

HEROKU DEPLOYMENT

Name: Nazrin Thanikattil Rafeeqe

Batch code: LISUM33

Submission date: 05-06-2024

Submitted To: Tutor

Step 1: Creating the wine_qaulity prediction model.

```
import pandas as pd
import numpy as np

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

data = pd.read_csv("C:\\Users\\Vhp\\Downloads\\WineQT.csv")

data.head(5)
data = data.drop('Id', axis = 1)

data.isnull().sum()
...
fixed acidity      0
volatile acidity   0
citric acid        0
residual sugar     0
chlorides          0
free sulfur dioxide 0
total sulfur dioxide 0
density           0
```

Step 2 : Performing EDA on Wine quality dataset

```
data.describe()

data['quality'].value_counts()

x_train = data.drop('quality', axis = 1)
y_train = data['quality']

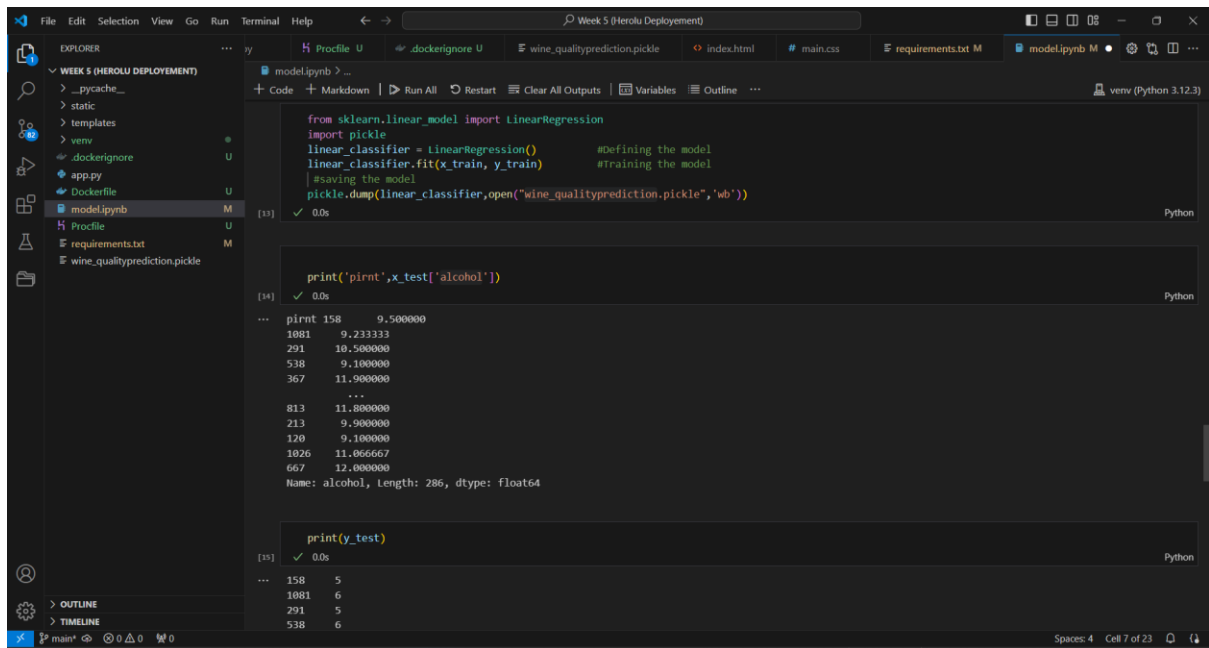
x_train

print('y_train', y_train)

x_train

x_train, x_test, y_train, y_test = train_test_split(x_train, y_train, test_size=0.25, random_state=42)
x_train.shape, x_test.shape, y_train.shape, y_test.shape
print('x_train', x_train)
print('y_train', y_train)
```

Step 3: Defining the Linear Regression model , training the dataset and saving.



