Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине «Прикладные интеллектуальные системы и экспертные системы»

Классификация текстовых данных

Студент Коровайцев А.А.

Группа М-ИАП-23-1

Руководитель Кургасов В.В.

Доцент

Липецк 2023 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 данных.

Вариант №7

Классы 3, 7, 13 ('comp.os.ms-windows.misc', 'misc.forsale', 'sci.electronics')

Методы RF, LR, SVM

Случайный лес (RF):

• количество деревьев решений,

• критерий (параметр criterion: ‘gini’, ‘entropy’),

• глубина дерева (параметр max_depth от 1 до 5 с шагом 1, далее до 100 с шагом 20).

Логистическая регрессия (LR):

• метод нахождения экстремума (параметр solver: ‘newton-cg’, ‘lbfgs’, ‘sag’, ‘liblinear’),

• регуляризация (параметр penalty: ‘L1’, ‘L2’)

Обратить внимание, что разные виды регуляризации работают с разными методами нахождения экстремума.

Метод опорных векторов (SVM):

• функция потерь (параметр loss: ‘hinge’, ‘squared_hinge’),

• регуляризация (параметр penalty: ‘L1’, ‘L2’)

Обратить внимание, что разные виды регуляризации работают с разными функциями потерь.

Ход работы

Загрузим обучающую и тестовую выборку в соответствии с вариантом. Код для загрузки данных представлен на рисунке 1.

```
categories = ['comp.os.ms-windows.misc', 'misc.forsale', 'sci.electronics']
remove = ('headers', 'footers', 'quotes')

twenty_train_full = fetch_20newsgroups(subset='train', categories=categories, shuffle=True, random_state=42, remove=remove)
twenty_test_full = fetch_20newsgroups(subset='test', categories=categories, shuffle=True, random_state=42, remove=remove)
```

Рисунок 1 – Код для загрузки данных из лабораторной работы №2

Зададим параметры, которые будем варьировать, чтобы найти наиболее оптимальные. Параметры для каждого из методов представлены на рисунке 2.

```
stop_words = [None, 'english']
max_features_values = [100, 500, 1000, 5000, 10000]
use_idf = [True, False]
```

```
rf_first = range(1, 5, 1)
rf_second = range(5, 100, 20)

rf_tree_max_depth = [*rf_first, *rf_second]
```

```
parameters_rf = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
    'clf__n_estimators': range(1, 10, 1),
    'clf__criterion': ('gini', 'entropy'),
    'clf__max_depth': rf_tree_max_depth,
}

parameters_lr = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
    'clf__solver': ['newton-cg', 'lbfgs', 'sag', 'liblinear'],
    'clf__penalty': ['l2']
}

parameters_lr_l1 = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
    'clf__solver': ['liblinear'],  # Используем только 'liblinear' для l1
    'clf__penalty': ['l1'],
}

parameters_svm = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
}
```

Рисунок 2 – Параметры для нахождения оптимальных значений классификации

Проведем классификацию методами RF, LR (и LR_L1) и SVM. Код всех методов представлен в приложении А.

После проведения обучения моделей на обучающем наборе данных рассчитаем характеристики качества классификации по каждому методу.

Качество модели случайного леса для данных без применения стемминга и оптимальные для неё параметры представлены на рисунке 3.

```
Случайный лес (RF) без стемминга

                         precision    recall  f1-score   support

comp.os.ms-windows.misc       0.78      0.77      0.77       394
           misc.forsale       0.84      0.81      0.82       390
        sci.electronics       0.68      0.71      0.69       393

               accuracy                           0.76      1177
              macro avg       0.77      0.76      0.76      1177
           weighted avg       0.77      0.76      0.76      1177

{'clf__criterion': 'gini', 'clf__max_depth': 85, 'clf__n_estimators': 9, 'tfidf__use_idf': True, 'vect__max_feature
s': 5000, 'vect__stop_words': 'english'}
```

Рисунок 3 – Качество модели случайного леса для данных без применения стемминга и оптимальные для неё параметры

