Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»


Рубежный контроль №2
по дисциплине
# «Методы машинного обучения»
на тему

**«Методы обработки текстов.»**

Выполнил:
студент группы ИУ5и-22М
Се Цзявэнь

Москва — 2024 г.

# Оглавление

# Варианты заданий

Решайте проблемы классификации текста с любым набором данных по вашему выбору. Классификация может быть бинарной или многоуровневой. Целевые объекты в выбранном вами наборе данных могут иметь любое физическое значение; одним из примеров является задача анализа тональности текста.

Необходимо сгенерировать два варианта векторизации признаков — на основе CountVectorizer и на основе TfidfVectorizer.

В качестве классификатора вы должны использовать два классификатора в зависимости от опций вашей группы:

| Группа | Классификатор №1 | Классификатор №2 |
|---|---|---|
| ИУ5И-22М | RandomForestClassifier | LogisticRegression |

# Текстовое описание набора данных:

Загрузите набор данных «20 групп новостей». Набор данных «20 групп новостей» представляет собой широко используемый набор данных для классификации текста, содержащий около 20 000 статей групп новостей, охватывающих 20 групп новостей с различными темами. По каждой теме около 1000 статей. Эти темы включают в себя: автомобили, электронные устройства, спорт, медицина, религия и многое другое. Текстовые данные в наборе данных были разделены на обучающий и тестовый наборы.

При загрузке набора данных вы можете использовать параметры subset='train' и subset='test', чтобы выбрать обучающий набор и тестовый набор. Кроме того, вы можете использовать параметр Remove=('headers', 'footers',

'quotes') для удаления верхних, нижних колонтитулов и кавычек из текста для лучшего анализа контента.

```
from sklearn.datasets import fetch_20newsgroups

data_train = fetch_20newsgroups(subset='train', remove=('headers', 'footers', 'quotes'))
data_test = fetch_20newsgroups(subset='test', remove=('headers', 'footers', 'quotes'))
```

# Предварительная обработка данных и извлечение признаков

Сначала были импортированы необходимые библиотеки и модули, включая экстракторы текстовых объектов (CountVectorizer и TfidfVectorizer), два классификатора (случайный лес и логистическая регрессия) и индикаторы оценки (точность). Далее, путем инициализации объектов CountVectorizer и TfidfVectorizer, текстовые данные преобразуются в матрицу частот слов (CountVectorizer) и матрицу TF-IDF (TfidfVectorizer). Затем эти матрицы функций используются для извлечения функций из текстовых данных обучающего набора и тестового набора. Наконец, целевые функции (метки категорий) обучающего набора и тестового набора извлекаются для обучения и оценки модели.

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

count_vectorizer = CountVectorizer()
tfidf_vectorizer = TfidfVectorizer()

X_train_count = count_vectorizer.fit_transform(data_train.data)
X_train_tfidf = tfidf_vectorizer.fit_transform(data_train.data)

X_test_count = count_vectorizer.transform(data_test.data)
X_test_tfidf = tfidf_vectorizer.transform(data_test.data)

y_train = data_train.target
y_test = data_test.target
```

# Обучение и оценка модели.

- **случайный классификатор леса**

Классификатор случайного леса — это алгоритм машинного обучения, основанный на деревьях решений. Он состоит из нескольких деревьев решений и делает прогнозы путем объединения этих деревьев.

```
rf_classifier = RandomForestClassifier()
rf_classifier.fit(X_train_count, y_train)
rf_count_acc = accuracy_score(y_test, rf_classifier.predict(X_test_count))

rf_classifier_tfidf = RandomForestClassifier()
rf_classifier_tfidf.fit(X_train_tfidf, y_train)
rf_tfidf_acc = accuracy_score(y_test, rf_classifier_tfidf.predict(X_test_tfidf))
```

- **классификатор логистической регрессии**

Логистическая регрессия использует логистическую функцию (также называемую сигмовидной функцией) для преобразования линейной комбинации признаков в значение вероятности, которое представляет вероятность принадлежности выборки к определенной категории.

```
lr_classifier = LogisticRegression()
lr_classifier.fit(X_train_count, y_train)
lr_count_acc = accuracy_score(y_test, lr_classifier.predict(X_test_count))

lr_classifier_tfidf = LogisticRegression()
lr_classifier_tfidf.fit(X_train_tfidf, y_train)
lr_tfidf_acc = accuracy_score(y_test, lr_classifier_tfidf.predict(X_test_tfidf))
```

## Распечатать результаты

```
Random Forest Classifier Accuracy (CountVectorizer): 0.5920074349442379
Random Forest Classifier Accuracy (TfidfVectorizer): 0.5920074349442379
Logistic Regression Accuracy (CountVectorizer): 0.6058151885289432
Logistic Regression Accuracy (TfidfVectorizer): 0.6736590546999469
```

## Вывод:

Эти результаты показывают точность использования различных методов представления объектов (CountVectorizer и TfidfVectorizer) и разных классификаторов (случайный лес и логистическая регрессия) в определенном наборе данных. В частности, точность классификатора случайного леса при использовании функций CountVectorizer и TfidfVectorizer составляет 0,592, а точность классификатора логистической регрессии при использовании функции CountVectorizer составляет 0,606, что немного выше, чем у классификатора случайного леса при использовании TfidfVectorizer. Особенность: Точность представления достигла 0,674, что является самым высоким показателем среди всех моделей. Это показывает, что в этом наборе данных модель логистической регрессии работает лучше, чем модель случайного леса, а метод представления объектов TfidfVectorizer достигает самой высокой точности среди всех моделей, вероятно, потому, что он лучше отражает функции в текстовых данных. Эти результаты подчеркивают влияние выбора подходящего метода представления признаков на производительность модели и то, что при обработке текстовых данных крайне важно учитывать характеристики данных.