```
from IPython.display import display, Javascript

display(Javascript('''
    function saveNotebook() {
        console.log("Автосохранение ноутбука...");
        IPython.notebook.save_checkpoint();
    }
    setInterval(saveNotebook, 60000);  // Сохранение каждые 60 секунд
'''))
```

```
!pip install datasets
```

Show hidden output

```
import pandas as pd
from datasets import Dataset, DatasetDict
from transformers import AutoTokenizer, AutoModelForSequenceClassification, TrainingArguments, Trainer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, precision_recall_fscore_support, classification_report
import numpy as np
import re


# Загрузка данных с делением текстов по стилям
df = pd.read_excel('/content/RuFoLa_new texts_styles.xlsx')


X = df['text']
y = df['text_genre']

# Разделение датасета на обучающий и тестовый
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Кодирование данных
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)


# Конвертация в формат, подходящий для модели
train_dataset = Dataset.from_pandas(pd.DataFrame({'text': X_train, 'label': y_train_encoded}))
test_dataset = Dataset.from_pandas(pd.DataFrame({'text': X_test, 'label': y_test_encoded}))

dataset = DatasetDict({
    'train': train_dataset,
    'test': test_dataset
})


# Загрузка модели и токенизатора
model_name = "bert-base-multilingual-cased" #Относительно быстрая модель для классификации
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSequenceClassification.from_pretrained(
    model_name,
    num_labels=len(label_encoder.classes_),
    hidden_dropout_prob=0.3
)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 100%                                 49.0/49.0 [00:00<00:00, 2.72kB/s]
config.json: 100%                                           625/625 [00:00<00:00, 24.6kB/s]
vocab.txt: 100%                                             996k/996k [00:00<00:00, 17.8MB/s]
tokenizer.json: 100%                                        1.96M/1.96M [00:00<00:00, 44.1MB/s]
model.safetensors: 100%                                     714M/714M [00:10<00:00, 47.9MB/s]
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-multilingual-cased an
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```
# TТокенизация и сохранение данных
def preprocess_function(examples):
    return tokenizer(examples['text'], truncation=True, padding='max_length', max_length=128)

tokenized_datasets = dataset.map(preprocess_function, batched=True)

# Фукнция для вычисления метрик
def compute_metrics(p):
    preds = np.argmax(p.predictions, axis=1)
    precision, recall, f1, _ = precision_recall_fscore_support(p.label_ids, preds, average='weighted')
    acc = accuracy_score(p.label_ids, preds)
    return {
        'accuracy': acc,
        'precision': precision,
        'recall': recall,
        'f1': f1,
    }
```

```
Map: 100%                                       1788/1788 [00:02<00:00, 655.26 examples/s]
Map: 100%                                       448/448 [00:00<00:00, 491.86 examples/s]
```

```
# Определение аргументов для обучения
training_args = TrainingArguments(
    output_dir='./results',
    evaluation_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    num_train_epochs=4, #оптимальное количество эпох, определенное в результате запуска на 20 и 10 эпох
    weight_decay=0.01,
    logging_dir='./logs',
    logging_steps=10,
    save_strategy="epoch",
    load_best_model_at_end=True,
)


# Инициализация
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets['train'],
    eval_dataset=tokenized_datasets['test'],
    tokenizer=tokenizer,
    compute_metrics=compute_metrics,
)

# Обучение
trainer.train()
```

```
[896/896 3:02:00, Epoch 4/4]
```

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.892600	0.848819	0.611607	0.615344	0.611607	0.589592
2	0.742200	0.810568	0.627232	0.639273	0.627232	0.623765
3	0.691900	0.820346	0.622768	0.652203	0.622768	0.613751

```
[18/56 00:50 < 01:52, 0.34 it/s]
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1471: UndefinedMetricWarning: Precision and F-score
  _warn_prf(average, modifier, msg_start, len(result))
[896/896 3:06:22, Epoch 4/4]
```

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.892600	0.848819	0.611607	0.615344	0.611607	0.589592

```
# Оценка модели
print("Оценка модели...")
metrics = trainer.evaluate()

print(f"Метрики: {metrics}")
```

```
Оценка модели...
[56/56 02:45]
Метрики: {'eval_loss': 0.8105681538581848, 'eval_accuracy': 0.6272321428571429, 'eval_precision': 0.6392732590025414, 'eval_rec
```

```
predictions = trainer.predict(tokenized_datasets['test'])
preds = np.argmax(predictions.predictions, axis=1)
y_pred = label_encoder.inverse_transform(preds)
```

```
# Отчет
accuracy = accuracy_score(y_test, y_pred)
classification_report_str = classification_report(y_test, y_pred)

print(f'Точность: {accuracy:.2f}')
print('Отчет о классификации:')
print(classification_report_str)
```

```
Точность: 0.63
Отчет о классификации:
                         precision    recall  f1-score   support

         Научный стиль       0.22      0.25      0.24        16
Публицистический стиль       0.62      0.66      0.64       154
Разговорно-бытовой стиль       0.60      0.88      0.71        74
   Художественный стиль       0.70      0.54      0.61       204

              accuracy                           0.63       448
             macro avg       0.54      0.58      0.55       448
          weighted avg       0.64      0.63      0.62       448
```