Качество модели случайного леса для данных с применением стемминга и оптимальные для неё параметры представлены на рисунке 4.

```
Случайный лес (RF) со стеммингом

                         precision    recall  f1-score   support

comp.os.ms-windows.misc       0.73      0.56      0.63       394
           misc.forsale       0.76      0.75      0.76       390
        sci.electronics       0.56      0.70      0.62       393

               accuracy                           0.67      1177
              macro avg       0.68      0.67      0.67      1177
           weighted avg       0.68      0.67      0.67      1177

{'clf__criterion': 'gini', 'clf__max_depth': 45, 'clf__n_estimators': 9, 'tfidf__use_idf': False, 'vect__max_featur
es': 500, 'vect__stop_words': 'english'}
```

Рисунок 4 – Качество модели случайного леса для данных с применением стемминга и оптимальные для неё параметры

Качество модели логистической регрессии для данных без применения стемминга и оптимальные для неё параметры представлены на рисунке 5.

```
Логистическая регрессия (LR) без стемминга

                         precision    recall  f1-score   support

comp.os.ms-windows.misc       0.89      0.82      0.85       394
           misc.forsale       0.89      0.86      0.88       390
        sci.electronics       0.78      0.86      0.82       393

               accuracy                           0.85      1177
              macro avg       0.85      0.85      0.85      1177
           weighted avg       0.85      0.85      0.85      1177

{'clf__penalty': 'l2', 'clf__solver': 'newton-cg', 'tfidf__use_idf': True, 'vect__max_features': 5000, 'vect__stop_
words': 'english'}
```

Рисунок 5 – Качество модели логистической регрессии для данных без применения стемминга и оптимальные для неё параметры

Качество модели логистической регрессии L1 для данных без применения стемминга и оптимальные для неё параметры представлены на рисунке 6.

```
Логистическая регрессия_l1 (LR) без стемминга

                         precision    recall  f1-score   support

comp.os.ms-windows.misc       0.82      0.74      0.78       394
           misc.forsale       0.91      0.82      0.86       390
        sci.electronics       0.70      0.83      0.76       393

               accuracy                           0.80      1177
              macro avg       0.81      0.80      0.80      1177
           weighted avg       0.81      0.80      0.80      1177

{'clf__penalty': 'l1', 'clf__solver': 'liblinear', 'tfidf__use_idf': True, 'vect__max_features': 1000, 'vect__stop_
words': None}
```

Рисунок 6 – Качество модели логистической регрессии L1 для данных без применения стемминга и оптимальные для неё параметры

Качество модели логистической регрессии для данных с применением стемминга и оптимальные для неё параметры представлены на рисунке 7.

```
Логистическая регрессия (LR) со стеммингом

                         precision    recall  f1-score   support

comp.os.ms-windows.misc       0.87      0.77      0.81       394
           misc.forsale       0.82      0.87      0.84       390
        sci.electronics       0.76      0.80      0.78       393

               accuracy                           0.81      1177
              macro avg       0.81      0.81      0.81      1177
           weighted avg       0.81      0.81      0.81      1177

{'clf__penalty': 'l2', 'clf__solver': 'newton-cg', 'tfidf__use_idf': True, 'vect__max_features': 10000, 'vect__stop
_words': 'english'}
```

Рисунок 7 – Качество модели логистической регрессии для данных с применением стемминга и оптимальные для неё параметры

Качество модели логистической регрессии L1 для данных с применением стемминга и оптимальные для неё параметры представлены на рисунке 8.

```
Логистическая регрессия_l1 (LR) со стеммингом

                      precision    recall  f1-score   support

comp.os.ms-windows.misc       0.75      0.63      0.69       394
           misc.forsale       0.87      0.74      0.80       390
         sci.electronics      0.59      0.76      0.66       393

               accuracy                           0.71      1177
              macro avg       0.73      0.71      0.72      1177
           weighted avg       0.73      0.71      0.72      1177

{'clf__penalty': 'l1', 'clf__solver': 'liblinear', 'tfidf__use_idf': True, 'vect__max_features': 500, 'vect__stop_w
ords': 'english'}
```

Рисунок 8 – Качество модели логистической регрессии L1 для данных с применением стемминга и оптимальные для неё параметры

