием предварительной обработки данных.

Вариант 1

Методы: KNN, RF, LR ;

Название класса: [comp.graphics, comp.os.ms-windows.misc, rec.autos]

Ход работы:

Подключаем все возможные библиотеки, которые потребуется для выполнения лабораторной работы. Все данные лабораторной работы и ее выполнение рассматриваются на рисунках. Загружаем данные dataset в наш код из 3 лабораторной работы (2 лабораторной работы). Зададим параметры для наших методов и запустим программу.

[1]:
```
# Лабораторная работа 3
# Вариант 1
# Используемые методы (KNN,RF,LR)
# Подключение библиотек
import pandas as pd
from sklearn.datasets import fetch_20newsgroups
import nltk
from nltk import word_tokenize
from nltk.stem import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.datasets import fetch_20newsgroups
import nltk
from sklearn.preprocessing import StandardScaler

# Выбор классов
categories = ['comp.graphics', 'comp.os.ms-windows.misc', 'rec.autos']
remove = ('headers', 'footers', 'quotes')

twenty_train = fetch_20newsgroups(subset='train', shuffle=True, random_state=35, categories=categories, remove=remove)
twenty_test = fetch_20newsgroups(subset='test', shuffle=True, random_state=35, categories=categories, remove=remove)
print(twenty_train.data[3])
print(twenty_test.data[3])

nltk.download('punkt')

# Применить стемминг
def stemn(data):
    porter_stemmer = PorterStemmer()
    stem = []
    for text in data:
        nltk_tokens = word_tokenize(text)
        line = ''.join([' ' + porter_stemmer.stem(word) for word in nltk_tokens])
        stem.append(line)
    return stem

porter_stemmer = PorterStemmer()
stem_train = []
for text in twenty_train.data:
    nltk_tokens = word_tokenize(text)
    line = ''
    for word in nltk_tokens:
        line += ' ' + porter_stemmer.stem(word)
    stem_train.append(line)
print(stem_train[0])

stem_test = stemn(twenty_test.data)
```

Рисунок 1 – Исходные данные

```
[3]: X_train, X_test, y_train, y_test = stem_train, stem_test, twenty_train.target, twenty_test.target

# Пайплайн для KNN
knn_pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf', KNeighborsClassifier())
])

# Параметры для KNN
knn_param_grid = {'vect__ngram_range': [(1, 1), (1, 2)],
                  'tfidf__use_idf': (True, False),
                  'clf__n_neighbors': [3, 5, 7]}

# Пайплайн для RF
rf_pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf', RandomForestClassifier())
])

# Параметры для RF
rf_param_grid = {'vect__ngram_range': [(1, 1), (1, 2)],
                 'tfidf__use_idf': (True, False),
                 'clf__n_estimators': [50, 100, 200]}


# Параметры для LR
lr_pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('scaler', StandardScaler(with_mean=False)),
    ('clf', LogisticRegression(max_iter=1000))
])

# Параметры для  LR
lr_param_grid = {'vect__ngram_range': [(1, 1), (1, 2)],
                 'tfidf__use_idf': (True, False),
                 'clf__C': [0.1, 1, 10]}

# Список параметров
pipelines = [knn_pipeline, rf_pipeline, lr_pipeline]
param_grids = [knn_param_grid, rf_param_grid, lr_param_grid]
```

