National University
of computer and emerging sciences

CS4085 MLOps

Assignment #2

Members:

Fahad Baig - 19i-1883

Abdullah Hameed - 19i-1881

Code Explanation:

In our assignment, i191881_i191883.py is a Flask web application which has a machine learning model in it to predict the closing price of a stock. Here is a step-by-step breakdown of what the code is doing:

Importing the libraries:

- The Initial lines of code import the required libraries such as Scikit-Learn, Flask, Pandas, NumPy, and Joblib. These libraries are used to create the Flask web application and build the machine learning model to predict stock prices.
  Loading the dataset file:
- The next few lines of code load the stock price dataset from a CSV file named result.csv. We read it using the pandas library and it is split into training and testing sets in the ratio of 80:20.

We are defining input and target variables:

- The features of input for the machine learning model have been defined as 'Open', 'High', 'Low', and 'Volume'. The target variable is defined as 'Close' that needs to be predicted.
- Creating degree 2 of polynomial features.
- In the next step, a PolynomialFeatures object has been created with a degree of 2. This object is used to transform the input features into higher order polynomials, which is useful in capturing non-linear relationships between the target variable and the input features.

Training the machine learning model:

- The LinearRegression model is trained on the transformed input features and target variable using the model.fit() method. The model is used to predict the 'Close' values for the test data.
- Now we are saving the trained model:
- The trained model is saved using the joblib library, which is a utility and used for saving and loading Python objects.

Defining the Flask routes:

- The Flask application is defined with three different routes:
- '/' - this is the route where the home page of the web application shows the stock prices charts and predicted stock prices using the ML model we have trained.
- '/input' - this is the input route where the page shows a form where users can input the stock price features for which they want to predict the stock price.

- '/predict' - this is the input route where the page shows the prediction of stock prices. It takes the user input from the form and passes it to the trained ML model to predict the stock price.

Implementing the routes:

- The '/' route displays the chart of stock prices and predicted prices by rendering an HTML template with the chart data and predicted prices. The predicted prices are calculated using the trained ML model and they are passed as arguments to the template.
- The '/input' route displays an HTML page with a form where a user can input the stock price features.
- The '/predict' route takes the user input from the form and passes it to the trained ML model which predicts the stock price. The predicted price is then rendered in an HTML template along with an output message.

Running the Flask application:

- Finally, the application can be run using the run() method, we are specifying the host and port number. If the debug mode is enabled, Flask will print out and show detailed error messages and restart the application automatically if there are any code changes.

Front End:

Prediction.html:

The code of the HTML file contains a statement which is conditional that displays the predicted closed stock value if there is a prediction value available, otherwise it displays an output message.
The curly braces with percentage signs surrounding the code is a Jinja template syntax which allows the embedding of dynamic content within the HTML template. In this case, the template takes two variables to be passed to it: prediction and output. If the prediction variable is zero or not available, then the output variable will be displayed. Otherwise, if the prediction variable is available, then the predicted closed stock value will be displayed.

Index.html:

This file defines the layout dynamics and content of a web page for a stock exchange dashboard. The page displays a High Low Chart for a given set of data and provides some metrics related to the chart, including R-squared, Mean Squared Error, and Mean Absolute Error. The page also has a form with a "Next" button that allows the user to input data for prediction. This chart is created using the Highcharts library and the page is styled with the help of Bootstrap/CSS.

Input.html:

This html file allows users to input data of stock and predict its closing price. This form can take up to four input fields for the opening price of the stock, stock volume, maximum stock price, and minimum stock price. There is a "Predict" button in the page that, when clicked, will send the data to a server-side script for prediction and processing further. The form is created by CSS styling to make it look visually appealing and user-friendly.

Setup:

JENKINS:

For this assignment we used the cloud variant of Jenkins of AWS (Amazon Web Services). Using the cloud, we are able to run our jobs and our applications from anywhere if the Amazon ec2 instance was up and running. Due to the free availability of AWS, we have to change the webhook of GitHub every time as the public IP is not static. For AWS we used windows for that we needed the PUTTY.

What we did in Jenkins is we have created a job that tells us whenever a push happens it will notify us. Whenever a commit is made in GitHub it is reflected on Jenkins due to the webhook.

Link to our Repository:

https://github.com/NUCES-ISB/i191881_i191883_Mlops_A-2