Качество модели метода опорных векторов для данных без применения стемминга и оптимальные для неё параметры представлены на рисунке 9.

```
Метод опорных векторов (SVM) без стемминга

                      precision    recall  f1-score   support

comp.os.ms-windows.misc       0.87      0.81      0.84       394
           misc.forsale       0.91      0.87      0.89       390
         sci.electronics      0.78      0.87      0.82       393

               accuracy                           0.85      1177
              macro avg       0.85      0.85      0.85      1177
           weighted avg       0.85      0.85      0.85      1177

{'tfidf__use_idf': True, 'vect__max_features': 10000, 'vect__stop_words': 'english'}
```

Рисунок 9 – Качество модели метода опорных векторов для данных без применения стемминга и оптимальные для неё параметры

Качество модели метода опорных векторов для данных с применением стемминга и оптимальные для неё параметры представлены на рисунке 10.

```
Метод опорных векторов (SVM) со стеммингом

                      precision    recall  f1-score   support

comp.os.ms-windows.misc       0.82      0.79      0.80       394
           misc.forsale       0.82      0.85      0.83       390
         sci.electronics      0.76      0.76      0.76       393

               accuracy                           0.80      1177
              macro avg       0.80      0.80      0.80      1177
           weighted avg       0.80      0.80      0.80      1177

{'tfidf__use_idf': True, 'vect__max_features': 10000, 'vect__stop_words': 'english'}
```

Рисунок 10 – Качество модели метода опорных векторов для данных с применением стемминга и оптимальные для неё параметры

Вывод

В результате выполнения данной лабораторной работы я получил практические навыки решения задачи классификации текстовых данных в среде Jupiter Notebook.

Также научился проводить предварительную обработку текстовых данных, настраивать параметры методов классификации и обучать модели, оценивать точность полученных моделей.

Мною были применены следующие методы: случайного леса (RF), логистической регрессии (LR) и метод опорных векторов (SVM).

Наилучшей точностью классификации для данного набора данных обладают модели логистической регрессии и метода опорных векторов без применения стемминга. Их точность составляет 85%. Параметры для данных моделей представлены соответственно на рисунках 5 и 9.

