Crude Oil Price Prediction- Using LSTM

1. Introduction

1.1 Overview

Crude oil price prediction using Long Short-Term Memory (LSTM) is a machine learning technique that involves training an LSTM neural network on historical crude oil price data to make predictions about future prices.

The LSTM model is specifically designed to handle sequential data, making it well-suited for time-series prediction tasks such as crude oil price forecasting. The model takes in historical price data as input and learns to identify patterns and trends in the data.

To train an LSTM-based crude oil price prediction model, several steps are involved, including data preprocessing, feature engineering, model training, model evaluation, and prediction.

Data preprocessing involves cleaning and normalizing the historical price data and dividing it into training, validation, and test sets. Feature engineering involves selecting and transforming the input features that will be fed into the LSTM model.

The LSTM model is then trained on the training set using backpropagation and gradient descent. The model is tuned using a combination of hyperparameter optimization and cross-validation.

The model's performance is evaluated on the validation set, and the model is adjusted as needed to improve its accuracy. Once the model has been trained and evaluated, it can be used to make predictions about future crude oil prices based on new input data.

Then the input of the past ten days' price of the crude oil is given by the user through a web app and the predicted price is published.

1.2 Purpose

Crude oil price prediction using LSTM can have several practical applications. Some of these include:

Trading: Traders can use the predictions generated by the LSTM model to make informed decisions about buying or selling crude oil futures contracts or other financial instruments that are sensitive to crude oil prices.

Risk management: Companies that are exposed to crude oil price risks, such as oil producers, refiners, and airlines, can use the predictions generated by the LSTM model to manage their exposure to price fluctuations and to hedge their positions.

Investment: Investors can use the predictions generated by the LSTM model to make informed decisions about investing in oil-related assets such as energy stocks, exchange-traded funds (ETFs), or mutual funds.

Strategic planning: Governments, policymakers, and industry analysts can use the predictions generated by the LSTM model to plan and forecast future energy supply and demand, as well as to develop strategies for reducing the impact of price volatility on the economy.

Overall, the use of LSTM-based crude oil price prediction can provide valuable insights and information for various stakeholders involved in the crude oil industry, and it can help them make better-informed decisions and strategies

2. Literature Survey

2.1 Existing problem

Modeling the price of oil is difficult, because of many fluctuating variables over time. As can be seen in, the oil price can dramatically change over a short time, which makes it very difficult to predict. Oil demand and supply are quite inelastic in a short time, meaning when the demand for oil exceeds supply, the price will rise extremely high. On the other hand, the oil price is affected heavily by political turbulence.

In general, there are 3 different approaches to forecasting crude oil prices. They are

- Long term
- Medium term
- Short term

This project focuses on the short-term forecasting of future oil prices, specifically WTI crude oil. Forecasting such unpredictable economic series stays one of the main challenges for econometricians. In the literature, there are several different models used

In the literatures, there are several different models used 4 to predict and forecast short term oil prices. Historically linear structural models have not performed well for oil price forecasting and nonlinear time series models have performed much better in forecasting oil prices (LAM, 2013),(Bosler, 2010) and (Moshiri and Foroutan, 2006). F. Bosler examined a time series approach, which includes linear and nonlinear time series analysis and also structural models (LAM, 2013). He compared linear ARIMA model and neural network autoregressive model for nonlinear time series analysis and confrmed that thenonlinear models forecasts perform the better and follow the volatility of the oil price. In another work, D. Lam modeled oil prices based on univariate time series using the Box-Jenkins methodology. Based on the ACF and PACF techniques ARIMA model was chosen, and followed by GARCH and APARCH as model residuals (Bosler, 2010). He also built a regression model to compare with his nonlinear model.

The regression model was based on eight explanatory variables, including production, consumption, net import, ending stock, refnery utilization rate, U.S. interest rate, NYMEX oil futures contract 4 and S&P 500 index. But at the end, the conclusion was that GARCH and APARCH perform the best. S. Moshiri et. al. furthermore proposed another nonlinear model to forecast daily crude oil futures prices listed in NYMEX. They used a nonlinear and flexible artificial neural network (ANN) model to forecast the series and concluded it will improve forecasting accuracy. This work claims that “If the data generating process is nonlinear, applying linear models could result in large forecast errors.

2.2 Proposed Solution

Since the linear model does not perform well on time series prediction so we used a recurrent neural network called LSTM. Typically recurrent neural networks (RNN) have short term memory in that they use persistent previous information to be used in the current neural network. Typical recurrent neural networks can experience a loss in information, often referred to as the vanishing gradient problem. This is caused by the repeated use of the recurrent weight matrix in RNN. In an LSTM model, the recurrent weight matrix is replaced by an identify function in the carousel and controlled by a series of gates. The input gate, output gate and forget gate acts like a switch that controls the weights and creates the long term memory function.
In today’s environment, demand forecasting is complex and the data needed for accurately forecasting at scale isn’t always straightforward. Using LSTM, time series forecasting models can predict future values based on previous, sequential data. This provides greater accuracy for demand forecasters which results in better decision making for the business.
there are several advantages of using LSTM over other types of neural networks, such as traditional feedforward neural networks or simple recurrent neural networks (RNNs):

1.Ability to handle long-term dependencies: One of the biggest advantages of LSTMs over traditional RNNs is their ability to handle long-term dependencies. Traditional RNNs suffer from the vanishing gradient problem, which makes it difficult for them to retain information over long periods of time. LSTMs solve this problem by incorporating a gating mechanism that allows them to selectively retain or forget information.

2.Robustness to noisy input: LSTMs are also more robust to noisy input than traditional RNNs. The gating mechanism in LSTMs allows them to filter out irrelevant information and focus on the most important features of the input.

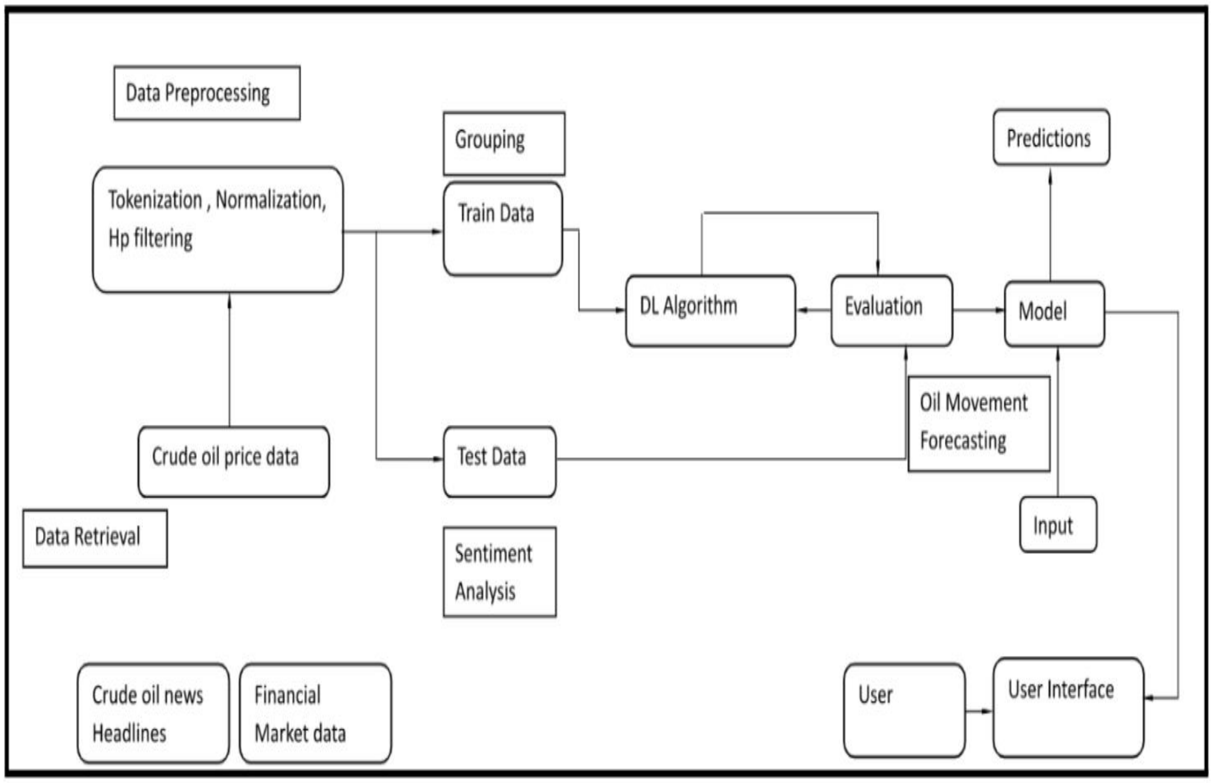
3.Improved training efficiency: LSTMs can be trained more efficiently than traditional RNNs. This is because the gating mechanism in LSTMs allows them to maintain a

constant error flow through time, which makes it easier to train them using backpropagation through time (BPTT).
4.Versatility: LSTMs can be applied to a wide range of tasks, including natural language processing, speech recognition, and time series prediction. They have been shown to achieve state-of-the-art performance on many benchmarks in these fields.

Overall, LSTMs offer several advantages over traditional RNNs, making them a popular choice for many applications in machine learning and artificial intelligence.