The screenshot shows a VS Code editor with a Python file named `model.ipynb`. The code defines a linear regression model, trains it on a dataset, and saves the model to a pickle file. The output shows the model's predictions for a test set.

```
from sklearn.linear_model import LinearRegression
import pickle
linear_classifier = LinearRegression() #Defining the model
linear_classifier.fit(x_train, y_train) #training the model
#saving the model
pickle.dump(linear_classifier, open("wine_qualityprediction.pickle", 'wb'))

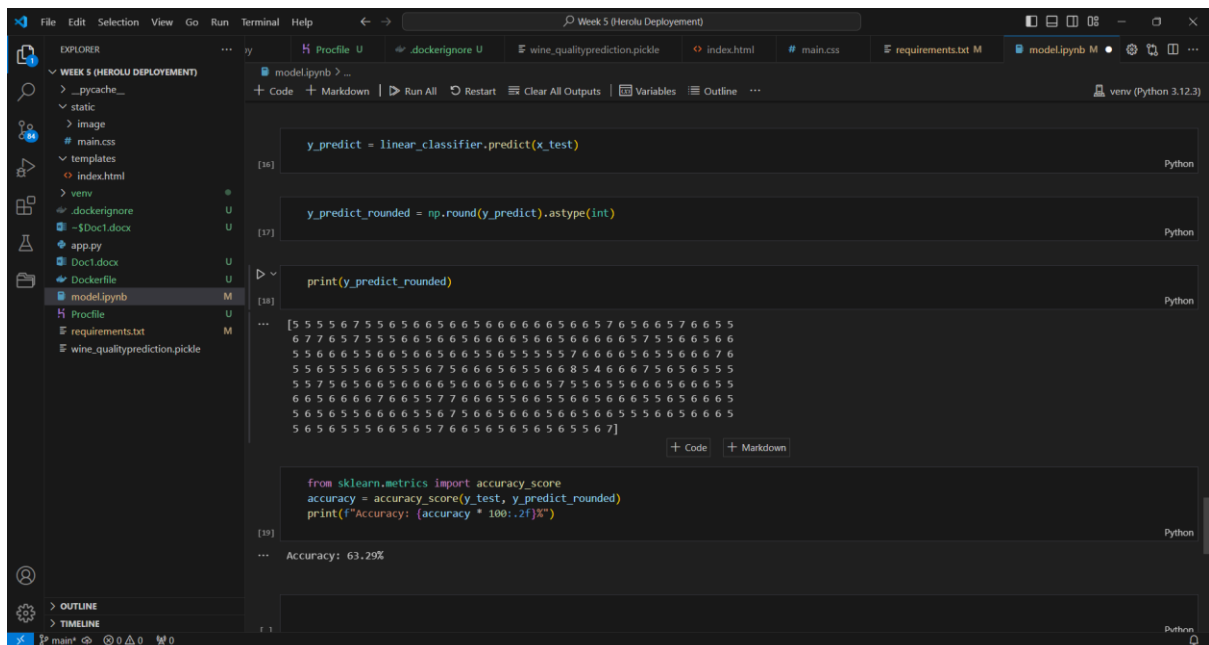
print('pirt', x_test['alcohol'])

...
pirt 158 9.500000
1081 9.233333
291 10.500000
538 9.100000
367 11.900000
...
813 11.800000
213 9.900000
120 9.100000
1026 11.066667
667 12.000000
Name: alcohol, Length: 286, dtype: float64

print(y_test)

...
158 5
1081 6
291 5
538 6
```

Step 4: Testing the model prediction with test data .



The screenshot shows a VS Code editor with a Python file named `model.ipynb`. The code defines a linear regression model, trains it on a dataset, and saves the model to a pickle file. The output shows the model's predictions for a test set.

```
y_predict = linear_classifier.predict(x_test)

y_predict_rounded = np.round(y_predict).astype(int)

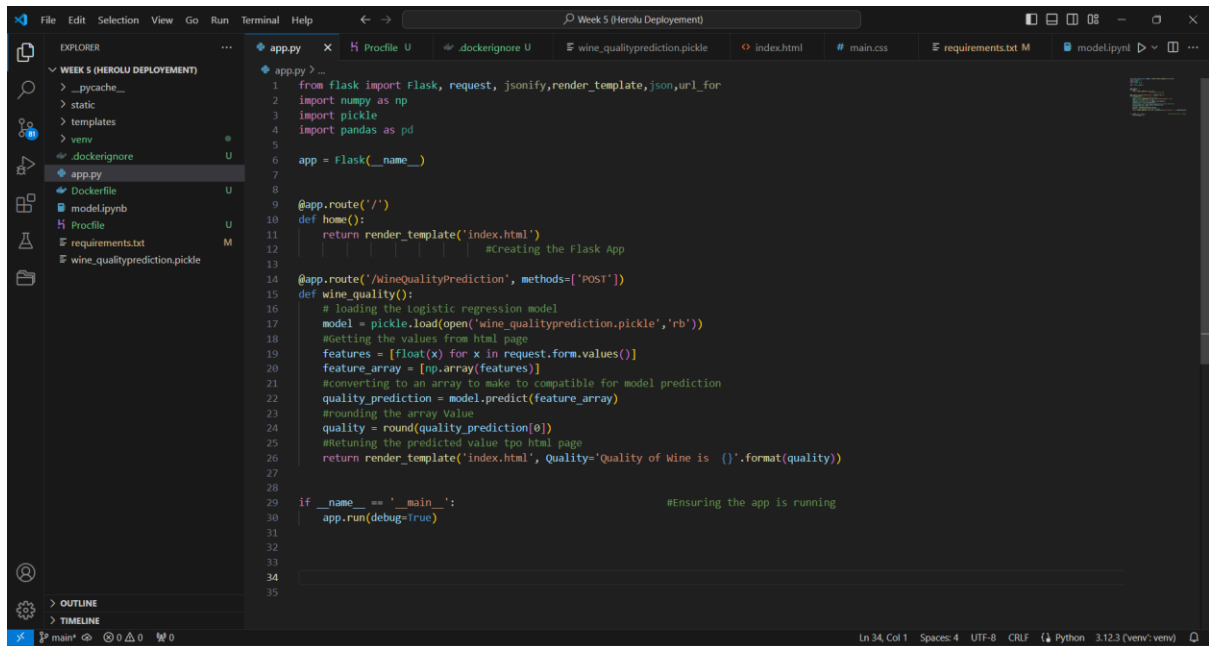
print(y_predict_rounded)

...
[5 5 5 6 7 5 5 6 5 6 6 5 6 6 5 6 6 6 6 6 6 5 6 6 5 7 6 5 6 6 5 5
 6 7 6 5 7 5 5 6 6 5 6 6 5 6 6 6 5 6 6 5 6 6 6 6 5 7 5 5 6 6 6 6
 5 6 6 6 5 5 6 6 5 6 6 5 6 6 5 5 5 5 7 6 6 6 6 5 6 5 6 6 6 7 6
 5 5 6 5 5 6 6 5 5 6 7 5 6 6 6 5 5 6 6 8 5 4 6 6 6 7 5 6 5 6 5 5
 5 5 7 5 6 6 6 5 6 6 6 6 5 6 6 6 5 7 5 5 6 5 5 6 6 6 5 6 6 6 5
 6 6 5 6 6 6 6 7 6 6 5 5 7 6 6 6 5 5 6 6 5 6 6 6 5 6 5 6 6 6 5
 5 6 5 5 5 6 6 6 6 5 5 6 7 5 6 6 5 6 6 6 5 6 6 5 5 5 6 6 5 6 6 5
 5 6 5 5 5 6 6 5 6 5 7 6 6 5 6 6 5 6 5 6 5 5 5 6 6 5 6 6 6 5]

from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_predict_rounded)
print(f"Accuracy: {accuracy * 100:.2f}%")

...
Accuracy: 63.29%
```

