Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине

«Прикладные интеллектуальные системы и экспертные системы»

Студент Курдюков И.Ю.

Группа М-ИАП-23-1

Руководитель Кургасов В. В.

доцент, канд. пед. наук

Липецк 2023 г.

Цель работы:

Получить практические навыки решения задачи бинарной классификации данных в среде Jupiter Notebook. Научиться загружать данные, обучать классификаторы и проводить классификацию. Научиться оценивать точность полученных моделей.

Задание кафедры

1. В среде Jupiter Notebook создать новый ноутбук (Notebook)
2. Импортировать необходимые для работы библиотеки и модули
3. Загрузить данные в соответствии с вариантом
4. Вывести первые 15 элементов выборки (координаты точек и метки класса)
5. Отобразить на графике сгенерированную выборку. Объекты разных классов должны иметь разные цвета.
6. Разбить данные на обучающую (train) и тестовую (test) выборки в пропорции 75% — 25% соответственно.
7. Отобразить на графике обучающую и тестовую выборки. Объекты разных классов должны иметь разные цвета.
8. Реализовать модели классификаторов, обучить их на обучающем множестве. Применить модели на тестовой выборке, вывести результаты классификации:
   - Истинные и предсказанные метки классов
   - Матрицу ошибок (confusion matrix)
   - Значения полноты, точности, f1-меры и аккуратности
   - Значение площади под кривой ошибок (AUC ROC)
   - Отобразить на графике область принятия решений по каждому классу

   В качестве методов классификации использовать:

   a) Метод к-ближайших соседей (n_neighbors = {1, 3, 5, 9})
   b) Наивный байесовский метод
   c) Случайный лес (n_estimators = {5, 10, 15, 20, 50})
9. По каждому пункту работы занести в отчет программный код и результат вывода.
10. По результатам п.8 занести в отчет таблицу с результатами классификации всеми методами и выводы о наиболее подходящем методе классификации ваших данных.
11. Изучить, как изменится качество классификации, если на тестовую часть выделить 10% выборки, 35% выборки. Для этого повторить п.п. 6 – 10.

Ход работы

На Листинге 1 представлены все необходимые и импортированные библиотеки.

```
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer,
TfidfTransformer
```

Загрузка выборки

```
from sklearn.datasets import fetch_20newsgroups

remove = ('headers', 'footers', 'quotes')


def get_train_data(categories):
    if type(categories) is not list:
        categories = [categories]
    return fetch_20newsgroups(subset='train', shuffle=True,
categories=categories, random_state=42, remove=remove)


all_categories = ['comp.graphics', 'rec.sport.baseball',
'sci.electronics']
train_bunch = get_train_data(all_categories)
test_bunch = fetch_20newsgroups(subset='test', shuffle=True,
random_state=42, categories=all_categories, remove=remove)


def get_sample(bunch, category_idx):
    for idx, target in enumerate(bunch.target):
        if target == category_idx:
            return bunch.data[idx]
```

Вывод слов из документов

```
get_sample(train_bunch, all_categories.index('comp.graphics'))
```

```
'I'm interested in simulating reverse (or negative) color video\nmathematically.  What is the transform?  Is it a simple\nreversal of the hue value in the HSV color space?  I
s it\na manipulation in the YUV color space?  How is it related\nto solarization?\n\nIf you want to see something truly wild, turn on the\nreverse video effect on a camcorder
so equipped,\nand point it at the monitor.  This creates a chaotic\ndynamical system whose phase space is continuous along\nrotation, zoom, focus, etc.  Very very surprising
and \nlovely.  I'd like to write a simulation of this effect\nwithout analog grunge.  Thanks for any info you may have.\n\nPlease e-mail any info to me.  I'll post a summar
y.\n\nThanks,\n\n-- '
```

Рисунок 1 – Результат

```
get_sample(train_bunch, all_categories.index('rec.sport.baseball'))
```

```
'I had heard the rumors about LA, Cin, Hou, and SD all being\ninterested in Mark Davis, so it doesn\'t surprise me that a\nteam had to give up something and cash to actually get him.\n\nLynch "MOB"'
```

