

STEPS OF DEPLOYMENT USING FLASK

NAME: VIVIAN KERUBO MOSOMI

BATCH CODE: LISUM39

SUBMISSION DATE: 27/11/2024

1. Loading the data

The screenshot shows a Jupyter Notebook interface with the following details:

- File Explorer:** Shows files in the "WEEK4" directory, including "Stroke Prediction.ipynb", "model.py", "app.py", and "X_test_encoded.pkl".
- Code Cell 179:** Displays Python code for importing libraries and defining models.
- Code Cell 180:** Displays Python code for loading a CSV dataset and showing its first 5 rows.
- Data Preview:** A table showing the first few rows of the loaded dataset:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked

2. Data Preparation

File Edit Selection View Go Run ... ← → 🔍 Week4

Stroke Prediction.ipynb X stroke.html # stroke.css app.py \ 1 app.py C:\...\deployment model.py 1 X test_encoded practice-env (Python 3.10.14)

Code Markdown Run All Restart Clear All Outputs Variables Outline ...

```
[181] Python
data.info()
[181]
```

```
... <class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   id               5110 non-null    int64  
 1   gender            5110 non-null    object  
 2   age                5110 non-null    float64 
 3   hypertension       5110 non-null    int64  
 4   heart_disease     5110 non-null    int64  
 5   ever_married      5110 non-null    object  
 6   work_type          5110 non-null    object  
 7   Residence_type    5110 non-null    object  
 8   avg_glucose_level 5110 non-null    float64 
 9   bmi                4909 non-null    float64 
 10  smoking_status    5110 non-null    object  
 11  stroke              5110 non-null    int64  
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
```

From the above information, we can see that the column bmi has 4909 entries out of the 5110 which may mean that it has missing values. Let's confirm if it does

File Edit Selection View Go Run ... 🔍 Week4

Stroke Prediction.ipynb X stroke.html # stroke.css app.py \ 1 app.py C:\...\deployment model.py 1 X test_encoded practice-env (Python 3.10.14)

Code Markdown Run All Restart Clear All Outputs Variables Outline ...

```
From the above information, we can see that the column bmi has 4909 entries out of the 5110 which may mean that it has missing values. Let's confirm if it does
```

```
[182] Python
data.isna().sum()
[182]
```

```
... id             0
gender          0
age             0
hypertension    0
heart_disease  0
ever_married    0
work_type       0
Residence_type 0
avg_glucose_level 0
bmi             201
smoking_status  0
stroke          0
dtype: int64
```

There are 201 missing entries in the bmi. This is close to 4% of the values in that column that are missing. This is quite a small percentage of missing values so we can drop them

```
data.dropna(subset=['bmi'], inplace=True)
```

File Edit Selection View Go Run ... 🔍 Week4

Stroke Prediction.ipynb X stroke.html # stroke.css app.py \ 1 app.py C:\...\deployment model.py 1 X test_encoded practice-env (Python 3.10.14)

Code Markdown Run All Restart Clear All Outputs Variables Outline ...

File Edit Selection View Go Run ... ← → Week4

Stroke Prediction.ipynb X stroke.html # stroke.css app.py \ 1 app.py C:\...\deployment model.py 1 X test_encoded practice-env (Python 3.10.14)

Code Markdown Run All Restart Clear All Outputs Variables Outline ...

There are 201 missing entries in the bmi. This is close to 4% of the values in that column that are missing. This is quite a small percentage of missing values so we can drop them

```
[183] data.dropna(subset=['bmi'],inplace=True) Python
```

```
[184] data.isna().sum() Python
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke	dtype:
...	0	0	0	0	0	0	0	0	0	0	0	0	int64

Now there are no missing values

File Edit Selection View Go Run ... Week4

Stroke Prediction.ipynb X stroke.html # stroke.css app.py \ 1 app.py C:\...\deployment model.py 1 X test_encoded practice-env (Python 3.10.14)

Code Markdown Run All Restart Clear All Outputs Variables Outline ...

Now there are no missing values

Checking for duplicates

