Statistical Learning Lab

Assignment - 8

Recurrent Neural Network for Stock Prediction

Name: Semanti Ghosh

Roll No.: 22IM10036

1. Choose a stock of your choice from NIFTY 50 list from Yahoo Finance.
2. Take last 5 years stock price data of the selected stock.
3. Create test dataset for past 3 months, and training set from 5 years to the date before 3 months.
4. Use a predictive model using 3 LSTM layers, with past 60 days data, ntimestep = 60, dropout regularization ndrop = 0.2.
5. Create the plots comparing observed value of the test data and the predictive value.
6. Use grid search to optimize hyperparameters such as ndrop , ntimestep and batch size. Compare test result with previous findings.

Give the result and the code in a single pdf document along with plots.

In [ ]:
```
pip install --upgrade tensorflow
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.11/dist-packa
ges (2.18.0)
Collecting tensorflow
  Downloading tensorflow-2.19.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x
86_64.whl.metadata (4.1 kB)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-p
ackages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.11/dis
t-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.11/
dist-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/
lib/python3.11/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.11/d
ist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.11/dist
-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.11/dis
t-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packag
es (from tensorflow) (24.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.2
1.4,!=4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/lib/python3.11/dist-packages (from
tensorflow) (5.29.4)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/d
ist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packa
ges (from tensorflow) (75.2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-pack
ages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist
-packages (from tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python
3.11/dist-packages (from tensorflow) (4.13.0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-pa
ckages (from tensorflow) (1.17.2)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/d
ist-packages (from tensorflow) (1.71.0)
Collecting tensorboard~=2.19.0 (from tensorflow)
  Downloading tensorboard-2.19.0-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.11/dist-pac
kages (from tensorflow) (3.8.0)
Requirement already satisfied: numpy<2.2.0,>=1.26.0 in /usr/local/lib/python3.11/
dist-packages (from tensorflow) (2.0.2)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.11/dist-pac
kages (from tensorflow) (3.13.0)
Collecting ml-dtypes<1.0.0,>=0.5.1 (from tensorflow)
  Downloading ml_dtypes-0.5.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86
_64.whl.metadata (21 kB)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/loca
l/lib/python3.11/dist-packages (from tensorflow) (0.37.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.11/di
st-packages (from astunparse>=1.6.0->tensorflow) (0.45.1)
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (f
rom keras>=3.5.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages
(from keras>=3.5.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages
(from keras>=3.5.0->tensorflow) (0.14.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python
```

```
3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (2025.1.31)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.11/dist-packages (from tensorboard~=2.19.0->tensorflow) (3.7)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from tensorboard~=2.19.0->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from tensorboard~=2.19.0->tensorflow) (3.1.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.11/dist-packages (from werkzeug>=1.0.1->tensorboard~=2.19.0->tensorflow) (3.0.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorflow) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorflow) (2.18.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich->keras>=3.5.0->tensorflow) (0.1.2)
Downloading tensorflow-2.19.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (644.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 644.9/644.9 MB 2.9 MB/s eta 0:00:00
Downloading ml_dtypes-0.5.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.7/4.7 MB 73.3 MB/s eta 0:00:00
Downloading tensorboard-2.19.0-py3-none-any.whl (5.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.5/5.5 MB 97.3 MB/s eta 0:00:00
Installing collected packages: ml-dtypes, tensorboard, tensorflow
  Attempting uninstall: ml-dtypes
    Found existing installation: ml-dtypes 0.4.1
    Uninstalling ml-dtypes-0.4.1:
      Successfully uninstalled ml-dtypes-0.4.1
  Attempting uninstall: tensorboard
    Found existing installation: tensorboard 2.18.0
    Uninstalling tensorboard-2.18.0:
      Successfully uninstalled tensorboard-2.18.0
  Attempting uninstall: tensorflow
    Found existing installation: tensorflow 2.18.0
    Uninstalling tensorflow-2.18.0:
      Successfully uninstalled tensorflow-2.18.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
tf-keras 2.18.0 requires tensorflow<2.19,>=2.18, but you have tensorflow 2.19.0 which is incompatible.
tensorflow-text 2.18.1 requires tensorflow<2.19,>=2.18.0, but you have tensorflow 2.19.0 which is incompatible.
Successfully installed ml-dtypes-0.5.1 tensorboard-2.19.0 tensorflow-2.19.0
```

In [26]:

```python
import yfinance as yf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.optimizers import Adam
```

```
#from keras.models import Sequential
#from keras.layers import LSTM, Dense, Dropout
#from keras.optimizers import Adam
from sklearn.model_selection import ParameterGrid
```

