DATA GLACIER VIRTUAL INTERNSHIP

Deployment Documentation for BTC Closing Price Prediction App

Name: Stephanie Dawsonn-Andoh

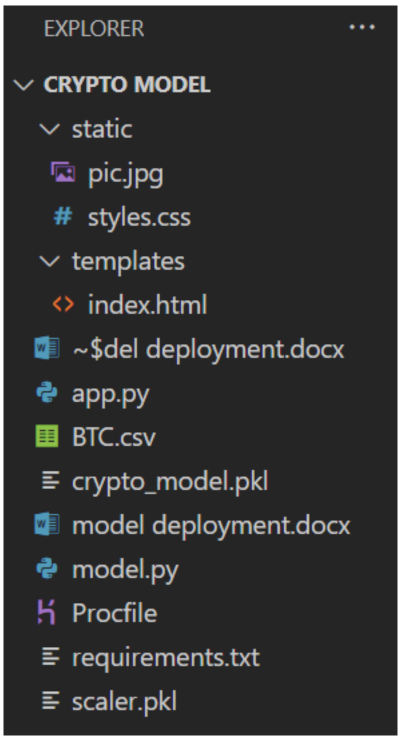
Batch Code: LISUM36

Submission Date: 9/4/2024

Submitted To: Data Glacier

---

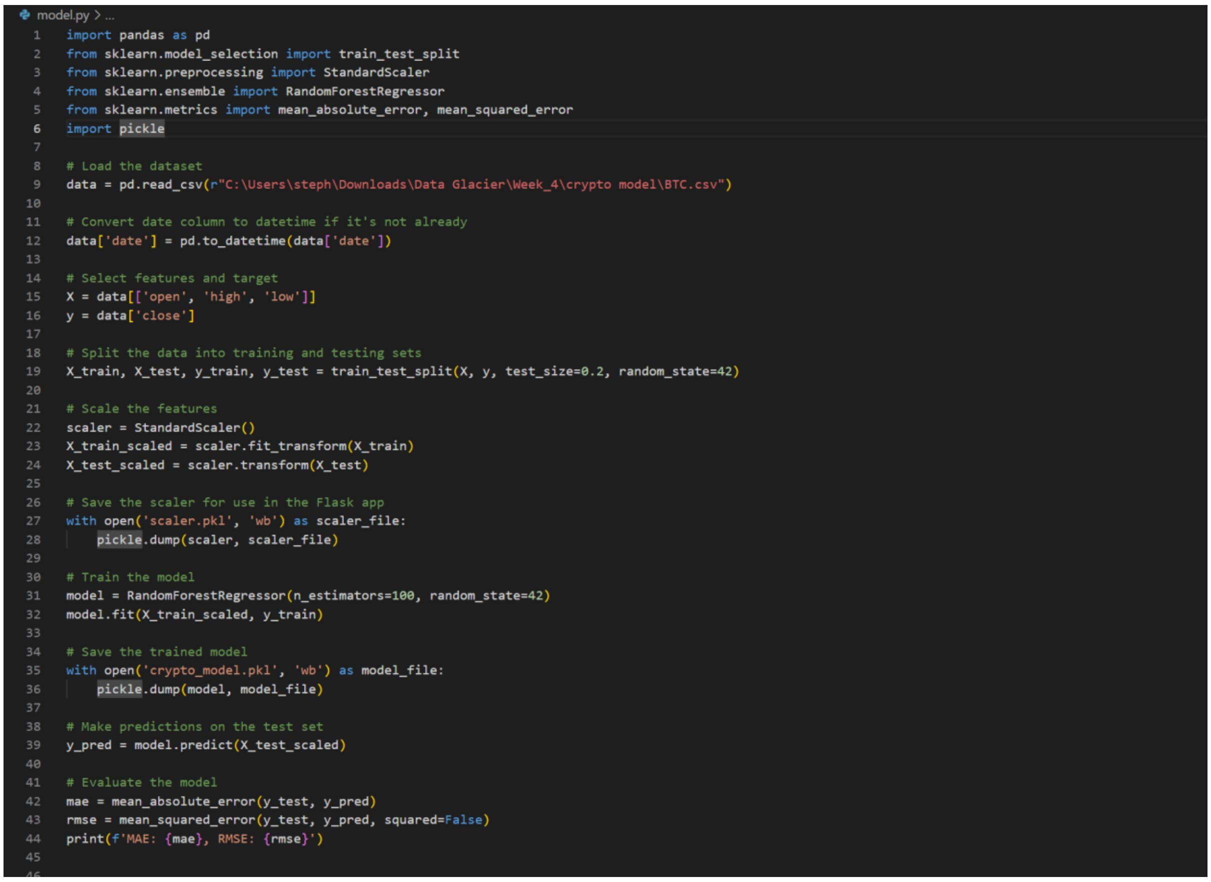
1. File Structure Setup



```
EXPLORER
CRYPTO MODEL
  static
    pic.jpg
    styles.css
  templates
    index.html
  ~$del deployment.docx
  app.py
  BTC.csv
  crypto_model.pkl
  model deployment.docx
  model.py
  Procfile
  requirements.txt
  scaler.pkl
```