```
[185] data.duplicated().sum() Python
```

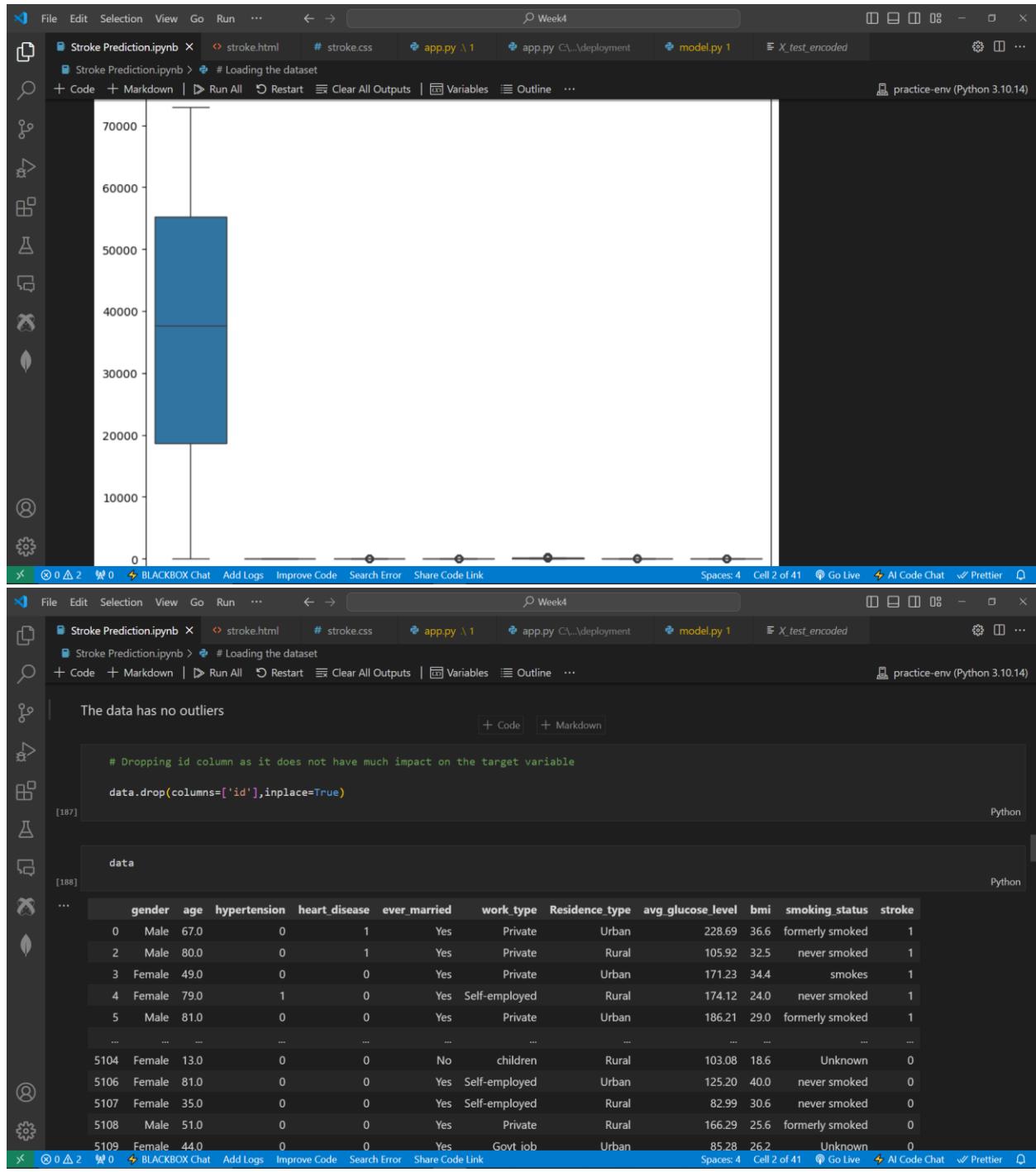
	0
...	0

There are no duplicated values in the dataset

```
[186] # Checking for outliers  
plt.figure(figsize=(10,8))  
sns.boxplot(data) Python
```

... <Axes: >

70000



File Edit Selection View Go Run ... ← → Week4

Stroke Prediction.ipynb X stroke.html # stroke.css app.py \ 1 app.py C:\...\deployment model.py 1 X test_encoded practice-env (Python 3.10.14)

Code Markdown Run All Restart Clear All Outputs Variables Outline ...

```
[189] data['gender'].value_counts()
... gender
Female    2897
Male      2011
Other       1
Name: count, dtype: int64
```

Gender other is only one out of all the gender values. This may have a very minute effect on our dataset hence let's drop it

```
[190] data.drop(data[data['gender'] == 'Other'].index, axis=0, inplace=True)
```

```
[191] # Confirming if the row has been dropped
data['gender'].value_counts()
... gender
Female    2897
Male      2011
Name: count, dtype: int64
```

File Edit Selection View Go Run ... ↵ Week4

Stroke Prediction.ipynb X stroke.html # stroke.css app.py \ 1 app.py C:\...\deployment model.py 1 X test_encoded practice-env (Python 3.10.14)

Code Markdown Run All Restart Clear All Outputs Variables Outline ...

```
[192] # Converting age column to int from float
data['age'] = data['age'].astype(int)
```

```
[193] # Checking for class imbalance
data['stroke'].value_counts(normalize=True)*100
... stroke
0    95.741646
1     4.258354
Name: proportion, dtype: float64
```

In this data, most people have no stroke as this category has carried 95% of the dataset. This may bring bias to our model so let's try correct it.

```
[194] data['stroke'].value_counts().plot(kind='bar', figsize=(10,8), title='Stroke Distribution')
plt.ylabel('Value Counts')
plt.xlabel('Stroke')
... Text(0.5, 0, 'Stroke')
```

ONE-HOT ENCODING

The model can not take categorical data so we have to encode so that it can be input to the model

```
[195] data
...      gender age hypertension heart_disease ever_married work_type Residence_type avg_glucose_level bmi smoking_status stroke
0     Male   67           0          1       Yes    Private      Urban        228.69  36.6 formerly smoked  1
2     Male   80           0          1       Yes    Private      Rural         105.92  32.5 never smoked  1
3   Female  49           0          0       Yes    Private      Urban        171.23  34.4 smokes            1
4   Female  79           1          0       Yes  Self-employed  Rural         174.12  24.0 never smoked  1
5     Male   81           0          0       Yes    Private      Urban        186.21  29.0 formerly smoked  1
...     ...   ...           ...          ...      ...      ...        ...        ...  ...        ...
5104  Female  13           0          0      No    children    Rural        103.08  18.6 Unknown            0
5106  Female  81           0          0       Yes  Self-employed  Urban        125.20  40.0 never smoked  0
5107  Female  35           0          0       Yes  Self-employed  Rural         82.99  30.6 never smoked  0
5108   Male   51           0          0       Yes    Private      Rural        166.29  25.6 formerly smoked  0
5109  Female  44           0          0       Yes  Govt_job      Urban        85.28  26.2 Unknown            0
4908 rows × 11 columns
```

17°C Partly cloudy 00:33 27/11/2024

```
[196] data['work_type'].value_counts()
... work_type
Private      2810
Self-employed  775
children      671
Govt_job      638
Never_worked    22
Name: count, dtype: int64
```

```
[154] # Splitting into train and test
X = data.drop(columns=['stroke'])
y = data['stroke']

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=42)
```

17°C Partly cloudy 00:33 27/11/2024

```
[155] data
...      gender age hypertension heart_disease ever_married work_type Residence_type avg_glucose_level bmi smoking_status stroke
0     Male   67           0          1       Yes    Private      Urban        228.69  36.6 formerly smoked  1
2     Male   80           0          1       Yes    Private      Rural         105.92  32.5 never smoked  1
```

17°C Partly cloudy 00:33 27/11/2024

