Министерство образования и науки Республики Башкортостан

Государственное бюджетное профессиональное образовательное учреждение Уфимский колледж статистики, информатики и вычислительной техники

**Программная документация**

**Автор**: Никифорова Карина,

студентка группы 21П-1

Уфа — 2024

Программа разработана на языке программирования Python. Библиотека для API – fastapi. Интерфейс написан с помощью библиотеки streamlit.

1. Серверная часть – main.py
   1. Используемые библиотеки

from fastapi import FastAPI  
from fastapi.middleware.cors import CORSMiddleware  
import pickle  
import numpy as np  
import pandas as pd  
from arch import arch\_model  
from pydantic import BaseModel  
from datetime import timedelta

* 1. Разработанные функций для предсказания уровня опасности и добавление информаций в датасет

def check\_dangerous(calendar\_information):  
 clf\_predict\_ = model.predict(calendar\_information)  
 calendar\_information['clf\_predict'] = clf\_predict\_  
 return calendar\_information  
  
def preprocess\_calendar\_information(calendar\_information):  
 calendar\_information2 = calendar\_information.drop(columns=['location', 'date'])  
 calendar\_information2 = check\_dangerous(calendar\_information2)  
 calendar\_information['clf\_predict'] = calendar\_information2['clf\_predict']  
 return calendar\_information

* 1. Разработанная функция передаёт данные о заражений в стране за определённый период

def get\_calendar\_information(country\_name, from\_date, before\_date):  
 x = df[['location', 'date', 'total\_cases', 'new\_cases', 'new\_cases\_smoothed', 'total\_deaths', 'new\_deaths',  
 'new\_deaths\_smoothed',  
 'total\_cases\_per\_million', 'new\_cases\_per\_million', 'new\_cases\_smoothed\_per\_million',  
 'total\_deaths\_per\_million',  
 'new\_deaths\_per\_million', 'new\_deaths\_smoothed\_per\_million', 'new\_vaccinations\_smoothed',  
 'new\_people\_vaccinated\_smoothed',  
 'population\_density', 'median\_age', 'gdp\_per\_capita', 'extreme\_poverty', 'female\_smokers', 'male\_smokers',  
 'handwashing\_facilities',  
 'hospital\_beds\_per\_thousand', 'human\_development\_index', 'population', 'RT']]  
 x['date'] = pd.to\_datetime(x['date'])  
 if pd.to\_datetime(from\_date) >= pd.to\_datetime('2024-02-04') or pd.to\_datetime(before\_date) >= pd.to\_datetime(  
 '2024-02-04'):  
 x = month\_prediction(country\_name)  
 calendar\_information = x[(x['location'] == country\_name) & (x['date'] >= pd.to\_datetime(from\_date)) &  
 (x['date'] <= pd.to\_datetime(before\_date))]  
 return calendar\_information

* 1. Разработанная функция возвращает уровень опасности нахождения в стране за определённый период

def prediction(country\_name, from\_date, before\_date):  
 calendar\_information = get\_calendar\_information(country\_name, from\_date, before\_date)  
 calendar\_information = preprocess\_calendar\_information(calendar\_information)  
 med = calendar\_information[(calendar\_information['date'] >= pd.to\_datetime(from\_date)) & (calendar\_information['date'] <= pd.to\_datetime(before\_date))]['clf\_predict'].median()  
 return str(round(med))

* 1. Функция для возврата дат с определённым уровнем опасности

# функция возвращает даты с выбранным уровнем опасности  
def get\_days\_hazard\_lvl(country\_name, from\_date, before\_date, hazard\_lvl):  
 # получаем df с нужной страной и переодом  
 calendar\_information = get\_calendar\_information(country\_name, from\_date, before\_date)  
 # добавляем столбец с предсказанными значениями уровня  
 calendar\_information = preprocess\_calendar\_information(calendar\_information)  
 # Фильтрация DataFrame по значению 'hazard\_lvl' и выбор столбца 'date'  
 filtered\_dates = calendar\_information[calendar\_information['clf\_predict'] == hazard\_lvl]['date']  
 # Преобразование дат в формат без времени  
 dates\_without\_time = filtered\_dates.dt.date.tolist()  
 return dates\_without\_time

* 1. Функций для предсказания данных

# предсказания данных на 3 месяца вперёд  
def month\_prediction(country\_name):  
 x = df[['location', 'date',  
 'total\_cases', 'new\_cases', 'new\_cases\_smoothed',  
 'total\_deaths', 'new\_deaths', 'new\_deaths\_smoothed',  
 'total\_cases\_per\_million', 'new\_cases\_per\_million', 'new\_cases\_smoothed\_per\_million',  
 'total\_deaths\_per\_million', 'new\_deaths\_per\_million', 'new\_deaths\_smoothed\_per\_million',  
 'new\_vaccinations\_smoothed', 'new\_people\_vaccinated\_smoothed',  
 'population\_density', 'median\_age', 'gdp\_per\_capita', 'extreme\_poverty', 'female\_smokers', 'male\_smokers',  
 'handwashing\_facilities', 'hospital\_beds\_per\_thousand', 'human\_development\_index', 'population', 'RT']]  
 x['date'] = pd.to\_datetime(x['date'])  
 x.set\_index('date', inplace=True) # устанавливаем индеком дату для обучения модели  
  