- *Description: This shows the organization of files within the project directory.*

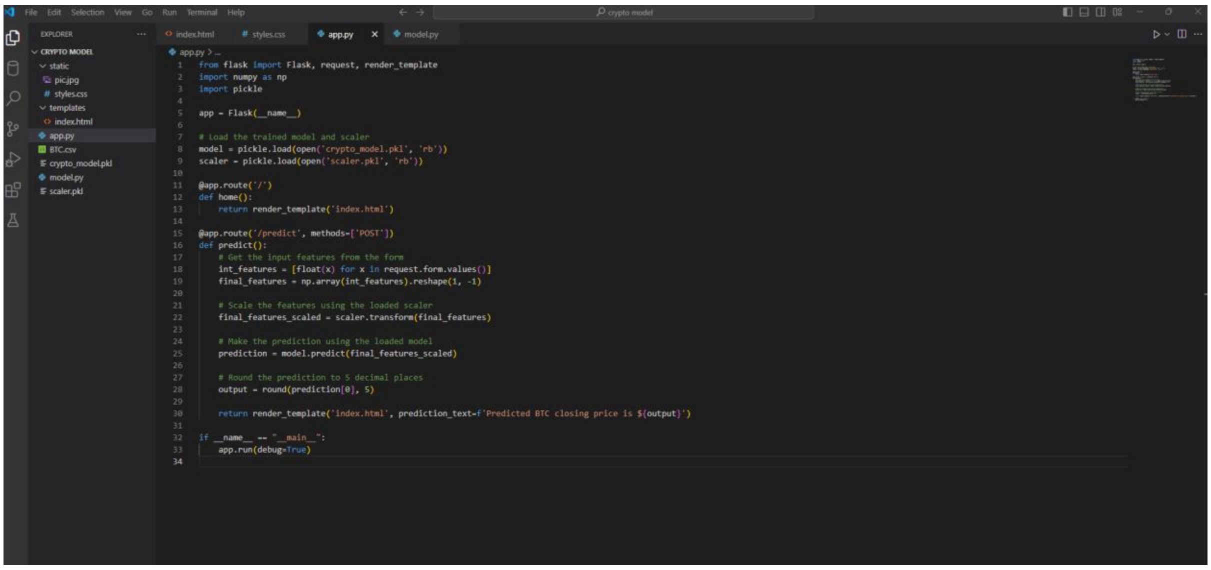
2. Model Training



```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error
import pickle

# Load the dataset
data = pd.read_csv(r"C:\Users\steph\Downloads\Data Glacier\Week_4\crypto model\BTC.csv")

# Convert date column to datetime if it's not already
data['date'] = pd.to_datetime(data['date'])

# Select features and target
X = data[['open', 'high', 'low']]
y = data['close']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Save the scaler for use in the Flask app
with open('scaler.pkl', 'wb') as scaler_file:
    pickle.dump(scaler, scaler_file)

# Train the model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train_scaled, y_train)

# Save the trained model
with open('crypto_model.pkl', 'wb') as model_file:
    pickle.dump(model, model_file)

# Make predictions on the test set
y_pred = model.predict(X_test_scaled)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
print(f'MAE: {mae}, RMSE: {rmse}')
```

