Лабораторная работа 6

Загружаем библиотеки

```
pip install pandas umap-learn transformers datasets scikit-learn torch evaluate
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: umap-learn in /usr/local/lib/python3.11/dist-packages (0.5.7)
Requirement already satisfied: transformers in /usr/local/lib/python3.11/dist-packages (4.51.3)
Requirement already satisfied: datasets in /usr/local/lib/python3.11/dist-packages (2.14.4)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: torch in /usr/local/lib/python3.11/dist-packages (2.6.0+cu124)
Requirement already satisfied: evaluate in /usr/local/lib/python3.11/dist-packages (0.4.3)
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.9.0.post0
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: scipy>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from umap-learn) (1.15.3)
Requirement already satisfied: numba>=0.51.2 in /usr/local/lib/python3.11/dist-packages (from umap-learn) (0.60.0)
Requirement already satisfied: pynndescent>=0.5 in /usr/local/lib/python3.11/dist-packages (from umap-learn) (0.5.13)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from umap-learn) (4.67.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from transformers) (3.18.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.30.0 in /usr/local/lib/python3.11/dist-packages (from transformers)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from transformers) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.11/dist-packages (from transformers) (2024.11.6
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from transformers) (2.32.3)
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.21
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.11/dist-packages (from transformers) (0.5.3)
Requirement already satisfied: pyarrow>=8.0.0 in /usr/local/lib/python3.11/dist-packages (from datasets) (18.1.0)
Requirement already satisfied: dill<0.3.8,>=0.3.0 in /usr/local/lib/python3.11/dist-packages (from datasets) (0.3.7)
Requirement already satisfied: xxhash in /usr/local/lib/python3.11/dist-packages (from datasets) (3.5.0)
Requirement already satisfied: multiprocess in /usr/local/lib/python3.11/dist-packages (from datasets) (0.70.15)
Requirement already satisfied: fsspec>=2021.11.1 in /usr/local/lib/python3.11/dist-packages (from fsspec[http]>=2021.11.1-
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-packages (from datasets) (3.11.15)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.5.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.11/dist-packages (from torch) (4.13.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch) (3.1.6)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch) (1
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch) (1
Requirement already satisfied: nvidia-cudnn-cu12==9.1.0.70 in /usr/local/lib/python3.11/dist-packages (from torch) (9.1.0.
Requirement already satisfied: nvidia-cublas-cu12==12.4.5.8 in /usr/local/lib/python3.11/dist-packages (from torch) (12.4.
Requirement already satisfied: nvidia-cufft-cu12==11.2.1.3 in /usr/local/lib/python3.11/dist-packages (from torch) (11.2.1
Requirement already satisfied: nvidia-curand-cu12==10.3.5.147 in /usr/local/lib/python3.11/dist-packages (from torch) (10.
Requirement already satisfied: nvidia-cusolver-cu12==11.6.1.9 in /usr/local/lib/python3.11/dist-packages (from torch) (11.
Requirement already satisfied: nvidia-cusparse-cu12==12.3.1.170 in /usr/local/lib/python3.11/dist-packages (from torch) (1
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in /usr/local/lib/python3.11/dist-packages (from torch) (0.6.
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/dist-packages (from torch) (2.21.5)
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch) (12.4.12
Requirement already satisfied: nvidia-nvjitlink-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch) (12
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/dist-packages (from torch) (3.2.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-packages (from torch) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy==1.13.1->torch) (1
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.3.2
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (25.3.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.6.0
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (6.4
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (0.3.1
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.20
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.metrics import adjusted_rand_score, silhouette_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import re
import nltk
from nltk.corpus import stopwords
```

```
from nltk.stem import WordNetLemmatizer
import umap
from mpl_toolkits.mplot3d import Axes3D
import random
from collections import defaultdict


nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

Сбор данных (Collect data) и Загрузка данных:

Датасет 20 Newsgroups (20 Новостных Групп) из библиотеки sklearn. Этот датасет содержит коллекцию сообщений, опубликованных в разных группах новостей в 1990-е годы. Он включает текстовые данные, которые относятся к 20 различным категориям. Эти категории представляют собой разные темы, например, «спорт», «наука», «компьютеры», «прикладная математика», «политика» и т. д.