 # берём локацию которую выбрал пользователь  
 df\_loc = x[x['location'] == country\_name]  
 pred\_size = 120  
  
 # перебираем количество значений который нужно предсказать, обучаем модель и записываем предсказанные данные  
 for i in range(pred\_size):  
 # предсказание новых случаев  
 pred\_cases = pred\_garch(df\_loc['new\_cases\_smoothed'])  
 # предсказание новых смертей  
 pred\_deaths = pred\_garch(df\_loc['new\_deaths\_smoothed'])  
 # записываем новую запись в df  
 write\_to\_df(df\_loc, pred\_cases, pred\_deaths)  
  
 df\_loc.reset\_index(inplace=True)  
 return df\_loc  
  
  
# функция записывает новую запись в df  
def write\_to\_df(df, pred\_c, pred\_d):  
 last\_date = df.loc[df.index[-1]]  
 new\_case = round(np.sqrt(pred\_c.variance.values[-1, :][0]))  
 new\_death = round(np.sqrt(pred\_d.variance.values[-1, :][0]))  
 total\_cases = last\_date['total\_cases'] + new\_case  
 total\_deaths = last\_date['total\_deaths'] + new\_death  
 population = last\_date['population']  
  
 # записываем новую запись в df, индекс это последняя дата плюс одна  
 df.loc[pd.to\_datetime(df.index[-1] + timedelta(days=1))] = [  
 last\_date['location'],  
 total\_cases, new\_case, new\_case,  
 total\_deaths, new\_death, new\_death,  
 round(total\_cases / (population / 1000000), 4), round(new\_case / (population / 1000000), 4),  
 round(new\_case / (population / 1000000), 4),  
 round(total\_deaths / (population / 1000000), 4), round(new\_death / (population / 1000000), 4),  
 round(new\_death / (population / 1000000), 4),  
 last\_date['new\_vaccinations\_smoothed'], last\_date['new\_people\_vaccinated\_smoothed'],  
 last\_date['population\_density'], last\_date['median\_age'], last\_date['gdp\_per\_capita'],  
 last\_date['extreme\_poverty'], last\_date['female\_smokers'], last\_date['male\_smokers'],  
 last\_date['handwashing\_facilities'], last\_date['hospital\_beds\_per\_thousand'],  
 last\_date['human\_development\_index'], population, last\_date['RT']  
 ]  
  
  
# Предсказания моделью GARCH. Не возможно взять готовую и обученую модель, так как GARCH предсказывает один день и  
# должна заново обучаться с новыми данными.  
def pred\_garch(df\_train):  
 train = df\_train  
 model = arch\_model(train, p=1, q=1,  
 mean='constant', vol='GARCH', dist='normal')  
 model\_fit = model.fit(disp='off')  
 pred = model\_fit.forecast(horizon=1)  
 return pred

Класс для получения данных

class Item(BaseModel):  
 country\_name: str  
 from\_date: str  
 before\_date: str  
 hazard\_lvl: int

* 1. Методы получения данных API

@app.post("/get\_calendar\_information")  
def calendar\_information(item: Item):  
 return {'calendar\_information': get\_calendar\_information(item.country\_name, item.from\_date, item.before\_date)}  
  
@app.post("/predict\_dangerous\_country")  
def predict(item: Item):  
 return {'danger\_level': prediction(item.country\_name, item.from\_date, item.before\_date)}  
  
@app.post("/days\_hazard\_lvl")  
def days\_hazard\_lvl(item: Item):  
 return {'days': get\_days\_hazard\_lvl(item.country\_name, item.from\_date, item.before\_date, item.hazard\_lvl)}

1. Интерфейс – front.py
   1. Используемые библиотеки

from datetime import datetime  
import requests  
import streamlit as st  
import pandas as pd

* 1. Вывод «меню»

st.sidebar.title('Меню')  
menu\_selection = st.sidebar.radio('Выберите раздел:', ['Виджет', 'Справка'])  
  
if menu\_selection == 'Виджет':  
 st.title("Туристический виджет")

from\_date = datetime(2024, 3, 1)  
 before\_date = datetime(2024, 4, 30)

# создание полей ввода  
 st\_country = st.text\_input("Страна:", value='Morocco')  
 st\_from\_date = st.date\_input("Период от:", value=from\_date)  
 st\_before\_date = st.date\_input("Период до:", value=before\_date)  
 st\_hazard\_lvl = st.selectbox(  
 'Уровень опасности',  
 ('Все', 'Малый', 'Средний', 'Большой')  
 )  
  
 # собитые нажатия на кнопку  
 if st.button('Отобразить'):  
  
 if pd.to\_datetime(st\_from\_date) > pd.to\_datetime(st\_before\_date):  
 st.write('Первая дата должна быть меньше второй.')  
  
