МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский Технический Университет Связи И Информатики (MTUCI)»

Кафедра «Математическая кибернетика и информационные технологии»

Лабораторная Работа 4

по дисциплине

«Машинное обучение»

Выполнил: студент 3 курса гр. БВТ2201
Ньяти Каелиле

Москва 2025 г

Содержание

1. Задание

Написать BERT модель которая будет способна классифицировать текст на токсичный или не токсичный. Датасет можно выбрать любой, но если лень искать то можете взять этот. Аккуратно выбирать размеры модели так как у RNN моделей большие проблемы со скоростью обучения. В качестве токенизатора можно взять любой токенизатор с hf либо написать свой эмбеддинг слой который будет превращать именно слова в эмбеддинги, тут по желанию.

В качестве лосса это BCE. В качестве оптимизатор Adam/AdamW, lr=3e-4. Но если захотите что другое то можно другое. Loss можете взвесить так как явный дисбаланс классов. По метрикам считайте accuracy, f1, roc-auc, pr-auc. Не забудьте проводить расчеты на cuda.

2. Ход работы

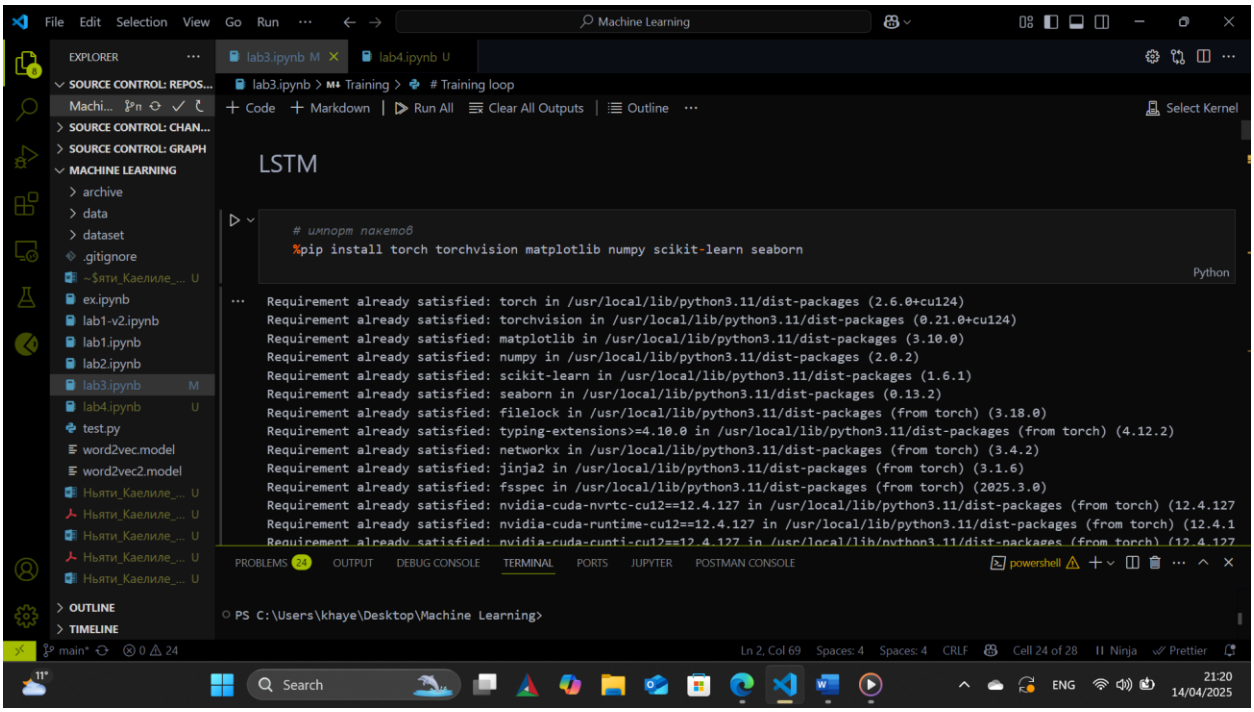


Рис 1. Библиотекы

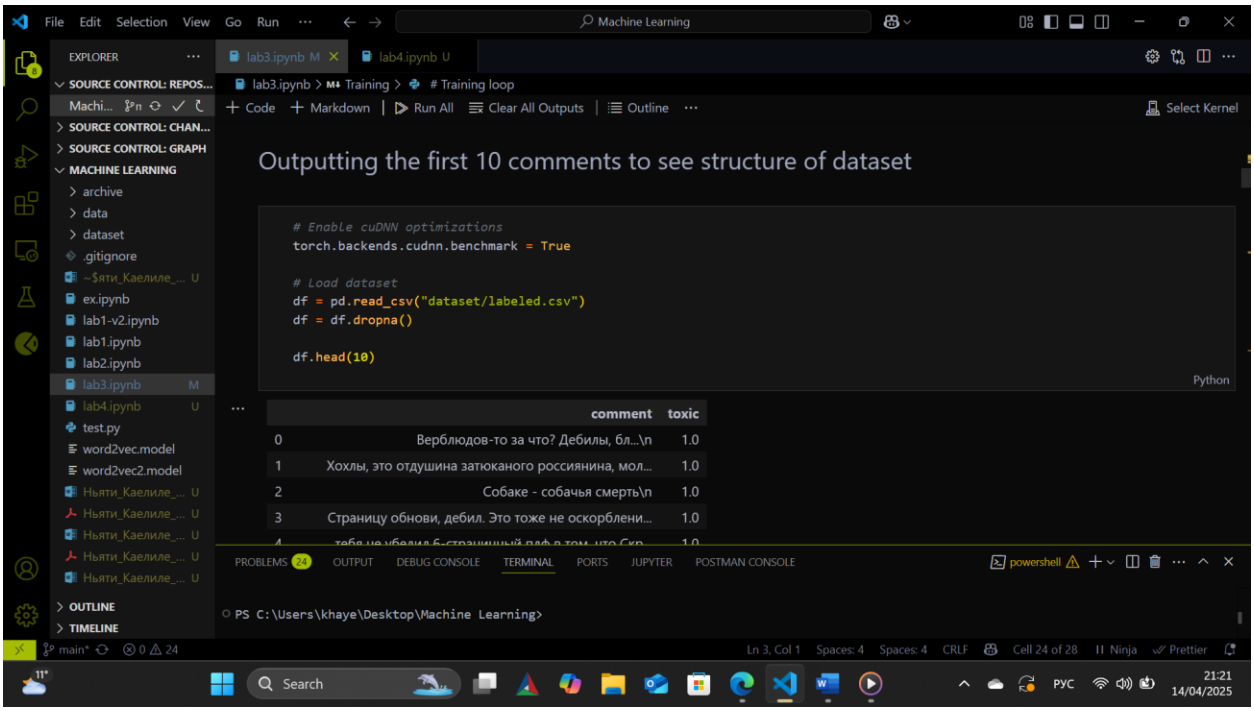
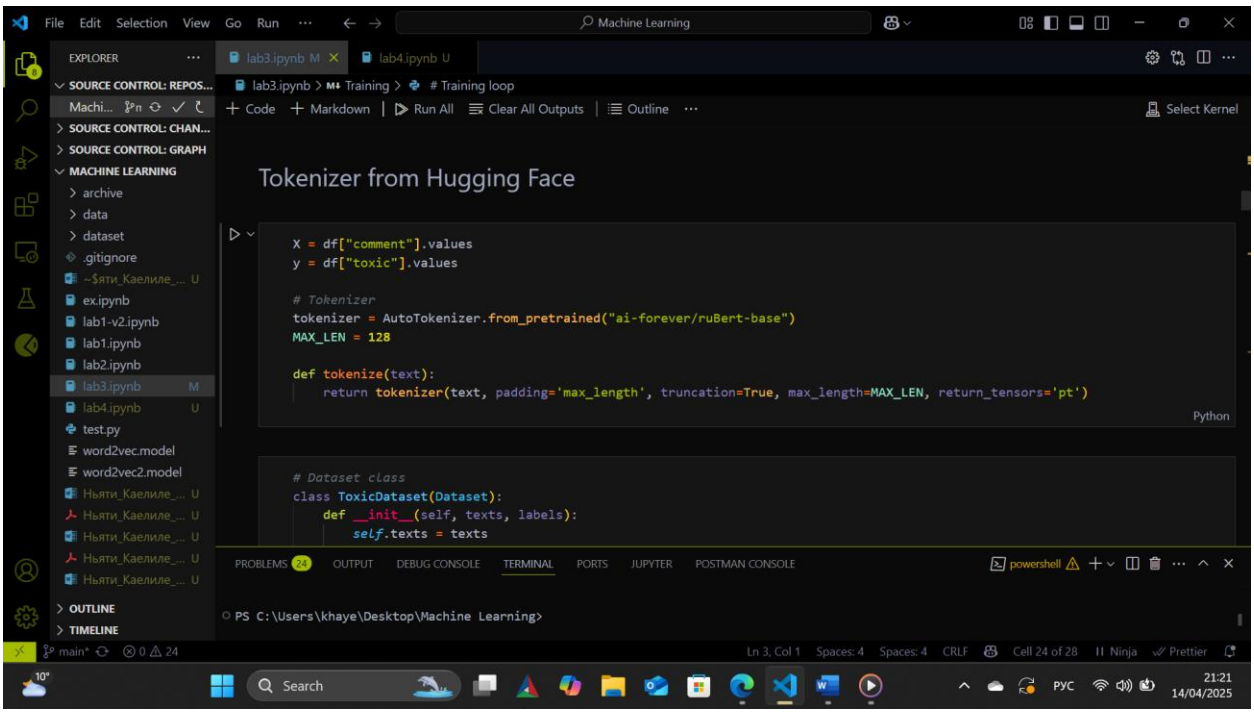

