

Final code

Team ID	PNT2022TMID13501
Project Name	Project - Crude Oil Price Prediction

Index.html:

```
<html>

<head>

<title>Home</title>

<meta charset="utf-8">

  <meta name="viewport" content="width=device-width, initial-scale=1">

<style>

body

{

background-image:url("https://etimg.etb2bimg.com/photo/94227420.cms");

background-position: center;

font-family:Times-new roman;

background-size:cover;

margin-top:40px;

}

.pd{

padding-bottom:100%;}

.navbar

{

margin-left:10px;

padding:10px;

background-color:hsl(180, 96%, 52%);

font-family:'Roboto',sans-serif;

font-style: italic;
```

```
border-radius:30px;
font-size:30px;
box-sizing: border-box;
max-width: 18%;
text-align:right;
```

```
}
```

```
a
```

```
{
```

```
color:grey;
float:right;
text-decoration:none;
font-style:normal;
padding-right:20px;
```

```
}
```

```
a:hover{
```

```
background-color:black;
color:white;
border-radius:15px;0
font-size:30px;
padding-left:10px;
```

```
}
```

```
p
```

```
{
```

```
color:turquoise;
font-style:italic;
font-size:30px;
text-align: left;
padding-left: 500px;
```

```

text-align: justify;
}
</style>
</head>
<body>
<div class="navbar">
<a href="/predict" >Predict</a>
<a href="/">Home</a>
<br>
</div>
<br>
<center><b class="pd"><font color="200FF3" size="15" font-family="Comic Sans MS" >Crude Oil Price
Prediction</font></b></center><br><br>
</body>
</html>

```

Web.html:

```

<html>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <style>
div.header{
  top: 0;
  position: fixed;
  padding-left: 400px;}
div.header1{
  top:20;
  position: fixed;
  padding-left: 490px;

```

```
}
```

```
*{
```

```
margin:0;
```

```
padding:0;
```

```
border:0;
```

```
outline:0;
```

```
text-decoration:none;
```

```
font-family:montserrat;
```

```
}
```

```
.navbar
```

```
{
```

```
margin-left:10px;
```

```
padding:10px;
```

```
background-color:hsl(180, 96%, 52%);
```

```
font-family:'Roboto',sans-serif;
```

```
font-style: italic;
```

```
border-radius:30px;
```

```
font-size:30px;
```

```
box-sizing: border-box;
```

```
max-width: 18%;
```

```
text-align:center;
```

```
}
```

```
a:hover{
```

```
background-color:black;
```

```
color:white;
```

```
border-radius:16px;
```

```
font-size:30px;
```

```
padding:10px;
```

```
}
```

```
body
```

```
{
```

```
background-image:url('https://wallpapercave.com/wp/wp9553498.jpg');
```

```
background-position: center;
```

```
font-family:sans-serif;
```

```
background-size:cover;
```

```
margin-top:40px;
```

```
}
```

```
.main input[type="text"],.main input[type="text"],.main input[type="text"],.main  
input[type="text"],.main input[type="text"],.main input[type="text"],.main input[type="text"]{
```

```
border:0;
```

```
background:none;
```

```
display:block;
```

```
margin:20px auto;
```

```
text-align:center;
```

```
border:2px solid white;
```

```
padding:15px 3px;
```

```
width:400px;
```

```
outline:none;
```

```
color:white;
```

```
border-radius:100px;
```

```
transition:0.25s;
```

```
font-size:20;
```

```
}
```

```
.bor{
```

```
border:0;

background:none;

display:block;

margin:20px auto;

text-align:center;

border:2px solid white;

padding:10px 3px;

width:500px;

outline:none;

color:black;

transition:0.25s;

}
```

```
.main input[type="text"]:focus,.main input[type="text"]:focus,.main input[type="text"]:focus,.main
input[type="text"]:focus,.main input[type="text"]:focus,.main input[type="text"]:focus,.main
input[type="text"]:focus{

width:280px;

border-color:black;

}
```

```
.logbtn{

display:block;

width:35%;

height:50px;

border:none;

border-radius:24px;

background:linear-gradient(120deg,#3498db,#8e44ad,#3498db,#8e44ad);

background-size:200%;

color:black;

outline:none;

cursor:pointer;

}
```

```
    transition:.5s;
    font-size:25;
}
```

```
input::placeholder{
    color:white;
}
```

```
.bottom-text{
    margin-top:60px;
    text-align:center;
    font-size:13px;
}
```

```
</style>
```

```
<body>
```

```
<div class="navbar">
```

```
<a href="index.html">Home</a>
```

```
<br>
```

```
</div>
```

```
    <center><div><font color="Powderblue" font-family="sans-serif" size=8 ><b>Crude Oil Price
Prediction</b></font></div></center>
```

```
<br><br><br><br>
```

```
<form class="main" action="/login" method="post">
```

```
<br>
```

```

        <font size=20><input type="text" name="year1" placeholder="Enter previous 10th day
price"/></font>

        <font size=20><input type="text" name="year2" placeholder="Enter previous 9th day
price"/></font>

        <font size=20><input type="text" name="year3" placeholder="Enter previous 8th day
price"/></font>

        <font size=20><input type="text" name="year4" placeholder="Enter previous 7th day
price"/></font>

        <font size=20><input type="text" name="year5" placeholder="Enter previous 6th day
price"/></font>

        <font size=20><input type="text" name="year6" placeholder="Enter previous 5th day
price"/></font>

        <font size=20><input type="text" name="year7" placeholder="Enter previous 4th day
price"/></font>

        <font size=20><input type="text" name="year8" placeholder="Enter previous 3th day
price"/></font>

        <font size=20><input type="text" name="year9" placeholder="Enter previous 2nd day
price"/></font>

        <font size=20><input type="text" name="year10"placeholder="Enter previous 1st day
price"/></font>

        <center><input type="submit" class="logbtn" value="Predict"></center>

        <div class="bor"><b><font color="white" size=5>Next Day Price</font></b></div>

    </form>

</div>

</body>

</html>

