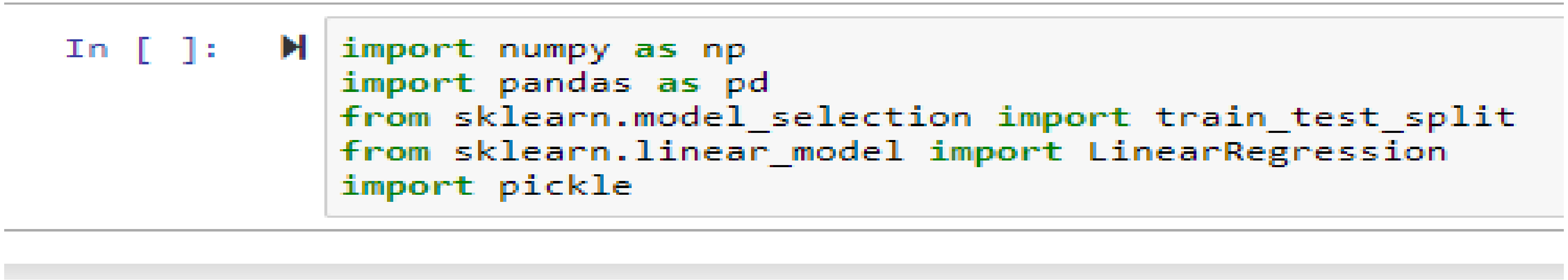
Milestone 5: Web Flask Deployment

Step 1: Building a regression model for share price prediction

- Import the following library

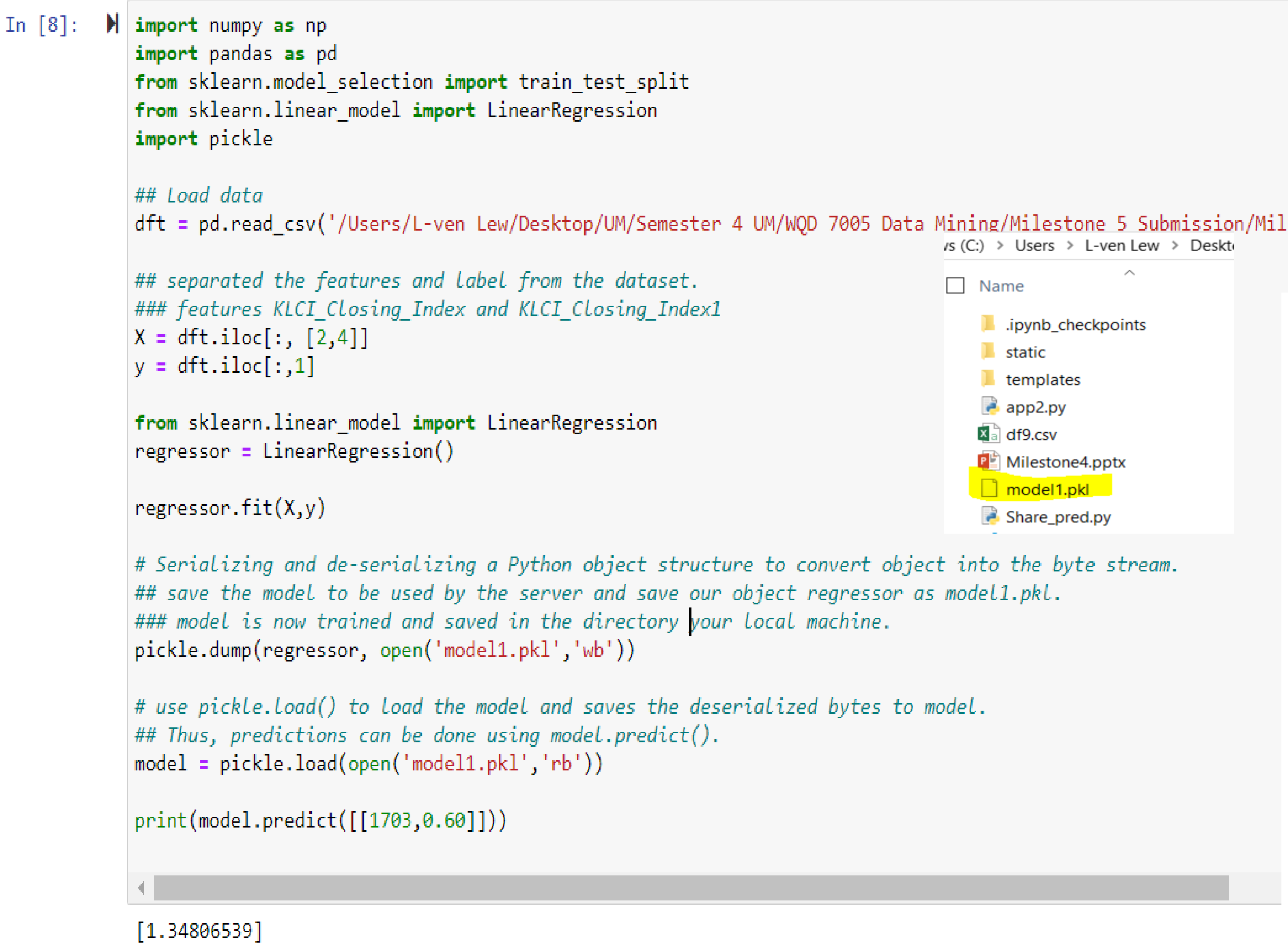


```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import pickle
```