Outputting the first 10 comments to see structure of dataset
# Enable cuDNN optimizations
torch.backends.cudnn.benchmark = True
# Load dataset
df = pd.read_csv("dataset/labeled.csv")
df = df.dropna()
df.head(10)
Python
comment toxic
0 Верблюдов-то за что? Дебилы, бл...\n 1.0
1 Хохлы, это отдушина затюканого россиянина, мол... 1.0
2 Собаке - собачья смерть\n 1.0
3 Страницу обнови, дебил. Это тоже не оскорблени... 1.0
PS C:\Users\khaye\Desktop\Machine Learning>

Рис 2. Дата


Tokenizer from Hugging Face
X = df["comment"].values
y = df["toxic"].values
# Tokenizer
tokenizer = AutoTokenizer.from_pretrained("ai-forever/ruBert-base")
MAX_LEN = 128
def tokenize(text):
return tokenizer(text, padding='max_length', truncation=True, max_length=MAX_LEN, return_tensors='pt')
Python
# Dataset class
class ToxicDataset(Dataset):
def __init__(self, texts, labels):
self.texts = texts
PS C:\Users\khaye\Desktop\Machine Learning>

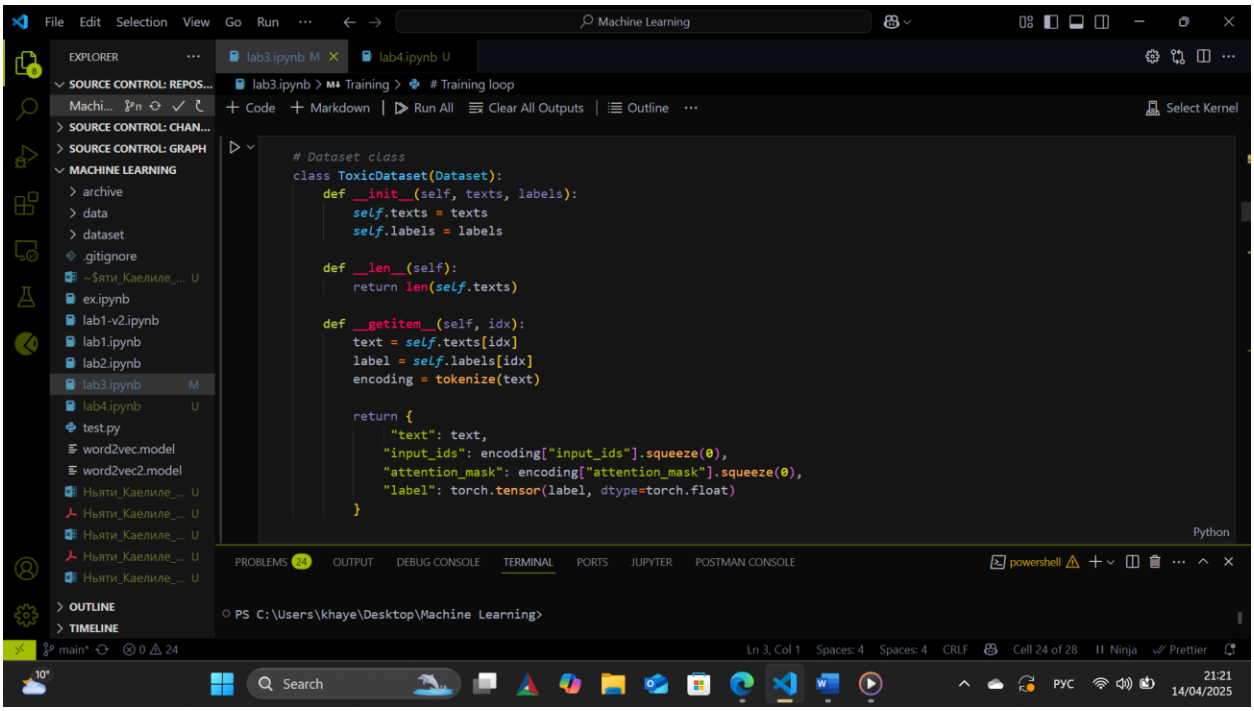
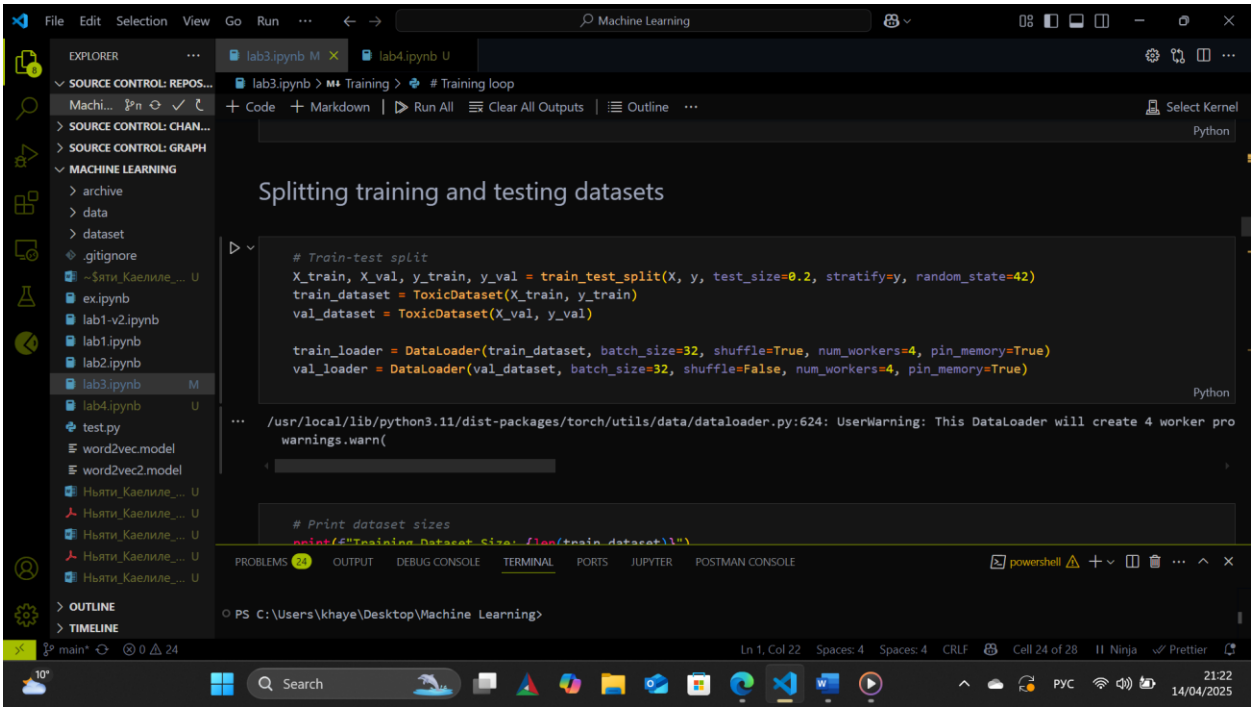
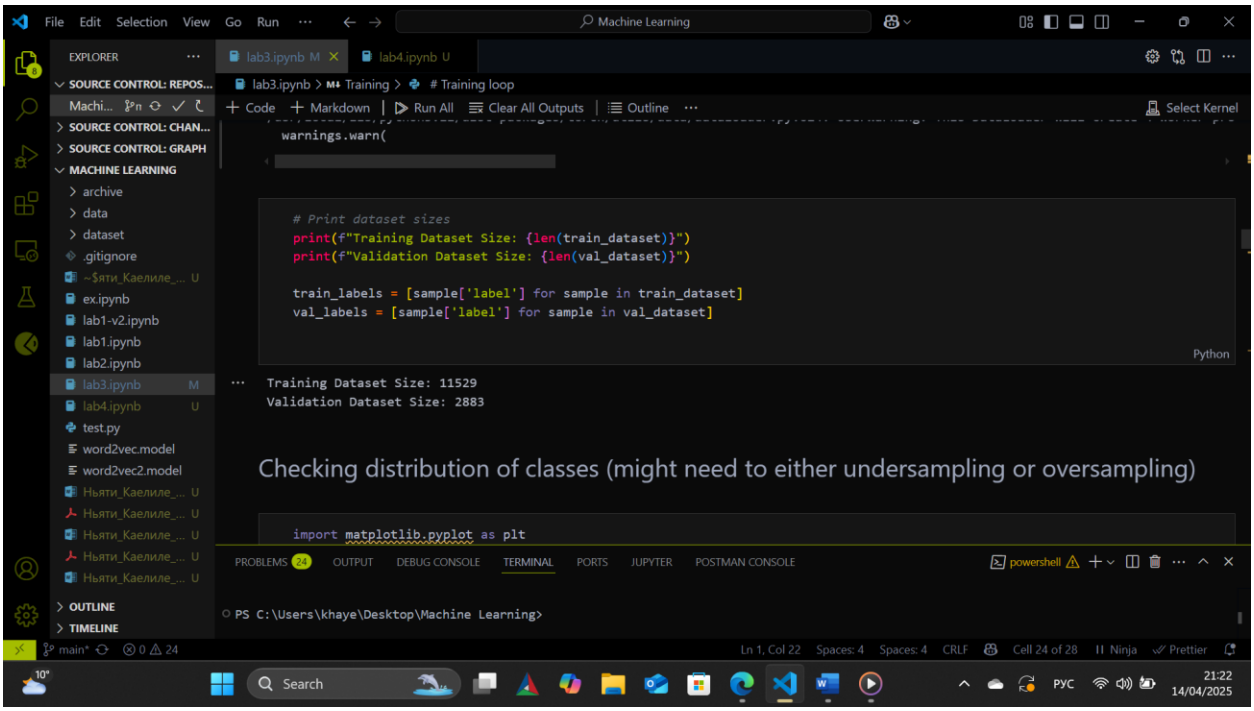
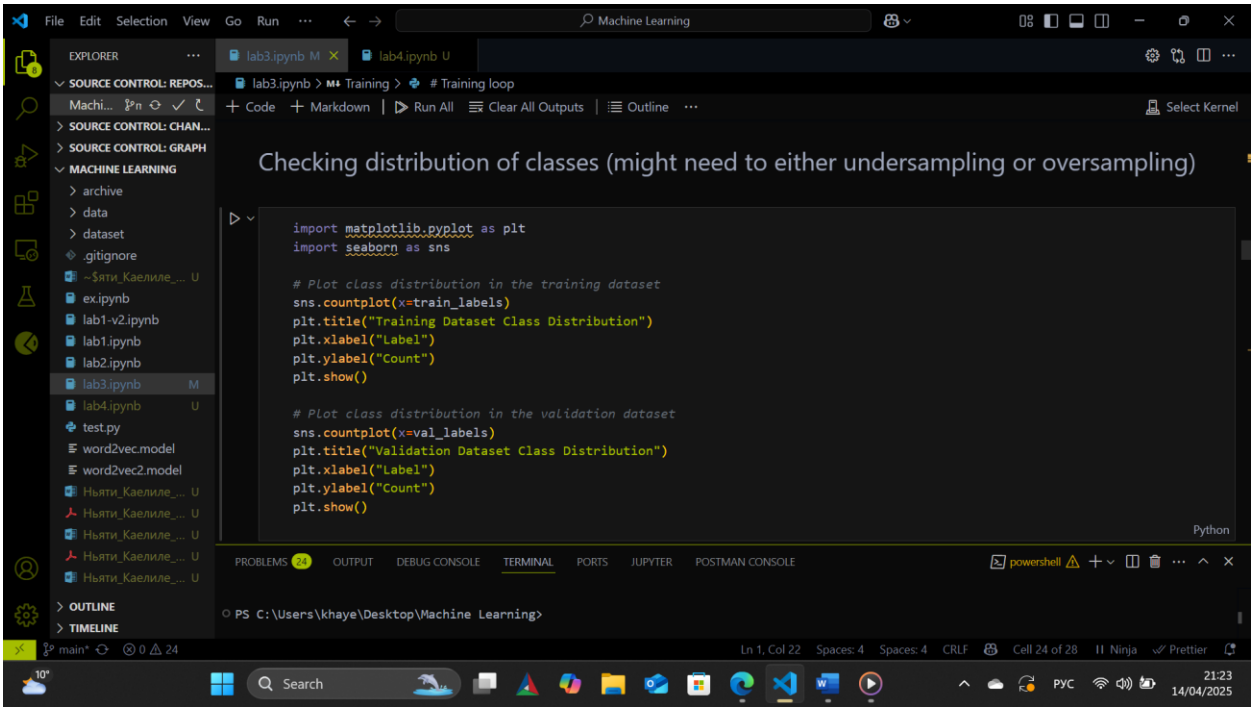
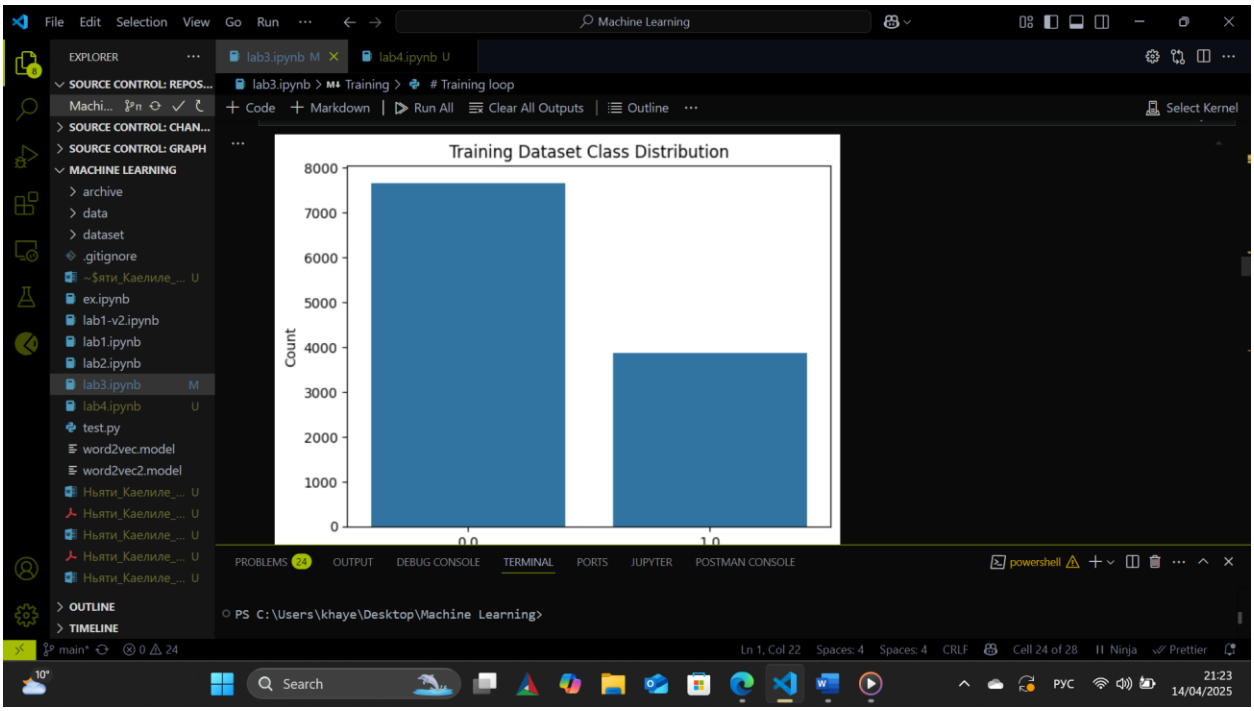