3. Theoretical Analysis

3.1 Block Diagram



3.2 Hardware/Software Designing

3.2.1 platform and software application :
Google Colab - Colab is a hosted Jupyter notebook service that requires no setup to use,
while providing access free of charge to computing resources including GPUs.

Visual Studio Codes - Visual Studio Code combines the simplicity of a source code editor
with powerful developer tooling, like IntelliSense code completion and debugging.

3.2.2 Hardware requirements:
Recommended:
- Memory : 8gb
- Graphics Card : AMD Radeon RX 480
- CPU : INTEL CORE I5-4590
- OS : Windows 7

Minimum:
- Memory : 4gb
- Graphics Card : NVIDIA GeForce GTX
- CPU : INTEL CORE I5-4590
- OS : Windows 7

4. Experimental Investigations

Evaluation of Prediction Accuracy: Compute the prediction accuracy of your LSTM model using various evaluation metrics such as mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE). Compare the prediction results of your LSTM model with those of other models (e.g., ARIMA, GARCH, etc.) commonly used for time-series analysis.

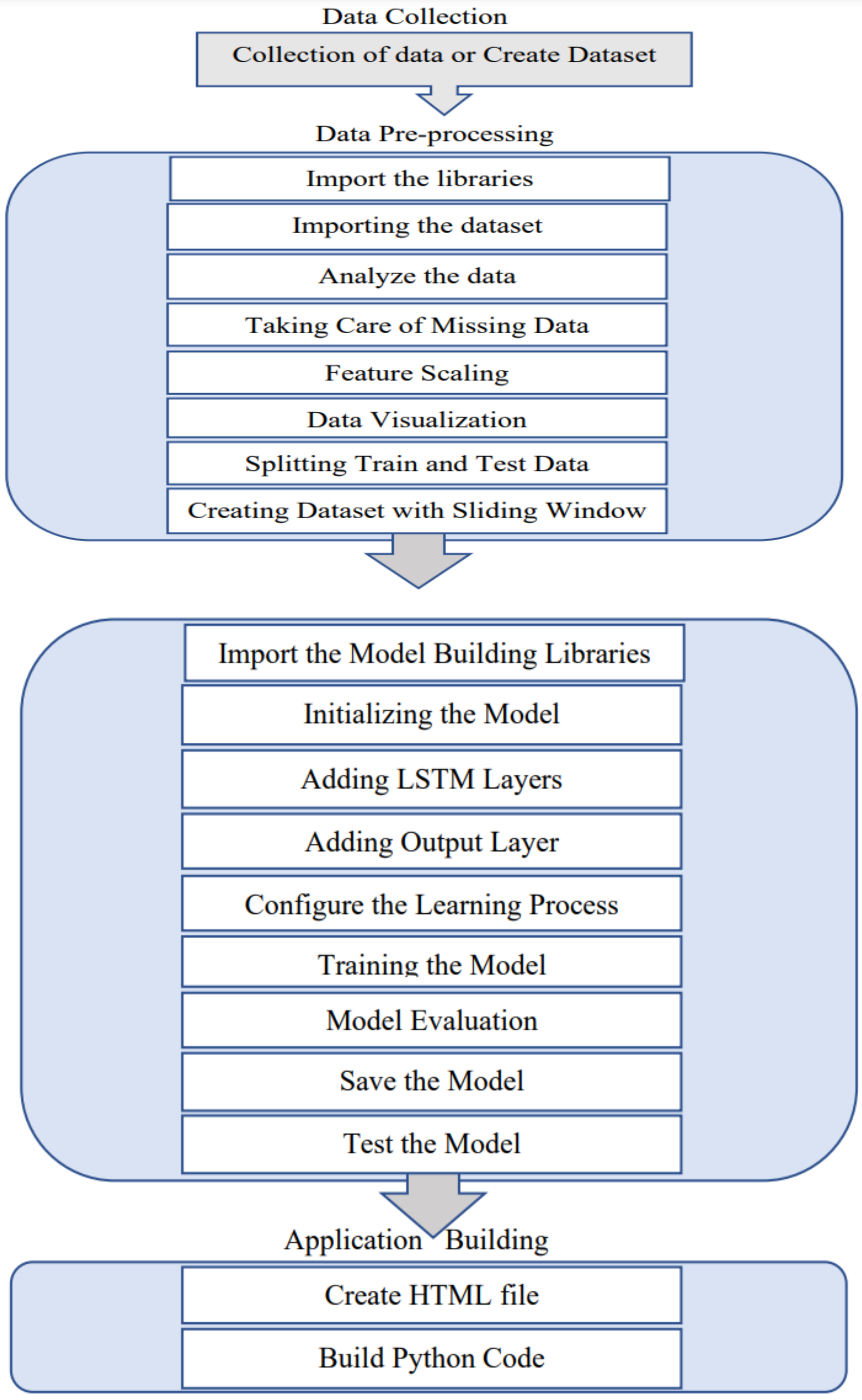
Sensitivity Analysis: Investigate the sensitivity of your LSTM model to different hyperparameters such as the number of hidden layers, number of neurons per layer, batch size, learning rate, and sequence length. Assess how changes in these hyperparameters affect the prediction accuracy of your model.

Feature Importance Analysis: Analyze the importance of different features used as input to your LSTM model. Determine which features have the greatest impact on the prediction accuracy and which ones are less important.

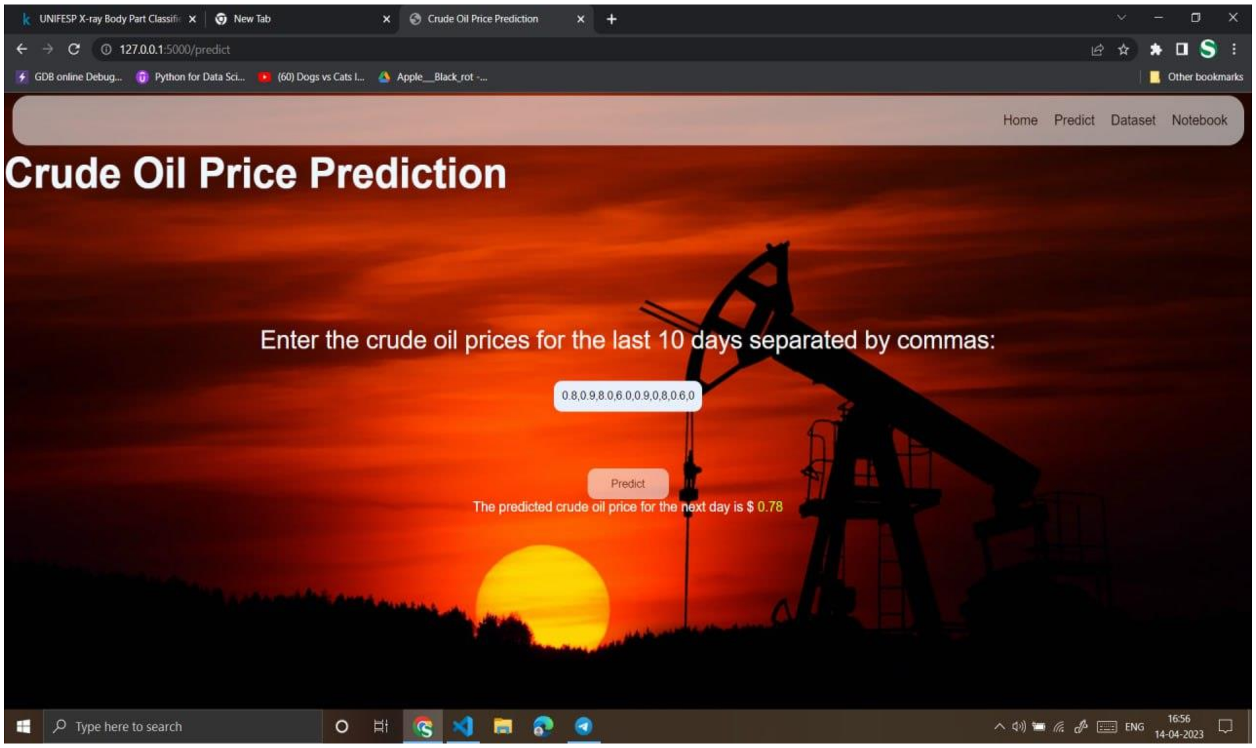
Out-of-Sample Testing: Conduct an out-of-sample testing of your LSTM model to evaluate its performance on data that it has not seen before. Use a hold-out sample of data to validate the accuracy of your model's predictions.

Online Testing: Test the performance of your LSTM model in real-time by feeding it with new data as it becomes available.

5. Flow Chart


Data Collection
Collection of data or Create Dataset
Data Pre-processing
Import the libraries
Importing the dataset
Analyze the data
Taking Care of Missing Data
Feature Scaling
Data Visualization
Splitting Train and Test Data
Creating Dataset with Sliding Window
Import the Model Building Libraries
Initializing the Model
Adding LSTM Layers
Adding Output Layer
Configure the Learning Process
Training the Model
Model Evaluation
Save the Model
Test the Model
Application Building
Create HTML file
Build Python Code

6. Result



Crude Oil Price Prediction

Enter the crude oil prices for the last 10 days separated by commas:

0.8,0.9,8.0,6.0,0.9,0.8,0.6,0

The predicted crude oil price for the next day is $ 0.78

7. Advantages & Disadvantages

Advantages:

Ability to capture long-term dependencies:
Flexibility:
High accuracy:

Disadvantages:

Complex architecture:
Need for large amounts of data:
Computational resources:
Prone to overfitting:

Overall, LSTM has several advantages for crude oil price prediction, including the ability to capture long-term dependencies and high accuracy. However, it also has some disadvantages, such as a complex architecture, the need for large amounts of data and computational resources, and the risk of overfitting.

