

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

Курдюков И.Ю.

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 соответствии 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 10

10	DT, SVM, 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=remove)
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, SVM, LR

```
[3] from sklearn.tree import DecisionTreeClassifier
     from sklearn.neighbors import KNeighborsClassifier
     from sklearn.linear_model import LogisticRegression
     from sklearn.ensemble import RandomForestClassifier
```

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

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

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

```
[7] 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)],
}

parameters_svm_l1 = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf
}

parameters_svm_l2 = {
    'vect__max_features': max_features_values,
    'vect__stop_words': stop_words,
    'tfidf__use_idf': use_idf,
    'clf__loss': ['hinge', 'squared_hinge']
}

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': ['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
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_words': '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)
```



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_words': 'english'}
```

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

Качество модели метода опорных векторов L1 для данных без применения стемминга и оптимальные для неё параметры представлены на рисунке 7.

Метод опорных векторов (SVM) l1 без стемминга

	precision	recall	f1-score	support
comp.windows.x	0.97	0.86	0.91	395
rec.sport.baseball	0.76	0.91	0.83	397
rec.sport.hockey	0.92	0.83	0.88	399
accuracy			0.87	1191
macro avg	0.88	0.87	0.87	1191
weighted avg	0.88	0.87	0.87	1191

```
{'tfidf__use_idf': True, 'vect__max_features': 5000, 'vect__stop_words': 'english'}
```

Рисунок 7 – Качество модели метода опорных векторов L1 для данных без применения стемминга и оптимальные для неё параметры

Качество модели метода опорных векторов L1 для данных с применением стемминга и оптимальные для неё параметры представлены на рисунке 8.

Метод опорных векторов (SVM) l1 со стеммингом

	precision	recall	f1-score	support
comp.windows.x	0.96	0.81	0.88	395
rec.sport.baseball	0.69	0.83	0.76	397
rec.sport.hockey	0.82	0.78	0.80	399
accuracy			0.81	1191
macro avg	0.82	0.81	0.81	1191
weighted avg	0.82	0.81	0.81	1191

```
{'tfidf__use_idf': True, 'vect__max_features': 1000, 'vect__stop_words': 'english'}
```

Рисунок 8 – Качество модели метода опорных векторов L1 для данных с применением стемминга и оптимальные для неё параметры

Качество модели метода опорных векторов L2 для данных без применения стемминга и оптимальные для неё параметры представлены на

рисунке 9.

Метод опорных векторов (SVM) l2 без стемминга

	precision	recall	f1-score	support
comp.windows.x	0.98	0.94	0.96	395
rec.sport.baseball	0.86	0.91	0.88	397
rec.sport.hockey	0.91	0.89	0.90	399
accuracy			0.91	1191
macro avg	0.92	0.91	0.92	1191
weighted avg	0.92	0.91	0.91	1191

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

Рисунок 9 – Качество модели метода опорных векторов L2 для данных без применения стемминга и оптимальные для неё параметры

Качество модели метода опорных векторов L2 для данных с применением стемминга и оптимальные для неё параметры представлены на рисунке 10.

Метод опорных векторов (SVM) l2 со стеммингом

	precision	recall	f1-score	support
comp.windows.x	0.98	0.86	0.92	395
rec.sport.baseball	0.78	0.87	0.82	397
rec.sport.hockey	0.85	0.85	0.85	399
accuracy			0.86	1191
macro avg	0.87	0.86	0.86	1191
weighted avg	0.87	0.86	0.86	1191

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

Рисунок 10 – Качество модели метода опорных векторов L2 для данных с применением стемминга и оптимальные для неё параметры

Качество модели Логистическая регрессия (LR) для данных без применения стемминга и оптимальные для неё параметры представлены на рисунке 11.

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 11 - Логистическая регрессия (LR) без стэмминга

Логистическая регрессия (LR) без стэмминга показана на рисунке 12.

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)
```

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**

- ▼ CountVectorizer
 - CountVectorizer()
- ▼ TfidfTransformer

```
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_words': 'english'}
```

Рисунок 12 - Логистическая регрессия (LR) со стеммингом

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.