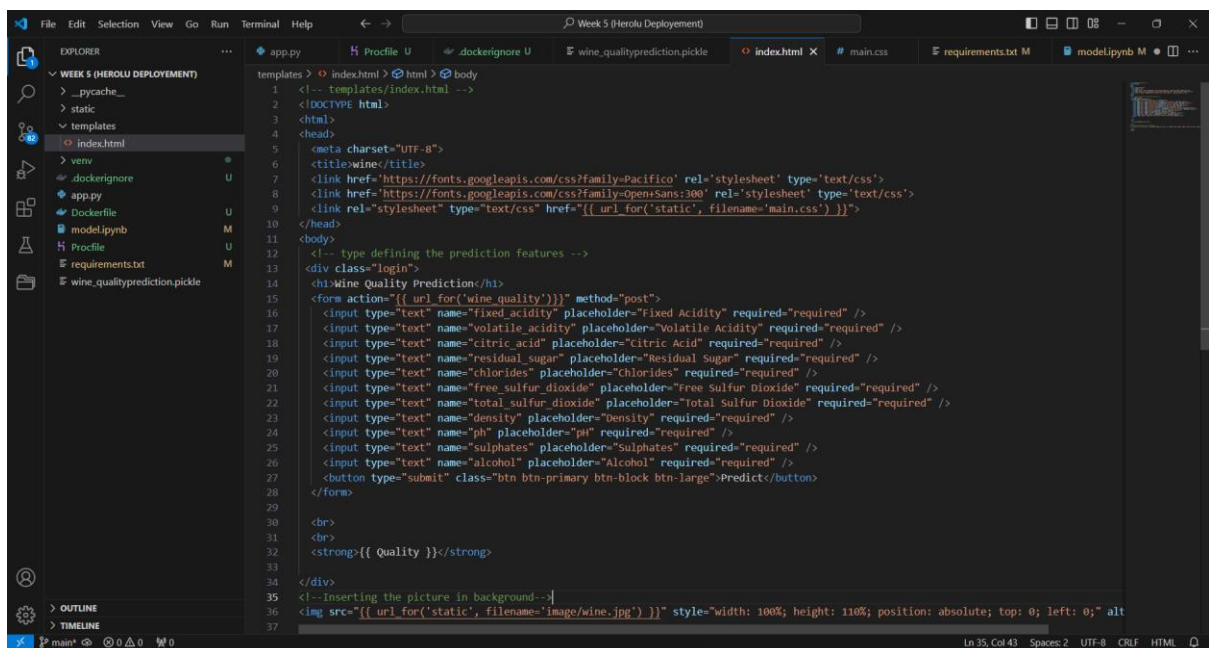
Step 5 : Creating the Flask app routes in app.py



The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the project structure for 'WEEK 5 (HEROKU DEPLOYMENT)'. The file 'app.py' is selected and open in the editor. The code defines a Flask application with two routes: a home route and a wine quality prediction route. The prediction route uses a pre-trained logistic regression model to predict wine quality based on input features.