- *Description: This captures the Python code used to train and save the model.*

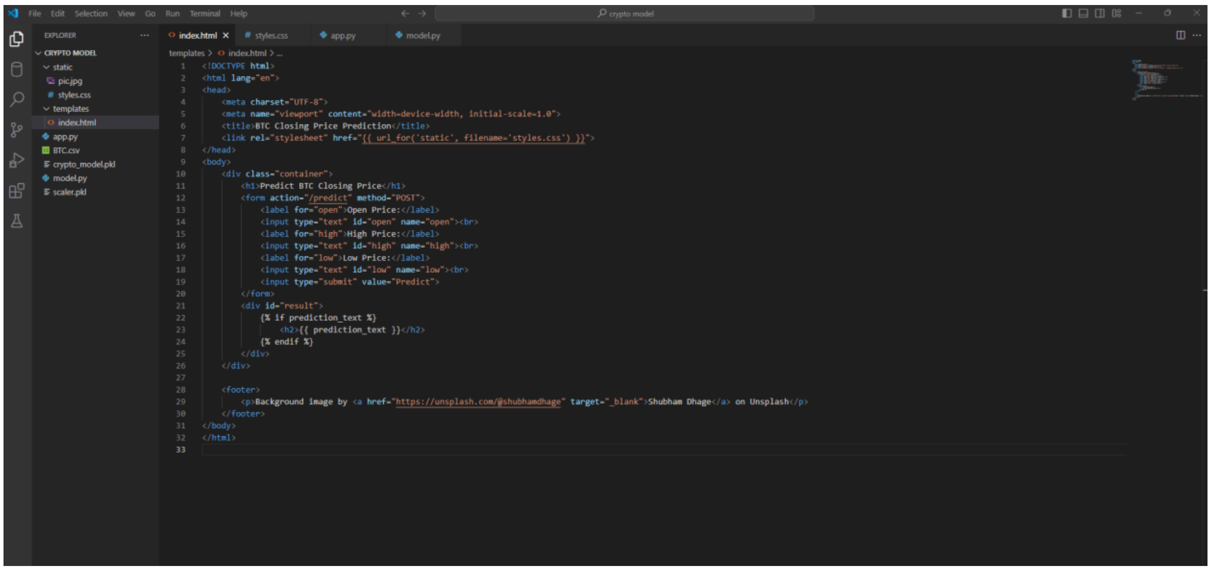
3. Flask App Setup



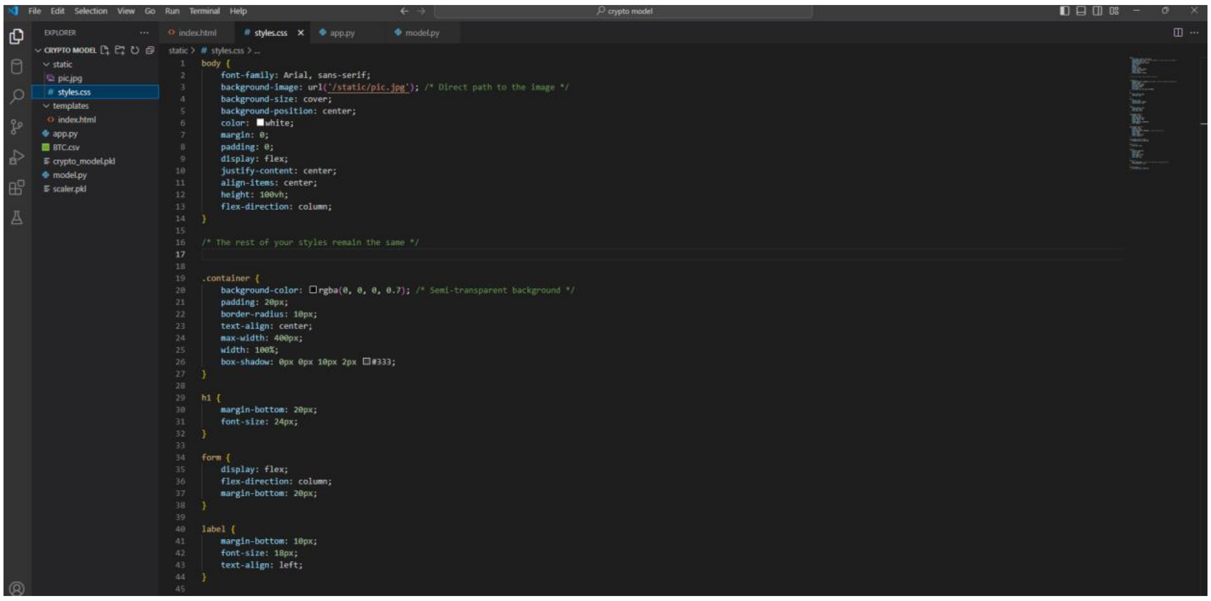
```python
from flask import Flask, request, render_template
import numpy as np
import pickle

app = Flask(__name__)

# Load the trained model and scaler
model = pickle.load(open('crypto_model.pkl', 'rb'))
scaler = pickle.load(open('scaler.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    # Get the input features from the form
    int_features = [float(x) for x in request.form.values()]
    final_features = np.array(int_features).reshape(1, -1)

    # Scale the features using the loaded scaler
    final_features_scaled = scaler.transform(final_features)

    # Make the prediction using the loaded model
    prediction = model.predict(final_features_scaled)

    # Round the prediction to 5 decimal places
    output = round(prediction[0], 5)

    return render_template('index.html', prediction_text=f'Predicted BTC closing price is ${output}')

if __name__ == "__main__":
    app.run(debug=True)
```