Приложение А

Исходный код

```
#!/usr/bin/env python
# coding: utf-8

# # Лабораторная работа №3
# ### Выгрузка данных из ЛР №2 (вариант №7) ('comp.os.ms-windows.misc',
'misc.forsale', 'sci.electronics')

# In[1]:


import warnings
from sklearn.datasets import fetch_20newsgroups
warnings.simplefilter(action='ignore', category=FutureWarning)


# In[2]:


categories = ['comp.os.ms-windows.misc', 'misc.forsale', 'sci.electronics']
remove = ('headers', 'footers', 'quotes')

twenty_train_full = fetch_20newsgroups(subset='train', categories=categories,
shuffle=True, random_state=42, remove=remove)
twenty_test_full = fetch_20newsgroups(subset='test', categories=categories,
shuffle=True, random_state=42, remove=remove)


# ### Применение стемминга

# In[3]:


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

nltk.download('punkt')


# In[4]:


def stemming(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


# In[5]:


stem_train = stemming(twenty_train_full.data)
stem_test = stemming(twenty_test_full.data)
```

```
# ### Задание
# ### Вариант №7
# ### Методы: [RF, LR, SVM]

# In[6]:


from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC


# In[7]:


stop_words = [None, 'english']
max_features_values = [100, 500, 1000, 5000, 10000]
use_idf = [True, False]


# In[8]:


rf_first = range(1, 5, 1)
rf_second = range(5, 100, 20)

rf_tree_max_depth = [*rf_first, *rf_second]


# In[9]:


parameters_rf = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
    'clf__n_estimators': range(1, 10, 1),
    'clf__criterion': ('gini', 'entropy'),
    'clf__max_depth': rf_tree_max_depth,
}

parameters_lr = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
    'clf__solver': ['newton-cg', 'lbfgs', 'sag', 'liblinear'],
    'clf__penalty': ['l2']
}

parameters_lr_l1 = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
    'clf__solver': ['liblinear'],  # Используем только 'liblinear' для l1
    'clf__penalty': ['l1'],
}

parameters_svm = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
}
```

```
# In[10]:


from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer


# ### Случайный лес (RF)

# #### Без использования стемминга

# In[11]:


text_clf_rf = Pipeline([('vect', CountVectorizer()),
                        ('tfidf', TfidfTransformer()),
                        ('clf', RandomForestClassifier())])
gscv_rf = GridSearchCV(text_clf_rf, param_grid=parameters_rf, n_jobs=-1)
gscv_rf.fit(twenty_train_full.data, twenty_train_full.target)


# #### С использованием стема

# In[12]:


text_clf_rf_stem = Pipeline([('vect', CountVectorizer()),
                             ('tfidf', TfidfTransformer()),
                             ('clf', RandomForestClassifier())])
gscv_rf_stem = GridSearchCV(text_clf_rf_stem, param_grid=parameters_rf,
n_jobs=-1)
gscv_rf_stem.fit(stem_train, twenty_train_full.target)


# ### Логистическая регрессия (LR)

# #### Без использования стемминга

# In[13]:


text_clf_lr = Pipeline([('vect', CountVectorizer()),
                        ('tfidf', TfidfTransformer()),
                        ('clf', LogisticRegression())])
gscv_lr = GridSearchCV(text_clf_lr, param_grid=parameters_lr, n_jobs=-1)
gscv_lr.fit(twenty_train_full.data, twenty_train_full.target)

text_clf_lr_l1 = Pipeline([('vect', CountVectorizer()),
                        ('tfidf', TfidfTransformer()),
                        ('clf', LogisticRegression())])
gscv_lr_l1 = GridSearchCV(text_clf_lr_l1, param_grid=parameters_lr_l1,
n_jobs=-1)
gscv_lr_l1.fit(twenty_train_full.data, twenty_train_full.target)


# #### С использованием стемминга

# In[14]:


text_clf_lr_stem = Pipeline([('vect', CountVectorizer()),
                             ('tfidf', TfidfTransformer()),
                             ('clf', LogisticRegression())])
```

```
gscv_lr_stem = GridSearchCV(text_clf_lr_stem, param_grid=parameters_lr,
n_jobs=-1)
gscv_lr_stem.fit(stem_train, twenty_train_full.target)

text_clf_lr_l1_stem = Pipeline([('vect', CountVectorizer()),
                                ('tfidf', TfidfTransformer()),
                                ('clf', LogisticRegression())])
gscv_lr_l1_stem = GridSearchCV(text_clf_lr_l1_stem,
param_grid=parameters_lr_l1, n_jobs=-1)
gscv_lr_l1_stem.fit(stem_train, twenty_train_full.target)


# ### Метод опорных векторов (SVM)

# #### Без использования стемминга

# In[15]:


text_clf_svm = Pipeline([('vect', CountVectorizer()),
                         ('tfidf', TfidfTransformer()),