Chosen Stock is **TCS** on Nifty (NSE)

1. Download from Yahoo Finance
2. Get the Train & Test Data

In [32]:

```
# Section 1: Fetch Data and Create Train-Test Split
# Define the stock ticker for TCS on NSE
ticker = "TCS.NS"

# Define end date (today)
end_date = datetime.today().strftime('%Y-%m-%d')

# Fetch more than 5 years to ensure we get full 90 trading days in test set
start_date = (datetime.today() - timedelta(days=5*365 + 30)).strftime('%Y-%m-%d'

# Download stock data
df = yf.download(ticker, start=start_date, end=end_date)

# Ensure data was fetched
if df.empty:
    print(f"Failed to fetch data for {ticker}. Please check the ticker symbol or
else:
    print(f"Total data points: {len(df)}")

# Adjust test_start_date to ensure 90 **trading days**
test_dates = df.index[-90:]  # Last 90 actual trading days
test_start_date = test_dates[0]  # First date of test set

# Split into train and test sets
train_data = df.loc[:test_start_date, ['Close']].copy()
test_data = df.loc[test_start_date:, ['Close']].copy()

print(f"Training Data Size: {train_data.shape}")
print(f"Testing Data Size: {test_data.shape}")
```

```
[*********************100%***********************]  1 of 1 completed
Total data points: 1258
Training Data Size: (1169, 1)
Testing Data Size: (90, 1)
```

1. Perform scaling using MinMaxScalar
2. Put n_steps to 30 so that we can get at least 60 test data, from the given ntimestep = 60
3. The first n_steps (30 days) are used as input for the first prediction. This process continues, sliding forward by 1 day each time. As a result, the last n_steps samples cannot be used as independent targets, reducing the effective dataset size by n_steps.

In [33]:

```
from sklearn.preprocessing import MinMaxScaler

# Normalize price data
```

```
scaler = MinMaxScaler(feature_range=(0, 1))
train_scaled = scaler.fit_transform(train_data[['Close']])
test_scaled = scaler.transform(test_data[['Close']])

# Function to create sequences for LSTM
def create_sequences_with_dates(data, dates, n_steps):
    X, y, y_dates = [], [], []
    for i in range(len(data) - n_steps):
        X.append(data[i:i + n_steps])
        y.append(data[i + n_steps])
        y_dates.append(dates[i + n_steps])  # Capture correct date
    return np.array(X), np.array(y), np.array(y_dates, dtype="datetime64[D]")  #

n_steps = 30  # Reduce from 60 to 30 to get enough test samples

# Create sequences for LSTM
X_train, y_train, _ = create_sequences_with_dates(train_scaled, train_data.index
X_test, y_test, test_dates = create_sequences_with_dates(test_scaled, test_data.

print(f"Final Training Samples: {X_train.shape[0]}")
print(f"Final Test Samples: {X_test.shape[0]}")  # Should be ~60 test points
```

```
Final Training Samples: 1139
Final Test Samples: 60
```

Create the predictive model using 3 LSTM layers

In [34]:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.optimizers import Adam

def build_model(n_steps, ndrop):
    model = Sequential([
        LSTM(50, return_sequences=True, input_shape=(n_steps, 1)),
        Dropout(ndrop),
        LSTM(50, return_sequences=True),
        Dropout(ndrop),
        LSTM(50),
        Dropout(ndrop),
        Dense(1)
    ])
    model.compile(optimizer=Adam(learning_rate=0.001), loss='mean_squared_error'
    return model

model = build_model(n_steps, 0.2)

# Reshape X_train for LSTM input format
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)

# Train the model
model.fit(X_train, y_train, epochs=50, batch_size=32, verbose=1)
```

```
Epoch 1/50
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarn
ing: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Seq
uential models, prefer using an `Input(shape)` object as the first layer in the m
odel instead.
  super().__init__(**kwargs)
```