Рисунок 2 – Настройка параметров

```
# Цикл обучения
for i, pipeline in enumerate(pipelines):
    print(f"Оптимизация параметров для {pipeline.named_steps['clf'].__class__.__name__}")

    # Данные со стеммингом
    grid_search_stem = GridSearchCV(pipeline, param_grids[i], cv=5, scoring='accuracy', n_jobs=-1)
    grid_search_stem.fit(X_train, y_train)

    print("Лучшие параметры с использованием стемминга:", grid_search_stem.best_params_)
    y_pred_stem = grid_search_stem.predict(X_test)
    print("Точность с использованием стемминга:", accuracy_score(y_test, y_pred_stem))
    print("Точность с использованием стемминга:", precision_score(y_test, y_pred_stem, average='weighted'))
    print("Полнота с использованием стемминга:", recall_score(y_test, y_pred_stem, average='weighted'))
    print("F1-мера с использованием стемминга:", f1_score(y_test, y_pred_stem, average='weighted'))

    # Данные без стемминга
    grid_search_no_stem = GridSearchCV(pipeline, param_grids[i], cv=5, scoring='accuracy', n_jobs=-1)
    grid_search_no_stem.fit(twenty_train.data, twenty_train.target)

    print("Лучшие параметры без стемминга:", grid_search_no_stem.best_params_)
    y_pred_no_stem = grid_search_no_stem.predict(twenty_test.data)
    print("Точность без стемминга:", accuracy_score(twenty_test.target, y_pred_no_stem))
    print("Точность без стемминга:", precision_score(twenty_test.target, y_pred_no_stem, average='weighted'))
    print("Полнота без стемминга:", recall_score(twenty_test.target, y_pred_no_stem, average='weighted'))
    print("F1-мера без стемминга:", f1_score(twenty_test.target, y_pred_no_stem, average='weighted'))
    print("\n")
```

Оптимизация параметров для KNeighborsClassifier

Рисунок 3 – Цикл обучения программы для всех методов

Получаем результаты на выходе.

```
Оптимизация параметров для KNeighborsClassifier
Лучшие параметры с использованием стемминга: {'clf__n_neighbors': 5, 'tfidf__use_idf': False, 'vect__ngram_range': (1, 1)}
Точность с использованием стемминга: 0.5131467345207803
Точность с использованием стемминга: 0.5676424720913685
Полнота с использованием стемминга: 0.5131467345207803
F1-мера с использованием стемминга: 0.5019390442761362
Лучшие параметры без стемминга: {'clf__n_neighbors': 5, 'tfidf__use_idf': False, 'vect__ngram_range': (1, 1)}
Точность без стемминга: 0.47837150127226463
Точность без стемминга: 0.5142921279053392
Полнота без стемминга: 0.47837150127226463
F1-мера без стемминга: 0.4701362633442179


Оптимизация параметров для RandomForestClassifier
Лучшие параметры с использованием стемминга: {'clf__n_estimators': 200, 'tfidf__use_idf': False, 'vect__ngram_range': (1, 1)}
Точность с использованием стемминга: 0.8125530110262935
Точность с использованием стемминга: 0.8170167261958017
Полнота с использованием стемминга: 0.8125530110262935
F1-мера с использованием стемминга: 0.8102399764430948
Лучшие параметры без стемминга: {'clf__n_estimators': 200, 'tfidf__use_idf': False, 'vect__ngram_range': (1, 1)}
Точность без стемминга: 0.818490245971162
Точность без стемминга: 0.8248192181601309
Полнота без стемминга: 0.818490245971162
F1-мера без стемминга: 0.8166716958601928


Оптимизация параметров для LogisticRegression
Лучшие параметры с использованием стемминга: {'clf__C': 0.1, 'tfidf__use_idf': True, 'vect__ngram_range': (1, 2)}
Точность с использованием стемминга: 0.8210347752332485
Точность с использованием стемминга: 0.8234307425514724
Полнота с использованием стемминга: 0.8210347752332485
F1-мера с использованием стемминга: 0.8188652867787675
Лучшие параметры без стемминга: {'clf__C': 0.1, 'tfidf__use_idf': True, 'vect__ngram_range': (1, 1)}
Точность без стемминга: 0.8363019508057676
Точность без стемминга: 0.835902120081434
Полнота без стемминга: 0.8363019508057676
F1-мера без стемминга: 0.8343603023621231
```

Рисунок 4 – Вывод результатов

Код программы: https://github.com/lipadirka/Prikladnie-intelektualn-systems.git

Вывод

В ходе выполнения лабораторной работы были использованы методы KNN, RF, LR. В результате работы наилучшей точностью обладает метод логистической регрессии без стемминга, в том числе и со стеммингом показал лучший результат по сравнению с другими методами.