8. Applications

Investment decision-making: Accurate crude oil price prediction can help investors make informed decisions about buying and selling crude oil futures, stocks, and other related assets.

Risk management: Accurate crude oil price prediction can help energy companies and other stakeholders manage risks associated with price fluctuations, such as hedging against price changes or adjusting production levels.

Trading strategy development: Accurate crude oil price prediction can be used to develop profitable trading strategies for traders and investment firms.

Economic forecasting: Accurate crude oil price prediction can help governments and central banks forecast economic trends and plan policies related to energy and the broader economy.

Energy market analysis: Accurate crude oil price prediction can help energy market analysts and researchers understand the dynamics of energy markets, including supply and demand factors and geopolitical events that influence crude oil prices.

9. Conclusion

In this study, utilizing the data on oil prices for the period spanning February 1986 to May 2021, we employed an LSTM model to forecast two kinds of prices. Besides that, the typical ARIMA and ANN models were selected as the comparison models to evaluate the forecasting accuracy and forecasting stability of the LSTM model. From the empirical results, we can come to the following conclusions.
First, the LSTM model has strong generalization ability. And, the fitting effect on crude oil prices was stable for different timescales. On the one hand, in the short or long term, whether it is the WTI or Brent crude oil price, the fitting effect of the LSTM model performs better than the ANN model and ARIMA model. On the other hand, in the medium term, the fitting effect of the LSTM model is much better than the ARIMA model and slightly weaker than the ANN model.
With the better forecasting performance of the LSTM model, it is beneficial for researchers with a deep understanding of the crude oil market. The results also reveal that the LSTM model is suitable for nonlinear and volatile changes in oil prices. Finally, the accurate predictions by the LSTM model also facilitate a deep understanding of the fluctuation mechanism for crude oil prices, as well as of the production operations and investment of enterprises, which can improve domestic economic development and security. Of course, there may be other better methods for crude oil price prediction, but, due to the limitations of personal energy and ability, other improvements to methods could not be implemented. For example, changing the rolling window of the

ARIMA model may improve the accuracy and validity of the prediction. The LSTM model also has its disadvantages. It is still a cyclic network, so if the input sequence has 1000 characters, the LSTM unit will be called 1000 times, which is a long gradient path. Although adding a long-term memory channel will be helpful, its capacity is still limited. In addition, due to LSTM being recursive in nature, it cannot be trained in parallel.

10. Future Scope

Integration with other models: In the future, LSTM-based models for crude oil price prediction can be integrated with other machine learning models, such as deep neural networks and convolutional neural networks, to improve the accuracy of predictions and capture more complex patterns in the data.

Improved data sources: The accuracy of LSTM-based models for crude oil price prediction can be improved by incorporating additional data sources, such as social media sentiment, satellite imagery, and news articles.

Optimization of hyperparameters: More research can be done to optimize the hyperparameters of LSTM-based models for crude oil price prediction to achieve better prediction performance.

11. Bibillography

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Brownlee, J. (2017). Time series forecasting with long short-term memory networks in Python. Machine Learning Mastery.

Narayan, P. K., & Narayan, S. (2013). Do oil prices predict economic growth? New evidence from a non-linear panel data methodology. Energy Economics, 39, 28-33.

Raza, S. A., Malik, I., & Raza, S. A. (2019). Forecasting crude oil prices with machine learning techniques: A review. Energy Reports, 5, 918-931.

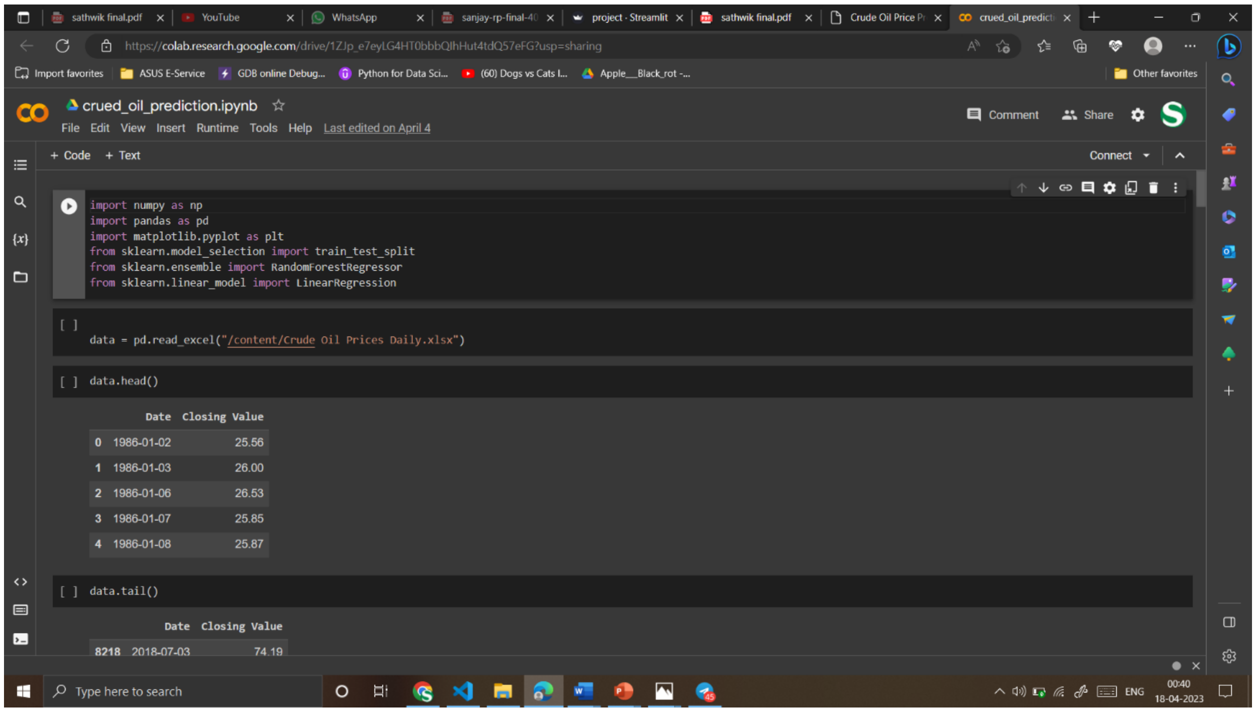
Sharma, A., & Zare, A. (2020). Predicting crude oil prices using deep learning models. Energies, 13(10), 2502.

Zhang, X., & Ma, Y. (2019). Crude oil price forecasting using LSTM neural network based on ensemble empirical mode decomposition. Energies, 12(21), 4131.

Yang, S., Dong, Y., & Zhang, J. (2021). Crude oil price forecasting using hybrid LSTM and long short-term memory neural network. Energies, 14(4), 1024.

11. 1 Appendix

A. SOURCE CODE



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
```

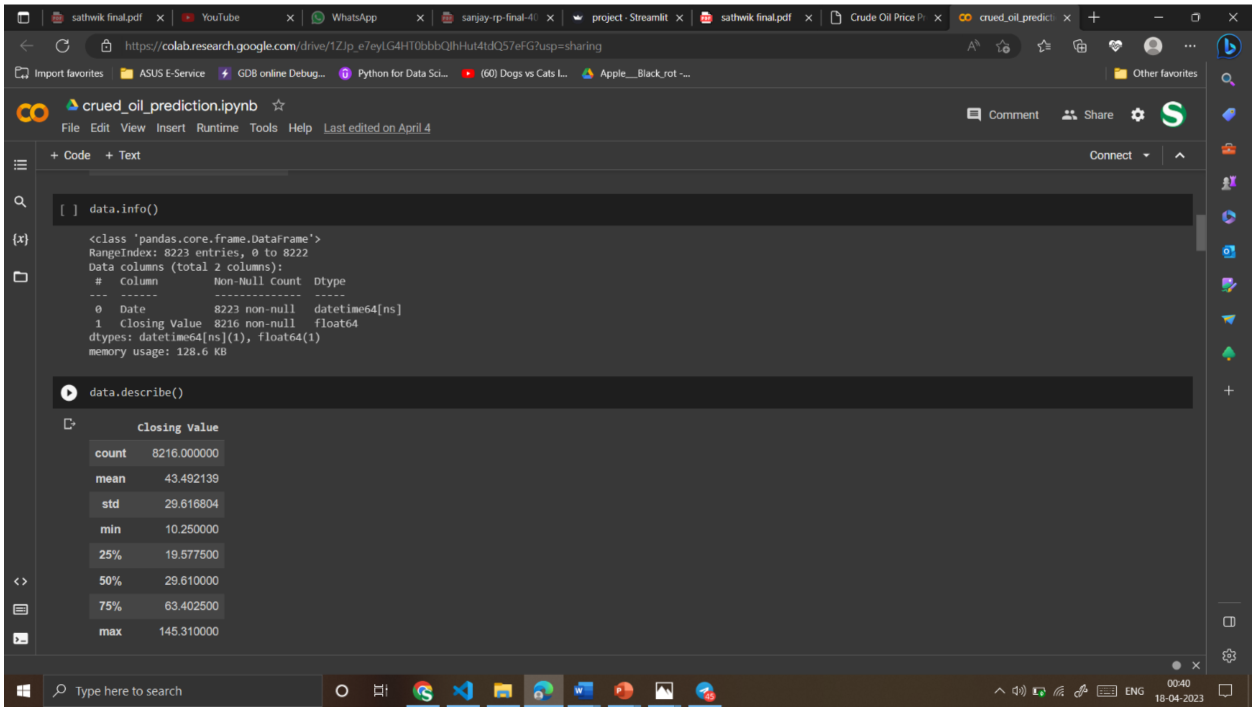
```
data = pd.read_excel("/content/Crude Oil Prices Daily.xlsx")
```