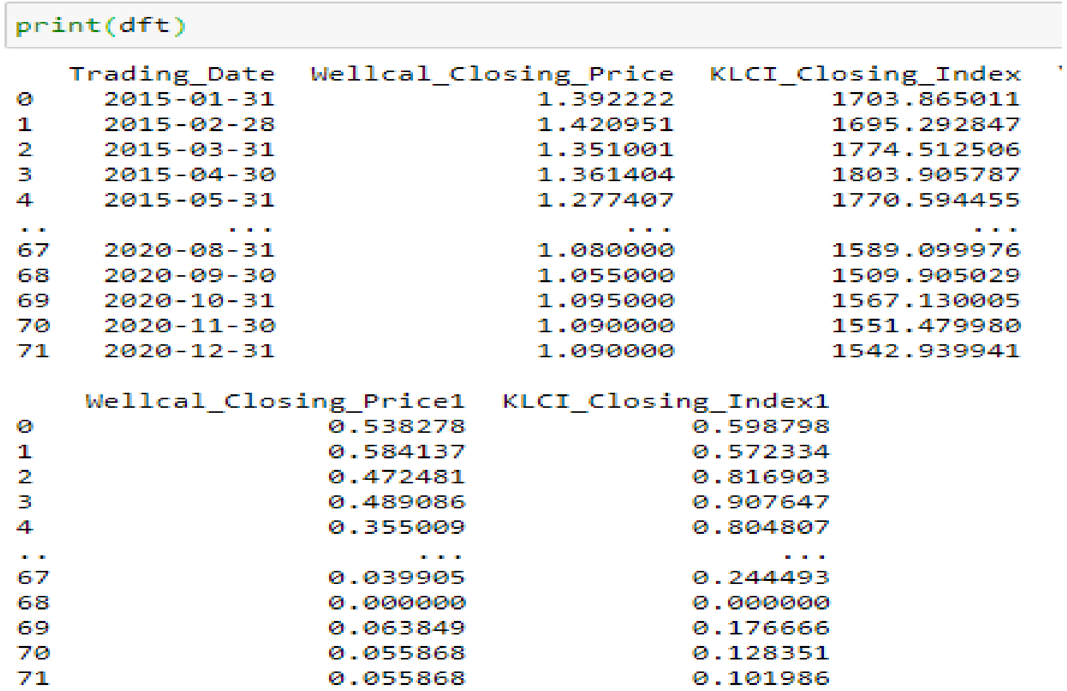
import numpy as np – To manipulate the matrices
import pandas as pd – To manipulate the data
from sklearn.model_selection import train_test_split - split data into training and testing set
from sklearn.linear_model import LinearRegression – build linear regression model to train the predictive model
import pickle - to save our trained model to the disk

