

Проект по РВДНиПЗ за второй семестр

1. Цель работы

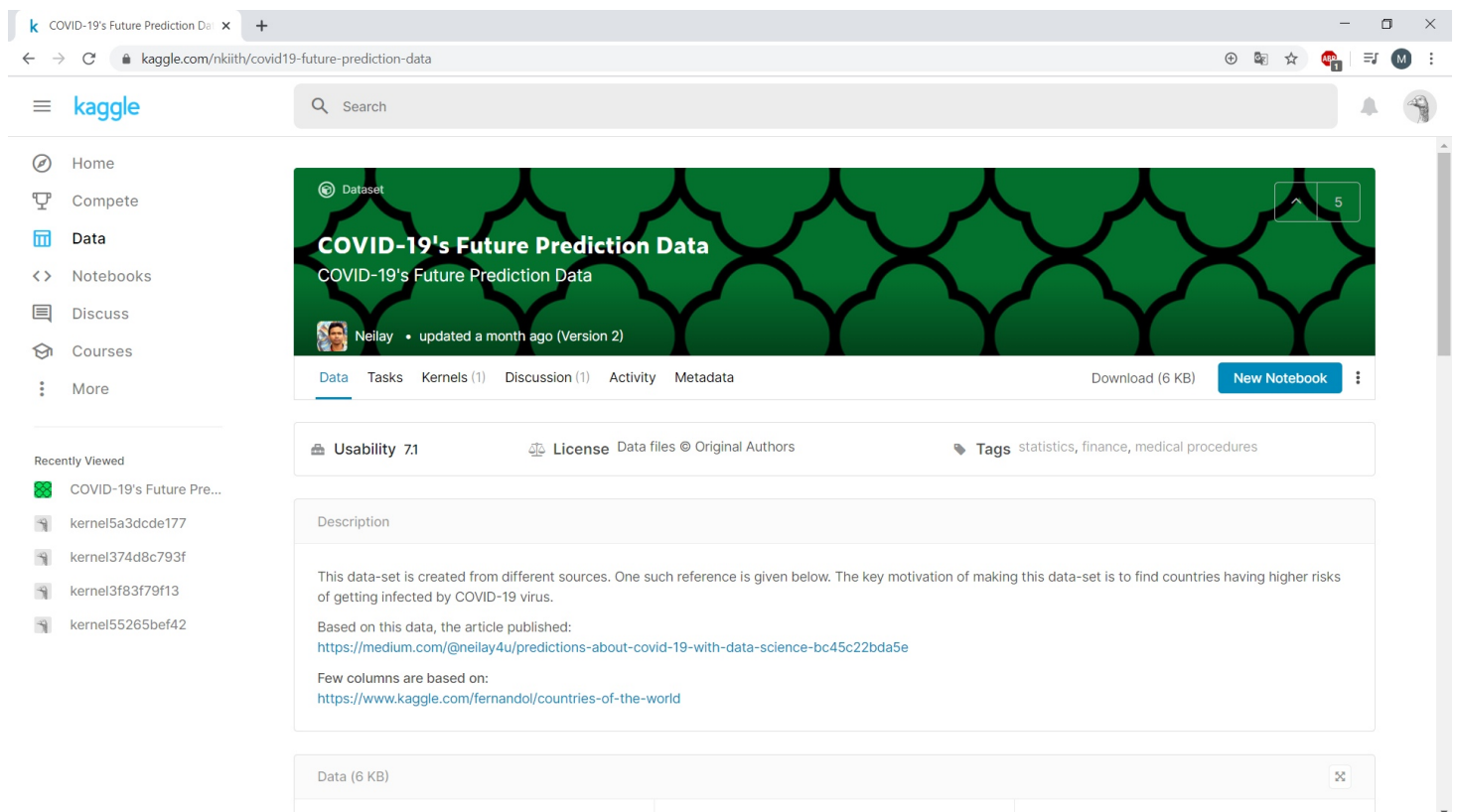
Лучше познакомиться с языком *python*, изучить фреймворк *Django*, библиотеки *numpy*, *pandas*. Узнать принципы работы с веб-приложениями.

