Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Системы обработки информации и управления

ОТЧЕТ

К ДОМАШНЕМУ ЗАДАНИЮ № 3

По дисциплине «Методы поддержки принятия решений»

МЕТОДЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ НА ОСНОВЕ ЭКСПЕРТНЫХ СИСТЕМ И НЕЧЕТКИХ МНОЖЕСТВ

Студент ИУ5-73б _____ Д.К. Пермяков
(Группа) (Подпись, дата) (И.О.Фамилия)

Преподаватель _____ А.А. Коценко
(Подпись, дата) (И.О.Фамилия)

Оценка _____

Москва – 2024

Цель

Научиться проектировать веб-приложение с помощью python django, использующее нейросеть для классификации изображений.

Задание

Необходимо создать веб-приложение для классификации изображений с использованием предобученной модели на основе датасета cifar100.

Выполнение

Скачал предобученную модель:

```
ИУ5-73Б
Вариант 13

Модель – cifar100_resnet
Класс 1 – номер в группе + номер группы      = 16
Класс 2 – номер в группе + номер группы + 30 = 46
Класс 3 – номер в группе + номер группы + 60 = 76

Где номер группы – 1, 2, 3, 4 и т.д.
```

```
[ ] pip install onnx
```

Показать скрытые выходные данные

```
[ ] import torch
```

```
[ ] device = torch.device('cuda'  if torch.cuda.is_available()  else  'cpu')
```

```
[ ] model = torch.hub.load("chenyaofo/pytorch-cifar-models",
      'cifar100_resnet20',
      pretrained=True)
```

Сохранил модель в формате ONNX:

```
import torch
```

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

```
model = torch.hub.load("chenyaofo/pytorch-cifar-models",
  'cifar100_resnet20',
  pretrained=True)
```

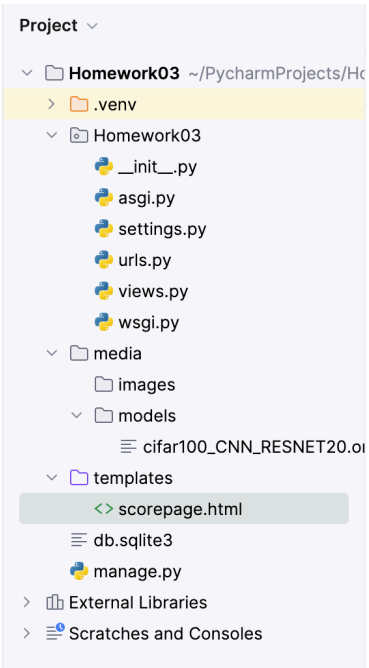
Показать скрытые выходные данные

```
model.to(device)
```

Показать скрытые выходные данные

```
x = torch.randn(1, 3, 32, 32, requires_grad=True).to(device)

torch.onnx.export(model, # модель
          x, # входной тензор (или кортеж нескольких тензоров)
          "cifar100_CNN_RESNET20.onnx", # куда сохранить (либо путь к файлу либо fileObject)
          export_params=True, # сохраняет веса обученных параметров внутри файла модели
          opset_version=9, # версия ONNX
          do_constant_folding=True, # следует ли выполнять укорачивание констант для оптимизации
          input_names = ['input'], # имя входного слоя
          output_names = ['output'], # имя выходного слоя
          dynamic_axes={'input' : {0 : 'batch_size'}, # динамичные оси, в данном случае только размер пакета
          'output' : {0 : 'batch_size'}})
```

Зашёл в PyCharm, создал папку Homework03, в ней создал виртуальное окружение и в нём создал проект django под названием Homework03. В проекте создал папку media, в которой есть 2 папки: images и models. В папку models загрузил нашу модель.