```

# Encoding using pd.get_dummies

X_train_encoded = pd.get_dummies(X_train,columns=['gender','ever_married','work_type','Residence_type','smoking_status'],drop_first=True)
X_test_encoded = pd.get_dummies(X_test,columns=['gender','ever_married','work_type','Residence_type','smoking_status'],drop_first=True)

# Converting true and false to 1 and 0 respectively
pd.set_option('future.no_silent_downcasting',True)

X_train_encoded.replace({True:1,False:0},inplace=True)
X_test_encoded.replace({True:1,False:0},inplace=True)

```

X_test_encoded

	age	hypertension	heart_disease	avg_glucose_level	bmi	gender_Male	ever_married_Yes	work_type_Never_worked	work_type_Private	work_type_Self-employed	wo
4857	32	0	0	102.13	32.3	1	1	0	1	0	
3709	42	0	0	84.03	31.4	0	0	0	1	0	
964	66	1	0	74.90	32.1	1	1	0	1	0	
2971	21	0	0	71.06	25.3	0	0	0	1	0	
3262	47	0	0	88.49	22.2	1	1	0	1	0	

3. Modelling

MODELLING

```

# Correcting class imbalance
smote = SMOTE(random_state=42)

X_train_resampled,y_train_resampled = smote.fit_resample(X_train_encoded,y_train)

# Training random forest on the resampled data
lr = LogisticRegression(random_state=42,max_iter=1000)

# Fitting the model
lr.fit(X_train_resampled,y_train_resampled)

# Predicting the data
prediction = lr.predict(X_test_encoded)

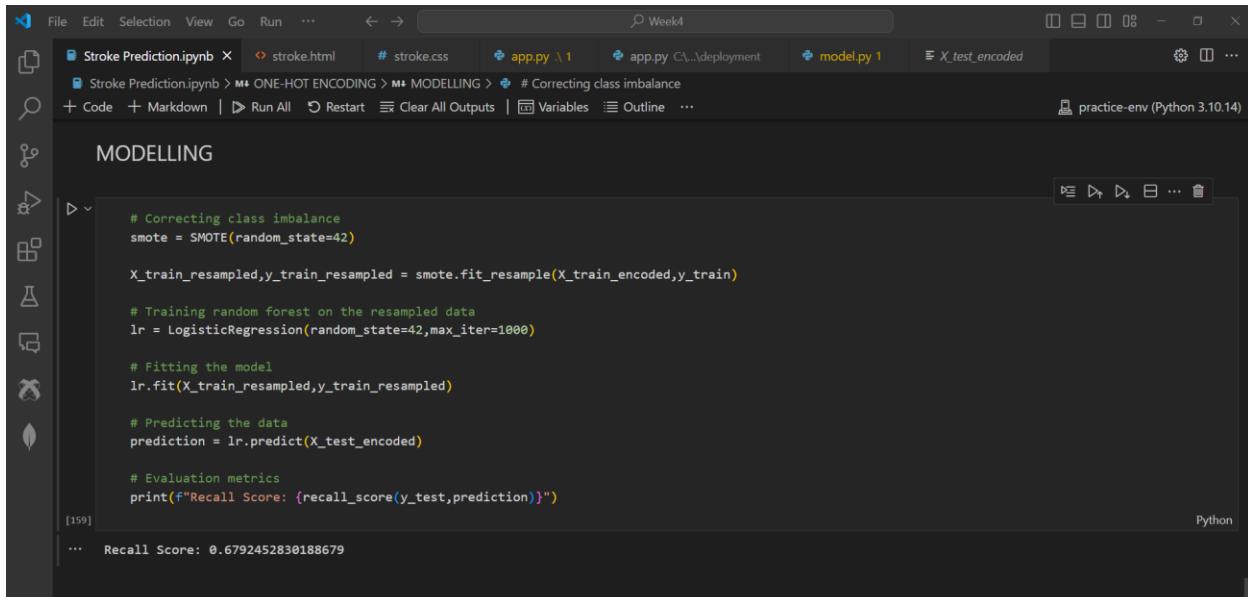
# Evaluation metrics
print(f"Recall Score: {recall_score(y_test,prediction)}")

```

Recall Score: 0.6792452830188679

Our model is performing quite well from the precision score results. The Logistic Regression is able to identify actual positive instances(Stroke instances) out of the total predictions

4. Deployment



The screenshot shows a Jupyter Notebook interface with the following code in the cell:

```
# Correcting class imbalance
smote = SMOTE(random_state=42)

X_train_resampled,y_train_resampled = smote.fit_resample(X_train_encoded,y_train)