2. Задание

Создать сайт, используя фреймворки *Django* или *Flask*, который будет выводить графики по выбранному датасету.

3. Ход работы

Первым шагом в создании проекта был выбор подходящего датасета на сайте Kaggle. Я решил остановиться на актуальной тогда(да и сейчас) теме - **COVID-19**.

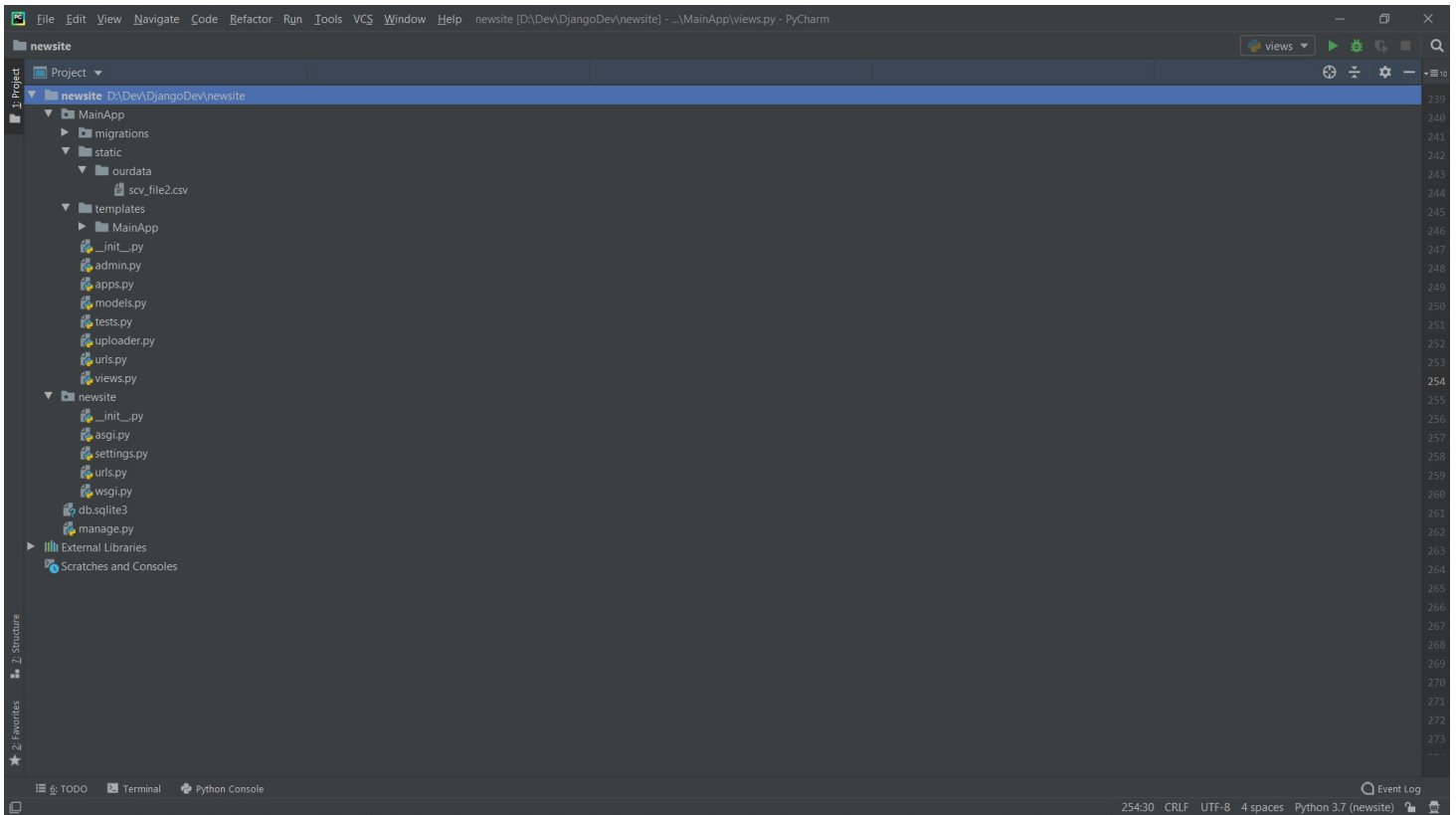


The screenshot shows the Kaggle website interface. The browser address bar displays 'kaggle.com/nkiith/covid19-future-prediction-data'. The Kaggle logo is in the top left, and a search bar is in the top center. On the left sidebar, navigation links include Home, Compete, Data, Notebooks, Discuss, Courses, and More. Below these, a 'Recently Viewed' section lists several datasets and kernels. The main content area features a green header for the 'COVID-19's Future Prediction Data' dataset by user 'Neilay', updated a month ago (Version 2). Below the header, there are tabs for Data, Tasks, Kernels (1), Discussion (1), Activity, and Metadata. A 'Download (6 KB)' button and a 'New Notebook' button are visible. Further down, the 'Usability' is 7.1, the license is 'Data files © Original Authors', and tags include 'statistics, finance, medical procedures'. The 'Description' section explains that the dataset is created from different sources and provides a link to an article published on Medium. It also lists a few columns based on a link to a Kaggle dataset.