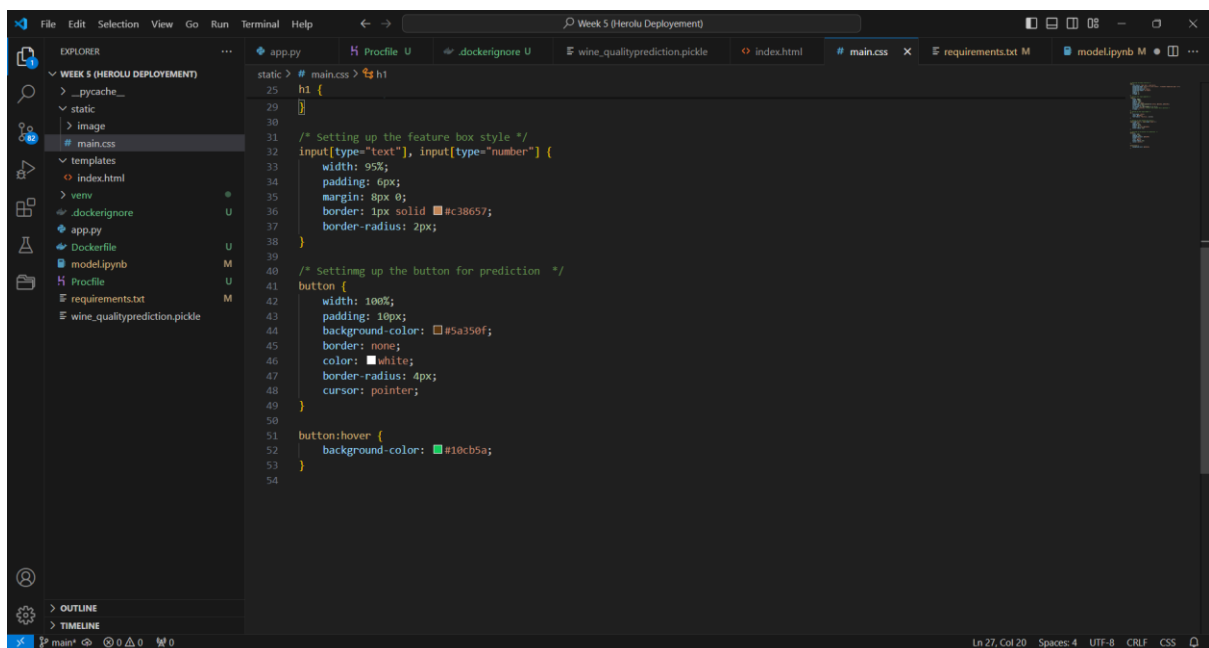
```
1 from flask import Flask, request, jsonify, render_template, jsonify, url_for
2 import numpy as np
3 import pickle
4 import pandas as pd
5
6 app = Flask(__name__)
7
8
9 @app.route('/')
10 def home():
11     return render_template('index.html')
12     #creating the Flask App
13
14 @app.route('/wineQualityPrediction', methods=['POST'])
15 def wine_quality():
16     # loading the logistic regression model
17     model = pickle.load(open('wine_qualityprediction.pickle', 'rb'))
18     #getting the values from html page
19     features = [float(x) for x in request.form.values()]
20     feature_array = [np.array(features)]
21     #converting to an array to make it compatible for model prediction
22     quality_prediction = model.predict(feature_array)
23     #rounding the array Value
24     quality = round(quality_prediction[0])
25     #returning the predicted value to html page
26     return render_template('index.html', Quality='Quality of Wine is {}'.format(quality))
27
28
29 if __name__ == '__main__':
30     app.run(debug=True)
31
32
33
34
35
```