# Training random forest on the resampled data
lr = LogisticRegression(random_state=42,max_iter=1000)

# Fitting the model
lr.fit(X_train_resampled,y_train_resampled)

# Predicting the data
prediction = lr.predict(X_test_encoded)

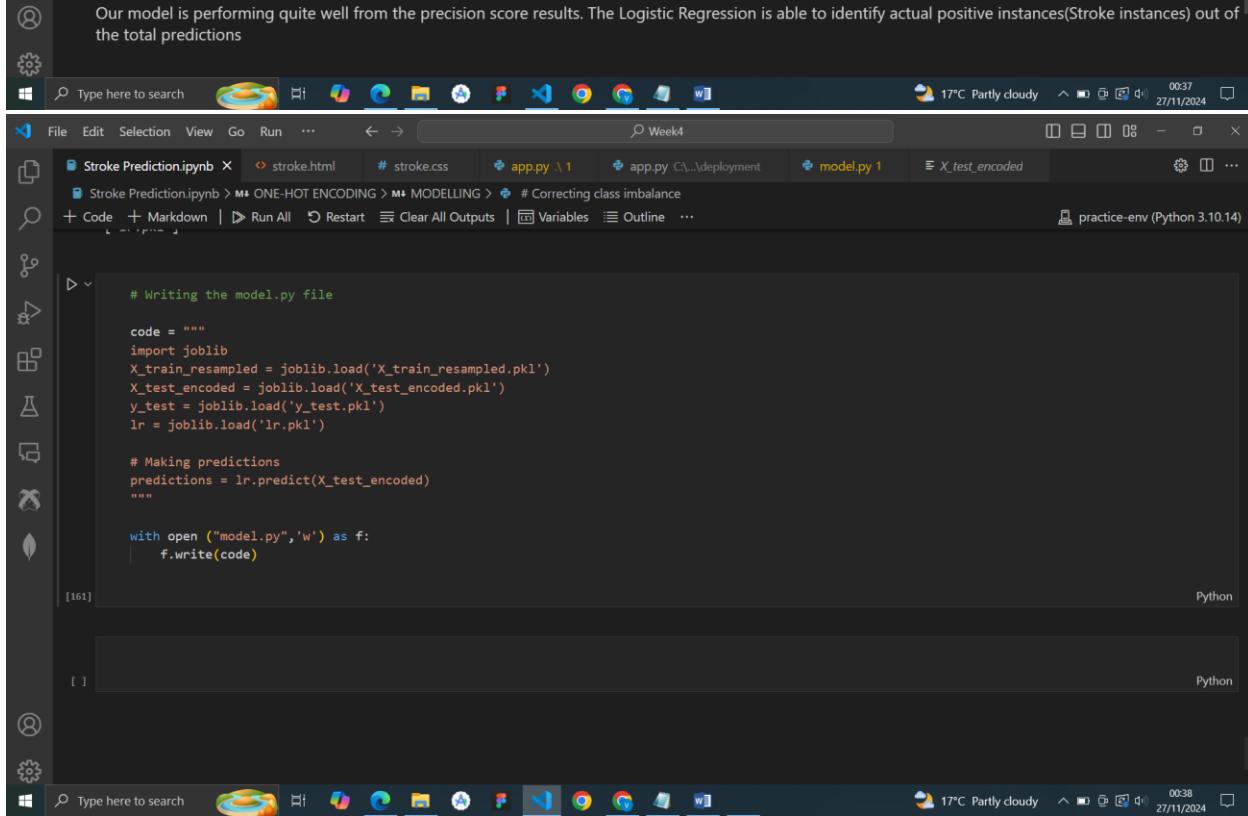
# Evaluation metrics
print(f"Recall Score: {recall_score(y_test,prediction)}")
```

Output:

```
[159] ... Recall Score: 0.6792452830188679
```

Text:

Our model is performing quite well from the precision score results. The Logistic Regression is able to identify actual positive instances(Stroke instances) out of the total predictions



The screenshot shows a Jupyter Notebook interface with the following code in the cell:

```
# Writing the model.py file

code = """
import joblib
X_train_resampled = joblib.load('X_train_resampled.pkl')
X_test_encoded = joblib.load('X_test_encoded.pkl')
y_test = joblib.load('y_test.pkl')
lr = joblib.load('lr.pkl')

# Making predictions
predictions = lr.predict(X_test_encoded)
"""

with open ("model.py",'w') as f:
    f.write(code)
```

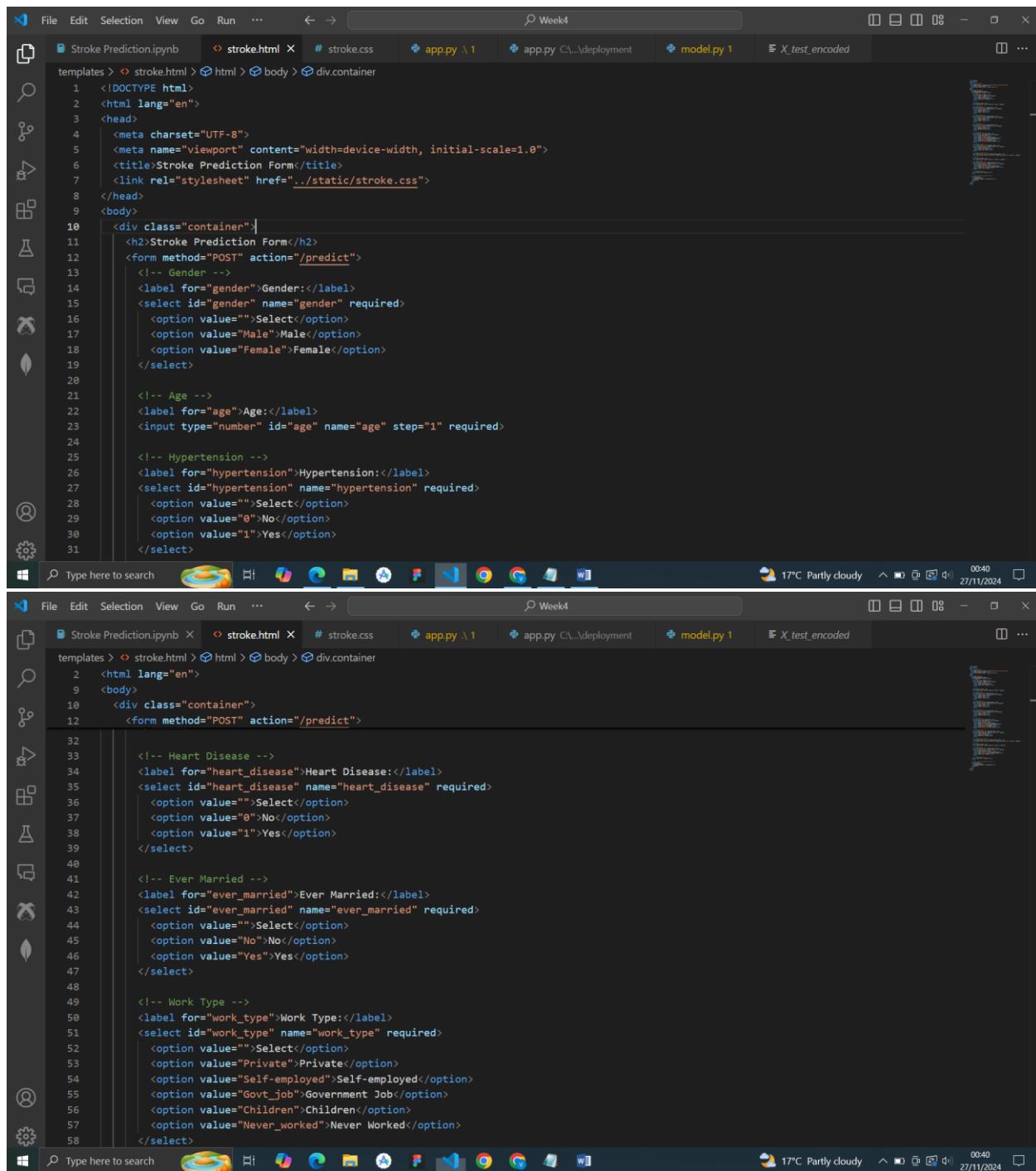
Output:

```
[161] [ ]
```

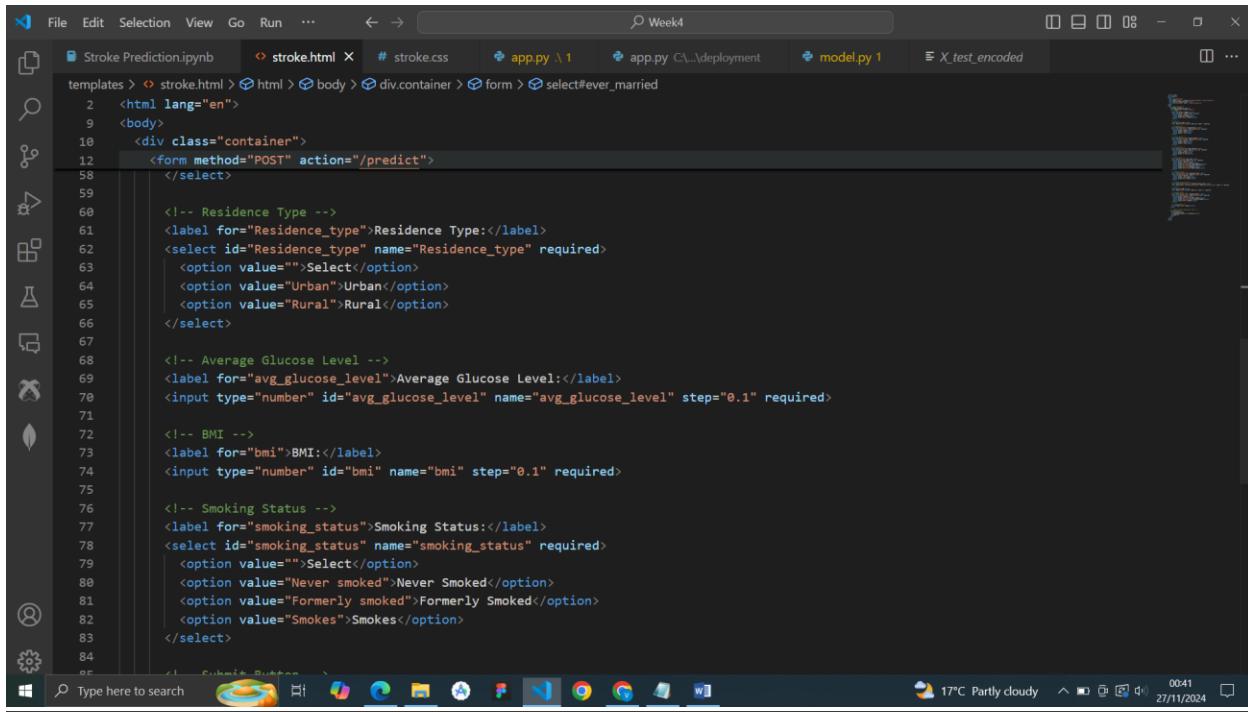
Text:

17°C Partly cloudy 00:38 27/11/2024

i) Html File

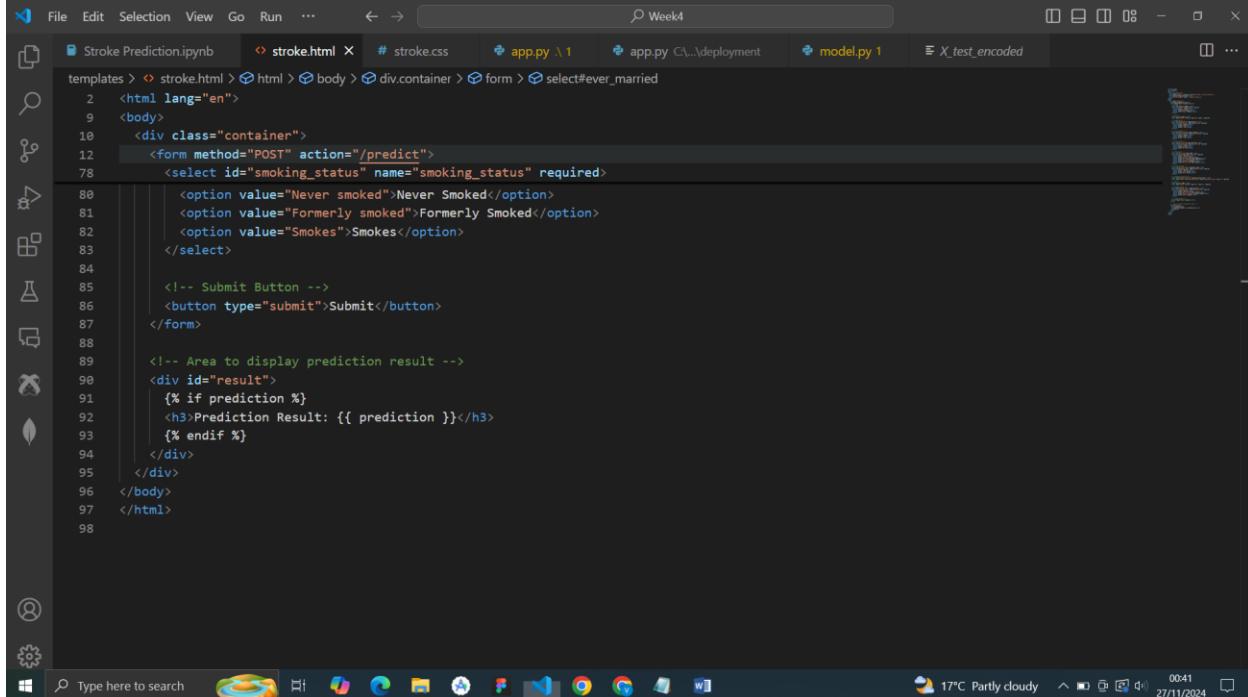