```
# Загрузка данных из 20 Newsgroups
def load_newsgroups_data(categories=None):
    return fetch_20newsgroups(subset='all', categories=categories, remove=('headers', 'footers', 'quotes'))

# Функция для создания подвыборки текстов и меток
def create_subset(texts, labels, samples_per_class=100):
    class_counts = defaultdict(int)
    texts_subset, labels_subset = [], []

    # Перемешиваем тексты и метки
    combined = list(zip(texts, labels))
    random.shuffle(combined)

    for text, label in combined:
        if class_counts[label] < samples_per_class:
            texts_subset.append(text)
            labels_subset.append(label)
            class_counts[label] += 1

        # Проверяем, достигли ли мы нужного количества образцов для всех классов
        if all(count >= samples_per_class for count in class_counts.values()):
            break

    return texts_subset, labels_subset


newsgroups = load_newsgroups_data()
texts, labels = newsgroups.data, newsgroups.target
label_names = newsgroups.target_names

# Создаем подвыборку
texts_subset, labels_subset = create_subset(texts, labels, samples_per_class=100)

# Обновляем переменные
texts, labels = texts_subset, labels_subset


print(f"Количество документов в подвыборке: {len(texts)}")
print(f"Количество уникальных классов: {len(set(labels))}")
```

```
Количество документов в подвыборке: 2000
Количество уникальных классов: 20
```

Распределение сообщений по категориям

```
category_distribution = defaultdict(int)
for label in labels_subset:
    category_distribution[newsgroups.target_names[label]] += 1

print("Распределение сообщений по категориям:")
```

```
for category, count in category_distribution.items():
    print(f"{category}: {count}")

print("\n---\n")
```

```
Распределение сообщений по категориям:
sci.electronics: 100
soc.religion.christian: 100
comp.graphics: 100
sci.crypt: 100
rec.sport.baseball: 100
sci.space: 100
alt.atheism: 100
talk.politics.mideast: 100
rec.sport.hockey: 100
comp.sys.ibm.pc.hardware: 100
talk.religion.misc: 100
comp.windows.x: 100
rec.autos: 100
comp.sys.mac.hardware: 100
sci.med: 100
misc.forsale: 100
comp.os.ms-windows.misc: 100
rec.motorcycles: 100
talk.politics.guns: 100
talk.politics.misc: 100

---
```

```
for text, label in zip(texts, labels):
    print(f"Категория: {newsgroups.target_names[label]}\nТекст: {text}\n{'-'*80}")
```

```
Категория: sci.electronics
Текст:
Not necessarily true; a short in one, if near the maximum series
voltage drop, will overvoltage the other one and short it too, more
--------------------------------------------------------------------------------
Категория: soc.religion.christian
Текст:
[stuff deleted for brevity]


Your very starting point is wrong.  Christianity is not based on following
a moral standard.  "For it is by grace you have been saved, through faith...
NOT BY WORKS so that no man may boast."  (Eph. 2:7-8)  You say that
you know the Bible well, and can recognize (do you mean recite?) many
passages from memory.  That could very well be so.  However, it looks like
there are a few more passages that you should pay attention to.  (Titus 3:5
and James 2:10 are among them.)

Obedience to the moral law is imporant.  However, it is supposed to be the
result of turning your life over to Christ and becoming a Christian.  It is
by no means the starting point.

--------------------------------------------------------------------------------
Категория: sci.electronics
Текст: Greetings. I've recently decided to chuck the linear regulators
        and learn the "black magic" art of switching power supplies...
        (before anyone flames me, I KNOW, both have their place :-)

        Anyways, I've built the basic up & down converters with pretty
        good results (>80% efficiency) but I'm running into problems
        when I try to design & build anything that puts out serious
        amps... I know it can be done (I have some 5V@200A guts on my
        bench) but something puzzles me: I'm using a simple choke as
        the storage element (basicly a toroid with a single winding)
        but ALL commercial models use transformers with MANY windings.
        I traced a few and they seem to use some of the winding for
        the usual error/feedback but some of the others seem to loose
        me... What are they for? Better than that, anyone have a full
        schematic for one of these that I could get a copy of? I'd
        love to see how they manage to squeeze out so much from such
        low volume :-)