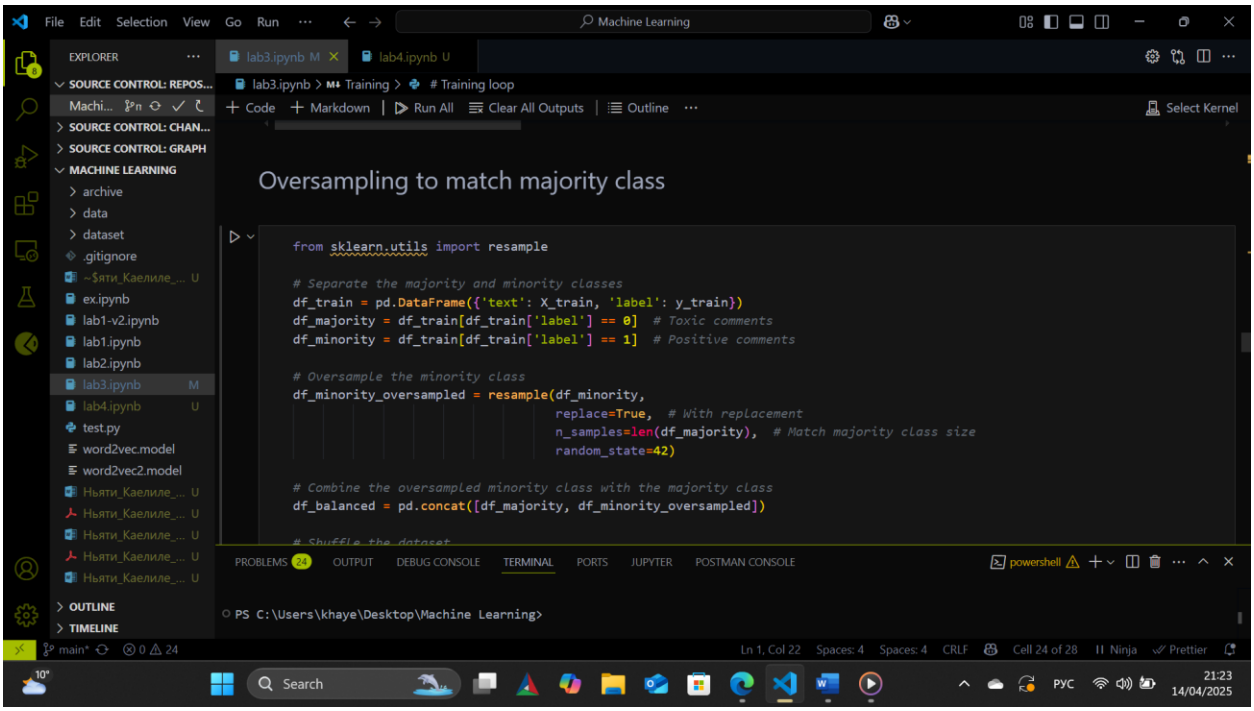
Рис 3. Токенизатор




warnings.warn(
# Print dataset sizes
print(f"Training Dataset Size: {len(train_dataset)}")
print(f"Validation Dataset Size: {len(val_dataset)}")
train_labels = [sample['label'] for sample in train_dataset]
val_labels = [sample['label'] for sample in val_dataset]
Python
Training Dataset Size: 11529
Validation Dataset Size: 2883
Checking distribution of classes (might need to either undersampling or oversampling)
import matplotlib.pyplot as plt
PS C:\Users\khaye\Desktop\Machine Learning>


Checking distribution of classes (might need to either undersampling or oversampling)
import matplotlib.pyplot as plt
import seaborn as sns
# Plot class distribution in the training dataset
sns.countplot(x=train_labels)
plt.title("Training Dataset Class Distribution")
plt.xlabel("Label")
plt.ylabel("Count")
plt.show()
# Plot class distribution in the validation dataset
sns.countplot(x=val_labels)
plt.title("Validation Dataset Class Distribution")
plt.xlabel("Label")
plt.ylabel("Count")
plt.show()
Python
PS C:\Users\khaye\Desktop\Machine Learning>