```
data.head()
```

```
         Date  Closing Value
0  1986-01-02          25.56
1  1986-01-03          26.00
2  1986-01-06          26.53
3  1986-01-07          25.85
4  1986-01-08          25.87
```

```
data.tail()
```

```
            Date  Closing Value
8218  2018-07-03          74.19
```

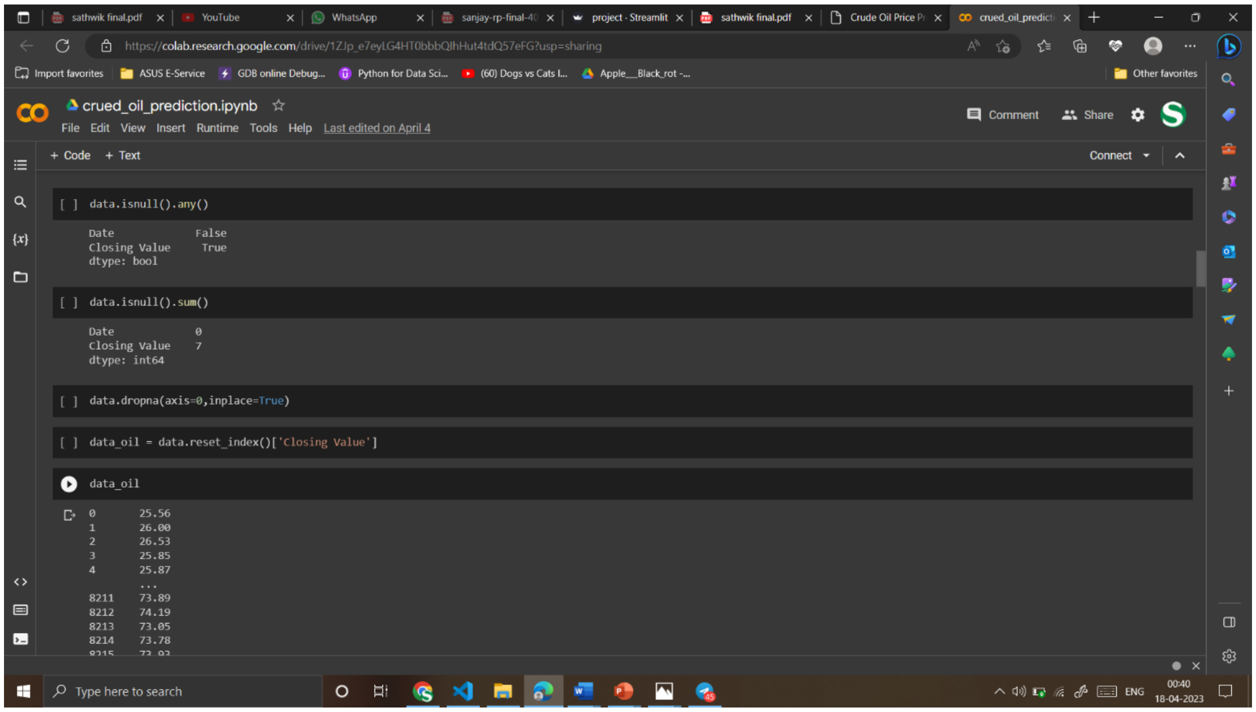


```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8223 entries, 0 to 8222
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Date           8223 non-null   datetime64[ns]
 1   Closing Value  8216 non-null   float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 128.6 KB
```

```
data.describe()
```

```
       Closing Value
count    8216.000000
mean       43.492139
std        29.616804
min        10.250000
25%        19.577500
50%        29.610000
75%        63.402500
max       145.310000
```



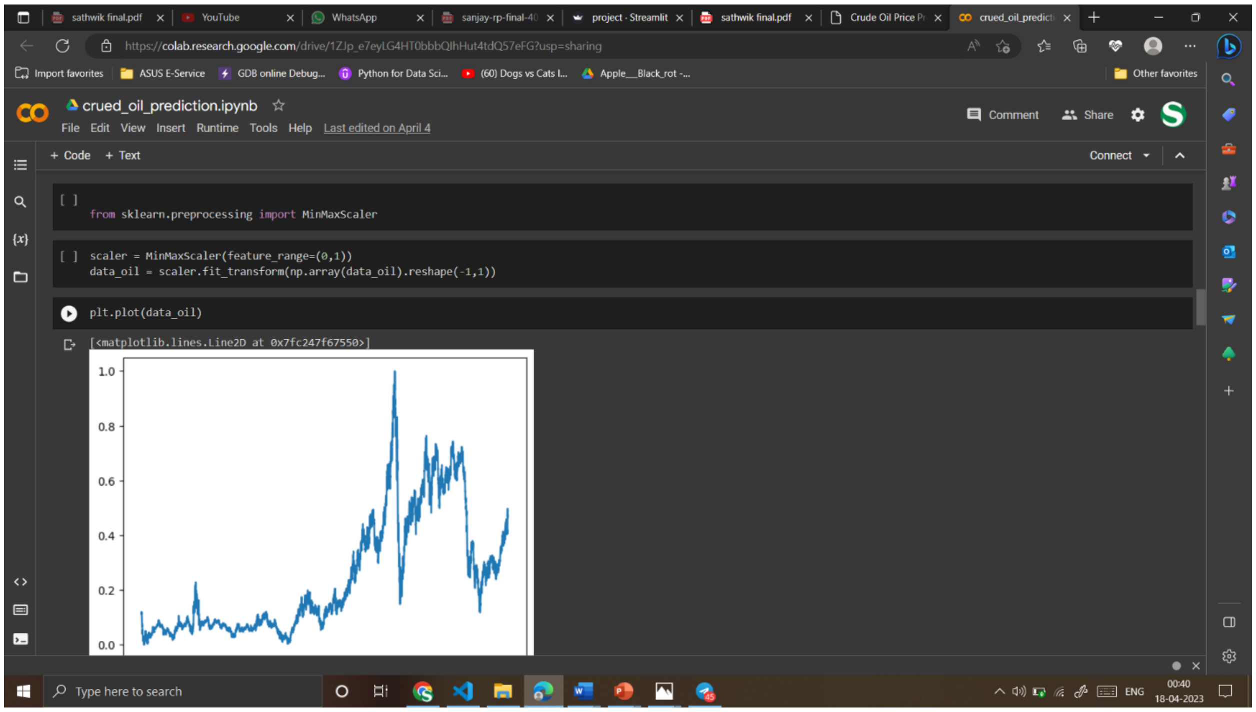
```
data.isnull().any()

Date             False
Closing Value     True
dtype: bool
```

```
data.isnull().sum()

Date             0
Closing Value    7
dtype: int64
```

```
data.dropna(axis=0,inplace=True)
```

```
data_oil = data.reset_index()['Closing Value']
```

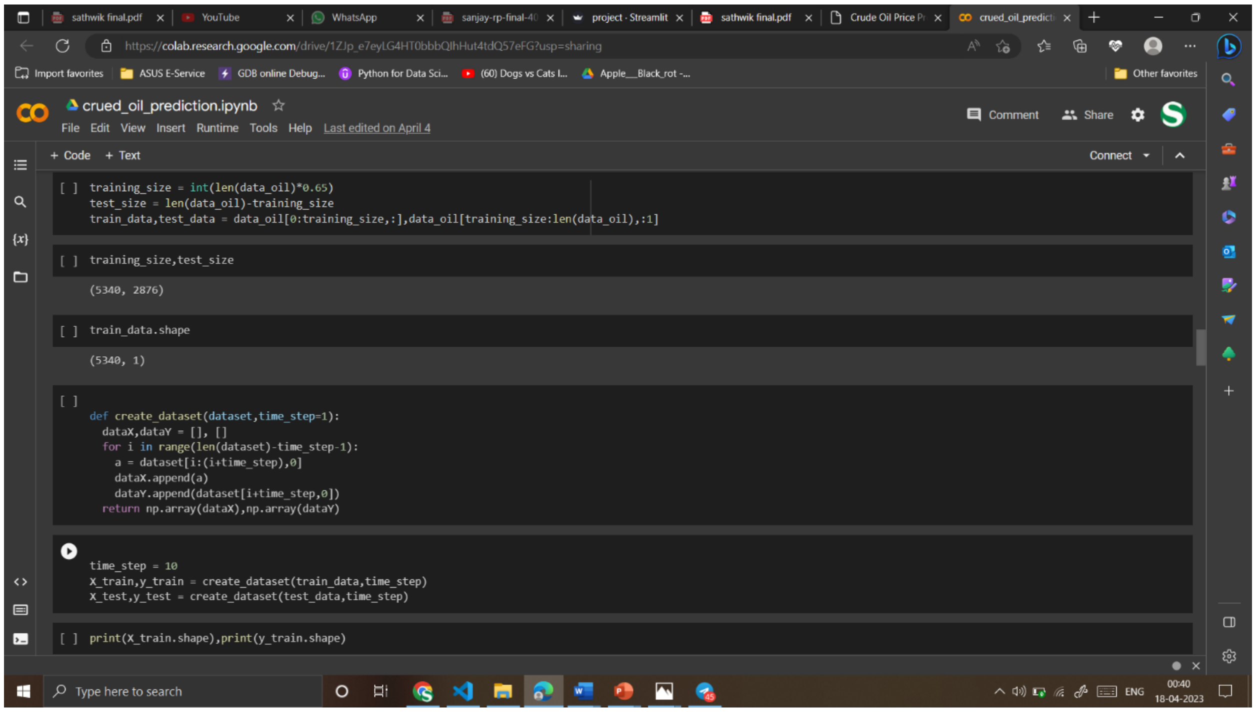
```
data_oil