Рисунок 2 – Результат

```
get_sample(train_bunch, all_categories.index('sci.electronics'))
```

```
'I had the insturment panel go out in my car (a 1990 Lincoln Contenintal) which\nis a digital dash. They replaced the whole thing with a 1991 dash (thank god it\nwas under th
e warrenty ! :-) Anyway, the odometer was reading the exact milage\nfrom the old panel. It must have a EEPROM of some sort in it that is up-dated.\nSeems to me that removing
the battery would erase it, but it doesn't. So I\nguess they swapped the NVM chip (non-volitile memory) and installed it in the\nnew dash. No, they wouldn't let me have the o
ld dash to tinker with :-(\n'
```

Рисунок 3– Результат

Разделение выборки

```
import nltk
from nltk.stem import *
from nltk import word_tokenize

nltk.download('punkt')


def stemminize(documents: list[str]) -> list[str]:
    porter_stemmer = PorterStemmer()
    stem_train = []
    for document in documents:
        nltk_tokens = word_tokenize(document)
        line = ''
        for word in nltk_tokens:
            line += ' ' + porter_stemmer.stem(word)
        stem_train.append(line)
    return stem_train


train_tokenized = stemminize(train_bunch.data)
test_tokenized = stemminize(test_bunch.data)
```

Используем наивный байесовский метод

```
# Наивный байесовский метод

def naiveBayes(x, y, test_size = 0.25):
  x_train, x_test, y_train, y_test = train_test_split(x, y, test_size =
test_size, random_state = 1)
  gnb = GaussianNB()
  gnb.fit(x_train, y_train)
  prediction = gnb.predict(x_test)

  print ('Prediction and test:')
  print ('Prediction: \t', prediction)
  print ('Test: \t\t', y_test)
  print("\n\n")
```

```
  print ('Confusion matrix: ')
  print (confusion_matrix(y_test, prediction)[0])
  print (confusion_matrix(y_test, prediction)[1])
  print("\n\n")
  print ('Accuracy score: ', accuracy_score(prediction, y_test))
  print("\n\n")
  print('Classification Report\n', classification_report(y_test,
prediction))
  print("\n\n")
  print('ROC AUC')
  print(roc_auc_score(y_test, prediction))
  print("\n\n")

  # обучающая и тестовая выборки
  plt.title('Division into training (Blue) and test (Red) samples')
  plt.scatter (x_train[:, 0], x_train[:, 1], color = 'blue')
  plt.scatter (x_test[:, 0], x_test[:, 1], color = 'red')
  plt.show()

  plt.xlabel("first feature")
  plt.ylabel("second feature")
  plot_2d_separator(gnb, x, fill=True)
  plt.scatter(x[:, 0], x[:, 1], c=y, s=70)
```

Применим процедуру отсечения стоп-слов и повторим вывод полученных результатов. Код для обработки данных путем отсечения стоп-слов. Представлено на рисунке 4.

```
vect = CountVectorizer(max_features=10000)
train_data = vect.fit_transform(train_bunch.data)


def get_20_freq_words(vect, data):
    words = list(zip(vect.get_feature_names_out(),
np.ravel(data.sum(axis=0))))
    words.sort(key=lambda x: x[1], reverse=True)
    return words[:20]


count_column = get_20_freq_words(vect, train_data)
df_train['Count', 'Без стоп-слов'] = count_column
count_column
```

```
[('the', 11217),
 ('to', 5625),
 ('and', 4649),
 ('of', 4275),
 ('is', 3512),
 ('in', 3298),
 ('for', 2692),
 ('it', 2578),
 ('that', 2456),
 ('you', 2210),
 ('on', 1742),
 ('this', 1636),
 ('be', 1581),
 ('have', 1505),
 ('with', 1494),
 ('are', 1410),
 ('or', 1393),
 ('if', 1301),
 ('as', 1189),
 ('but', 1182)]
```

Рисунок 4 – Результат

```
import nltk
from nltk.stem import *
from nltk import word_tokenize

nltk.download('punkt')


def stemminize(documents: list[str]) -> list[str]:
    porter_stemmer = PorterStemmer()
    stem_train = []
    for document in documents:
        nltk_tokens = word_tokenize(document)
        line = ''
        for word in nltk_tokens:
            line += ' ' + porter_stemmer.stem(word)
        stem_train.append(line)
    return stem_train


train_tokenized = stemminize(train_bunch.data)
test_tokenized = stemminize(test_bunch.data)
```