Training Dataset Class Distribution
Count
8000
7000
6000
5000
4000
3000
2000
1000
0
0.0
1.0
PS C:\Users\khaye\Desktop\Machine Learning>


File Edit Selection View Go Run
Machine Learning
EXPLORER
lab3.ipynb M
lab4.ipynb U
lab3.ipynb > Training > # Training loop
+ Code + Markdown | Run All Clear All Outputs | Outline
Select Kernel
Oversampling to match majority class
from sklearn.utils import resample
# Separate the majority and minority classes
df_train = pd.DataFrame({'text': X_train, 'label': y_train})
df_majority = df_train[df_train['label'] == 0] # Toxic comments
df_minority = df_train[df_train['label'] == 1] # Positive comments
# Oversample the minority class
df_minority_oversampled = resample(df_minority,
replace=True, # With replacement
n_samples=len(df_majority), # Match majority class size
random_state=42)
# Combine the oversampled minority class with the majority class
df_balanced = pd.concat([df_majority, df_minority_oversampled])
PROBLEMS 24 OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER POSTMAN CONSOLE
powershell
PS C:\Users\khaye\Desktop\Machine Learning>
Ln 1, Col 22 Spaces: 4 Spaces: 4 CRLF Cell 24 of 28 Ninja Prettier
Search
РУС
21:23
14/04/2025

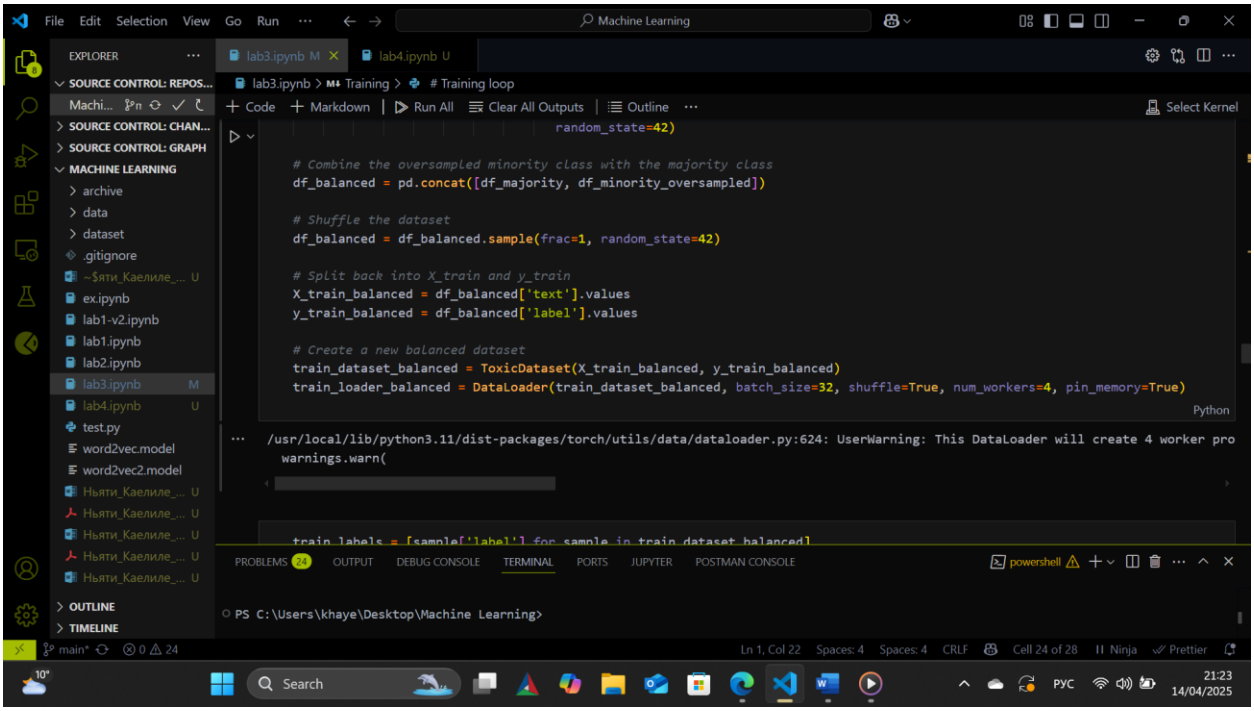
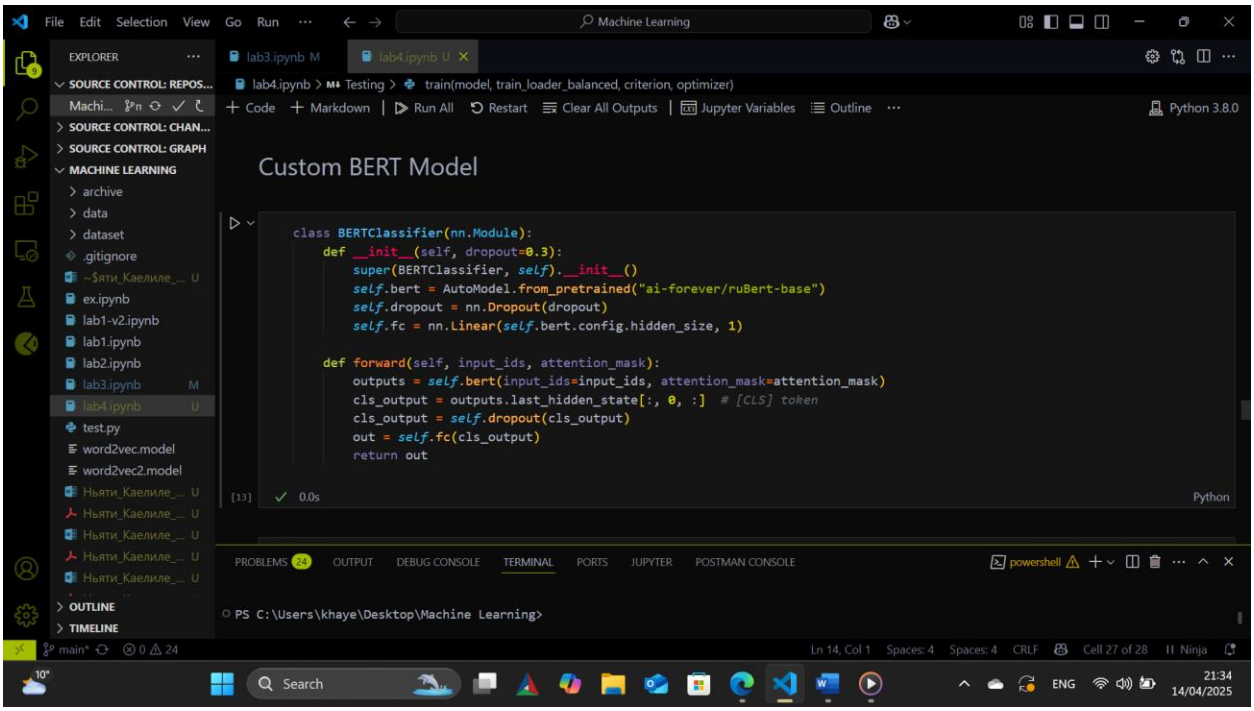

File Edit Selection View Go Run
Machine Learning
EXPLORER
lab3.ipynb M
lab4.ipynb U
lab3.ipynb > Training > # Training loop
+ Code + Markdown | Run All Clear All Outputs | Outline
Select Kernel
random_state=42)
# Combine the oversampled minority class with the majority class
df_balanced = pd.concat([df_majority, df_minority_oversampled])
# Shuffle the dataset
df_balanced = df_balanced.sample(frac=1, random_state=42)
# Split back into X_train and y_train
X_train_balanced = df_balanced['text'].values
y_train_balanced = df_balanced['label'].values
# Create a new balanced dataset
train_dataset_balanced = ToxicDataset(X_train_balanced, y_train_balanced)
train_loader_balanced = DataLoader(train_dataset_balanced, batch_size=32, shuffle=True, num_workers=4, pin_memory=True)
Python
/usr/local/lib/python3.11/dist-packages/torch/utils/data/dataloader.py:624: UserWarning: This DataLoader will create 4 worker pro
warnings.warn(
train_labels = [sample['label'] for sample in train_dataset_balanced]
PROBLEMS 24 OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER POSTMAN CONSOLE
powershell
PS C:\Users\khaye\Desktop\Machine Learning>
Ln 1, Col 22 Spaces: 4 Spaces: 4 CRLF Cell 24 of 28 Ninja Prettier
Search
РУС
21:23
14/04/2025

Рис 4. Oversampling



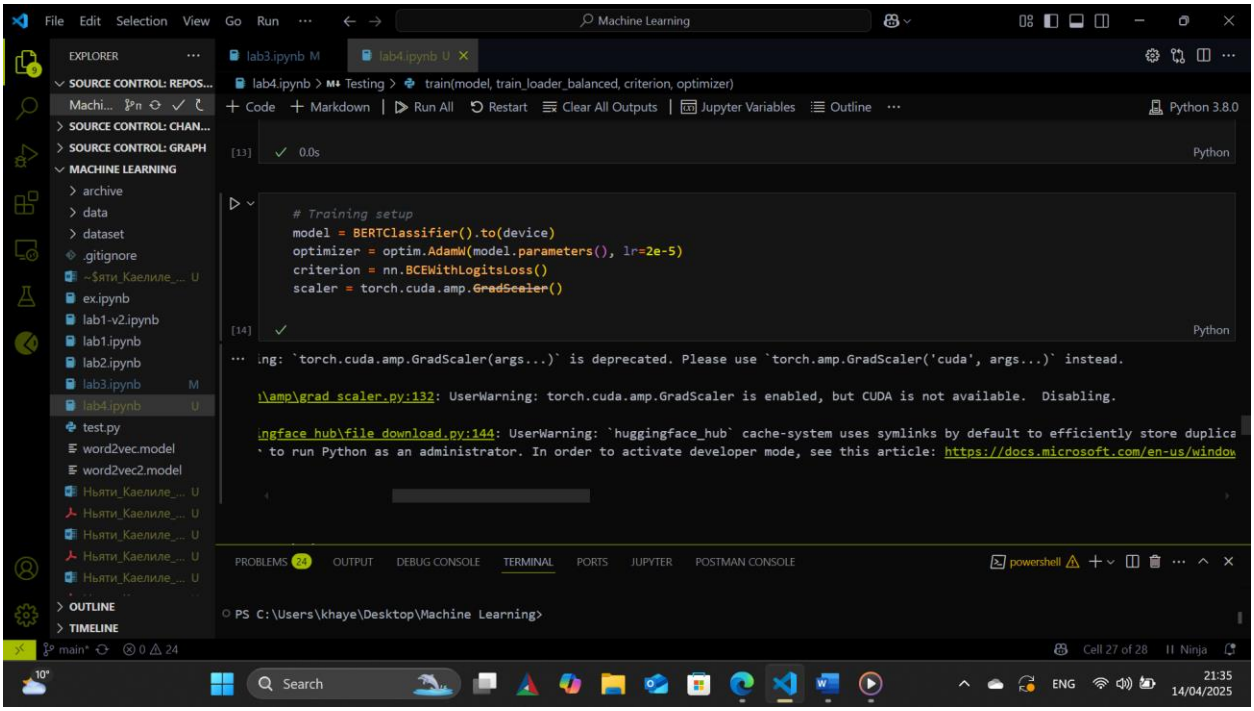


Рис 5. BERT model

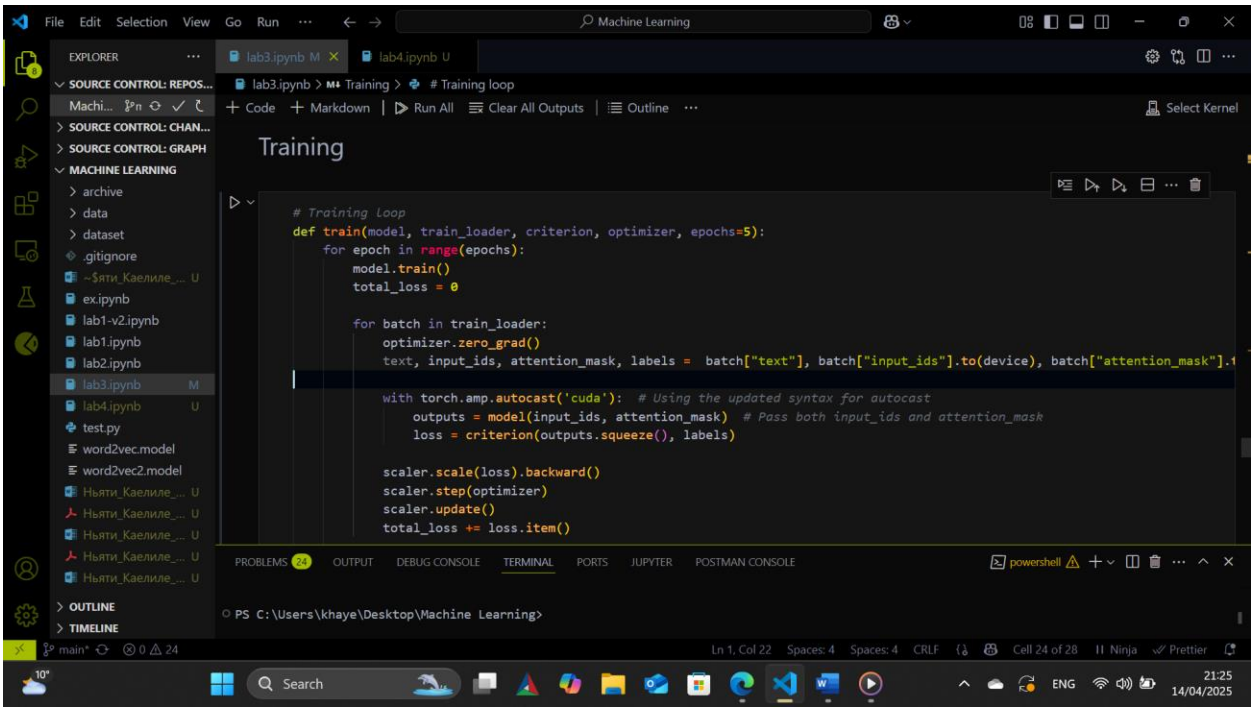
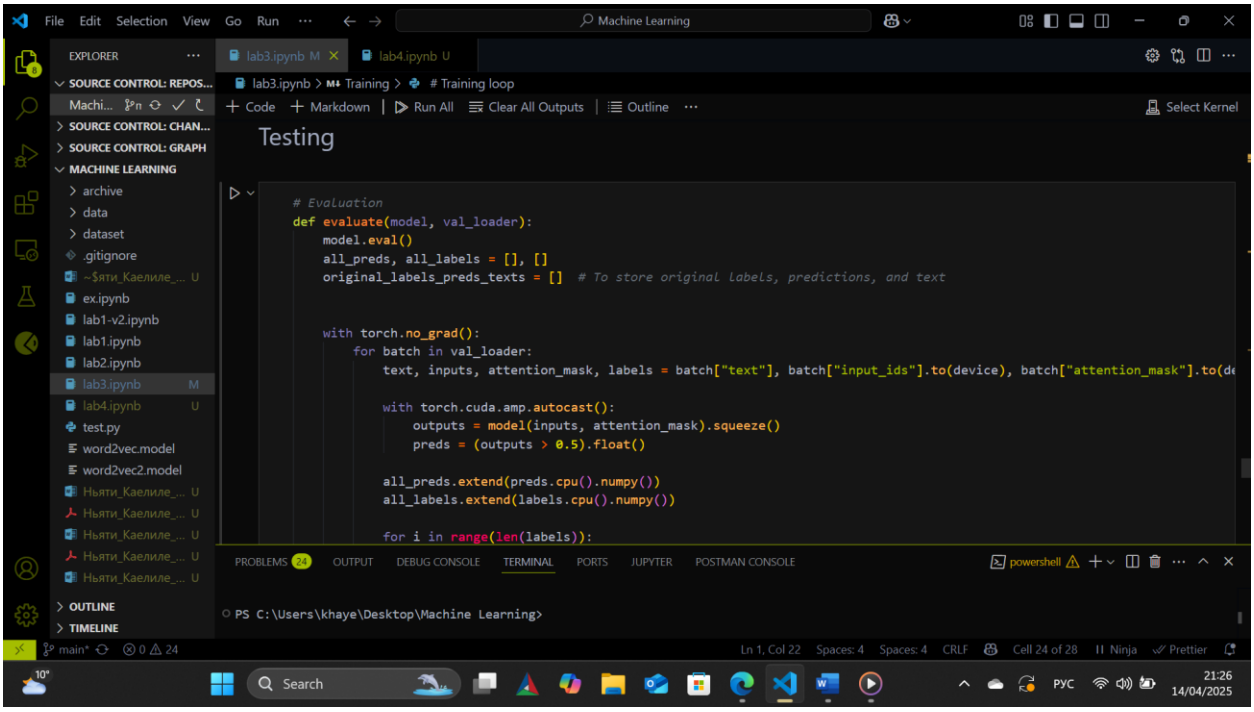
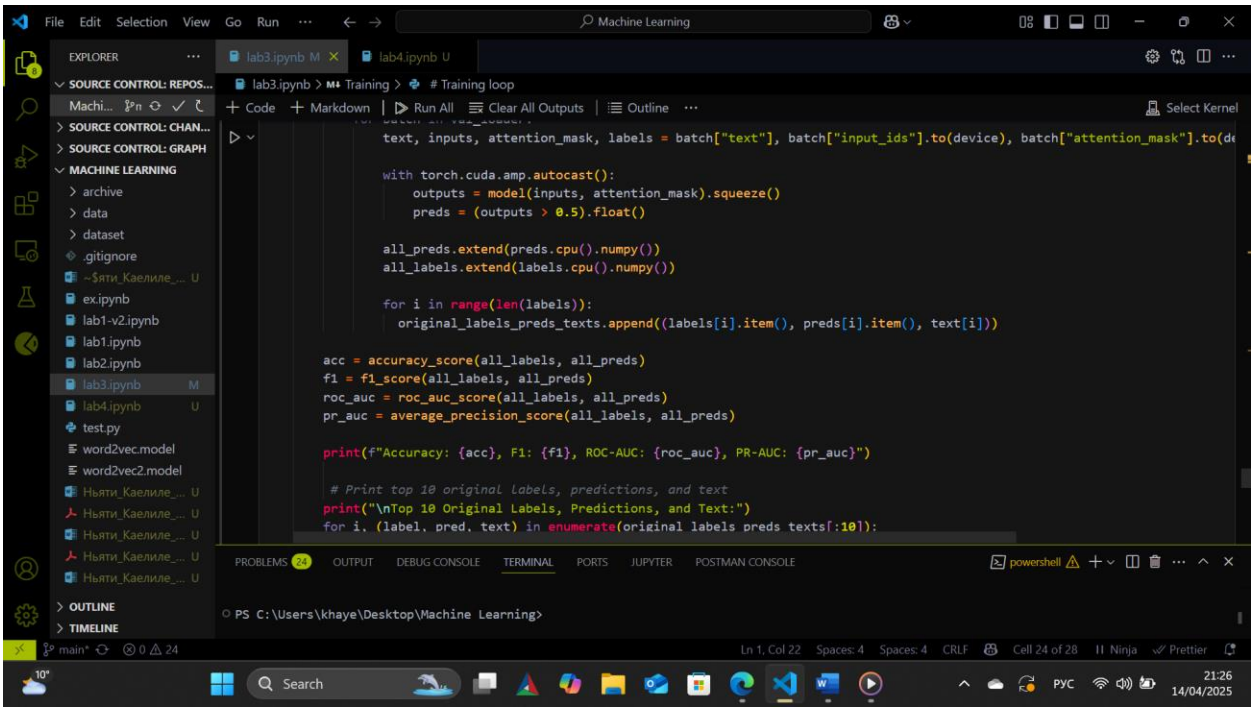
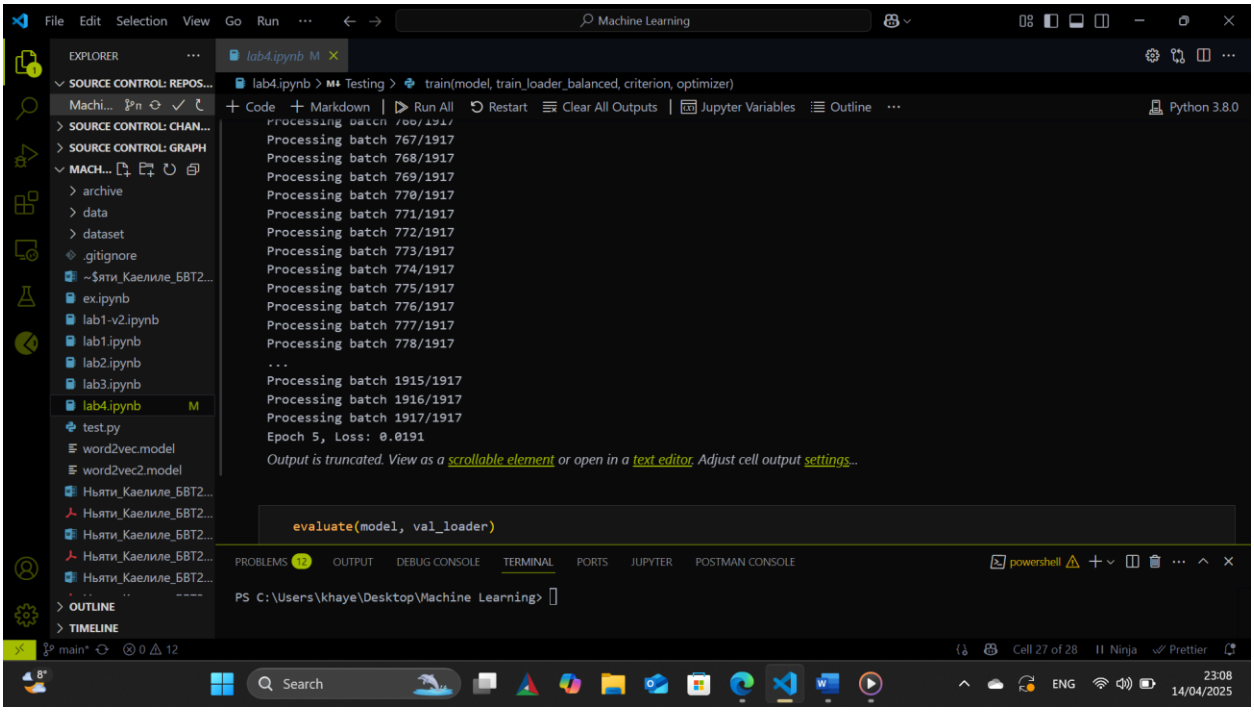