К сожалению, датасет уже давно не обновлялся, так что придётся работать с сильно устаревшими данными.

Далее по списку нужно было создать и структурировать свой *Django* проект.

Финальный вид его получился таковым



Основной проблемой при создании сайта стала передача изображения графика на страницу рендера. Было несколько путей её разрешения, но самым эффективным мне показалось создание "холста" с помощью библиотеки *matplotlib* и переноса туда изображения, после чего оно отпечатывалось с холста на объект *response*, который, в свою очередь, зашпрашивался при рендере другой страницы. Таким образом одна функция *view* "возвращала" *html* страницу, а другая передавала на эту страницу изображение.

Вот пример подобных функций

```

def boxAge(response):
    fig = Figure()
    canvas = FigureCanvas(fig)
    ax = fig.add_subplot(111)
    data_df = pd.read_csv("MainApp/static/ourdata/scv_file2.csv")
    data_df = pd.DataFrame(data_df)
    data_df.boxplot(ax=ax, column='Median_Age', by='Severity')
    response = HttpResponse(content_type='image/jpg')
    canvas.print_jpg(response)
    return response

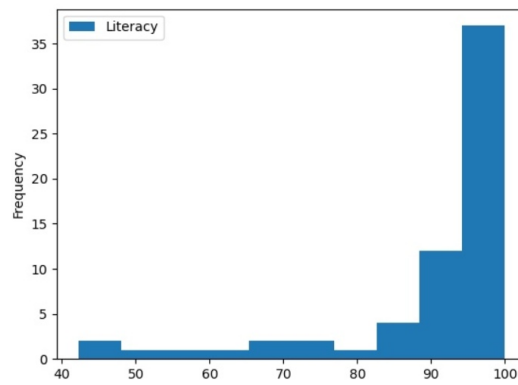
def boxAgeRender(response):
    return render(response, 'MainApp/boxAgeRender.html')