Milestone 5: Web Flask Deployment

Continued from Step 1: Building a regression model for share price prediction



```
In [8]:  import numpy as np
         import pandas as pd
         from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LinearRegression
         import pickle

         ## Load data
         dft = pd.read_csv('/Users/L-ven Lew/Desktop/UM/Semester 4 UM/WQD 7005 Data Mining/Milestone 5 Submission/Milestone 5 final su

         ## separated the features and label from the dataset.
         ### features KLCI_Closing_Index and KLCI_Closing_Index1
         X = dft.iloc[:, [2,4]]
         y = dft.iloc[:,1]

         from sklearn.linear_model import LinearRegression
         regressor = LinearRegression()

         regressor.fit(X,y)

         # Serializing and de-serializing a Python object structure to convert object into the byte stream.
         ## save the model to be used by the server and save our object regressor as model1.pkl.
         ### model is now trained and saved in the directory your local machine.
         pickle.dump(regressor, open('model1.pkl','wb'))

         # use pickle.load() to load the model and saves the deserialized bytes to model.
         ## Thus, predictions can be done using model.predict().
         model = pickle.load(open('model1.pkl','rb'))

         print(model.predict([[1703,0.60]]))

[1.34806539]
```

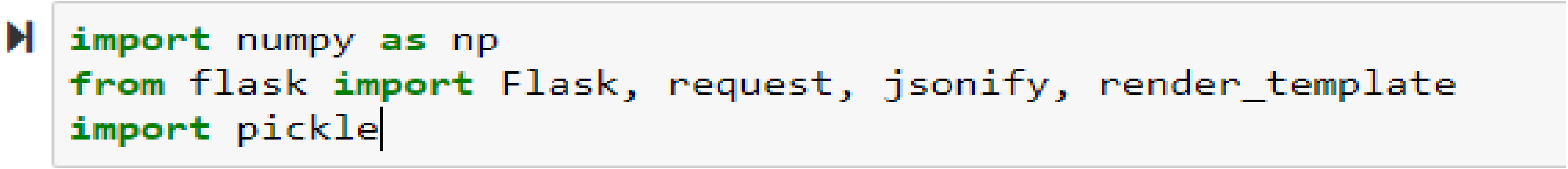


```
print(dft)

   Trading_Date  Wellcal_Closing_Price  KLCI_Closing_Index
0    2015-01-31               1.392222         1703.865011
1    2015-02-28               1.420951         1695.292847
2    2015-03-31               1.351001         1774.512506
3    2015-04-30               1.361404         1803.905787
4    2015-05-31               1.277407         1770.594455
..          ...                    ...                 ...
67   2020-08-31               1.080000         1589.099976
68   2020-09-30               1.055000         1509.905029
69   2020-10-31               1.095000         1567.130005
70   2020-11-30               1.090000         1551.479980
71   2020-12-31               1.090000         1542.939941

    Wellcal_Closing_Price1  KLCI_Closing_Index1
0                 0.538278             0.598798
1                 0.584137             0.572334
2                 0.472481             0.816903
3                 0.489086             0.907647
4                 0.355009             0.804807
..                     ...                  ...
67                0.039905             0.244493
68                0.000000             0.000000
69                0.063849             0.176666
70                0.055868             0.128351
71                0.055868             0.101986
```

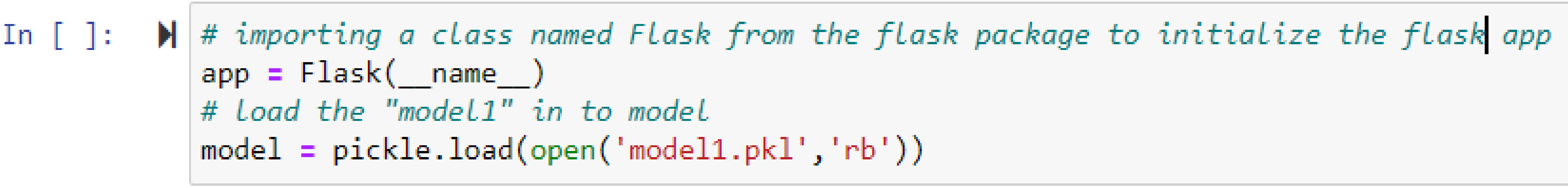
Milestone 5: Web Flask Deployment

Step 2: flask-app Set-up in the localhost

- Load the library below：

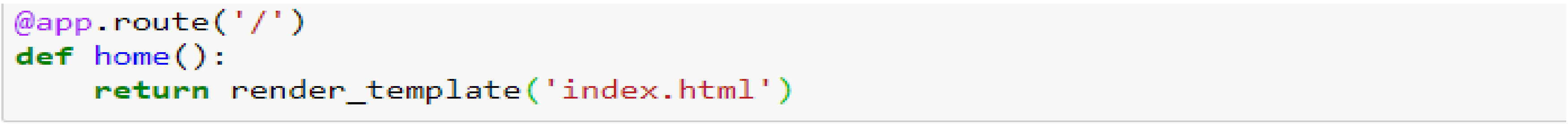


```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
```



