CS4085 MLOps

Assignment #2

Name:

Mahreen Athar (19i-1712)

Adeen Ayub Vine (19i-0553)

Submitted to:

Sir Hussain Shahbaz

Introduction:

The livestream data used as input for training our chosen model is Google stock known as “GOOG” from the year of 2019 to the present date which we garnered through yfinance. The model chosen by us is Linear Regression. The trained model is wrapped in a Flask application which will show the live data along with the model’s prediction in regards to the data provided to it. All of the files were pushed to a Github repository and synced with Jenkins for the delivery phase. The final step included making a docker image of the above application.

Procedure:

We began by creating the web pages using HTML and CSS.

Then we loaded our live stream data of Google stock known as “GOOG”by using yfinance. We stored this dataset in a variable called “data”.

Loading the dataset

```
ticker = yf.Ticker("GOOG")
data = ticker.history(start=datetime.date(2019, 1, 1), end=datetime.date.today())
data.reset_index(inplace=True)
```

We split this data into the ratio of 80:20. With 80% of the data being used for training our model and 20% of the data being used for testing our model’s prediction accuracy.

Splitting Data

80% training, 20% testing

```
train_size = int(len(data) * 0.8)
train_data = data.iloc[:train_size]
test_data = data.iloc[train_size:]
```

We defined our model’s input features to be “Open, High, Low, and Volume” and defined our model’s target variable to be “Close”.

Define input features

Open, High, Low, and Volume

```
X_train = train_data[['Open', 'High', 'Low', 'Volume']]
X_test = test_data[['Open', 'High', 'Low', 'Volume']]
```

Define target variable (Close)

```
y_train = train_data['Close']
y_test = test_data['Close']
```

After this, we trained our model by using Linear Regression and stored this in a pickle file using joblib.

```
model = joblib.load('model.pkl')
```

We load this pickle file in Flask so that we may use our trained model in the Flask application. We define our routes and our pre-made web page as templates in our Flask application.

Our Github repository is synced with Jenkins through the use of a webhook for the delivery phase. After the completion of the application a docker image was created of the application and the image was also pushed to the repository.