0       25.56
1       26.00
2       26.53
3       25.85
4       25.87
        ...
8211    73.89
8212    74.19
8213    73.05
8214    73.78
```



```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler = MinMaxScaler(feature_range=(0,1))
data_oil = scaler.fit_transform(np.array(data_oil).reshape(-1,1))
```

```
plt.plot(data_oil)

[<matplotlib.lines.Line2D at 0x7fc247f67550>]
```



```
[ ] training_size = int(len(data_oil)*0.65)
    test_size = len(data_oil)-training_size
    train_data,test_data = data_oil[0:training_size,:],data_oil[training_size:len(data_oil),:1]
```

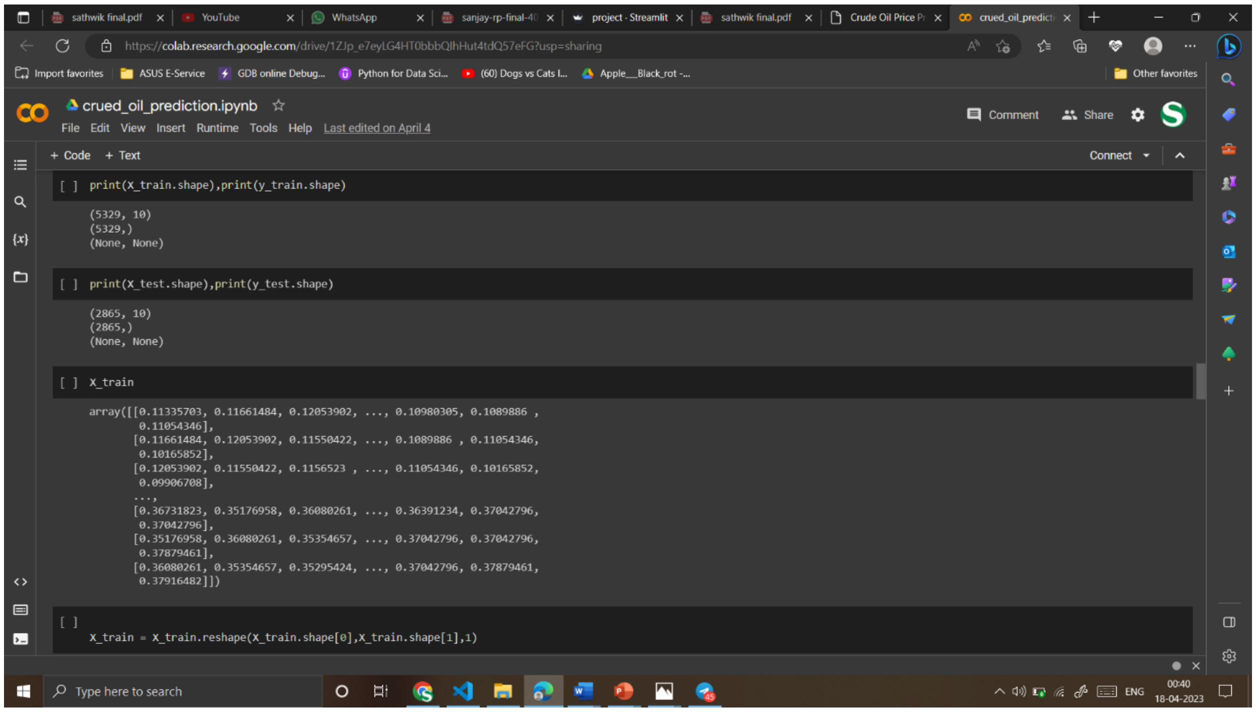
```
[ ] training_size,test_size

    (5340, 2876)
```

```
[ ] train_data.shape

    (5340, 1)
```

```
[ ]
    def create_dataset(dataset,time_step=1):
      dataX,dataY = [], []
      for i in range(len(dataset)-time_step-1):
        a = dataset[i:(i+time_step),0]
        dataX.append(a)
        dataY.append(dataset[i+time_step,0])
      return np.array(dataX),np.array(dataY)
```

```
    time_step = 10
    X_train,y_train = create_dataset(train_data,time_step)
    X_test,y_test = create_dataset(test_data,time_step)
```

```
[ ] print(X_train.shape),print(y_train.shape)
```

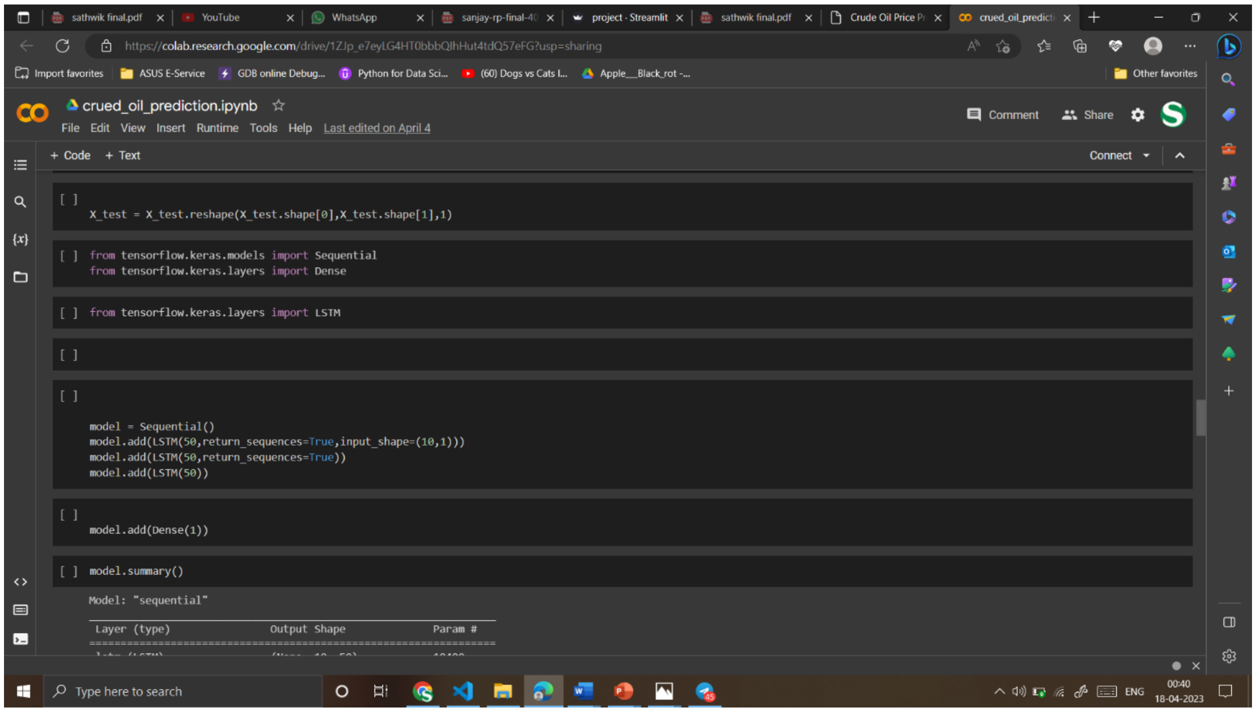


```
[ ] print(X_train.shape),print(y_train.shape)

    (5329, 10)
    (5329,)
    (None, None)
```

```
[ ] print(X_test.shape),print(y_test.shape)

    (2865, 10)
    (2865,)
    (None, None)
```

```
[ ] X_train

    array([[0.11335703, 0.11661484, 0.12053902, ..., 0.10980305, 0.1089886 ,
            0.11054346],
           [0.11661484, 0.12053902, 0.11550422, ..., 0.1089886 , 0.11054346,
            0.10165852],
           [0.12053902, 0.11550422, 0.1156523 , ..., 0.11054346, 0.10165852,
            0.09906708],
           ...,
           [0.36731823, 0.35176958, 0.36080261, ..., 0.36391234, 0.37042796,
            0.37042796],
           [0.35176958, 0.36080261, 0.35354657, ..., 0.37042796, 0.37042796,
            0.37879461],
           [0.36080261, 0.35354657, 0.35295424, ..., 0.37042796, 0.37879461,
            0.37916482]])
```