```

Python:

```

import numpy as np

from flask import Flask,render_template,request

from tensorflow.keras.models import load_model

```



```

app=Flask(__name__)
model=load_model('crude.h5')

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

@app.route('/about')
def home1():
    return render_template("index.html")

@app.route('/predict')
def home2():
    return render_template("web.html", showcase="")

@app.route('/login',methods=['POST'])
def login():
    x_input=[]
    for i in request.form:
        x_input.append(float(request.form[i]))
    x_input=np.array(x_input).reshape(1,-1)
    temp_input=list(x_input)
    temp_input=temp_input[0].tolist()
    lst_output=[]
    n_steps=10
    i=0
    while(i<1):
        if(len(temp_input)>10):
            x_input=np.array(temp_input[1:])
            print("{}day input{}".format(i,x_input))
            x_input=x_input.reshape(1,-1)
            x_input=x_input.reshape((1,n_steps,1))
            yhat=model.predict(x_input,verbose=0)

```

```
print("{} day output {}".format(i,yhat))
```

```
temp_input.extend(yhat[0].tolist())
```

```
temp_input=temp_input[1:]
```

```
lst_output.extend(yhat.tolist())
```

```
i=i+1
```

```
else:
```

```
x_input=x_input.reshape((1,n_steps,1))
```

```
yhat=model.predict(x_input,verbose=0)
```

```
print(yhat[0])
```

```
temp_input.extend(yhat[0].tolist())
```

```
print(len(temp_input))
```

```
lst_output.extend(yhat.tolist())
```

```
i=i+1
```

```
return render_template("web.html",showcase='Next Day Predicted Price Is:'+str(lst_output[0][0]))
```

```
if __name__=='main':
```

```
app.run(debug=True,port=5000)
```

