https://drive.google.com/file/d/1wkQDfUIBwY4SO5mC1JAua9d6JBa1sCcd/view?usp=sharing

```
! gdown 1wkQDfUIBwY4SO5mC1JAua9d6JBa1sCcd
```

```
Downloading...
From: https://drive.google.com/uc?id=1wkQDfUIBwY4SO5mC1JAua9d6JBa1sCcd
To: /content/train_dataset_train.csv
100% 50.5M/50.5M [00:00<00:00, 116MB/s]
```

```
import pandas as pd
import seaborn as sns
from sklearn.metrics import f1_score
```

```
import numpy as np
import pandas as pd
import torch
# import transformers
from tqdm import notebook
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import GridSearchCV
```

```
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('omw-1.4')
import re, string
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
```

```
df = pd.read_csv('/content/train_dataset_train.csv')
```

```
df.head()
```

| | id | name | groups |
|---|---|---|---|
| 0 | 1034 | ШОК-ЦЕНА Пена д/душа/бритья КУППЕР 200 мл АКС | 10 |
| 1 | 1035 | Мин.вода Нагутская №26 0.5л | 0 |
| 2 | 1036 | Пельмени Домашние вес ПО Прямицино. | 10 |
| 3 | 1037 | ПЕЧЕНЬЕ ОВСЯНО-ГРЕЧНЕВОЕ ЭКО БОТ 4600508719365 | 10 |
| 4 | 1038 | Спред растительно-жировой Масляничка 62% 170г ... | 10 |

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 658064 entries, 0 to 658063
Data columns (total 3 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   id      658064 non-null  int64
 1   name    658064 non-null  object
 2   groups  658064 non-null  int64
dtypes: int64(2), object(1)
memory usage: 15.1+ MB
```

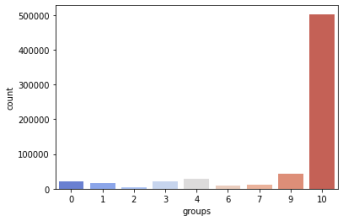
```
df.isna().sum()
```

```
id        0
name      0
groups    0
dtype: int64
```

```
# Функция для очистки текста
def clean_text(text):
  text = text.lower()
  return " ".join(re.sub(r"[^а-яа-z]", ' ', text).split())

# создаем список для хранения преобразованных данных
processed_text = []
# загружаем стоп-слова для английского языка
stop_words = stopwords.words('russian')
# инициализируем лемматайзер
lemmatizer = WordNetLemmatizer()

# для каждого сообщения text из столбца data['Message']
for text in df['name']:
    # cleaning
    text = clean_text(text)
    # tokenization
    text = word_tokenize(text)
    # удаление стоп-слов
    text = [word for word in text if word not in stop_words]

    # лемматизация
    text = [lemmatizer.lemmatize(w) for w in text]

    text = [word for word in text if len(word) >= 3]       #исключить короткие слова и символы

    # добавляем преобразованный текст в список processed_text
    processed_text.append(text)

# Сохраняем результат преобразования в новой колонке 'Processed_msg'


df['lemm_text'] = processed_text

df
```

| | id | name | groups | lemm_text | lemm_text_str |
|---|---|---|---|---|---|
| 0 | 1034 | ШОК-ЦЕНА Пена д/душа/бритья КУППЕР 200 мл АКС | 10 | [шок, цена, пена, душа, бритья, куппер, акс] | шок цена пена д душа бритья куппер мл акс |
| 1 | 1035 | Мин.вода Нагутская №26 0.5л | 0 | [мин, вода, нагутская] | мин вода нагутская л |
| 2 | 1036 | Пельмени Домашние вес ПО Прямицино. | 10 | [пельмени, домашние, вес, прямицино] | пельмени домашние вес прямицино |
| 3 | 1037 | ПЕЧЕНЬЕ ОВСЯНО-ГРЕЧНЕВОЕ ЭКО БОТ 4600508719365 | 10 | [печенье, овсяно, гречневое, эко, бот] | печенье овсяно гречневое эко бот |
| 4 | 1038 | Спред растительно-жировой Масляничка 62% 170г ... | 10 | [спред, растительно, жировой, масляничка, змж] | спред растительно жировой масляничка г змж |