                         ('clf', SVC())])
gscv_svm = GridSearchCV(text_clf_svm, param_grid=parameters_svm, n_jobs=-1)
gscv_svm.fit(twenty_train_full.data, twenty_train_full.target)


# #### С использованием стемминга

# In[16]:


text_clf_svm_stem = Pipeline([('vect', CountVectorizer()),
                              ('tfidf', TfidfTransformer()),
                              ('clf', SVC(kernel='linear', C=1.0))])
gscv_svm_stem = GridSearchCV(text_clf_svm_stem, param_grid=parameters_svm,
n_jobs=-1)
gscv_svm_stem.fit(stem_train, twenty_train_full.target)


# ### Вывод полученных результатов анализа

# In[17]:


from sklearn.metrics import classification_report


# In[18]:


predicted_rf = gscv_rf.predict(twenty_test_full.data)
print('Случайный лес (RF) без стемминга\n')
print(classification_report(twenty_test_full.target, predicted_rf,
target_names=categories))
print(gscv_rf.best_params_)


# In[19]:


predicted_rf_stem = gscv_rf_stem.predict(twenty_test_full.data)
print('Случайный лес (RF) со стеммингом\n')
print(classification_report(twenty_test_full.target, predicted_rf_stem,
target_names=categories))
```

```
print(gscv_rf_stem.best_params_)


# In[20]:


predicted_lr = gscv_lr.predict(twenty_test_full.data)
print('Логистическая регрессия (LR) без стемминга\n')
print(classification_report(twenty_test_full.target, predicted_lr,
target_names=categories))
print(gscv_lr.best_params_)

predicted_lr_l1 = gscv_lr_l1.predict(twenty_test_full.data)
print('Логистическая регрессия_l1 (LR) без стемминга\n')
print(classification_report(twenty_test_full.target, predicted_lr_l1,
target_names=categories))
print(gscv_lr_l1.best_params_)


# In[21]:


predicted_lr_stem = gscv_lr_stem.predict(twenty_test_full.data)
print('Логистическая регрессия (LR) со стеммингом\n')
print(classification_report(twenty_test_full.target, predicted_lr_stem,
target_names=categories))
print(gscv_lr_stem.best_params_)

predicted_lr_l1_stem = gscv_lr_l1_stem.predict(twenty_test_full.data)
print('Логистическая регрессия_l1 (LR) со стеммингом\n')
print(classification_report(twenty_test_full.target, predicted_lr_l1_stem,
target_names=categories))
print(gscv_lr_l1_stem.best_params_)


# In[22]:


predicted_svm = gscv_svm.predict(twenty_test_full.data)
print('Метод опорных векторов (SVM) без стемминга\n')
print(classification_report(twenty_test_full.target, predicted_svm,
target_names=categories))
print(gscv_svm.best_params_)


# In[23]:


predicted_svm_stem = gscv_svm_stem.predict(twenty_test_full.data)
print('Метод опорных векторов (SVM) со стеммингом\n')
print(classification_report(twenty_test_full.target, predicted_svm_stem,
target_names=categories))
print(gscv_svm_stem.best_params_)


# # Сравнительная таблица

# In[24]:


import pandas as pd


# In[25]:
```

```
writer = pd.ExcelWriter('result.xlsx', engine='openpyxl')

# Случайный лес (RF) без стемминга
df1 = pd.DataFrame(classification_report(predicted_rf,
twenty_test_full.target, output_dict=True))

# Случайный лес (RF) со стеммингом
df2 = pd.DataFrame(classification_report(predicted_rf_stem,
twenty_test_full.target, output_dict=True))

# Логистическая регрессия (LR) без стемминга
df3 = pd.DataFrame(classification_report(predicted_lr,
twenty_test_full.target, output_dict=True))

# Логистическая регрессия_l1 (LR) без стемминга
df4 = pd.DataFrame(classification_report(predicted_lr_l1,
twenty_test_full.target, output_dict=True))

# Логистическая регрессия (LR) со стеммингом
df5 = pd.DataFrame(classification_report(predicted_lr_stem,
twenty_test_full.target, output_dict=True))

# Логистическая регрессия_l1 (LR) со стеммингом
df6 = pd.DataFrame(classification_report(predicted_lr_l1_stem,
twenty_test_full.target, output_dict=True))

# Метод опорных векторов (SVM) без стемминга
df7 = pd.DataFrame(classification_report(predicted_svm,
twenty_test_full.target, output_dict=True))

# Метод опорных векторов (SVM) со стеммингом
df8 = pd.DataFrame(classification_report(predicted_svm_stem,
twenty_test_full.target, output_dict=True))

df1.to_excel(writer, sheet_name='RF без стемминга')
df2.to_excel(writer, sheet_name='RF со стеммингом')

df3.to_excel(writer, sheet_name='LR без стемминга')
df4.to_excel(writer, sheet_name='LR_l1 без стемминга')

df5.to_excel(writer, sheet_name='LR со стеммингом')
df6.to_excel(writer, sheet_name='LR_l1 со стеммингом')

df7.to_excel(writer, sheet_name='SVM без стемминга')
df8.to_excel(writer, sheet_name='SVM со стеммингом')

writer.close()
```