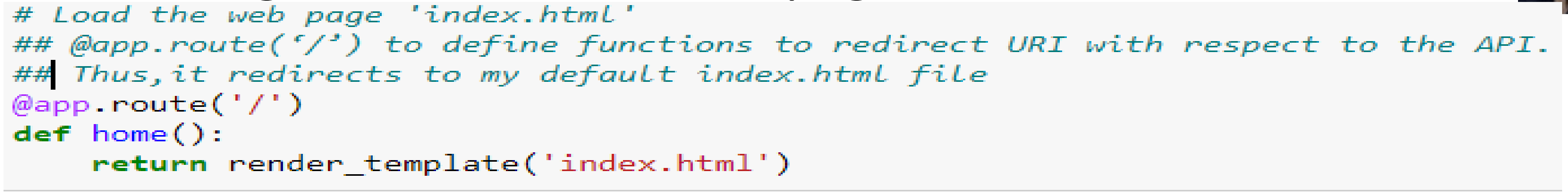
```
In [ ]:  # importing a class named Flask from the flask package to initialize the flask app
          app = Flask(__name__)
          # load the "model1" in to model
          model = pickle.load(open('model1.pkl','rb'))
```

- Load the web page @ index.html



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

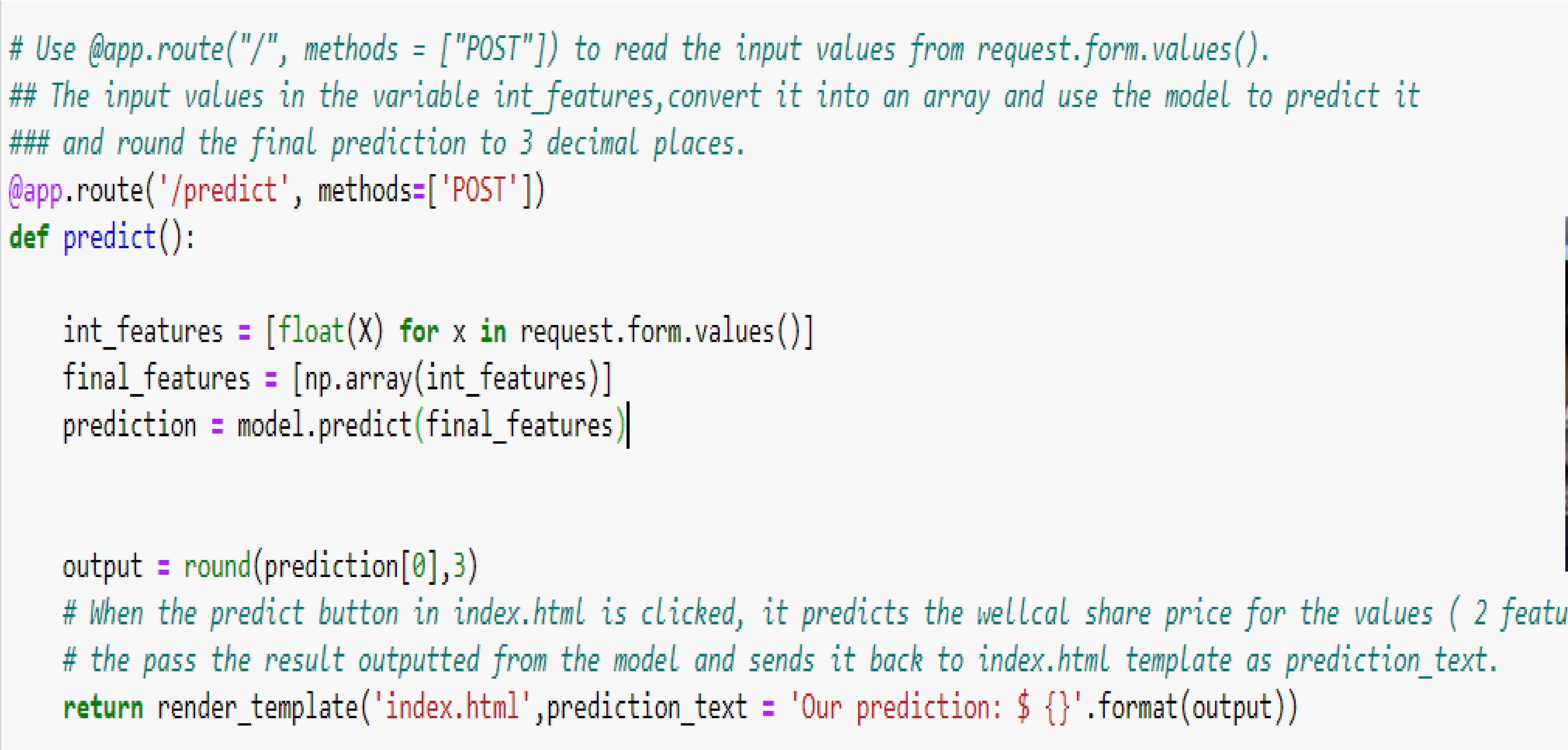
- Redirecting the API to the home page index.html



```
# Load the web page 'index.html'
## @app.route(‘/’) to define functions to redirect URI with respect to the API.
## Thus,it redirects to my default index.html file
@app.route('/')
def home():
    return render_template('index.html')
```

