Методы машинного обучения в автоматизированных системах

Рубежный контроль №2

Олейников И.И. ИУ5-22М

##Тема: Методы обработки текстов.

- Необходимо решить задачу классификации текстов на основе любого выбранного Вами датасета (кроме примера, который рассматривался в лекции). Классификация может быть бинарной или многоклассовой. Целевой признак из выбранного Вами датасета может иметь любой физический смысл, примером является задача анализа тональности текста.
- Необходимо сформировать два варианта векторизации признаков - на основе CountVectorizer и на основе TfidfVectorizer.

Импортирование необходимых библиотек

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

Подключение гугл диска

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Загрузка данных из CSV-файла

```
train_data =
pd.read_csv('/content/drive/MyDrive/RK/Electric_train.csv',
encoding='latin1')
test_data = pd.read_csv('/content/drive/MyDrive/RK/Electric_test.csv',
encoding='latin1')
```

```
train_data.head()
```


```
{"type":"dataframe","variable_name":"train_data"}
```


```
train_data.shape
```

```
(149503, 17)
```

```
test_data.head()
```


```
{"summary":"{\n  \"name\": \"test_data\",\n  \"rows\": 37376,\n  \"fields\": [\n    {\n      \"column\": \"VIN (1-10)\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 7396,\n        \"samples\": [\n          \"YV4BR0CL7L\",\n          \"KNDC4DLCXP\",\n          \"3FA6P0SU0K\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"County\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 160,\n        \"samples\": [\n          \"Pulaski\",\n          \"Kootenai\",\n          \"Riverside\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"City\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 573,\n        \"samples\": [\n          \"Potomac\",\n          \"Tonasket\",\n          \"White Salmon\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"State\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 37,\n        \"samples\": [\n          \"GA\",\n          \"NC\",\n          \"NJ\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Postal Code\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 4014.3277284200067,\n        \"min\": 1905.0,\n        \"max\": 99577.0,\n        \"num_unique_values\": 698,\n        \"samples\": [\n          98409.0,\n          98571.0,\n          98564.0\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Model Year\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 3.0457230295239657,\n        \"min\": 1997.0,\n        \"max\": 2024.0,\n        \"num_unique_values\": 20,\n        \"samples\": [\n          2024.0,\n          2000.0,\n          2010.0\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Make\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 38,\n        \"samples\": [\n          \"LUCID\",\n          \"BENTLEY\",\n          \"CHEVROLET\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Model\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 135,\n        \"samples\": [\n
```

```
\"EQE-CLASS SEDAN\",\n          \"TUCSON\",\n          \"RAV4\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Electric Vehicle Type\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          \"Plug-in
Hybrid Electric Vehicle (PHEV)\",\n          \"Battery Electric
Vehicle (BEV)\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Clean Alternative Fuel Vehicle (CAFV) Eligibility\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n
\"Eligibility unknown as battery range has not been researched\",\n
\"Clean Alternative Fuel Vehicle Eligible\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Electric Range\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
90.28720590541825,\n        \"min\": 0.0,\n        \"max\": 337.0,\n
\"num_unique_values\": 100,\n        \"samples\": [\n          87.0,\n
16.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Base MSRP\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 7896.078516960399,\n        \"min\":
0.0,\n        \"max\": 184400.0,\n        \"num_unique_values\": 29,\n
\"samples\": [\n          184400.0,\n          64950.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Legislative District\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
14.805024261762014,\n        \"min\": 1.0,\n        \"max\": 49.0,\n
\"num_unique_values\": 49,\n        \"samples\": [\n          45.0,\n
49.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"DOL
Vehicle ID\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 73544927.55559883,\n        \"min\":
24629.0,\n        \"max\": 479254772.0,\n
\"num_unique_values\": 37230,\n        \"samples\": [\n
196775799.0,\n          261329155.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Vehicle Location\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 697,\n        \"samples\": [\n          \"POINT
(-122.47913 47.2198)\",\n          \"POINT (-81.3035506 28.6039904)\"\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Electric Utility\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 62,\n
\"samples\": [\n          \"BONNEVILLE POWER ADMINISTRATION||CITY OF
PORT ANGELES - (WA)\",\n          \"CITY OF SUMAS - (WA)||PUD NO 1 OF
WHATCOM COUNTY\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
```

```
\"2020 Census Tract\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 2572968353.369728,\n        \"min\":
1001020100.0,\n        \"max\": 55009020504.0,\n
\"num_unique_values\": 1919,\n        \"samples\": [\n
53011041320.0,\n          53063010800.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"test_data"}
```

```
test_data.shape
```

```
(37376, 17)
```

Удаляем строки, содержащие пропущенные значения (NaN)

```
train_data.dropna(inplace=True)
test_data.dropna(inplace=True)

X_train = train_data['City']
y_train = train_data['Electric Vehicle Type']

X_test = test_data['City']
y_test = test_data['Electric Vehicle Type']
```

