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 Цыганов Н.А.

Группы М-ИАП-23

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

Задание:

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.

Ход работы

Вариант по журналу 18, вариантов 12, следовательно: вариант 6 представлен на рисунке 1.

6	DT, KNN, LR

Рисунок 1 - Вариант для выполнения

На рисунке 2 изображен импорт библиотек для загрузки данных.

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

Рисунок 2 - Импорт необходимых библиотек

```
categories = ['alt.atheism', 'sci.space', 'soc.religion.christian']
remove = ('headers', 'footers', 'quotes')

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

Рисунок 3 - Загрузка данных

Код методов анализа на рисунке 4.

```
##DT, KNN, LR
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
```

```
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]
```

```
dt_parameters = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
    'clf__criterion': ('gini', 'entropy'),
    'clf__max_depth': [*range(1, 6, 1), *range(25, 101, 20)],
}

knn_parameters = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
    'clf__n_neighbors': [3, 5, 7, 9],
    'clf__metric': ['euclidean', 'manhattan'],
}

lr_parameters = {
    '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': ['l1', 'l2'],
}
```

Рисунок 4 - – Параметры для нахождения оптимальных значений классификации

Проведем классификацию методами по варианту и после проведения обучения моделей на обучающем наборе данных рассчитаем характеристики качества классификации по каждому методу. Качество модели дерево

решений (DT) для данных без применения стемминга и оптимальные для неё параметры представлены на рисунке 5.

Дерево решений (DT):

• критерий (параметр criterion: ‘gini’, ‘entropy’), • глубина дерева (параметр max_depth от 1 до 5 с шагом 1, далее до 100 с шагом 20).

Без использования стемминга

```
text_clf_dt = Pipeline([('vect', CountVectorizer()),
                        ('tfidf', TfidfTransformer()),
                        ('clf', DecisionTreeClassifier())])
gscv_dt = GridSearchCV(text_clf_dt, param_grid=dt_parameters, n_jobs=-1)
gscv_dt.fit(twenty_train_full.data, twenty_train_full.target)
```

```
GridSearchCV(estimator=Pipeline(steps=[('vect', CountVectorizer()),
                                       ('tfidf', TfidfTransformer()),
                                       ('clf', DecisionTreeClassifier())]),
             n_jobs=-1,
             param_grid={'clf__criterion': ('gini', 'entropy'),
                         'clf__max_depth': [1, 2, 3, 4, 5, 25, 45, 65, 85],
                         'tfidf__use_idf': [True, False],
                         'vect__max_features': [100, 500, 1000, 5000, 10000],
                         'vect__stop_words': [None, 'english']})
```

estimator: Pipeline

```
Pipeline(steps=[('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),
                ('clf', DecisionTreeClassifier())])
```

CountVectorizer

```
CountVectorizer()
```

TfidfTransformer

```
TfidfTransformer()
```

DecisionTreeClassifier

```
DecisionTreeClassifier()
```

```
Дерево решений (DT)

                        precision    recall  f1-score   support

           alt.atheism       0.57      0.46      0.51       319
             sci.space       0.70      0.85      0.77       394
soc.religion.christian       0.71      0.67      0.69       398

              accuracy                           0.67      1111
             macro avg       0.66      0.66      0.65      1111
          weighted avg       0.67      0.67      0.66      1111

{'clf__criterion': 'gini', 'clf__max_depth': 65, 'tfidf__use_idf': False, 'vect__max_features': 5000, 'vect__stop_wor
ds': 'english'}
```

Рисунок 5 - Дерево решений (DT) без стемминга

Дерево решений (DT) со стеммингом представлено на рисунке 6.

С использованием стемминга

```
2]: text_clf_dt_stem = Pipeline([('vect', CountVectorizer()),
                                 ('tfidf', TfidfTransformer()),
                                 ('clf', DecisionTreeClassifier())])
gscv_dt_stem = GridSearchCV(text_clf_dt_stem, param_grid=dt_parameters, n_jobs=-1)
gscv_dt_stem.fit(stem_train, twenty_train_full.target)
```

```
2]: GridSearchCV
GridSearchCV(estimator=Pipeline(steps=[('vect', CountVectorizer()),
                                       ('tfidf', TfidfTransformer()),
                                       ('clf', DecisionTreeClassifier())]),
             n_jobs=-1,
             param_grid={'clf__criterion': ('gini', 'entropy'),
                         'clf__max_depth': [1, 2, 3, 4, 5, 25, 45, 65, 85],
                         'tfidf__use_idf': [True, False],
                         'vect__max_features': [100, 500, 1000, 5000, 10000],
                         'vect__stop_words': [None, 'english']})
estimator: Pipeline
Pipeline(steps=[('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),
                ('clf', DecisionTreeClassifier())])
CountVectorizer
CountVectorizer()
TfidfTransformer
TfidfTransformer()
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
Дерево решений (DT) со стеммингом