The screenshot shows the Spyder Python IDE interface. The left pane displays the code for `app.py`, which is a Flask web application. The code includes imports for `numpy`, `Flask`, `render_template`, `request`, `load_model`, and `load_model`. It defines a `home` function that returns `render_template("index.html")` and a `login` function that returns `render_template("web.html", showcase="")`. The `login` function also includes a `while` loop that processes a request form, appends the input to `temp_input`, and prints the output. The right pane shows the console output, which includes a message about the TensorFlow binary being optimized with oneAPI Deep Neural Network Library (oneDNN) and a warning about the production environment. The status bar at the bottom indicates the current file is `app.py` and the environment is `conda: base (Python 3.9.12)`.

```
1 import numpy as np
2 from flask import Flask,render_template,request
3 from tensorflow.keras.models import load_model
4 app=Flask(__name__)
5 model=load_model('crude.h5')
6 @app.route('/')
7 def home():
8     return render_template("index.html")
9 @app.route('/about')
10 def home1():
11     return render_template("index.html")
12 @app.route('/predict')
13 def home2():
14     return render_template("web.html", showcase="")
15 @app.route('/Login',methods=['POST'])
16 def login():
17     x_input=[]
18     for i in request.form:
19         x_input.append(float(request.form[i]))
20     x_input=np.array(x_input).reshape(1,-1)
21     temp_input=list(x_input)
22     temp_input=temp_input[0].tolist()
23     lst_output=[]
24     n_steps=10
25     i=0
26     while(i<1):
27         if(len(temp_input)>10):
28             x_input=np.array(temp_input[1:])
29             print("{} day input {}".format(i,x_input))
30             x_input=x_input.reshape(1,-1)
31             x_input=x_input.reshape((1,n_steps,1))
32             yhat=model.predict(x_input,verbose=0)
33             print("{} day output {}".format(i,yhat))
34             temp_input.extend(yhat[0].tolist())
35             temp_input=temp_input[1:]
36             i=i+1
37         else:
38             x_input=x_input.reshape((1,n_steps,1))
39             yhat=model.predict(x_input,verbose=0)
40             print(yhat[0])
41             temp_input.extend(yhat[0].tolist())
42             print(len(temp_input))
43             lst_output.extend(yhat.tolist())
44             i=i+1
45     return render_template("web.html",showcase='Next Day Predicted Price Is:'+str(lst_output[0][0]))
46 if __name__=='main':
47     app.run(debug=True,port=5000)
```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in **Preferences > Help**.

New to Spyder? Read our [tutorial](#)

failed call to cuInit: UNKNOWN ERROR (303)

2022-11-18 10:58:02.197496: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: MSI

2022-11-18 10:58:02.197573: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: MSI

2022-11-18 10:58:02.199664: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2 To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

* Serving Flask app "app" (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: on

* Restarting with watchdog (windowsapi)

LSP Python: ready conda: base (Python 3.9.12) Line 20, Col 44 ASCII CRLF RW Mem 75%

```
Anaconda Prompt (Anaconda3) - python app.py

(base) C:\Users\navee>cd C:\Users\navee\OneDrive\Desktop\ibm\Flask

(base) C:\Users\navee\OneDrive\Desktop\ibm\Flask>python app.py
2022-11-18 10:59:38.481532: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlderror: cudart64_110.dll not found
2022-11-18 10:59:38.481840: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
2022-11-18 10:59:40.474957: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'nvcuda.dll'; dlderror: nvcuda.dll not found
2022-11-18 10:59:40.475285: W tensorflow/stream_executor/cuda/cuda_driver.cc:263] failed call to cuInit: UNKNOWN ERROR (383)
2022-11-18 10:59:40.479945: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: MSI
2022-11-18 10:59:40.480640: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
* Serving Flask app "app" (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: on
* Restarting with watchdog (windowsapi)
2022-11-18 10:59:41.891538: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlderror: cudart64_110.dll not found
2022-11-18 10:59:41.891874: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
2022-11-18 10:59:43.865788: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'nvcuda.dll'; dlderror: nvcuda.dll not found
2022-11-18 10:59:43.865987: W tensorflow/stream_executor/cuda/cuda_driver.cc:263] failed call to cuInit: UNKNOWN ERROR (383)
2022-11-18 10:59:43.870168: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: MSI
2022-11-18 10:59:43.870528: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: MSI
2022-11-18 10:59:43.871023: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
* Debugger is active!
* Debugger PIN: 342-672-433
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