Воспользуемся векторизацией выборки с помощью TfidfTransformer (с использованием TF и TF-IDF взвешиваний). Векторизация выборки с использованием TfidfTransformer для набора данных без использования

стоп-слов представлен на рисунке 5, с использованием стоп-слов представлен на рисунке 6.

```
def get_20_freq_words_idf(feature_names, tfidf_values):
    result = []
    word_weights = dict(zip(feature_names, tfidf_values))
    sorted_words = sorted(word_weights.items(), key=lambda x: x[1],
reverse=True)
    for word, weight in sorted_words[:20]:
        result.append((word, weight))
    return result


vectorizer = CountVectorizer(max_features=10000)
dtm = vectorizer.fit_transform(train_bunch.data)
tfidf = TfidfTransformer(use_idf=False).fit_transform(dtm)

feature_names = vectorizer.get_feature_names_out()
tfidf_values = tfidf.toarray().sum(axis=0)

tf_column = get_20_freq_words_idf(feature_names, tfidf_values)
df_train['TF', 'Без стоп-слов'] = tf_column
tf_column
```

```
[('the', 533.4922807849437),
 ('to', 275.8748829930202),
 ('and', 210.74679041879125),
 ('of', 203.27421480192316),
 ('in', 168.05567850652892),
 ('is', 165.6458215892349),
 ('for', 139.50376330402037),
 ('it', 137.5933837024795),
 ('that', 128.1830973532055),
 ('you', 106.7291231981497),
 ('on', 90.99388015313521),
 ('have', 87.11801374241186),
 ('this', 86.68927655273733),
 ('be', 77.9877263998729),
 ('with', 70.49674206221073),
 ('if', 69.45979671449946),
 ('are', 68.55843840506661),
 ('or', 66.36867298924544),
 ('but', 65.53939178170177),
 ('can', 57.68500973851248)]
```

Рисунок 6 – Результат

```
vectorizer = CountVectorizer(max_features=10000, stop_words='english')
dtm = vectorizer.fit_transform(test_bunch.data)
```

```
tfidf = TfidfTransformer(use_idf=True).fit_transform(dtm)

feature_names = vectorizer.get_feature_names_out()
tfidf_values = tfidf.toarray().sum(axis=0)

tf_idf_stop_test = get_20_freq_words_idf(feature_names, tfidf_values)
df_test['TF-IDF', 'С стоп-словами'] = tf_idf_stop_test
tf_idf_stop_test
```

```
[('know', 21.58985409837126),
 ('like', 19.0432238776313),
 ('thanks', 17.243050835722823),
 ('does', 17.21331485116614),
 ('just', 16.893197620659922),
 ('don', 16.827878554078698),
 ('think', 15.961774996967483),
 ('graphics', 14.152144747044698),
 ('time', 14.027722341198748),
 ('game', 14.003738256027455),
 ('use', 13.74412905058512),
 ('edu', 12.529370396463287),
 ('program', 12.14189017243623),
 ('good', 11.781849219177335),
 ('need', 11.678500624748112),
 ('ve', 11.055266820963013),
 ('image', 10.88628915030409),
 ('help', 10.859661452807238),
 ('year', 10.809326893529198),
 ('games', 10.683050419564378)]
```

Рисунок 7 – Результат

Составим сводную таблицу для отображения результатов векторизации и сохраним её в файл Excel. Составленная таблица для обучающего набора данных без применения стемминга представлена на рисунке 8. Для тестового набора данных без применения стемминга представлена на рисунке 9.