```
36/36 ━━━━━━━━━━━━━━━━━━━━ 7s 41ms/step - loss: 0.0789
Epoch 2/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 42ms/step - loss: 0.0081
Epoch 3/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 40ms/step - loss: 0.0051
Epoch 4/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 47ms/step - loss: 0.0055
Epoch 5/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 50ms/step - loss: 0.0048
Epoch 6/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 40ms/step - loss: 0.0052
Epoch 7/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 41ms/step - loss: 0.0045
Epoch 8/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 42ms/step - loss: 0.0042
Epoch 9/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - loss: 0.0046
Epoch 10/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 47ms/step - loss: 0.0038
Epoch 11/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 66ms/step - loss: 0.0044
Epoch 12/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 40ms/step - loss: 0.0042
Epoch 13/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 40ms/step - loss: 0.0040
Epoch 14/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - loss: 0.0034
Epoch 15/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 41ms/step - loss: 0.0048
Epoch 16/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - loss: 0.0041
Epoch 17/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 48ms/step - loss: 0.0032
Epoch 18/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 46ms/step - loss: 0.0036
Epoch 19/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 42ms/step - loss: 0.0035
Epoch 20/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 40ms/step - loss: 0.0029
Epoch 21/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 41ms/step - loss: 0.0039
Epoch 22/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - loss: 0.0027
Epoch 23/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 70ms/step - loss: 0.0033
Epoch 24/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - loss: 0.0033
Epoch 25/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - loss: 0.0027
Epoch 26/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 42ms/step - loss: 0.0028
Epoch 27/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 40ms/step - loss: 0.0032
Epoch 28/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 41ms/step - loss: 0.0026
Epoch 29/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 64ms/step - loss: 0.0031
Epoch 30/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 42ms/step - loss: 0.0031
Epoch 31/50
```

```
36/36 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - loss: 0.0028
Epoch 32/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 41ms/step - loss: 0.0027
Epoch 33/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - loss: 0.0028
Epoch 34/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 42ms/step - loss: 0.0027
Epoch 35/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 60ms/step - loss: 0.0029
Epoch 36/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 41ms/step - loss: 0.0028
Epoch 37/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 41ms/step - loss: 0.0024
Epoch 38/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 41ms/step - loss: 0.0027
Epoch 39/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 41ms/step - loss: 0.0022
Epoch 40/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 52ms/step - loss: 0.0023
Epoch 41/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 60ms/step - loss: 0.0026
Epoch 42/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 41ms/step - loss: 0.0024
Epoch 43/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - loss: 0.0021
Epoch 44/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 41ms/step - loss: 0.0023
Epoch 45/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 41ms/step - loss: 0.0022
Epoch 46/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 4s 70ms/step - loss: 0.0022
Epoch 47/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 4s 40ms/step - loss: 0.0028
Epoch 48/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 41ms/step - loss: 0.0023
Epoch 49/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 3s 41ms/step - loss: 0.0023
Epoch 50/50
36/36 ━━━━━━━━━━━━━━━━━━━━ 2s 42ms/step - loss: 0.0022
```

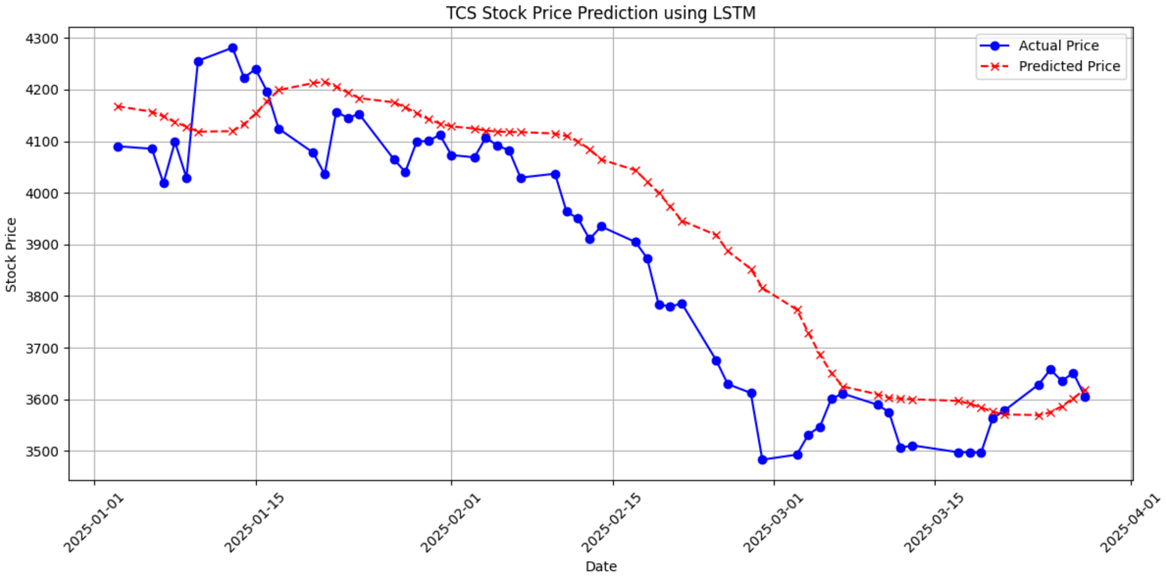
```
Out[34]:  <keras.src.callbacks.history.History at 0x7e25fad631d0>
```

Plots comparing observed value of the test data and the predictive value.

