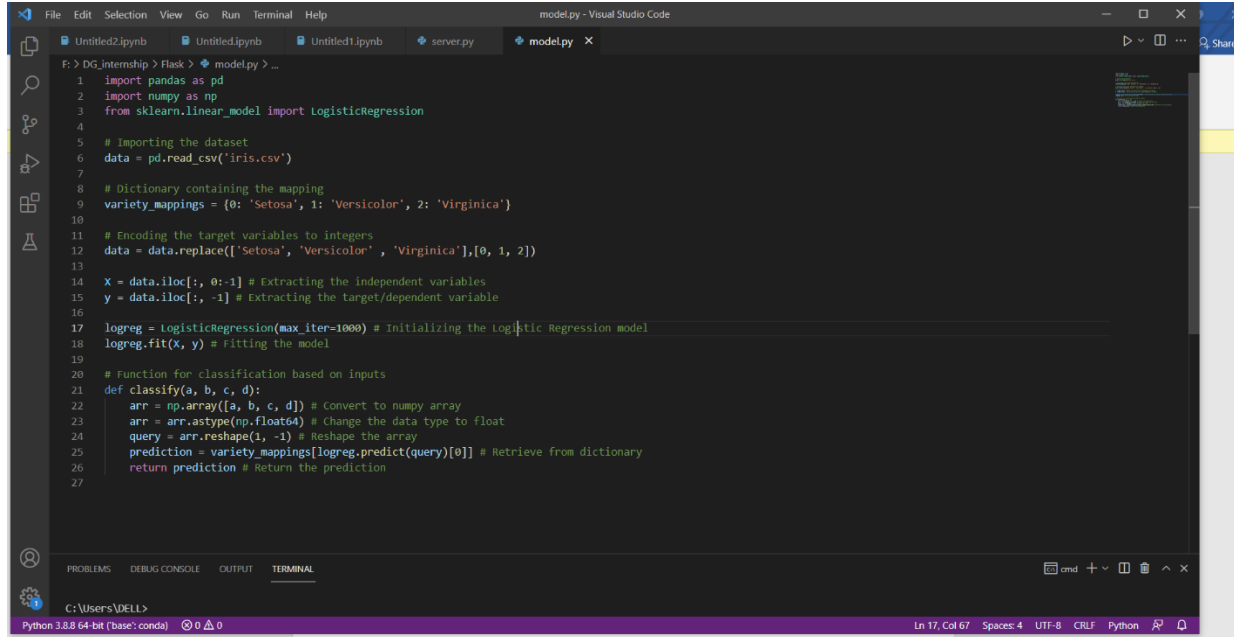


[illegible]

2- Model creation:

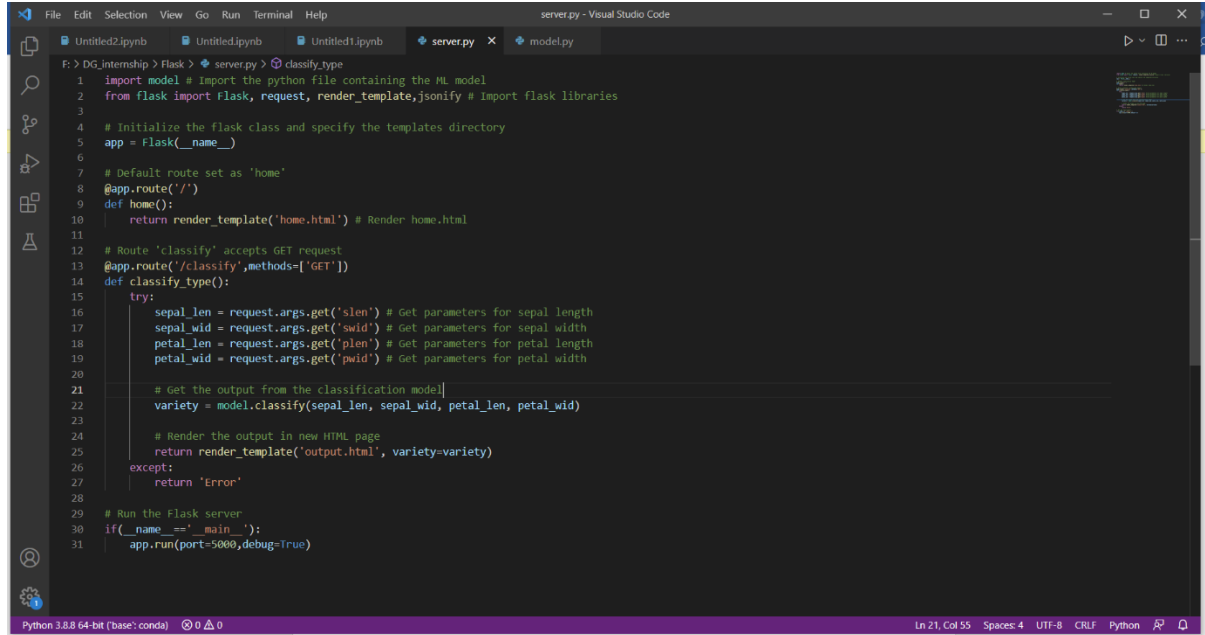


The image shows a Visual Studio Code editor window with a Python file named `model.py`. The code implements a logistic regression model for the Iris dataset. It includes imports for pandas, numpy, and sklearn's LogisticRegression. The dataset is loaded from `iris.csv`, and the target variable is encoded using a dictionary. The model is trained on the data, and a `classify` function is defined to make predictions on new input data.

```
F:\> DG_internship > Flask > model.py > ...
1 import pandas as pd
2 import numpy as np
3 from sklearn.linear_model import LogisticRegression
4
5 # importing the dataset
6 data = pd.read_csv('iris.csv')
7
8 # Dictionary containing the mapping
9 variety_mappings = {0: 'Setosa', 1: 'Versicolor', 2: 'Virginica'}
10
11 # Encoding the target variables to integers
12 data = data.replace(['Setosa', 'Versicolor', 'Virginica'], [0, 1, 2])
13
14 X = data.iloc[:, 0:-1] # Extracting the independent variables
15 y = data.iloc[:, -1] # Extracting the target/dependent variable
16
17 logreg = LogisticRegression(max_iter=1000) # Initializing the Logistic Regression model
18 logreg.fit(X, y) # fitting the model
19
20 # Function for classification based on inputs
21 def classify(a, b, c, d):
22     arr = np.array([a, b, c, d]) # Convert to numpy array
23     arr = arr.astype(np.float64) # Change the data type to float
24     query = arr.reshape(1, -1) # Reshape the array
25     prediction = variety_mappings[logreg.predict(query)[0]] # Retrieve from dictionary
26     return prediction # Return the prediction
27
```

The bottom status bar indicates the file is at line 17, column 67, with 4 spaces, UTF-8 encoding, CRLF line endings, and the Python interpreter is set to the base conda environment.

3-Flask deployment:



The image shows a Visual Studio Code editor window with a Python file named `server.py` open. The code is a Flask application designed for deployment. It includes imports for `Flask`, `request`, `render_template`, and `jsonify`. The application initializes a Flask app, sets a default route for `/` to `home`, and defines a `classify` route that accepts GET requests. The `classify` route uses a `model` to classify input parameters (sepal length, sepal width, petal length, and petal width) and renders the result in an `output.html` template. The application is run on port 5000 with debug mode enabled.

```
F:\> DG_internship > Flask > server.py > classify_type
1 import model # Import the python file containing the ML model
2 from flask import Flask, request, render_template, jsonify # Import flask libraries
3
4 # Initialize the flask class and specify the templates directory
5 app = Flask(__name__)
6
7 # Default route set as 'home'
8 @app.route('/')
9 def home():
10     return render_template('home.html') # Render home.html
11
12 # Route 'classify' accepts GET request
13 @app.route('/classify', methods=['GET'])
14 def classify_type():
15     try:
16         sepal_len = request.args.get('slen') # Get parameters for sepal length
17         sepal_wid = request.args.get('swid') # Get parameters for sepal width
18         petal_len = request.args.get('plen') # Get parameters for petal length
19         petal_wid = request.args.get('pwid') # Get parameters for petal width
20
21         # Get the output from the classification model
22         variety = model.classify(sepal_len, sepal_wid, petal_len, petal_wid)
23
24         # Render the output in new HTML page
25         return render_template('output.html', variety=variety)
26     except:
27         return 'Error'
28
29 # Run the Flask server
30 if __name__ == '__main__':
31     app.run(port=5000, debug=True)
```

Python 3.8.8 64-bit (base: conda) 0 0 0 Ln 21, Col 55 Spaces: 4 UTF-8 CRLF Python

4-Running server:

```
19 petal_wid = request.args.get('pwid') # Get parameters for petal width
20
21 # Get the output from the classification model
22 variety = model.classify(sepal_len, sepal_wid, petal_len, petal_wid)
```

(torchTens) F:\DG internship\Flask\python server.py
* Serving Flask app 'server' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 142-997-265
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

Html for home page

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Flower Variety</title>
7 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.0.0/css/bootstrap.min.css">
8 <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
9
10 <html>
11 <style>
12 overflow: hidden;
13
14
15 body{
16 position: absolute;
17 width: 100%;
18 height: 100%;
19 margin: 0;
20 padding: 0;
21
22 #login-form-container{
23 position: absolute;
24 width: 100%;
25 height: 100%;
26 display: flex;
27 align-items: center;
28 justify-content: center;
29
30 }
31 </style>
32 </head>
33 <body>
34 <div id="login-form-container">
35 <form action="/classify" method="GET">
36 <div class="card" style="width: 400px">
37 <div class="card-content">
38 <div class="media">
39 <div class="is-size-4 has-text-centered">Flower Variety Classification</div>
40 </div>
41 <div class="content">
42 <div class="field">
43 <div class="control">
44 Sepal Length: <input class="input" type="number" value="0" step="0.01" name="slen" id="slen">
45 </div>
46 </div>
47 <div class="field">
48 <div class="control">
49 Sepal Width: <input class="input" type="number" value="0" step="0.01" name="swid" id="swid">
50 </div>
51 </div>
52 <div class="field">
53 <div class="control">
54 Petal Length: <input class="input" type="number" value="0" step="0.01" name="plen" id="plen">
55 </div>
56 </div>
57 <div class="field">
58 <div class="control">
59 Petal Width: <input class="input" type="number" value="0" step="0.01" name="pwid" id="pwid">
60 </div>
61 </div>
62 </div>
63 </div>
64 </form>
65 </div>
66 </body>
67 </html>
```

Flower Variety Classification

Sepal Length:

6.9

Sepal Width:

4

Petal Length:

6

Petal Width:

3

Submit

The screenshot shows a web browser window with the title "Flower Variety". The address bar displays the URL "127.0.0.1:5000/classify?len=6.9&csid=4&plen=6&pwd=3". The page content includes a login form with the following elements:

- A title "Flower Variety" centered at the top.
- A login form container with the following fields:
 - A text input field for "username" with a placeholder "username".
 - A text input field for "password" with a placeholder "password".
 - A "login" button.
- A "register" link below the login button.

[Retry](#)