	Count		TF		TF-IDF	
	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
0	('the', 9066)	('image', 666)	('the', 351.8833889205307)	('know', 42.06416144310403)	('the', 132.73987348944183)	('know', 21.435150574064025)
1	('to', 5360)	('jpeg', 526)	('to', 215.21048224463186)	('like', 36.822127481402006)	('to', 85.24139589140572)	('like', 18.950713675342715)
2	('of', 4137)	('use', 516)	('of', 153.66848738794562)	('use', 36.792352685034714)	('of', 64.8113206497389)	('use', 18.420212557191185)
3	('and', 4073)	('edu', 468)	('and', 140.95239513687198)	('just', 32.40544450511485)	('and', 59.482599607653924)	('thanks', 17.098531494337998)
4	('is', 3074)	('graphics', 462)	('is', 120.607868681005)	('don', 30.469003399281732)	('is', 54.52831275498656)	('does', 16.98361584501106)
5	('in', 2610)	('like', 408)	('it', 110.88904071825)	('does', 29.885181249520144)	('it', 53.75496178917013)	('just', 16.684961307647363)
6	('it', 2402)	('file', 389)	('in', 104.33536769410595)	('thanks', 27.29259689277586)	('that', 47.770429081592496)	('don', 16.428887334276713)
7	('for', 2362)	('don', 378)	('that', 98.57686193779719)	('think', 24.82104074312471)	('in', 47.06683267580104)	('think', 14.557804515797576)
8	('that', 2228)	('data', 368)	('for', 92.40323586413749)	('used', 21.459669060506144)	('you', 45.42774190604445)	('graphics', 14.252209873941238)
9	('you', 2086)	('know', 355)	('you', 81.3739413442512)	('need', 20.686196868337543)	('for', 43.184971775965785)	('program', 13.64987318610155)
10	('be', 1535)	('just', 339)	('be', 65.317483332802)	('graphics', 20.679160827517396)	('be', 35.43111523777322)	('government', 12.964530900368489)
11	('this', 1472)	('bit', 337)	('on', 62.06126483104049)	('time', 20.129142247758516)	('this', 34.08603226381701)	('chip', 12.860296724296857)
12	('on', 1462)	('available', 325)	('this', 61.59725664266683)	('program', 19.584807797820034)	('on', 32.90027716297289)	('used', 12.439336643011975)
13	('or', 1295)	('software', 324)	('have', 57.61931147617616)	('people', 19.102579785729354)	('have', 32.025491313098804)	('people', 12.333813168768687)
14	('with', 1258)	('images', 307)	('or', 51.777520899187046)	('chip', 18.54157887888643)	('if', 29.12956299199104)	('need', 12.240898928564446)
15	('have', 1215)	('program', 298)	('if', 50.32542218561264)	('edu', 18.28349107306669)	('or', 29.011357783500337)	('bit', 12.015183440146776)
16	('are', 1186)	('does', 291)	('can', 49.21418012871439)	('ve', 18.270863639958304)	('can', 28.82297538311695)	('edu', 11.906194792288247)
17	('if', 1154)	('time', 282)	('with', 48.18761054093589)	('government', 18.089022419581)	('are', 27.38799920531715)	('ve', 11.870735103204861)
18	('can', 1101)	('used', 272)	('are', 45.11043625995436)	('good', 17.967957804096248)	('with', 27.03257285835745)	('time', 11.745018935759806)
19	('as', 1026)	('ftp', 271)	('not', 43.53696710250099)	('bit', 17.440629791897027)	('not', 26.9403506133788)	('key', 11.670618169470233)