```
[ ]
    X_train = X_train.reshape(X_train.shape[0],X_train.shape[1],1)
```



```
X_test = X_test.reshape(X_test.shape[0],X_test.shape[1],1)
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

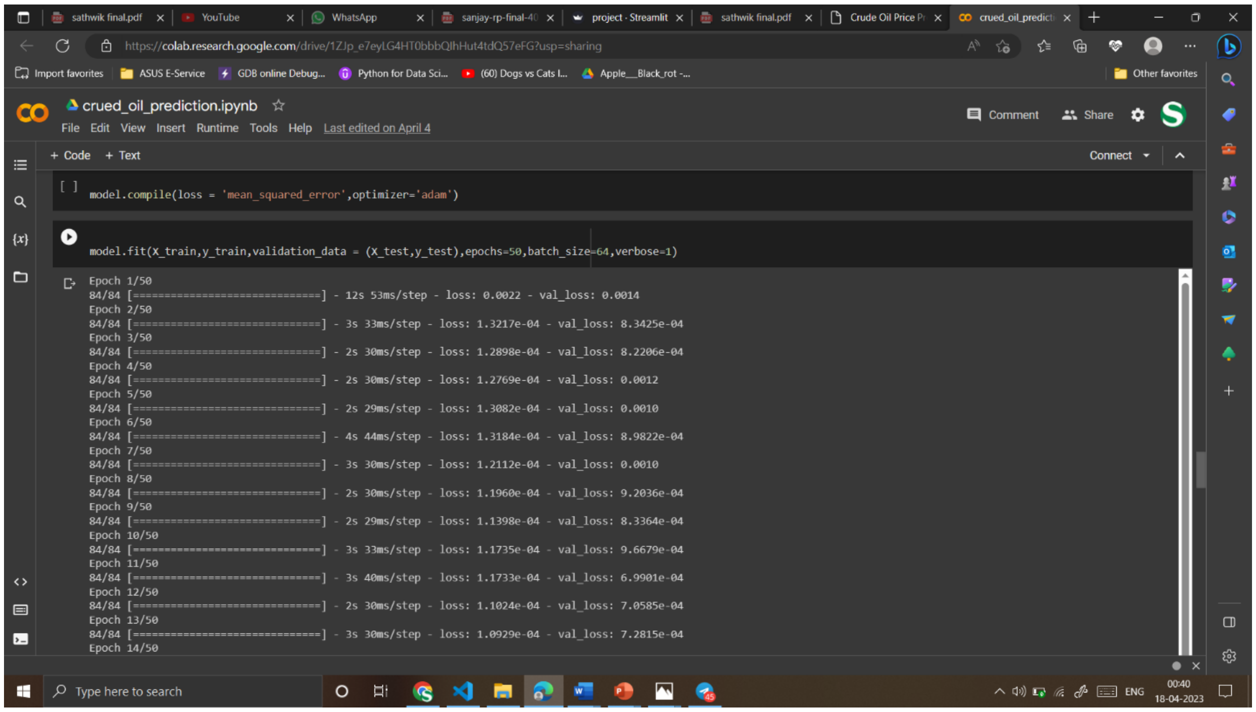
```
from tensorflow.keras.layers import LSTM
```

```
model = Sequential()
model.add(LSTM(50,return_sequences=True,input_shape=(10,1)))
model.add(LSTM(50,return_sequences=True))
model.add(LSTM(50))
```

```
model.add(Dense(1))
```

```
model.summary()
```

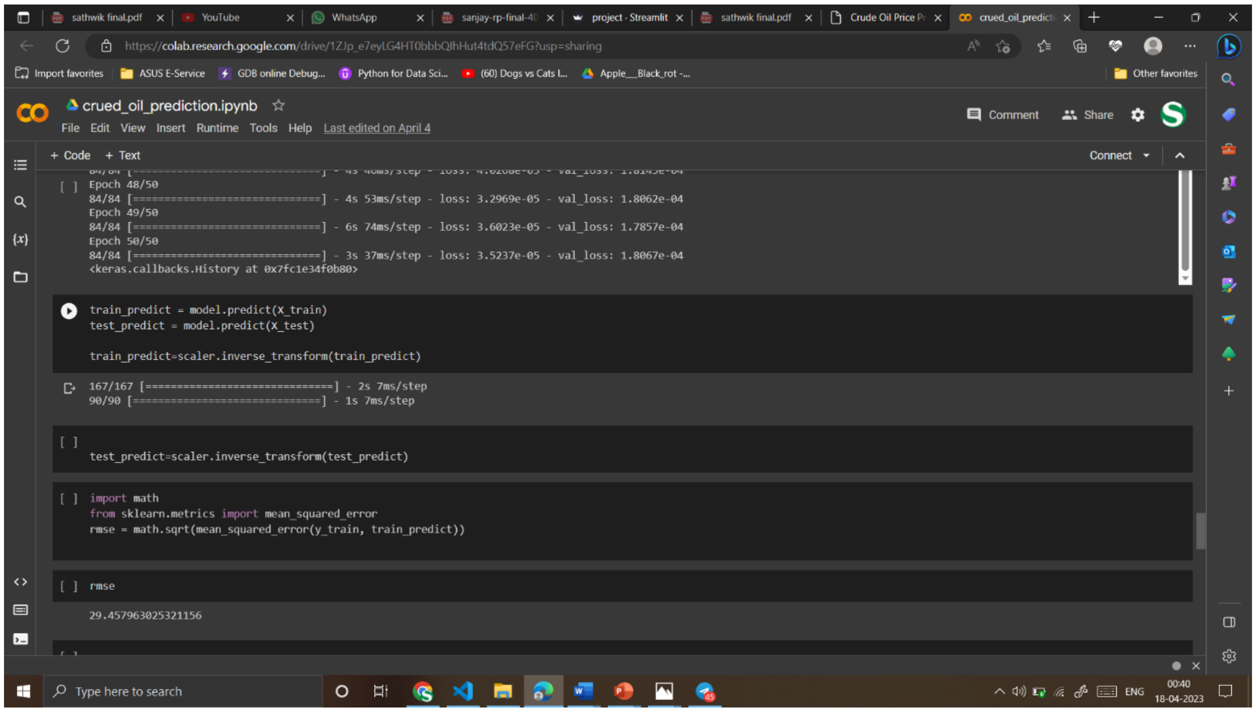
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```



```
model.compile(loss = 'mean_squared_error',optimizer='adam')
```

```
model.fit(X_train,y_train,validation_data = (X_test,y_test),epochs=50,batch_size=64,verbose=1)
```

```
Epoch 1/50
84/84 [==============================] - 12s 53ms/step - loss: 0.0022 - val_loss: 0.0014
Epoch 2/50
84/84 [==============================] - 3s 33ms/step - loss: 1.3217e-04 - val_loss: 8.3425e-04
Epoch 3/50
84/84 [==============================] - 2s 30ms/step - loss: 1.2898e-04 - val_loss: 8.2206e-04
Epoch 4/50
84/84 [==============================] - 2s 30ms/step - loss: 1.2769e-04 - val_loss: 0.0012
Epoch 5/50
84/84 [==============================] - 2s 29ms/step - loss: 1.3082e-04 - val_loss: 0.0010
Epoch 6/50
84/84 [==============================] - 4s 44ms/step - loss: 1.3184e-04 - val_loss: 8.9822e-04
Epoch 7/50
84/84 [==============================] - 3s 30ms/step - loss: 1.2112e-04 - val_loss: 0.0010
Epoch 8/50
84/84 [==============================] - 2s 30ms/step - loss: 1.1960e-04 - val_loss: 9.2036e-04
Epoch 9/50
84/84 [==============================] - 2s 29ms/step - loss: 1.1398e-04 - val_loss: 8.3364e-04
Epoch 10/50
84/84 [==============================] - 3s 33ms/step - loss: 1.1735e-04 - val_loss: 9.6679e-04
Epoch 11/50
84/84 [==============================] - 3s 40ms/step - loss: 1.1733e-04 - val_loss: 6.9901e-04
Epoch 12/50
84/84 [==============================] - 2s 30ms/step - loss: 1.1024e-04 - val_loss: 7.0585e-04
Epoch 13/50
84/84 [==============================] - 3s 30ms/step - loss: 1.0929e-04 - val_loss: 7.2815e-04
Epoch 14/50
```



crued_oil_prediction.ipynb

```
Epoch 48/50
84/84 [==============================] - 4s 53ms/step - loss: 3.2969e-05 - val_loss: 1.8062e-04
Epoch 49/50
84/84 [==============================] - 6s 74ms/step - loss: 3.6023e-05 - val_loss: 1.7857e-04
Epoch 50/50
84/84 [==============================] - 3s 37ms/step - loss: 3.5237e-05 - val_loss: 1.8067e-04
<keras.callbacks.History at 0x7fc1e34f0b80>
```

```
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)

train_predict=scaler.inverse_transform(train_predict)
```