Milestone 5: Web Flask Deployment

Step 3: Redirecting the API to predict the result (Wellcal Share price based on KLCI Index) using the regression prediction model:

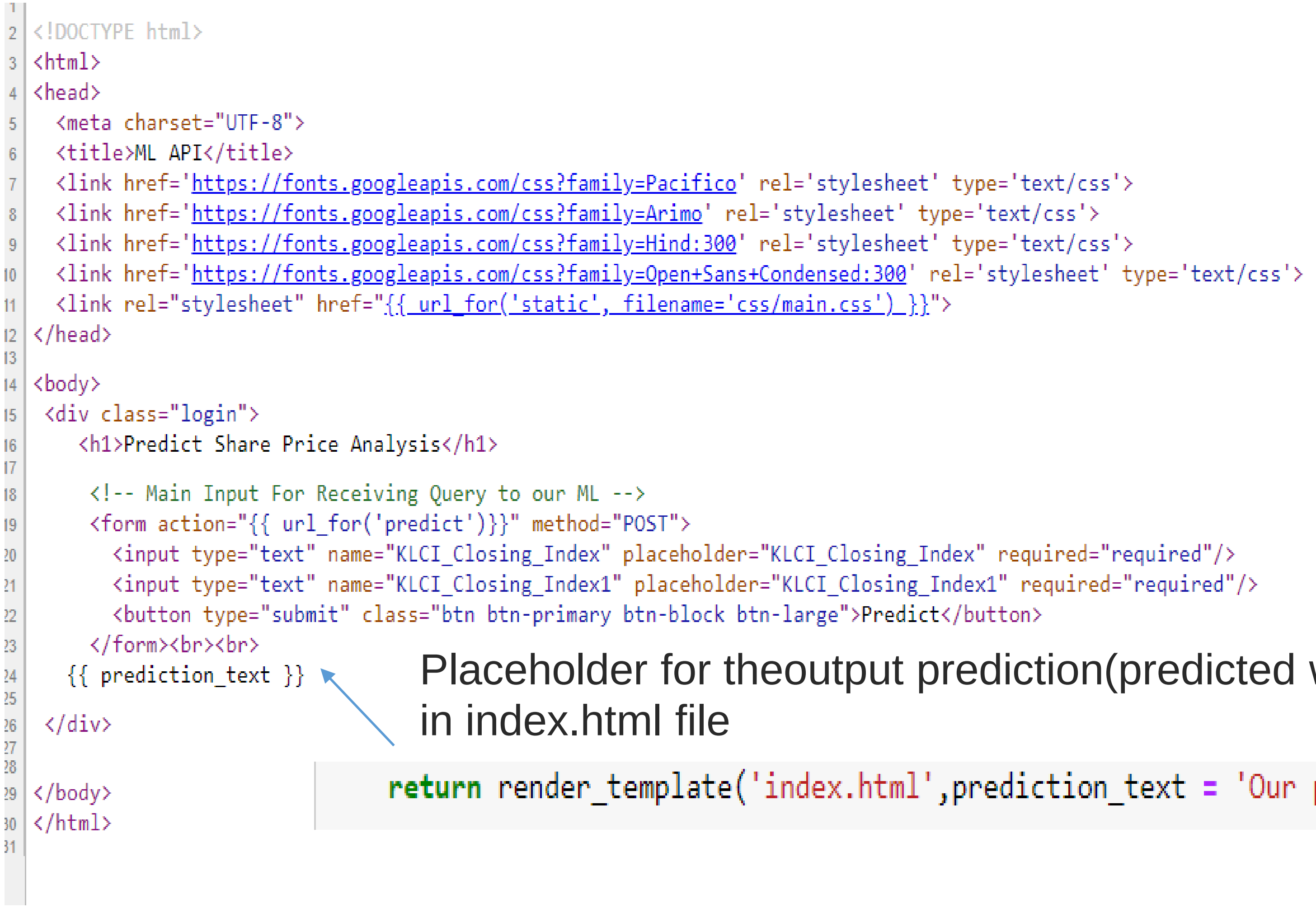


```
# Use @app.route("/", methods = ["POST"]) to read the input values from request.form.values().
## The input values in the variable int_features,convert it into an array and use the model to predict it
### and round the final prediction to 3 decimal places.
@app.route('/predict', methods=['POST'])
def predict():

    int_features = [float(X) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)



    output = round(prediction[0],3)
    # When the predict button in index.html is clicked, it predicts the wellcal share price for the values ( 2 features)
    # the pass the result outputted from the model and sends it back to index.html template as prediction_text.
    return render_template('index.html',prediction_text = 'Our prediction: $ {}'.format(output))
```