Рисунок 8 – Результат

	Count		TF		TF-IDF	
	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами	Без стоп-слов	С стоп-словами
0	(the, 9063)	(thi, 1472)	(the, 344.5586615001451)	(thi, 97.49594840488756)	(the, 133.4538752492904)	(thi, 97.49594840488756)
1	(to, 5360)	(use, 1097)	(to, 211.00830323462205)	(use, 71.55533662918268)	(to, 85.99632390188842)	(use, 71.55533662918268)
2	(of, 4137)	(imag, 998)	(of, 150.4948643044924)	(ani, 44.25521347634618)	(of, 65.18554426844001)	(ani, 44.25521347634618)
3	(and, 4073)	(file, 615)	(and, 137.86770734884337)	(know, 43.06179116896538)	(and, 59.63534071325143)	(know, 43.06179116896538)
4	(is, 3139)	(jpeg, 531)	(is, 121.90454220930677)	(wa, 39.53046546637826)	(it, 55.94816487314925)	(wa, 39.53046546637826)
5	(in, 2612)	(wa, 510)	(it, 114.00279976905662)	(like, 36.7672996472524)	(is, 55.94487331494873)	(like, 36.7672996472524)
6	(it, 2562)	(ani, 505)	(in, 102.13673946564619)	(ha, 34.4237897180173)	(that, 48.31727363290116)	(ha, 34.4237897180173)
7	(for, 2362)	(program, 497)	(that, 96.8675995748726)	(doe, 34.19333412527219)	(in, 47.30170083345097)	(doe, 34.19333412527219)
8	(that, 2237)	(ha, 479)	(for, 90.52244749715135)	(just, 29.513176624612083)	(you, 45.68450447600251)	(just, 29.513176624612083)
9	(you, 2086)	(edu, 468)	(you, 79.67798552554372)	(thank, 28.549402258522097)	(for, 43.47570804261264)	(thank, 28.549402258522097)
10	(be, 1647)	(like, 457)	(be, 68.85460938269497)	(anyon, 28.298548977191338)	(be, 37.344622742852174)	(anyon, 28.298548977191338)
11	(thi, 1472)	(bit, 451)	(on, 61.11286152167876)	(work, 26.494787158165767)	(thi, 34.3491285954964)	(work, 26.494787158165767)
12	(on, 1469)	(format, 411)	(have, 60.90208541002496)	(think, 24.640467411932118)	(have, 34.04975283966364)	(think, 24.640467411932118)
13	(have, 1298)	(know, 401)	(thi, 60.34198459030236)	(need, 24.56412549445321)	(on, 33.209071780256735)	(need, 24.56412549445321)
14	(or, 1295)	(doe, 386)	(or, 50.6519636067793)	(program, 24.56261574830257)	(if, 29.438597446898267)	(program, 24.56261574830257)
15	(with, 1260)	(data, 369)	(if, 49.388084620385364)	(look, 24.441912389427163)	(or, 29.11817727731375)	(look, 24.441912389427163)
16	(are, 1212)	(onli, 344)	(with, 47.226534170832274)	(make, 24.08146082002033)	(can, 28.41866932660334)	(make, 24.08146082002033)
17	(if, 1154)	(work, 344)	(can, 46.08463643662672)	(key, 23.553337023165067)	(are, 28.194428691654128)	(key, 23.553337023165067)
18	(use, 1097)	(make, 341)	(are, 45.258192235058026)	(pleas, 23.374159220189213)	(do, 27.903939642519983)	(pleas, 23.374159220189213)
19	(not, 1077)	(just, 339)	(do, 44.57070053438678)	(onli, 22.125484501080784)	(not, 27.900212314318484)	(onli, 22.125484501080784)

Рисунок 9 – Результат

Проведем классификацию методом случайного леса с параметром «n_estimators» = 10.

```
Prediction and test:
Prediction:       [1 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 0 0 0]
Test:             [1 1 0 1 1 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 0 0]
Confusion matrix:
[13  0]
[3 9]
Accuracy score:  0.88
Classification Report
              precision    recall  f1-score   support

           0       0.81      1.00      0.90        13
           1       1.00      0.75      0.86        12

    accuracy                           0.88        25
   macro avg       0.91      0.88      0.88        25
weighted avg       0.90      0.88      0.88        25
ROC AUC
0.875
```

```
from sklearn.metrics import classification_report
from sklearn.feature_extraction.text import CountVectorizer,
TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline

pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf', MultinomialNB()),
])

parameters = {
    'vect__max_features': (500, 1000, 2500, 5000, 10000, None),
    'vect__stop_words': ('english', None),
    'tfidf__use_idf': (True, False),
}

grid_search = GridSearchCV(pipeline, parameters, n_jobs=-1, verbose=1)

grid_search.fit(train_bunch.data, train_bunch.target)

print("Best score: %0.3f" % grid_search.best_score_)
print("Best parameters set:")
grid_search.best_params_
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
Best score: 0.893
Best parameters set:
{'tfidf__use_idf': True,
 'vect__max_features': 5000,
 'vect__stop_words': 'english'}
```

Вывод

В ходе выполнения данной лабораторной работы я получил базовые навыки работы с языком python и набором функций для анализа данных.