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 Мастылина А.А.

Группа М-ИАП-22

Руководитель Кургасов В.В.

Липецк 2022 г.

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

Ход работы

1) Загрузить выборки по варианту из лабораторной работы №2

- pandas - предоставляет специальные структуры данных и операции для манипулирования числовыми таблицами и временными рядами.

- sklearn - включает все алгоритмы и инструменты, которые нужны для задач классификации, регрессии и кластеризации, методы оценки производительности модели машинного обучения.

- numpy - поддерживает многомерные массивы, высокоуровневые математические функций, предназначенные для работы с многомерными массивами

- pyplot - это коллекция функций в стиле команд, которая позволяет использовать matplotlib почти так же, как MATLAB

- nltk - пакет библиотек и программ для символьной и статистической обработки естественного языка, написанных на языке программирования Python.

- модуль itertools стандартизирует основной набор быстрых эффективных по памяти инструментов, которые полезны сами по себе или в связке с другими инструментами.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import MultinomialNB
from nltk.stem import *
from nltk import word_tokenize
import itertools
```

Рисунок 1 – Необходимые библиотеки

```
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import LinearSVC
from sklearn import metrics

from sklearn.utils._testing import ignore_warnings
from sklearn.exceptions import FitFailedWarning, ConvergenceWarning
```

Рисунок 2 – Необходимые библиотеки

```
categories = ['comp.sys.mac.hardware', 'soc.religion.christian', 'talk.religion.misc']
remove = ['headers', 'footers', 'quotes']
twenty_train = fetch_20newsgroups(subset='train', shuffle=True, random_state=42, categories=categories, remove=remove)
twenty_test = fetch_20newsgroups(subset='test', shuffle=True, random_state=42, categories=categories, remove=remove)
```

Рисунок 3 – Выгрузка данных по варианту

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.

```
%%time
parameters = {
  'KNeighborsClassifier': {
    'vect__max_features': (1000,5000,10000),
    'vect__stop_words': ('english', None),
    'tfidf__use_idf': (True, False),
    'clf__n_neighbors': (1, 3, 5, 10),
    'clf__p': (1, 2)
  },
  'DecisionTreeClassifier': {
    'vect__max_features': (1000,5000,10000),
    'vect__stop_words': ('english', None),
    'tfidf__use_idf': (True, False),
    'clf__criterion': ('gini', 'entropy'),
    'clf__max_depth': [*range(1,5,1), *range(5,101,20)]
  },
  'LinearSVC': [{
```

```
            'vect__max_features': (1000,5000,10000),
            'vect__stop_words': ('english', None),
            'tfidf__use_idf': (True, False),
            'clf__loss': ['squared_hinge'],
            'clf__penalty': ('l1', 'l2')
        },
            {
            'vect__max_features': (1000,5000,10000),
            'vect__stop_words': ('english', None),
            'tfidf__use_idf': (True, False),
            'clf__loss': ['hinge'],
            'clf__penalty': ['l2']
        }],
    }

    gs = {}
    for clf, param in parameters.items():
        text_clf = Pipeline([
            ('vect', CountVectorizer()),
            ('tfidf', TfidfTransformer()),
            ('clf', eval(clf)())
        ])
        gs[clf] = GridSearchCV(text_clf, param, n_jobs=-1, error_score=0.0)
        gs[clf].fit(X = twenty_train['data'], y = twenty_train['target'])
    for clf, param in parameters.items():
        predicted = gs[clf].predict(twenty_test['data'])
        print(metrics.classification_report(twenty_test.target, predicted,
target_names=categories))
    r = {}
    def highlight_max(x, color):
        return np.where(x == np.nanmax(x.to_numpy()), f"color: {color};", None)
    total_style = pd.Series("font-weight: bold;", index=[1])
    for clf, param in parameters.items():
        predicted = gs[clf].predict(twenty_test['data'])
```

```
pd.DataFrame(gs[clf].cv_results_).to_excel('all' + clf + '.xlsx')
pd.DataFrame(classification_report(predicted, twenty_test.target, output_dict=True)).to_excel('best' + clf + '.xlsx')
```

В коде представлены параметры и ограничения, по которым будет проводится поиск по сетке.

3) Оформим сравнительную таблицу с результатами классификации различными методами.

Таблица 1 – Итоговая таблица

	0	1	2	accuracy	macro avg	weighted avg
precision	0,927273	0,660804	0,38247	0,692456	0,656848956	0,738369877
recall	0,730061	0,720548	0,533333	0,692456	0,661314209	0,69245648
f1-score	0,816934	0,689384	0,445476	0,692456	0,650597762	0,707244998
support	489	365	180	0,692456	1034	1034
	0	1	2	accuracy	macro avg	weighted avg
precision	0,58961	0,68593	0,219124	0,53675	0,498221181	0,579946926
recall	0,610215	0,541667	0,348101	0,53675	0,499994329	0,536750484
f1-score	0,599736	0,605322	0,268949	0,53675	0,491335321	0,551912616
support	372	504	158	0,53675	1034	1034
	0	1	2	accuracy	macro avg	weighted avg
precision	0,979221	0,861809	0,438247	0,802708	0,759759	0,853583
recall	0,878788	0,736052	0,791367	0,802708	0,802069	0,802708
f1-score	0,92629	0,793981	0,564103	0,802708	0,761458	0,817973
support	429	466	139	0,802708	1034	1034

Таким образом, лучшим методом оказался LinearSVC, так как значение аккуратности равняется 0,8, худшим методом оказался KNeighborsClassifier, так как значение аккуратности равняется 0,54.