Указал в файле settings.py папку media:

```
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(os.path.dirname(os.path.dirname(os.path.abspath(__file__))), 'media')
```

Создал файл с представлениями views.py. В нём указал путь до модели и ограничил классы по варианту (это классы ‘camel, ‘lobster, ‘skunk):

```python
from django.shortcuts import render
from django.core.files.storage import FileSystemStorage
import onnxruntime
import numpy as np
from PIL import Image
from io import BytesIO
import base64
from torchvision import transforms

imageClassList = {15: 'camel', 45: 'lobster', 75: 'skunk'}


/  1 usage
def scoreImagePage(request):
    return render(request, template_name: 'scorepage.html')


/predictImage  1 usage
def predictImage(request):
    fileObj = request.FILES['filePath']
    fs = FileSystemStorage()
    filePathName = fs.save('images/' + fileObj.name, fileObj)
    filePathName = fs.url(filePathName)
    modelName = request.POST.get('modelName')
    scorePrediction, img_uri = predictImageData(modelName, '.' + filePathName)
    context = {'scorePrediction': scorePrediction, 'filePathName': filePathName, 'img_uri': img_uri}
    return render(request, template_name: 'scorepage.html', context)


1 usage
def predictImageData(modelName, filePath):...


1 usage
def to_numpy(tensor):
    return tensor.detach().cpu().numpy() if tensor.requires_grad else tensor.cpu().numpy()


def to_image(numpy_img):
    img = Image.fromarray(numpy_img, mode: 'RG')
    return img
```