Step 6: Creating the Index.html



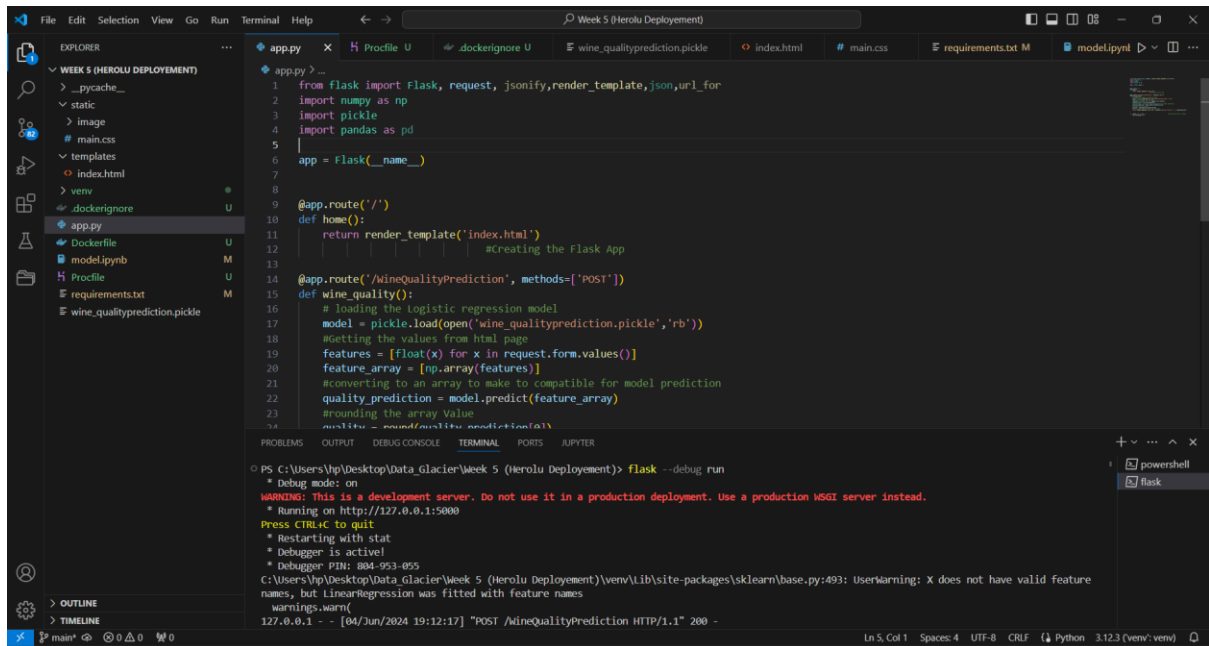
The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the project structure. The file 'index.html' is selected and open in the editor. The HTML file contains a form for wine quality prediction, with input fields for various chemical and physical properties of wine. The form is styled with a 'login' class and includes a 'Predict' button. A background image of a wine bottle is also included.

```
1 <!-- templates/index.html -->
2 <!DOCTYPE html>
3 <html>
4 <head>
5     <meta charset="UTF-8">
6     <title>Wine</title>
7     <link href="https://fonts.googleapis.com/css?family=Pacifico" rel="stylesheet" type="text/css">
8     <link href="https://fonts.googleapis.com/css?family=Open+Sans:300" rel="stylesheet" type="text/css">
9     <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='main.css') }}">
10 </head>
11 <body>
12     <!-- type defining the prediction features -->
13     <div class="login">
14         <h1>Wine Quality Prediction</h1>
15         <form action="{{ url_for('wine_quality') }}" method="post">
16             <input type="text" name="fixed_acidity" placeholder="Fixed Acidity" required="required" />
17             <input type="text" name="volatile_acidity" placeholder="Volatile Acidity" required="required" />
18             <input type="text" name="citric_acid" placeholder="Citric Acid" required="required" />
19             <input type="text" name="residual_sugar" placeholder="Residual Sugar" required="required" />
20             <input type="text" name="chlorides" placeholder="Chlorides" required="required" />
21             <input type="text" name="free_sulfur_dioxide" placeholder="Free Sulfur Dioxide" required="required" />
22             <input type="text" name="total_sulfur_dioxide" placeholder="Total Sulfur Dioxide" required="required" />
23             <input type="text" name="density" placeholder="Density" required="required" />
24             <input type="text" name="ph" placeholder="pH" required="required" />
25             <input type="text" name="sulphates" placeholder="Sulphates" required="required" />
26             <input type="text" name="alcohol" placeholder="Alcohol" required="required" />
27             <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
28         </form>
29
30         <br>
31         <br>
32         <strong>{{ Quality }}</strong>
33     </div>
34     <!-- Inserting the picture in background -->
35      # main.css > h1
1  /* Defining the body arbitries */
2  body {
3      font-family: 'Open Sans', sans-serif;
4      background-image: url('{{ url_for("static", filename="image/wine.jpg") }}');
5      background-size: cover;
6      background-position: center;
7      background-repeat: no-repeat;
8      margin: 0;
9      padding: 0;
10     z-index: 0;
11 }
12 /* Setting the login page style */
13 .login {
14     width: 400px;
15     height: 700px;
16     padding: 15px;
17     margin: 15px auto;
18     background: linear-gradient(to bottom, #2e281a, #e6c91);
19     border-radius: 5px;
20     box-shadow: 0 0 0px #000;
21     position: relative; /* Ensure the element has a position */
22     z-index: 1;
23 }
24 /* Setting up the login header */
25 h1 {
26     text-align: center;
27     color: #f46c7;
28     font-family: 'Pacifico', cursive;
29 }
30
31 /* Setting up the feature box style */
32 input[type="text"], input[type="number"] {
33     width: 95%;
34     padding: 6px;
35     margin: 8px 0;
36     border: 1px solid #c38657;
37     border-radius: 2px;
```



```
static > # main.css > h1
25 h1 {
26 }
27
28
29
30
31 /* Setting up the feature box style */
32 input[type="text"], input[type="number"] {
33     width: 95%;
34     padding: 6px;
35     margin: 8px 0;
36     border: 1px solid #c38657;
37     border-radius: 2px;
38 }
39
40 /* Setting up the button for prediction */
41 button {
42     width: 100%;
43     padding: 10px;
44     background-color: #5a359f;
45     border: none;
46     color: white;
47     border-radius: 4px;
48     cursor: pointer;
49 }
50
51 buttonshover {
52     background-color: #106b5a;
53 }
54
```