```

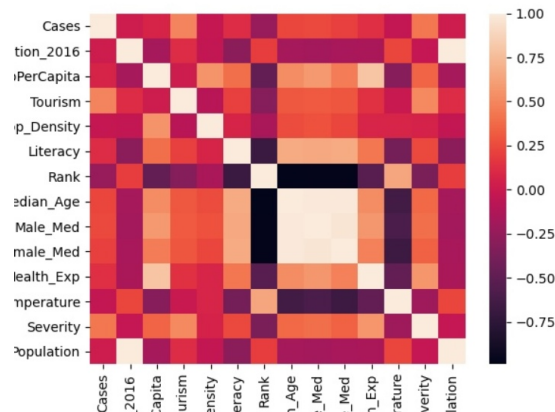
Проблема, однако, не была решена полностью, потому что такой подход создавал некоторые ограничения на возможные виды графиков. Так например функция *seaborn.pairplot* не может быть напечатана на используемом "холсте". Но я решил пожертвовать некоторым функционалом ради эффективности программы.(и стоит ещё отметить, что некоторые функции использовали слишком много вычислительной мощности, из-за чего программа могла запускаться около 5 минут)

В конечном счёте я решил остановиться на 4 разных типах графиков. Конечно, далеко не все они действительно информативны, но всё же и из них можно вычлениить полезную информацию и сделать определённые выводы. Почти все они ориентированы на соотношение какого-либо критерия к "*Severity index*". Это что-то вроде меры опасности вирусной угрозы.

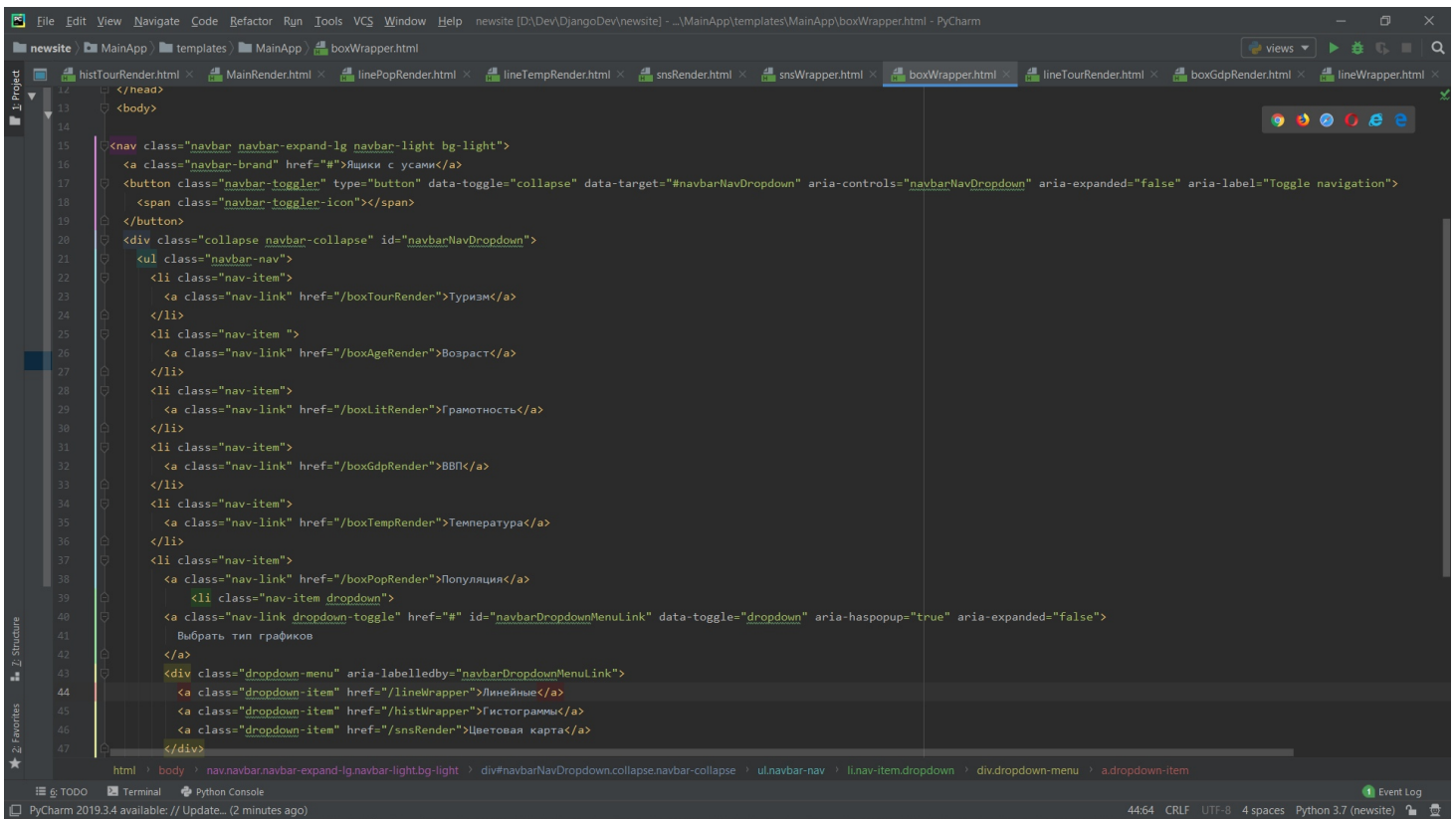
Также я старался пользоваться только языком *python*, не прибегая к *JavaScript*, что тоже наложило определённые ограничения. Однако он всё же встроен в *Bootstrap*, который я использовал для удобоваримого интерфейса на странице браузера.