```
File Edit Selection View Go Run ... ← → ⌂ Week4 ⌂ X_test_encoded
Stroke Prediction.ipynb stroke.html # stroke.css app.py \ 1 app.py C:\...\deployment model.py 1
templates > stroke.html > html > body > div.container
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Stroke Prediction Form</title>
7      <link rel="stylesheet" href="../../static/stroke.css">
8  </head>
9  <body>
10     <div class="container">
11         <h2>Stroke Prediction Form</h2>
12         <form method="POST" action="/predict">
13             <!-- Gender -->
14             <label for="gender">Gender:</label>
15             <select id="gender" name="gender" required>
16                 <option value="">Select</option>
17                 <option value="Male">Male</option>
18                 <option value="Female">Female</option>
19             </select>
20
21             <!-- Age -->
22             <label for="age">Age:</label>
23             <input type="number" id="age" name="age" step="1" required>
24
25             <!-- Hypertension -->
26             <label for="hypertension">Hypertension:</label>
27             <select id="hypertension" name="hypertension" required>
28                 <option value="">Select</option>
29                 <option value="0">No</option>
30                 <option value="1">Yes</option>
31             </select>
32
33             <!-- Heart Disease -->
34             <label for="heart_disease">Heart Disease:</label>
35             <select id="heart_disease" name="heart_disease" required>
36                 <option value="">Select</option>
37                 <option value="0">No</option>
38                 <option value="1">Yes</option>
39             </select>
40
41             <!-- Ever Married -->
42             <label for="ever_married">Ever Married:</label>
43             <select id="ever_married" name="ever_married" required>
44                 <option value="">Select</option>
45                 <option value="No">No</option>
46                 <option value="Yes">Yes</option>
47             </select>
48
49             <!-- Work Type -->
50             <label for="work_type">Work Type:</label>
51             <select id="work_type" name="work_type" required>
52                 <option value="">Select</option>
53                 <option value="Private">Private</option>
54                 <option value="Self-employed">Self-employed</option>
55                 <option value="Govt_job">Government Job</option>
56                 <option value="Children">Children</option>
57                 <option value="Never_worked">Never Worked</option>
58             </select>
00:40 17°C Partly cloudy 27/11/2024
File Edit Selection View Go Run ... ← → ⌂ Week4 ⌂ X_test_encoded
Stroke Prediction.ipynb stroke.html # stroke.css app.py \ 1 app.py C:\...\deployment model.py 1
templates > stroke.html > html > body > div.container
2  <html lang="en">
9  <body>
10     <div class="container">
12         <form method="POST" action="/predict">
13
14             <!-- Heart Disease -->
15             <label for="heart_disease">Heart Disease:</label>
16             <select id="heart_disease" name="heart_disease" required>
17                 <option value="">Select</option>
18                 <option value="0">No</option>
19                 <option value="1">Yes</option>
20             </select>
21
22             <!-- Ever Married -->
23             <label for="ever_married">Ever Married:</label>
24             <select id="ever_married" name="ever_married" required>
25                 <option value="">Select</option>
26                 <option value="No">No</option>
27                 <option value="Yes">Yes</option>
28             </select>
29
30             <!-- Work Type -->
31             <label for="work_type">Work Type:</label>
32             <select id="work_type" name="work_type" required>
33                 <option value="">Select</option>
34                 <option value="Private">Private</option>
35                 <option value="Self-employed">Self-employed</option>
36                 <option value="Govt_job">Government Job</option>
37                 <option value="Children">Children</option>
38                 <option value="Never_worked">Never Worked</option>
39             </select>
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
00:40 17°C Partly cloudy 27/11/2024
```



The screenshot shows a Jupyter Notebook interface with the file "Stroke Prediction.ipynb" open. The code editor displays the "stroke.html" template, which contains the following HTML code:

```
templates > stroke.html > # stroke.css > app.py \ 1 > app.py C:\...\deployment > model.py 1 > X_test_encoded
1 <html lang="en">
2   <body>
3     <div class="container">
4       <form method="POST" action="/predict">
5         <select id="ever_married" name="ever_married" required>
6           <option value="Never married">Never married</option>
7           <option value="Married">Married</option>
8           <option value="Divorced">Divorced</option>
9           <option value="Widow">Widow</option>
10          <option value="Separated">Separated</option>
11          <option value="Husband missing">Husband missing</option>
12        </select>
13
14        <!-- Residence Type -->
15        <label for="Residence_type">Residence Type:</label>
16        <select id="Residence_type" name="Residence_type" required>
17          <option value="">Select</option>
18          <option value="Urban">Urban</option>
19          <option value="Rural">Rural</option>
20        </select>
21
22        <!-- Average Glucose Level -->
23        <label for="avg_glucose_level">Average Glucose Level:</label>
24        <input type="number" id="avg_glucose_level" name="avg_glucose_level" step="0.1" required>
25
26        <!-- BMI -->
27        <label for="bmi">BMI:</label>
28        <input type="number" id="bmi" name="bmi" step="0.1" required>
29
30        <!-- Smoking Status -->
31        <label for="smoking_status">Smoking Status:</label>
32        <select id="smoking_status" name="smoking_status" required>
33          <option value="">Select</option>
34          <option value="Never smoked">Never Smoked</option>
35          <option value="Formerly smoked">Formerly Smoked</option>
36          <option value="Smokes">Smokes</option>
37        </select>
38
39        <!-- Submit Button -->
40        <button type="submit">Submit</button>
41      </form>
42
43      <!-- Area to display prediction result -->
44      <div id="result">
45        {% if prediction %}
46          <h3>Prediction Result: {{ prediction }}</h3>
47        {% endif %}
48      </div>
49    </div>
50  </body>
51</html>
```



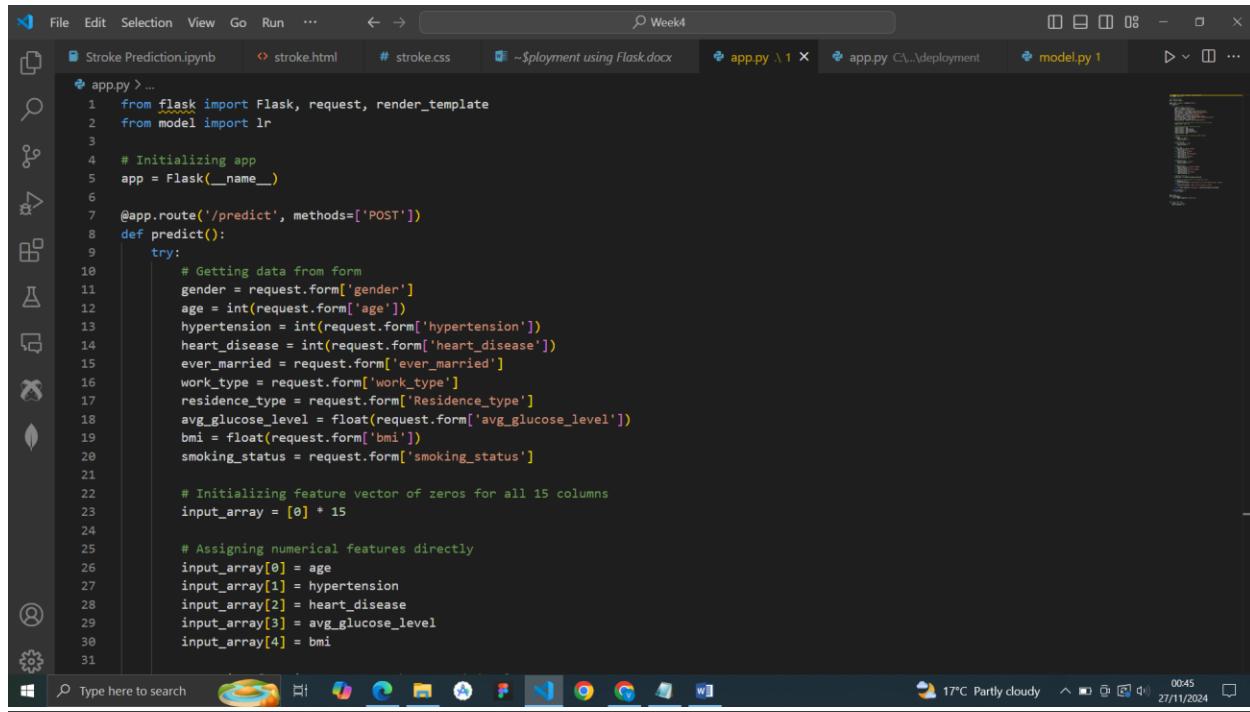
This screenshot shows the same Jupyter Notebook interface as the first one, but the code editor highlights the "smoking_status" section of the "stroke.html" template. The highlighted code is identical to the one shown in the first screenshot.

ii) **Css File**

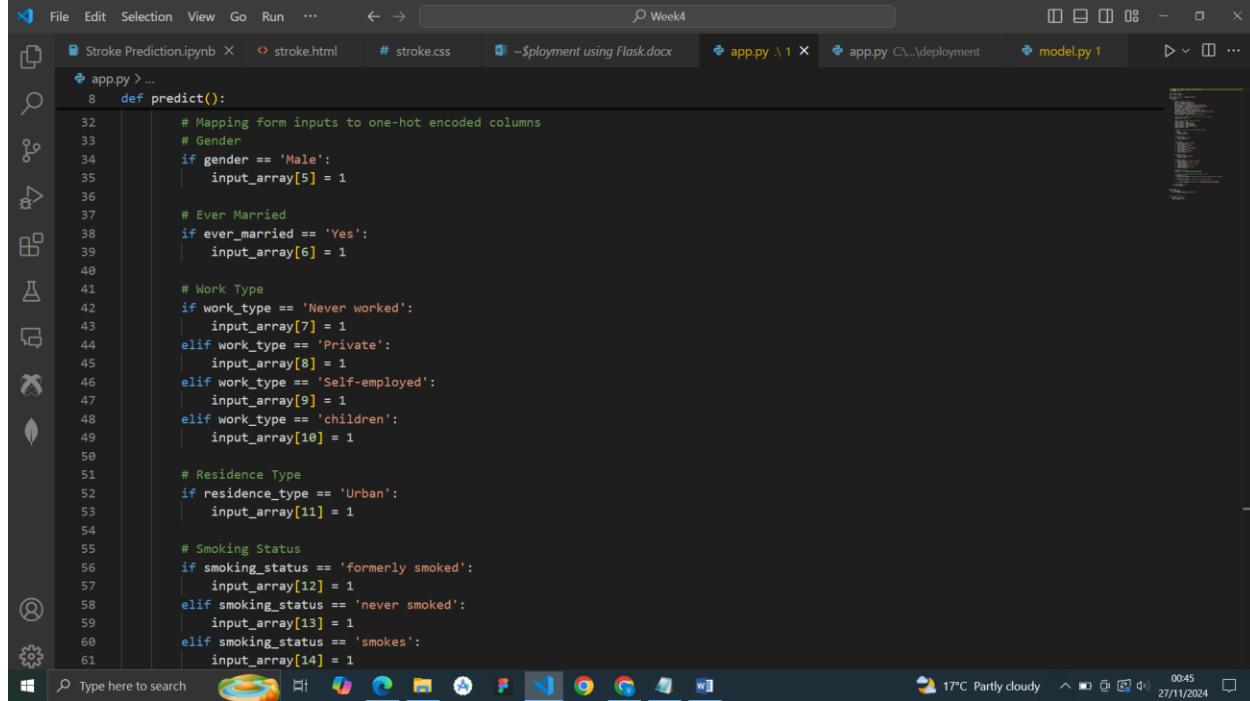
```
static > # stroke.css > .container
1 body {
2     font-family: Arial, sans-serif;
3     background-color: #bisque;
4     display: flex;
5     justify-content: center;
6     align-items: center;
7     height: 100vh;
8 }
9
10 .container {
11     background-color: #white;
12     padding: 50px;
13     border-radius: 8px;
14     box-shadow: 0 4px 8px #rgba(0, 0, 0, 0.1);
15     width: 300px;
16     margin-top: 150px;
17 }
18
19
20 h2 {
21     text-align: center;
22     margin-bottom: 20px;
23 }
24
25 label {
26     display: block;
27     margin-bottom: 5px;
28 }
29
30 input, select {
31     width: 100%;
```