```
In [35]:  # Section 4: Predict and Plot Results

import matplotlib.pyplot as plt
import numpy as np

# Make predictions
predicted_prices = model.predict(X_test)

# Convert back to original scale
predicted_prices = scaler.inverse_transform(predicted_prices.reshape(-1, 1)).fla

# Ensure test_dates are correctly formatted
test_dates = np.array(test_dates, dtype='datetime64[D]')  # Ensures daily timest

# Adjust index to align predictions with actual test data
actual_prices = test_data.iloc[n_steps:]['Close'].values  # Extract correspondin
```

```python
# Plot last 90 days of test data
plt.figure(figsize=(14, 6))
plt.plot(test_dates[-90:], actual_prices[-90:], label="Actual Price", color='blu
plt.plot(test_dates[-90:], predicted_prices[-90:], label="Predicted Price", colc
plt.title("TCS Stock Price Prediction using LSTM")
plt.xlabel("Date")
plt.ylabel("Stock Price")
plt.xticks(rotation=45)  # Rotate for better visibility
plt.legend()
plt.grid()
plt.show()
```

```
1/2 ━━━━━━━━━━━━━━━━━━━━ 0s 433ms/step
WARNING:tensorflow:5 out of the last 8 calls to <function TensorFlowTrainer.make_
predict_function.<locals>.one_step_on_data_distributed at 0x7e25fa60db20> trigger
ed tf.function retracing. Tracing is expensive and the excessive number of tracin
gs could be due to (1) creating @tf.function repeatedly in a loop, (2) passing te
nsors with different shapes, (3) passing Python objects instead of tensors. For
(1), please define your @tf.function outside of the loop. For (2), @tf.function h
as reduce_retracing=True option that can avoid unnecessary retracing. For (3), pl
ease refer to https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 450ms/step
```



Use grid search to optimize hyperparameters such as ndrop , ntimestep and batch size.

In [36]:
```python
from sklearn.model_selection import ParameterGrid
import pandas as pd

# Define hyperparameter search space
param_grid = {
    'ndrop': [0.2, 0.3],         # Dropout rates
    'batch_size': [16, 32],       # Batch sizes
    'n_steps': [30, 60]           # Lookback window
}

best_loss = float('inf')
best_params = None
results = []  # Store results for analysis
```

```
for params in ParameterGrid(param_grid):
    print(f"Training with params: {params}")

    # Create sequences
    X_train, y_train, _ = create_sequences_with_dates(train_scaled, train_data.i
    X_test, y_test, test_dates = create_sequences_with_dates(test_scaled, test_d

    # Build and train model
    model = build_model(params['n_steps'], params['ndrop'])
    history = model.fit(X_train, y_train, epochs=50, batch_size=params['batch_si

    # Evaluate model on test set
    loss = model.evaluate(X_test, y_test, verbose=0)

    # Store results
    results.append({'ndrop': params['ndrop'], 'batch_size': params['batch_size']

    # Track best parameters
    if loss < best_loss:
        best_loss = loss
        best_params = params

# Convert results to DataFrame and display
results_df = pd.DataFrame(results).sort_values(by="loss")
print("Hyperparameter Search Results:")
print(results_df)

print(f"\nBest Parameters: {best_params} with Loss: {best_loss:.4f}")
```

```
Training with params: {'batch_size': 16, 'n_steps': 30, 'ndrop': 0.2}
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarn
ing: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Seq
uential models, prefer using an `Input(shape)` object as the first layer in the m
odel instead.
  super().__init__(**kwargs)
```

```
Training with params: {'batch_size': 16, 'n_steps': 30, 'ndrop': 0.3}
Training with params: {'batch_size': 16, 'n_steps': 60, 'ndrop': 0.2}
Training with params: {'batch_size': 16, 'n_steps': 60, 'ndrop': 0.3}
Training with params: {'batch_size': 32, 'n_steps': 30, 'ndrop': 0.2}
Training with params: {'batch_size': 32, 'n_steps': 30, 'ndrop': 0.3}
Training with params: {'batch_size': 32, 'n_steps': 60, 'ndrop': 0.2}
Training with params: {'batch_size': 32, 'n_steps': 60, 'ndrop': 0.3}
Hyperparameter Search Results:
   ndrop  batch_size  n_steps      loss
3    0.3          16       60  0.000723
1    0.3          16       30  0.000843
2    0.2          16       60  0.000918
6    0.2          32       60  0.000965
0    0.2          16       30  0.001517
4    0.2          32       30  0.001531
7    0.3          32       60  0.002155
5    0.3          32       30  0.002760