Step 8 : Running the flask app in localhost

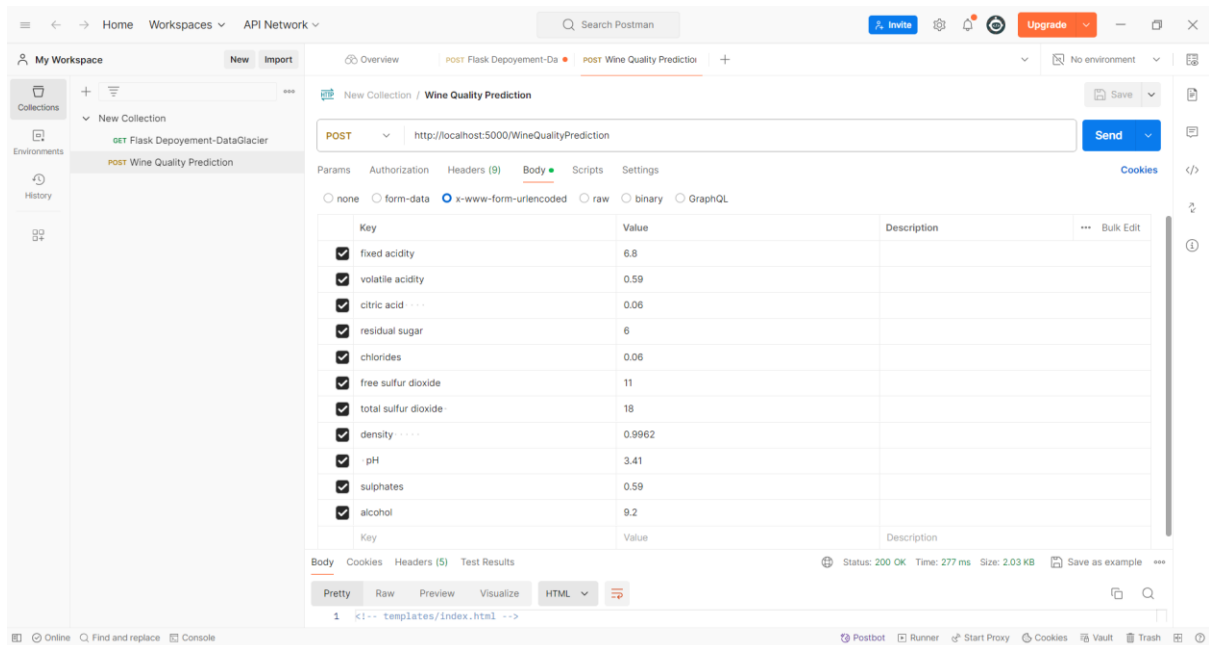


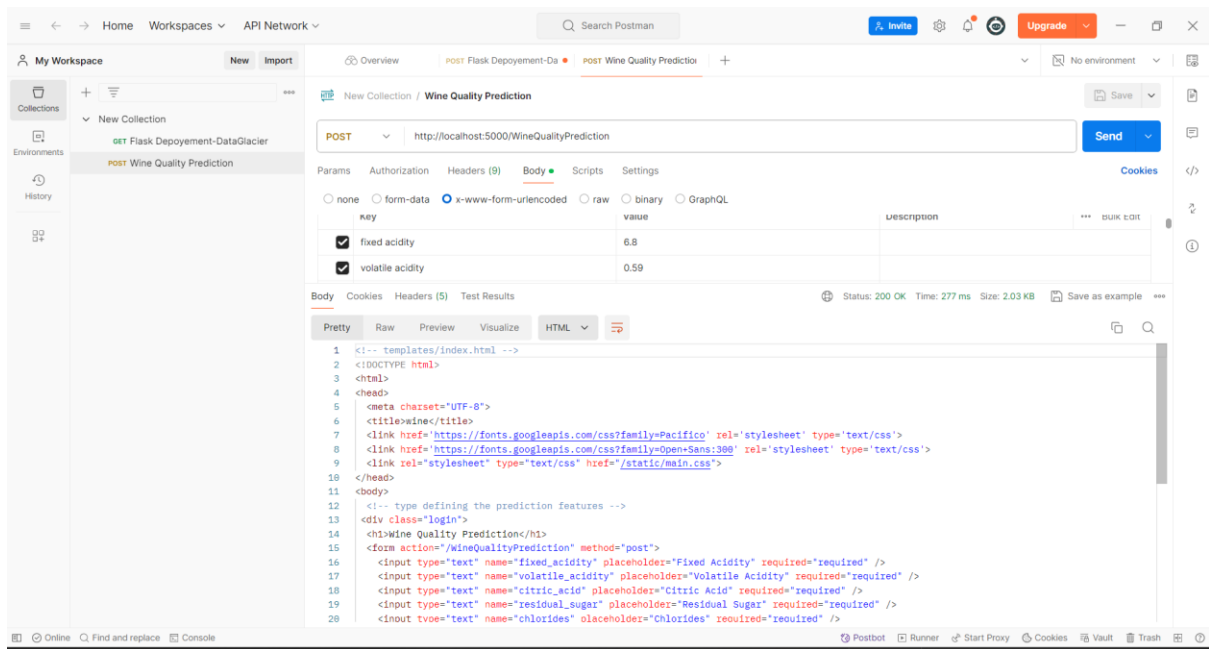
The screenshot shows a VS Code editor with a project named 'Week 5 (Heroku Deployment)'. The file explorer on the left shows a directory structure with files like `__pycache__`, `static`, `image`, `main.css`, `templates`, `index.html`, `venv`, `.dockerignore`, `app.py`, `Dockerfile`, `model.py`, `Profile`, `requirements.txt`, and `wine_qualityprediction.pickle`. The `app.py` file is open in the editor, showing the following code:

```
1 from flask import Flask, request, jsonify, render_template, jsonify, url_for
2 import numpy as np
3 import pickle
4 import pandas as pd
5
6 app = Flask(__name__)
7
8
9 @app.route('/')
10 def home():
11     return render_template('index.html')
12     #creating the Flask App
13