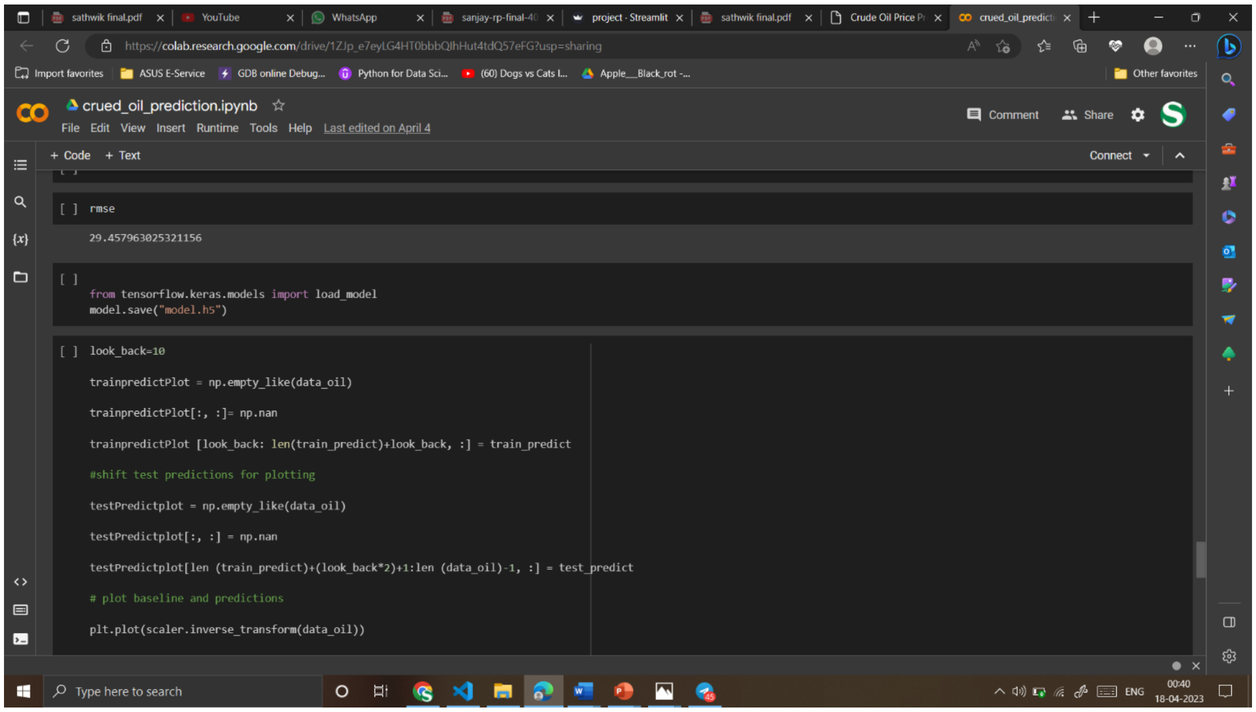
```
167/167 [==============================] - 2s 7ms/step
90/90 [==============================] - 1s 7ms/step
```

```
test_predict=scaler.inverse_transform(test_predict)
```

```
import math
from sklearn.metrics import mean_squared_error
rmse = math.sqrt(mean_squared_error(y_train, train_predict))
```

```
rmse
```

```
29.457963025321156
```



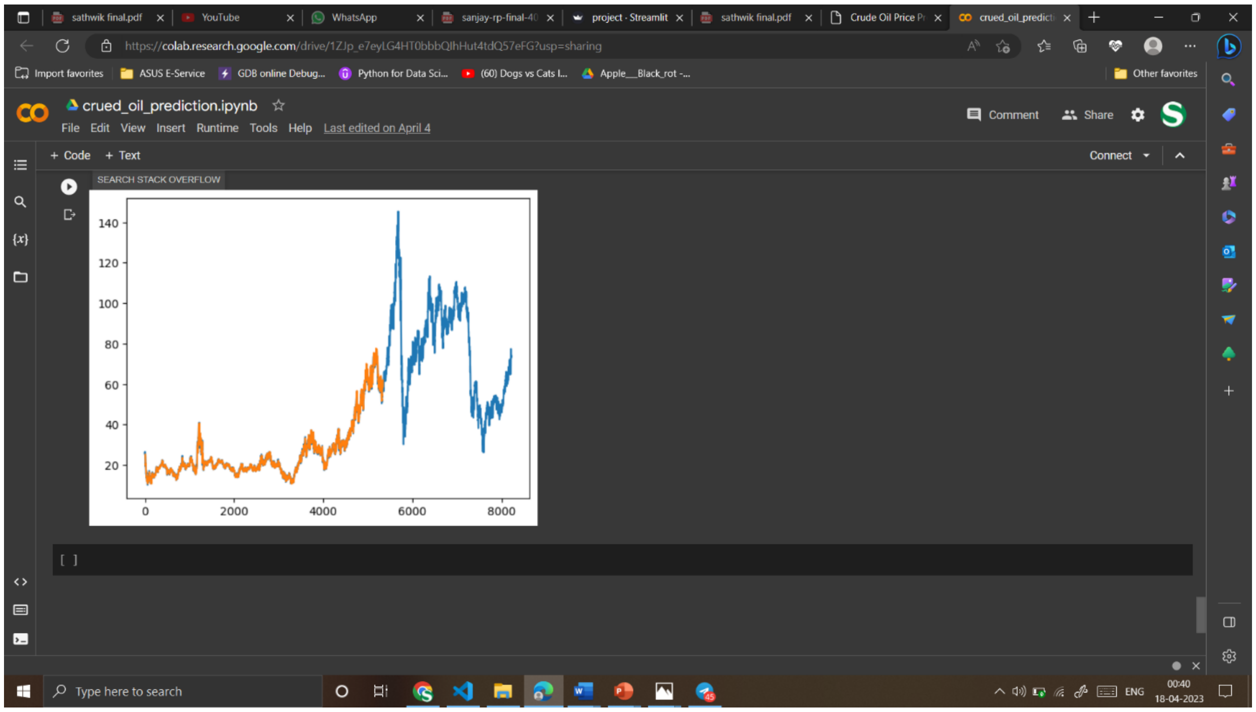
crued_oil_prediction.ipynb

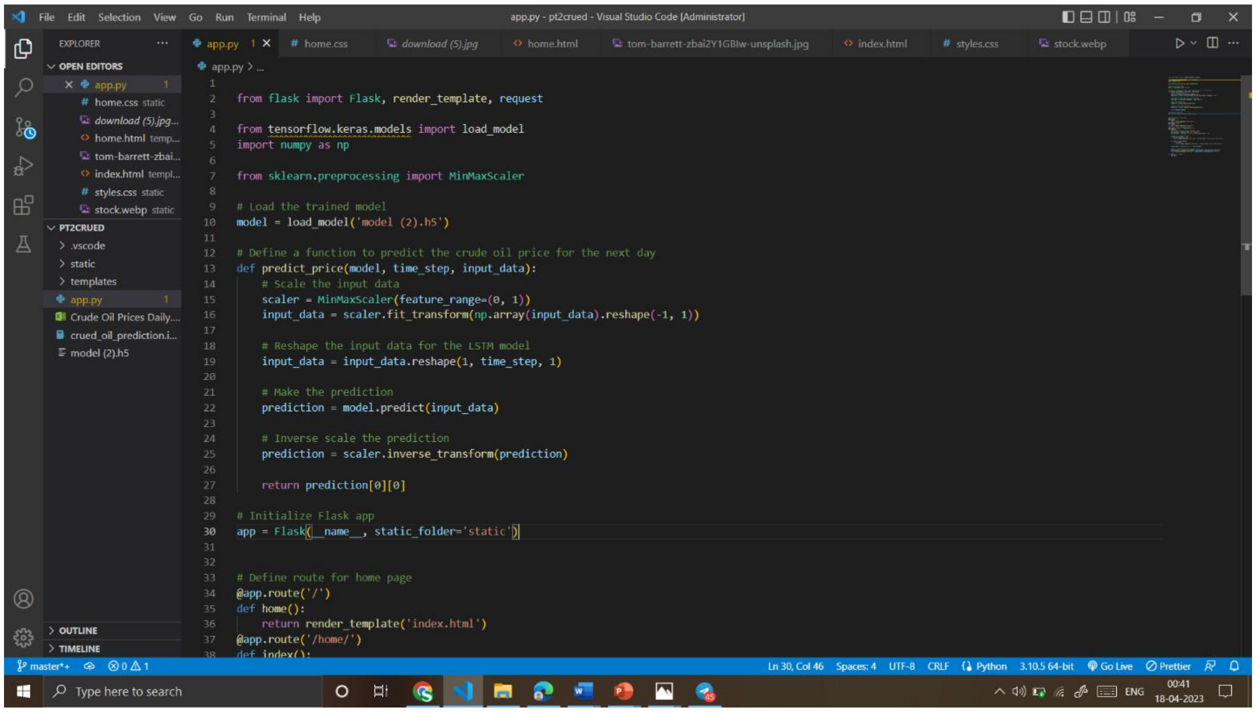
```
rmse
```

```
29.457963025321156
```