Best Parameters: {'batch_size': 16, 'n_steps': 60, 'ndrop': 0.3} with Loss: 0.000
7
```

Train with best found model and plot

In [37]:

```
# --- Train the Final Model with Best Parameters ---
best_n_steps = best_params['n_steps']
```

```
best_ndrop = best_params['ndrop']
best_batch_size = best_params['batch_size']

X_train, y_train, _ = create_sequences_with_dates(train_scaled, train_data.index
X_test, y_test, test_dates = create_sequences_with_dates(test_scaled, test_data.

# Build and train optimized LSTM model
best_model = build_model(best_n_steps, best_ndrop)
best_model.fit(X_train, y_train, epochs=50, batch_size=best_batch_size, verbose=

# Make predictions
predicted_prices = best_model.predict(X_test)
predicted_prices = scaler.inverse_transform(predicted_prices.reshape(-1, 1))  #

# Convert test_dates to correct format
test_dates = np.array(test_dates, dtype='datetime64[D]')

# --- Plot the Actual vs Predicted Prices ---
plt.figure(figsize=(14, 6))
plt.plot(test_dates, test_data.iloc[best_n_steps:]['Close'], label="Actual Price
plt.plot(test_dates, predicted_prices, label="Predicted Price", color='red')
plt.title("TCS Stock Price Prediction using Optimized LSTM")
plt.xlabel("Date")
plt.ylabel("Stock Price")
plt.xticks(rotation=45)  # Rotate for better visibility
plt.legend()
plt.show()
```

```
Epoch 1/50
```

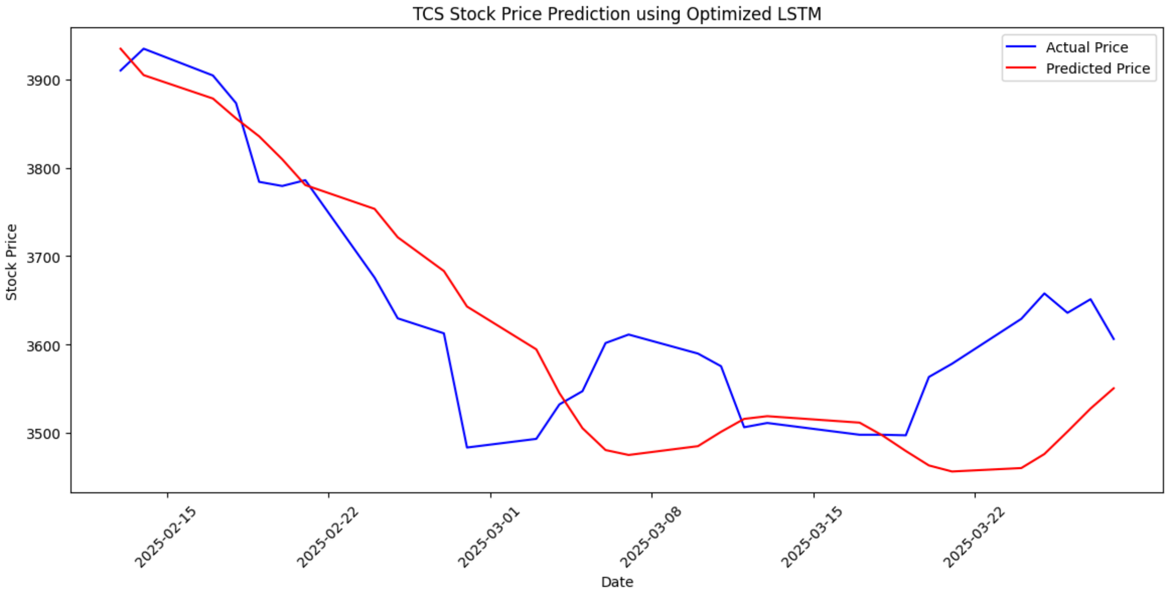
```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/rnn/rnn.py:200: UserWarn
ing: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Seq
uential models, prefer using an `Input(shape)` object as the first layer in the m
odel instead.
  super().__init__(**kwargs)
```