Проверить наличие пропущенных значений (NaN) в различных наборах данных.

```
def check_missing(data, name):
    missing = data.isnull().sum()
    print(f'У {name} {missing} пропущенных строк')

check_missing(train_data, 'train_data')
check_missing(test_data, 'test_data')
check_missing(X_train, 'X_train')
check_missing(X_test, 'X_test')
check_missing(y_train, 'y_train')
check_missing(y_test, 'y_test')
```

```
У train_data VIN (1-10)                                0
County                                                 0
City                                                   0
State                                                  0
Postal Code                                            0
Model Year                                             0
Make                                                   0
Model                                                  0
Electric Vehicle Type                                  0
Clean Alternative Fuel Vehicle (CAFV) Eligibility      0
Electric Range                                         0
Base MSRP                                              0
Legislative District                                   0
DOL Vehicle ID                                         0
```

```
Vehicle Location                                    0
Electric Utility                                    0
2020 Census Tract                                   0
dtype: int64 пропущенных строк
У test_data VIN (1-10)                                              0
County                                              0
City                                                0
State                                               0
Postal Code                                         0
Model Year                                          0
Make                                                0
Model                                               0
Electric Vehicle Type                               0
Clean Alternative Fuel Vehicle (CAFV) Eligibility   0
Electric Range                                      0
Base MSRP                                           0
Legislative District                                0
DOL Vehicle ID                                      0
Vehicle Location                                    0
Electric Utility                                    0
2020 Census Tract                                   0
dtype: int64 пропущенных строк
У X_train 0 пропущенных строк
У X_test 0 пропущенных строк
У y_train 0 пропущенных строк
У y_test 0 пропущенных строк
```

Векторизация текста

```
# Векторизация с помощью CountVectorizer
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)
X_test_counts = count_vect.transform(X_test)

# Векторизация с помощью TfidfVectorizer
tfidf_vect = TfidfVectorizer()
X_train_tfidf = tfidf_vect.fit_transform(X_train)
X_test_tfidf = tfidf_vect.transform(X_test)
```

Выполним оценку точности двух моделей машинного обучения (RandomForestClassifier и LogisticRegression) на двух разных типах векторизованных данных (CountVectorizer и TfidfVectorizer), полученных из текста

```
def evaluate_model(vectorizer_name, vectorizer_train, vectorizer_test,
model, model_name):
    start_time = time.time()
    obj_model = model
    obj_model.fit(vectorizer_train, y_train)
    predictions = obj_model.predict(vectorizer_test)
```

```
    accuracy = accuracy_score(y_test, predictions)
    duration = (time.time() - start_time) / 60

    print(f'Точность {vectorizer_name} + {model_name}: {accuracy:.4f},
время обучения классификатора: {duration:.2f} мин.', )

# Для CountVectorizer
evaluate_model('CountVectorizer', X_train_counts, X_test_counts,
RandomForestClassifier(), 'RandomForestClassifier')
evaluate_model('CountVectorizer', X_train_counts, X_test_counts,
LogisticRegression(max_iter=1000), 'LogisticRegression')

# Для TfidfVectorizer
evaluate_model('TfidfVectorizer', X_train_tfidf, X_test_tfidf,
RandomForestClassifier(), 'RandomForestClassifier')
evaluate_model('TfidfVectorizer', X_train_tfidf, X_test_tfidf,
LogisticRegression(max_iter=1000), 'LogisticRegression')
```

```
Точность CountVectorizer + RandomForestClassifier: 0.7791, время
обучения классификатора: 0.28 мин.
Точность CountVectorizer + LogisticRegression: 0.7794, время обучения
классификатора: 0.03 мин.
Точность TfidfVectorizer + RandomForestClassifier: 0.7788, время
обучения классификатора: 0.29 мин.
Точность TfidfVectorizer + LogisticRegression: 0.7794, время обучения
классификатора: 0.03 мин.
```

Анализ качества классификации

- Точность: Точность всех моделей практически идентична - около 0.7792. Это говорит о том, что задача не очень сложная, или, возможно, выборка ограничена и модели переобучены.
- Время обучения: LogisticRegression работает значительно быстрее, чем RandomForestClassifier, как с CountVectorizer, так и с TfidfVectorizer.

Вывод:

В данном случае LogisticRegression с CountVectorizer или TfidfVectorizer показывает практически одинаковое качество, но при этом работает существенно быстрее.

В конечном итоге выбор лучшего варианта зависит от приоритетов:

- Если скорость важнее всего: Логистическая регрессия с CountVectorizer.
- Если интерпретируемость важна: Логистическая регрессия с CountVectorizer.
- Если необходимо попробовать более сложную модель: RandomForestClassifier с CountVectorizer или TfidfVectorizer.