                        precision    recall  f1-score   support

           alt.atheism       0.58      0.34      0.43       319
             sci.space       0.59      0.91      0.72       394
soc.religion.christian       0.71      0.58      0.64       398

              accuracy                           0.62      1111
             macro avg       0.63      0.61      0.59      1111
          weighted avg       0.63      0.62      0.60      1111

{'clf__criterion': 'gini', 'clf__max_depth': 25, 'tfidf__use_idf': True, 'vect__max_features': 500, 'vect__stop_word
s': 'english'}
```

Рисунок 6 - Дерево решений (DT) со стеммингом представлено на рисунке 6.

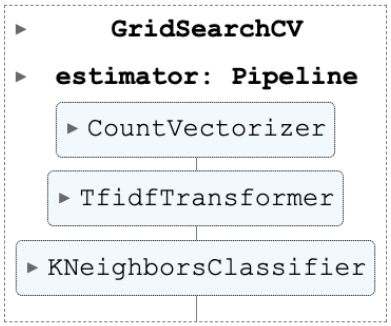
Качество модели K-ближайших соседей (KNN) для данных без применения стемминга и оптимальные для неё параметры представлены на рисунке 7.

K-ближайших соседей (KNN):

• количество ближайших соседей, • метрика (евклидова, городских кварталов)

Без использования стэмминга

```
text_clf_knn = Pipeline([('vect', CountVectorizer()),
                         ('tfidf', TfidfTransformer()),
                         ('clf', KNeighborsClassifier())])
gscv_knn = GridSearchCV(text_clf_knn, param_grid=knn_parameters, n_jobs=-1)
gscv_knn.fit(twenty_train_full.data, twenty_train_full.target)
```




```
K-ближайших соседей (KNN)

                        precision    recall  f1-score   support

           alt.atheism       0.40      0.59      0.48       319
             sci.space       0.75      0.56      0.64       394
soc.religion.christian       0.66      0.57      0.61       398

              accuracy                           0.57      1111
             macro avg       0.60      0.57      0.57      1111
          weighted avg       0.61      0.57      0.58      1111

{'clf__metric': 'euclidean', 'clf__n_neighbors': 3, 'tfidf__use_idf': True, 'vect__max_features': 100, 'vect__stop_wo
rds': 'english'}
```

Рисунок 7 - K-ближайших соседей (KNN) без стэмминга

Теперь K-ближайших соседей (KNN) со стэммингом, это показано на рисунке 8.

С использованием стэмминга

```
text_clf_knn_stem = Pipeline([('vect', CountVectorizer()),
                              ('tfidf', TfidfTransformer()),
                              ('clf', KNeighborsClassifier())])
gscv_knn_stem = GridSearchCV(text_clf_knn_stem, param_grid=knn_parameters, n_jobs=-1)
gscv_knn_stem.fit(stem_train, twenty_train_full.target)
```


```
print('\nK-ближайших соседей (KNN) со стеммингом\n')
print(classification_report(twenty_test_full.target, predicted_knn_stem, target_names=categories))
print(gscv_knn_stem.best_params_)
```

```
K-ближайших соседей (KNN) со стеммингом

                        precision    recall  f1-score   support

           alt.atheism       0.41      0.58      0.48       319
             sci.space       0.69      0.65      0.67       394
soc.religion.christian       0.72      0.52      0.60       398

              accuracy                           0.58      1111
             macro avg       0.60      0.58      0.58      1111
          weighted avg       0.62      0.58      0.59      1111

{'clf__metric': 'euclidean', 'clf__n_neighbors': 3, 'tfidf__use_idf': True, 'vect__max_features': 100, 'vect__stop_wo
rds': 'english'}
```

Рисунок 8 - K-ближайших соседей (KNN) со стэммингом

Качество модели Логистическая регрессия (LR) для данных без применения стемминга и оптимальные для неё параметры представлены на рисунке 9.

Логистическая регрессия (LR):

• метод нахождения экстремума (параметр solver: ‘newton-cg’, ‘lbfgs’, ‘sag’, ‘liblinear’), • регуляризация (параметр penalty: ‘L1’, ‘L2’)

Без использования стемминга

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

```
5]: GridSearchCV
GridSearchCV(estimator=Pipeline(steps=[('vect', CountVectorizer()),
                                       ('tfidf', TfidfTransformer()),
                                       ('clf', LogisticRegression())]),
             n_jobs=-1,
             param_grid={'clf__penalty': ['l1', 'l2'],
                         'clf__solver': ['newton-cg', 'lbfgs', 'sag',
                                         'liblinear'],
                         'tfidf__use_idf': [True, False],
                         'vect__max_features': [100, 500, 1000, 5000, 10000],
                         'vect__stop_words': [None, 'english']})
estimator: Pipeline
Pipeline(steps=[('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),
                ('clf', LogisticRegression())])
CountVectorizer
```