14 @app.route('/wineQualityPrediction', methods=['POST'])
15 def wine_quality():
16     # loading the logistic regression model
17     model = pickle.load(open('wine_qualityprediction.pickle', 'rb'))
18     #getting the values from html page
19     features = [float(x) for x in request.form.values()]
20     feature_array = [np.array(features)]
21     #converting to an array to make to compatible for model prediction
22     quality_prediction = model.predict(feature_array)
23     #rounding the array Value
24     quality = round(quality_prediction[0])
```

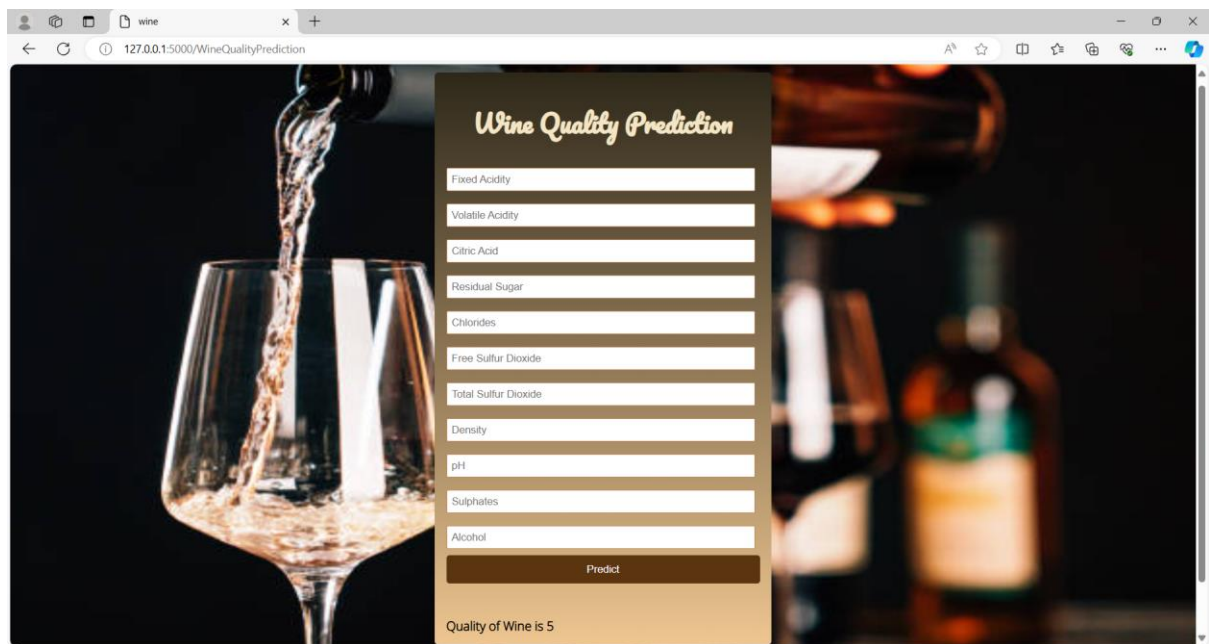
The terminal at the bottom shows the command `flask --debug run` being executed. The output indicates that the application is running on `http://127.0.0.1:5000` and that the debugger is active. A warning message is also visible: `UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names`.