- *Description: This shows the Flask application code in app.py.*

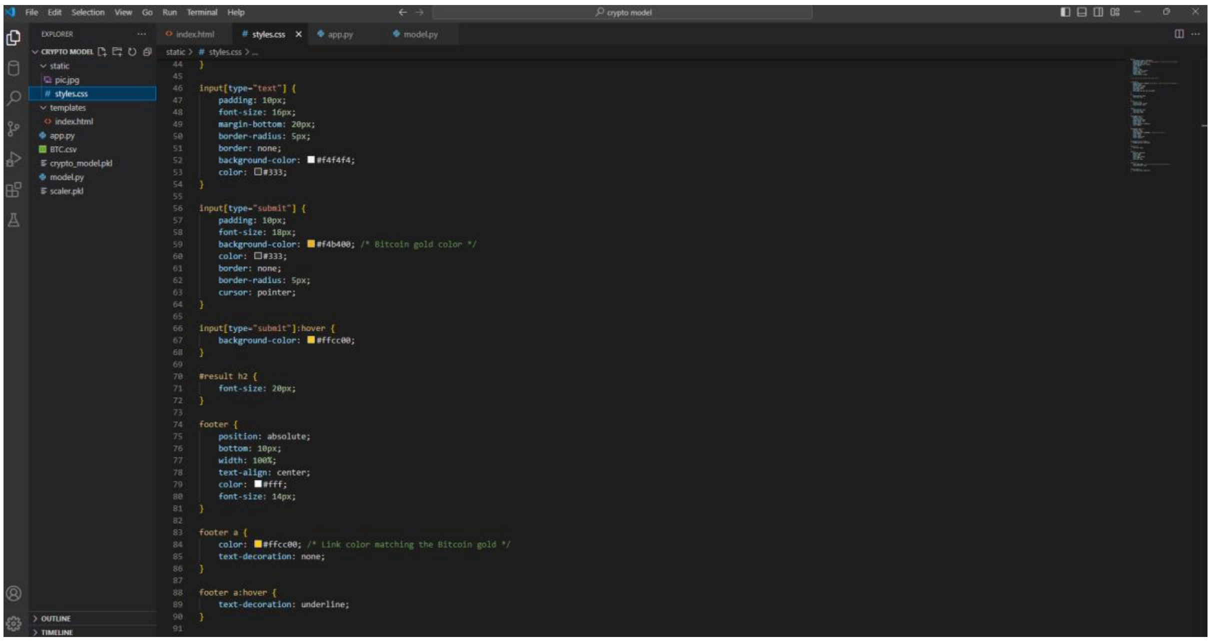
4. HTML and CSS Setup



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>BTC Closing Price Prediction</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <div class="container">
        <h1>Predict BTC Closing Price</h1>
        <form action="/predict" method="POST">
            <label for="open">Open Price:</label>
            <input type="text" id="open" name="open"><br>
            <label for="high">High Price:</label>
            <input type="text" id="high" name="high"><br>
            <label for="low">Low Price:</label>
            <input type="text" id="low" name="low"><br>
            <input type="submit" value="Predict">
        </form>
        <div id="result">
            {% if prediction_text %}
                <h2>{{ prediction_text }}</h2>
            {% endif %}
        </div>
    </div>

