00:00-28:35

Deploying a Machine Learning Model to Production: A Comprehensive Guide

Introduction

In this article, we will walk through the process of deploying a machine learning model to production. Specifically, we will focus on deploying a Bangalore home price prediction website to Amazon EC2, a cloud computing service provided by Amazon Web Services (AWS). By the end of this tutorial, you will have a fully functional website accessible through a URL, which can be mapped to your own domain. Let's dive into the details and explore the step-by-step process of deploying your code end-to-end.

Architecture Overview

Before we proceed with the deployment, let's take a brief look at the architecture of our application in the Amazon EC2 instance. The key components involved in the deployment are as follows:

1. Nginx: We will use Nginx as our lightweight web server. It will handle the incoming HTTP requests and serve the static files of our website.

2. Python Flask Server: Our Python Flask server will run on the same EC2 instance. It will handle the API requests and utilize the saved machine learning model to make predictions.

3. Reverse Proxy Setup: To route the API requests from Nginx to our Flask server, we will configure a reverse proxy setup on Nginx.

With this architecture in mind, we can now proceed to set up Nginx in the development environment and then replicate the setup on an Ubuntu server in our EC2 instance.

Setting Up Nginx on Windows

1. Download Nginx by visiting the official website or searching for it on Google.

2. Unzip the downloaded file and copy the extracted folder to a convenient location on your system (e.g., C:\Program Files).

3. Locate the Nginx configuration file (nginx.conf) within the Nginx folder and open it.

4. Verify that the Nginx server is running correctly by double-clicking the executable file. This will start the web server, and you should be able to access the default Nginx welcome page by navigating to `localhost` in your web browser.

5. Edit the configuration file (nginx.conf) and update the root directory to point to your Bangalore home prices page. Replace the existing root directory path with the path to your website's HTML file.

6. Save the configuration file and restart the Nginx server.

7. Verify the changes by refreshing the web page. Now, when you access `localhost`, your Bangalore home prices page should be loaded instead of the default Nginx welcome page.

8. Test the functionality of the website, including the prediction feature, by interacting with it in the browser. You can use the browser's developer tools (press F12) to monitor incoming and outgoing network requests for debugging purposes.

Configuring the Reverse Proxy for Flask Server

To enable the communication between Nginx and the Flask server running on port 5000, we need to set up a reverse proxy on Nginx. Follow these steps to configure the reverse proxy:

1. Open the Nginx configuration file (nginx.conf) mentioned earlier.

2. Add the following configuration section to the file:

```
location /api/ {
    proxy_pass http://localhost:5000/;
}
```

This configuration ensures that any HTTP request with a path starting with `/api/` is routed to the Flask server running on port 5000.

3. Save the configuration file and restart the Nginx server.

Running the Flask Server

Before we proceed to deploy our application on the cloud, we need to run the Flask server locally to ensure everything is working as expected. Follow these steps to run the server:

1. Open a command prompt or terminal and navigate to the server directory of your project.

2. Run the command `python server.py` to start the Flask server on port 5000.

3. Verify that the server is running correctly by refreshing the website. The website should now be fully functional, and the API requests should be successfully routed to the Flask server.

Deploying to Amazon EC2

To deploy our application on Amazon EC2, follow these steps:

1. Sign in to the AWS Management Console using your Amazon account.

2. Search for and select the Amazon EC2 service.

3. Click on "Launch Instance" to create a new virtual machine (EC2 instance).

4. Select the desired machine configuration, such as the operating system (e.g., Ubuntu) and the instance type. Choose the free tier if applicable.

5. Follow the default settings provided by Amazon, including storage configuration.

6. Configure the security group to allow incoming HTTP requests on port 80. This will ensure that your web server is accessible from the outside world.

7. Review the instance details and launch the instance.

8. Create a new key pair or select an existing one for secure SSH access to the EC2 instance.

9. Download the key pair file (.pem) and store it in a secure location on your local machine.

10. Launch the instance and wait for it to be initialized.

11. Connect to the EC2 instance using SSH. You can use a tool like Git Bash to establish an SSH connection by specifying the key pair file and the public DNS of the instance.

12. Once connected to the EC2 instance, use a tool like WinSCP to copy your codebase from your local machine to the EC2 instance.

13. Install Nginx on the EC2 instance by running the necessary commands to update the packages and install Nginx.

14. Update the Nginx configuration files (nginx.conf and bhp.conf) on the EC2 instance to reflect the correct paths and settings.

15. Restart the Nginx server to apply the changes.

16. Start the Flask server on the EC2 instance using the appropriate command.

17. Verify that the website is accessible by visiting the public DNS of your EC2 instance in a web browser.

18. Test the website's functionality, including the prediction feature, to ensure everything is working as expected.

Conclusion

Congratulations! You have successfully deployed your machine learning model to production by hosting a Bangalore home price prediction website on Amazon EC2. Throughout this tutorial, we covered the step-by-step process of setting up Nginx, configuring the reverse proxy for the Flask server, running the server locally, and deploying the application to the cloud. Remember to practice and experiment on your own to gain a deeper understanding of the deployment process. Feel free to refer to the provided GitHub repository for the code and additional instructions. If you encounter any issues or have any questions, don't hesitate to ask in the comments. Happy deploying!