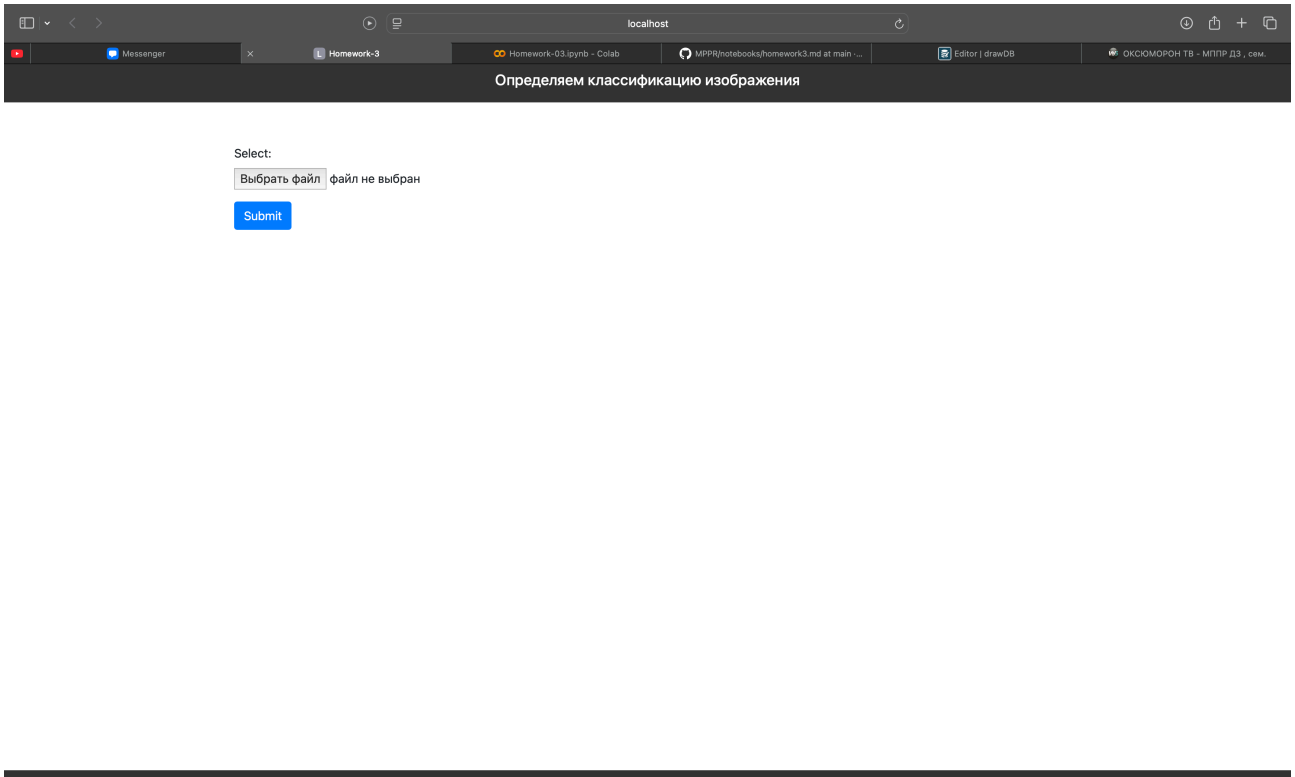
В файле urls.py настроил роутинг:

```python
/
from django.contrib import admin
from django.urls import path
from django.conf.urls.static import static
from django.conf import settings
from . import views

/
urlpatterns = [
    /admin/
    path('admin/', admin.site.urls),
    /
    path('', views.scoreImagePage, name='scoreImagePage'),
    /predictImage
    path('predictImage', views.predictImage, name='predictImage'),
]

urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

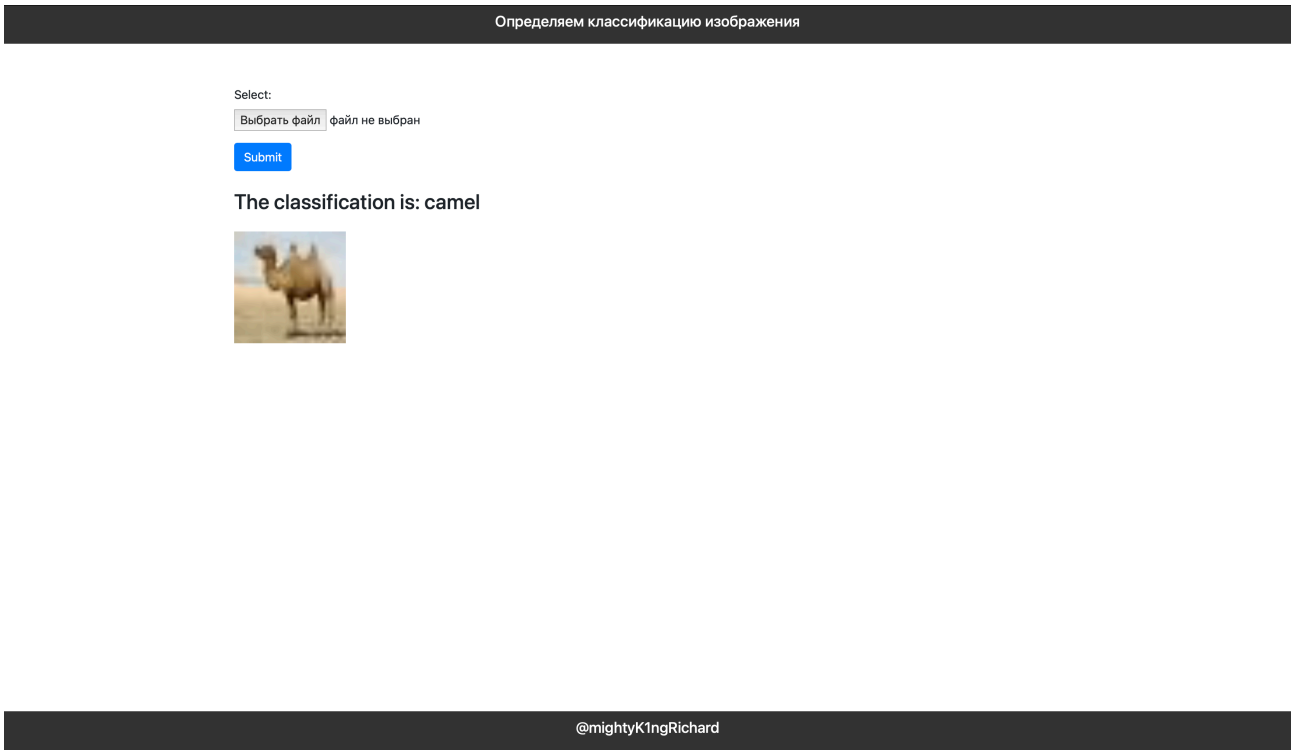
Создал папку templates, в ней файл scorepage.html с кодом отрисовки страницы:

```
<html lang="ru">
<head>
    <meta charset="utf-8">
    <title>Homework-3</title>
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
    <style...>
</head>
<body>

<div class="fixed-header">
    <div class="container text-center">
        <h5>Определяем классификацию изображения</h5>
    </div>
</div>

<div class="container mt-5">
    <form action="predictImage" method="post" enctype="multipart/form-data">
        {% csrf_token %}

        <div class="form-group">
            <label for="FilePath">Select:</label>
            <input name="filePath" type="file" class="form-control-file">
        </div>

        <button type="submit" class="btn btn-primary">Submit</button>
    </form>

    {% if scorePrediction %}
        <div class="mt-4">
            <h3>The classification is: {{ scorePrediction }}</h3>
            <img src="{{ img_uri }}" class="img-fluid mt-3">
        </div>
    {% endif %}
</div>

<div class="fixed-footer">
    <div class="container text-center">
        <h5>@mightyK1ngRichard</h5>
    </div>
</div>

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"></script>
```

Запустил проект:

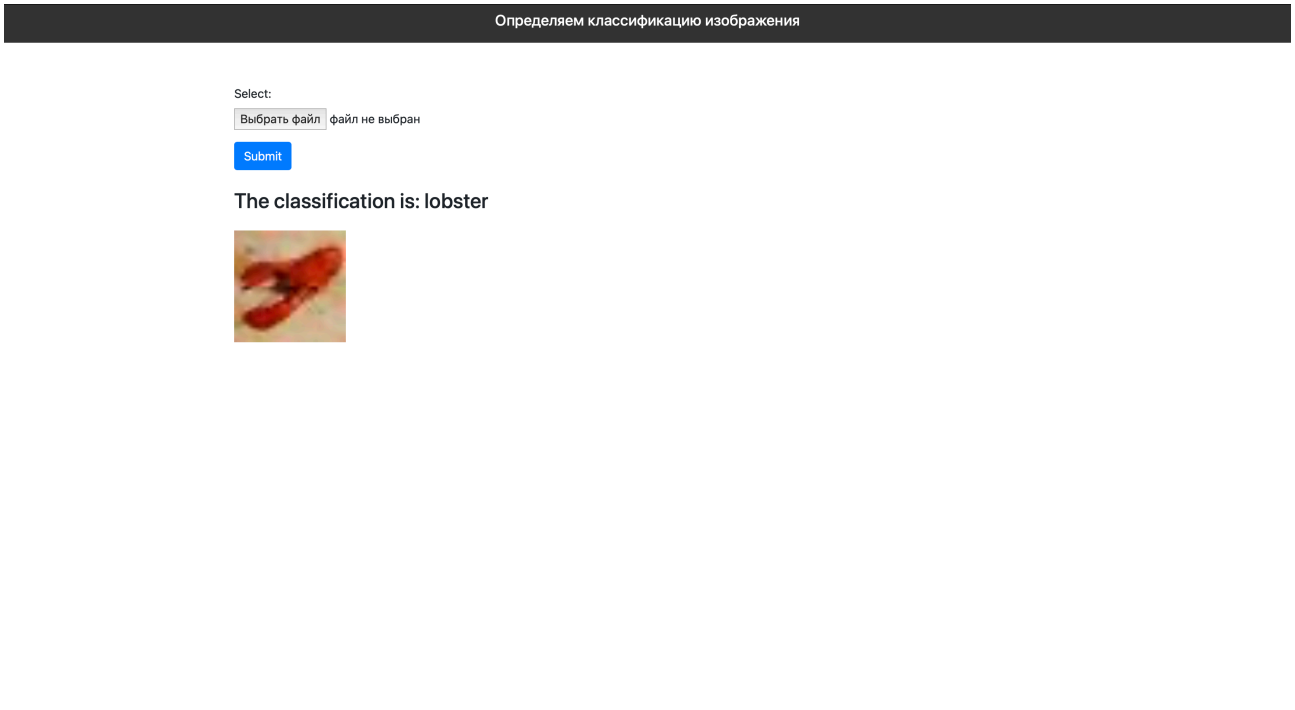


Загрузил картинку верблюда:

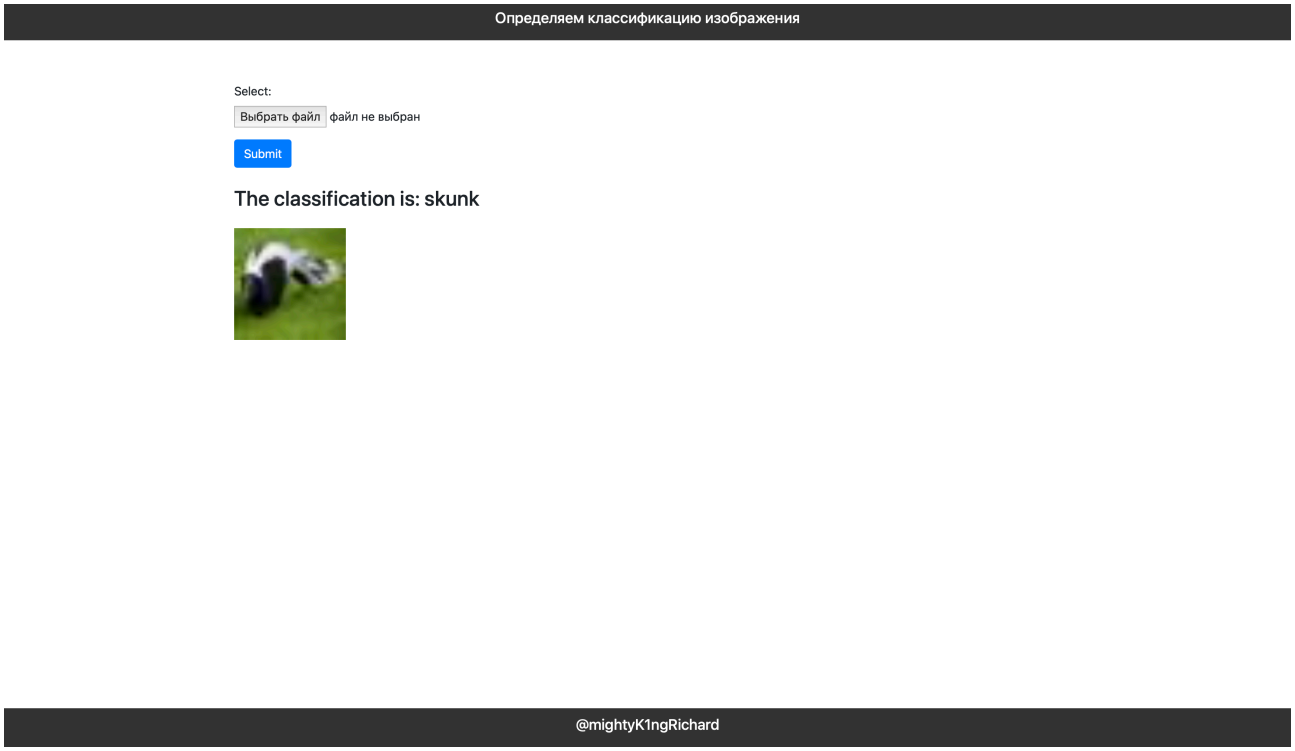


Модель верно классифицировала объект на картинке как верблюд.

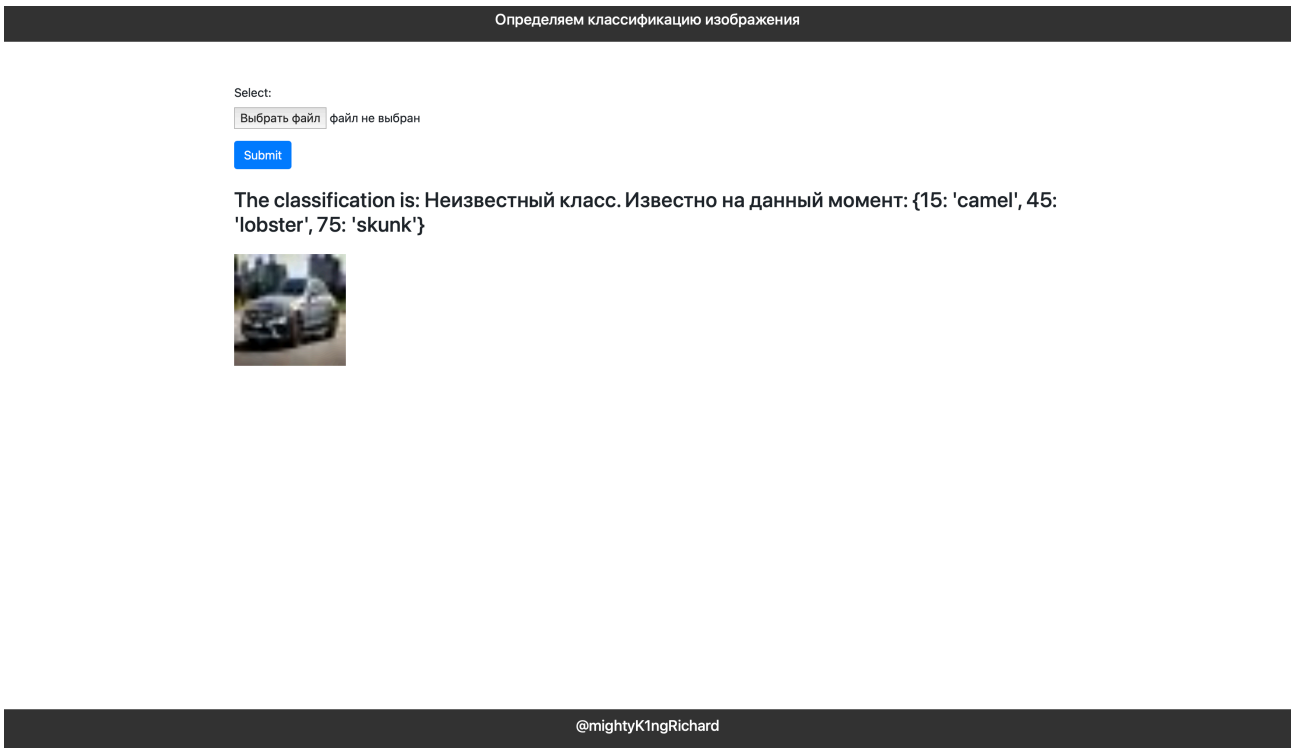
Попробую загрузить картинку с лобстером:



Верно. Теперь скунс:



Верно. Теперь попробую загрузить фото машины, которая не входит в перечень моих классов по варианту:



Написалось, что класс не известен. Значит, всё работает верно.

Выводы

В результате выполнения домашнего задания были получены навыки проектирования веб-приложения, использующего нейросеть.