```
sns.countplot(x='groups', data=df, palette='coolwarm');
```




```
df['lemm_text_str'] = df['lemm_text'].apply(lambda x: ' '.join(x))
```

```
df
```

| | id | name | groups | lemm_text | lemm_text_str |
|---|---|---|---|---|---|
| 0 | 1034 | ШОК-ЦЕНА Пена д/душа/бритья КУППЕР 200 мл АКС | 10 | [шок, цена, пена, душа, бритья, куппер, акс] | шок цена пена душа бритья куппер акс |
| 1 | 1035 | Мин.вода Нагутская №26 0.5л | 0 | [мин, вода, нагутская] | мин вода нагутская |
| 2 | 1036 | Пельмени Домашние вес ПО Прямицино. | 10 | [пельмени, домашние, вес, прямицино] | пельмени домашние вес прямицино |
| 3 | 1037 | ПЕЧЕНЬЕ ОВСЯНО-ГРЕЧНЕВОЕ ЭКО БОТ 4600508719365 | 10 | [печенье, овсяно, гречневое, эко, бот] | печенье овсяно гречневое эко бот |
| 4 | 1038 | Спред растительно-жировой Масляничка 62% 170г ... | 10 | [спред, растительно, жировой, масляничка, змж] | спред растительно жировой масляничка змж |
| ... | ... | ... | ... | ... | ... |
| 658059 | 659093 | КАРАЧИНСКАЯ 1,5 л *6 шт мин вода | 0 | [карачинская, мин, вода] | карачинская мин вода |
| 658060 | 659094 | Хлеб ДОНСКОЙ 350г Рузский х/з | 9 | [хлеб, донской, рузский] | хлеб донской рузский |
| 658061 | 659095 | Печенье сахарное FORSITE Сэндвич с шоколадно-с... | 6 | [печенье, сахарное, forsite, сэндвич, шоколадн... | печенье сахарное forsite сэндвич шоколадно сли... |
| 658062 | 659096 | Хлеб Бабушкин подов пшен 0,55кг п/уп(ШХЗ) | 9 | [хлеб, бабушкин, подов, пшен, шхз] | хлеб бабушкин подов пшен шхз |
| 658063 | 659097 | Газ.напиток ЭКСТРА-СИТРО ст/б 0.5л | 4 | [газ, напиток, экстра, ситро] | газ напиток экстра ситро |

658064 rows × 5 columns

```
# df_lemm·=·pd.read_csv('/content/df_lemm·(1).csv')
```

```
df_lemm.head()
```

| | lemm_str |
|---|---|
| 0 | шок цена пена д душа бритье куппер мл акс |
| 1 | мина вода нагутский л |
| 2 | пельмень домашний вес прямицино |
| 3 | печение овсяный гречневый эко бот |
| 4 | спред растительный жировой масляничка г змж |

```
df = df.join(df_lemm)
```

```
df
```

| | id | name | groups | lemm_str |
|---|---|---|---|---|
| 0 | 1034 | ШОК-ЦЕНА Пена д/душа/бритья КУППЕР 200 мл АКС | 10 | шок цена пена д душа бритье куппер мл акс |
| 1 | 1035 | Мин.вода Нагутская №26 0.5л | 0 | мина вода нагутский л |
| 2 | 1036 | Пельмени Домашние вес ПО Прямицино. | 10 | пельмень домашний вес прямицино |
| 3 | 1037 | ПЕЧЕНЬЕ ОВСЯНО-ГРЕЧНЕВОЕ ЭКО БОТ 4600508719365 | 10 | печение овсяный гречневый эко бот |
| 4 | 1038 | Спред растительно-жировой Масляничка 62% 170г ... | 10 | спред растительный жировой масляничка г змж |
| ... | ... | ... | ... | ... |
| 658059 | 659093 | КАРАЧИНСКАЯ 1,5 л *6 шт мин вода | 0 | карачинская л шт мина вода |
| 658060 | 659094 | Хлеб ДОНСКОЙ 350г Рузский х/з | 9 | хлеб донской г рузский х з |
| 658061 | 659095 | Печенье сахарное FORSITE Сэндвич с шоколадно-с... | 6 | печение сахарный сэндвич шоколадный сливочный ... |
| 658062 | 659096 | Хлеб Бабушкин подов пшен 0,55кг п/уп(ШХЗ) | 9 | хлеб бабушкин под пшен кг п уп шхз |
| 658063 | 659097 | Газ.напиток ЭКСТРА-СИТРО ст/б 0.5л | 4 | газ напиток экстра ситро ст б л |

658064 rows × 4 columns

## Train Test Split

[ ] ↳ *Скрыто 2 ячейки.*

## TF-IDF

[ ] ↳ *Скрыто 4 ячейки.*

## LogisticRegression