        My other problems (in getting high amps & good efficiency) are
        1) Lack of sources of ideal components (calculated) and 2)
        Limited knowledge of the whole topic... I'm doing this on my
        own (not school) mind you (in fact, I have yet to take any
        course that covers transistors ;-)

        So, is the answer to #1 the accumulation of dead commercial
        models and truning into a scavanger (not that it's not what
        I'm doing now...) and #2 getting & understanding schematics
```

```
and a bit more of the [mind-boggling] theory?

Take care.

P.S. My goal is 12V @ ~25A in (car battery) -> 250VAC out and
```

Предобработка данных

```
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def preprocess(text):
    text = text.lower()
    text = re.sub(r'[^a-z\s]', '', text)  # убрать пунктуацию
    words = text.split()
    words = [lemmatizer.lemmatize(word) for word in words if word not in stop_words and len(word) > 2]
    return ' '.join(words)

clean_texts = [preprocess(text) for text in texts]
print(clean_texts)
```

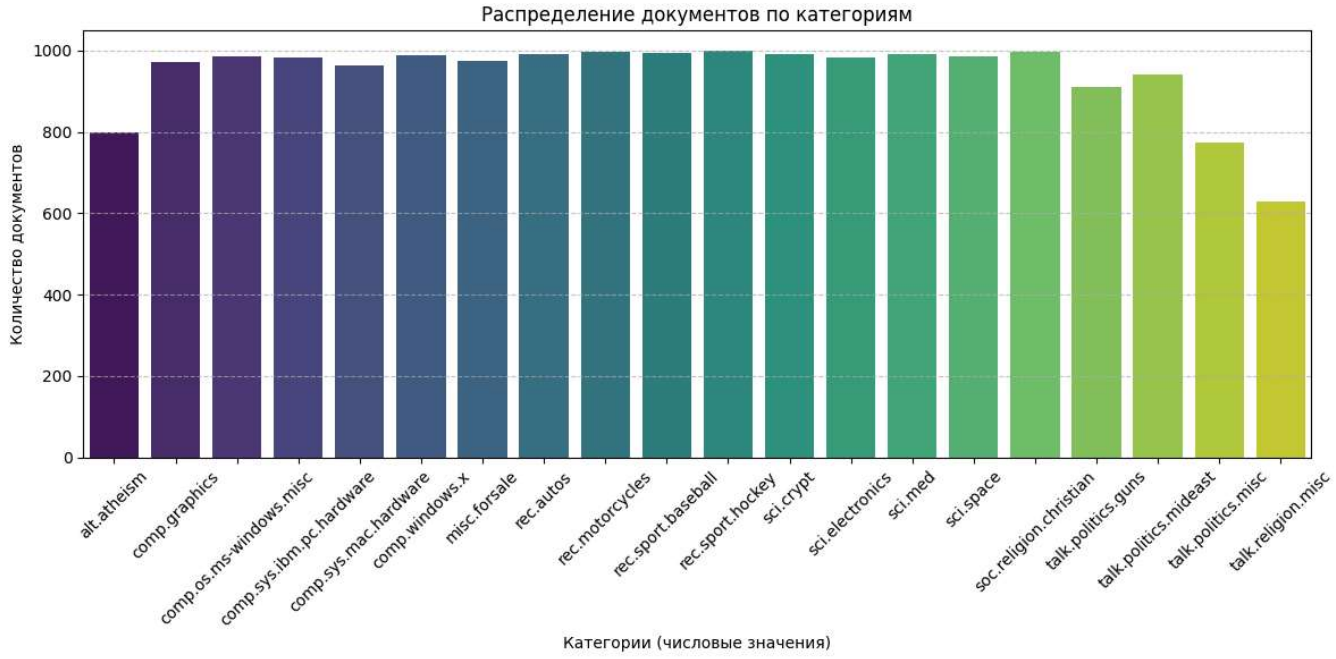
```
['necessarily true short one near maximum series voltage drop overvoltage one short', 'stuff deleted brevity starting point w
```

3. Анализируем данные(распределение статей по категориям) - тут я вновь взяла старые данные, просто с указанием единички, чтобы было проще

Анализ данных