Step 8 : Testing the endpoints in postman





Step 9 : Webpage wine at localhost.



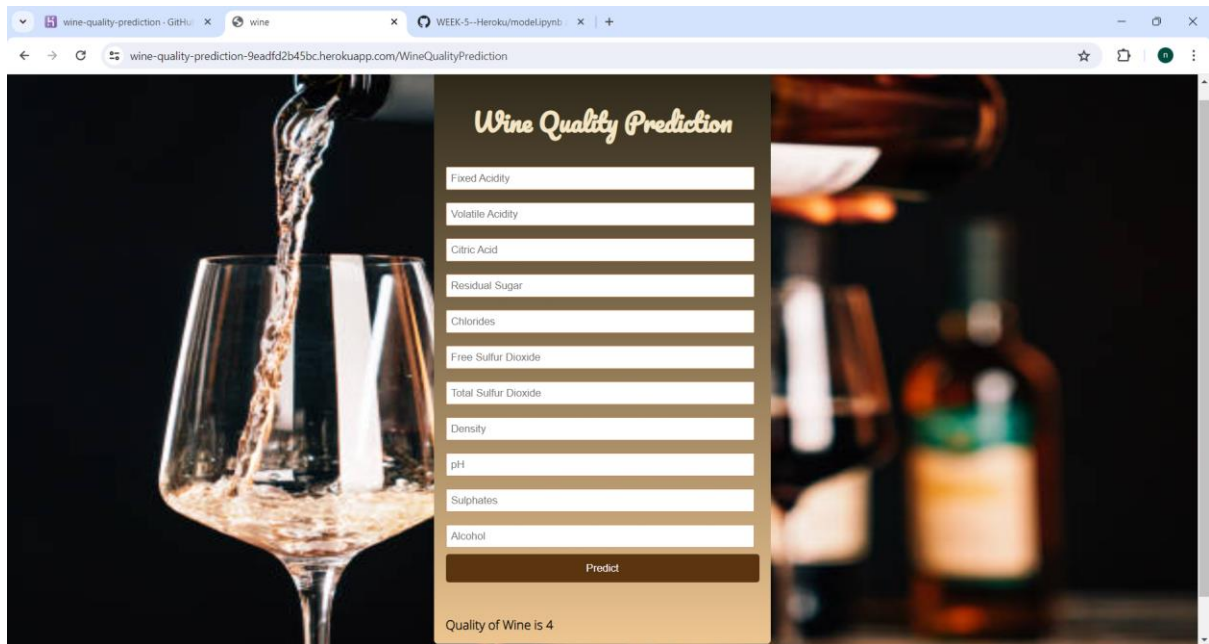
Step 10 : Connecting Heroku to the git repository where the app resides.

The screenshot shows the Heroku dashboard for the 'wine-quality-prediction' app. The app is connected to the GitHub repository '101nazrin/WEEK-5--Heroku'. The dashboard includes tabs for Overview, Resources, Deploy, Metrics, Activity, Access, and Settings. The 'Deploy' tab is active, showing options to add the app to a pipeline or connect it to GitHub. The 'Deployment method' section shows 'Heroku Git' and 'GitHub Connected' as options. The 'App connected to GitHub' section shows the app is connected to the repository '101nazrin/WEEK-5--Heroku' by user '101nazrin'. A 'Disconnect...' button is available.

Step 11 : Deployment successful

The screenshot shows the Heroku dashboard for the 'wine-quality-prediction' app, displaying the 'Manual deploy' section. The 'main' branch is selected for deployment. The 'Deploy Branch' button is visible. Below the deployment options, a list of deployment steps is shown with green checkmarks indicating success: 'Receive code from GitHub', 'Build main 4af7a87d', 'Release phase', and 'Deploy to Heroku'. A message at the bottom states 'Your app was successfully deployed.' with a 'View' button.

Step 12 : Viewing the website and testing the quality of wine with Xtest value at <https://wine-quality-prediction-9eadfd2b45bc.herokuapp.com/> .Obtained the predicted Quality of wine as 4.



The screenshot shows a web browser window with the URL wine-quality-prediction-9eadfd2b45bc.herokuapp.com/WineQualityPrediction. The page features a central form titled "Wine Quality Prediction" with a background image of wine being poured into a glass. The form contains the following input fields:

- Fixed Acidity
- Volatile Acidity
- Citric Acid
- Residual Sugar
- Chlorides
- Free Sulfur Dioxide
- Total Sulfur Dioxide
- Density
- pH
- Sulphates
- Alcohol

Below the input fields is a "Predict" button. At the bottom of the form, the text "Quality of Wine is 4" is displayed.
