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 Осипов 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.

- 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.

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

```
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

```
categories = ['comp.os.ms-windows.misc', 'comp.sys.mac.hardware', 'sci.space']
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 2 – 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'])
```

3) Оформим сравнительную таблицу с результатами классификации различными методами.

	0	1	2	accuracy	macro avg	weighted avg
precision	0,677665	0,65974	0,903553	0,747656	0,746986178	0,776055369
recall	0,831776	0,803797	0,664179	0,747656	0,766584091	0,747655584
f1-score	0,746853	0,724679	0,765591	0,747656	0,745707858	0,749441963
support	321	316	536	0,747656	1173	1173

	0	1	2	accuracy	macro avg	weighted avg
precision	0,365482	0,277922	0,862944	0,503836	0,502116158	0,67981947
recall	0,637168	0,611429	0,440415	0,503836	0,56300374	0,50383632
f1-score	0,464516	0,382143	0,58319	0,503836	0,47661646	0,53033131
support	226	175	772	0,503836	1173	1173

	0	1	2	accuracy	macro avg	weighted avg
precision	0,784264	0,815584	0,928934	0,843137	0,842927462	0,850813166
recall	0,885387	0,865014	0,793926	0,843137	0,848108947	0,843137255
f1-score	0,831763	0,839572	0,85614	0,843137	0,842491889	0,843760219
support	349	363	461	0,843137	1173	1173

Рисунок 3 – Итоговая таблица

Таким образом, лучшим методом оказался LinearSVC, так как значение аккуратности равняется 0,84, худшим методом оказался KNeighborsClassifier, так как значение аккуратности равняется 0,50.