    <footer>
        <p>Background image by <a href="https://unsplash.com/@shubhamdhage" target="_blank">Shubham Dhage</a> on Unsplash</p>
    </footer>
</body>
</html>
```



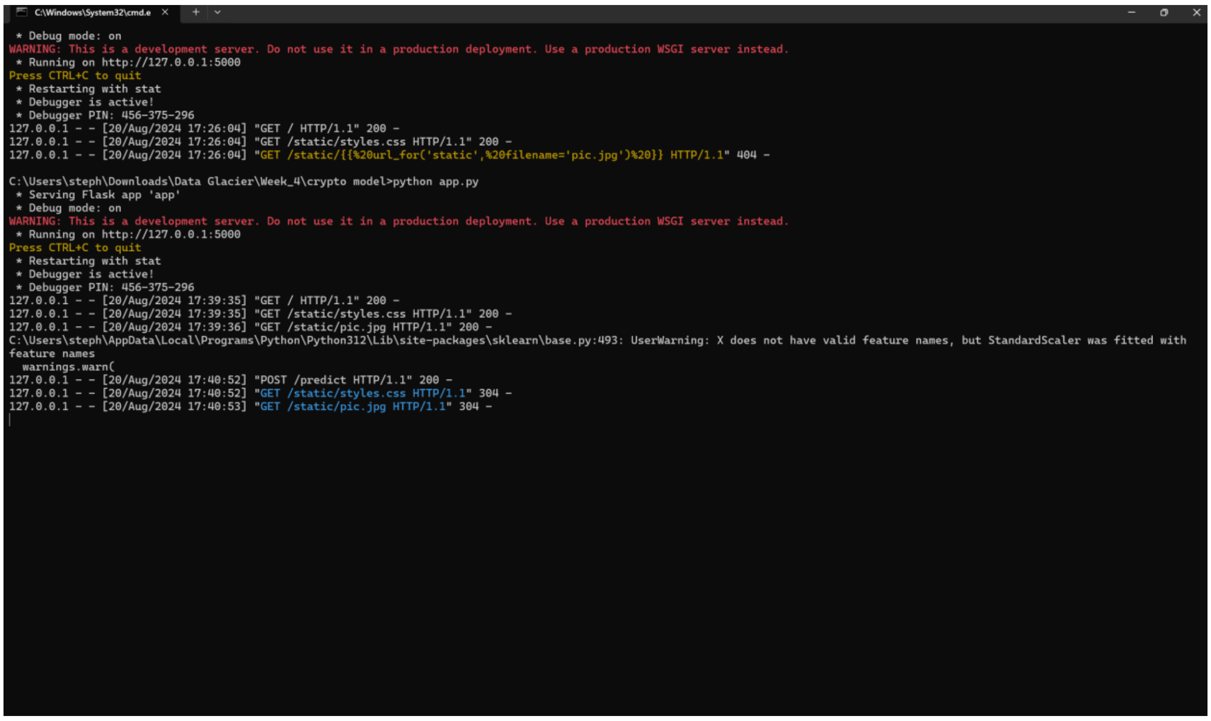
```css
body {
    font-family: Arial, sans-serif;
    background-image: url('/static/pic.jpg'); /* Direct path to the image */
    background-size: cover;
    background-position: center;
    color: white;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    flex-direction: column;
}

/* The rest of your styles remain the same */


.container {
    background-color: rgba(0, 0, 0, 0.7); /* Semi-transparent background */
    padding: 20px;
    border-radius: 10px;
    text-align: center;
    max-width: 400px;
    width: 100%;
    box-shadow: 0px 0px 10px 2px #333;
}

h1 {
    margin-bottom: 20px;
    font-size: 24px;
}

form {
    display: flex;
    flex-direction: column;
    margin-bottom: 20px;
}

label {
    margin-bottom: 10px;
    font-size: 18px;
    text-align: left;
}

```