```
70/70 ━━━━━━━━━━━━━━━━━━━━ 11s 68ms/step - loss: 0.0717
Epoch 2/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 6s 82ms/step - loss: 0.0078
Epoch 3/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 10s 82ms/step - loss: 0.0077
Epoch 4/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 9s 68ms/step - loss: 0.0093
Epoch 5/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 6s 82ms/step - loss: 0.0066
Epoch 6/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 10s 82ms/step - loss: 0.0061
Epoch 7/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 5s 69ms/step - loss: 0.0058
Epoch 8/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 5s 67ms/step - loss: 0.0056
Epoch 9/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 6s 79ms/step - loss: 0.0048
Epoch 10/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 10s 82ms/step - loss: 0.0055
Epoch 11/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 5s 68ms/step - loss: 0.0047
Epoch 12/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 5s 77ms/step - loss: 0.0048
Epoch 13/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 10s 69ms/step - loss: 0.0050
Epoch 14/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 6s 82ms/step - loss: 0.0040
Epoch 15/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 10s 83ms/step - loss: 0.0037
Epoch 16/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 9s 68ms/step - loss: 0.0040
Epoch 17/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 6s 83ms/step - loss: 0.0046
Epoch 18/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 5s 68ms/step - loss: 0.0034
Epoch 19/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 6s 83ms/step - loss: 0.0038
Epoch 20/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 9s 69ms/step - loss: 0.0035
Epoch 21/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 6s 79ms/step - loss: 0.0033
Epoch 22/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 10s 83ms/step - loss: 0.0034
Epoch 23/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 10s 77ms/step - loss: 0.0030
Epoch 24/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 5s 75ms/step - loss: 0.0032
Epoch 25/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 11s 84ms/step - loss: 0.0032
Epoch 26/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 10s 82ms/step - loss: 0.0027
Epoch 27/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 9s 68ms/step - loss: 0.0027
Epoch 28/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 6s 83ms/step - loss: 0.0027
Epoch 29/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 10s 82ms/step - loss: 0.0026
Epoch 30/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 9s 69ms/step - loss: 0.0026
Epoch 31/50
```

```
70/70 ━━━━━━━━━━━━━━━━━━━━ 6s 84ms/step - loss: 0.0024
Epoch 32/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 5s 69ms/step - loss: 0.0022
Epoch 33/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 6s 83ms/step - loss: 0.0022
Epoch 34/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 9s 68ms/step - loss: 0.0023
Epoch 35/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 6s 81ms/step - loss: 0.0023
Epoch 36/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 10s 83ms/step - loss: 0.0021
Epoch 37/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 9s 69ms/step - loss: 0.0018
Epoch 38/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 6s 78ms/step - loss: 0.0019
Epoch 39/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 10s 82ms/step - loss: 0.0019
Epoch 40/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 5s 69ms/step - loss: 0.0018
Epoch 41/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 5s 76ms/step - loss: 0.0020
Epoch 42/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 5s 74ms/step - loss: 0.0019
Epoch 43/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 11s 83ms/step - loss: 0.0017
Epoch 44/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 5s 69ms/step - loss: 0.0017
Epoch 45/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 6s 82ms/step - loss: 0.0018
Epoch 46/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 9s 68ms/step - loss: 0.0018
Epoch 47/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 6s 82ms/step - loss: 0.0017
Epoch 48/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 5s 69ms/step - loss: 0.0015
Epoch 49/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 6s 83ms/step - loss: 0.0015
Epoch 50/50
70/70 ━━━━━━━━━━━━━━━━━━━━ 9s 71ms/step - loss: 0.0017
```

```
WARNING:tensorflow:6 out of the last 9 calls to <function TensorFlowTrainer.make_
predict_function.<locals>.one_step_on_data_distributed at 0x7e25ecce4b80> trigger
ed tf.function retracing. Tracing is expensive and the excessive number of tracin
gs could be due to (1) creating @tf.function repeatedly in a loop, (2) passing te
nsors with different shapes, (3) passing Python objects instead of tensors. For
(1), please define your @tf.function outside of the loop. For (2), @tf.function h
as reduce_retracing=True option that can avoid unnecessary retracing. For (3), pl
ease refer to https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 655ms/step
```


TCS Stock Price Prediction using Optimized LSTM
Actual Price
Predicted Price
Stock Price
3900
3800
3700
3600
3500
Date
2025-02-15
2025-02-22
2025-03-01
2025-03-08
2025-03-15
2025-03-22