```
: print('\nЛогистическая регрессия (LR)\n')
print(classification_report(twenty_test_full.target, predicted_lr, target_names=categories))
print(gscv_lr.best_params_)
```

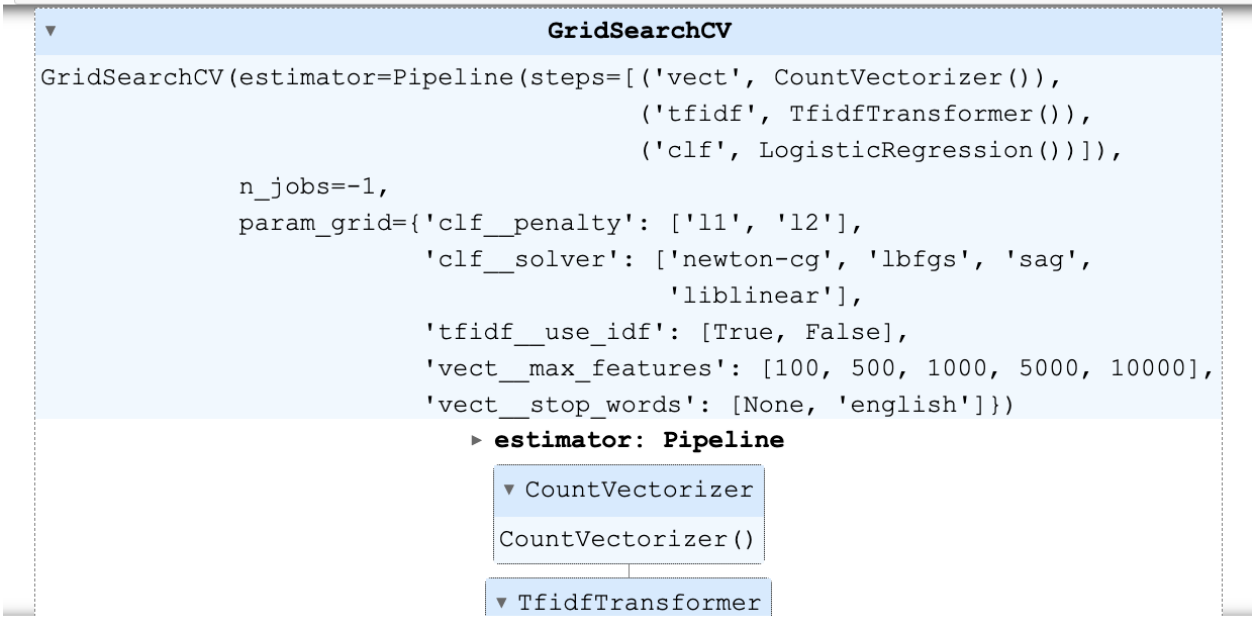
```
Логистическая регрессия (LR)

                        precision    recall  f1-score   support

           alt.atheism       0.80      0.52      0.63       319
             sci.space       0.79      0.95      0.86       394
soc.religion.christian       0.79      0.85      0.82       398

              accuracy                           0.79      1111
             macro avg       0.79      0.77      0.77      1111
          weighted avg       0.79      0.79      0.78      1111

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

Рисунок 9 - Логистическая регрессия (LR) без стэмминга

Логистическая регрессия (LR) без стэмминга показана на рисунке 10.

С использованием стемминга

```
text_clf_lr_stem = Pipeline([('vect', CountVectorizer()),
                             ('tfidf', TfidfTransformer()),
                             ('clf', LogisticRegression())])
gscv_lr_stem = GridSearchCV(text_clf_lr_stem, param_grid=lr_parameters, n_jobs=-1
gscv_lr_stem.fit(stem_train, twenty_train_full.target)
```




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

```
Логистическая регрессия (LR) со стеммингом

                        precision    recall  f1-score   support

           alt.atheism       0.73      0.56      0.64       319
             sci.space       0.70      0.96      0.81       394
soc.religion.christian       0.85      0.71      0.77       398

              accuracy                           0.76      1111
             macro avg       0.76      0.74      0.74      1111
          weighted avg       0.77      0.76      0.75      1111

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

Рисунок 10 - Логистическая регрессия (LR) со стэммингом

Вывод

В результате выполнения работы получены практические навыки обработки текстовых данных в среде Jupiter Notebook. Проведена предварительная обработка текстовых данных и выявлены параметры обработки, позволяющие добиться наилучшей точности классификации.