```css
}

input[type="text"] {
    padding: 10px;
    font-size: 16px;
    margin-bottom: 20px;
    border-radius: 5px;
    border: none;
    background-color: #f4f4f4;
    color: #333;
}

input[type="submit"] {
    padding: 10px;
    font-size: 18px;
    background-color: #f4b400; /* Bitcoin gold color */
    color: #333;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

input[type="submit"]:hover {
    background-color: #ffcc00;
}

#result h2 {
    font-size: 20px;
}

footer {
    position: absolute;
    bottom: 10px;
    width: 100%;
    text-align: center;
    color: #fff;
    font-size: 14px;
}

footer a {
    color: #ffcc00; /* Link color matching the Bitcoin gold */
    text-decoration: none;
}

footer a:hover {
    text-decoration: underline;
}

```

- *Description: This shows the structure of the HTML file and the styling applied via CSS.*

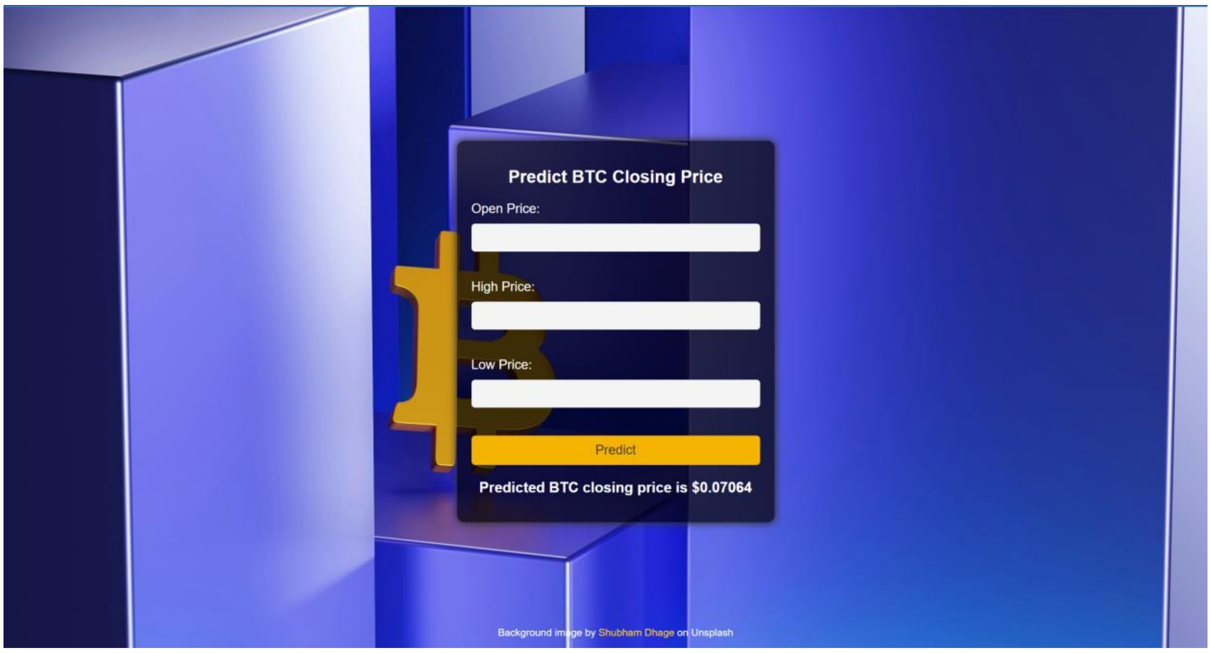
5. Running the Flask App



```
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 456-375-296
127.0.0.1 - - [20/Aug/2024 17:26:04] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [20/Aug/2024 17:26:04] "GET /static/styles.css HTTP/1.1" 200 -
127.0.0.1 - - [20/Aug/2024 17:26:04] "GET /static/{{%20url_for('static',%20filename='pic.jpg')%20}} HTTP/1.1" 404 -

C:\Users\steph\Downloads\Data Glacier\Week_4\crypto model>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 456-375-296
127.0.0.1 - - [20/Aug/2024 17:39:35] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [20/Aug/2024 17:39:35] "GET /static/styles.css HTTP/1.1" 200 -
127.0.0.1 - - [20/Aug/2024 17:39:36] "GET /static/pic.jpg HTTP/1.1" 200 -
C:\Users\steph\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but StandardScaler was fitted with
feature names
  warnings.warn(
127.0.0.1 - - [20/Aug/2024 17:40:52] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [20/Aug/2024 17:40:52] "GET /static/styles.css HTTP/1.1" 304 -
127.0.0.1 - - [20/Aug/2024 17:40:53] "GET /static/pic.jpg HTTP/1.1" 304 -
```

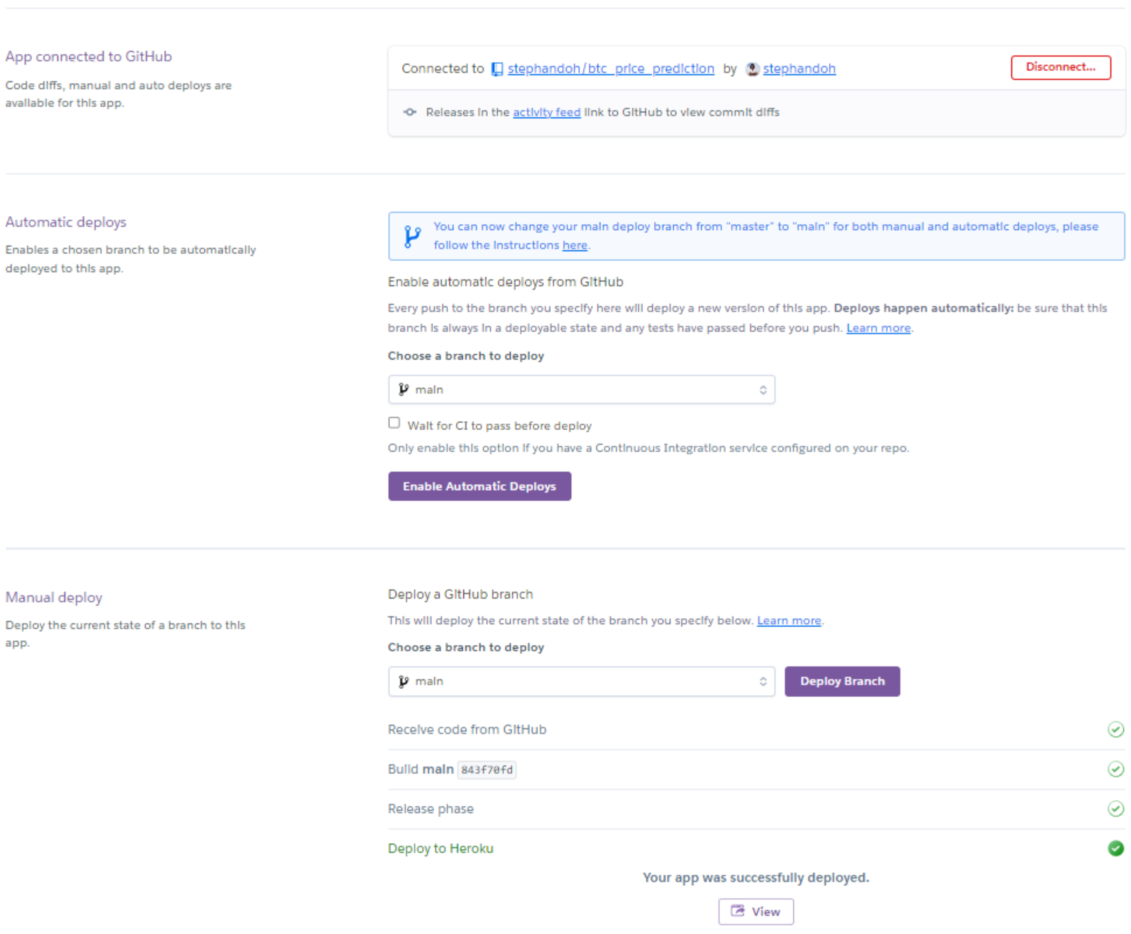
- *Description: This captures the terminal output showing the Flask app running.*

6. Final Webpage


Predict BTC Closing Price
Open Price:
High Price:
Low Price:
Predict
Predicted BTC closing price is $0.07064
Background image by Shubham Dhage on Unsplash

- *Description: This shows the final webpage with the app running in the browser.*

6. Deployment on Heroku


App connected to GitHub

Code diffs, manual and auto deploys are available for this app.

Connected to stephandoh/btc_price_prediction by stephandoh

Releases in the activity feed link to GitHub to view commit diffs

Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions here.

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch is always in a deployable state and any tests have passed before you push. Learn more.

Choose a branch to deploy

main

Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. Learn more.

Choose a branch to deploy

main

Receive code from GitHub ✓

Build **main** `843f70fd` ✓

Release phase ✓

Deploy to Heroku ✓

Your app was successfully deployed.

- *Description: This shows the deployment process on Heroku*
    - *Btcprediction app was first created*
    - *Github repository called btc_price_prediction was connected*
    - *Main branch was set for deployment*
    - *Application was successfully deployed*