Рис 6. Тренирование




File Edit Selection View Go Run
Machine Learning
lab3.ipynb M
lab4.ipynb U
lab3.ipynb > Training > # Training loop
Code Markdown Run All Clear All Outputs Outline
Select Kernel
text, inputs, attention_mask, labels = batch["text"], batch["input_ids"].to(device), batch["attention_mask"].to(de
with torch.cuda.amp.autocast():
outputs = model(inputs, attention_mask).squeeze()
preds = (outputs > 0.5).float()
all_preds.extend(preds.cpu().numpy())
all_labels.extend(labels.cpu().numpy())
for i in range(len(labels)):
original_labels_preds_texts.append((labels[i].item(), preds[i].item(), text[i]))
acc = accuracy_score(all_labels, all_preds)
f1 = f1_score(all_labels, all_preds)
roc_auc = roc_auc_score(all_labels, all_preds)
pr_auc = average_precision_score(all_labels, all_preds)
print(f"Accuracy: {acc}, F1: {f1}, ROC-AUC: {roc_auc}, PR-AUC: {pr_auc}")
# Print top 10 original labels, predictions, and text
print("\nTop 10 Original Labels, Predictions, and Text:")
for i, (label, pred, text) in enumerate(original_labels_preds_texts[:10]):
PROBLEMS 24 OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER POSTMAN CONSOLE
powershell
PS C:\Users\khaye\Desktop\Machine Learning>
Search
РУС
21:26
14/04/2025


File Edit Selection View Go Run
Machine Learning
lab4.ipynb M
lab4.ipynb > Testing > train(model, train_loader_balanced, criterion, optimizer)
Code Markdown Run All Restart Clear All Outputs Jupyter Variables Outline
Python 3.8.0
Processing batch 766/1917
Processing batch 767/1917
Processing batch 768/1917
Processing batch 769/1917
Processing batch 770/1917
Processing batch 771/1917
Processing batch 772/1917
Processing batch 773/1917
Processing batch 774/1917
Processing batch 775/1917
Processing batch 776/1917
Processing batch 777/1917
Processing batch 778/1917
...
Processing batch 1915/1917
Processing batch 1916/1917
Processing batch 1917/1917
Epoch 5, Loss: 0.0191
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
evaluate(model, val_loader)
PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER POSTMAN CONSOLE
powershell
PS C:\Users\khaye\Desktop\Machine Learning>
Search
ENG
23:08
14/04/2025

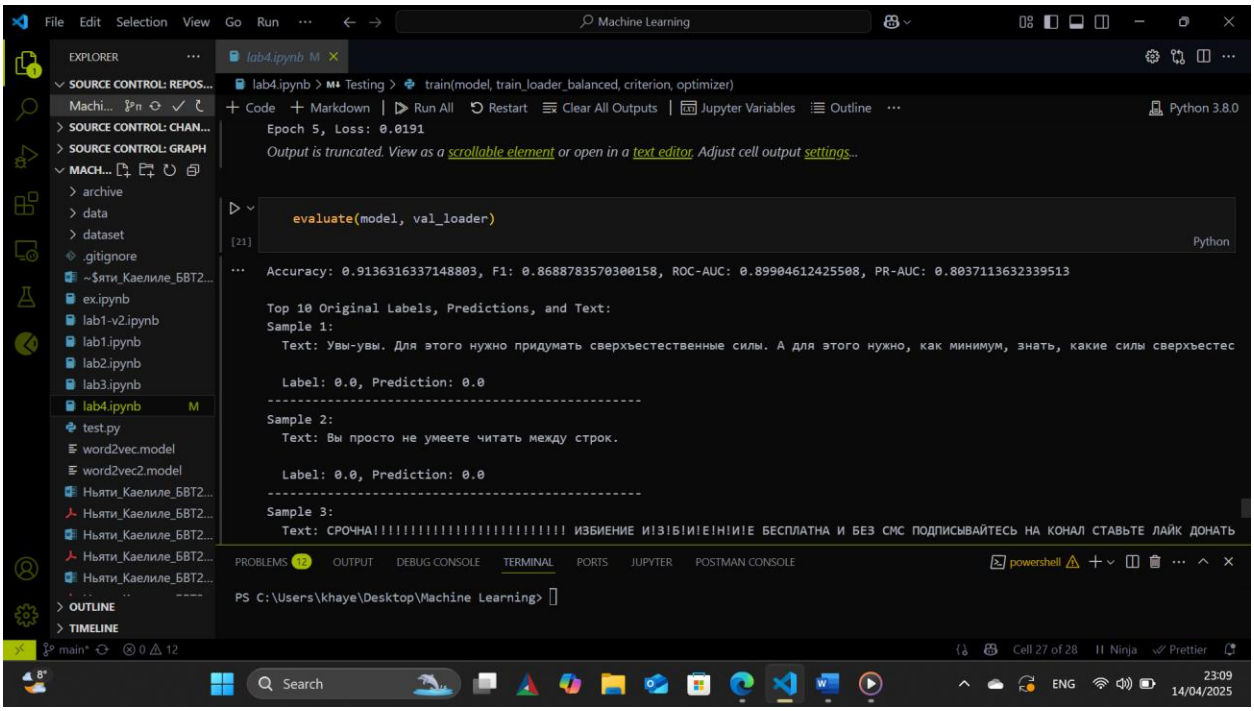


Рис 7. Результаты

3. Выводы

В рамках данной задачи была разработана модель на основе BERT для бинарной классификации текста на токсичный и нетоксичный. Благодаря мощным контекстным эмбеддингам BERT и использованию токенизатора из библиотеки Hugging Face, модель получила возможность учитывать семантику и структуру языка при анализе текста. Для борьбы с дисбалансом классов была применена взвешенная функция потерь Binary Cross-Entropy, а оптимизация производилась с помощью AdamW с learning rate 3e-4.

Обучение проводилось на GPU (CUDA), что позволило эффективно справляться с вычислительной нагрузкой. Модель оценивалась по метрикам accuracy, F1-score, ROC-AUC и PR-AUC, что обеспечило объективную оценку качества работы модели в условиях неравномерного распределения классов.

Полученные результаты показали высокую эффективность подхода на основе BERT в задаче определения токсичности текста и подтвердили его применимость в реальных сценариях автоматической модерации и фильтрации пользовательского контента.