Приложение статьи.

Часть сервера:

```
from fastapi import FastAPI
import pickle
from pydantic import BaseModel
import pandas as pd
import numpy as np
import string
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import pymorphy3



app = FastAPI()

with open('model_lr.pkl', 'rb') as file:
    model = pickle.load(file)

with open('vectorizer1.pkl', 'rb') as file:
    vectorizer = pickle.load(file)

def fun_punctuation_text(text):
    text = text.lower()
    text = ''.join([ch for ch in text if ch not in string.punctuation])
    text = ''.join([i if not i.isdigit() else '' for i in text])
    text = ''.join([i if i.isalpha() else ' ' for i in text])
    text = re.sub(r'\s+', ' ', text, flags=re.I)
    text = re.sub('[a-z]', '', text, flags=re.I)
    st = '❯\xa0'
    text = ''.join([ch if ch not in st else ' ' for ch in text])
    return text


def fun_lemmatizing_text(text):
    tokens = word_tokenize(text)
    res = list()
    for word in tokens:
        p = pymorphy3.MorphAnalyzer(lang='ru').parse(word)[0]
        res.append(p.normal_form)
    text = " ".join(res)
    return text


def fun_tokenize(text):
    nltk.download('stopwords')
    russian_stopwords = nltk.corpus.stopwords.words('russian')
    russian_stopwords.extend(['и', 'в', 'во', 'не', 'что', 'он', 'на', 'я',
'с', 'со', 'как', 'а', 'то', 'все',
                              'она', 'так', 'его', 'но', 'да', 'ты', 'к',
'у', 'же', 'вы', 'за', 'бы', 'по', 'только',
                              'ее', 'мне', 'было', 'вот', 'от', 'меня',
'еще', 'нет', 'о', 'из', 'ему', 'теперь',
                              'когда', 'даже', 'ну', 'вдруг', 'ли', 'если',
'уже', 'или', 'ни', 'быть', 'был', 'него',
                              'до', 'вас', 'нибудь', 'опять', 'уж', 'вам',
'ведь', 'там', 'потом', 'себя', 'ничего', 'ей',
                              'может', 'они', 'тут', 'где', 'есть', 'надо',
```