Milestone 5: Web Flask Deployment

Continued from Step 3: Redirecting the API to predict the result (Wellcal Share price based on KLCI Index) using the regression prediction model:

- Look at the html file



```html

<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>ML API</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
  <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
  <link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
  <link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
  <link rel="stylesheet" href="{{ url_for('static', filename='css/main.css') }}">
</head>

<body>
 <div class="login">
    <h1>Predict Share Price Analysis</h1>

     <!-- Main Input For Receiving Query to our ML -->
     <form action="{{ url_for('predict')}}" method="POST">
       <input type="text" name="KLCI_Closing_Index" placeholder="KLCI_Closing_Index" required="required"/>
       <input type="text" name="KLCI_Closing_Index1" placeholder="KLCI_Closing_Index1" required="required"/>
       <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
     </form><br><br>
   {{ prediction_text }}

 </div>


</body>
</html>

```

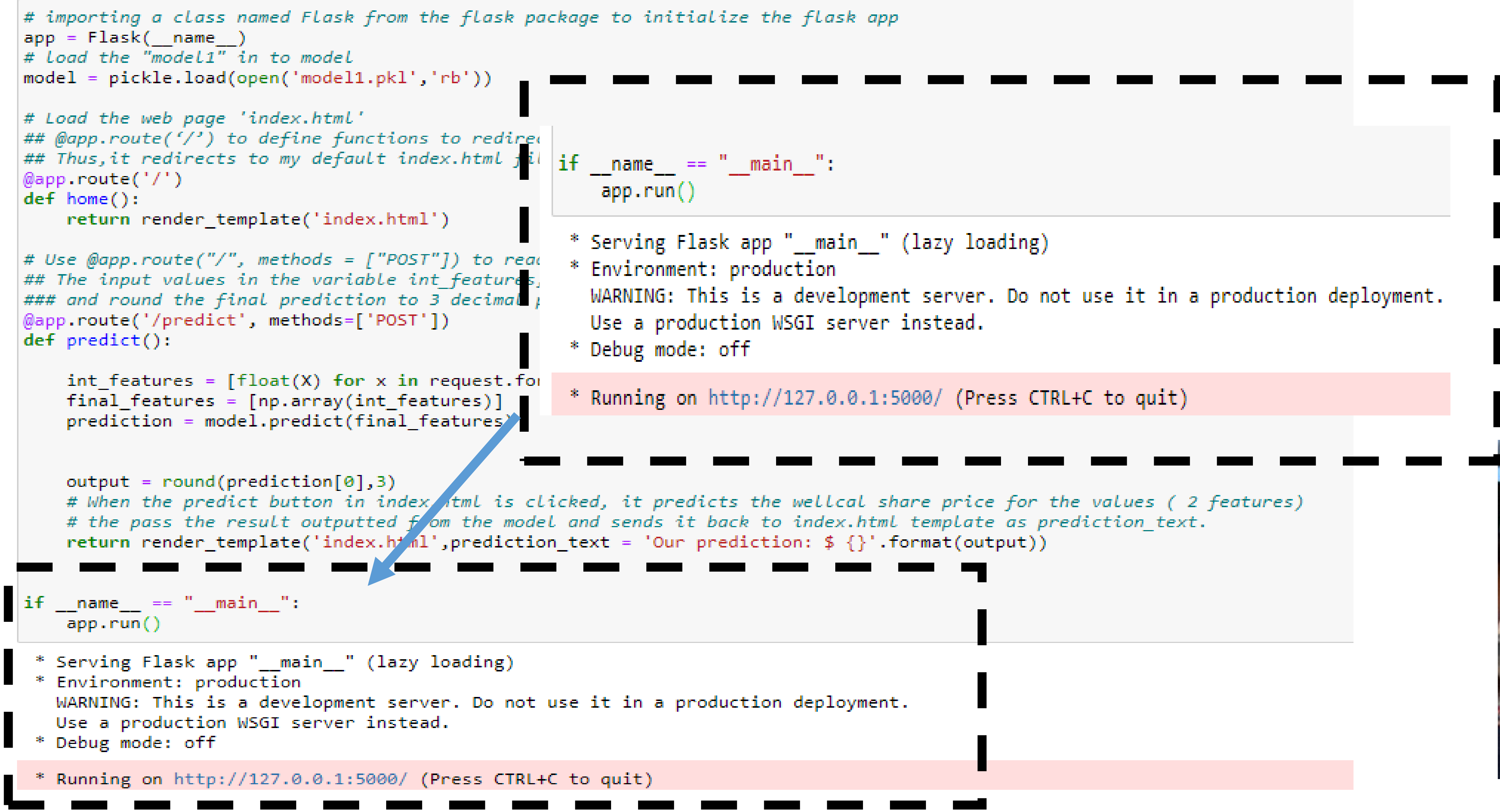
Placeholder for theoutput prediction(predicted wellcal share price) from the model in index.html file

```python
    return render_template('index.html',prediction_text = 'Our prediction: $ {}'.format(output))
```

Milestone 5: Web Flask Deployment

Step 4: Start the flask server

- Call app.run() and run the web page hosted on local computer.



```
# importing a class named Flask from the flask package to initialize the flask app
app = Flask(__name__)
# Load the "model1" in to model
model = pickle.load(open('model1.pkl','rb'))

# Load the web page 'index.html'
## @app.route('/') to define functions to redire
## Thus,it redirects to my default index.html fil
@app.route('/')
def home():
    return render_template('index.html')

# Use @app.route("/", methods = ["POST"]) to read
## The input values in the variable int_features,
### and round the final prediction to 3 decimal p
@app.route('/predict', methods=['POST'])
def predict():

    int_features = [float(X) for x in request.fo
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)


    output = round(prediction[0],3)
    # When the predict button in index.html is clicked, it predicts the wellcal share price for the values ( 2 features)
    # the pass the result outputted from the model and sends it back to index.html template as prediction_text.
    return render_template('index.html',prediction_text = 'Our prediction: $ {}'.format(output))
```