- Ящики с усами
- Линейные
- Гистограммы



Внешнее оформление сайта (меню, позиционирование графиков) было написано с помощью вышеупомянутого фреймворка *Bootstrap*, используя стандартные средства. Так выглядит стандартная страница, отображающая график и меню выбора.



```
12 </head>
13 <body>
14
15 <nav class="navbar navbar-expand-lg navbar-light bg-light">
16   <a class="navbar-brand" href="#">Ящерицы с усами</a>
17   <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavDropdown" aria-controls="navbarNavDropdown" aria-expanded="false" aria-label="Toggle navigation">
18     <span class="navbar-toggler-icon"></span>
19   </button>
20   <div class="collapse navbar-collapse" id="navbarNavDropdown">
21     <ul class="navbar-nav">
22       <li class="nav-item">
23         <a class="nav-link" href="/boxTourRender">Туризм</a>
24       </li>
25       <li class="nav-item">
26         <a class="nav-link" href="/boxAgeRender">Возраст</a>
27       </li>
28       <li class="nav-item">
29         <a class="nav-link" href="/boxLitRender">Грамотность</a>
30       </li>
31       <li class="nav-item">
32         <a class="nav-link" href="/boxGdpRender">ВВП</a>
33       </li>
34       <li class="nav-item">
35         <a class="nav-link" href="/boxTempRender">Температура</a>
36       </li>
37       <li class="nav-item">
38         <a class="nav-link" href="/boxPopRender">Популяция</a>
39       </li>
40       <li class="nav-item dropdown">
41         <a class="nav-link dropdown-toggle" href="#" id="navbarDropdownMenuLink" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
42           Выбрать тип графиков
43         </a>
44         <div class="dropdown-menu" aria-labelledby="navbarDropdownMenuLink">
45           <a class="dropdown-item" href="/lineWrapper">Линейные</a>
46           <a class="dropdown-item" href="/histWrapper">Гистограммы</a>
47           <a class="dropdown-item" href="/snsRender">Цветовая карта</a>
48         </div>
49       </li>
50     </ul>
51   </div>
52 </div>
```

В целом, остальная структура проекта не представляет собой чего-то особенного и довольно тривиальна, так что не вижу смысла объяснять принцип её работы.

4. Вывод

Во время работы я изучил довольно много информации, касающейся вышеприведённых средств разработки и веб-программирования в целом. Однако, к сожалению, далеко не все эти новые знания я задействовал в проекте в силу их бесполезности в данном контексте или сложности реализации. В целом, данный проект познакомил меня с миром веб-программирования открыл новую область знаний, на изучение которой я несомненно потрачу ещё много своего времени в будущем.