 elif (pd.to\_datetime(st\_from\_date) < pd.to\_datetime('2020-01-01') or pd.to\_datetime(st\_from\_date) > pd.to\_datetime('2024-06-03')) \  
 or (pd.to\_datetime(st\_before\_date) < pd.to\_datetime('2020-01-01') or pd.to\_datetime(st\_before\_date) > pd.to\_datetime('2024-06-03')):  
 st.write('Данные предоставлены от 2020-01-01 до 2024-06-03')  
  
 else:  
 try:  
 on\_button\_click()  
 except:  
 st.write('Ошибка')  
  
elif menu\_selection == 'Справка':  
 st.title('Справка')  
 show\_help()

* 1. Функция для вывода информаций в справку

def show\_help():  
 st.write("Добро пожаловать в справочное приложение!")  
 st.write("Это справка предоставляет информацию о его функциональности.")  
 st.write("С помощью данного виджета, вы можете просмотреть заражения covid-19 за определённый период "  
 "в какой либо стране. Введите название страны на английском, и нажмите кнопку \"Отобразить\","  
 " после данных действий загрузится светофор с отображением уровня опасности и график заражений за период. "  
 "Так же вы можете указать период, просто выберите дату в полях."  
 " ")  
 st.write("Цвета светофора: зелёный - безопасно, желтый - средняя опасность, красный - очень опасно.")  
 st.write("Данные предоставлены от 2020-01-01 до 2024-06-03.")

* 1. Функция нажатия на кнопку. Вывод информация о заражениях.

def on\_button\_click():  
 if st\_country is not None:  
 df = pd.DataFrame()  
 lvl = '0'  
 days = []  
 mapping = {'Все': -1, 'Малый': 0, 'Средний': 1, 'Большой': 2}  
 selected\_value\_numeric = mapping[st\_hazard\_lvl]  
  
 # запись данный из полей ввода в data  
 data = {  
 "country\_name": st\_country,  
 "from\_date": st\_from\_date.strftime('%Y-%m-%d'),  
 "before\_date": st\_before\_date.strftime('%Y-%m-%d'),  
 "hazard\_lvl": selected\_value\_numeric  
 }  
  
 # обращение к апи и получение данных заражений  
 url\_get\_calendar\_information = "http://127.0.0.1:8000/get\_calendar\_information"  
 response\_calendar = requests.post(url\_get\_calendar\_information, json=data)  
 result\_calendar = response\_calendar.json()  
 df = pd.DataFrame(result\_calendar.get("calendar\_information"))  
  
 # обращение к апи и получение опасности заражения  
 url\_dangerous = "http://127.0.0.1:8000/predict\_dangerous\_country"  
 response\_dangerous = requests.post(url\_dangerous, json=data)  
 result\_dangerous = response\_dangerous.json()  
 lvl = result\_dangerous.get("danger\_level")  
  
 # обращение к апи и получение дат с определённым уровнем заражения  
 if selected\_value\_numeric >= 0:  
 url\_days\_dangerous = "http://127.0.0.1:8000/days\_hazard\_lvl"  
 response\_days\_dangerous = requests.post(url\_days\_dangerous, json=data)  
 result\_days\_dangerous = response\_days\_dangerous.json()  
 days = (result\_days\_dangerous.get("days"))  
  
 st.write("""  
 #### Уровень опасности   
 """)  
  
 # в зависимости от уровня опасности выводим нужную картинку  
 path = ''  
 if lvl == '1':  
 path = 'draws/yellow.jpg'  
 elif lvl == '0':  
 path = 'draws/green.jpg'  
 elif lvl == '2':  
 path = 'draws/red.jpg'  
 st.image(path, caption='Уровень опасности за период', use\_column\_width=False, width=200)  
  
 st.markdown(f"""  
 #### Заражения в {st\_country} от {st\_from\_date} до {st\_before\_date}  
 """)  
  
 df = df[['date', 'new\_cases']]  
 df.set\_index('date', inplace=True)  
 df.index = pd.to\_datetime(df.index)  
  
 # Извлечение только даты из индекса  
 df.index = df.index.date  
 st.line\_chart(df['new\_cases'])  
  
 # вывод дат если пользователь указывал уровень опасности  
 if selected\_value\_numeric >= 0:  
 st.markdown(f"""  
 #### Даты заражения уровня \"{st\_hazard\_lvl}\" в {st\_country} от {st\_from\_date} до {st\_before\_date}  
 """)  
 if len(days) <= 0:  
 st.write("Данных такого уровня опасности нет.")  
 else:  
 dates\_text = '\n'.join(days)  
 st.text\_area("Даты:", value=dates\_text, height=200)  
  
 else:  
 st.write("Введите название страны.")

1. Файлы для работы приложения
   1. Модель классификация для прогнозирования уровня опасности - model.pkl.
   2. Данные о заражений covid-19 - result\_1.csv.
   3. Папка draws с картинками светофора.