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

```
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

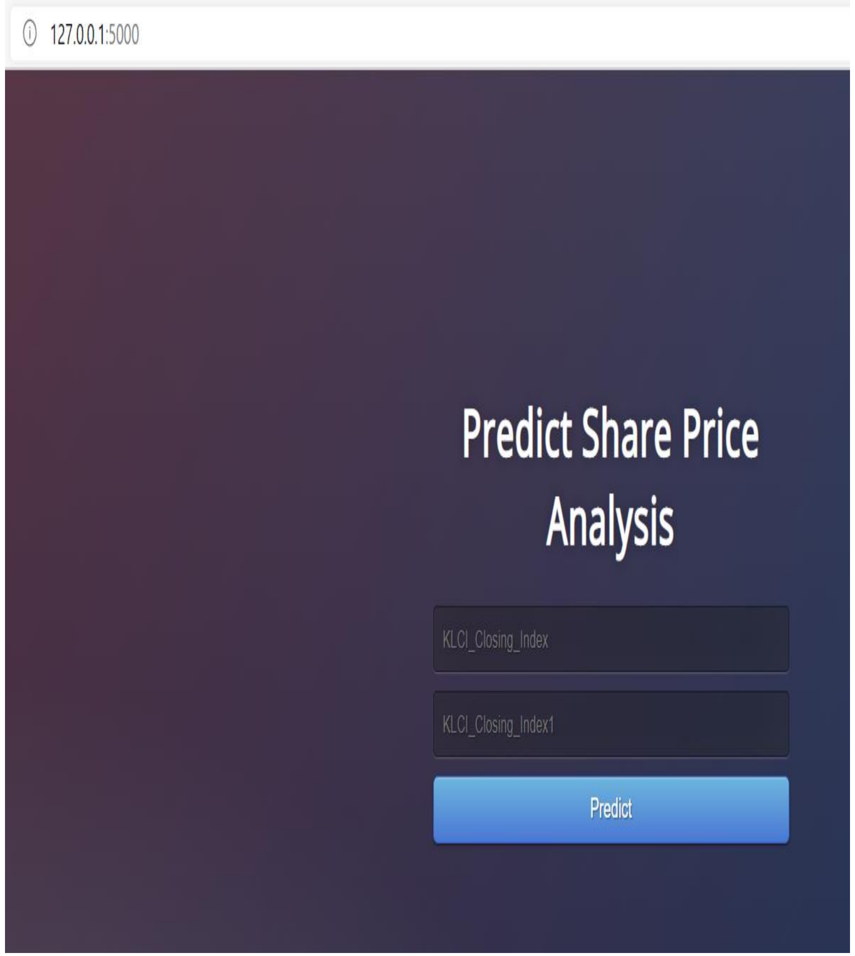
```
if __name__ == "__main__":
    app.run()
```

```
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

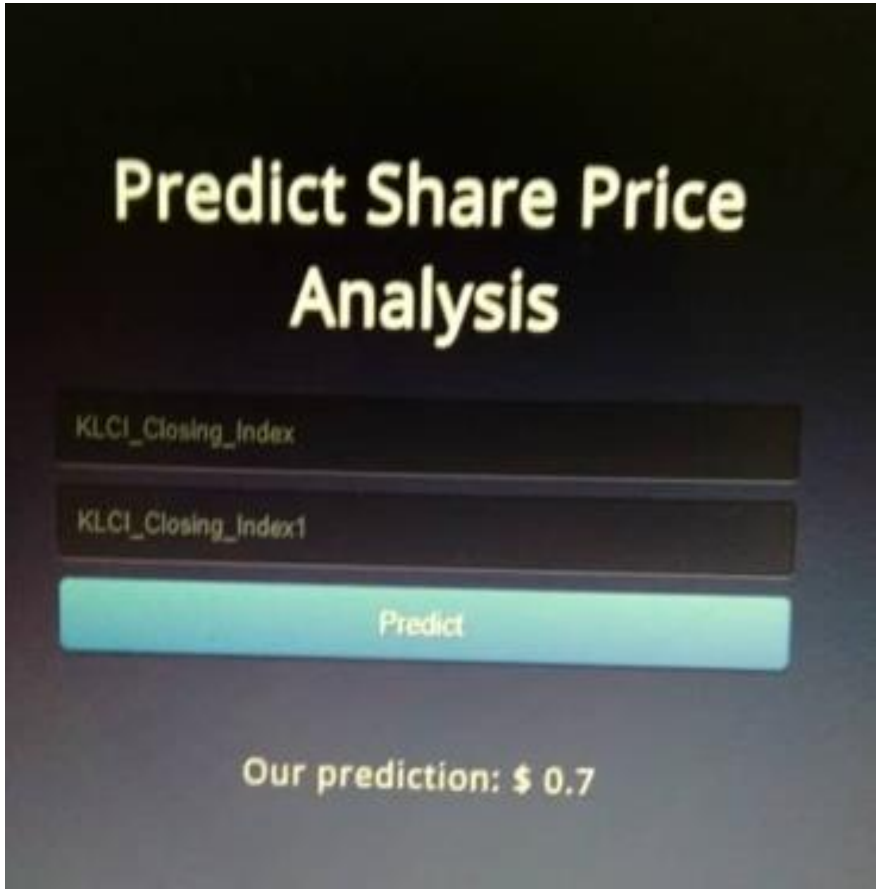
Milestone 5: Web Flask Deployment

Continued from Step 4: Start the flask server

- Web Flask app is successfully launched



Predict Share Price Analysis



Predict Share Price Analysis

Our prediction: $ 0.7