```
newsgroups = load_newsgroups_data()

texts1 = newsgroups.data
labels1 = newsgroups.target
label_names1 = newsgroups.target_names

clean_texts1 = [preprocess(text) for text in texts1]


print(f"Общее количество документов в наборе: {len(clean_texts1)}")
print(f"Всего категорий: {len(label_names1)}")


plt.figure(figsize=(12, 6))
sns.countplot(x=labels1, palette='viridis')
plt.title("Распределение документов по категориям")
plt.xlabel("Категории (числовые значения)")
plt.ylabel("Количество документов")
plt.xticks(ticks=range(len(label_names1)), labels=label_names1, rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```
Общее количество документов в наборе: 18846
Всего категорий: 20
<ipython-input-9-a44b7f5931b8>:15: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and

  sns.countplot(x=labels1, palette='viridis')
```



Векторизация текста

```
vectorizer = TfidfVectorizer(max_features=20000)
X = vectorizer.fit_transform(clean_texts)
X1 = vectorizer.fit_transform(clean_texts1)
print(X1.shape)
```

```
(18846, 20000)
```

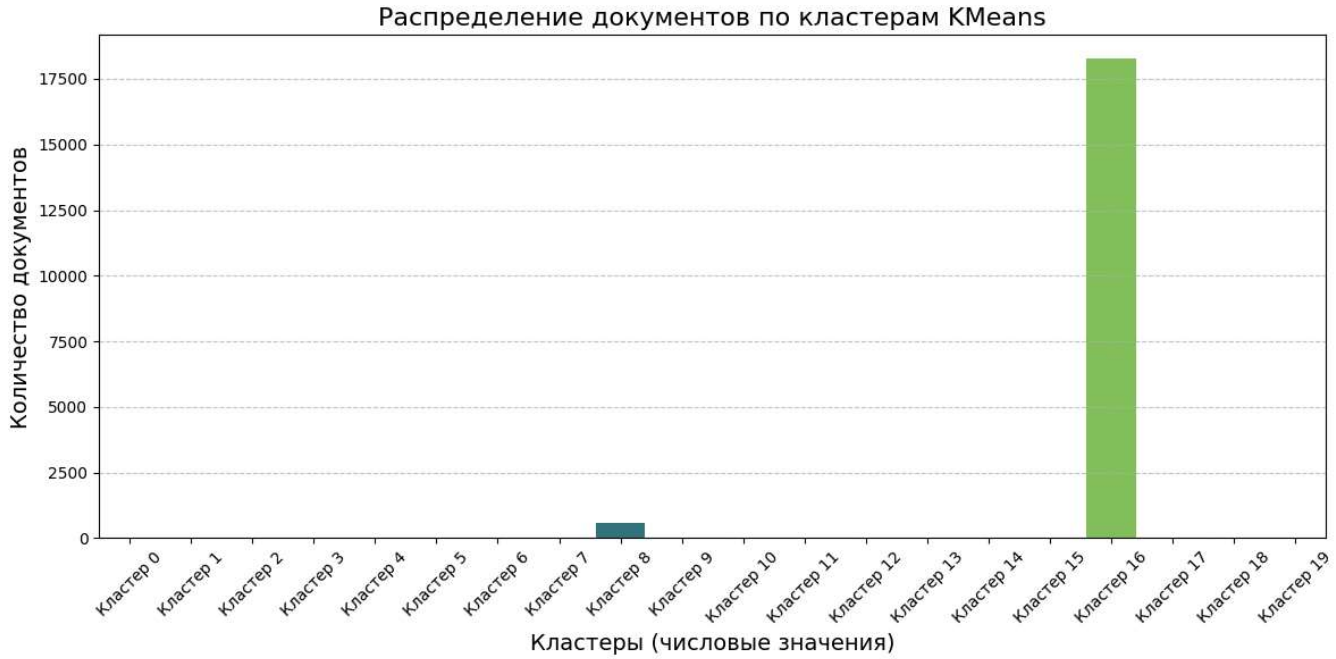
Кластеризация

```
kmeans = KMeans(n_clusters=len(label_names1), random_state=42)
clusters = kmeans.fit_predict(X1)
```

```
# Кластеризация KMeans
n_clusters = len(label_names1)  # Используем количество категорий как количество кластеров
```

```
# Построение графика распределения документов по кластерам
plt.figure(figsize=(12, 6))
sns.countplot(x=clusters, palette='viridis')
plt.title("Распределение документов по кластерам KMeans", fontsize=16)
plt.xlabel("Кластеры (числовые значения)", fontsize=14)
plt.ylabel("Количество документов", fontsize=14)
plt.xticks(ticks=range(n_clusters), labels=[f'Кластер {i}' for i in range(n_clusters)], rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```
<ipython-input-25-316ba4a2a182>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and

  sns.countplot(x=clusters, palette='viridis')
```



Разделение получилось весьма неудачно. Данные разделились на 2 кастера. Большенство данных были отнесены к 16 класеру (soc.religion.christian)

Сравнение данных

```
ari = adjusted_rand_score(labels1, clusters)
sil_score = silhouette_score(X1, clusters)

print(f"Adjusted Rand Index (ARI): {ari:.4f}")
print(f"Silhouette Score: {sil_score:.4f}")
```

```
Adjusted Rand Index (ARI): 0.0007
Silhouette Score: -0.0163
```

- ARI = 0.0007 — очень близко к нулю, что означает, что полученные кластеры практически не соответствуют исходным категориям. Другими словами, кластеризация не выявила структуру, совпадающую с настоящими классами.
- Silhouette Score = -0.0163 — слегка отрицательное значение, что указывает на слабую или плохую структуру кластеров. Кластеры, скорее всего, плохо отделены, и объекты могут быть распределены неудачно.

7. Разделение на train / val / test

```
#train+val / test
X_trainval, X_test, y_trainval, y_test = train_test_split(X, labels, test_size=0.15, stratify=labels, random_state=42)

#train / val
X_train, X_val, y_train, y_val = train_test_split(X_trainval, y_trainval, test_size=0.1765, stratify=y_trainval, random_state=42)
# 0.1765 * 0.85 ≈ 0.15, чтобы val тоже была 15%

print(f"Train size: {X_train.shape[0]}")
print(f"Validation size: {X_val.shape[0]}")
print(f"Test size: {X_test.shape[0]}")
```

```
Train size: 1399
Validation size: 301
Test size: 300
```

ЧАСТЬ 2

```
from sklearn.model_selection import train_test_split
from transformers import AutoTokenizer, AutoModelForSequenceClassification
from torch.utils.data import Dataset
from transformers import Trainer, TrainingArguments
import evaluate
import numpy as np
import torch
```

Разделение на выборку. Используем модель huggingface

```
idx_all = list(range(len(clean_texts)))

idx_trainval, idx_test = train_test_split(idx_all, test_size=0.15, stratify=labels, random_state=42)
idx_train, idx_val = train_test_split(idx_trainval, test_size=0.1765, stratify=[labels[i] for i in idx_trainval], random_state=42

X_train_texts = [clean_texts[i] for i in idx_train]
y_train = [labels[i] for i in idx_train]

X_val_texts = [clean_texts[i] for i in idx_val]
y_val = [labels[i] for i in idx_val]

X_test_texts = [clean_texts[i] for i in idx_test]
y_test = [labels[i] for i in idx_test]
```

Загружаем модель

```
model_name = "distilbert-base-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSequenceClassification.from_pretrained(model_name, num_labels=len(label_names))
```

```
Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back to regular HTTP download. For b
WARNING:huggingface_hub.file_download:Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Fallin
model.safetensors: 100%  268M/268M [00:04<00:00, 23.1MB/s]
Some weights of DistilBertForSequenceClassification were not initialized from the model checkpoint at distilbert-base-uncased
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

Класс для токенизации модели и хранения меток классов

```
class TextDataset(Dataset):
    def __init__(self, texts, labels):
        self.encodings = tokenizer(texts, truncation=True, padding=True, max_length=512)
        self.labels = labels
    def __getitem__(self, idx):
        item = {k: torch.tensor(v[idx]) for k, v in self.encodings.items()}
        item["labels"] = torch.tensor(self.labels[idx], dtype=torch.long)
        return item

    def __len__(self):
        return len(self.labels)


train_dataset = TextDataset(X_train_texts, y_train)
val_dataset = TextDataset(X_val_texts, y_val)
test_dataset = TextDataset(X_test_texts, y_test)
```

Настройка обучения модели

```
training_args = TrainingArguments(
    output_dir="./results",
    eval_strategy="epoch",  # eval_strategy заменен на evaluation_strategy
    save_strategy="epoch",  # добавлено save_strategy, чтобы совпало с evaluation_strategy
    num_train_epochs=3,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    logging_dir="./logs",
    logging_steps=10,
```

```
    load_best_model_at_end=True,
    metric_for_best_model="accuracy"
)
```

```
accuracy = evaluate.load("accuracy")
```

```
Downloading builder script: 100%  4.20k/4.20k [00:00<00:00, 302kB/s]
```

```
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    preds = np.argmax(logits, axis=1)
    return accuracy.compute(predictions=preds, references=labels)
```

```
trainer = Trainer(
model=model,
args=training_args,
train_dataset=train_dataset,
eval_dataset=val_dataset,
compute_metrics=compute_metrics,
)
```

Тренируем модель

```
trainer.train()
```

[525/525 04:04, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Accuracy
1	1.277400	1.400599	0.524917
2	0.814600	1.437330	0.561462
3	0.575900	1.349267	0.581395

```
TrainOutput(global_step=525, training_loss=0.9212184329259964, metrics={'train_runtime': 245.2415,
'train_samples_per_second': 17.114, 'train_steps_per_second': 2.141, 'total_flos': 556144139612160.0, 'train_loss':
0.9212184329259964, 'epoch': 3.0})
```

Training Loss падает с 1.28 до 0.58 — это хороший знак, модель обучается и лучше подстраивается под обучающие данные. Validation Loss изменяется от 1.40 до 1.35, но при этом не снижается стабильно, а даже немного растёт во 2-й эпохе. Это может означать, что модель пока не очень хорошо обобщается на новые данные. Accuracy растёт с 52.5% до 58.1% — тоже положительный сигнал, модель лучше предсказывает на валидационном наборе. Разрыв между Training Loss и Validation Loss говорит о том, что модель может немного переобучаться (слишком хорошо подстраивается под обучающие данные, но хуже работает на новых).

Вывод

Модель учится: Training Loss падает, Accuracy растёт. Но обобщение пока не очень: Validation Loss не снижается стабильно, что может говорить о переобучении или недостаточной сложности модели.

```
trainer.evaluate(test_dataset)
```

[19/19 00:01]

```
{'eval_loss': 1.6916236877441406,
 'eval_accuracy': 0.5533333333333333,
 'eval_runtime': 2.016,
 'eval_samples_per_second': 74.405,
 'eval_steps_per_second': 9.425,
 'epoch': 3.0}
```

eval_loss = 1.6916: Это значение потери на валидационном наборе данных. Чем ниже это значение, тем лучше модель справляется с предсказаниями. В данном случае, значение потери относительно высокое, что может указывать на то, что модель не совсем точно предсказывает результаты.

eval_accuracy = 0.5533: Это значение точности, показывающее, что модель правильно предсказывает примерно 55.33% случаев. Это достаточно низкий уровень точности, особенно если у вас есть классы, которые модель должна различать. Это может означать, что модель нуждается в улучшении.

eval_runtime = 2.016 секунд: Время, затраченное на оценку модели. Это относительно быстро, что хорошо для производительности.

eval_samples_per_second = 74.405: Это скорость обработки данных во время оценки. Модель обрабатывает более 74 образцов в секунду, что является хорошим показателем.

eval_steps_per_second = 9.425: Скорость шагов оценки. Это также хороший показатель, показывающий, что модель эффективно