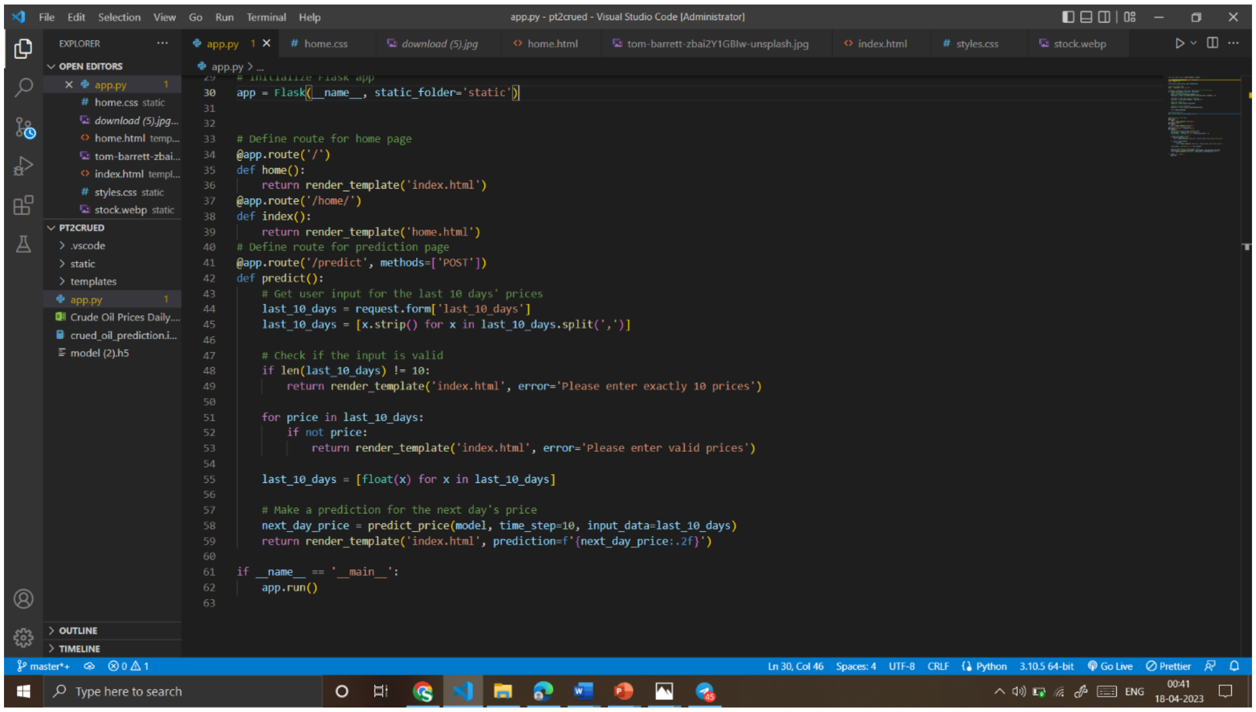
```
from tensorflow.keras.models import load_model
model.save("model.h5")
```

```
look_back=10

trainpredictPlot = np.empty_like(data_oil)

trainpredictPlot[:, :]= np.nan

trainpredictPlot [look_back: len(train_predict)+look_back, :] = train_predict

#shift test predictions for plotting

testPredictplot = np.empty_like(data_oil)

testPredictplot[:, :] = np.nan

testPredictplot[len (train_predict)+(look_back*2)+1:len (data_oil)-1, :] = test_predict

# plot baseline and predictions

plt.plot(scaler.inverse_transform(data_oil))
```

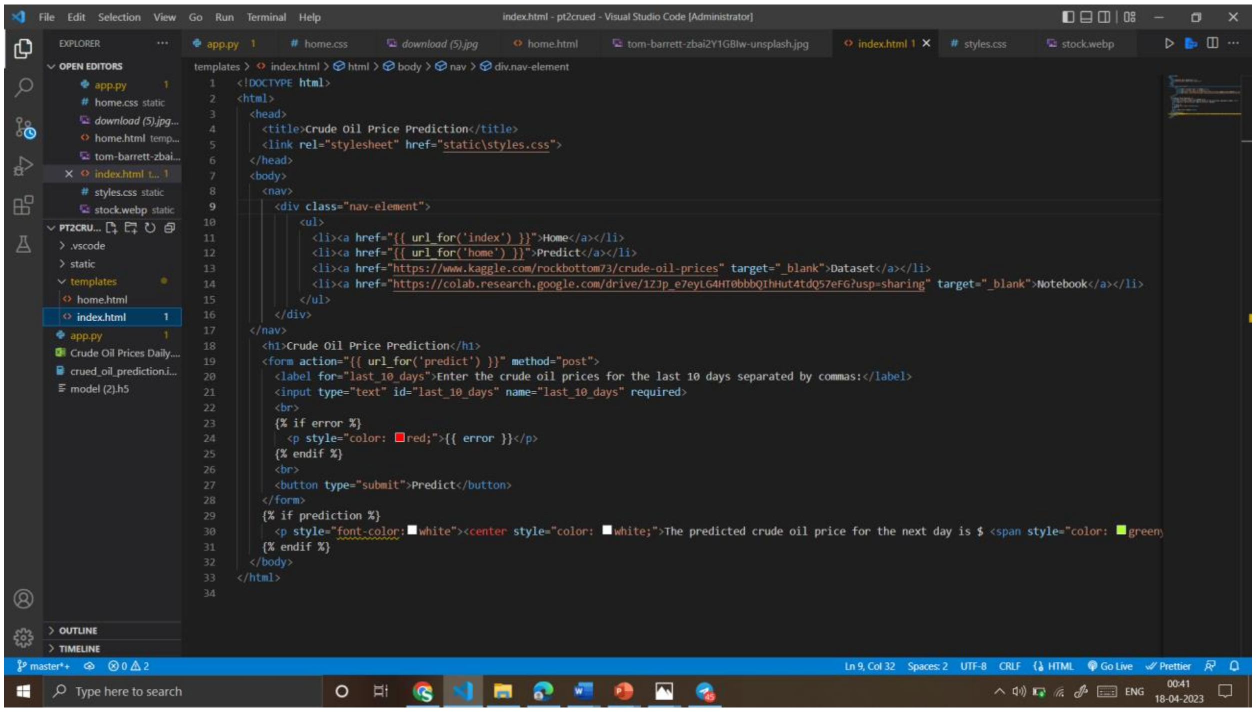

crued_oil_prediction.ipynb
File Edit View Insert Runtime Tools Help Last edited on April 4
+ Code + Text
SEARCH STACK OVERFLOW
140
120
100
80
60
40
20
0
2000
4000
6000
8000



```python

from flask import Flask, render_template, request

from tensorflow.keras.models import load_model
import numpy as np

from sklearn.preprocessing import MinMaxScaler

# Load the trained model
model = load_model('model (2).h5')

# Define a function to predict the crude oil price for the next day
def predict_price(model, time_step, input_data):
    # Scale the input data
    scaler = MinMaxScaler(feature_range=(0, 1))
    input_data = scaler.fit_transform(np.array(input_data).reshape(-1, 1))

    # Reshape the input data for the LSTM model
    input_data = input_data.reshape(1, time_step, 1)

    # Make the prediction
    prediction = model.predict(input_data)

    # Inverse scale the prediction
    prediction = scaler.inverse_transform(prediction)

    return prediction[0][0]

# Initialize Flask app
app = Flask(__name__, static_folder='static')


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



```python
# Initialize Flask app
app = Flask(__name__, static_folder='static')


# Define route for home page
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/home/')
def index():
    return render_template('home.html')
# Define route for prediction page
@app.route('/predict', methods=['POST'])
def predict():
    # Get user input for the last 10 days' prices
    last_10_days = request.form['last_10_days']
    last_10_days = [x.strip() for x in last_10_days.split(',')]

    # Check if the input is valid
    if len(last_10_days) != 10:
        return render_template('index.html', error='Please enter exactly 10 prices')

    for price in last_10_days:
        if not price:
            return render_template('index.html', error='Please enter valid prices')

    last_10_days = [float(x) for x in last_10_days]

    # Make a prediction for the next day's price
    next_day_price = predict_price(model, time_step=10, input_data=last_10_days)
    return render_template('index.html', prediction=f'{next_day_price:.2f}')

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



```html
<!DOCTYPE html>
<html>
  <head>
    <title>Crude Oil Price Prediction</title>
    <link rel="stylesheet" href="static\styles.css">
  </head>
  <body>
    <nav>
      <div class="nav-element">
          <ul>
            <li><a href="{{ url_for('index') }}">Home</a></li>
            <li><a href="{{ url_for('home') }}">Predict</a></li>
            <li><a href="https://www.kaggle.com/rockbottom73/crude-oil-prices" target="_blank">Dataset</a></li>
            <li><a href="https://colab.research.google.com/drive/1ZJp_e7eyLG4HT0bbbQIhHut4tdQ57eFG?usp=sharing" target="_blank">Notebook</a></li>
          </ul>
      </div>
  </nav>
    <h1>Crude Oil Price Prediction</h1>
    <form action="{{ url_for('predict') }}" method="post">
      <label for="last_10_days">Enter the crude oil prices for the last 10 days separated by commas:</label>
      <input type="text" id="last_10_days" name="last_10_days" required>
      <br>
      {% if error %}
        <p style="color: red;">{{ error }}</p>
      {% endif %}
      <br>
      <button type="submit">Predict</button>
    </form>
    {% if prediction %}
      <p style="font-color:white"><center style="color: white;">The predicted crude oil price for the next day is $ <span style="color: green
    {% endif %}
  </body>
</html>
```