```
static > # stroke.css > .container
25 label {
26 }
27
28
29 input, select {
30     width: 100%;
31     padding: 8px;
32     margin-bottom: 10px;
33     border: 1px solid #ccc;
34     border-radius: 4px;
35 }
36
37 button {
38     width: 100%;
39     padding: 10px;
40     background-color: #007bff;
41     color: #white;
42     border: none;
43     border-radius: 4px;
44     cursor: pointer;
45 }
46
47
48 button:hover {
49     background-color: #0056b3;
50 }
```

iii) App.py File (Flask)



```
File Edit Selection View Go Run ... ← → ⌂ Week4
app.py > ...
1  from flask import Flask, request, render_template
2  from model import lr
3
4  # Initializing app
5  app = Flask(__name__)
6
7  @app.route('/predict', methods=['POST'])
8  def predict():
9      try:
10          # Getting data from form
11          gender = request.form['gender']
12          age = int(request.form['age'])
13          hypertension = int(request.form['hypertension'])
14          heart_disease = int(request.form['heart_disease'])
15          ever_married = request.form['ever_married']
16          work_type = request.form['work_type']
17          residence_type = request.form['Residence_type']
18          avg_glucose_level = float(request.form['avg_glucose_level'])
19          bmi = float(request.form['bmi'])
20          smoking_status = request.form['smoking_status']
21
22          # Initializing feature vector of zeros for all 15 columns
23          input_array = [0] * 15
24
25          # Assigning numerical features directly
26          input_array[0] = age
27          input_array[1] = hypertension
28          input_array[2] = heart_disease
29          input_array[3] = avg_glucose_level
30          input_array[4] = bmi
31
```



```
File Edit Selection View Go Run ... ← → ⌂ Week4
app.py > ...
8  def predict():
32      # Mapping form inputs to one-hot encoded columns
33      # Gender
34      if gender == 'Male':
35          input_array[5] = 1
36
37      # Ever Married
38      if ever_married == 'Yes':
39          input_array[6] = 1
40
41      # Work Type
42      if work_type == 'Never worked':
43          input_array[7] = 1
44      elif work_type == 'Private':
45          input_array[8] = 1
46      elif work_type == 'Self-employed':
47          input_array[9] = 1
48      elif work_type == 'children':
49          input_array[10] = 1
50
51      # Residence Type
52      if residence_type == 'Urban':
53          input_array[11] = 1
54
55      # Smoking Status
56      if smoking_status == 'formerly smoked':
57          input_array[12] = 1
58      elif smoking_status == 'never smoked':
59          input_array[13] = 1
60      elif smoking_status == 'smokes':
61          input_array[14] = 1
```

A screenshot of a Windows desktop environment. In the foreground, a code editor window titled "Stroke Prediction.ipynb" is open, displaying Python code for a Flask application. The code includes functions for prediction and rendering templates. In the background, the Windows taskbar is visible, featuring the Start button, a search bar, and icons for various applications like File Explorer, Edge, and File History.

```
File Edit Selection View Go Run ... ← → Week4
Stroke Prediction.ipynb stroke.html # stroke.css ~$ployment using Flask.docx app.py 1 app.py C:\..\deployment model.py 1
app.py > ...
8 def predict():
61     |     input_array[14] = 1
62
63     # Making prediction
64     prediction = lr.predict([input_array])[0]
65
66     # Creating a message based on the prediction result
67     if prediction == 1:
68         prediction_message = "This person is at risk; they may have a stroke."
69     else:
70         prediction_message = "Not at risk of having a stroke."
71
72     return render_template('stroke.html', prediction=prediction_message)
73
74 except Exception as e:
75     return str(e)
76
77
78 @app.route('/')
79 def stroke_page():
80     return render_template('stroke.html')
81
82
83 # Run the Flask app
84 if __name__ == '__main__':
85     app.run(debug=True)
86
```

OUTPUT

Two screenshots of a web browser showing a "Stroke Prediction Form".

The form fields are:

- Gender: Male
- Age: 61
- Hypertension: Yes
- Heart Disease: Yes
- Ever Married: Yes
- Work Type: Private
- Residence Type: Urban
- Average Glucose Level: 228.6
- BMI: 36.6
- Smoking Status: Formerly Smoked

The second screenshot shows the same form with the "Smoking Status" field added.

Stroke Prediction Form

Heart Disease: Yes

Ever Married: Yes

Work Type: Private

Residence Type: Urban

Average Glucose Level: 228.6

BMI: 36.6

Smoking Status: Formerly Smoked

Prediction Result: This person is at risk; they may have a stroke.

Submit

127.0.0.1:5000/predict

Week4: Deployment on Flask

(3) Jireh, Make A Way, Yeshua ||

Stroke Prediction Form

17°C Partly cloudy 01:18 27/11/2024