```
lg_model = LogisticRegression(random_state=19, class_weight='balanced')
```

```
%%time
lg_model.fit(train_features, y_train)
```

```
CPU times: user 1min 56s, sys: 36 s, total: 2min 32s
Wall time: 1min 50s
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
LogisticRegression(class_weight='balanced', random_state=19)
```

```
y_train_pred  = lg_model.predict(train_features)
```

```
f1_logistic_train = f1_score(y_train, y_train_pred, average='weighted')
f1_logistic_train
```

```
0.9604280411887569
```

```
from sklearn.metrics import recall_score
```

```
recall_score (y_train, y_train_pred, average='weighted')
```

```
0.9571233434134168
```

```
y_test_pred  = lg_model.predict(test_features)
```

```
f1_logistic_test = f1_score(y_test, y_test_pred, average='weighted')
f1_logistic_test
```

```
0.9537580103819576
```

```
recall_score(y_test, y_test_pred, average='weighted') # юез исключения коротких слов 0.960019450818302
```

```
0.9496558116917653
```

Попытка 1

recall_score(y_train, y_train_pred, average='weighted') - **0.9678317640198233**

f1_logistic_test = f1_score(y_test, y_test_pred, average='weighted') - 0.9652490431918442

**recall_score(y_test, y_test_pred, average='weighted') 0.9627468411175187**

---

Попытка 2

Разделен векторайзер, добавлены английские буквы, тест 0,2, min_df=2:

recall_score(y_train, y_train_pred, average='weighted') **0.9667775348512967**

f1_logistic_test = f1_score(y_test, y_test_pred, average='weighted') 0.963154431234822

**recall_score(y_test, y_test_pred, average='weighted') 0.9600647352465183**

---

Попытка 3

Разделен векторайзер, добавлены английские буквы, тест 0,1, min_df=1:

recall_score(y_train, y_train_pred, average='weighted') 0.96076580930369

recall_score(y_test, y_test_pred, average='weighted') 0.9572233956873889

в чемпионате: 0,9696

submission_2try

---

Попытка 4

без лемматизатора, тестовая 0,1, min_df=2: чуть выше попытки 3

---

Попытка 5

лемматизатор, тестовая 0,1, min_df=2:

recall_score(y_train, y_train_pred, average='weighted') 0.9622371635210363

recall_score(y_test, y_test_pred, average='weighted') 0.9590317139514033

submission_3try

---

Попытка 6

лемматизатор, тестовая 0,2, векторайзер на все данные, агшл буквы, min_df=2:

recall_score(y_train, y_train_pred, average='weighted') 0.9666521670582827

recall_score(y_test, y_test_pred, average='weighted') 0.9617059105103599

submission_3try

---

Попытка 7

лемматизатор, тестовая 0,2, векторайзер на все данные, агшл буквы, min_df=2, удалены слова короче 3:

recall_score(y_train, y_train_pred, average='weighted') 0.9666521670582827

recall_score(y_test, y_test_pred, average='weighted') 0.9617059105103599

lr_submission_4try

---

## XGBClassifier

[ ] *Скрыто 14 ячеек.*

## CatBoostClassifier

[ ] *Скрыто 6 ячеек.*

LogisticRegression и GridSearch

Финальный тест

[ ] ↳ *Скрыто 17 ячеек.*