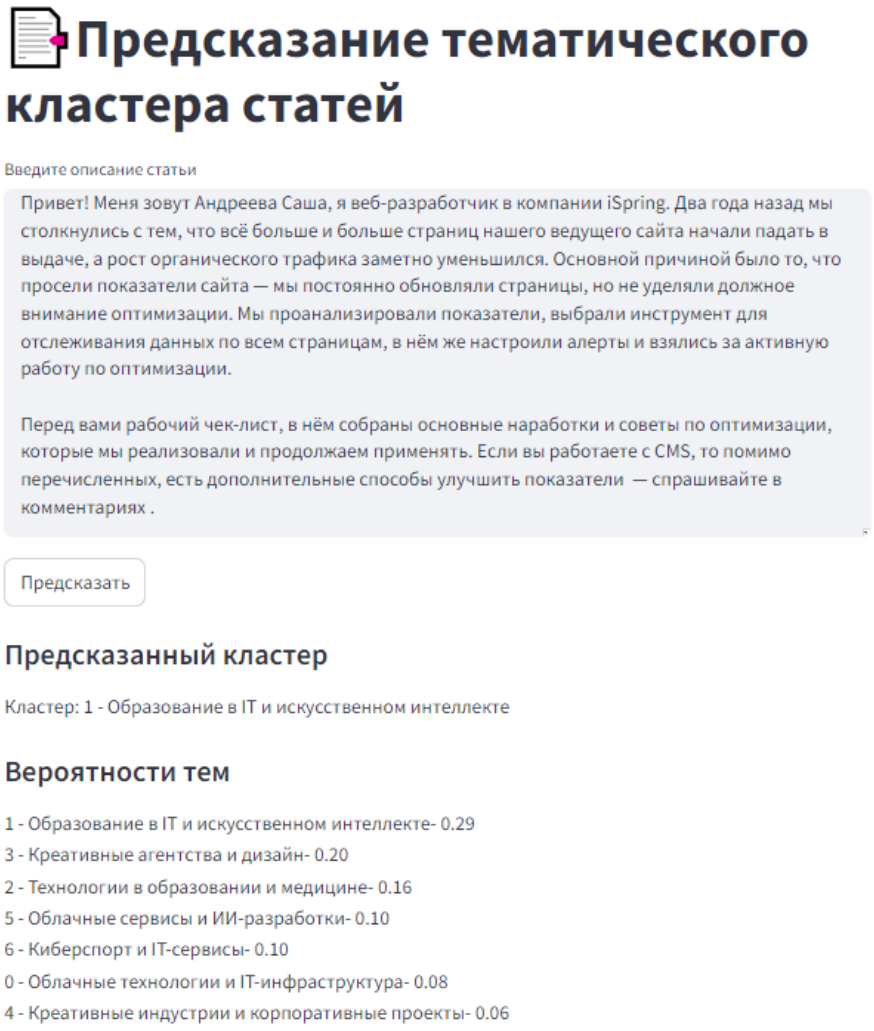
```
'ней', 'для', 'мы', 'тебя', 'их', 'чем', 'была',
                                'сам', 'чтоб', 'без', 'будто', 'чего', 'раз',
'тоже', 'себе', 'под', 'будет', 'ж', 'тогда',
                                'кто', 'этот', 'того', 'потому', 'этого',
'какой', 'совсем', 'ним', 'здесь', 'этом', 'один',
                                'почти', 'мой', 'тем', 'чтобы', 'нее',
'сейчас', 'были', 'куда', 'зачем', 'всех', 'никогда',
                                'можно', 'при', 'наконец', 'два', 'об',
'другой', 'хоть', 'после', 'над', 'больше', 'тот',
                                'через', 'эти', 'нас', 'про', 'всего', 'них',
'какая', 'много', 'разве', 'три', 'эту', 'моя',
                                'впрочем', 'хорошо', 'свою', 'этой', 'перед',
'иногда', 'лучше', 'чуть', 'том', 'нельзя', 'такой',
                                'им', 'более', 'всегда', 'конечно', 'всю',
'между'])
    t = word_tokenize(text)
    tokens = [token for token in t if token not in russian_stopwords]
    text = " ".join(tokens)
    return text

def fun_pred_text(text):
    text = fun_punctuation_text(text)
    text = fun_lemmatizing_text(text)
    text = fun_tokenize(text)
    return text




def predict_cluster(text):
    cluster_description = {
        0: "0 - Облачные технологии и IT-инфраструктура",
        1: "1 - Образование в IT и искусственном интеллекте",
        2: "2 - Технологии в образовании и медицине",
        3: "3 - Креативные агентства и дизайн",
        4: "4 - Креативные индустрии и корпоративные проекты",
        5: "5 - Облачные сервисы и ИИ-разработки",
        6: "6 - Киберспорт и IT-сервисы"
    }

    processed_text = fun_pred_text(text)
    text_vectorized = vectorizer.transform([processed_text])

    cluster_prediction = model.predict(text_vectorized)[0]
    probabilities = model.predict_proba(text_vectorized)[0]
    main_cluster = cluster_description[cluster_prediction]

    probability_lines = [
        f"{cluster_description[cluster_idx]}- {prob:.2f}"
        for prob, cluster_idx in sorted(
            zip(probabilities, range(len(probabilities))),
            key=lambda x: -x[0]
        )
    ]

    return main_cluster, "\n".join(probability_lines)


class Item(BaseModel):
    text: str

@app.post("/predict")
def post_pred_text(item: Item):
    return {'cluster': predict_cluster(item.text)}
```

Часть клиента:

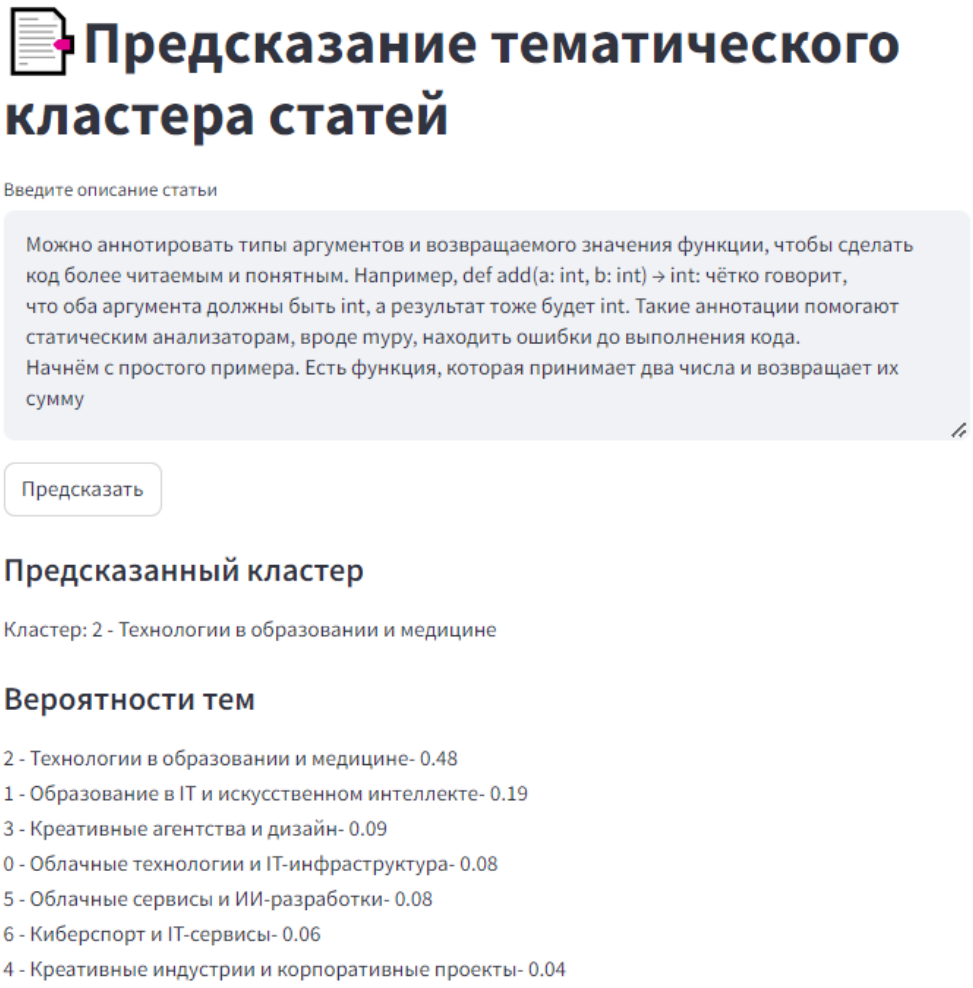
```
import streamlit as st
import requests
import os

st.set_page_config(
    page_title="Предсказание вероятности тем статей",
    page_icon="",
)

os.environ['HTTP_PROXY'] = ''
os.environ['HTTPS_PROXY'] = ''

st.title("Предсказание тематического кластера статей")
input_text = st.text_area("Введите описание статьи", height=200)

if st.button("Предсказать"):
    if input_text == "":
        st.write("Введите описание фильма")
    else:
        with st.spinner("Анализируем описание..."):
            data = {"text": input_text}
            url = "http://127.0.0.1:8000/predict"
            response = requests.post(url, json=data)
            result = response.json()
            clust = result.get("cluster")

            st.markdown(f"""
            #### Предсказанный кластер
            """)
            st.write(f"Кластер: {clust[0]}")

            st.markdown("""
            #### Вероятности тем
            """)
            st.text(clust[1])
```

Результат:

Статья с сайта хабр:



Статья из Pdf файла:



Статья из Json файла: